RとSTANで学ぶフリーで楽しい心理統計の世界

心理学データ解析
応用

データの背後のメカニズムを解析する方法

小杉考司

心理学データ解析応用

小杉　考司

Last Compiled on 2024.1.23

はじめに

昨今はデータサイエンス，情報科学の領域が非常に隆盛で，コンピュータを使ってデータを分析し，経済の動向や購買行動などの予測に用いられることが広く行われている。

人の行動や考え方をどのようにデータにするかについては，当然ながら心理学には一日の長がある。また，人が頭の中でどのような考え方のプロセスをたどるのか，それをどのように検証するのかについても，心理学はその短い歴史の中で徹底的にその技法を洗練させてきた。このような根源的なレベルでの理論や方法論は時代が変わっても色褪せることなく，また今後ますます必要とされてくる時代になっている。

本講ではデータ解析の応用段階として，より実践的なテーマを扱う。すなわち，**心理尺度が作られる理論的背景**と，**データの背後のメカニズムを解析する方法**を知ることである。

心理学研究法の 1 つとして，調査研究がある。紙とペンで回答を集めた時，回答者がある反応カテゴリにまるをつけたことが，どうして数値処理の対象になるのか。そこには数字を割り振るルールとしての「尺度化」の手続きがある。残念ながら応用的側面が発展しすぎたため，回答に数字を割り振る原理について語られることが少なくなってきてしまい，それに対する反動からか，近年改めてこの根本原理についての理解と解説が求められている。この講義では，尺度化の原理や目に見えない潜在変数を想定して分析するとはどういうことかについて，理論と演習を交えながら習得することを目指す。この理論的側面を考えるためには，どうしても線形代数・行列計算の知識が必要になってくる。線形代数については特別な事前知識は不要で，定義から改めて解説するので安心してもらいたい。

心理学研究法のまた 1 つの大きな柱として，実験的研究がある。実験的研究はその手続きが厳格に準備されていることで，結果は群間の平均値差を推定すれば十分である，とされてきた。心理学の基礎領域では，そのため，標本の平均値から母集団の平均値を推測する方法が主なテーマになる。しかしこの方法は逆に，結果を群間の平均値差に帰着させるための実験計画を必要とする。結果に方法が規定されているのである。本来考えたかったことは群間の差だけではなく，どのようにして心理的なメカニズムからデータが生成されているか，という問いであったことを思い出そう。そして群間の平均値に拘ることなく，心理的なメカニズムを数式的に表現することで分析する**数理モデリング**というアプローチがある。このモデリングの基礎的な知識，方法論の習得を目指す。

授業のテーマ

データから意味のある情報を取り出すための，さまざまな分析法を習熟するにあたって，その背後にあって語られることのない「発想」の観点から理解する。数値だけに振り回される状態から脱却し，数値を算出する数式に込められた意味について考える視点を持つ。またこれらに習熟することで，どのような研究対象に対してどのような心理統計的アプローチができるかを，俯瞰的に見れるようになろう。

一年を通じて伝えたいポイント

尺度化とは何か　心理学で行われるアンケート調査やその後の分析はどういう原理があって「心を測定した」といえるのか。その原理やモデルを理解して利用できるようになる。

多変量解析から何がわかるのか　調査研究などで得られた多変量を分析することで何がわかるのか。あるいは何をしてわかったというのか。

多変量解析の基礎となる数式的原理　多変量解析の背景にあるのは線形代数という数学であり，線形代数の基礎を学ぶことで多変量解析のメカニズムを統合的に理解できる。

データ生成メカニズム　データから情報を取り出す受け身的な分析ではなく，データに数字を与えたり，データが生まれてくるメカニズムをリバースエンジニアリングすることで，さらに積極的にデータ解析に立ち向かおう。

統計環境 R と確率的プログラミング言語 Stan による実践　統計環境 R と確率的プログラミング言語 Stan に習熟することで実際に計算し，確認しながら分析を進めることができる。

その他

授業シラバスとこの講義資料を掲載したサイト (`https://kosugitti.github.io/psychometrtics_syllabus/`) で，最新版のシラバスと授業資料，授業で用いるサンプルデータやコードの配布を行なっています。

目次

第 I 部

心理学データ解析応用 1

第 1 章

導入；多変量データと心理学

この授業は基礎的な心理統計を修めた人向けの，応用コースになっています。基礎的な心理統計，という言葉に私が込めた意味は，

- 記述統計；得られたデータの統計量を算出したり，可視化することによってデータの特徴を把握する。
- 推測統計；得られたデータが母集団からの標本であると考え，標本の特徴を使って母数を推測する。さらに標本の特徴から母集団について何らかの判断 (意思決定) を行う。

という 2 点です。とくに推測統計学の領域では，確率の話やさまざまな推定法，それに伴う技術などが必要ですから，これだけでも膨大な量だったのではと思います。こうした基礎的な知識や技術を身につけると，とりあえず手元のデータを使って差があるとかないとか，どの程度効果があったのか，と言った基本的な判断はできると思います。複雑な計算式のところは機械 (統計環境 R など) がやってくれますので，出た結果だけをみて判断すれば良いのです[*1]。

しかし基礎的なところだけで満足していると，「はて，何がしたかったのかな」とそもそもの問題意識を忘れてしまうことにもなりかねません。とりあえず教わった (膨大な！) プロセスを経て実験結果を見てみれば OK なんでしょう，というだけでは表面的な理解に止まっていると言わざるを得ません。さまざまなシーンに適用される方法論，その計算は何を意味していて，元々の数式にはどのような意味があり，式から得られるものは何をどこまで指し示しているのか，というところまで考えるのが，この講義の狙いです。数式は数式に過ぎない，というのはその通りなのですが，その数式にどのような意味があるのか，数式から得られる結果にどのような意味があるのか，を理解した上で数値 (データ) を扱うようになることが目的です。数値だけに振り回される状態からの脱却，が狙いなわけです。

この授業の前半 (前期) は，講義の形をとります。テーマとしては，**因子分析 (Factor Analysis)** を扱います。因子分析は，いわゆる質問紙調査を行った後で適用される多変量解析法の一種で，たくさんの質問項目の背後にある「因子」を見つけ出します。その因子には XXX 特性，XXX 傾向といった名前がつけられ，その上で心理学的に考察することが一般的です。因子分析の結果として性格特性が得られる，などといったりするわけですから，これはもう心理学の王道中の王道，と言えるかもしれません。しかし実際は統計ソフトウェアが計算してくれて，何だかわからないまま使っている，という人も少なくありません。質問紙調査で分析して何かがわかる，というのはどういうことなのか，原理的なところから解説を始めていきます。またこの講義ではその基礎的なところ，数式レベルでしっかりと把握した上で，数値例に進みます。講義が終わる頃には，意味内容をしっかり理解したうえで因子分析を使えるようになっていることが期待されます。

この授業の後半 (後期) は，実践・演習を主にした学習になります。我々が手にしたデータは，何らかのモデ

[*1] もちろん結果の見方が間違っていたり，拙速だったりしてはいけないなど，注意すべきところは多々あります。

ル･仮定のもとで生成されたものである，という考え方に立ち，**データから生成メカニズムを推測する**ことにチャレンジします。これは非常に夢のある話です。だってデータ生成メカニズムというのは，私たちの心の中の機序そのものだからです。推測に当たってはさまざまな前提･仮定が必要ですが，得られる結果は非常に含蓄に富み，さまざまな角度から心を考えさせてくれるヒントになります。線形モデルは直線的な関係でしたが，より柔軟で豊富な表現力を持つ技術へと理解を進めていきましょう。

今回は初回ですので，基礎的な心理統計で学んだことを改めて確認･復習することを中心に，今後の講義でも用いられる数学的記号の準備をします。

1.1 正規線形モデルの世界

因子分析法や (重) 回帰分析では，基本的に扱う変数が複数あります。これまでの相関･回帰分析や，実験計画の中では，説明変数と被説明変数がひとつずつ，という二変数の世界でした。これが多くなったシーンは，一般に**多変量解析 (Multivariate Analysis)** と呼ばれます。変量あるいは**変数 (Variables)** は，ケースごとに変わる数のことを指します。変わる「数」と言っていますが，この後述べるように数字以外のものも数字として扱いますので，「ケースごとに変わるもの」の総称だと思ってください。多変量はそれが多くあるもの，データセット全体を指します。**たくさんのデータセットの中から意味ある情報を引きだすこと**，これが多変量解析の目的です。

イメージとしては，表計算ソフトのスプレッドシートに数字がずらっと並んでいる世界です。たとえば性格検査の尺度の一種である，YG 性格検査は被験者に 120 の項目について回答させます。ひとりにつき 120 の変数があり，これを何百，何千人に対して実施するのですから，非常に大量のデータになっているわけです。スプレッドシートにデータを入れていくときは一般に，一行 (横方向) に 1 ケース (1 オブザベーション，1 個人) であるようにし，列方向 (縦方向) に変数を並べるのが一般的です。数学記号では次のように表現します。

$$x_{ij}$$

ここで i は個人を，j は変数を指します。たとえば x_{13} で第 1 番目の個人 (ケース) の第 3 番目の変数 (についての値) を意味するわけです。このように一般化することで，120 ある変数でも何千分ものデータでも 1 つの記号で表現できますね。

さて，このような大量のデータを今から分析していくわけですが，どのような方法があるでしょうか。データ分析の領域には，さまざまなモデル，手法があります。数え方にもよりますが，何百という種類の分析法があるかもしれません。もちろん似通ったものもありますし，同じ目的に使う異なる手法もあります。これを大きく分けるなら，まず「線形モデル」と「非線形モデル」に分割できます。

■線形モデル 線形モデルは，回帰分析や要因計画などが含まれます。関係が直線的であること，つまり変数に 2 乗，3 乗の項が入っていないので，同じパターンで先々まで考えることができる，単純な関係です。線形というのは $y = ax + b$ のグラフを書くとわかるように，直線的な関係になることからきています。変数が x, y だけでなく,$y = ax + bz$ のように 2 つあったとしても，グラフでは線が面になるだけで，ある断面で見ると直線関係であることに変わりはありません。実際の多変量データではたくさんの項が含まれ，関数全体を可視化することは不可能ですが，次元が多くなっても直線関係であることに変わりはありません。一般に線形モデルは次の形で表現されます。

$$y = a_1x_1 + a_2x_2 + \cdots + a_nx_n + b$$

ここで a_m は第 m 番目の変数 x_m につく**係数 (coefficients)** であり，変数の重要性を示す数字です。足し合わせる時に，変数の重要性を変えるものなので**重み (weight)** と呼ばれることもあります。この**重みつき線型結合**が線形モデルの基本です。

線形モデルの代表的な分析方法としては，**回帰分析**，**重回帰分析**，**パス解析 (Path Analysis)**，階層回帰分析，**因子分析**，**主成分分析 (Principle Component Analysis)**，共分散分析，判別分析などが含まれます。また今あげたモデルをすべて含んだ表現形式である**構造方程式モデリング (Structural Equation Modeling)** があります[*2]。構造方程式モデリングは総合的な表現方法で，先にあげたモデルを下位モデルとして含むものですから，これをしっかり学べば各手法をいちいち学ぶ必要がない，とも言える究極的なモデルです。本講義では第 12 講で触れることになります。

線形モデルの中には他にも**グラフィカルモデリング** (宮川, 1997) などが含まれますが，それは本講の範囲を超えるので専門書に譲ります[*3]。さらに言えばこれら線形モデルのほとんどは**正規分布 (Gaussian Curve)** を仮定した確率モデル群だと言えます。合わせて正規線形モデルといいます。

■非線形モデル　(正規) 線形モデルは，変数間関係を直線的なものだと考えるのでした。しかし，世の中のことは必ずしも線形関係ではありません。むしろ線形でない関係の方が一般的でしょう。線形関係というのは $A \to B$ のように，「こうすれば，こうなる」というわかりやすい関係ですが，人間の場合はとくに「叩いたら，泣く」「優しい言葉をかければ，喜ばれる」といったことでも成立しないことがいっぱいあるわけです。叩かれた人が，強がって見せるためにグッと我慢するとか，優しい言葉に絆されて泣いてしまうといった人間の感情の機微は，本当に興味深く複雑な仕組みですよね。

線形モデルは，現状に当てはまらないこともありますが，わかりやすさを優先して作られたモデルです。それに対して，現状に当てはまることを目的にすると，とても線形の関係では無理です。そこで非線形な関係でもいいから，データに適したモデルはないか，と考えられているのがこの非線形モデルです。非線形だからといって，曲線である，というだけではありません。たとえば条件分岐のように，枝分かれしていくような関係なども含まれます。

非線形モデルの代表的な分析方法としては，決定木，ランダム森，サポートベクターマシン，ニューラルネットワーク，ベイジアンネットワーク，アソシエーションルール，自己組織化マップなどがあります。入門書としては豊田 (2008) などが網羅的で良いですが，より専門的には**機械学習 (Machine Learning)** などのキーワードで選書すると良いでしょう。ここでいう学習とは，データに合わせて重みを調節する方法を指し，機械が自動的に重みを調節していく様を「学習している」と表現しているわけです。巷で A.I.(人工知能) と呼ばれているものはこうした手法の総称で，データに適していることを目的にしているので，パターンがわかれば行動の予測に使えます[*4]。行動の予測ができれば，たとえば小売業では売り上げに直結する戦略が取れるでしょうし，犯罪者のプロファイリングなど，応用的側面はいろいろ考えられます。

じゃあもう非線形モデルが最強じゃないか，と思うかもしれませんが，この系列の総合的な問題点は「機械がなぜそのような予測をしたのか説明できない」という点にあります。機械には学び方を教えていますが，どう学んだかは機械次第なのです。理屈はわからなくても正解が出せる，というのは実用的にはいいのですが，科学的な研究の場合は少し困ります。機械が勝手に人の心を「うんうん，わかりますよ」と言ったとしても，

[*2] これは**共分散構造分析 (Covariance Structural Analysis)** と呼ばれることもあります。

[*3] この方法は類似の名称があること，あまりメジャーな分析方法でないこともあって，専門書もすくないのですが，技術としては非常に興味深い手法です。SEM が線形モデルの王道であり線形関係を正面から捉えているのに対し，グラフィカルモデリングは関係を裏から捉える裏線形とでもいうべき手法です。もう少し言葉を足すと，SEM がどこに線形関係があるのかを探っていくのに対し，グラフィカルモデリングはどこに関係がないか，どことどこの関係が弱いか，というところを探して無意味な関係を切断していき，残った関係が考察すべきものだ，と考えるのです。

[*4] 線形回帰モデルで予測しただけでも，知らない人にとっては人工知能＝機械が計算した予測式だ，と思ってしまっているかもしれません。

「じゃあどうわかったのか，理屈を教えてください」といっても答えられなかったり，同じアルゴリズムでも違う機械が学習すると違う重み係数になったりして一般的な理屈が出てこないということがあります。心理学は実践的な側面もありますが，究極的には人間行動の理論を探しているので，そういう意味では非線形モデルは向いていないのです。

■モデルの展開と全体像　多変量解析の分析方法を，線形モデルと非線形モデルに分割して説明してきました。一言でいうなら，線形モデルは「当てはまらないこともあるけど，理屈で説明がつくモデル」であり，いわば理屈が先，現象が後です。非線形モデルは「理屈はわからないけど，データには当てはまるモデル」であり，いわば現象が先，理屈が後なのです。

どちらもデータとの当てはまりを基準にしてはいますが，その背後の考え方が違っているわけですね。(正規) 線形モデルも非線形モデルも，制約を増やしたり減らしたりしながら限界とされている点を克服するべく，日々モデルの改良がなされています。

たとえば線形モデルの世界でも，正規分布以外の**確率分布**を仮定できます。それらは**一般化線形モデル** **(Generalized Liner Model)** と呼ばれ，さまざまな確率分布を仮定した上で，線形関係を見出す手法です。心理学の研究対象としているデータは，目に見えない心的状態を対象とすることが多いので，正規分布を仮定することが一般的でした。もちろん数学的に，正規分布を仮定するモデルの方が単純な形になるので，そちらの発展が先に進んだという実情もあります。正規分布以外の形をするデータがなかったわけではありません。所得のデータは対数正規分布のようになりますし，比率のデータは 0 から 1 までの範囲にしか値を取らないので平均が 0.5 からズレれば左右対称の形にはなりません。友人の数を数える時のように，0，1，2... と正の整数しか取らない離散的なデータというのもあります。しかし分析モデルが正規分布を仮定したものしかなかったので，これまではデータを正規分布の形になるように変換して分析する，ということが行われてきました。今は一般化線形モデルを使って，データの形式にあった分析をすることが基本です[*5]。

このように，正規分布の線形モデルが非常に多くあるのですが，その制約を外す方向で統計モデルが展開してきているわけです。制約の外し方としては，ここで挙げた「正規分布以外の確率モデルを使う」ということもありますし，分布の歪度・尖度といったより高次の**積率 (moment)** を使う方法があります[*6]。

また，**数理モデリング (Mathematical Modeling)** というアプローチもあります。これは線形の仮定を外し，理屈の通る変数関係を数式で表現してデータにフィットさせるというアプローチです。当然非線形な関係になりますし，正規分布以外の確率分布も使います。この時，さまざまな確率分布が入れ子になったモデルになるので，推定方法としてはベイズ法を使うことになります。**ベイズ法**によるモデリングアプローチは最近の計算機技術革新によってとても身近なものになりました。この授業でも第 16 講から扱うことになります。

さて，多変量データを扱う世界の全体像がわかったところで，まずは線形モデルの世界から少しずつ進んでいくことにしましょう。そこで，統計モデルの種類にも大きく関わる尺度水準や，記述統計量について，改めて説明を加えておきたいと思います。

1.2 尺度の四水準

心理統計のテキストは何を開いても，まず Stevens (1946) による**尺度水準 (level of measurement)** についての言及があります。何を測定した数値であるかとは別に，その数値にどのような算術処理を施すことができるかによって，数字を 4 つのレベルに分けるのでした。

---

*5 古い論文を読むと，正解率のような比率のデータに対して**分散分析**を行ったりしていましたが，このような理由から今では推奨されません。

*6 積率について，詳しくはこの後の 1.3 節で。

■名義尺度水準　名義尺度水準 (nominal scale) は数字と対象が 1 対 1 で対応していることだけが重要です。男性を 1, 女性を 2 とコード化するようなもので，この時「女性は男性の 2 倍である」といった数としての意味はありません。男性を 0, 女性を 42 としても本質的に変わりがないからです。このような名前だけの数字は，計算ができませんので，せいぜい 1 が何件あったかという度数を数えて集計するにとどまります。

とはいえ，数字が直接対象を指し示しているわけですから，もっとも意味のある数字かもしれません。

■順序尺度水準　順序尺度水準 (ordinal scale) は，数字が大小関係の意味を持っているものです。レースで 1 位 2 位と順番がつくと，1 位のほうが 2 位より優れていることがわかります。人間の心理的な反応，とくに 5 段階や 7 段階で評定させる心理尺度は，この水準に相当します。選好の順序は明確でも，量的な違いがわからないからです。

■間隔尺度水準　間隔尺度水準 (interval scale) は，数字と数字の間隔が等しいことが制約として加わります。たとえば気温で 10 ℃と 20 ℃の差は，25 ℃と 35 ℃の差に等しいと言えます。これは摂氏が氷点を 0 ℃, 沸点を 100 ℃としたうえで百等分したという定義から明らかなことです。間隔が整っているので加法・減法の計算は可能ですが，原点が定かでないので比を考えることはできません。たとえば 10 ℃は 20 ℃の倍の熱量を持っている，とは言えないのです。なぜでしょうか。たとえば，同じエネルギー状態を別の温度体系に置き換えてみたとしましょう。新しい温度体系は，氷点が 100 で沸点が 200 だったとします。そうすると 10 ℃は 110，20 ℃は 120 に該当しますが，120 は 110 の 2 倍にはなっていないからです。

■比率尺度水準　比率尺度水準 (ratio scale) は，さらに絶対 0 点の制約を付け加えたものです。これで原点からどれ位離れているか，を基準にして計算ができますので，乗法・除法もできることになりました。物理的な単位系はこの尺度水準にあるものがほとんどですから，緻密な計算モデルを作ることができるのですね。

さて早足で 4 つの水準について説明をしてきました。これが重要なのは，尺度水準によってできる計算が変わってくる点にあります。名義尺度水準は数え上げぐらいしかできません。順序尺度水準も同様で，名義や順序といった**質的変数 (categorical variables)** の場合，たとえば代表値を求める時も度数を数えて最頻値を報告する，というぐらいがせいぜいなのです。

これに対して，間隔尺度水準や比率尺度水準の**量的変数 (numeric variables)** では，加減乗除の計算ができますので，平均値を求めたり標準偏差を求めたり，ということができるようになります。

先ほど心理尺度は順序尺度水準でしかない，という話をしましたが，質問紙調査の研究例では尺度平均点を出したり，さらに進んだ統計手法で分析したりします。実はそれができるようになるためには，尺度につけられたカテゴリー (「非常に当てはまる」「どちらとも言えない」など) を，尺度値 (「非常に当てはまる」を 5,「やや当てはまる」を 4 とする，など) にする作業を経ており，その時に「非常に当てはまる」を 5 とするのはなぜか，という理屈が必要です。それについては心理尺度の作成の折に詳しく説明しますが[*7]，少なくとも盲目的に行っているわけではないことに注意が必要です[*8]。

尺度値の付与の仕方は色々考えられます。順序尺度水準でとられたデータであっても，その背後に連続的な心理的実態があると考えてその時の相対的な大きさをつけることもできます[*9]。あるいは 0 か 1 かという二値データであっても，それを重みづけて合算することで連続的な値にすることもできます[*10]。さらに名義的な尺度水準であっても，データ全体なかで直線的な関係が最も大きくなるように数字を与えることもでき

[*7] 第 3 講を参照してください。
[*8] 残念ながら，この辺りの理屈を気にせずに分析する人が少なくありません。そんな人に，「数字では心がわからない」なんて言って欲しくないですねえ。
[*9] 因子分析法の一種，**段階反応モデル (Gradad Response Model)** と呼ばれる手法がこれにあたります。
[*10] これはテスト理論の考え方です。0 が誤答，1 が正答としてテストの点数から学力を考えるのですね。

ます[*11]。

このように，尺度水準がかわると，適用できる分析モデルが変わります。たとえば因子分析は間隔尺度水準以上のデータにしか適用できませんが，順序尺度水準のデータであれば因子分析ではなく，段階反応モデルのようなカテゴリカル因子分析を適用しなければなりません。名義尺度水準のデータであれば双対尺度法を適用しなければなりません。間隔尺度水準以上のデータに対して，双対尺度法を適用することは可能ですが，その逆は不可能です。たとえば身長のデータとして $X = \{170, 175, 165\}$ というのがあったときに，連続変数として平均値を求めることもできますし，名義尺度水準として各一件とカウントすることもできるのですが，男性 2 名と女性 1 名がいたので平均 1.3 の性別があるというのは意味をなさないからです。

データがどの水準にあるのかを見極められないと，間違えた分析をすることにもなりますので，注意してください。

1.3　平均と分散

間隔尺度水準以上の数字であれば，平均値や標準偏差の計算が可能です。(算術)**平均 (mean)** は次の式によって計算されるのでした。

$$\bar{x_j} = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$$

総和の記号である $\sum$ の使い方などを再確認しておいてください。また変数 j の平均は $\bar{x_j}$ と表しますが，$\sum$ の記号の中では $i = 1$ から N までと，i だけが変化しています。j は変化していません。

$\sum$ の記号はこの後も所々出てきます。計算の際に次のような変形をすることがありますので確認しておいてください。

$$\sum_{i=1}^{N} cx_i = (cx_1 + cx_2 + \cdots + cx_n) = c(x_1 + x_2 + \cdots + x_n) = c\sum x_i$$

$$\sum_{i=1}^{N}(x_i + y_i) = (x_1 + y_1 + x_2 + y_2 + \cdots) = \sum x_i + \sum y_i$$

続いて**分散 (variance)** の式を確認しましょう。

$$s_x^2 = \frac{1}{N}\sum(x_i - \bar{x})^2$$

言葉でいうなら，平均偏差 $(x_i - \bar{x})$ の二乗の平均です。平均偏差は各点が平均点からどれぐらい離れているかを表します。その平均をとる操作 $(\frac{1}{N}\sum)$ なので，平均的にどれぐらい離れているかがわかるのですが，そのまま平均偏差の平均をとるとゼロになりますので[*12]，二乗するものをその値にするのでした。

この式は次のように展開できます。

[*11] 双対尺度法という考え方がこれにあたります。詳しくは西里 (2010) を参照。

[*12] 平均値が全部のデータの真ん中に位置するように撮られた指標だから当然です。

$$
\begin{aligned}
s_x^2 &= \frac{1}{N}\sum(x_i-\bar{x})^2 && \text{定義より}\\
&= \frac{1}{N}\sum(x_i-\bar{x})(x_i-\bar{x}) \\
&= \frac{1}{N}\sum(x_i^2-2x_i\bar{x}+\bar{x}^2) && \text{(後ろのカッコを展開)}\\
&= \frac{1}{N}\sum x_i^2-\frac{1}{N}\sum(2x_i\bar{x})+\frac{1}{N}\sum\bar{x}^2 && \textstyle\sum \text{記号の分配}\\
&= \frac{1}{N}\sum x_i^2-2\bar{x}\frac{1}{N}\sum x_i+\frac{1}{N}N\bar{x}^2 && i\ \text{が変化するところだけにつく}\\
&= \frac{1}{N}\sum x_i^2-2\bar{x}^2+\bar{x}^2 \\
&= \frac{1}{N}\sum x_i^2-\bar{x}^2
\end{aligned}
$$

計算するときは最後の形を使うことがあります。

この式で表される分散は，変数がどの程度変化しうるかということを表す値であり，変数から引き出せる情報の上限でもあります。分散が 0，つまり変数がまったく変化しなければ，どういうときに値が大きくなってどういうときに値が小さくなるのかという違いがわからないということです。違いがないものについては，それ以上考察のしようがないか，当たり前のことを言っているに過ぎないということになります。たとえば質問紙調査で「他人に激しく殴られるのが好きですか」という聞き方をすると，誰しも「まったく当てはまらない」と答えると思います[*13]。この項目から何かがわかるか，といわれても当たり前のことすぎて何もわからない，としか言えないでしょう。これに対して「対面している相手の手足の動きが気になりますか」というような項目であれば，気にする人もいるでしょうし，気にしない人もいるでしょうから，回答に分散が生まれます。そうすると，気になった人はどういう人なのか，気にならなかった人はどういう人なのか，という考察に進むことになるわけです。このように，調査データから意味のある考察をするためには，分散の大きな項目を作らなければならないのです。

ところで，分散の式の中には二乗の項が入っていますから，このままでは元のデータと単位が異なってしまいます。そこでこの単位を整えるために，分散の正の平方根をとったものを**標準偏差 (Standard Deviation)** といって散らばりの指標に使います。本講では標準偏差の記号を s_x と表すことにします。

平均は中心化傾向の指標の一種で，他にも中央値，最頻値などがあります。分散や標準偏差は散らばりの指標の一種で，他に最大値，最小値，範囲，IQR，パーセンタイルなどがあります。これら記述統計量は，データの特徴を記述するため，データ分析の最初のステップで確認すべき数字です。また，分散は平均偏差を二乗したものの平均でしたが，これを三乗したものは**歪度 (skewness)** といいます。歪度 s_x^3 の式は次のとおりです。

$$s_x^3 = \frac{1}{N}\sum(x_i-\bar{x})^3$$

歪度はマイナスになると左方向に，プラスになると右方向に，分布が歪んでいることを示します。ゼロに近ければ左右対称に近いことがわかります。これも記述統計量としてデータの特徴記述に利用できるでしょう。さらに四乗したものは，**尖度 (kurtiosis)** と言われます。

$$s_x^4 = \frac{1}{N}\sum(x_i-\bar{x})^4$$

[*13] 中にはマゾヒズムの人がいるかもしれない，という屁理屈はここでは脇に置いておいてください。

尖度がゼロであれば，分布の山の尖りぐあいが平均的で，マイナスになれば潰れた山，プラスになれば尖った山の形をした分布になることがわかります。

分散が二乗，歪度が三乗，尖度が四乗でした。これらは分布の中心からどれぐらい離れているかについての累乗で，平均値は一乗したものと理解することができます。これを力学の用語を借りて**モーメント (moment)** といいます。日本語では**積率**と訳されています。重心からの距離に関する指標という意味で一般化された表現です。多変量解析のほとんどは，二次のモーメント (分散) までしか活用しませんが，3 次，4 次のモーメントを使うとなるとデータから得られる情報が増えることになり，より表現力を増した分析ができるようになります[*14]。

1.4　共分散と相関係数

さて，分散が 1 変数の変動を表現する数字であり，そこから得られる情報の上限であるという説明をしました。実際に調査的な研究をするときは，いくつもの変数を同時に扱うことになるわけですが，そうすると変数と変数の関係を考える必要があります。

改めて分散の式を見ると，次のようになっているのでした。

$$s_x^2 = \frac{1}{N}\sum(x_i - \bar{x})^2 = \frac{1}{N}\sum(x_i - \bar{x})(x_i - \bar{x})$$

この式を少し変えて次のようにすることで，変数 x と y の関係を考える式になります。

$$s_{xy} = \frac{1}{N}\sum(x_i - \bar{x})(y_i - \bar{y})$$

この式で表される数字を**共分散 (covariance)** といいます。異なる変数ですので異なる記号で現しましたが，変数を x_{ij} のように個人 i と変数 j という形で表すならば，変数 j と k の共分散を表す式は次のように書けます。添字のつき方に注意して理解してください。

$$s_{jk} = \frac{1}{N}\sum(x_{ij} - \bar{x_j})(x_{ik} - \bar{x_k})$$

さてこの共分散は，カッコの中がそれぞれの変数の平均偏差を現しており，それを個人ごとに集めて平均しています。あるケースにおいて，平均偏差が同じ方向，すなわちどちらも平均より上であるか，どちらも平均より下であれば，積の符号は正になるのでこの数字は増えます。逆にあるケースにおいて平均偏差が異なる方向，つまり一方は平均より上なのに他方は平均より下であるようなことがあれば，積の符号は負になりますので，この数字は減ります。これを平均するわけですから，データ全体で同じ方向にずれる傾向があるのか，そうでないのかを表す指標ということになります。

同じ方向にずれるということは，一方の動きがわかれば他方の動きが推測できます。このように，共分散の数字は変数間関係から予測，考察をするための情報量を現しているとも言えます。共分散がまったくない，0 であれば，ケースごとにさまざまな可能性があるので一般的なことが言えないわけです。

この共分散は，分散と同じで単位に依存します。たとえば身長と体重の共分散を計算すると，メートルとキログラムをかけ合わさった単位になります。このように，変数ごとに単位が変わると共分散同士の比較が難しくなりますので，データの標準化を考えましょう。すなわち，どのようなデータであっても単位に捉われずに比較できるようにします。スコアの標準化は次の式で行います。

$$z_i = \frac{x_i - \bar{x}}{s_x}$$

[*14] 詳しくは豊田 (2007) を参照してください。

このように平均偏差を標準偏差で割ることで，あるデータがその変数の平均からどれぐらい離れているか，標準偏差を単位とした相対的な大きさで表すことができます。標準化されたデータは，平均が 0，分散が 1 になります。このように，標準化されたスコアは相対的に同じサイズに整えられ，単位に依存しませんので，標準化したスコア同士は相対的に比較可能です。

この標準化スコアしたスコアで共分散を計算したものが，**相関係数 (correlation coefficient)** です。

$$r_{jk} = \frac{1}{N} \sum z_{ij} z_{ik}$$

この相関係数はどんな変数であっても，-1 から $+1$ の範囲に入りますので，変数間関係の相対的な比較が可能です。つまり身長と体重の関係の強さは，身長と足のサイズの関係の強さよりも大きいとかちいさい，といった表現が可能になるわけです。

多変量解析の世界は，変数がたくさんありますから，この共分散の組み合わせもたくさんあります。分散が変数から得られる情報，共分散が変数間関係から得られる情報ですから，あるデータセットから得られるこれらの情報の数はどれぐらいになるか計算してみましょう。

変数が M 個あったとします。変数番号が $1,2,3...M$ ととすると，組み合わせは 1 と 2，1 と 3.... となります。このとき i と i は分散，i と j が共分散になりますが，s_{ij} は s_{ji} と同じですから，$M \times (M-1)/2$ 個の共分散と M 個の分散が，このデータセットから得られる情報のすべてということになります。合計で $M \times (M+1)/2$ 個になりますね。このように組み合わせまで考えると，変数の数がひとつ増えるだけでも得られる情報，分析のヒントがぐっと増えることになります。多変量解析は，こうした変数間関係の情報を使って分析を進めていくことになります。

ちなみに，変数間関係を表す方法は共分散や相関係数だけではありません。これらはあくまでも変数間の直線的な関係を表す指標であることにも注意が必要です。多変量解析全体としては，変数同士の数字のズレをたしあわせた**距離 (distance)** や，ある変数が他の変数と同時に出現した回数をカウントした共頻度などを扱うモデルもあります。

1.5 課題

■尺度水準　4 つの尺度水準の具体例をそれぞれ挙げなさい。

■平均と分散　平均，分散，標準偏差，標準得点の定義を式で表現しなさい。

■共分散と相関係数　共分散の定義式を書きなさい。そのとき，スコアが標準化されていると相関係数になることを示しなさい。

第 2 章

心理尺度を作る

2.1 はじめに

心理学の研究法は調査，実験，観察の 3 つが代表的なものです。とくに調査法は，質問紙を用いて多くの人に回答を依頼し，それを統計的に分析することで研究仮説を確認しようとするものです。調査対象者をどのように集めるか，調査票をどのようにデザインするか，得られたデータをどのように分析するか，といったことでそれぞれ 90 分以上話せるぐらいに色々考えるべきことはありますが，ここではとくにその本質について考えてみましょう。すなわち，なぜ紙で用紙に丸をつけたものを集めただけで，心の何かがわかったと思えるのか，ということです。

アンケート調査は一見すると，なんのひねりもない研究法に思えます。すなわち，聞きたいことを当の本人に聞いてみる，これだけです。しかも答えやすいように (集計しやすいように？) 紙に問題が書いてあって，該当するところに丸をつけるだけでよかったりするのです。この手の研究は少なくとも 100 人，できれば数百人，大きい規模だと千以上の桁数の協力者に回答を求めます。それを PC に入力して集計するわけです[*1]。しかし「そう思う」を 5 点，「ややそう思う」を 4 点，以下 3，2，1 点· · · と入力するのはなぜでしょう。これらの反応は**順序尺度水準**でしかない，と言われますが，それを間隔尺度水準であると「みなし」て，平均値を求めたりします。なんでそんなことが許されるのでしょう。

本稿ではその謎について迫っていきたいと思います。

2.1.1 測ろうとしているもの；態度

調査票で質問する内容は，(お客様アンケートとかではなく) 心理学の研究であれば**尺度 (scale)** を使います。尺度とは，測定したい対象に数字を割り振るルールのことであり，心理尺度は項目とその採点方法が規格化されたものを指します。心理尺度の測ろうとしているものの多くは**態度 (Attitude)** と呼ばれるものです。これは社会心理学の用語で，簡単に定義すると態度とは「行動の準備状態」のことです[*2]。

社会心理学も心理学ですから，研究は基本的に観察可能で客観的なものを対象にします。社会的な行動をテーマにするというのがもちろんですが，社会的な行動というのは状況によって変わるものですし，研究したい状況を待っていても自動的に出てくるものではありません。もちろん実験室などでその状況を作り出すこともやりますが，選挙のような公的なものであれば実験的に作り出すこともできません。しかしたとえば選挙結果などは，社会心理学における政治的行動に大きな影響を及ぼす (あるいは政治的行動の結果になる) ものですから，研究としては非常に重要なテーマになります。選挙になるまでは仕事ができない，というのでは

[*1] もっとも最近では，web で調査の回答を得ることで，入力や誤入力のチェックなどの手間が大幅に軽減されています。
[*2] この定義は Allport (1967) によるものです。

社会心理学者も困りますから,「あなたはどこに投票するつもりですか」と聞いて普段の政治的な態度を調べるという研究手法ができました。社会調査の始まりです。

そしてこれを心理学一般に応用しているのが, 質問紙調査です。心理学の研究テーマは行動ですが, 行動を作り出せない場合や普段の状態を査定したいときに,「調査票に回答を求める」という方法を取るのです。もちろん調査に対する回答は, 嘘や見栄, 間違いや勘違いなども入り込む可能性がありますので, 設計は丁寧に, 分析は慎重に行う必要があります。調査票で過去のことを聞いたりすると記憶が歪んでいる可能性がありますし, 未来のことを聞いたら嘘八百を答えられるかもしれません。今の気持ちを聞いても, 自分の現在の状況がはっきりわかっているかどうか, 怪しいものです。それでもある程度の真実味はあるだろうと考えて, こうした阻害要因を取り除いて本質を掴むために, いろいろな工夫がなされています。

尺度を使って感情や気分, 考え方 (認知傾向) や今後どう行動するつもりか (行動意図) を調べることがなされていますが, 本質的にはこれらすべてが, 広い意味での態度です。態度の認知的側面, 感情的側面, 行動的側面などと言われたりします (藤原, 2001)。この「態度」の説明や定義は, 実はそれほど明確ではありません。しかし, 一般に特定の対象に対する, 正負・量的な評価が可能なものと考えられています。たとえば「自民党に対する態度」というのは, 自民党という対象に対して,「好き」「嫌い」というポジティブ・ネガティブの評価ができ, さらに「とても好き」「やや嫌い」のように量的に表現できるものでもある, と仮定されます。多くの人に多くの項目で調査するのは, 項目同士の誤差が相殺しあってこれが正規分布すると考えられるからであり, 社会的な態度は極端な値が少なく平均的な値を持つひとが多いものとして, 相対的に評価されます。

基本的に測定しようとしているものは, こうした特徴を持ったなにかである, ということをまずは踏まえておきたいと思います[*3]。

2.1.2　3 つの方法

心理尺度の作り方には大きく分けて 3 つのスタイルがあります。

1 つ目はサーストン法による尺度で, **等現間隔法 (method of equal-appearing intervals)** と呼ばれるものです。2 つ目はリッカート法 (Likert 法, ライカートと読む人もいる) と呼ばれるもので, 5 件法, 7 件法など数段階のカテゴリラベルのもっとも近いところに丸をつけるという方法です。3 つ目は SD(Semantic Differential 法, 意味微分法と訳されることも) 法とよばれるものです。SD 法は態度測定というより, イメージの測定を目的としたもので, 対象を提示しつつペアになった形容詞を列挙して提示します。たとえば「専修大学」という対象に対して,「激しい-落ち着いた」「慎重な-軽快な」といった形容詞対ではどちらの表現が近いかを評定してもらいます。形容詞の対が作る軸上でいうと平均的にどのあたりに対象がプロットされるのか, を見ることで対象ごとのイメージの違いを表現するのが基本的なアイデアです。SD 法は複数の対象に対するイメージの相対的比較ですから, スコアの点数化にはそれほど重きを置いていないのでここでは取り上げません。

サーストン法とリッカート法は, これを使って態度のスコアをつけることができます。小杉の自民党に対する態度は 4.8 点だ, といったように, です。こうした数値化がどのような理屈でなされるのかを, 今からみていこうと思います。

*3 性格のように特定の対象を持たないものであっても, 正規分布に従うと考えるのは自然ですから, この後説明する統計技法が適用できるものもあります。感情や気分といったものは, 持続時間が短いので生理指標などで測定するべきであり, 調査票によるアプローチは不向きですが, 逆に言えばある程度一定の安定した心理状態であれば測定することができると考えられているのかもしれません。

2.2 サーストンの等現間隔法

サーストンの等現間隔法（とうげんかんかくほう）は，社会的な態度について絶対評価を与える方法です。この方法で作成された尺度は，各項目 (態度表明文と呼ばれます) に尺度値がついており，回答者は提示された項目に賛成であればその尺度値がその人の態度得点になります。この尺度値は事前に複数の評定者によって決めておく必要があります。すなわち，尺度を作る前の入念な準備が必要です。また，サーストンの尺度は 1 次元性を有していることが前提となります。

具体的な作成方法は次のような手順で行います。

1. 項目の収集
2. 評定者集団による評定
3. 尺度値の算出
4. 項目の選定

以下順に説明します。

■項目の収集　まずは測定したい社会的態度のテーマに沿って，項目を準備します。たとえば「自民党に対する態度」のように，誰でも思い描ける具体的な対象が良いでしょう。このようなテーマが決まれば，これに対する態度項目を色々考えます。「自民党のことを考えると夜も眠れない」とか「自民党に関係したニュースはなるべく見るようにしている」「近所の自民党員の事務所に行くことがある」というポジティブな態度もあるでしょうし，「自民党のニュースはなるべく聞きたくない」「自民党には投票しない」「自民党は不正まみれの悪い政党である」といったネガティブな態度もあるでしょう。こうした文言をなるべく多く，強い態度から弱い態度まで，ポジティブなものからネガティブなものまで網羅的に準備します。ニュートラルな項目も考えておく必要があります。

■評定者集団による評定　尺度値を決めるための事前準備です。まず評定者を無作為に集めます。少なくとも十数人は必要でしょう。評定者には事前に準備した項目が好意的－非好意的（または肯定的－否定的）の 1 次元にそって，7〜11 段階ぐらいの多段階に分類してもらいます。「自民党のことを考えると夜も眠れない」というのは非常にポジティブなので 11 点，「自民党には投票しない」というのはかなりネガティブなので 2 点，といったようにです。

■尺度値の算出　このように各態度表明文を複数人で評価してもらいますから，その項目の平均値，中央値，分散などの記述統計量を計算できます。この中央値 (あるいは平均値) をその項目の尺度値とします。ただし，ここで分散が大きい項目は，評定者によって評定の仕方がバラバラだということを意味しますよね。値が人によって定まらないというのは，その項目が刺激としてあまり好ましくないと考えられるので，項目候補から削除します。誰がみても 10 点とか誰がみても 3 点，といった分散が少ない項目が望ましいでしょう。

■項目の選出　さてこうして尺度値が計算できたら，それを順に並べていきます。「夜も眠れない」は 10.7 点，「事務所に行くことがある」は 9.5 点，「ニュースをなるべく見る」は 7.9 点・・・というようにしていくことができますね[*4]。このとき，項目間の間隔が均等になるように項目を選別します。たとえば「夜も眠れない」と「事務所に行くことがある」の間隔は 1.2 点ですが，「事務所に行くことがある」と「ニュースをなるべく見る」の間隔は 1.6 点になっています。これでは等間隔と言えないので，1.2 点間隔すなわち 8.3 点ぐらいの項目を選

[*4] 中央値なのになぜ小数点があるのだ，と思う人がいるかもしれません。平均値でもいいですし，評定者が偶数人の場合は中央値も両得点の平均や重みつき平均で小数点が出るからです。

出します。

このことからわかるように，サーストン法で尺度を作る場合は，事前に多くの項目を準備しておかないと「ちょうどいいところの表明文がない」となってしまう恐れがあります。ですから最終的にできる尺度に含まれる項目の，5 倍から 10 倍ぐらいを事前に準備し，うまく等間隔に項目が選出できるようにしなければなりません。

なぜ等間隔に選ぶのかというと，もうお分かりですね，これで得られる尺度値を**間隔尺度水準**として扱いたいからです。間隔が等しくなければ順序尺度にしかなりませんが，間隔が等しいことがわかっていると，平均や分散などの計算をし，相対的な比較をできるからです。またこの尺度を使うときは，すでに評定者集団によって尺度値がわかっていますから，回答者にずらりと並べられた尺度を見てもっとも自分の意見に近い項目を選出してもらえば，その項目の尺度値がその人の態度得点だということができます。

評定者集団をなるべく偏りなく多く集めることで，事前に尺度の値を確定させておき，あとは本来研究対象にしたかった人にその尺度を当てれば尺度値 (尺度得点) が求められる方法ですから，準備が大変だけど使うときは確実で絶対的なスコアを与えることができるというのがこの方法の利点です。欠点はその準備コストの高さと，1 次元的な態度しか用いられないことでしょうか。また尺度構成の観点から重要なのは，選出プロセスによって項目間の尺度値が均等であることが保証されている点です。均等に選んだ後で，1，2，3，4，5 と数字を付け直しても構いません。大事なのは，こうしたプロセスのおかげで間隔尺度水準が維持され，以後の分析に耐えうるスコアになっているという点です。

2.3 リッカートのシグマ法

次に紹介するのはリッカートの**シグマ法 (σ method)** です。これはいわゆる 5 件法，7 件法と呼ばれる採点方法で，「私は自民党の政治のやり方が好きだ」といった項目に対して，「まったく当てはまる」「かなり当てはまる」「やや当てはまる」「どちらとも言えない」「やや当てはまらない」「あまり当てはまらない」「まったく当てはまらない」といった順序づけられたカテゴリーにたいしてもっとも自分の考え・態度と近いところに丸をする，という方法で反応が得られます。この時の反応カテゴリーが，今回は 7 つありますから 7 件法 (7-points scale) で回答を求めた，などと言います。5 段階なら 5 件法，4 段階なら 4 件法です。普通は「どちらとも言えない」というところを用意するために奇数 (3,5,7,9) 件法を使いますが，日本人は「どちらとも言えない」を選びやすいという中庸傾向があるとも言われていますので，意見をはっきりさせるために 4,6 件法も使われたりします。

項目はこれも複数あって，たくさん集められた項目を分析するために，もっとも当てはまるを 7，かなり当てはまるを 6，以下同様にしてまったく当てはまらないを 1，とコード化し分析するのが一般的です。ただし注意して欲しいのは，もっとも当てはまる＝ 7 としたのは**名義尺度水準**の数字の割り当て方と一緒で，このカテゴリーが 7 という尺度値を持っているわけではない点です。そもそもこの評定カテゴリーは，統計学的にはせいぜい順序尺度水準の性質しか持っていませんから (もっとも＞かなり＞やや)，カテゴリーに割り当てた数字からそのまま平均や分散の計算をするのはおかしいはずなのです。もしこれらのカテゴリーの間隔が等しかったとしても，3，4 段階しかないようであればやはり間隔尺度水準の計算ができるほどの精度は持っていません。数量的に分析するには (等間隔が担保された上で)9 から 11 段階は必要と言われています。

それではリッカート法ではどのようにして尺度値を決めるのでしょうか。リッカート法も測定しようとしているのは態度であって，表に出てくる反応カテゴリーの背後には連続的な心理的態度というのがある，と仮定しています。またこの (社会) 心理学的態度は，向きと大きさがあって正規分布を仮定できます。リッカート法も正規分布に従う潜在的な連続変数があると仮定するのです。

さて，ある項目について，多くの人からデータを集めて「もっとも当てはまる」「かなり当てはまる」といったカ

テゴリーごとの集計ができたとしましょう。多くの人の態度も集積すれば正規分布に従いますから，きっとこのヒストグラムも正規分布を反映したものになっているはずです。しかし我々が知りたいのはその背後にある連続体上の数字なわけです。

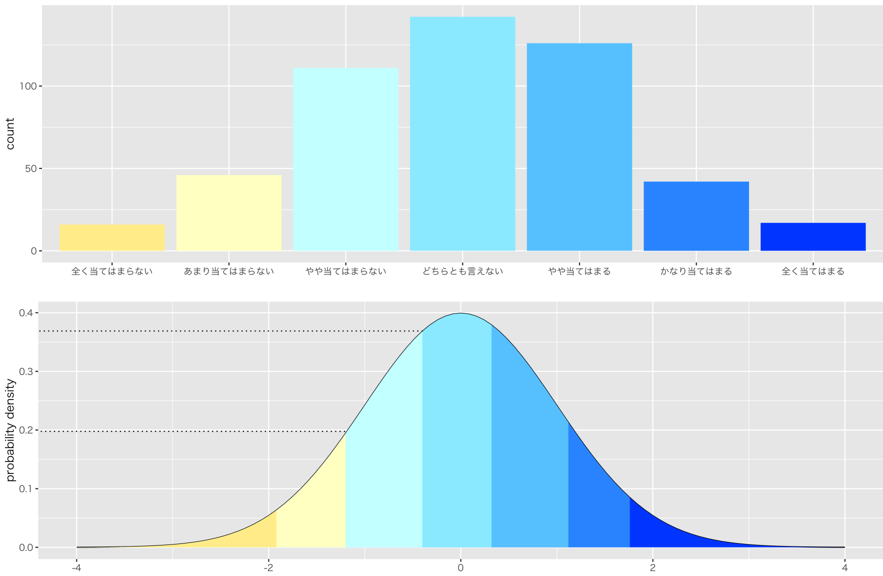


図 2.1　カテゴリ反応と背後の連続値

ここで図 2.1 を見てください。上段にあるのがある項目のヒストグラムの例です。しかし知りたいのは，下段にあるような正規分布の形をした連続体の変数のはずです。上段のヒストグラムは下段の状態を反映しているはずですから，上段のカテゴリの相対頻度を元に，下段の正規分布を分割します。具体的な数字との対応は表 2.1 を見てください。出現度数を相対頻度にし，正規分布の面積を順に分割していくことになります。カテゴリの下の方から順に分割するということで，表 2.1 の三段目には累積 (相対) 頻度を書いてあります。そしてこれを使って，標準正規分布の下から面積を考えます。統計環境 R では，`qnorm` 関数をつかうと累積確率の確率点が求められるのでした (表 2.1 の 4 段目)。またその時の確率密度も求めてあります (表の 5 段目。R では `dnorm` 関数を使って求めます)。

さて，ではここからどのようにして尺度値を求めればいいでしょうか。一般に C 件法で，下から $1, 2, 3, \cdots c \cdots, C$ とカテゴリ順に数字を割り振ったとして，第 c カテゴリの尺度値 Z_c を考えるとします。このカテゴリ c は標準正規分布において上限 z_c，下限 z_{c-1} の確率点で挟まれる領域としていますから，この幅 $([z_{c-1}, z_c])$ の平均を取ることを考えます。この点は，次の式で求められます (証明は付録 B,Pp.377 参照)。

$$Z_c = \frac{(y_{z_{c-1}} - y_{z_c})}{p_c}$$

ここで y_c は z_c, z_{c-1} における確率密度，p_c はカテゴリ c の相対頻度です。具体例でいきましょう。表 2.1 の数字を使うと，「やや当てはまらない」$(c = 3)$ の尺度値は，$\frac{0.20 - (0.37)}{0.22} = -0.7727273$ となります (分子は図 2.1 の点線部の差分，分母は該当箇所の面積になります)[*5]。このようにして計算された尺度値が，表 2.1 の一番下の段にある数字です。

[*5] 表 2.1 は小数点下 2 桁までに丸めているので，正確な値ではありません。

表 2.1　カテゴリと数値の対応

反応カテゴリ	まったく当てはまらない	あまり当てはまらない	やや当てはまらない	どちらとも言えない	やや当てはまる	かなり当てはまる	まったく当てはまる
出現度数	16	46	111	142	126	42	17
相対度数	0.03	0.09	0.22	0.28	0.25	0.08	0.03
累積相対度数	0.03	0.12	0.35	0.63	0.88	0.97	1.00
累積相対度数の確率点	-1.85	-1.16	-0.40	0.33	1.19	1.83	∞
累積相対度数の確率密度	0.07	0.20	0.37	0.38	0.20	0.08	0.00
付与される尺度値	-2.33	-1.44	-0.77	-0.04	0.72	1.50	2.67

このように，累積度数をつかって尺度値を決めるリッカートの方法を**シグマ法**と言います。こうして作られた尺度値は連続体上の数字ですから間隔尺度水準になり，平均や分散をはじめとした数値計算に耐えうる値になっています。機械的に「まったく当てはまらない」から「まったく当てはまる」まで，1.0 刻みで数字を割り振っているのではないのです！

・・・と言いたいところですが，今回の尺度値を眺めてみるとそこそこ等間隔 (間隔は 0.8 ぐらいでしょうか) に並んでいますね。試しに各尺度値を 0.8 で割ってみると，$-2.91, -1.81, -0.97, -0.04, 0.90, 1.88, 3.33$ となります。四捨五入して小数点をなくしてみると，$-3, -2, -1, 0, 1, 2, 3$ となりますね。そう，つまり**非常にラフな近似値でよければ，機械的に 1,2,3... と数字を振っても同じことになります**。ですから，実際の研究ではとくに深く考えずに 1,2,3... と割り振った数字をそのまま使われたりするのです。大山鳴動して鼠一匹といいますか，苦労した割に得るものがないじゃないか，とお叱りを受けそうですが，少なくとも「なぜリッカート法は順序尺度ではなく間隔尺度のように扱って良いのか」という問いには答えられると思います。また，ここにくるまでに，態度の 1 次元性や正規分布の仮定などが含まれていたことを改めて思い出してください。今回は数値例ですので，綺麗に七段階に分かれるようなものを用意しましたが，実際の調査では正規分布しないものや，平均が低すぎるとか高すぎるものが結構みられます。それらに対して機械的につけた数字で分析するのは決して適切な方法ではなく，シグマ法を始めその他の手法で適切な尺度値を付与すべきなのですが，人間は易きに流れるもので**ほとんど考慮されていないのが現状です**[*6]。

2.4　尺度を評価する

このようにして作られた心理尺度は，それがきちんと測定したいものを測定できているか，評価する必要があります。ここでは**信頼性 (reliability)** と**妥当性 (validity)** の 2 つの側面から説明します。

[*6] この状況は決して良いものではなく，悪しき研究上の風習だと思われます。幸い，第 4 講で説明する**項目反応理論 (Item Response Theory)** の一種，**段階反応モデル (Greaded Response Model)** を用いると，この問題点をカヴァーしつつ有用な情報が得られますので，みなさんは一足飛びにその手法を身につけた方が良いと思います。

2.4.1 信頼性

信頼性は測定の安定性と言い換えてもいいかもしれません。すなわち，同じものを 2 回測っても同じ数字がつくことですね。測定するたびに数字が変わるようでは，その測定器 (ここでは尺度ですが) は信用ならない，というわけです。

テスト理論の文脈では，テストのスコア X を本当に測りたいもののスコア t と誤差 e とに分割して考えます。古典的テスト理論のモデル式は次の通りです。

$$X = t + e$$

ここで，各項目についても同じことが言えると考えると，$X_i = t_i + e_i$ ということになります。複数項目で測定するのは，これの平均値を考えると $\bar{e} = 0$(誤差の平均値はゼロになる) という仮定から，$\bar{X} = \bar{t}$ となって真のスコアを得ることができる，ということがわかります[*7]また，テストの分散 $Var(X)$ を考えると，テスト全体の分散は $Var(X) = Var(t) + Var(e)$ となり，真のスコアの分散と誤差の分散に分解できることがわかります。ここから，信頼性 Rel を次のように表現できます。

$$Rel = \frac{Var(t)}{Var(X)} = \frac{Var(X) - Var(e)}{Var(X)} = 1 - \frac{Var(e)}{Var(X)}$$

言葉で言えば，信頼性は全分散に占める真分散の割合であり，全体から誤差分散の割合を引き算したものであるとも言えます。

信頼性のない尺度があれば，その後のお話は先に進みませんから，まずもって「尺度が信頼できるかどうか」を評価する必要があります。このことを信頼性は妥当性の上限，と表現したりします。信頼性を評価する方法は，測定値の安定の程度を評価できればいいのですから，複数の測定を持ってその相関係数を計算することでひとまず達成できます。しかし同じ尺度を何度も使うのは，調査回答者に要らぬ構えを持たせてしまいますから，普通は 1 回の尺度を分割してその特徴を見ることにします。尺度全体から計算される回答者の値は，項目の尺度値の合計であるのが普通です (テストの点数も正答数に対応していますね)。ですから，ある項目 j の尺度値は，j を除いた残り $M - j$ 個の尺度値の和と高い相関をするはずです。このように，各項目が尺度全体の値とどの程度相関しているかを見る **IT 相関 (Item-Total correlations)** は，尺度の信頼性を見る 1 つの指標になります。ある項目が，尺度全体と相関していなければ，それはその項目が尺度の中で目的と違うものを測定している可能性があり，それは必然的に尺度の安定を損ねる結果になるからです。

この考え方を発展させ，各項目が他の項目とどの程度相関しあっているか，つまり尺度の中でどの程度整合性がとれたもの＝一貫して同じものを測定しているのかを評価する指標として，**α 係数 (alpha coefficient)** があります[*8]。これは，テストに含まれる項目数を M，テスト全体の分散を V_t，項目 j の分散を V_j と表すと，次の式で表されます。

$$\alpha = \frac{M}{M-1} \times \left(1 - \frac{\sum V_j}{V_t}\right)$$

この指標は，各項目が他の項目とどれぐらい相関するかを総合的に表した指標で，とくに**内的整合性信頼性**と呼ばれます。このようにして，尺度の安定の程度である信頼性は数値化できますが，次にお話しする妥当

[*7] この点については，心理学データ解析基礎の授業でも触れていますので，あっさりとした説明になっています。よくわからない人は「基礎」の方の資料を読み直して確認しておいてください。

[*8] **クロンバックのアルファ (Cronbach's alpha)** とも呼ばれます。

性については, 数値化できないものです。

2.4.2 妥当性

妥当性 (validity) は, 信頼性をその上限とした上で, それが何を測っているのかを改めて考える指標です。

たとえば身長を測ろうとして, 体重計を使うとします。成長に応じて, 身長が伸びますが, それは体重とも関係がありますので, 身長の伸びに応じて体重も増えていくでしょう。体重計は安定した計測器で, 信頼性は十分あると思いますが, 体重計で身長が測れていると言えるでしょうか。相関する変数ですので, 部分的に Yes といえそうですが, やはり身長は身長計で測ったほうが良いでしょう。身長計の方が, 身長という特徴を的確に捉え, より本質に近い測定をしているからです。このように, 作ったものがしっかりとその本質を捉えているかどうか, これが妥当性の基本的なポイントです。

妥当性はですから, そもそも概念としてその測定しようとしているものが適切かどうか (**構成概念妥当性** (construct validity)) とか, その文言でちゃんと質問できているか (**内容的妥当性** (content validity)), 理屈通りその測定値が結果と変動しているか (**基準関連妥当性** (criterion validity)) と言った面から検証されます。最後の基準関連妥当性については, 基準値と尺度値をつかって数量的に検証できますが, 構成概念妥当性や内容的妥当性は中身の問題であったり, 言葉と概念の対応であったりするので, 数理モデル的アプローチができるものではありません。数値化できないか大きな問題ではない, というのはもちろん逆で, 数値化できないところであるからこそ, 専門的な観点, 幅広い視野, 批判的思考でもって検証していかなければなりません。

量的に評価する方法としては, 今後説明していく**因子分析** (Factor Analysis) によって**因子的妥当性** (factorial validity) を見る方法ですとか, 理論通りの因子に分離できているかを見る**弁別的妥当性** (distinctive validity), あるいは**収束的妥当性** (convergent validity) などがあります。これらをまとめて, **検証的因子分析** (confirmatory factor analysis) によって理論通りの分類ができているかをモデル適合度の観点から評価する方法もあります。これらは次回以降お話ししていく, テスト理論の発展系のなかで考えていくものですので, どうぞお楽しみに。

2.5 課題

■リッカート法 シグマ法でなく機械的に数字を割り振るとどのような問題が生じるか, 自分で数値例を作って検証してみてください。

■信頼性の記述と報告 心理学の尺度作成に関する論文[*9]を読み, 信頼性についてどのように記述されているかを確認してみましょう。

■さまざまな妥当性 妥当性にはさまざまなものがあります。Grimm and Yarnold (2001) などを参考に, 妥当性について自分なりに調べてみてください。

■リッカートのシグマ法 表 2.2 のように,「かなり当てはまる」や「まったく当てはまる」など尺度の右の方に丸をつける人が多かった項目があったとします。この時の尺度値をリッカートのシグマ法に則って算出してみてください。

[*9] 日本心理学会が出している「心理学研究」という学会誌では,【資料】というカテゴリーで毎回のように新しい尺度が報告されています。

表 2.2　天井効果の出た尺度

反応カテゴリ	まったく当てはまらない	あまり当てはまらない	やや当てはまらない	どちらとも言えない	やや当てはまる	かなり当てはまる	まったく当てはまる
出現度数	1.00	3.00	5.00	18.00	24.00	53.00	56.00
相対度数	0.01	0.02	0.03	0.11	0.15	0.33	0.35
累積相対度数	0.01	0.03	0.06	0.17	0.32	0.65	1.00
累積相対度数の確率点	-2.50	-1.96	-1.59	-0.96	-0.47	0.39	∞
累積相対度数の確率密度	0.02	0.06	0.11	0.25	0.36	0.37	0.00

第 3 章

テスト理論と因子分析

3.1　古典的テスト理論

前回の信頼性についての議論のなかで，古典的テスト理論について触れました。古典的テスト理論のモデルは $X = t + e$ で表されます。すなわち，テストの点数 X は真のスコア t と誤差 e に分割できるというものです。非常に単純なモデルですが，測定したものには誤差がついているという考え方，言い換えると目に見えるものだけが真実ではないという考え方がしめされています。この考え方は，ソフトサイエンスの領域においては重要なことです。

またこの古典的テスト理論から，いくつかの重要な考えを読み取ることができます。ひとつは誤差についての考え方です。このモデルを $X_j = t_j + e_j$ のように，ある個人の変化しない属性について，j 回測定したとします。この時，測定の平均は次のように計算できます[*1]。

$$
\begin{aligned}
\bar{X} &= \frac{1}{n}\sum_{j=1}^{n} X_j \\
&= \frac{1}{n}\sum_{i=j}^{n}(t_j + e_j) && \text{定義より} \\
&= \frac{1}{n}\sum_{j=1}^{n} t_j + \frac{1}{n}\sum_{j=1}^{n} e_j && \text{分配して} \\
&= \bar{t} + \bar{e}
\end{aligned}
$$

さて古典的テスト理論では，誤差に関して次のことが仮定されます。

- 誤差の平均はゼロ。つまり誤差が出現するときは，「正に偏る」「負に偏る」といった一貫した傾向がないと考える。
- 真のスコアと誤差との相関はゼロ。つまり誤差は真のスコアに関係なく影響してくるもので，真のスコアと共変動するようであれば偶然によるものとはいえない。
- 異なる測定誤差間の相関はゼロ。誤差同士がなにか意味のある変動をしているのであれば，それはもう偶然によるものとはいえない。

これらはいずれも，誤差が「偶然によって現れる影響で，制御不可能なもの」という考え方からは自然

[*1] この式は，ある測定を多くの個人 i に対して行ったものとして，$X_i = t_i + e_i$ と考えることもできますが，添字が異なるだけで式の展開に違いはありません。

な仮定だといえるでしょう。より詳しくいえば，誤差は測定に応じて毎回一定の傾向で生じる**系統誤差 (systematic error)** と，全く傾向のつかめない**偶然誤差 (random error)** に分けて考えられますが，系統誤差は測定に際して工夫して取り除くべき問題であり，ここでは全くの偶然による誤差についての議論だからです。

さて誤差の平均がゼロ，つまり $\bar{e} = 0$ ですから，$\bar{X} = \bar{t}$ となって，いつかは誤差がなくなって真のスコアを得ることができるようになる，ということが示されます。

また平均は 0 ですが分散はゼロではありません[*2]。このテストの分散を考えると，次のようなことがわかります。

$$
\begin{aligned}
Var(X) &= \frac{1}{n}\sum_{i=j}^{n}(X_j - \bar{X})^2 && \text{定義より} \\
&= \frac{1}{n}\sum_{j=1}^{n}((t_j + e_j) - (\bar{t} + \bar{e}))^2 && \text{X をテスト理論のモデルに} \\
&= \frac{1}{n}\sum_{j=1}^{n}((t_j - \bar{t}) + (e_j - \bar{e}))^2 && \text{同じ記号でまとめる} \\
&= \frac{1}{n}\sum_{j=1}^{n}((t_j - \bar{t})^2 + 2(t_j - \bar{t})(e_j - \bar{e}) + (e_j - \bar{e})^2) && \text{展開する} \\
&= \frac{1}{n}\sum_{i=1}^{n}(t_j - \bar{t})^2 + \frac{1}{n}\sum_{j=1}^{n}2(t_j - \bar{t})(e_j - \bar{e}) + \frac{1}{n}\sum_{j=1}^{n}(e_j - \bar{e})^2 && \textstyle\sum \text{ を分配} \\
&= Var(t) + 2Cov(te) + Var(e)
\end{aligned}
$$

ここで Cov とは共分散を表しています。第二項の $2Cov(te)$ は真のスコアと誤差との共分散 (を 2 倍したもの) を表していることになりますが，共分散 (相関) がそもそも線形関係を表す指標であったことを思い出してください。相関係数は共分散を標準化したものだったわけですが，そういう意味ではこの $Cov(te)$ というのは真のスコアと誤差との相関関係を表しているようなものです。さて，ここでも誤差の仮定から，相関はゼロです。すなわち，誤差とはどのような傾向もなく出現するもの，という考えられているのです。どのような傾向もないわけですから，当然なにかと相関関係にあるはずがない，すなわち $Cov(te) = 0$ であるとなります。

これを踏まえると，$Var(X) = Var(t) + Var(e)$ のように，テストの分散が真のスコアの分散と誤差の分散に完全に分割されました。ここから，信頼性の定義は $\dfrac{Var(t)}{Var(X)} = \dfrac{Var(t)}{Var(t) + Var(e)}$ と表すことができるのですね。言葉で言えば，**信頼性**の定義は「全分散中にしめる真のスコアの分散の割合」ということになります。

ここまでは古典的テスト理論から示されることであり，これまでの復習になります。ここから，このテスト理論をより展開させていくことを考えましょう。

3.2 因子分析モデル

3.2.1 単因子モデル

今からお話しするのは，**因子分析 (Factor Analysis)** というモデルです。因子分析モデルは古典的テスト理論の拡張であり，もっとも簡単な 1 因子モデルは次のように表されます。

[*2] ガウスの考えた誤差論から，誤差は確率**正規分布 (Gaussian Curve)** に従うと考えられます。

$$z_{ij} = a_j f_i + e_{ij}$$

ここで Z_{ij} は個人 i の項目 j に対する反応を標準得点で表したもの[*3]，a_j は項目 j の**因子負荷量 (factor loading)**，f_i は個人 i の**因子得点 (factor score)**，e_{ij} は個人 i と項目 j の組み合わさった時に生じた誤差と呼ばれます。

因子負荷量 (factor loading) とは，因子というこのテストで測定したい特性と，項目との関係の強さを表しているものです。**因子得点 (factor score)** とは，因子というこのテストで測定したい特性と，個人との関係の強さを表しているもので，その人のスコアだということができます。

記号についている添字に注目してください。添字 i は個人を，添字 j は項目を表していますが，因子負荷量は a_j と表されています。つまり項目によって変わる変量です。因子得点は f_i と表されています。つまり人によって変わる変量です。古典的テスト理論をこの添字を使って表現するならば，$X_{ij} = t_i + e_{ij}$ となりますが，これと比べてみると t_i が $a_j f_i$ に変わったのが因子分析モデルだということになります。t_i は個人についての真のスコアなのですが，古典的テスト理論の場合，テストの点数は個人のこの能力だけを反映していると考えられていたことになります。もしテストの問題が難しすぎて，まったく答えることができなければ，その人の能力はゼロということになるわけです。しかし中には悪い問題というのもあって，たとえば小学生に高校で習う知識が必要な問題を解かせるような問題があれば，誰だって解けないかもしれません。解けない問題を出して「学力が低い」と結論づけるのはやや暴力的ですらありますね。このように，古典的テスト理論は項目の良し悪しといったものが評価できないモデルだったのです。

因子分析モデルはこれを改良し，$a_j f_i$ としました。すなわち，ある項目に対する反応 z_{ij} は，その項目が測定したい特徴を十分に反映しているかどうか (a_j) と，その人がその特徴を有しているかどうか (f_i) の両方が成立している必要があるわけです。掛け算ですから，一方がゼロであれば結果もゼロになります。すなわち測定したい特徴を反映していない項目 $(a_j = 0)$ であれば，どれほどそれについての能力 (f_i) が高くても反応できないのです。たとえば美的センスに非常に秀でた人がいても，数学のテストでその能力を反映させることはできませんよね。これは数学のテストというのが数学力を測定するものであって，美的センスを測定するものではないからです。

因子分析は知能検査や性格検査の文脈から生まれてきたものです。心理学において「知能」とは，何にでも応用可能な一般的な知能というのがあるのか，あるいは語彙力や計算力といった複数の個別の能力があるのか，という議論がありました。知能検査としていろいろなものが考えられますが，それらがきちんと当該能力を測定する検査法だったかどうかもわからないわけで，因子分析モデルはそこを評価できるようにした，とも言えます。性格検査についても同様で，特性論的に考えるならば人間の性質というのは複数のもの，たとえば外向性，神経症傾向，開放性，協調性，勤勉性[*4]などがあり，ある性格検査の項目は協調性を測定するのには向いているけれども，神経症傾向を測定するには向いていないということがあるわけです。このように，心理学と因子分析モデルは深い関係があります。

3.2.2 多因子モデル

さて先ほどは一因子，あるいは単因子ともいいますが，測定したい特徴が 1 つだけのモデルでした。学力テストなどは一因子で問題ありません。国語のテストは国語の能力を，数学のテストは数学の能力を測定して

[*3] **標準得点 (Standard Score)** とは，素点 X_j を $Z_j = \dfrac{X_j - \bar{X}}{\sigma_j}$ と変換したものであることを思い出してください。標準化されたスコアは平均が 0，分散が 1 になりますので，単位の異なるもの同士であっても標準得点を使うと相互に比較可能になるのでした。

[*4] 小塩 (2020) のビッグファイブについての解説に基づいています。

いれば良いのであって，数学のテストを解くのに美的センス (真美的能力) が必要というのは，むしろ困った状況ですらありますね。しかし，知能検査や性格検査の場合はそうではありません。ある行動傾向，ある形容詞，ある検査がたった 1 つの能力･性質･心理的要因だけを反映しているとは限りません。たとえば人に優しく振る舞うといっても，その背後に外向性があるのか，あるいはそうすると自分がよく見られるからという利己的な性格があるのか等々が考えられます。1 つの項目に複数の要素 (因子) が複合的に影響していることを考えるべきです。そこで因子の数が 1 つではなく，複数ある多因子モデルを考えることにします。多因子モデルは次のように表現されます。

$$z_{ij} = a_{j1}f_{i1} + a_{j2}f_{i2} + a_{j3}f_{i3} + \cdots + a_{jm}f_{im} + d_j u_{ij} \tag{3.1}$$

記号の意味は単因子モデルと基本的には同じです。z_{ij} は個人 i の項目 j に対する反応を標準得点で表したものであり，a_{jm} は第 m 因子の**因子負荷量**，f_{im} は第 m 因子の**因子得点**を表しています。因子の数が複数あるモデルですから，$a_{j1}, a_{j2}, \cdots, a_{jm}$ とか $f_{i1}, f_{i2}, \cdots, f_{im}$ のように因子の番号と項目･個人の添字の組み合わせになっていることを確認してください。最後の e_{ij} が $d_j u_{ij}$ となっていますが，これは誤差についても他の因子と形を同じくし，項目に依存するものとそれ以外に区別しているだけです。

さて，ここでは因子を m 個あるとしています。この因子はどの項目にも共通して働くので，**共通因子 (common factor)** と呼ばれます。共通因子がいくつあるかは事前にわかりませんが，一般的にその数は数個〜十数個になります。性格心理学は長い研究の中で，性格を表す言葉に共通する因子はおおよそ 5 つぐらいであろう，という答えを得るに至りましたが，それ以外の領域では領域ごとの見解があるでしょう。知能が何種類の因子に分かれるのか，あるいはとある心の状態がどういう構造をしているのか，というのは心理学的にみても十分興味のある考え方です。もちろん因子分析によって得られる因子が，人間の潜在的な知能や概念に直接対応しているとは言えないのですが[*5]，それでも因子がどのような構造 (しくみ) をしているのかについての一定の情報を与えてくれます。多因子モデルが心理学一般で広まったのは，こうした心の「構造」に注目する学問との相性が良かったということでしょう。

3.3 因子分析の定理

3.3.1 因子分析モデルの展開

因子分析モデルも古典的テスト理論のように，式の展開から何が見えてくるか考えてみましょう[*6]。

左辺の z_{ij} は観測されたデータから算出できるものですが，右辺の因子負荷量，因子得点はいずれも未知数です。データに対して未知数が多すぎるようで，これではどのようにして答えを見つけ出せば良いのかわからないかもしれません。たとえばある人のある項目に対する標準得点が 0.12 であるとして，それが 0.4×0.3 で得られるのか，0.2×0.6 で得られるのか，はたまた他の数値の組み合わせで得られるのか，を解く数学的技術は存在しません。この方程式はこのままでは解けないのです。

そこで，この未知数だらけの方程式を解くために，因子について以下のような条件を置きます。

- 共通因子の因子得点，独自因子の因子得点は，標準化されている。すなわち，いずれの因子得点も平均点は 0 であり，分散は 1 である。
- 独自因子は共通因子，他の独自因子と相関しない。

[*5] むしろテスト項目や調査票などに対する反応パターンが因子として出てくるだけで，性格や知能が数次元あるというより，我々は性格や知能を数次元で捉えることしかできない，という言い方の方が正しいでしょう。

[*6] 以下このセクションは小杉 (2018) の原稿を再構成したものです。

この他に，状況に応じて因子同士の間に相関を仮定します。

- 共通因子同士の相関を認めないのを「直交因子モデル」，認めるのを「斜交因子モデル」と呼ぶ。

このような仮定を置いたら問題が解けるようになるのでしょうか？ 実はこの問題を解く鍵は，多変量データであればなんとかなるのです！

2 つの変数，j と k の標準得点から，

$$r_{jk} = \frac{1}{N}\sum_{i=1}^{N} z_{ij} z_{ik} \tag{3.2}$$

のように，相関係数が算出されることを思い出してください。先ほどの因子分析の基本代数式 (式 3.1) をこの式に代入してみましょう。

$$\begin{aligned} r_{jk} &= \frac{1}{N}\sum_{i=1}^{N} z_{ij} z_{ik} \\ &= \frac{1}{N}\sum_{i=1}^{N}(a_{j1}f_{i1} + a_{j2}f_{i2} + \dots + a_{jm}f_{im} + d_j u_{ij})(a_{k1}f_{i1} + a_{k2}f_{i2} + \dots + a_{km}f_{im} + d_k u_{ik}) \end{aligned} \tag{3.3}$$

これは代数の計算としてやっていくと，非常に煩雑で間違いが起きやすそうです。そこで，少し視覚化してわかりやすくしてみましょう。多項式の掛け算は，各項目の総当たり戦ですので，列方向に z_{ij}，行方向に z_{ik} の各要素を置いて，要素同士の組み合わせ表を作ります (図 3.1)。

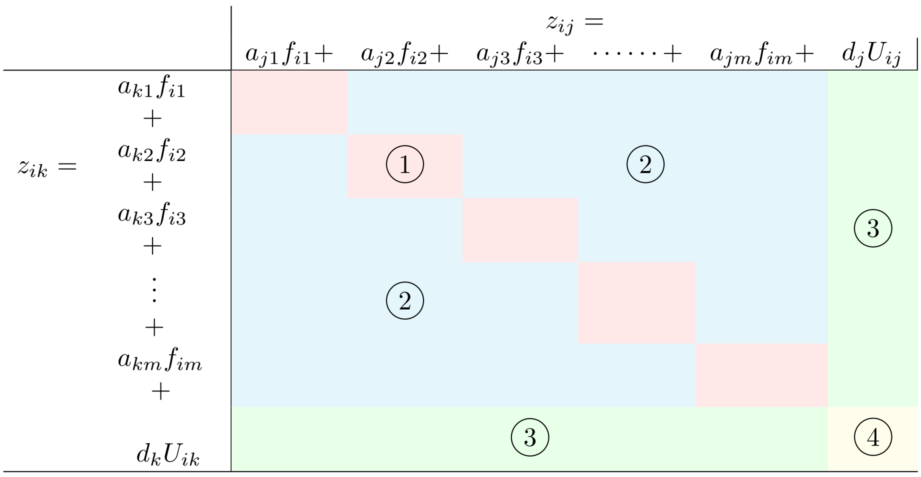


図 3.1 項目同士の総当たりを考える

図 3.1 に示されたのは個人 i についてのものであり，これが人数分ある，すなわち $\sum_{i=1}^{N}$ をつけないといけないことに注意してください。さて図を軽く色分けしてあるのですが，ここにあるように計算すべき領域を 4 つに分けて考えていきましょう。

1の領域 因子 p と p の積和部分 (同じ因子同士の掛け合わせ)
2の領域 因子 p と q の積和部分 (異なる因子同士の掛け合わせ)
3の領域 因子 $p(q)$ と独自因子の積和部分
4の領域 独自因子同士の積和部分

この各パートを順に計算していきましょう。まず①ですが，たとえば第一因子については

$$\frac{1}{N}\sum_{i=1}^{N} a_{j1}a_{k1}F_{i1}^2 \tag{3.4}$$

となることがわかります。ここで，a_{j1} と a_{k1} は N には関係がない（$\sum$ は i が 1 から N まで変化することを意味しているが，係数 i はどちらにも入っていない）ので，総和して割る意味がないことに気づきます。そうなると，必然的にこの式は，

$$\frac{1}{N}\sum_{i=1}^{N} a_{j1}a_{k1}F_{i1}^2 = a_{j1}a_{k1}\frac{1}{N}\sum F_{i1}^2 \tag{3.5}$$

となります。この F_{i1} は因子得点であり，上の仮定より標準化されたものだということになります。さて，標準得点と標準得点の積和平均は相関係数になることをもう一度思い出してください！そうすると，これは自分自身との相関係数を表していることになりますから，当然 $F_{i1}^2 = 1.0$ であることがわかります。

結局，①のエリアは

$$\frac{1}{N}\sum_{i=1}^{N} a_{j1}a_{k1}F_{i1}^2 = a_{j1}a_{k1}\frac{1}{N}\sum F_{i1}^2 = a_{j1}a_{k1} \tag{3.6}$$

と，とてもあっさり書き下すことができるのです。

つづいて②を見てみましょう。ここは異なる因子がかけ合わさった部分ですね。落ち着いて，第一因子と第二因子を例にして考えてみましょう。この箇所で得られるのは，

$$\frac{1}{N}\sum a_{j1}F_{i1}a_{k2}F_{i2} = a_{j1}a_{k2}\frac{1}{N}\sum F_{i1}F_{i2} \tag{3.7}$$

ということになります。ここで，F_{i1} および F_{i2} はそれぞれ第一，第二因子における個人 i の因子得点を意味しています。因子得点は標準化されていることをもう一度思い出すと，これは第一因子と第二因子の相関係数になります。ここで，この因子分析が直交因子モデルだと考えますと，因子同士に相関がないわけですから，数字としては 0.0 で消えてしまいます。するとこの部分は，

$$\frac{1}{N}\sum a_{j1}F_{i1}a_{k2}F_{i2} = a_{j1}a_{k2}\frac{1}{N}\sum F_{i1}F_{i2} = 0 \tag{3.8}$$

となることがわかりました。つまり，この領域②は，すべて 0 になってしまうのです。

続いて③の部分について考えてみましょう。これはある共通因子と独自因子の積和部分です。例によって標準得点同士の関係から，相関係数を算出することになりますが，独自因子は共通因子と無相関であることを考えると，

$$\frac{1}{N}\sum a_{i1}F_{i1}d_jU_{ij} = a_{ik}d_j\frac{1}{N}\sum U_{ij}F_{i1} = 0 \tag{3.9}$$

とこのように，この箇所もすべて 0 になってしまいます。

最後の④に至っては，独自因子と独自因子の積和ですから，これも

$$\frac{1}{N}\sum d_jd_kU_{ij}U_{ik} = d_jd_k\frac{1}{N}\sum U_{ij}U_{ik} = 0 \tag{3.10}$$

のように 0 になります。結局，消えて無くなるのがほとんどで，残るのは①の部分だけであり，r_{jk} を考えるときはそこだけ考慮すればよいことになります。

整理すると，

$$r_{jk} = a_{j1}a_{k1} + a_{j2}a_{k2} + \cdots + a_{jm}a_{km} \tag{3.11}$$

ということがわかります。つまり，項目 j と項目 k の相関係数は，項目 j の因子負荷量と項目 k の因子負荷量を，すべての因子について総和したものであるということです。因子分析の基本モデルから導出されるこの定理を，とくに**因子分析の第二定理**と呼びます。

ここで同じ項目同士の相関を考えてみましょう。項目 j と項目 j の相関係数は，もちろん 1.0 になりますね。これを因子分析の基本式で表すと，次のように表現できます。

$$r_{jj} = a_{j1}^2 + a_{j2}^2 + \cdots + a_{jm}^2 + d_j^2 = 1.0 \tag{3.12}$$

さて，この式が意味するのはなんでしょうか。意味を考えてみると，ある項目それ自身との相関係数は，因子負荷の二乗和からなっている，ということがわかります。これこそ**因子分析の第一定理**と呼ばれるものであり，解けるはずのなかった方程式を解くための鍵となる式なのです。

3.3.2 因子分析の定理

数式の展開はいったんここまでにして，第一定理は次のような形をしているのでした。

$$a_{j1}^2 + a_{j2}^2 + \cdots + a_{jm}^2 + d_j^2 = 1.0$$

ここで共通因子部分を，$a_{j1}^2 + a_{j2}^2 + \cdots + a_{jm}^2 = h_j^2$ のようにすると，この式は単に $h_j^2 + d_j^2 = 1.0$ となります。この h_j^2 のことをとくに**共通性 (communality)** といいますが，この式は共通性と独自因子の二乗和が 1.0 になることを意味しています。言い換えると，全体を 100% とした比率で共通性と誤差を比較できるということです。共通性は因子負荷量の二乗和で，共通因子はそのテストの背後にある共通の要因，すなわちテストで測定したいものだったわけです。古典的テスト理論では，モデル式の展開から信頼性を全分散中に示る真のスコアの割合と定義しましたが，因子分析モデルはこのように 1 つの項目 j における共通因子の割合を算出し，項目の信頼性を考えることができます。因子分析モデルにおける信頼性は，1 項目の中に含まれる共通因子の大きさだとも言えるわけです。逆に $d_j^2 = 1 - h_j^2$ は**独自性 (uniqueness)** は，当該項目がそのテストで測っていないものの大きさを表しており，この割合があまりにも大きいと「この項目は全然関係ないものを測っちゃってるんじゃないか？ 」と疑われることになります。多因子モデルにおいては，多角的に対象を切り分けるために多くの質問を投げかけるわけですが，独自性の高い項目は回答者に負担をかけるだけの邪魔なものですから，実践上はこうした項目を除外することが少なくありません。因子分析には**単純構造の原則 (principle of simple structure)** と呼ばれるものがあり，項目は該当する因子を適切に反映し，かつ，他の因子と関係ないことが美しいとされます。尺度構成段階では，共通性 (独自性) をみて項目の良し悪しが判断されるのです。

次に第二定理を見てみましょう。第二定理は次のような形をしているのでした。

$$r_{jk} = a_{j1}a_{k1} + a_{j2}a_{k2} + \cdots + a_{jm}a_{km}$$

2 つの項目 j と k の相関係数は，それぞれの因子負荷量の積和の形で表される，というものです。ここに誤差の話は入ってこず，共通因子だけで話ができています。

左辺の相関係数は，2 つの項目がどれほど同じものを測定しているかの指標です。相関係数 (の絶対値) が高ければ高いほど，2 つの項目は同じものを指し示しているわけです。逆に相関係数が低いということは，2 つの項目に関係がないことを表します。ここで右辺に目をやりますと，右辺の各項目は因子負荷量の積の形になっています。左辺の値が小さくなる 1 つの理由は，ある共通因子 m が項目 j, k に対して，異なる方向で寄与しているからだと考えられるでしょう。そしてそのパターンが一貫していないという状況です。そもそも相関係数が小さいところからは因子を見つけ出すのは難しいのですが，そうした状況があるのはある項目ペアについて因子同士の向きがバラバラに影響しているからだと言えます。そのような状況は，測定がきちんと

できているかどうか怪しいですね。**測定の一義性**とも言われますが，そのような尺度は妥当性が低いといえるでしょう。

このように，因子分析モデルは第一定理で信頼性を，第二定理で妥当性をあらわすものになっているのです。

3.4 課題

■テスト理論と因子分析　因子分析モデルは古典的テスト理論をどのように発展させたのか，添字に注意しながら数式で表現してみよう。

■因子分析の定理の導出　因子分析の定理の導出を自分でできるようになろう。

■因子分析の定理の意味　因子分析の定理が何を意味しているのか，自分なりの言葉で説明してみよう。

第 4 章

現代テスト理論

4.1　因子分析とテスト理論

ここまで，古典的テスト理論と因子分析の話を見てきました。古典的テスト理論から，テストの信頼性と妥当性の話を導きました。次に因子分析モデルによって，古典的テスト理論が多因子 (多次元) モデルに展開されるのでした。

テストの理論も心理学の研究も，目に見えない「学力」や「性格」といったものを測定するという意味で，ツールとしては同じものを使うわけです。一方ではテスト，他方では質問紙とか尺度と呼ばれますが，狙いは回答者の反応パターンから潜在的な性質を見出そうとするものです。ここで，改めてテストの理論に戻りたいと思います。ただしこれまで古典的テスト理論と呼ばれていたものは，その名の通り古典的であって，現代的なテスト理論はどうなっているのか，というところを考えてみたいと思います。

現代的なテスト理論，新しいテスト理論ともいわれますが，それは**項目反応理論 (Item Response Theory)**，あるいは項目応答理論とよばれます。略して **IRT** と表現されることも多いですね。この理論はいわゆる「学力テスト」などの要請から展開してきたものです。心理学的なアプローチからは，従属変数が連続的であったり[*1]，心理的な構造が知りたいために多因子であったりするのが，自然な発想でした。これに対して学力テストのようなものは，各項目の結果が「正答/誤答」の二種類しかありません[*2]。数値としては 1/0 の二値，バイナリデータであり，尺度水準は名義になります。また，測定したいものは一因子です。国語のテストは国語の能力を，算数のテストは算数の能力を測定するべきだと言えるからです。このように，項目反応理論は因子分析の特殊系だということができます。

受講生のみなさんは心理学での応用例の方が興味があるかと思いますが，後ほどこの項目反応理論のモデルが展開し，再び因子分析モデルに戻ってきますのでお楽しみに。それまではひとまず，テストの項目を分析するというのはどのようなことがなされているのかをみていきたいと思います。

4.2　通過率と累積正規分布

みなさんは大学入学共通テスト (旧センター試験，さらにその前は共通一次試験と言いました) や，学内の定期テスト，模試など色々なシーンでテストを受けてきたことと思います。模試などでは偏差値が明らかになり，自分の実力が相対的にどのあたりにあるのかがわかるようになっていたかと思います。大学共通テストなどは 50 万人ぐらいが一度に受験しますから，さまざまな学力の人がそこには含まれるのですが，成績を図に

[*1] 因子分析モデルの式 3.1 が Z_{ij} から始まっていたことを思い出してください。標準化されているということは，平均や標準偏差が求められているということであり，**間隔尺度水準**以上の数字が前提とされています。

[*2] 部分点というのがあるじゃないか，と思うかもしれませんが，それはひとまず横に置いてください。

するとてても綺麗な正規分布になることが知られています[*3]。正規分布は誤差の分布でもありますが，多くの要因が考えられる際の集積的データも，自然とこの形になることがわかります。

さて，学力のような潜在変数が標準正規分布に従うと仮定しましょう。この分布の形はどこの確率点がどれぐらいの確率密度を持っているか，あるいはある確率点以上・以下の面積が全体の何 % かを表すものですが，縦軸をある点以下の累積確率に書き直してみましょう (図 4.1)。図 4.1 の上の図がいわゆる正規分布の分布の形，確率密度関数です。下の図はこれを累積確率に書き換えたものになっています。累積確率は 0% から始まって，最終的に 100% にまでどのように増えていくかを示した図になります。

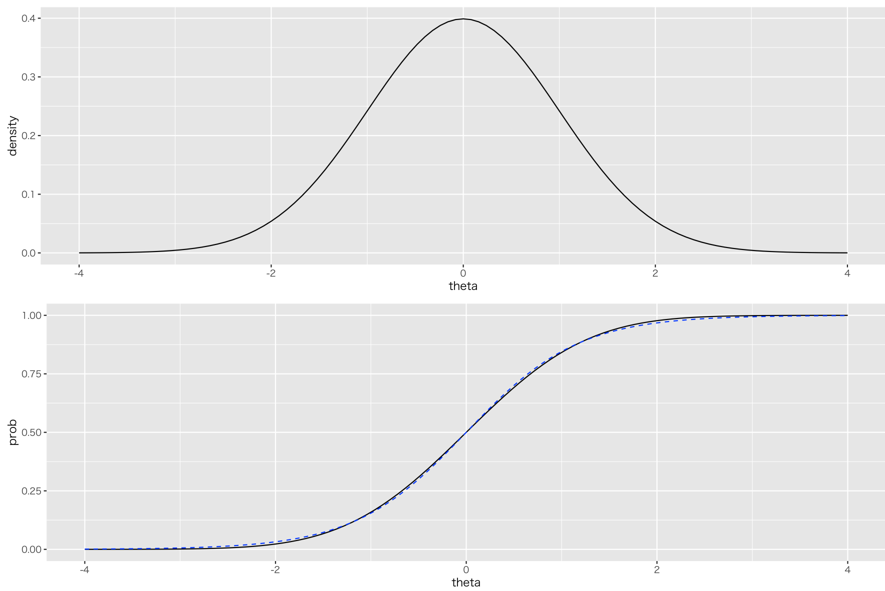


図 4.1　正規分布の確率密度関数 (上) と累積確率関数 (下)

累積正規確率は，テスト理論と密接な関係があります。というのも，学力が正規分布すると考えるなら，累積正規分布の形はあるテスト項目の**通過率 (pass ratio)** と同じ形になると考えられるからです。

通過率とは，あるテスト項目に正答する人の割合のことです。ここで複数の項目からなる，あるテストをしたとしましょう。正答数を数えてその人の成績とすると，よくできたテストであれば成績は正規分布に従います。さらに，ある項目と成績との相関 (**IT 相関 (Item-Total correlations)**) は高いはずですね。つまりその項目に正答することが，テスト全体の成績と高く関係しているので，その項目はテスト全体が測ろうとしているものを反映していると考えられるからです。また，成績をもとに被験者集団を 5 群に分けたとしましょう。「成績上位群 (HH)」，「成績やや上位 (MH)」，「成績中程度 (M)」，「成績やや下位 (ML)」，「成績下位 (LL)」です。このとき，各群の平均通過率を考えると，図 4.2 左上図のようになるのが理想的です。つまり，成績が高い人たちの通過率は高く，成績が低い人達の通過率は低くなるはずですね。同じ図の右上は，LL 群でも半分ぐらいが通過し，その後の群は過半数，ほとんどが通過するようになっています。これは，この項目が簡単すぎたことを意味しています。簡単すぎる問題は，それはそれで被験者の特徴が弁別できないという意味で悪い項目です。逆に図の左下にあるのは，HH 群でも半分以下の通過率しかありません。つまり難しすぎる問題です。ほとんどの人が間違えてしまうわけですから，これも良い試験問題とは言えないでしょう。右下に至っては逆転していて，どうやったらこういう項目が作れるのか却ってわからないほどですが，学力の低い

[*3] 山内 (2010) の見返し (表紙を開いた最初の内側のページ) に，センター試験の成績分布が載っており，綺麗な正規分布であることが示されています。

人だけが正答できて，学力の高い人は正答しない項目ということになります。もちろんこんな項目はよくないわけで，IT 相関で負の相関が出ているわけですから，テストの文脈でいうなら「そのテストで測っていない何か別の能力を測っている」と考えるしかありません。

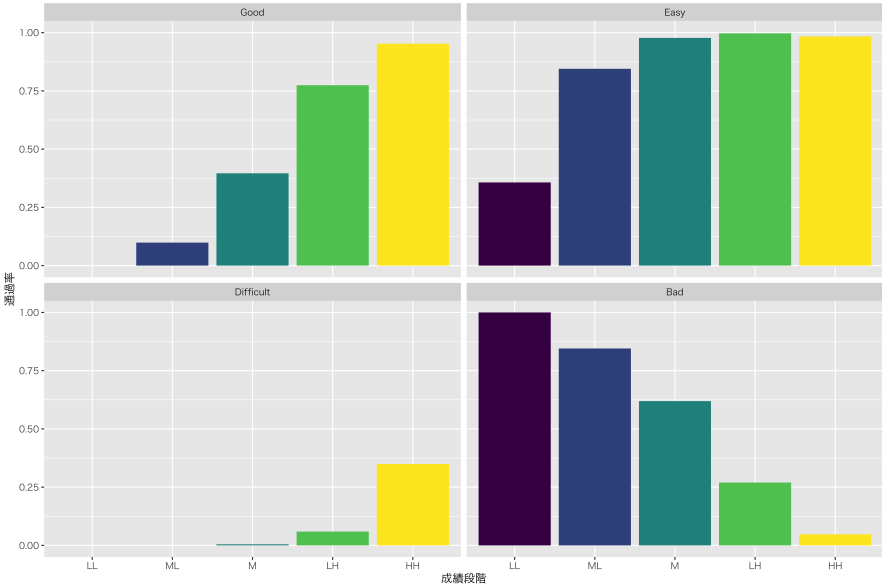


図 4.2　群ごとの平均通過率。左上が良いパターン。右上は簡単すぎる，左下は難しすぎる項目。右下は逆転していて良くない項目。

ともあれ，このようなやり方で項目の良し悪しを見ていくことができます。また，図 4.2 は 5 段階ですが，7 段階，9 段階とよりきめ細かくしていくと，理想的な形は累積正規分布により近づいていきます。新しいテスト理論による項目分析はこの累積正規分布の形を基本とし，これを拡張することで各項目の特徴を描いていくことになります。

ところで 1 つ前の図 4.1 の下の図には，実線と点線の 2 つの線が絡んでいることにお気づきでしょうか。実線の方は確率分布関数から累積確率を出して描いたものですが[*4]，点線のほうは次の関数を使って描いています[*5]。

$$f(x) = \frac{1}{1 + exp(-1.7x)}$$

この関数，図から明らかなように累積正規分布とほとんど同じですよね。累積正規分布の関数を直接使うと，積分計算 ($\int$ を使うやつ) が入ってくるのでちょっと計算が面倒ですから，こちらの関数の方を近似関数として用います。この関数のことを**ロジスティック関数 (logistic function)** と言います。ロジスティック関数そのものは，先の式から 1.7 という係数を除いた $\frac{1}{1 + exp(-x)}$ で表されるもので，$-\infty$ から $+\infty$ までのどんな数字が与えられても，答えを 0 から 1 の範囲に変換してしまう関数です。この特徴はとても便利です。というのも，結果が 0 と 1 の間に入るということは，比率を表していると考えられるからです。0/1 のバイナリデータが従属変数のときに，独立変数をこのロジスティック関数で変換してやれば 0 か 1 のどちらに近いか，どれぐらいの比率で 1 の目が出るかがわかります。項目反応理論も結果が 0/1(誤答/正答) ですから，

[*4] R では pnorm 関数を使って描きます。数式でいうなら，$\int_{-\infty}^{p} \frac{1}{\sqrt{2\pi}} exp(-\frac{x^2}{2})$ となります。
[*5] exp というのは指数関数で，$exp(x) = e^x$ のことです。ここで e は数学定数で，$e = 2.718282...$ という数字です。

「学力」のような目に見えない能力をロジスティック変換してやれば，結果が正答率になるというのは大変便利なのですね。

それではこのロジスティック関数をつかった項目分析の話に進んでいきましょう。

4.3 項目母数の特徴

ロジスティック曲線が累積正規分布の近似関数になっていること，テスト項目の分析には通過率を使って考えることを見てきました。とくに通過率の分析 (図 4.2) では，その項目が難しい設問だったのか，簡単なものだったのかを見ることができました。ロジスティック曲線もこの「項目の難しさ」を表現できるように，次のように拡張できます。

$$p(\theta) = \frac{1}{1 + exp(-1.7(\theta - b))}$$

左辺の $p(\theta)$ に含まれる θ は，潜在変数のスコア，**因子得点**であり，ここでは標準化された学力ですから，偏差値のようなものだと思ってください[*6]。$p(\theta)$ は能力 θ の人がこの項目に正答する確率=(通過率) です。

ここで b という変数が入ってきました。これが**困難度 (difficulty)** を表す指標です。$b = 0$ のときは標準正規分布と同じ形になりますが，$b = 1$ ならばこの式は右に，$b = -1$ ならば左に動きます。つまり $b > 0$ であれば難しく，$b < 0$ であれば簡単であることを表現していることになります。図 4.3 に困難度が $b = -1, 0, +1$ の時の曲線を書いてみましたので確認してください。このように困難度を表現するパラ

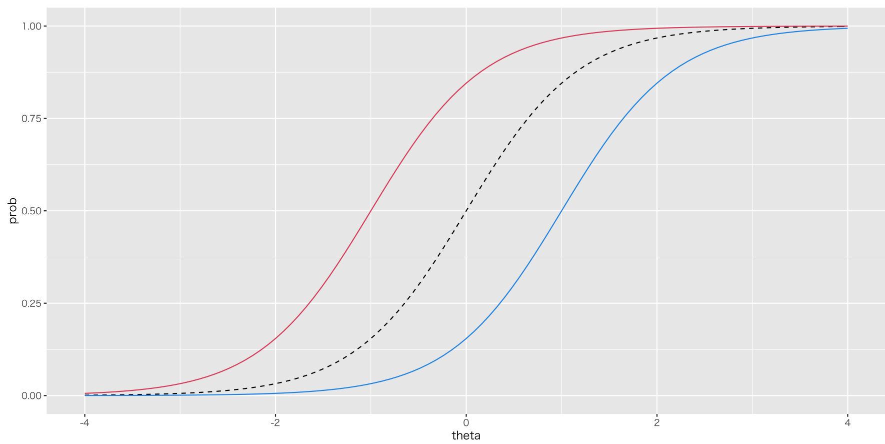


図 4.3　困難度母数の入ったロジスティック曲線。1PL ロジスティックモデルともいう。点線が $b = 0$ の標準的曲線。赤が $b = -1$，青が $b = +1$ の例

メータを追加したモデルを **1 パラメータ・ロジスティックモデル (One Parameter Logistic model)** と言います。実際のテストの回答パターンにたいし，各項目にこの曲線を当てはめて困難度を推定することで，項目を評価できるようになります。このように項目の特徴を描く曲線のことを**項目特性曲線 (Item Characteristic Curve, ICC)** といいます。

さらにパラメータを追加して，次のようにすると **2 パラメータ・ロジスティックモデル (Two Parameter Logistic model)** になります。

$$p(\theta) = \frac{1}{1 + exp(-1.7a(\theta - b))}$$

[*6] 偏差値は標準化スコア z_i を $10z_i + 50$ と変換したものを指します。ここはその変換前の z_i と同じです。

ここで a という母数 (パラメータ) が入ってきました。これは $a = 1$ だともともとのモデルのままなのですが，これが小さくなると曲線が傾き，大きくなると曲線のカーブが強くなります (図 4.4)。曲線が緩やかになると (図 4.3 の赤線)，θ の違いに対して通過率の変化が乏しくなります。言い換えると感度が悪くなるわけです。逆に曲線の立ち上がりが強くなると (図 4.4 の青線)，θ がある一定のところを超えるかどうかで正答率がグッと変化することになります。つまりこのパラメータは，回答者の能力 θ を分類する力の強さを表しているのです。このパラメータのことをとくに**識別力 (discriminant)** と呼びます。

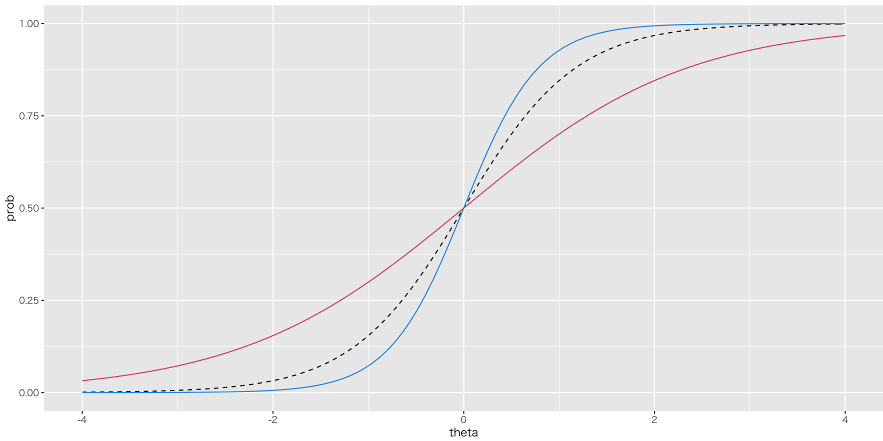


図 4.4　識別力母数の入ったロジスティック曲線。2PL ロジスティックモデルともいう。点線が $a = 1$ の標準的曲線。赤が $a = 0.5$，青が $a = 1.5$ の例

一般にはここまで紹介した 2PL モデルがよく用いられます。他にも 3，4，5 つとパラメータが増えたモデルもありますが[*7]，もとのロジスティック曲線に特徴を付け足していったもので，曲線をずらしたり，曲げたりしながらもとのデータにうまく当てはまるようにしつつ，その特徴を解釈できるように工夫しています。

いずれにせよ，テストの結果から各項目の特徴を記述します。少し例示したほうがわかりやすいでしょう。図 4.5 には心理学データ解析基礎で行った試験の結果から，2PL ロジスティックモデルを当てはめて項目分析をした例です[*8]。同じデータの項目母数を表 4.1 にも示しました。表 4.1 の数字と図 4.5 の曲線の対応をよく確認してください。

たとえば，項目 `I0022` は困難度が-1.92，識別力が 1.30 です。困難度がマイナスですので，これはかなり簡単な問題だということになります。具体的には，偏差値 50 すなわち $\theta = 0$ の人であっても 98.58% 正解するわけですから，ほとんどの人にとって容易い問題であることがわかります。ちなみにこの問題，具体的には帰無仮説検定に関する問いで，「「差がない」「偏りがない」といった仮説は何と呼ばれるか」というものでした[*9]。

一方，困難度が 0 近いところの例として項目 `M0605` をあげますが，これが平均的な難易度の質問になっています。困難度母数 $b = 0.0$ であれば偏差値 50，すなわち $\theta = 0$ の人が正答する確率が 50% の質問ということになりますが，今回の `M0605` はそれより少し難しいので，偏差値 50 の人で 20.70% の割合で正答できます。この問いについて偏差値が $70(\theta = 2)$ であれば，92.96% の確率で正答できることになりますし，偏

[*7] 詳しくは http://antlers.rd.dnc.ac.jp/~shojima/exmk/jindex.htm を参照してください。3 つめのパラメータはあて推量母数，4 つ目は上方漸近母数，5 つ目は非対称母数と呼ばれています。

[*8] 心理学データ解析基礎の授業では，過去の受講生のデータとさまざまな質問をプールしたデータを貯めてあります。みなさんが受けたテストには入っていない項目かもしれませんが，これまでどこかで出題され，回答パターンが得られている実際のテストです。

[*9] いうまでもないですが，答えは「帰無仮説」です。

差値 30($\theta = -2$) であれば 0.51%，つまりほとんど正答は望めないということになります。ちなみにこの問題は「重回帰分析において，標準化されたデータを使って分析をすることで，モデルの適合度を上げることができる」を Yes か No かで判断させるという質問でした。

他にも項目 I0017 は困難度が 2.17，識別力は 0.69 です。困難度が最も高いグラフで，図の曲線は一番右にあるラインになっています。ただ識別力がやや低いので，曲線はよりなだらかになっていますね。困難度が高いので，$\theta = 0$ でも 7.24% しか正答できません。難しい！ $\theta = 2$ で 45.10% ですから，かなり能力の高い人でも半分は間違えるような問題です。ちなみにこれは標本分散の期待値が母分散からどれぐらいずれるのかを計算する問題でした。

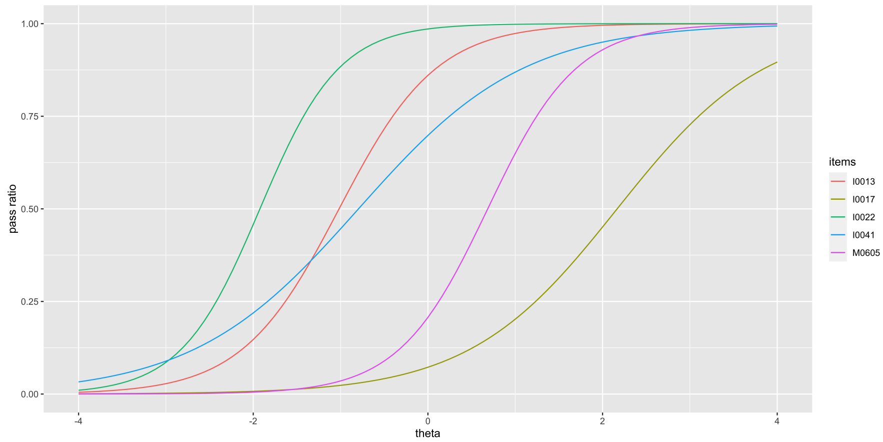


図 4.5　実際のテストに 2PL モデルを適用した例

表 4.1　各項目の困難度と識別力

	項目 ID	困難度	識別力
1	I0013	-1.02	1.05
2	I0017	2.17	0.69
3	I0022	-1.92	1.30
4	I0041	-0.79	0.62
5	M0605	0.68	1.15

4.4 被験者母数の推定

項目反応理論における因子得点の推定は，項目の特徴を表す項目母数に対して**被験者母数**と呼ばれ，上述の項目特性に基づいて行われます。先ほどの図 4.5 をもとに説明します。ある回答者が項目 I0013 に正答したとしましょう。その人の能力値 (因子得点,θ) はどの辺りにあるかといえば，確率の曲線に沿った下の領域のどこかということになります (図 4.6)。この曲線，ICC は項目の特徴を表したもので，ある θ の値の能力があればどの程度の確率で正答できるかを表した項目の特徴でもありますが，逆にある人の θ がどのあたりにありそうかを示しているともいえます。たとえばこの ICC において，θ が 0.5 のときの通過率は 86.05% ですが，言い換えればこの項目に正解した人が $\theta = 0.5$ である可能性も高そうです。$\theta = -2$ の通過率は 14.71% ぐらいですから，能力がこんなに低いとは思えませんし，1 つの項目の話でしかないですが，希望的

観測をするなら θ がもっと高い可能性もあるでしょう[*10]。

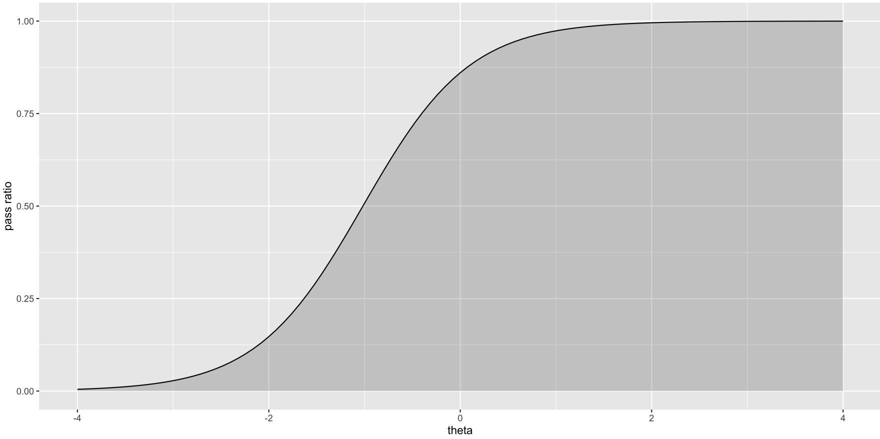


図 4.6 実際のテスト項目 I0013 に正答した人の能力がありそうな領域

次に，困難度のより高い項目である I0017 には**誤答**したとしましょう。その人は，I0017 の ICC の下の領域には**ない**はずです。項目 I0013 の ICC の下で，かつ，項目 I0017 の ICC の上にあるはずなので，図 4.7 のように塗りつぶされた領域の中に入ります。

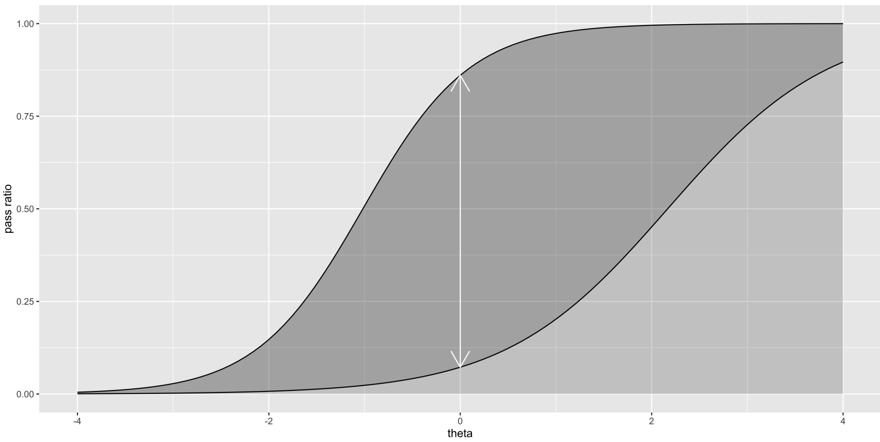


図 4.7 つぎのテスト項目 I0013 には誤答した人の能力がありそうな領域

この 2 つの曲線の間にある，濃く彩られた領域の幅が，回答者の能力値がありそうな程度を表しているのです。図 4.7 には $\theta = 0$ の可能性の大きさを矢印で示してありますが，θ の値はここだけに限らずこの曲線の幅のどこかです。ただ $\theta = 2$ や $\theta = -2$ よりは $\theta = 0$ のほうが，より「ありそう」な値だということがわかります。

ここでさらに同じ人に，項目 M0605 の質問をして，この人がそれにも正解したとしましょう。この人の能力 θ のありそうな範囲はさらに絞り込むことができます (図 4.8)。項目 I0013 よりは難しい質問に正解したわけですから，$\theta = 0$ の可能性はグッと小さくなり，それよりも $\theta = 2$ ぐらいの方がありそうだ，ということになっ

[*10] θ のありそうな「確率」とは言ってないことに注意してください。すぐにわかることですが，この ICC の下の面積を積分しても 1.0 にはなりませんのでこれは確率ではなく，**尤度 (likelihood)** のほうなのです！

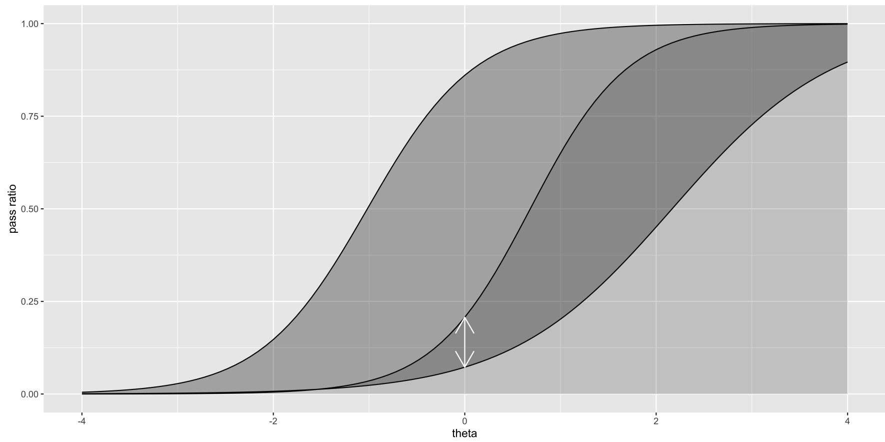


図 4.8　さらにテスト項目 M0605 には正答した人の能力がありそうな領域

てきます。このように，１つ１つの項目からこの人の能力 θ がどのあたりにありそうか，というのを絞っていき，テストに含まれる項目全部を使えば，かなり狭い領域で「この辺りにあるはずだ」と推定できるでしょう。

この IRT を用いた推定方法は，このようにテストの項目ごとにその特徴を分析できるのが特徴です。テストの中でも良い項目，悪い項目というのはあるでしょうが，どのように悪いのかを困難度や識別力といった項目の特徴を使って表現できます。これらの数字は，テストに含まれる項目ごとの難易度を相対的に比較していく中で作られるものであり，回答者の能力に依存するものではありません。テスト理論が被験者の特徴と項目の特徴を分離するところから始まったことを，改めて思い出してください。

また，このような項目同士の比較から求められた項目の特徴をつかって，被験者の能力 (因子得点) を推定する方法についても紹介しました。その過程の中で気づいたと思いますが，複数の回答を通じてある回答者の能力 θ のありそうな領域が狭められていく中で，明らかにその人にとって簡単すぎる問題，難しすぎる問題は意味を成しません。たとえば図 4.8 の段階まで絞り込まれているときに，項目 I0013 よりも簡単な I0022 を出題しても，おそらくほぼ確実に正答し，そのことは「領域を狭める」ことにはなんの貢献もしないでしょう。回答者の能力に相応しい質問を選んで出すことができれば，とても効率よくその領域を絞り込んでいくことができます。こうした考え方は，次回お話しする**コンピュータ適応型テスト (Computer Adapted Test)** として実装されます。

今回はテスト理論について紹介してきましたが，この考え方は性格テストなどで用いられているリッカート形式の尺度に応用できるように，発展していきます。次回はその辺りを解説していこうと思います。

4.5　課題

■テスト理論と因子分析　項目反応理論のロジスティックモデルについて，項目母数が何を意味しているか，項目母数が変わると ICC がどのように変化するかを自分で説明できるようになろう。

■項目反応理論の利点　項目反応理論を用いた採点方法を使うと，どういう長短所があるだろうか。次回の内容に先駆けて，自分なりに考えてみよう。

第 5 章

現代テスト理論その 2

5.1 現代テスト理論の特徴

前回は現代テスト理論として項目反応理論 (IRT) を紹介しました。ロジスティック曲線を応用して項目の特徴を描画し，それを使って被験者母数を推定する方法についても解説しました。この一連の手続きに基づき，現代テスト理論の利点を考えてみたいと思います。

5.1.1 現代テスト理論の利点 1: 項目母数と被験者母数の分離

古典的テスト理論からの発展として，現代テスト理論では被験者母数 θ_i と項目母数 a_j, b_j を区分して考えるようになりました。項目母数は通過率のアイデアを精緻にしたものですが，この通過率は項目群の総和を元に考えられていたことを思い出してください。すなわち，項目母数の計算には項目の相対的な困難度だけを用いています。イメージとしては鉱物の硬度検査のようなものです。2 つの異なる硬さの物をぶつけて崩れた方が負け＝より硬度が低いと考えるように，2 つの異なる項目を被験者に与えて，より正答者数が少ない方がより難しいと考えるのです。これはつまり，回答者の学力が高かろうが低かろうが，困難度が $b_x < b_y$ であるという関係に違いはないという考え方です。

これまでのテストや心理尺度の作り方に比べると，この点が大きく違います。学校などのテストは教員が作っていますから，教員が自分の感覚で「こちらの方がより発展的な内容だ」「こちらの方が難しいだろう」という問いに大きな配点がなされたりするでしょう。その後テストの平均点をみて「今回のテストは簡単にしすぎたか」という判断をしたりするでしょう。しかし IRT では項目それ自体に困難度を決めさせますから，そこに作成者や回答者の意図は含まれません。平均正答率が高いからと言って簡単な問題なのではなく，項目の特徴として困難度が決まるのです。

たとえばサーストン尺度の作り方を思い出してください (セクション 2.2,Pp.27 参照)。サーストン尺度では尺度適用前に評定者集団によって尺度値を決めます。この評定者集団が偏った思想の持ち主だけで固められていた場合，尺度の点数は極端なものになり，普通の人がその尺度に回答すると極めて低い点数，高い点数になってしまうかもしれません。あるいはリッカート尺度の作り方を思い出してください。(セクション 2.3,Pp.28 参照)。リッカート尺度では回答者の累積度数から尺度値を算出します。先ほど同様，回答者集団の態度に偏りがあれば，尺度の点数は標準的な物ではなくなるでしょう。つまり「誰を対象に測定するか」によって目盛りが変わるようなものです。これでは測定結果の一般化は難しいでしょう。たとえば本学で作られた尺度を，他大学でやってみると違った尺度値になるのですから，研究結果はせいぜい「その大学ではそうなんだろう」となり一般化できなくなります。従来の方法は，回答者と項目の特徴が関連しすぎていたのです。

これに対し，IRT を使って項目の特徴を計算する場合は，相対的な難易度に違いはありませんから，どの

大学で作った尺度であっても統一的な解釈が可能です。尺度作成時に幅広くデータを集め，項目母数を確定させてしまえばどこででも統一的な評価ができます。テストなど学力を測定する際に大学間での違いが見られたとしても，その難易度を調整するのも簡単で，共通する項目を入れておけばそこを基準に相対的な困難度調整ができます[*1]。良問と悪問の評価と，回答者の評価を分けることは重要なポイントなのです。

5.1.2 現代テスト理論の利点 2: 被験者母数の推定の利点

被験者母数と項目母数の分離は，さらに別の利点も生み出します。それはデータが一部欠落した場合の補完に関係します。

リッカート尺度では回答者全体の相対頻度から，カテゴリの尺度値を決定するのでした。ここである人が特定の項目にだけ回答をし忘れたとします (調査研究ではよくあることで，同じような目盛りが並んでいると一行飛ばして丸をつけちゃうようなことはよくあります)。そうすると，その項目だけ合計人数が変わりますから，計算が面倒です。また相関係数を計算する時にも，その人のその項目については計算できなくなります。因子分析は相関係数から計算を始めますから，一箇所でも欠損値があるとその人のデータを抜いてしまうか[*2]，他の値を代入して補完するか[*3]，手間でも計算の時にそこだけ外して計算するか[*4]といった工夫が必要です。因子分析結果に大きな違いは出なくても，その人の因子得点は計算できないことに違いはありません。

それに対して，IRT の被験者母数の推定は，項目母数をつかった ICC をもとに一人ずつ絞り込んでいくというものでした。もしある人が回答していないことがあっても，計算ができなくなることはありません。その項目の情報が得られないので絞り込み精度は上がりませんが，ヒントが減っただけで回答できないわけではないのです。このように，得られた情報すべてを使ってその人の被験者母数 (因子得点) を推定する方法のことを，**完全情報最尤推定 (Full Information Maximum Likelihood)** と言います。このように IRT を使うと，必ずしも全問に回答していなくてもスコアは計算できるということになります。欠損値があるからその人のデータは使えない，ということがないのでいいですね！

もっというと，IRT では全員が全員同じ問題に回答する必要はありません。たとえば能力値が $\theta_i = -0.3$ ぐらいにありそうだ，と絞り込めている段階で，次の問題の困難度母数が $b_j = 2.5$ であれば，おそらくほぼ確実にその人は回答できないでしょう[*5]。その人にいくら困難度母数の高い質問を繰り返しても，ほとんど誤答がつづくだけで，とくにその人の θ がありそうな領域を狭めるヒントにはなりません。むしろ $b_j = -1$ とか $b_j = -0.5$ のような簡単な問題を出して[*6]，それらに正答できるかどうかを見極め，絞り込んで行った方が効率的です。紙に印刷されているテスト (Paper Based Test) であれば，印刷された問題は変えようがありませんから，困難度順に問題を並べると，あるところから先はずーっと不正解が続く人が続出します。ずーっと不正解なところの問題はいくら良問でも，その人の能力を測るのには役立ちません。ヒントが増えないからです。であれば，回答者の学力に合った問題を，その都度その都度出題した方がいいですよね。コンピュータを使って回答者に相応しい質問をダイナミックに組み替える，コンピュータに基づいたテスト (Computer Based Test)，別名**コンピュータ適応型テスト (Computer Adaptive Test)** というのがそれです。CAT になると，回答者ごとに出題が変わりますからカンニング対策の必要も無くなって，とても便利

*1 こうしたテスト間の困難度調整のことをテストの**等価 (equation)** といいます。

*2 リストワイズ削除といいます。

*3 欠損値補完については，平均値や中央値を代入したり，同じようなパターンで回答している人の値を使い回したり，回帰分析でその項目の値の推定値を入れたり，とさまざまな方法が考えられてきました。欠損値発生のメカニズムにもよりますが，いずれもある程度バイアスのかかった値になってしまいます。統計的によりバイアスの少ない適切な代入法が考えられてはいますが，そもそも欠損値がないのが最も望ましいことに変わりはありません。

*4 ペアワイズ削除といいます。

*5 2PL モデルで $a = 1, b = 2.5$ のとき，$\theta = -0.3$ が正解する確率は 0.8492% です。

*6 2PL モデルで $a = 1, b = -1$ のとき，$\theta = -0.3$ が正解する確率は 76.674%，$b = -0.5$ のであれば 58.419% です。

になること間違いなしです。

5.1.3 現代テスト理論の利点 3: 信頼性についての考え方

IRT の考え方からいうと，どんな項目でも何らかの情報を提供してくれるはずです。たとえば $b_j = 3$ のようなとても難しい項目が合ったとします。これがどれぐらい難しいかというと，偏差値 70 の人でも 15% しか正解できないレベルです。ほとんどの人にとっては誤答にしかならない難しすぎる悪問だ，と言いたくなるかもしれませんが，学力が高い人がどれほど高いレベルでやれるのかを検証するためには必要な問題です。偏差値が 70 なのか，75 なのか，80 までいけるのか，といったことを見極めるためにはこの問題でないと情報が得られないことになります。逆に簡単すぎる問題であっても，より低いレベルを精緻に検証するためには必要なものなのです。

つまり，どの項目にもその項目が得意とする領域があるはずです。この項目はこの領域の回答者を絞り込む時に，最も有用な情報をもたらしてくれるはず，という θ の場所があるはずなのですね。これを表現するのが**項目情報曲線 (Item Information Curve)** といい，次の式で表される項目情報関数で描くことができます。

$$I(\theta) = a_j^2 p_j(\theta) q_j(\theta)$$

ここで $p_j(\theta)$ はその項目 j の θ における正答率 (通過率)，$q_j(\theta)$ は誤答率を表しています。能力が平均的，すなわち $\theta = 0$ のときは，0.5×0.5 になる平均的な困難度 $(b = 0)$ の設問が最も大きな値になる，というわけですね。図 5.1 にいくつかの ICC とそれに対応する IIC を描きましたので，確認してください。IIC の

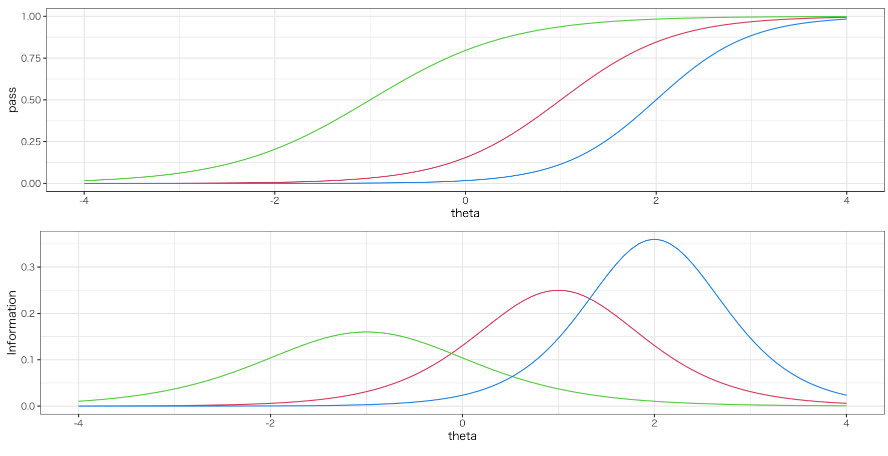


図 5.1　項目特徴曲線 (ICC, 上図) と項目情報曲線 (IIC, 下図) の例。左から順に $a_1 = 0.8, b_1 = -1$ の識別力が弱く簡単な項目，$a_2 = 1, b = 1$ のやや困難な項目，$a_3 = 1.2, b_3 = 2$ の識別力が高く困難な項目。

ピークは，対応する ICC の $\theta = 0.5$ のところにあること，識別力はピークの尖り具合に関わっていることを確認してください。

さて，IIC はその項目がどこで情報をもたらしてくれるか，ということを表しているのでした。言い方を変えると，IIC のピークはその項目の最も信頼できるところであるとも言えます。つまり IRT において**信頼性は潜在特性の関数**になっているのです。古典的テスト理論ではテスト全体の分散に占める真分散の割合のことを信頼性というのでした。因子分析理論では項目の中の共通因子負荷量の二乗和，共通性がその項目の信

頼性を表しているのでした。信頼性を見る水準がテスト全体から項目別の評価に発展したわけですが，IRT ではさらにその項目の最も感度の良いところを探る関数として，その信頼性を評価できるようになったといえるでしょう。

また IIC はある項目から得られる情報のことを意味しますが，テストに含まれているすべての情報関数を足し合わせることで，そのテストから得られる情報の大きさを関数として評価できます。すなわちテスト全体の情報量 $I_T(\theta)$ は，$I_T(\theta) = \sum_{j=1}^{M} I_j(\theta)$ であり，この関数のことを**テスト情報関数 (Test Infromation Curve)** といいます。図 5.1 の 3 つの項目からなる TIC を示したのが図 5.2 です。これを見ると，この 3 つ

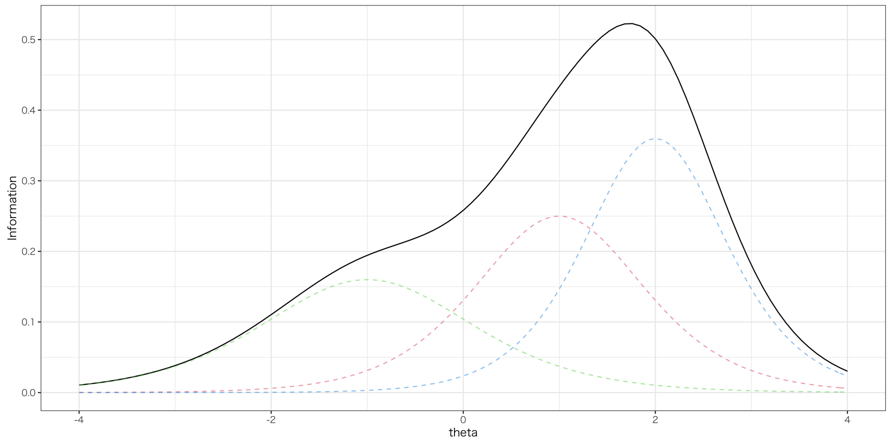


図 5.2　テスト情報関数 (黒の実線部)。理解を進めるために各 IIC を薄い点線で表現した。

の項目からなるテストは $\theta = 2$ より少し低いレベルを測定するときに最も鋭敏に働くということがわかります。このように，項目母数がわかっていれば事前にテストの特徴をどのあたりに持ってくるかを決定でき，自由自在にテストをデザインできるようになるわけです。

テストの例で話をしていますが，心理学的な領域でももちろん便利な手法です。たとえば高い認知能力レベルの人をとくに選出したいとか，精神的な健康度がごく低い人をしっかりと検出したいといった目的があれば，そのあたりにピークが来るような項目からなる質問項目からなる調査票を構成すれば良いのです[*7]。

5.1.4　現代テスト理論の問題点

ここまで見てきたように，IRT はさまざまな利点があります。しかし欠点がないわけではありません。

ここまでの話はすべて，項目母数が既にわかっていれば，という前提付きで進めてきました。では項目母数はどのようにして定めるのでしょう？ これはもちろん得られたデータから算出できるのですが，そのためには事前に多くの被験者から回答を集め，項目母数の値はほぼ間違いなくこれぐらいだろう，と言えるほど安定したものである必要があります。テストの場合，回答が 0/1 というバイナリデータで得られますから，そもそもそれほど情報がある反応ではありません。バイナリデータからその項目の特徴を安定して推定するためには，かなり多くの被験者 (数千から数万単位) を集めて項目に回答させておく必要があります。テスト項目は一度使ってみないと，項目母数がどうなるかわからないというのもポイントです。

また，CAT など項目をダイナミックに組み合わせるためには，選べるぐらいさまざまな項目を準備しておか

*7 具体例として小杉 (2014) をあげておきます。学校適応感を測定するため，テストのピークがやや低いところに来るようになっています。

なければなりません。項目を集めたものを**項目プール (Pool of Items)** といいますが，これも数千から数万の単位で用意しておく必要があります。なぜなら，テスト項目は事前に 1 回は使ってあるわけですから，数えるほどしか項目がなければ受験生が正答を事前に丸暗記できてしまうからです。もちろん項目プールが数千から数万あっても一度どこかで使われていますから，過去問をすべて完全に丸暗記すればその人は満点が取れてしまいます。もっともそれだけのものを覚えられるのは，ある意味学力が高いといっても差し支えないと思いますが。

日本でおこなわれる大学入学共通試験をはじめ，試験問題というのは極めて厳重な管理下に置かれ，事前にその情報が漏れることは公平性の観点から言って不適切であるとされています。しかし IRT で分析するためには，事前に項目の特徴を知っていなければならないのです。CAT をつかって入学試験などができれば，カンニング対策にもなりますし，受験生は何度でもチャレンジできるので利点も多いのですが，「公平性のために新しいテストでなければならない」となるとなかなか実用化できないところがあるというのも事実です[*8]。

ところで，心理学の場合は学力テストのように正答・誤答があるものではなく，「当てはまる」から「当てはまらない」といった軸上で多段階の反応を求めることが一般的です。テスト理論を多段階のモデルに応用できるのかと言えば，幸いにしてその答えは YES です。

5.2 段階反応モデル

リッカート法などで作られる尺度は，一般に 5，7 段階のものが多くあります。少ないものでは 3 件法[*9]であったり，ものによっては 4，6 件法であることもあります[*10]。しかしこれらの段階はいずれも順序尺度水準の情報しか持っておらず，そのままでは尺度値として使うことができません。シグマ法などで数値化すれば良いのですが，その手間を省いて分析する悪い習慣もあることは既に指摘した通りです。

IRT の多段階版はこうした問題に対応できる方法です。IRT の多段階モデルは大きくわけて 2 つあり，1 つは**段階反応モデル (Graded Response Model; GRM)**(Samejima, 1997)，もうひとつは**多段採点モデル (Partial Credit Model)**(Muraki, 1992) と呼ばれています。どちらも発想は似たようなところがあり，ここでは GRM について解説します。興味がある人は，豊田 (2012) や加藤・山田・川端 (2014) など専門書を参考にしてください。

GRM の考え方の基本は，段階反応の背後には正規分布する連続的な潜在特性 θ がある，と仮定するところです。心理的な能力，学力，性質などは連続的なのですが，それが表に出てくる時は離散的 (順序的) だというわけです。図 5.3 に図示されているように，正規分布がある**閾値 (threshold)**(これを b_k と表しますが) を超えると出現する時は次のカテゴリになる，ということを考えます。図は三段階の例ですが，図から明らかなように k 段階であれば閾値の数は $k-1$ 個あることになります。横軸 θ は心理的な態度や性質の強さだと思ってください。さてそうすると，$\theta = 2.0$ ぐらいであれば，ほぼ間違いなく「当てはまる」に回答することになりますし，$\theta = -2$ であれば「当てはまらない」に回答するようになるはずです。このように θ が大きくなればなるほど最後のカテゴリに反応する確率は上がっていきますから，ここは 2PL モデルの時のようにロジスティック曲線で「当てはまる」に回答する確率を表現できるでしょう。問題は，それより下の段階に反応する確率をどのように表現するか，です。

[*8] 令和 2 年度に大学入試センター試験から大学入学共通試験に変わりましたが，改革前の計画では CAT 化することが盛り込まれていました。しかし実際には，受験生のためのコンピュータやタブレットを準備したり，安定した通信網が必要であったり，というハード的な問題もあって見送られてしまいました。

[*9] たとえば YG 性格検査は 3 段階です。

[*10] 偶数の段階にすることで，必ずどちらかの極に寄るようにして集計できます。日本人は「どちらでもない」に回答しがちな中点集中傾向があるとも言われているので，わざと肯定・否定のどちらかに寄せようという考え方です。

ここである段階に反応する確率を考えるために，少し表現を改めます。すなわち，個人 i の項目 j に対する反応が，カテゴリ k **以上**になる確率として，$P^{+}_{jk}(\theta) = P(x_{ij} \geq k|\theta)$ をまず考えます。ここで $k = 0, 1, 2, ..., K$ とします。先ほど示したように，$k = K$，すなわち一番上のカテゴリ (ここでは「当てはまる」) に回答する確率は，2PL ロジスティック関数と同じ形をしていますから，次のように表現できます。

$$P_{jK}(\theta) = P^{+}_{jK}(\theta) = \frac{1}{1 + exp(-a_j(\theta - b_{jK}))}$$

ここで右辺の a_j は**識別力**，b_{jK} はカテゴリ K の**困難度**を表しています。左辺の $P_{jK}(\theta)$ は θ の人が項目 j のカテゴリ K に反応する確率で，それは k 以上に反応する確率 $P^{+}_{jk}(\theta)$ と一致していることを表しています。

次に，もっとも低い段階に回答する確率を考えましょう。これは θ が大きくなればなるほど減っていくはずで，いわばロジスティック曲線の逆のような形をするはずです。反応確率は最大でも 1.0 ですから，逆ということは 1.0 から引いてやればよいでしょう。

$$P^{+}_{j0}(\theta) = 1.0 - \frac{1}{1 + exp(-a_j(\theta - b_{j0}))}$$

問題は「どちらでもない」に反応する確率です。これは引き算で考えることができます。すなわち「どちらでもない」以上に反応する確率から，「当てはまる」以上に反応する確率を引いてやれば良いのです。

$$P_{jk}(\theta) = P^{+}_{jk}(\theta) - P^{+}_{j,k+1}(\theta)$$

ここにあるように，段階数が増えたとしても k 番目のカテゴリ以上に反応する確率から，$k + 1$ 番目に反応する確率を引いてやれば，k 番目のカテゴリに反応する確率が得られます。この計算をして描かれる曲線のことを**項目反応カテゴリ特性曲線 (Item Response Category Characteristic Curve; IRCCC)**，あるいは単に**カテゴリ確率曲線 (Category Probability Curve)** と呼ばれます。IRCCC は図 5.3 の下段に示されています。

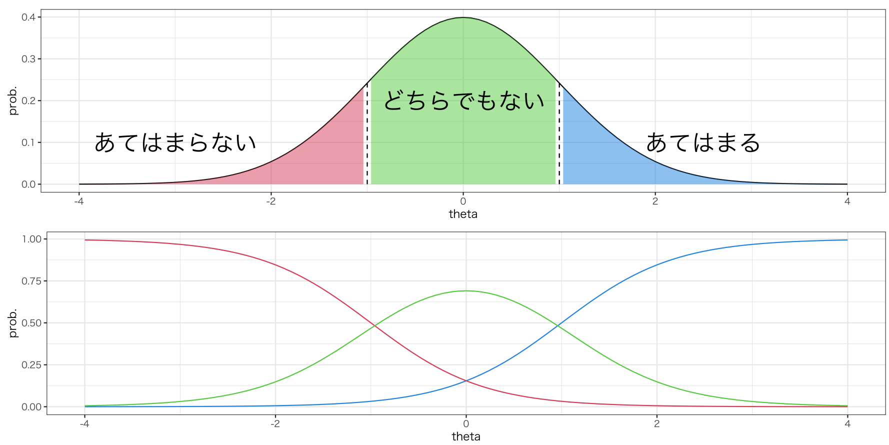


図 5.3　正規分布と閾値 (上図) と IRCCC(下図)

IRCCC を，θ の値がマイナスからプラスの方向に動かしながら見ていってください。最初は当然「当てはまらない」に反応する確率が一番高いのですが，それが徐々に下がっていきます。「どちらでもない」の反応確率は徐々に増えていき，閾値 b_{j1} で「当てはまらない」と逆転しピークを迎えることになります。その頃「当ては

まる」も徐々増えていき，閾値 b_{j2} でピークが逆転する，というようになります。ピークは逆転されても，他の確率が 0 になっているわけではなく，可能性は残っています。また IRCCC も ICC 同様に変換して，情報曲線に帰ることができます。すなわちどのあたりで鋭敏に情報を検出できるかを表現する項目反応カテゴリ情報曲線を描くこともできます。このようにして，段階反応でもその項目の特徴をデザインできるのです。

5.2.1 適切な反応段階を考える

実際の調査研究をすると，図 5.4 の上の段のような IRCCC が描かれることがあります。何かおかしいところがありませんか？これは 5 段階の反応モデルですが，4 番目の反応カテゴリがずいぶん低く，そのピークが 3 番目と 5 番目の反応カテゴリに潰されてしまっていますね。つまり，4 番目の反応がもっとも際立つシーンがないということです。言い換えるならば，これは尺度作成側が 5 段階だと思っていたにもかかわらず，回答者はどういう時に「やや当てはまる」と答えるのかがはっきりせず，実質 4 段階でしか反応していないことを表しています。

このような IRCCC が描かれてしまう場合は，$k=3$ の反応と $k=4$ の反応を合わせてひとつにしてしまうなど，段階の修正を考えると良いでしょう。具体的にはデータで 4 と入力していたものを，3 に置き換えたりします[*11]。修正したのが下の図になります。このように修正しても，情報関数は変わりません。同じデータから得られる情報は同じだからです。

このように 5 段階，7 段階を設定して回答者に無理やり回答を求めても，分析するとカテゴリのピークが潰れていることがあります。回答者の反応しやすいカテゴリ数を丁寧に設計してやることが重要です。もちろん無分別に尺度値をつけて，そのまま分析するのはもっとも不適切な方法です。

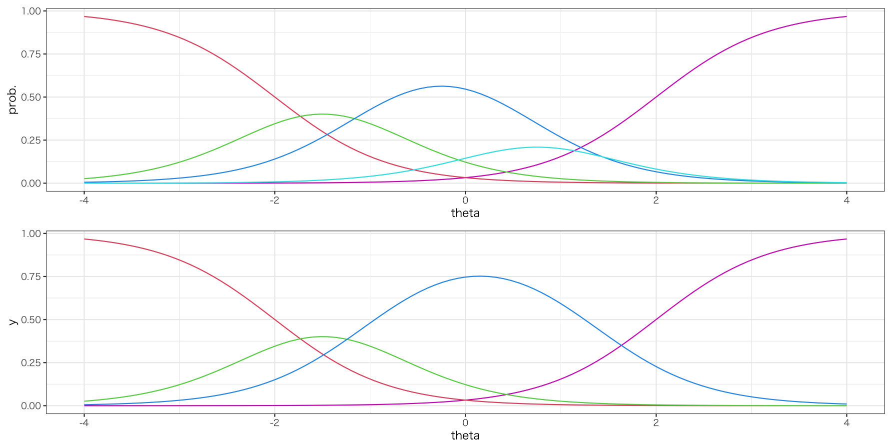


図 5.4 適切な反応段階をデザインする

5.3 因子分析の歴史と展開

ところで，因子分析モデルもテスト理論も，潜在変数モデルとしては同じでいずれも古典的テスト理論からの発展系なのでした。因子分析モデルは多段階反応が一般的で，多因子モデルで「潜在的な (心理学的な) 構造はどうか」ということを問題にします。ここでの目的は全体に共通する要素やその構造であり，何種類に別れて要素間の関係はどうなっているのか、というところが中心的関心事になります。一方，項目反応理論は

*11 4 を 5 に書き換えても構いません。ヒストグラムを見て，より正規分布っぽくなるようにすると良いでしょう

バイナリ反応が一般的で，因子数はひとつです。学力テストはその学力が測定できていることが重要で，因子の構造よりも因子得点をより精緻に推定できることの方が重要だからです。因子分析の言葉で言えば，因子得点をより精緻に表現しようとしているわけです。

さて，GRM は，項目反応理論の多段階モデルでした。実は GRM は因子分析モデルの発展系でもあるのです。因子分析は相関係数のモデルであったことを思い出してください。因子分析モデル自体は z_{ij} から始まっていましたが，変数同士の共分散 r_{ij} を考えるといくつかの仮定から**因子負荷量**だけのモデルに簡略化され，推定できるようになるのでした[12]。この標準化された共分散，すなわち相関係数はピアソンの積率相関係数とも言われ，間隔尺度水準以上の数値を使って計算されます。多段階の反応は順序尺度水準ですから，相関係数を計算するのは不適切で，そのまま因子分析することはできません[13]。では**順序尺度水準**の相関係数がないのかといわれると，あります。

順序尺度水準の変数 × 順序尺度水準の変数の相関は**ポリコリック相関係数 (polychoric correlation)** といいます。順序尺度水準の変数 × 間隔尺度水準の変数の相関は**ポリシリアル相関係数 (polyserial correlation)** といいます。ついでにバイナリ変数 × バイナリ変数の相関係数は**テトラコリック相関係数 (tetracholic correlation)** といいます。

これらの相関係数はいずれも，順序 (あるいはバイナリ) 変数の背後に連続体があると考え，潜在的な連続体 × 潜在的な連続体の相関係数を連続体のカテゴリが変わる閾値とともに推定するのです (図 5.5)。

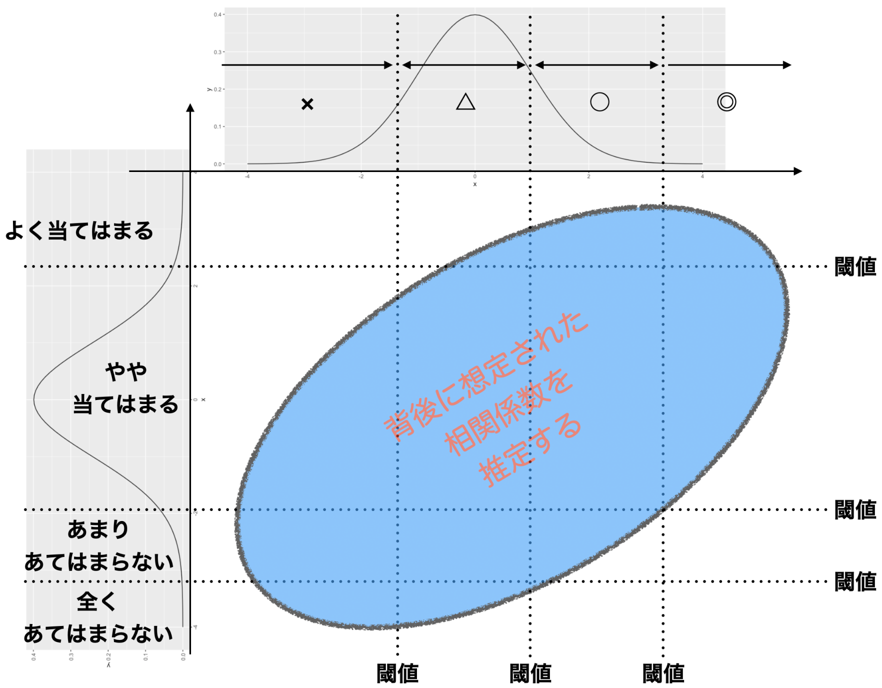


図 5.5 ポリコリック相関係数のイメージ

こうして推定された相関係数を使って因子分析を行うと，順序尺度水準に適した因子分析を行うことができます。この方法をとくに**カテゴリカル因子分析 (categorical factor analysis)** といいます。もっともこの名前を覚えておく必要はありません。**カテゴリカル因子分析は段階反応モデルと数学的に等価であるこ**

[12] 具体的にどうやって因子負荷量を算出するかは次回以降のお楽しみです。

[13] できないのですが，尺度値に変換することもなく機械的に分析してしまう悪い習慣が蔓延っているのは何度も指摘している通りです。くどいと思われるかもしれませんが，私は憤っているのです。

とがわかっています。GRM をすることはカテゴリカル因子分析をしていることと同じ，なのです。

もっとも GRM は項目反応理論の系列ですから，単因子構造を仮定しています。複数の因子を想定するカテゴリカル因子分析に対応するのは，正確には**多次元項目反応理論 (Multidimensional Item Response Theory)** といいます。しかし数学的･技術的には同じであり，カテゴリカル因子分析をした結果から IRCCC を描くこともできますし，実際に分析するソフト上では使用する変数がカテゴリカル (順序尺度水準) であることを指定するだけです。われわれユーザはもはや悩む必要はなく，ただただデータに適した分析をするだけで良いのです。

5.3.1 系譜の違いはどこに関係するか

因子分析モデルと項目反応理論が，カテゴリカル因子分析として概念的に統合されることを話してきました。本質的にはこのように違いがないのですが，系譜の違い，出自の違いはそれぞれの利用される文脈で，何を強調するかに影響してくることがあります。

たとえば因子分析の文脈では，共通性が低い項目は削除し，綺麗な因子構造を目指そうという考え方があります。尺度作成の中で 1 つの項目は 1 つの因子に対応しているべきであるという考え方があり (**単純構造の原理 (Principle of Simple Structure)** といいます)，もし 1 つの項目が複数の因子の影響を受けているようであれば，「美しくないので」削除されることがあります。因子分析は知能，性格，態度の研究で展開されてきたため，美しい「構造」を見つけ出すことに狙いがありますから，この美しさを損なうもの (項目) は取り除く，という方向に行きがちです。

一方で，項目反応理論はテスト理論の生まれです。もちろん回答者の能力や特性を測定するのに優れた項目とそうでない項目，という峻別はしますが，中でも「この項目は測定能力の偏差値 30 程度の回答者を測定するのに適している」とか，「偏差値 70 程度の回答者を測定するのに適している」と判断できます。偏差値 30 や 70 を測定するのに適した項目とは，非常に簡単 (ほとんどの人が正答する) だったり，非常に難易度が高い (ほとんどの人が誤答する) 項目です。心理尺度でいうと床効果，天井効果がみられる項目とされる，どちらかに偏った分布をもつ項目です。しかし，それを捨てるということにはならず，どのような項目でも回答者の能力を推定するための情報量がゼロではない，という考えから，さまざまな項目をどんどんためていく方向にいきがちです。項目反応理論の文脈では，あらゆる人に対する測定を準備しておく必要があり，むしろ回答者の特性にあわせて設問の方を選んだり，事前の項目特性から，前もってテストの項目構成をデザインする，ということを目的とするのです。

このように使われるシーンによって，「構造」か「機能 (=得点)」のどちらに注目するかが変わり，結果的に実践の方針がちがってくることもあるのです。図 5.6 にこの心理学的系譜 (左ルート) とテスト理論的系譜 (右ルート) の流れを描いてみました。

ポイントは「最終的には同じところに辿り着く」という点ですから，歴史的流れや個々のモデルの細かい数式を完全に理解していなくてもいいかもしれません。それよりは，心理学者として，あるいはテストをする側として，回答者に無理のない，それでいて必要な程度に精緻な情報が得られる適切な分析方法を選択できるようになることが重要です。

5.4 課題

■信頼性についての考え方 古典的テスト理論，因子分析モデル，項目反応理論，それぞれの信頼性の考え方を数式とともに自分のことばで説明できるようになろう。

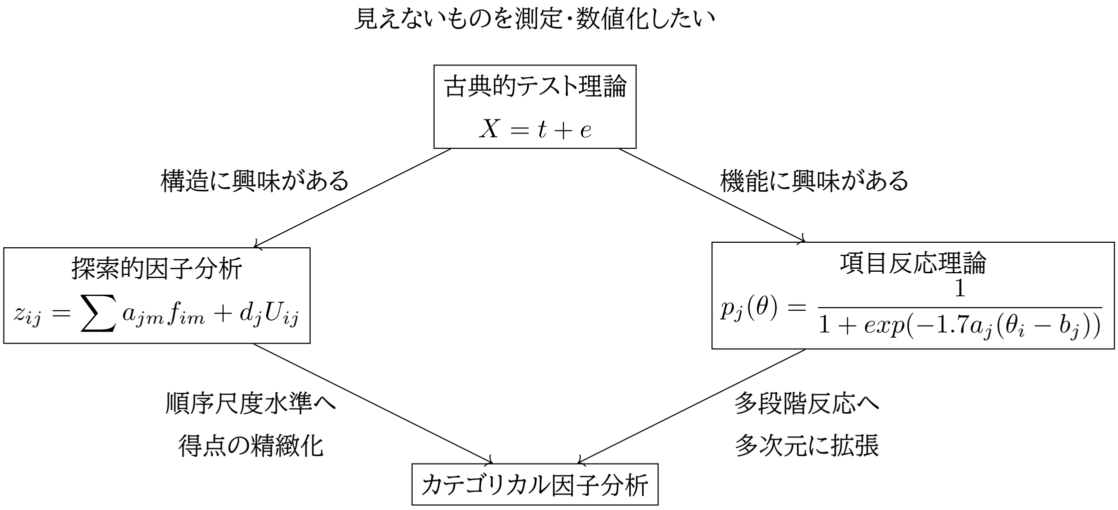


図 5.6 理論・モデルの流れ。左側のルートが心理学的系譜，右側のルートがテスト理論的系譜

■段階反応モデルの IRCCC 多段階モデルの IRCCC を，関数を描画するソフトなどを使って自分で描いてみよう。

第 6 章

行列計算の基礎

これまで古典的テスト理論，因子分析論，現代テスト理論を通じて，目に見えない潜在変数を数値化する方法について学んできました。潜在変数という心のモデルは，心理学の中心的関心事であり，実際多くの調査研究で潜在変数をモデルに組み込んで検証されています。その割には，どういったメカニズムで潜在変数が見出されているのかについての理解は十分行き渡っていないようです。たとえば因子分析モデルは，統計パッケージを使うと瞬時に「3 因子構造で因子負荷量はこれこれ，因子得点はこのようになっています」と答えを出してくれます。しかし，なぜそのような数字になったのか，どのようにそれが算出されたのかを知らなければ，何もわかっていないのと同じではないでしょうか？ 因子は「機械がやってくれるもの」と思考停止してしまうと，結局のところ私たちの知りたいことには辿り着けませんし，誤用の元になってしまいます。

なぜその肝心の箇所が放置されているかというと，数学的には**線形代数** (linear algebra) と呼ばれる計算が必要であり，そこについての文系数学的解説がないからです。線形代数はベクトルや行列の計算，文字と式の便利な表現形式です。これを知ることの利点は，**多くの数字のセットを簡単な記号で一般的に表現できるようになること**です。変数や回答者数が数十，数百，時には数万のサイズで得られた時，1 つ 1 つのデータにアルファベットを割り振っていたのでは間に合いませんので，線形代数はデータ解析には必須の知識です。

本講義では，線形代数の基礎を導入した上で，最終的には潜在変数，共通因子や**因子負荷量**と呼ばれるものがどのように算出されるのかを理解することを目的としています。事前の知識は必要なく，また目的に必要な最小限の知識だけで進めていきますので，一歩ずつ確実にフォローしてください[*1]。

6.1　行列とベクトル

行列やベクトルは，複数の数字をひとまとめにして扱うためのものです。まずはその基本的な形からみていきます。

■ベクトル　複数の数字を一行，あるいは一列にまとめて表現したものを，**行ベクトル** (row vector)，**列ベクトル** (column vector) といいます。

行ベクトルは次のように表します。

$$\boldsymbol{a} = \begin{pmatrix} a_1 & a_2 & \cdots & a_m \end{pmatrix}$$

[*1] ここからの話は小杉 (2018) の pp.148–179 に同内容のものがあります。もちろん線形代数のテキストとしては他にもいろいろあり，数学的な入門としては，基礎的には村上・佐藤・野澤・稲葉 (2016) が，発展的なところでは永田 (2005) が参考になるでしょう。より文系のデータ解析的解説が多いのは，絶版になってしまいましたが岡太 (2008) が最高です。

列ベクトルは次のように表します。

$$\boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

具体的には，a_1 とか b_2 のところには数字が入っています。つまり次のような形です。

$$\boldsymbol{a} = \begin{pmatrix} 1 & 3 & 5 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} 2 \\ 4 \\ 6 \\ 12 \\ 8 \end{pmatrix}$$

ここで今回の $\boldsymbol{a}$ は 3 つの要素が入っていますので，サイズは 3，同じく $\boldsymbol{b}$ はサイズが 5 のベクトルです。行列の言い方に合わせて 1×3 の (行) ベクトル，5×1 の (列) ベクトル，という言い方をすることもあります。このベクトルの中の数字は，とくに関係があるわけではありません。前に入っている数字がえらいとか，横にある方が重要だ，といったことはなく，ただただ数字をまとめて扱っているだけです。数字のセットを記号ひとつで表せるので，ずいぶん楽ですよね。

さて，行数も列数も 1 であるものつまり行列でない数字は，とくに**スカラー (scalar)** と呼びます。今までは $1 + 2 = 3$ といった計算をしていましたが，この $1, 2, 3$ はすべてスカラーだといえるわけです。

■行列　**行列 (matrix)** とは数を長方形に並べたものです。行列として並べられた数を成分といい，成分の横の並びを行，縦の並びを列と呼びます。

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}$$

この例は，n 行 m 列の行列を表しています。お気づきかと思いますが，行列やベクトルを表す場合は，アルファベットを太字にするのが慣例です。たとえば，A とか x は 1 つの数字を表していますが，$\boldsymbol{A}$ や $\boldsymbol{x}$ であれば行列やベクトルを表していることになります。成分を表す文字は，一般に a_{ij} のように，はじめの添え字で行番号，次の添え字で列番号を表します。行列の大きさは行数と列数とによって，$n \times m$ のように表現します。n と m が同じ，つまり行数と列数が同じであれば，これをとくに正方行列といいます。正方行列の例を次にあげておきます。

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

正方行列の中でも，i 行 j 列目の値が j 行 i 列目の値と同じである行列 $(a_{ij} = a_{ji})$ のことを**対称行列 (Symmetric Matrix)** といいます。

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 3 & 6 & 9 \end{pmatrix}$$

この (正方) 対称行列の形は，データ解析の中ではよくでてきます。たとえば 3 つの変数 x_1, x_2, x_3 について，その相関係数を考えたいとしましょう。相関係数は 2 つの数字の組み合わせですから，x_1 と x_2，x_1 と x_3，x_2 と x_3 について計算でき，それぞれ r_{12}, r_{13}, r_{23} と表したとします。i と j の相関係数 r_{ij} は，j と i

の相関係数と同じ $(r_{ij} = r_{ji})$ であり，また $r_{jj} = 1.0$ なのは定義から明らかです。これを行列で表すと次のようになります。

$$\boldsymbol{R} = \begin{pmatrix} 1 & r_{12} & r_{13} \\ r_{21} & 1 & r_{23} \\ r_{31} & r_{32} & 1 \end{pmatrix} = \begin{pmatrix} 1 & r_{12} & r_{13} \\ r_{12} & 1 & r_{23} \\ r_{13} & r_{23} & 1 \end{pmatrix}$$

このように対称行列になっています。この行列をとくに**相関行列 (Correlation Matrix)** といいます。また相関係数は標準化された共分散でもありました。標準化するまえの相関行列は，**分散共分散行列 (Covariance Matrix)** と言います。その名前の通り，自分自身との共分散が分散になるわけですから，右上から右下にかけての対角線上にある要素 (これをとくに**対角 (diagonal)** 要素といいます) が分散であり，それ以外が共分散になっている行列です。

$$\boldsymbol{V} = \begin{pmatrix} s_1^2 & s_{12} & s_{13} \\ s_{21} & s_2^2 & s_{23} \\ s_{31} & s_{32} & s_3^2 \end{pmatrix}$$

また，正方行列の中でもとくに対角要素にのみ値があって，それ以外はすべて 0 になっている行列のことを**対角行列 (diagonal matrix)**，対角行列の中でもとくに，対角項が 1 になっているものは**単位行列 (identity matrix)** と呼びます。単位行列は $\boldsymbol{I}$ とか $\boldsymbol{E}$ で表されます。

$$\boldsymbol{I} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

これは後ほど，掛け算をするときに「かけても変わらない状態」を表すために用いられます。

6.2 行列の四則演算と操作

行列の四則演算は，通常のスカラーのそれとは異なります。改めて，行列としての加減乗除を定義するのだと思ってください。

■加法・減法 まずは行列の足し算 (加法)，引き算 (減法) から説明します[*2]。これはそれぞれ対応する位置にある成分を加え合わせる (減じる) ことで表されます。

$$\boldsymbol{A} + \boldsymbol{B} = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1m} + b_{1m} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2m} + b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} + b_{n1} & a_{n2} + b_{n2} & \cdots & a_{nm} + b_{nm} \end{pmatrix}$$

数値例をみておきましょう。

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$$

これからわかるように，行列の加法，減法は大きさの等しい行列でないと成り立ちません。サイズが違うものを足そうとすると，演算できない箇所が出てしまうのです。このように行列では，「計算できない」という状態になることが少なからずあります。行列のサイズに注意が必要，ということがお分かりいただけるかと思います。

[*2] ベクトルは行列の中でも，行数あるいは列数が 1 のものですので，これで一般的に表現します。

■乗法　続いて掛け算です。まずスカラーと行列の積を見てみましょう[*3]。

$$\lambda \boldsymbol{A} = \boldsymbol{A}\lambda = \begin{pmatrix} \lambda a_{11} & \lambda a_{12} & \cdots & \lambda a_{1m} \\ \lambda a_{21} & \lambda a_{22} & \cdots & \lambda a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda a_{n1} & \lambda a_{n2} & \cdots & \lambda a_{nm} \end{pmatrix}$$

実際の計算は，各成分をスカラー倍すればよいだけですので，比較的簡単ですね。

$$2 \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 2 \times 1 & 2 \times 2 \\ 2 \times 3 & 2 \times 4 \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$$

次はベクトルとベクトルの掛け算です。これは形が変わってしまうので，注意が必要です。まずは行ベクトルに列ベクトルをかける例からみていきましょう。

$$\boldsymbol{ab} = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \sum_{j=1}^{n} a_j b_j$$

掛け算なのですが，足し合わせるという計算プロセスが入り込んでいるので，結果はスカラーになります。掛け算なのにどうして足し算の要素が入るんだ，というクレームは，今はなしです。このように計算することに決めたことで，あとあと便利なことが出て来ますから，作法にまず慣れてからにしましょう。数値例も確認しておきます。

$$\begin{pmatrix} 1 & 2 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 4 \\ 2 \end{pmatrix} = 1 \times 3 + 2 \times 4 + 1 \times 2 = 13$$

ここで注意して欲しいのは，両方のベクトルのサイズが同じ ($1 \times n$ ベクトルと，$n \times 1$ ベクトル，いずれもサイズは n) ということです。サイズが違うと，演算が対応しない要素が出てくるので，計算できない，が答えになります。

今度は向きを変えて，列ベクトルに右から行ベクトルをかけてみましょう。

$$\boldsymbol{ab} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \begin{pmatrix} b_1 & b_2 & \cdots & b_n \end{pmatrix} = \begin{pmatrix} a_1 b_1 & a_1 b_2 & \cdots & a_1 b_n \\ a_2 b_1 & a_2 b_2 & \cdots & a_2 b_n \\ \vdots & \vdots & & \vdots \\ a_n b_1 & a_2 b_2 & \cdots & a_n b_n \end{pmatrix}$$

今度は行列になりました。かける順番が変わるとサイズが変わる (ここでは，上の例では 1×1 のサイズ，下の例では $n \times n$ のサイズ) ことに注意してください。スカラーの計算では順番を入れ替えても，たとえば $2 \times 3 = 3 \times 2$ のように同じ答えになりましたが，行列の場合は必ずしもそうはならない，ということです。

$$\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 3 & 4 & 2 \end{pmatrix} = \begin{pmatrix} 1 \times 3 & 1 \times 4 & 1 \times 2 \\ 2 \times 3 & 2 \times 4 & 2 \times 2 \\ 1 \times 3 & 1 \times 4 & 1 \times 2 \end{pmatrix} = \begin{pmatrix} 3 & 4 & 2 \\ 6 & 8 & 4 \\ 3 & 4 & 2 \end{pmatrix}$$

行列とベクトルの積や，行列と行列の積はこの応用になってきます。まず行列に列ベクトルを右からかける例を見てみましょう。結果は列ベクトルになります。

[*3] 式中にでてくる λ はギリシア文字でラムダといいます。小文字が λ，大文字では Λ と書きます。

$$
\boldsymbol{Ab} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} = \begin{pmatrix} \displaystyle\sum_{j=1}^{m} a_{1j} b_j \\ \displaystyle\sum_{j=1}^{m} a_{2j} b_j \\ \vdots \\ \displaystyle\sum_{j=1}^{m} a_{nj} b_j \end{pmatrix}
$$

ここでも掛け算なのに足し算のプロセスが入ってきています。注意深く記号を読んでみてください。数値例でも確認しておきます。

$$
\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \times 2 + 2 \times 1 \\ 3 \times 2 + 4 \times 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 10 \end{pmatrix}
$$

今度は行列に行ベクトルを左からかけましょう。結果は行ベクトルになります。

$$
\boldsymbol{cA} = \begin{pmatrix} c_1 & c_2 & \cdots & c_n \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} = \begin{pmatrix} \displaystyle\sum_{j=1}^{n} a_{j1} c_j & \displaystyle\sum_{j=1}^{n} a_{j2} c_j & \cdots & \displaystyle\sum_{j=1}^{n} a_{jm} c_j \end{pmatrix}
$$

$$
\begin{pmatrix} 1 & 3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 \times 1 + 2 \times 3 & 1 \times 0 + 3 \times 3 \end{pmatrix} = \begin{pmatrix} 7 & 9 \end{pmatrix}
$$

さて，最後に行列と行列の積を考えます。行列 $\boldsymbol{A}$ と $\boldsymbol{B}$ の積が成立するのは，前者の列数と後者の行数とが等しいときに限られます。行列 $\boldsymbol{A}$ のサイズが $n \times m$，行列 $\boldsymbol{B}$ のサイズが $m \times l$ とすると，その積は $n \times l$ の行列になります。計算手続きは，次のようになります，

$$
\boldsymbol{AB} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1l} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{ml} \end{pmatrix} = \begin{pmatrix} \displaystyle\sum_{j=1}^{m} a_{1j} b_{j1} & \cdots & \displaystyle\sum_{j=1}^{m} a_{1j} b_{jl} \\ \vdots & \ddots & \vdots \\ \displaystyle\sum_{j=1}^{m} a_{nj} b_{j1} & \cdots & \displaystyle\sum_{j=1}^{m} a_{nj} b_{jl} \end{pmatrix}
$$

どうにもこれはややこしいかもしれません。足し算や掛け算が入り乱れるし，計算途中でどの要素を計算しているかわからなくなるからです。ベクトルと行列の積の時のように，前の行列の要素は左に進み，後ろの行列の要素は縦に進みますから，左手と右手で違う図形を描く認知課題のように，そもそも混乱しやすい作業なのです。

しかし 2 つほど注意をしておくと，間違いにくくなります。1 つは積によって得られる**結果の行列サイズを意識すること**です。先ほど，前の行列の列数と，後ろの行列の行数が同じでないと計算ができないといいました。つまり，$n \times m$ 行列と $m \times l$ 行列でないと計算できない (m が同じ) ということです。また，結果は $n \times l$ 行列になります。前の行列の行数，後ろの行列の列数が結果のサイズです。ここに注目しておくと，計算を始める前に，計算が可能かどうかと結果の行列サイズは想像がつくのです (図 6.1)。

また，実際に計算する際は，**前の行列に横の，後ろの行列に縦の補助線を入れる**とわかりやすいかもしれません。こうすることで，間違えて計算を進めることがないようになるからです。

$$
\left(\begin{array}{cc} 1 & 2 \\ \hline 3 & 4 \\ \hline 5 & 6 \end{array}\right) \left(\begin{array}{c|c|c} 0 & 1 & 1 \\ 1 & 0 & 1 \end{array}\right) =
$$

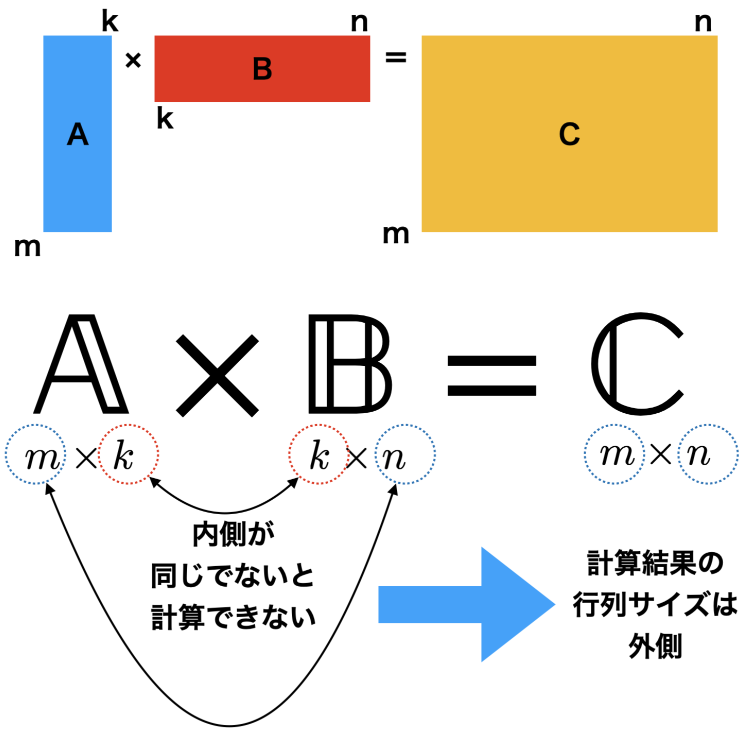


図 6.1　行列のサイズを把握する

$$\begin{pmatrix} 1\times 0+2\times 1 & 1\times 1+2\times 0 & 1\times 1+2\times 1 \\ 3\times 0+4\times 1 & 3\times 1+4\times 0 & 3\times 1+4\times 1 \\ 5\times 0+6\times 1 & 5\times 1+6\times 0 & 5\times 1+6\times 1 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 3 \\ 4 & 3 & 7 \\ 6 & 5 & 11 \end{pmatrix}$$

■転置　次に**転置** (**transpose**) と呼ばれる操作を説明します。これは計算の便宜上，よく使われる行列操作のひとつです。

大きさ $n \times m$ の行列 $\boldsymbol{A}$ における i 行 j 列成分を j 行 i 列成分とする $m \times n$ 行列のことを，元の行列 $\boldsymbol{A}$ の転置とよび，$\boldsymbol{A}'$ や $\boldsymbol{A}^T$ と表します。行列を転ばせたようなイメージです。

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \text{のとき，} \boldsymbol{A}' = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{pmatrix}$$

ベクトルも転置でき，行ベクトルを転置すると列ベクトルに，列ベクトルを転置すると行ベクトルになります。

$$\boldsymbol{a} = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \end{pmatrix} \text{のとき，} \boldsymbol{a}' = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

また，転置には以下のような性質があります。これは知識として知っておくだけでよいでしょう。

1. $(\boldsymbol{A}')' = \boldsymbol{A}$
2. $(\boldsymbol{A}+\boldsymbol{B})' = \boldsymbol{A}' + \boldsymbol{B}'$

3. $(\boldsymbol{AB})' = \boldsymbol{B}'\boldsymbol{A}'$
4. $(c\boldsymbol{A})' = c\boldsymbol{A}'$

■逆行列　最後に**逆行列**のお話をします。逆行列は割り算のイメージです。ある行列にその逆行列をかけると単位行列になる，つまり割ると 1 になるような行列のことです。

正確に表現すると，ある正方行列 $\boldsymbol{A}$ に対し，$\boldsymbol{AX} = \boldsymbol{I}$ となるような行列 $\boldsymbol{X}$ が存在するとき，これを $\boldsymbol{A}$ の**逆行列 (inverse)** と呼び，$\boldsymbol{A}^{-1}$ で表します。正方行列でない場合に逆行列はありませんし，正方行列であっても逆行列が存在しない場合もあります。逆行列の例をみてみましょう。$\boldsymbol{A} = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix}$ とすると，次の計算が成り立ちます。

$$\boldsymbol{AB} = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix} \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

このとき $\boldsymbol{B}$ は $\boldsymbol{A}$ の逆行列，すなわち $\boldsymbol{A}^{-1} = \boldsymbol{B}$ といえます。

とくに対角行列の逆行列は，対角成分の逆数をそれぞれ対角成分とする行列になります。

$$\boldsymbol{D} = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix} \text{のとき，} \boldsymbol{D}^{-1} = \begin{pmatrix} \frac{1}{d_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{d_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{d_n} \end{pmatrix}$$

逆行列は，行列の世界の割り算のようなものです。これで一通り四則演算の定義ができました。

6.3 行列を使うと便利なこと

さて，ここまでで行列の計算の話をして来ましたが，どこが良いのかいまいちピンとこない，という人もいるかもしれません。そこで最後にどうしてこのような計算をするのか，何が良いのかを説明してみたいと思います。

6.3.1 行列と方程式

線形代数は「便利な書き方」の学問です。便利な書き方をするためにルールが作られていますから，ルールから学ぶと「なんでそんな変な操作をするんだ」という気持ちになるのもわかります。

では何が便利になるのでしょうか。これは方程式を解くことと関係があります。たとえば，以下のような連立方程式があったとしましょう。

$$\begin{cases} x - 2y - 5z & = & 3 \\ 5x + 4y + 3z & = & 1 \\ 3x + y - 3z & = & 6 \end{cases}$$

これは行列で表現すると，次のようになります。

$$\begin{pmatrix} 1 & -2 & -5 \\ 5 & 4 & 3 \\ 3 & 1 & -3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 6 \end{pmatrix}$$

この左辺を行列とベクトルの式の計算ルールにのっとって展開してみてください。ちゃんと最初の連立方程

式の左辺になることがわかると思います。かけて足して，という面倒な計算ルールは，連立方程式を簡単に表記するためのものだったのですね。

最終的にはこの方程式を解いて，次のように答えを求めます。

$$\begin{pmatrix} 1 & -2 & -5 \\ 5 & 4 & 3 \\ 3 & 1 & -3 \end{pmatrix} \begin{pmatrix} x=-1 \\ y=3 \\ z=-2 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 6 \end{pmatrix}$$

皆さんも学校で習ったように，このような連立方程式を解く方法として，加減法や代入法というのがあります。ですがここはひとつ，行列を使った解法を考えて見ましょう。

そのような解法のひとつ，消去法は，ひとつの方程式を何倍かして，他の方程式に加えることにより，方程式をどんどん簡単にしていくというものです。まず，第一の式を 5 倍，あるいは 3 倍して，第二，第三の式から x の項を消去します。

$$\left\{ \begin{array}{rcl} x-2y-5z & = & 3 \\ -14y-28z & = & 14 \\ -7y-12z & = & 3 \end{array} \right.$$

第二の式の係数を簡単にしておきましょう。

$$\left\{ \begin{array}{rcl} x-2y-5z & = & 3 \\ y+2z & = & -1 \\ -7y-12z & = & 3 \end{array} \right.$$

第二の式を 7 倍して，第三の式から y を消去します。

$$\left\{ \begin{array}{rcl} x-2y-5z & = & 3 \\ y+2z & = & -1 \\ 2z & = & -4 \end{array} \right.$$

あとはこれの 3 行目から $z=-2$ が得られ，芋づる式に $x=-1$， $y=3$ が得られました。

この操作は，式を一本ずつ，あるいは 2 つの式を組み合わせて文字を消していく消去法を係数全体に行う操作になっています。実際，ここで操作される係数だけ見ていくと，次のようになります。

■第一段階

$$\begin{pmatrix} 1 & -2 & -5 \\ 0 & 1 & 2 \\ 0 & -7 & -12 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \\ 3 \end{pmatrix}$$

■第二段階

$$\begin{pmatrix} 1 & -2 & -5 \\ 0 & 1 & 2 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \\ -4 \end{pmatrix}$$

さらにこの方法を改良した，ガウスージョルダンの消去法というものがあります。この手法による係数の変化を，行列表記で見ていくことにします。

まず第一段目は同じです。

$$\begin{pmatrix} 1 & -2 & -5 \\ 0 & 1 & 2 \\ 0 & -7 & -12 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \\ 3 \end{pmatrix}$$

次に，第二の方程式を用いて第一と第三の式から y の係数を消してしまいます。

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 4 \end{pmatrix}$$

最後に，第三の式の z の係数を 1 にして，第一，第二式の z の係数を消してしまいましょう。

$$\begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}-1\\3\\-2\end{pmatrix}$$

最後の形を見ると，左辺は単位行列になっていますから，

$$\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}-1\\3\\-2\end{pmatrix}$$

と解が求められたことがわかります。ここで注目すべきは，連立方程式の解を求めるプロセスは係数行列を単位行列に変えていくプロセスだった，ということです。係数行列が単位行列になれば，それはもう答えを出したことになるのです。

さて，係数行列を $\boldsymbol{A}$ とすると，その逆行列 $\boldsymbol{A}^{-1}$ があれば $\boldsymbol{A}^{-1}\boldsymbol{A}=\boldsymbol{I}$ となるのでした。であれば，連立方程式の右辺にあったベクトルに $\boldsymbol{A}^{-1}$ をかけてやれば，一気に答えが求まるではないですか。

実際に見て見ましょう。

$$\begin{pmatrix}1 & -2 & -5\\5 & 4 & 3\\3 & 1 & -3\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}3\\1\\6\end{pmatrix}$$

この連立方程式に対して，次のような操作をします。

$$\begin{pmatrix}1 & -2 & -5\\5 & 4 & 3\\3 & 1 & -3\end{pmatrix}^{-1}\begin{pmatrix}1 & -2 & -5\\5 & 4 & 3\\3 & 1 & -3\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}1 & -2 & -5\\5 & 4 & 3\\3 & 1 & -3\end{pmatrix}^{-1}\begin{pmatrix}3\\1\\6\end{pmatrix}$$

とします[*4]。すると左辺は単位行列になりますから，次のように計算すれば一気に答えが求まることになるのです。

$$\begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}-1\\3\\-2\end{pmatrix}$$

つまり，連立方程式を解くという問題が，係数行列の逆行列を求める問題になります。また，逆行列は存在しないこともある，ということでしたが，その場合その連立方程式は解けない，ということになります。

6.4 課題

■線形代数の練習問題 行列 $\boldsymbol{A}=\begin{pmatrix}1 & 2 & -1\\3 & 1 & 0\end{pmatrix}$ のとき，次の計算をしなさい。なお，$\boldsymbol{I}_n$ とは $n\times n$ の単位行列，$\boldsymbol{O}$ とはすべての要素が 0 の適当なサイズの正方行列であることを表します。

1. $\boldsymbol{A}'\boldsymbol{A}$
2. $\boldsymbol{A}\boldsymbol{A}'$
3. $\boldsymbol{A}\boldsymbol{I}_3$
4. $\boldsymbol{A}$

[*4] 数値的には $\begin{pmatrix}1 & -2 & -5\\5 & 4 & 3\\3 & 1 & -3\end{pmatrix}^{-1}=\begin{pmatrix}15/28 & 11/28 & -1/2\\-6/7 & -3/7 & 1\\1/4 & 1/4 & -1/2\end{pmatrix}$ という行列です

■線形代数の練習問題その 2 行列 $\boldsymbol{A} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} 3 & 1 & 5 \\ 2 & 4 & 6 \end{pmatrix}, \boldsymbol{C} = \begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix}$, 列ベクトル $\boldsymbol{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$, 行ベクトル $\boldsymbol{y} = \begin{pmatrix} 2 & 8 \end{pmatrix}$ とするとき, 次の計算をしなさい。なお, 計算が定義されていないものについては「計算できない」と回答しなさい。

1. $\boldsymbol{A} + \boldsymbol{B}$
2. $\boldsymbol{A} - \boldsymbol{C}$
3. $\boldsymbol{AB}$
4. $\boldsymbol{AC}$
5. $\boldsymbol{B}'\boldsymbol{A}$
6. $\boldsymbol{A}\boldsymbol{y}'$
7. $\boldsymbol{xy}$
8. $\boldsymbol{xB}$
9. $\boldsymbol{x}'\boldsymbol{B}'$
10. $\boldsymbol{yx}$

■連立方程式を解く 次の連立方程式を解きなさい。

$$\begin{cases} x - 2y + 3z = & 1 \\ 3x + y - 5z = & -4 \\ -2x + 6y - 9z = & -2 \end{cases}$$

第 7 章

行列による関係の表現

前回から線形代数の話をしています。線形代数は数字をセットで扱うための表現方法，計算方法ですから，多変量データを分析しようという時には必須の技術になります。前回は線形代数の導入ですから，計算方法を解説してきましたが，今回はこの計算方法を使って具体的にデータをどのように表現し，どのように計算するのかを見ていくことになります。

7.1 データの行列表現

ここまで行列の形ばかり見て来ましたが，狙いはあくまでも調査研究など，多変量データを扱う場面での利用です。なぜ多変量データ分析をする際にこのような知識が必要なのか，思うかもしれません。ですが，得られるデータは行列として扱うと表現が大変便利なのです。たとえば質問項目が m 個あって，調査対象者 n 人から回答を得たとすると，データは次のように表現できます。

$$\boldsymbol{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}$$

データ全体をこうして，ひとつの記号で表現できたら便利ですよね。これらを使ったデータの表記に慣れておきましょう。

各反応の平均点は以下のように表現されます。まず，要素がすべて 1 からなるベクトルを次のように表します。

$$\mathbf{1} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

わかりにくいかもしれませんが，この $\mathbf{1}$ は太字でベクトルを表しており，スカラーの 1 とは違うことに注意してください。

さて，各項目の和はベクトルの掛け算の定義によって次のように表現できます。

$$\boldsymbol{X}'\boldsymbol{1} = \begin{pmatrix} \sum_{i=1}^{n} x_{i1} \\ \sum_{i=1}^{n} x_{i2} \\ \vdots \\ \sum_{i=1}^{n} x_{im} \end{pmatrix}$$

これを使って平均値 (列) ベクトル $\boldsymbol{m}$ を次のように表すことができます。

$$\boldsymbol{m} = \frac{1}{n}\boldsymbol{X}'\boldsymbol{1} = \begin{pmatrix} 1/n\sum_{i=1}^{n} x_{i1} \\ 1/n\sum_{i=1}^{n} x_{i2} \\ \vdots \\ 1/n\sum_{i=1}^{n} x_{im} \end{pmatrix} = \begin{pmatrix} \bar{x}_{\cdot 1} \\ \bar{x}_{\cdot 2} \\ \vdots \\ \bar{x}_{\cdot m} \end{pmatrix}$$

ここで $\bar{x}_{\cdot 1}$ は 1 つめの添字 i を足し合わせて割ることでなくしていますから，$\bar{x}_1$ のように省略して書くことがあります。このときの 1 は「第一番目の変数」という意味であり，個人の情報がなくなっている変数を意味していることに注意してください。

さて，平均からの偏差を要素に持つ行列 $\boldsymbol{V}$ を考えたとします。

$$\begin{aligned} \boldsymbol{V} &= \boldsymbol{X} - \boldsymbol{1}\boldsymbol{m}' \\ &= \boldsymbol{X} - \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \begin{pmatrix} \bar{x}_{\cdot 1} & \bar{x}_{\cdot 2} & \cdots & \bar{x}_{\cdot m} \end{pmatrix} \\ &= \boldsymbol{X} - \begin{pmatrix} \bar{x}_{\cdot 1} & \bar{x}_{\cdot 2} & \cdots & \bar{x}_{\cdot m} \\ \bar{x}_{\cdot 1} & \bar{x}_{\cdot 2} & \cdots & \bar{x}_{\cdot m} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{x}_{\cdot 1} & \bar{x}_{\cdot 2} & \cdots & \bar{x}_{\cdot m} \end{pmatrix} \end{aligned}$$

この行列 $\boldsymbol{V}$ のサイズは $n \times m$ であることに注意してください。これはまた，次のように表すこともできます。

$$\boldsymbol{V} = (\boldsymbol{I} - \frac{1}{n}\boldsymbol{1}\boldsymbol{1}')\boldsymbol{X}$$

ここで $\boldsymbol{I}$ は適切なサイズの単位行列です[*1]。

これを使うと，たとえば分散共分散行列 $\boldsymbol{S}$ は次のようになります。

$$\boldsymbol{S} = \frac{1}{n}\boldsymbol{V}'\boldsymbol{V} = \begin{pmatrix} s_1^2 & s_{12} & \cdots & s_{1m} \\ s_{21} & s_2^2 & \cdots & s_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m1} & s_{m2} & \cdots & s_m^2 \end{pmatrix}$$

[*1] 適切なサイズってなんだよ，と思いますよね。これは計算に合うようなサイズ，という意味です。具体的に考えてみますと，$\boldsymbol{I}$ の後ろは $\boldsymbol{1}\boldsymbol{1}'$ です。$\boldsymbol{1}$ は $n \times 1$ の列ベクトルで，転置したものと掛け合わせますから，$\boldsymbol{1}\boldsymbol{1}'$ のサイズは $n \times n$ です。行列の引き算は同じサイズでないと成立しませんから，ここでの $\boldsymbol{I}$ も $n \times n$ でなければなりません。カッコの中身が $n \times n$ で，それにサイズ $n \times m$ である $\boldsymbol{X}$ をかけますから，計算結果や右辺のサイズは $n \times m$ になります。

ここで s_j とあるのは第 j 変数の標準偏差を，s_{jk} とあるのは第 j 変数と第 k 変数の共分散です。添え字は変数番号になっています。また，ここでもサイズに注目してください。$\boldsymbol{V}'\boldsymbol{V}$ は，サイズで言うと $m \times n$ と $n \times m$ の積ですから，$m \times m$ になります。この行列は正方対称行列です。

また，対角項に各変数の標準偏差 s_j が入った行列 $\boldsymbol{Q}$ を以下のように定めるとしましょう。次のような行列です。

$$\boldsymbol{Q} = \begin{pmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_m \end{pmatrix}$$

そうすると，これの**逆行列**をつかって標準得点行列 $\boldsymbol{Z}$ を次のように表すことができます。

$$\boldsymbol{Z} = \boldsymbol{V}\boldsymbol{Q}^{-1}$$

さらに，これを用いて相関行列 $\boldsymbol{R}$ を次のように表すことができます。

$$\boldsymbol{R} = \frac{1}{n}\boldsymbol{Z}'\boldsymbol{Z} = \begin{pmatrix} 1 & r_{12} & \cdots & r_{1m} \\ r_{21} & 1 & \cdots & r_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & 1 \end{pmatrix}$$

データのサイズにかかわらず，一般的にこのように表現できるのはとてもわかりやすいですね。

7.2 線形モデルの行列表現

ベクトルや行列の記法をつかうと，**回帰分析**や**重回帰分析**の式がとても単純な形で表現できます。

回帰分析は，$Y = aX + b + e$ という式で表現できる，ということでしたが，式中の X や Y は観測されたデータですので，$X = \{x_1, x_2, \cdots, x_n\}$，$Y = \{y_1, y_2, \cdots, y_n\}$ というベクトルだと考えることができます。ですから，正確に書けば，ベクトルを使って次のように書くべきです。

$$\boldsymbol{Y} = a\boldsymbol{X} + \boldsymbol{b} + \boldsymbol{e}$$

これは，要素を表現しながら書くと[*2]次のようになります。

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = a \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} b \\ b \\ \vdots \\ b \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}$$

列ベクトル $\boldsymbol{X}$ に定数 a をかけて得られるのは同じサイズの列ベクトル，列ベクトル同士は足しても同じサイズの列ベクトルですから，左辺と右辺はどちらも列ベクトルで，対応関係が取れていることになります。

ここで少し表現の工夫をします。説明変数 X のベクトルの左に数字の 1 だけが入った列を作ります。また，係数もまとめてベクトル $\boldsymbol{\beta}$ を次のように用意します。

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} b \\ a \end{pmatrix}$$

[*2] エレメントワイズ element-wise の表現，と言ったりします。

このようにすると，回帰分析の式は

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{e}$$

と表すことができます。とても簡単な表現になりました (試しに各行を行列の計算式に則って計算してみてください。うまく表現できていることがわかると思います)。

さらにこの表現はありがたいことに，複数の説明変数がある重回帰分析の時でも同じ形で表すことができます。重回帰分析は，これまでの書き方ですと $Y = a_1X_1 + a_2X_2 + \cdots + a_nX_n + b + e$ というようにしていました。ここで，係数と切片を 1 つの行列で表現する時わかりやすくするために，少し書き換えて $Y = \beta_0 + \beta_1X_1 + \beta_2X_2 + \cdots + e$ としましょう。記号が変わっただけで中身は同じ，意味も同じです[*3]。ただ，切片 b を β_0 として右辺の一番前に持って来ました。というのも，そうするとベクトルで書く時にわかりやすいからです。

説明変数行列の左端に 1 を入れたベクトルを追加し，回帰係数 β もセットにして，次のように表現します。

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1m} \\ 1 & x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nm} \end{pmatrix}, \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{pmatrix}$$

とすると，重回帰分析の式は次のように簡単な表現に変わります。

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{e}$$

なんと，説明変数が m 個に増えたのに，式の形は (単) 回帰分析のそれと同じではありませんか！

このように，行列表現をすると多くの変数を一気に扱い，表現できるのです。このため，多変量解析ではベクトルの表記が基本になります。サイズを気にせず一般的に表現できるからです。

実際にこれらの式を読む時は，行列のサイズをイメージしながら読むと良いでしょう。たとえば左辺の $\boldsymbol{Y}$ はサイズ n のベクトルなのですから，右辺の $\boldsymbol{X\beta}$ もサイズ n の縦ベクトルになるはずなのです。実際，$\boldsymbol{X}$ は $n \times (m + 1)$ の行列で，β は $(m + 1) \times 1$ のベクトルですから，計算結果は $n \times 1$，つまりサイズ n の縦ベクトルです。

7.3 デザイン行列

ところで，心理統計と言えば平均値の差を見ることだ，という話はこれまで散々聞いてきたところかと思います。心理学は要因計画を立て，標本の平均値差から母集団に話を一般化するために，推測統計の知見を使って，帰無仮説検定やベイズ推定法を駆使するというやつです。この要因計画は実は線形モデルの一環であり，**一般線形モデル (General Linear Model)** と呼ばれています。これを行列で表現することを，ここでは少し考えてみたいと思います。

まずは回帰分析と要因計画は何が同じで何が違うのかを，はっきりさせましょう。同じところは線形モデルであるというところ，違うところは，回帰分析は説明変数も従属変数も連続変数であるのに対し，要因計画では一般に説明変数が離散変数であること，でした。離散変数であるとは，言い方を変えると名義尺度水準の数字だということです。すなわち「統制群」か「実験群」か，という違いを表すのに，$0, 1$ と言った数字を割り

[*3] 厳密に言えば記述統計学として誤差を最小にするように推定した係数はアルファベット $b_0, b_1, \ldots$ で，推測統計学として母数の推定値として算出した係数はギリシア文字 $\beta_0, \beta_1 \ldots$ で表現する，というルールです。すでに習ったように，最小二乗法での推定値と最尤法での推定値は，誤差が正規分布する場合一致しますので，ただ書き変わっただけだと思っていただいて問題ありません。

振ったものです。これは別に 3 と 12523, という数字を割り振ったと言ってもいいのです。だって名義尺度水準は, 数字と対象が一対一対応していれば良いのですから。

ここでは数学的に話をしやすくするために, 統制群を 0, 実験群を 1 とするとしましょう。線形モデルという枠組みは一緒なのですから, 従属変数 y_i が説明変数 x_i によって変わるわけですが, ここではこの x が $\{0, 1\}$, というわけです。線形モデルを要素ごとに表現すると次のようになります。

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

この時, i さんは統制群に割り振られていたとすると, $x_i = 0$ ですから, この式は次のようになります。

$$y_i = \beta_0 + e_i$$

逆に, i さんが実験群に割り振られていたとしますと, $x_i = 1$ ですから, この式は次のようになります。

$$y_i = \beta_0 + \beta_1 + e_i$$

これを見るとわかるように, 両群のベースライン β_0 は同じで, そこに効果 β_1 が乗っかるかどうか, が興味の的になります。この式の右辺に i は誤差成分しかなく, 誤差を抜きにすると従属変数は β_1 だけ変化するはずだ, というところから, 平均値の差を検証しましょうと言ってることになるのでした。

これも x_i が個人ごとに変わるベクトルだと考えると, 行列表現では次のようになります。

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{e}$$

同じ形ですね！ただし, ここでベクトル $\boldsymbol{x}$ は, その中身が $\boldsymbol{x} = (0, 0, 0, 1, 1, 0, 1, 0, ...)$ のように実験群か統制群かを分けるフラグが入っているだけになります。

以上は実験群と統制群という 2 群の話でしたが, 3 群以上になっても基本的なアイデアは同じです。たとえば表 7.1 のようなデータセットがあったとしましょう。

表 7.1 群間要因 (3 水準) のデータセット例

参加者番号	群わけ	従属変数
1	A	3
2	A	3
3	A	4
4	A	4
5	B	6
6	B	7
7	B	8
8	B	9
9	C	7
10	C	6
11	C	5
12	C	4

ここで群ごとの効果を表現したいとすると，次のように書くことになります。

$$
\begin{aligned}
y_1 &= \beta_0 + \beta_1 + e_1\\
y_2 &= \beta_0 + \beta_1 + e_2\\
y_3 &= \beta_0 + \beta_1 + e_3\\
y_4 &= \beta_0 + \beta_1 + e_4\\
y_5 &= \beta_0 + \beta_2 + e_5\\
y_6 &= \beta_0 + \beta_2 + e_6\\
y_7 &= \beta_0 + \beta_2 + e_7\\
y_8 &= \beta_0 + \beta_2 + e_8\\
y_9 &= \beta_0 + \beta_3 + e_9\\
y_{10} &= \beta_0 + \beta_3 + e_{10}\\
y_{11} &= \beta_0 + \beta_3 + e_{11}\\
y_{12} &= \beta_0 + \beta_3 + e_{12}
\end{aligned}
$$

添字の対応に注意しながらみてくださいね。群 A の効果は β_1，群 B の効果は β_2，群 C の効果は β_3 になります。

この β それぞれを該当するところ (割り当てられた群) だけに対応させつつ，統一的表現形である $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e}$ にするには，次のように書く必要があります。

$$
\begin{pmatrix} 3\\3\\4\\4\\6\\7\\8\\9\\7\\6\\5\\4 \end{pmatrix}
=
\begin{pmatrix}
1 & 1 & 0 & 0\\
1 & 1 & 0 & 0\\
1 & 1 & 0 & 0\\
1 & 1 & 0 & 0\\
1 & 0 & 1 & 0\\
1 & 0 & 1 & 0\\
1 & 0 & 1 & 0\\
1 & 0 & 1 & 0\\
1 & 0 & 0 & 1\\
1 & 0 & 0 & 1\\
1 & 0 & 0 & 1\\
1 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix} \beta_0\\ \beta_1\\ \beta_2\\ \beta_3 \end{pmatrix}
+
\begin{pmatrix} e_1\\e_2\\e_3\\e_4\\e_5\\e_6\\e_7\\e_8\\e_9\\e_{10}\\e_{11}\\e_{12} \end{pmatrix}
$$

このような表記になった時の $\boldsymbol{X}$ のことをとくに，実験のデザインを表している行列ということで，**デザイン行列 (design matrix)** といいます。デザイン行列は自分で書くと面倒な感じがしますが，ともかくこのような書き方で $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e}$ の統一表現は可能です。

ところでこのデザイン行列，$\boldsymbol{X}$ のサイズは今回 $n \times (m+1)$ になっていますね (m は水準数)。2 水準のときは 2 列で済んだものが，3 水準になると 4 列になるのはおかしくないですか？そうです，1 つ大事なポイントを忘れていました。各群の値はベースライン β_0 からの相対的な違いです。相対的な違いというのは，言い換えると $\sum \beta = 0$，すなわち全部の水準の和が 0 である必要があるのです。この式は今回の例だと $\beta_1 + \beta_2 + \beta_3 = 0$ であり，これを移項すると明らかなように $\beta_3 = 0 - \beta_1 - \beta_2$ です。つまり総和が決まっているので，自由に大きさを推定できるのは水準数 -1 になります。

これを踏まえてデザイン行列を次のように書き換えることができます。

$$\begin{pmatrix}3\\3\\4\\4\\6\\7\\8\\9\\7\\6\\5\\4\end{pmatrix}=\begin{pmatrix}1&1&0\\1&1&0\\1&1&0\\1&1&0\\1&0&1\\1&0&1\\1&0&1\\1&0&1\\1&-1&-1\\1&-1&-1\\1&-1&-1\\1&-1&-1\end{pmatrix}\begin{pmatrix}\beta_0\\\beta_1\\\beta_2\end{pmatrix}+\begin{pmatrix}e_1\\e_2\\e_3\\e_4\\e_5\\e_6\\e_7\\e_8\\e_9\\e_{10}\\e_{11}\\e_{12}\end{pmatrix}$$

この式は先ほどの式と内容的には同じで，表現の仕方が違うだけですが，$\sum\beta=0$ の条件がなければ計算結果は算出されません。計算するための制約が少ないと，答えが出なくなるのです。$\sum\beta=0$ の制約条件を別途書き加えてもいいですが，制約条件も含めた行列の書き方ができるというわけですね。

線形モデルの統一的表現，あるいは行列の計算方法に少しは慣れてきたでしょうか。これがさらに多くの変数，未知数を扱うことになると，その利点はよりはっきりしてきます[*4]。

7.4 因子分析モデルの行列表現

ということで，因子分析モデルの代数的表現ですが，これも行列を使って表現すると非常にシンプルに表現できるということを説明していきましょう。

内容はまったく同じですが，確認しておきましょう。標準得点行列 $\boldsymbol{Z}$ を因子負荷行列 $\boldsymbol{A}$ と因子得点行列 $\boldsymbol{F}$ をつかって，次のように表します。

$$\boldsymbol{Z}=\boldsymbol{F}\boldsymbol{A}'+\boldsymbol{U}\boldsymbol{D} \tag{7.1}$$

ここで，各行列の要素のサイズ感をつかんでおきましょう。まず $\boldsymbol{R}$ というのは相関行列ですから，m 個項目があるのでサイズは $m\times m$ の正方行列になります。次に $\boldsymbol{F}$ ですが，これは因子得点の行列です。得点は人数分ありますから行は n，因子の数が列になるのでこれを p とすると $n\times p$ です。$\boldsymbol{A}$ は因子負荷行列。因子負荷行列は因子の数と項目の数の組み合わせだけあるわけですから，$m\times p$ になりますね。$\boldsymbol{U}$ は独自因子得点です。得点ですから人数分，独自性分は各項目にありますから，サイズとしては $n\times m$ になります。最後に $\boldsymbol{D}$ ですが，これは独自因子の負荷量です。項目の数だけあるのですが，列ベクトルや行ベクトルで表現すると計算の時にサイズが変わって不便なことになります。ですから，対角項に d_j をもつ正方行列 $m\times m$ として表現しています。

サイズを確認したところで，実際に行列計算をしてみましょう。

[*4] ここでは触れませんが，被験者内計画・反復測定になっても行列表現はできます。個人差を表すデザイン行列を別途加えることになります。混合計画になると非常に複雑になりますが，それでも一般的な表現は可能です。

$$
\begin{aligned}
\boldsymbol{R} &= \frac{1}{N}\boldsymbol{Z}'\boldsymbol{Z} \\
&&& \boldsymbol{Z}\text{ を因子分析のモデル式にして} \\
&= \frac{1}{N}(\boldsymbol{FA}'+\boldsymbol{UD})'(\boldsymbol{FA}'+\boldsymbol{UD}) \\
&&& \text{前の項の転置を中に入れます} \\
&= \frac{1}{N}\left\{(\boldsymbol{FA}')'+(\boldsymbol{UD})'\right\}(\boldsymbol{FA}'+\boldsymbol{UD}) \\
&&& \text{転置のカッコを外すときは順番を入れ替えて転置} \\
&= \frac{1}{N}(\boldsymbol{AF}'+\boldsymbol{D}'\boldsymbol{U}')(\boldsymbol{FA}'+\boldsymbol{UD}) \\
&= \frac{1}{N}\boldsymbol{AF}'\boldsymbol{FA}'+\frac{1}{N}\boldsymbol{AF}'\boldsymbol{UD}+\frac{1}{N}\boldsymbol{D}'\boldsymbol{U}'\boldsymbol{FA}'+\frac{1}{N}\boldsymbol{D}'\boldsymbol{U}'\boldsymbol{UD}
\end{aligned} \tag{7.2}
$$

と，このように展開できました。記号を見ているとわかりにくいので，サイズ感を確認しましょう。最終的には，次のようになっています。

$$
\underset{m\times m}{\boldsymbol{R}} = \frac{1}{N}\underset{m\times p}{\boldsymbol{A}}\underset{p\times n}{\boldsymbol{F}'}\underset{n\times p}{\boldsymbol{F}}\underset{p\times m}{\boldsymbol{A}'} + \frac{1}{N}\underset{m\times p}{\boldsymbol{A}}\underset{p\times n}{\boldsymbol{F}'}\underset{n\times m}{\boldsymbol{U}}\underset{m\times m}{\boldsymbol{D}} + \frac{1}{N}\underset{m\times m}{\boldsymbol{D}'}\underset{m\times n}{\boldsymbol{U}'}\underset{n\times p}{\boldsymbol{F}}\underset{p\times m}{\boldsymbol{A}'} + \frac{1}{N}\underset{m\times m}{\boldsymbol{D}'}\underset{m\times n}{\boldsymbol{U}'}\underset{n\times m}{\boldsymbol{U}}\underset{m\times m}{\boldsymbol{D}}
$$

ここで，要素ごとに計算していた時のことを思い出してください。第二項 $\frac{1}{N}\boldsymbol{AF}'\boldsymbol{UD}'$ と第三項 $\boldsymbol{D}'\boldsymbol{U}'\boldsymbol{FA}'$ の中にある，$\boldsymbol{F}'\boldsymbol{U}$ と $\boldsymbol{U}'\boldsymbol{F}$ のところは，共通因子得点と独自因子得点の積ですし，いずれも標準化されていますから，$\frac{1}{N}$ と合わせて考えると，これは相関係数を表していることになります。また，共通因子と独自因子は相関しませんので，これはイコール $\boldsymbol{0}$ となり，この 2 つの項が消えてしまうのでした。

また，第一項の $\frac{1}{N}\boldsymbol{F}'\boldsymbol{F}$ は，共通因子同士の相関を表しています。$\boldsymbol{F}'\boldsymbol{F}=\boldsymbol{C}$ とすると，これは因子得点間相関 $\boldsymbol{C}$ を表すことになります。これが直交であると仮定する，つまり他の因子と相関しないと考えると，$\boldsymbol{C}=\boldsymbol{I}$，つまり単位行列です。単位行列は計算に影響を与えませんから，$\boldsymbol{AF}'\boldsymbol{FA}'=\boldsymbol{ACA}'=\boldsymbol{AIA}'=\boldsymbol{AA}'$ となり，この式は簡単に次のように変形できます。

$$
\boldsymbol{R}=\boldsymbol{AA}'+\boldsymbol{D}^2 \tag{7.3}
$$

先ほどの代数的展開を，そのまま行列で表現しただけですが，この方がシンプルに表現できていますね。この表現は，因子分析の第一定理と第二定理の両方を含んで一度に表せているのです。

いかがでしょうか。行列表現の便利さがわかっていただければ，と思います。しかしこれでもまだ謎は残りますね。我々が追っている謎は，行列からどのようにして因子負荷量を計算するのか，です。それを知るためには，もう 1 つ線形代数から明らかになる特徴を知らなければなりません。次回をどうぞお楽しみに。

7.5 課題

■行列計算を確認しておこう $\boldsymbol{V}=(\boldsymbol{I}-\frac{1}{n}\boldsymbol{11}')\boldsymbol{X}$ の要素を書き下してみよう。平均偏差行列ができているでしょうか。

■行列計算を確認しておこう 2 $\boldsymbol{S},\boldsymbol{Z},\boldsymbol{R}$ も，面倒でも要素レベルまで書き下してみよう。

■行列計算を確認しておこう 3 因子分析モデルの行列計算の結果出てくる，$\boldsymbol{R}=\boldsymbol{AA}'+\boldsymbol{D}^2$ の要素を確認し，エレメントワイズで表現していた式との対応を確認しよう。

第 8 章

固有値と固有ベクトルと因子分析モデルの関係

8.1　固有値と固有ベクトル

今回は正方行列にみられるおもしろい特徴である，**固有値** (eigenvalue) と**固有ベクトル** (eigenvector) についての話から始めます。ある正方行列 $\boldsymbol{A}$, 列ベクトル $\boldsymbol{x}$, スカラー λ が次のような関係にあった時，λ を固有値，$\boldsymbol{x}$ を固有ベクトルと言います。

$$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$$

一見すると，$\boldsymbol{x}$ が両辺に入っていますから，$\boldsymbol{A}$ が λ に置き換わった等式に見えます。しかし一方は行列で，他方はスカラーです。こんな奇妙なことが本当にあるのでしょうか？具体的な数値例をみてみましょう。

$$\boldsymbol{A} = \begin{pmatrix} 1 & 6 \\ 2 & 5 \end{pmatrix}, \boldsymbol{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

を例にします。この時次の関係が成り立ちます。

$$\boldsymbol{A}\boldsymbol{x} = \begin{pmatrix} 1 & 6 \\ 2 & 5 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 7 \\ 7 \end{pmatrix} = 7\begin{pmatrix} 1 \\ 1 \end{pmatrix} = 7\boldsymbol{x}$$

確かに成立する組み合わせがありますね。この行列 $\boldsymbol{A}$ に対して，7 が固有値，$(1, 1)$ が固有ベクトルになっています。また，実はこの行列 $\boldsymbol{A}$ については，-1 も固有値であり，そのときの固有ベクトルは $(-3, 1)$ も固有ベクトルです。

この固有値分解こそ，因子分析を元とする多変量解析の中心的な数学原理なのです。多変量解析の世界においては，分散共分散行列やそれを標準化した相関行列など，変数同士の関係を分析のスタートにおくのでした。これらの行列は正方行列ですから，その固有値や固有ベクトルを計算することで正方行列の特徴を別の視点から分解して考えられるようになります。

8.1.1　固有値の特徴

この固有値の数学的特徴は色々あるのですが，データ分析をする上で重要な点を押さえておきましょう。

固有値の特徴として，**固有値の総和が正方行列の対角要素の総和に合致する**，というのがあります。数式で表現すると，次のようになります。

$$\sum_{i=1}^{N} \lambda_i = trace(\boldsymbol{A}) = \sum_{i=1}^{N} a_{ii}$$

ここで a_{ij} は行列 $\boldsymbol{A}$ の要素であり，a_{ii} は i 行 i 列目，つまり対角要素です。この正方行列 $\boldsymbol{A}$ のサイズは N で，対角要素の総和をとくに**トレース (trace)** といい $trace(\boldsymbol{A})$ と表します。それが固有値の総和とイコールになる，ということを表しています。サイズ N の正方行列からは固有値が N 個算出できることがわかっており，それをすべて足し合わせたものがトレースと同じになっているのですね。先ほどの例で言えば，$\boldsymbol{A}$ のトレースは $1 + 5 = 6$ で，固有値の総和は $7 - 1 = 6$ であり，確かにこの関係が成立していることがわかります。

分散共分散行列のトレースは，分散の総和を意味します。項目同士の関係を表した行列であれば，分散はその項目から得られる情報の大きさであり，それを総和するということは，その調査研究・項目群から得られる情報の総和であると言ってもいいでしょう。相関行列のトレースは，対角項に入っているのが $r_{ii} = 1.0$ ですから，項目の数と一致します。1 つの項目の情報量を 1.0 に基準化して N 項目分の情報がある，ということを表しています。

固有値と行列の関係は冒頭で示したように，$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$ であり，正方行列の特徴をスカラーにしてしまうというものです。得られる N 個の固有値は，元の正方行列のエッセンスをスカラーにして表現しているわけです。

ところで $\boldsymbol{A}$ を n 次正方対称行列，つまり $n \times n$ サイズの対称行列だとすると，n 個の固有値が求められます。これを $\lambda_1, \lambda_2, \cdots, \lambda_n$ として，対応する固有ベクトルを $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$ とします。ここで各固有ベクトルのノルムが 1 であるとしましょう。行列と固有値・固有ベクトルの関係から，

$$\boldsymbol{A}\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$$

となりますが，このベクトルを並べた行列 $\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{pmatrix}$ を考えると，

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{\Lambda}$$

と書くことができます。ここで $\boldsymbol{\Lambda}$ は

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

のような行列です。

この両辺に $\boldsymbol{X}'$ をかけると

$$\boldsymbol{A}\boldsymbol{X}\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}'$$

となりますが，固有ベクトルの性質とノルムを整えていることから $\boldsymbol{X}\boldsymbol{X}' = \boldsymbol{I}$ であり，そこから

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}'$$

と書くことができます。

ここであらためて要素に注目すると，行列 $\boldsymbol{A}$ が次のように分解されていることがわかります。

$$\boldsymbol{A} = \lambda_1 \boldsymbol{x}_1 \boldsymbol{x}_1' + \lambda_2 \boldsymbol{x}_2 \boldsymbol{x}_2' + \cdots \lambda_m \boldsymbol{x}_m \boldsymbol{x}_m' = \sum \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i'$$

となります。

この分解例は 2×2 の簡単な例で確認しておきましょう。たとえば $\boldsymbol{A} = \left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} c \\ d \end{pmatrix} \right) = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ という行列と，対角行列 $\boldsymbol{\Psi} = \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix}$ があったとして，$\boldsymbol{A\Psi A}'$ の計算をしてみたいと思います。

$$
\begin{aligned}
\boldsymbol{A\Psi A}' &= \begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \\
&= \begin{pmatrix} \alpha a & \beta c \\ \alpha b & \beta d \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \\
&= \begin{pmatrix} \alpha aa + \beta cc & \alpha ab + \beta cd \\ \alpha ab + \beta cd & \alpha bb + \beta dd \end{pmatrix} \\
&= \alpha \begin{pmatrix} aa & ab \\ ab & bb \end{pmatrix} + \beta \begin{pmatrix} cc & cd \\ cd & dd \end{pmatrix} \\
&= \alpha \begin{pmatrix} a \\ b \end{pmatrix} \begin{pmatrix} a & b \end{pmatrix} + \beta \begin{pmatrix} c \\ d \end{pmatrix} \begin{pmatrix} c & d \end{pmatrix}
\end{aligned}
$$

と，このようにスカラーとベクトルの積和の形に書き換えられるのですね[*1]。

さて，これらをまとめて考えると，

1. 固有値分解は行列を列ベクトルとその転置ベクトルの積の形に分解する。
2. 固有値の総和は元の行列の対角要素の総和である。
3. 元の行列の対角要素は各項目の分散を表している。

ということですから，固有ベクトルは全体の情報量をそのままに重要度の大きさに並べ替えたもの，固有値分解は行列をその要素の重要度ごとに分解していくことである，といえます。これこそ**因子分析**で取り出そうとしている因子であり，固有値の大きさはその因子の重要度として，共通次元の判別 (どこまで共通次元とみなすか) に使われるのです。

8.2 固有値と固有ベクトルを求める

ここで少し数学の方に話を戻して，固有値と固有ベクトルの計算方法を考えましょう。元の式を書き換えて次のような方程式を考えます。

$$(\boldsymbol{A} - \lambda \boldsymbol{I})\boldsymbol{x} = \boldsymbol{0}$$

固有ベクトルは $\boldsymbol{x} = \boldsymbol{0}$ すなわち全部ゼロであれば当然成り立ちますから (自明な解といいます)，これは除外することにします ($\boldsymbol{x} \neq \boldsymbol{0}$)。行列の表現は連立方程式の解を求めることと同じなのでした。$\boldsymbol{A} - \lambda \boldsymbol{I}$ を連立方程式の係数行列だと考えれば，それが**逆行列**を持つと左辺にそれをかけてしまえば全部ゼロの答えになってしまいますから，そうでない答えを求めるには，この係数行列が逆行列を持たないことが重要です。

[*1] ただし，この計算が可能なのは分解する元の行列が実対称行列だからです。実対称行列は固有値と固有ベクトルで対角化可能であることが証明できます。実対称行列の固有値は全て十数ですし，固有値が全て実数であれば適当な直交行列をつかって対角化でき，実対称行列の固有ベクトルは互いに独立するのでこれらを使って直交行列を作ることができるからです。これらの性質については線形代数のテキストなどの証明を参照してください。また計算プロセスからもわかるように，同じ要素を持つ列ベクトルと行ベクトルの積ですから，結果は対称になってしまうからです。

さて，この授業の中では説明してきませんでしたが，方程式が解を持つかどうかを決定する計算方法があります。これを**行列式 (determinant)** といい[*2]，この値がゼロでなければその方程式は解を持つ，ということがわかっています。説明しなかったのは，この値を求める計算がとても面倒だからで，詳しくは線形代数のテキストにお任せするとして[*3]，ここでは簡便のために 2×2 方程式の行列について紹介します。

2×2 の係数行列，$\boldsymbol{P} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ の逆行列は次の式で求められることがわかっています。

$$\boldsymbol{P}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

この式から考えると，$ad - bc$ のところが 0 になるとこの計算はできませんから，逆行列が存在しないことになります。この $ad - bc$ にあたるところが行列式であり，$|\boldsymbol{P}|$ とか $det(\boldsymbol{P})$ のように表します。$ad - bc$ が 0 でなければ方程式は解けるのですが，今回の場合は解けると自明になってしまうので困ります。今回は $ad - bc = 0$ でなければならないのです。つまり一般的に書くと次のようになります。

$$|\boldsymbol{A} - \lambda \boldsymbol{I}| = 0$$

$\boldsymbol{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ とすると，この式は次のようになります。

$$\begin{vmatrix} a - \lambda & b \\ c & d - \lambda \end{vmatrix} = 0$$

$$(a - \lambda)(d - \lambda) - bc = 0$$

この方程式をとくに**固有方程式**といいますが，これを解いてやれば良いことになりますね。具体的に $\begin{pmatrix} 1 & 6 \\ 2 & 5 \end{pmatrix}$ の例で計算してみましょう。

$$\begin{vmatrix} 1 - \lambda & 6 \\ 2 & 5 - \lambda \end{vmatrix} = 0$$

$$(1 - \lambda)(5 - \lambda) - 12 = 0$$

$$\lambda^2 - 6\lambda - 7 = 0$$

$$(\lambda - 7)(\lambda + 1) = 0$$

ここから $\lambda = 7, -1$ が得られますね。

では固有ベクトルはどうなるでしょうか。固有値 7 の例で計算してみます。

$$\begin{pmatrix} 1 & 6 \\ 2 & 5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 7 \begin{pmatrix} x \\ y \end{pmatrix}$$

$$x + 6y = 7x \to 6x = 6y \to x = y$$

$$2x + 5y = 7y \to 2x = xy \to x = y$$

あれっ？ なんじゃこりゃ？ と思った人もいるかもしれません。でもこれ，間違いではありません。実は固有ベクトルは大きさが定まっておらず，今回の例で言えば $x = y$ つまり $x : y = 1 : 1$ の関係であればあらゆる

[*2] 行列式は数値であり，解が求まるかどうか決定 determinant する，という意味なのに，日本語訳はなぜか「式」といいます。変なの。

[*3] たとえば村上他 (2016) の第 3 章をみてください。

値が成立してしまうのです。固有ベクトルが $(1,1)$ でも $(2,2)$ でも $(100,100)$ でも，$\boldsymbol{Ax}=\lambda\boldsymbol{x}$ の関係に影響しませんから，普通はベクトルの長さ (**ノルム (norm)**) を 1.0 に規格化するという方策が取られます[*4]。先ほど「固有ベクトルを適当な大きさに選んでやれば」というような表現をしましたが，それはベクトルの大きさはいくらでもいいからできることなのですね。

さてここでみたように，2×2 の方程式であれば固有方程式を解くことはできるのですが，行列のサイズがどんどん大きくなると一般的に解けなくなって行くことは想像にかたくないと思います。実際我々は正方行列として，項目の情報が詰まった分散共分散行列とか相関行列を使いますから，それが 2 項目しかないなんてことはなくて，もっともっと大きなサイズになります。そうすると計算機を使って近似的に答えを求めて行くことになります。

8.3 固有値と固有ベクトルの幾何学的意味

固有値，固有ベクトルについて，今度は違う側面から見直してみましょう。

ある正方行列から固有値 λ と固有ベクトル $\boldsymbol{a}$ が得られたとします。このベクトル $\boldsymbol{a}$ のすべての要素を定数 c 倍したベクトル $\boldsymbol{b}=c\boldsymbol{a}$ を考えると，これもやはり同じ関係が成り立ちます。

$$\boldsymbol{Ab}=\lambda c\boldsymbol{a}=\lambda\boldsymbol{b} \tag{8.1}$$

先ほどの計算でも明らかになりましたが，固有ベクトルの値は絶対的なものではなく，要素間の相対的大きさを反映しているに過ぎないのでしたね。

さてこれを幾何学的に，図形として考えてみましょう。要素が 2 つのベクトルは，2 次元座標に表現できます。ベクトル $\boldsymbol{x}=(x,y)$ という座標を表しているというわけです。固有ベクトルも要素が 2 つであれば，座標で表現できます。先ほどの，要素を c 倍しても固有ベクトルとしての性質は変わらない，という話は，「固有ベクトルは大きさに意味はなく，方向を表したもの」ということになります。では何の方向を指し示しているのでしょうか。

固有値と固有ベクトルの話の最初にあった，$\boldsymbol{Ax}=\lambda\boldsymbol{x}$ というのを見直してみましょう。$\boldsymbol{x}$ がなんらかの座標を表していると考えると，それに正方行列をかけるとはどういう意味でしょうか。次の計算式を見てください。

$$\boldsymbol{Ax}=\begin{pmatrix}2&0\\0&3\end{pmatrix}\begin{pmatrix}1\\2\end{pmatrix}=\begin{pmatrix}2\\6\end{pmatrix} \tag{8.2}$$

これをみると，座標 $\boldsymbol{x}=\begin{pmatrix}1\\2\end{pmatrix}$ に $\boldsymbol{A}$ をかけたことで，座標が $\begin{pmatrix}2\\6\end{pmatrix}$ に変わった，と見ることもできますね。このように，ある座標が別の座標に移ることをとくに「変換」と呼びます[*5]。つまり**正方行列はなんらかの変換を施すもの**だ，と考えることができます。

今回の例では，行列 $\boldsymbol{A}$ には次のような性質があります。

$$\begin{pmatrix}2&0\\0&3\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=2\begin{pmatrix}1\\0\end{pmatrix} \tag{8.3}$$

$$\begin{pmatrix}2&0\\0&3\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}=3\begin{pmatrix}0\\1\end{pmatrix} \tag{8.4}$$

[*4] ノルムとは，要素の二乗和の平方根，$\sqrt{x_1^2+x_2^2\cdots x_n^2}$ のことです。

[*5] より一般的にいうと，以下のようになります。: 集合 X の各元 x に集合 Y の元 $f(x)$ を対応させる対応 f のことを，集合 X から集合 Y への写像 (mapping)，関数 (function)，あるいは変換 (transformation) という。

そう, お気づきのように, これは固有値・固有ベクトルです。この行列の固有値・固有ベクトルはそれぞれ $\lambda_1 = 2, \boldsymbol{x}_1 = (1,0)$, $\lambda_2 = 3, \boldsymbol{x}_2 = (0,1)$ であることがわかります。この固有値, 固有ベクトルの組み合わせをじっとみていると, おもしろい特徴がわかって来ます。

今回の行列から得られた 2 つの固有ベクトル, $(1,0)$ と $(0,1)$ は, 2 次元平面の単位ベクトルと呼ばれるものです。**2 次元座標の任意の点は, これら 2 つのベクトルの任意の線型結合で表現できます**。座標 (a,b) は $a \times (1,0)$ と $b \times (0,1)$ からなるベクトルですから, $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$ というように, です。

このように, 単位ベクトルは 2 次元世界の基礎となる単位ともいうべきもので, ここで $(1,0)$ は x 座標の, $(0,1)$ は y 座標の基盤となるベクトルであるということができます (これをとくに**基底**といいます)。つまり, 正方行列 $\boldsymbol{A}$ から得られる固有ベクトルは, その正方行列が作る空間の基盤を明らかにするものであったのです。

では固有値はどうでしょうか？ 今回は x 座標を 2 倍, y 座標を 3 倍に引き延ばす変換をしたわけですが, この座標の歪み (重み) が固有値に対応していますね。つまり, 固有値と固有ベクトルは新しい座標に変換する, その変換先の空間的性質を表していることになります。元の座標空間は $(1,0),(0,1)$ で作られる空間ですが, 変換先の空間は $(2,0),(0,3)$ で作られている空間, ということになります。

つまり, 正方行列は空間を変換するもの, あるいは正方行列の中に固有ベクトルを基底とした空間があるもの, ということです。

すべての正方行列に, こういった「変換」という解釈ができるのであれば, 相関行列にも同様のことがいえるでしょう。相関行列は正方行列ですので, 固有値分解できるのです。相関行列を固有値分解することは, 相関行列の中に潜む次元 (dimension) を抽出してくることです。固有ベクトル (因子負荷行列) は, 正方行列によって変換される, 変換先の単位ベクトルのことだったのです。そして, 固有値はその次元のゆがみ (重み, 重要性) という意味があったのです。

8.4 因子分析の数学的理解

さあ因子分析に戻って考えてみましょう。因子分析のモデルは次のようなものでした。

$$\boldsymbol{R} = \boldsymbol{A}\boldsymbol{A}' + \boldsymbol{D}^2 \tag{8.5}$$

ここで, 左辺の正方行列を相関行列 $\boldsymbol{R}$ とし, $\lambda \boldsymbol{x}\boldsymbol{x}' = \boldsymbol{a}\boldsymbol{a}'$ となるようにベクトルの大きさを整えてみましょう。これは λ を分解してベクトルの中に溶け込ませるようなものですから, $\boldsymbol{a} = \sqrt{\lambda}\boldsymbol{x}$ とすればよいでしょう。すると相関行列は次のように分解できます。

$$\boldsymbol{R} = \boldsymbol{a}_1\boldsymbol{a}_1' + \boldsymbol{a}_2\boldsymbol{a}_2' + \cdots + \boldsymbol{a}_m\boldsymbol{a}_m' + \boldsymbol{d}\boldsymbol{d}' \tag{8.6}$$

$$\boldsymbol{R} = \boldsymbol{a}_1\boldsymbol{a}_1' + \boldsymbol{a}_2\boldsymbol{a}_2' + \cdots + \boldsymbol{a}_m\boldsymbol{a}_m' + \boldsymbol{d}\boldsymbol{d}' \tag{8.7}$$

ここでは共通因子の数が m 個だとわかっている体で分解していますが, 基本的にはサイズ N の行列からは N 個の固有値がずらーっと並ぶわけです。それをどこかで「共通しているのはここまで」と判断し, 残りは誤差であるとしてまとめて $\boldsymbol{d}\boldsymbol{d}'$ にしているだけですね。このように数学的にはここからが共通因子, ここからが独自因子といった区別をすることなく, 最後のひとかけらまで固有値分解を行なっているのですが, その次元の重要度でもって共通因子と誤差因子に (研究者が恣意的に) 分割しているのが因子分析のやっていることなのです。

一般に, N よりも m のほうがグッと少なくなります。たとえば YG 性格検査では $N = 120$ であり, m はせいぜい 5 から 10 数個です。120 項目のつくる 120 次元空間の中で, そこに働きかけても方向の変わらない基礎的な少数の次元にのみ注目すれば, 効率よく情報圧縮ができるというものです。

相関係数を固有値分解すると，その固有値はすべて足し合わせるとサイズ N になるのでした。元のデータから計算される相関行列は，1 つの項目が一単位分の情報を持っていると考えますが，固有値分解はそれを次元の重要性順に並べ替えます。固有値は項目いくつ分の重要度があるかということを表す指標だと考えることができます。どこから共通因子でどこからが誤差か，ということを考えるときに，たとえば固有値が 1.0 よりも小さくなるようであれば，項目 1 つ分の情報もないのだからということで誤差因子だと判断することがあります。

ところで因子分析モデルの $\boldsymbol{A}$，因子負荷行列ですが，これは共通因子の固有ベクトルをセットで扱ったものです。つまり，$\boldsymbol{A} = \boldsymbol{a_1}, \boldsymbol{a_2}, \cdots, \boldsymbol{a_m}$ と縦ベクトルを並べたものになっています。エレメントワイズで表現すると次のようになります。

$$\boldsymbol{A} = \left(\begin{pmatrix} a_{11} \\ a_{12} \\ \vdots \\ a_{1N} \end{pmatrix}, \begin{pmatrix} a_{21} \\ a_{22} \\ \vdots \\ a_{2N} \end{pmatrix}, \cdots, \begin{pmatrix} a_{m1} \\ a_{m2} \\ \vdots \\ a_{mN} \end{pmatrix}, \right)$$

ここで $\boldsymbol{AA'}$ の間に単位行列 $\boldsymbol{I}$ を挟んでも，別に結果は変わりませんよね。単位行列はかけても変わらないのが特徴ですから，$\boldsymbol{AA'} = \boldsymbol{AIA}$ です。ここでの $\boldsymbol{I}$ のサイズは $m \times m$ であることに注意しつつ聞いてください。

$\boldsymbol{I} = \boldsymbol{TT'} = \boldsymbol{T'T}$ になるような m 次の行列を使うと，$\boldsymbol{AA'} = \boldsymbol{ATT'A'}$ の関係は保たれたままです。この $\boldsymbol{T}$ はこれまた行列の座標を変えてしまう変換行列であり，これを挟むことができるということは**因子負荷量の値はなんでもあり**だということになってしまいます。この変換行列 $\boldsymbol{T}$ はとくに**回転行列 (rotation matrix)** とも呼ばれますが，因子分析はこのようにどんな回転でもできる，**回転に関する不定性**があるのです。あらゆる因子負荷量の組み合わせがあり得る，というのは困りものなので，なんらかの形で因子負荷行列 $\boldsymbol{A}$ に制約をかける必要があります。言い方を変えれば，色々な制約の中ではあっても好きな値を取ることができますから，ユーザにとって便利な基準を考えてやれば良いでしょう。因子分析では一般に，**因子軸の回転**を行いますが，それはこうした理由からです。

ちなみに $\boldsymbol{TT'}$ に $diag(\boldsymbol{T\Phi T'}) = \boldsymbol{I}_m$ という制約のある行列 $\boldsymbol{\Phi}$ を挟んでやっても，元の計算モデルに影響はありません。この $\boldsymbol{\Phi}$ は**因子間相関 (factor correlations)** とよばれ，相関を持った因子軸の回転，すなわち**斜交回転 (oblique rotation)** も色々なものが考えられています[*6]。

ずいぶんややこしい話になってきました。この後は実践形式で因子分析を理解していければと思います。

8.5 課題

■固有値問題 行列 $\begin{pmatrix} 8 & 1 \\ 4 & 5 \end{pmatrix}$ の固有値・固有ベクトルを求めよ。

[*6] これに対して $\boldsymbol{TT'} = \boldsymbol{I}$ のような因子間相関がない (単位行列) な回転を**直交回転 (orthogonal rotation)** といいます。

第 9 章

R をつかっての行列計算

9.1　R による行列計算

今回は統計環境 R をつかって，演習によってこれまで習ったことを整理していきましょう。

9.1.1　環境の準備 (確認)

まずは環境の準備です。すでに R や RStudio のインストールは終わっているものとして話を進めます[*1]。次の 3 つのステップをたどって，実行の準備をしてください。

■RStudio の起動　RStudio を起動してください。パッケージのインストールをするときなど，管理人権限が必要になるケースがあります。

■プロジェクトを開く　RStudio を使う時の基本は，プロジェクトによる管理です。分析するデータ、テーマ、内容によってプロジェクトを切り替えながら使います。メニューバーから `File > New Project` と進んでください。すると「New Directory」「Existing Directory」「Version Control」の選択肢が出てきます。New Directory は新しいディレクトリ (=フォルダ) を作ってそこで関連ファイルをまとめますよ，という意味です。Existing Directory はすでに存在するディレクトリ (=フォルダ) をまとめる場所にしますよ，という意味です。みなさんの環境に応じて使い分けてください。すでにプロジェクトが存在する場合は，プロジェクトを開く (FIle→Open Project) から選択します。RStudio の右上などで，プロジェクトが開いていることを確認しましょう。

■R スクリプトを開く　今日のコードを書くための R スクリプトファイルを準備します。File > NewFile > R Script と進み，何も書いてない R スクリプトの画面を表示させてください。真っ白いファイルが開いたら問題ありません。

これでスクリプトのところにコードを書いていく準備ができました。

9.1.2　R におけるデータの型

統計環境 R は Excel や Numbers や Calcs など[*2]の表計算ソフトとは違って，データを行列・ベクトルとして扱うことができます。たとえば次のコードを実行してみてください。

[*1] まだだよー！という人は RStudio インストールといったキーワードで検索すると，親切にも案内してくれているサイトに色々出会えるでしょう。

[*2] Excel は Microsoft Office に含まれる表計算ソフト。Numbers は Apple の Mac や iPad，iPhone で使える表計算ソフト，Calcs は Libre Office というフリーソフトウェアの表計算ソフトです。

code : 9.1 ベクトルデータの保持

```
1 A <- 1:9
2 B <- c(3, 4, 5)
3 C <- matrix(A, ncol = 3)
4 D <- matrix(A, ncol = 3, byrow = T)
```

■コード解説

1 行目 `1:9` をオブジェクト A に代入しています。ここでコロン`:` は連続した数字を表しており，`c(1,2,3,4,5,6,7,8,9)` と同じ意味です。

2 行目 要素 3, 4, 5 からなるベクトルを作って B に代入しています。`c()` は結合する (combine) という関数です。

3 行目 9 つの要素があるベクトルを持ったオブジェクト A を `matrix` 型に変更しています。その時の列数は 3(`ncol=3`) です。

4 行目 同じくオブジェクト A を `matrix` 型に変換，列数は 3 ですが `byrow = T` で「行ごとに並べる」スイッチをオン (TRUE) にしています。

それぞれのオブジェクトに格納されているものを確認してみましょう。何がどうなっているかわかるでしょうか (出力 9.1)。

R の出力 9.1: R におけるベクトルと行列

```
> A
[1] 1 2 3 4 5 6 7 8 9
> B
[1] 3 4 5
> C
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> D
    [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

R のオブジェクトには色々な型がありますが，基本はベクトルです。数字のセットとして扱うのです。このときの R のベクトルに行・列の区別はありません。行列であることを明示するためには，`matrix` という関数を当てはめることになります。ベクトルに行，列の意味を持たせたければ，`matrix` 型で一列あるいは一行の行列である，という指定をしてください。

これまで，あるいは他の授業で学んだことのある R のオブジェクトの形としては，`list` 型が多かったのではないかと思います。`list` という形は，要素がベクトルでも数字でも文字列でもなんでもかまわない，というものでした。これを矩形 (長方形) に整え，変数名としての列名や，行番号としての行名をもっているものが `data.frame` 型です。`data.frame` 型は `list` 型の特殊系なわけです。`matrix` 型は矩形のオブジェクトではありますが，`data.frame` 型とは違います。`matrix` 型にも列名，行名をつけることはできますが，行列

であると明示してあるように，計算するときは行列演算の対象になるのです。data.frame 型は行列の四則演算はできません。

また一点注意が必要なこととして，R はベクトルの再利用をするということです。次の一行 (code:9.2) を実行して中身を見てみましょう。

code : 9.2 ベクトルデータの保持

```
1	E <- matrix(A, ncol = 3, nrow = 6)
2	G <- matrix(A, ncol = 3, nrow = 4)
```

■コード解説

1 行目 1:9 をオブジェクト E に代入している。ここで列数は 3，行数は 6 を指定している。

2 行目 1:9 をオブジェクト G に代入している。ここで列数は 3，行数は 4 を指定している。

R の出力 9.2: ベクトルの再利用

```
> E <- matrix(A, ncol = 3, nrow = 6)
> G <- matrix(A, ncol = 3, nrow = 4)
 警告メッセージ:
 matrix(A, ncol = 3, nrow = 4) で:
   データ長 [9] が行数 [4] を整数で割った、もしくは掛けた値ではありません
> E
     [,1] [,2] [,3]
[1,]    1    7    4
[2,]    2    8    5
[3,]    3    9    6
[4,]    4    1    7
[5,]    5    2    8
[6,]    6    3    9
> G
             [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6    1
[3,]    3    7    2
[4,]    4    8    3
```

オブジェクト A の要素数は 9 です。しかし行列 $\boldsymbol{E}$ を作る時に，3 行 6 列，すなわち 18 の要素が必要であることを要求しました。そうすると R は (勝手に and/or 親切にも？)A を再利用して足りないところを埋めてしまいます。今回はちょうど 2 回使えば埋まりましたので，エラーや警告もなくそのまま通っています。つくられた $\boldsymbol{E}$ の要素を確認し，再利用されていることをみてください。ちなみに $\boldsymbol{G}$ は 3 行 4 列，すなわち 12 の要素が必要であることを要求しました。そうすると R は警告を出して，再利用が途中で途切れますよと言ってくれます。とはいえ警告に過ぎないのでそのまま計算を続けることができ，作られた $\boldsymbol{G}$ には $1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3$ と 2 周目の最初の 3 要素だけ使われています[3]。

このように R は，可能な時には良かれと思って勝手に再利用してしまいますから，注意してくださいね。

[3] そんなことよりオブジェクトが A,B,C ときてるんだから E のつぎは F だろ，と思う人もいるかもしれません。F でもいいのですが，実は大文字の F と T はそれぞれ FALSE/TRUE の略語であり予約後語です。上書きして使うこともできますが，なるべく避けたほうがいいでしょう。ちなみに RStudio などでこれら一文字を使うと予約語になっているので表示の時にハイライトされます。

9.1.3　行列方と行列の演算

それでは R を使っての行列計算を進めましょう。

加法減法はサイズが同じでないとできない, という制約はありますが, 計算記号などに違いはありません。乗法は, スカラーとベクトル, スカラーと行列であれば普通に記述していただいて結構です。

code : 9.3　ベクトルの和･差, スカラーをかける

```
1  x <- 1:3
2  y <- 8:10
3  x + y
4  x - y
5  2 * x
6  y / 3
7  A <- matrix(c(1, 2, 3, 4), ncol = 2)
8  B <- matrix(c(5, 6, 7, 8), ncol = 2)
9  A + B
10 A * 3 + B * 2
```

■コード解説

1 行目　`1:3` をオブジェクト x に代入している。

2 行目　`1:9` をオブジェクト y に代入している。

3 行目　ベクトルの和 $(\boldsymbol{x}+\boldsymbol{y})$

4 行目　ベクトルの差 $(\boldsymbol{x}-\boldsymbol{y})$

5 行目　ベクトルにスカラーをかける $(2\boldsymbol{x})$

6 行目　ベクトルをスカラーで割る。スカラーの逆数をかけることと同じ。$(\boldsymbol{y}/3)$

7 行目　行列 $\boldsymbol{A}$ をつくる。サイズは 2×2

8 行目　行列 $\boldsymbol{B}$ をつくる。サイズは 2×2

9 行目　行列の和 $(\boldsymbol{A}+\boldsymbol{B})$

10 行目　行列にスカラーをかけ, それを足し合わせる。スカラー倍は各要素にかかる $(3\boldsymbol{A}+2\boldsymbol{B})$

出力結果はここでは提示しませんが, 皆さんそれぞれ計算できているか確認しておいてください。

続いて行列の乗法ですが, サイズが変わるような行列としての計算の場合はとくに `%*%` という記号を使うことになります。ただのアスタリスク*ではなく, それを `%` で囲むことで行列の積になっていることを表現しているのです。また行列の行列を反転させるのは, **転置**という操作になりますが, これは `t()` という関数を使います。行列の積を計算する例を `code:9.4` に示しました。計算結果が出力されますので, どういう計算をしているか一行ずつ確認しながら進めてください。

code : 9.4　行列の積

```
a <- c(1, 2, 1)
b <- c(3, 4, 2)
a * b
a %*% b
a %*% t(b)
A <- matrix(1:9, ncol = 3)
A * a
```

```
8  A %*% a
9  a %*% A
10 B <- matrix(1:6, nrow = 3, byrow = T)
11 C <- matrix(c(1, 0, 0, 1, 1, 1), ncol = 3)
12 B %*% C
13 B %*% t(C)
```

■コード解説

1 行目　ベクトル $\boldsymbol{a} = (1, 2, 1)$ を作る

2 行目　ベクトル $\boldsymbol{b} = (3, 4, 2)$ を作る

3 行目　掛け算記号*を使っているが，これはベクトルの掛け算ではなく要素ごとの掛け算を意味する。

4 行目　ベクトルの掛け算記号 %*% を使っており，$\boldsymbol{ab}$ の計算をしている。この時，ベクトル $\boldsymbol{a}$ は行ベクトル，$\boldsymbol{b}$ は列ベクトルと解釈されます。行列計算に適した形であり，デフォルトでは行方向が優位です。

5 行目　ベクトルの掛け算だが t() で転置，すなわち $\boldsymbol{b}'$ を行っており，行列に適した形に変換されて $\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 3 & 4 & 2 \end{pmatrix}$ と解釈されている。3×1 と 1×3 の計算なのでできあがる行列は 3×3 のサイズになる。

6 行目　行列 $\boldsymbol{A}$ を作る

7 行目　行列 $\boldsymbol{A}$ に対してベクトルをかけているように見えるが，ベクトルの掛け算でなく要素の掛け算記号*なので，各行を 1 倍，2 倍，1 倍する結果になっている。

8 行目　行列とベクトルの積 $\boldsymbol{Aa}$ であり，行列計算可能な 3×3 と 3×1 と解釈され，結果のサイズは 3×1 になっている。

9 行目　ベクトルと行列の積 $\boldsymbol{aA}$ であり，行列計算可能な 1×3 と 3×3 と解釈され，結果のサイズは 1×3 になっている。

10-11 行目　行列 $\boldsymbol{B}, \boldsymbol{C}$ を作る

12 行目　行列の積 $\boldsymbol{BC}$ を計算している。3×2 と 2×3 の行列の積なので，結果のサイズは 3×3 になっている。

13 行目　行列の積 $\boldsymbol{BC'}$ を計算しようとしているが，3×2 と 3×2 の積は計算できないので，エラーが返ってくる。

続いて**逆行列**の例を見てみましょう。逆行列の計算は，手計算でやると大変なのですが，R では関数 solve()) を使うと計算できます。

code : 9.5　逆行列の計算

```
1  A <- matrix(c(2,1,5,3),ncol=2)
2  solve(A)
3  A %*% solve(A)
```

■コード解説

1 行目　行列 $\boldsymbol{A}$ をつくる

2 行目　逆行列 $\boldsymbol{A}^{-1}$ の計算

3 行目　$\boldsymbol{AA^{-1}} = \boldsymbol{I}$ より，逆行列になっていたことの確認。

逆行列はかけると単位行列になるので，スカラーでいうところの逆数を意味します。逆数をかけると 1 になるように，行列の場合は逆行列をかけると単位行列 $\boldsymbol{I}$ になるのでした。これの何が嬉しいかというと，行列が連立方程式の係数を表していた場合，逆行列をかけることで連立方程式が解けるのでした (セクション 6.3, Pp.67 参照)。次の連立方程式を使って，実際に計算してみましょう。

$$\begin{cases} x-2y-5z &= 3 \\ 5x+4y+3z &= 1 \\ 3x+y-3z &= 6 \end{cases}$$

code : 9.6　連立方程式を解く

```
1  A <- matrix(c(1,5,3,-2,4,1,-5,3,-3),ncol=3)
2  b <- c(3,1,6)
3  solve(A) %*% b
```

1 行目　係数行列 $\boldsymbol{A}$ をつくる

2 行目　右辺のベクトルを作る

3 行目　$\boldsymbol{Ax}=\boldsymbol{b}$ から $\boldsymbol{A}^{-1}\boldsymbol{Ax}=\boldsymbol{A}^{-1}\boldsymbol{b}$ より，$\boldsymbol{x}=\boldsymbol{A}^{-1}b$ として方程式の解を求める

元の方程式との対応を確認しながら，「行列ってすごい，便利！」と思っていただければ幸いです。

9.2　データの行列表現

さて連立方程式が解けるのは嬉しいのですが，データ解析に直結するかと言われたら少し違いますね。データの記述統計量など，数的処理をする時に行列を使う便利さを体感してみましょう。

表 9.1　投手のデータ

Name	team	height	weight	salary	Win	Save
菅野　智之	Giants	186	92	65000	14	0
西　勇輝	Tigers	181	82	20000	11	0
秋山　拓巳	Tigers	188	101	3200	11	0
大野　雄大	Dragons	183	83	13000	11	0
石川　柊太	Softbank	185	86	4800	11	0
千賀　滉大	Softbank	187	90	30000	11	0
涌井　秀章	Eagles	185	85	12500	11	0
森下　暢仁	Carp	180	76	1600	10	0
大貫　晋一	DeNA	181	73	2500	10	0
小川　泰弘	Swallows	171	80	9000	10	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮

表 9.1 に示したのは，データ解析基礎でも扱った 2000 年代の野球選手のデータです[*4]。それをプロジェクトフォルダに置き，`code:9.6` のコードを実行してください。

[*4] 伴走サイト https://kosugitti.github.io/psychometrtics_syllabus/ より，サンプルデータにある「野球選手のデータ 10 年度分」を選んでください。ファイルは UTF-8 形式で保存されています。

code : 9.7 データファイルの読み込み

```
1 library(tidyverse)
2 dataset <- read_csv("baseballDecade.csv") %>%
3   dplyr::filter(Year=="2020年度") %>%
4   dplyr::filter(position == "投手") %>%
5   dplyr::select(Name, team, height, weight, salary, Win, Save) %>%
6   na.omit() %>%
7   arrange(-Win) %>%
8   select(height, weight, salary) %>%
9   as.matrix()
```

1 行目 読み込み他，ファイル操作を便利にするパッケージ tidyverse を読み込みます[*5]。

2-9 行目 データファイルを変形し，dataset オブジェクトに代入しています。ここで %>% はパイプ演算子と呼ばれるもので，左の操作を右の操作へつなげる役割をします。

- **3 行目** データを 2020 年度のものだけに絞るため，dplyr パッケージの filter 関数を適用しています。
- **4 行目** データを投手のものだけに絞るため，dplyr パッケージの filter 関数を適用しています。
- **5 行目** 変数を必要なものだけに絞るため，dplyr パッケージの select 関数を適用しています。
- **6 行目** データセットから欠損値を除いています。
- **7 行目** 表示用に，データを勝利数 (変数 Win) の多い順に並べ直しています。
- **8 行目** 以下の計算に使う変数だけに絞り込んでいます。
- **9 行目** データセットを行列方に変換しています。

ともかくこれで dataset オブジェクトは行列型の，335 行 3 列の行列になっています。今から行う計算は，第 7 講でやったように，データ行列 $\boldsymbol{X}$ から，平均ベクトル $\boldsymbol{m}$，平均偏差行列 $\boldsymbol{V}$，分散共分散行列 $\boldsymbol{S}$，標準偏差を対角に持つ行列 $\boldsymbol{Q}$，標準化スコア行列 $\boldsymbol{Z}$，相関行列 $\boldsymbol{R}$ を作る，という手順です。数式とコードを示しますので，確認しながら進めてください。

$$\boldsymbol{m} = \frac{1}{n}\boldsymbol{X}'\boldsymbol{1} \tag{9.1}$$

$$\boldsymbol{V} = \boldsymbol{X} - \boldsymbol{1}\boldsymbol{m}' \tag{9.2}$$

$$\boldsymbol{S} = \frac{1}{n}\boldsymbol{V}'\boldsymbol{V} \tag{9.3}$$

$$\boldsymbol{Z} = \boldsymbol{V}\boldsymbol{Q}^{-1} \tag{9.4}$$

$$\boldsymbol{R} = \frac{1}{n}\boldsymbol{Z}'\boldsymbol{Z} \tag{9.5}$$

code : 9.8 データの行列演算

```
n <- nrow(dataset)
one <- rep(1, n)
m <- t(dataset) %*% one / n
V <- dataset - one %*% t(m)
S <- t(V) %*% V / n
```

[*5] tidyverse パッケージがない人はインストールしてください。関連パッケージをいろいろまとめたパッケージ群パッケージなので，少し時間がかかります。

```
6  SD <- diag(S) %>% sqrt()
7  Q <- diag(SD)
8  Z <- V %*% solve(Q)
9  R <- t(Z) %*% Z / n
10 cor(dataset)
```

1 行目　行数をオブジェクト n に入れる。行数を数える関数が `nrow` です。

2 行目　データセットの列数と同じ長さの，1 だけを繰り返し入れたベクトルを作ります。繰り返しは `rep` 関数を使います。

3 行目　平均ベクトル $\boldsymbol{m}$ を作る操作です。数式 9.1 に対応する操作です。

4 行目　平均ベクトルを使って平均偏差行列 $\boldsymbol{V}$ を作る操作です。数式 9.2 に対応しています。

5 行目　分散共分散行列 $\boldsymbol{S}$ を作る操作です。数式 9.3 に対応しています。

6 行目　`diag` 関数を使って，行列 $\boldsymbol{S}$ の対角要素を抜き出し，その平方根を取り出しています。SD は標準偏差が入ったベクトルになります。

7 行目　ベクトルを `diga` 関数にいれると，ベクトルの要素を対角に持つ正方行列ができます。それを $\boldsymbol{Q}$ として表現しています。

8 行目　標準スコア行列 $\boldsymbol{Z}$ を作る操作です。平均偏差行列 $\boldsymbol{V}$ に $\boldsymbol{Q}^{-1}$ をかけています。数式 9.4 に対応しています。

9 行目　相関行列 $\boldsymbol{R}$ を作る操作です。数式 9.5 に対応しています。

10 行目　できあがった行列 $\boldsymbol{R}$ を検算するために，R の持っている相関行列を作る関数 `cor` を実行しています。作った $\boldsymbol{R}$ と同じになっていることを確認してください。

この一連のスクリプトは，行列のサイズが変わっても適用できます。 1 つの式表現で，どんなデータサイズでも一般的に表現できているところが行列表記のすごいところです。最終的に `cor` 関数があるのなら，こんな手間をかけなくてもいいじゃないかと思うかもしれません。しかし理屈を知っていて使うことと，理屈を知らずに使うことは違いますからね。

9.3　R による固有値計算

つづいて行列計算のとくにおもしろいところ，固有値計算をしたいと思います。

固有値の計算は，小さなサイズであれば手計算でもできますが，大きなサイズになると n 次連立方程式を解くことになるのでとても大変です。解の公式で解くということもできませんので，近似計算手続きが必要です。その方法としてパワー法，ヤコビ法，ハウスホルダー法などいろいろ考えられていますが，数値計算の細かい理論に入っていくのは少し寄り道が過ぎますので割愛して，R の関数を使って計算しましょう。

先ほど計算した相関行列 $\boldsymbol{R}$ を使ってこれを確認します。

code : 9.9　相関行列の固有値分解

```
eig <- eigen(R)
eig$values
sum(eig$values)
sum(diag(R))
eig$vector
eig$vector[,1]^2 %>% sum
```

固有値分解をする関数は `eigen` です。計算結果を `eig` オブジェクトに入れ，固有値 `values` と固有ベク

トル vector を表示させてみました (出力 9.3)。固有値が大きい方から 1.7043160, 0.9486112, 0.3470728 となっています。ここでは変数が 3 つあって，それぞれの情報の大きさは相関行列で標準化されていて 1.0 です。この行列がもっている 3.0 の分散を，1.7 : 0.9 : 0.3 に再分配したことになります。ちなみに固有値の総和 sum(eig$values) は，行列のトレース sum(diag(R)) と等しくなっていることが確認できます。

　また，固有ベクトルはそれぞれの固有値に対して計算されます。各列が各固有値に対応しており，第一固有値 1.7 に対応する固有ベクトルは (0.68, 0.68, 0.26) です。固有ベクトルは向きだけあって大きさがありませんから，この数字はノルムすなわち二乗和したものが 1 になるようにして算出されているのがわかります。

R の出力 9.3: 固有値と固有ベクトル

```
> eig <- eigen(R)
> eig$values
[1] 1.7043160 0.9486112 0.3470728
> sum(eig$values)
[1] 3
> sum(diag(R))
[1] 3
> eig$vector
          [,1]       [,2]        [,3]
[1,] 0.6839162 -0.1734073  0.70865257
[2,] 0.6812174 -0.1959156 -0.70537929
[3,] 0.2611541  0.9651668 -0.01586178
> eig$vector[,1]^2 %>% sum
[1] 1
```

　このような数値計算を駆使して，因子分析や回帰分析が実際に計算されているのですね。当然ですが，手計算よりも機械で計算させた方が圧倒的に早いですね。計算機の発展スピードのおかげで，今は何万，何十万というデータでも扱うことができる用意なりました。またデータのサイズが変わってもスクリプトを変える必要がありません。

　こうした原理を知っておくと，ビッグデータなど恐るるに足らず，というところではないでしょうか。

9.4 課題

■線形代数の練習問題　行列 $\boldsymbol{A} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} 3 & 1 & 5 \\ 2 & 4 & 6 \end{pmatrix}, \boldsymbol{C} = \begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix}$，列ベクトル $\boldsymbol{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$，行ベクトル $\boldsymbol{y} = \begin{pmatrix} 2 & 8 \end{pmatrix}$ とするとき，次の計算を R で実行するコードを書きなさい。なお，計算できないものについては「計算できない」としてコードは書かないこと。

1. $\boldsymbol{A} + \boldsymbol{B}$
2. $\boldsymbol{A} - \boldsymbol{C}$
3. $\boldsymbol{AB}$
4. $\boldsymbol{AC}$
5. $\boldsymbol{B}'\boldsymbol{A}$
6. $\boldsymbol{A}\boldsymbol{y}'$
7. $\boldsymbol{xy}$

8. $\boldsymbol{xB}$
9. $\boldsymbol{x}'\boldsymbol{B}'$
10. $\boldsymbol{yx}$

■連立方程式を解く 次の連立方程式を R で解きなさい。

$$\begin{cases} x - 2y + 3z = & 1 \\ 3x + y - 5z = & -4 \\ -2x + 6y - 9z = & -2 \end{cases}$$

第 10 章

R をつかった因子分析と尺度作成法

前回に続いて，R をつかった演習を進めましょう。今回は R を使っての因子分析および尺度作成です。

10.1 調査研究の手順

まずは心理尺度作成の手順を大まかに理解しておきましょう。

■構成概念の設定　心理尺度を作り始めるにあたって，まずは何を測りたいのかを明確にする必要があります。**構成概念妥当性 (Construct Varidity)** についてのそもそも論ですね。ある概念を測定したいとして，そういう概念が本当に存在するのか，どこの誰がどのように持つ概念なのかを明確にする必要があります。たとえば文部科学省がスローガンのように掲げる**生きる力**というのを測ってみたいと思っても，それは何なのかわかりません。死んでなければ生きているのですから，生きる力はありそうに思えます。でもそういうことじゃないらしい。じゃあどういうことなのか，ということを考えて行かなければなりません。あるいは，みなさんは**いちびり**という関西弁をご存知ですか。いたずらっ子，わざと変なことをして注目を引きたい子，のような意味なのですが，このいちびり感は関西の人間にしかわからない感覚かもしれません。であれば調査対象者が限定的な，一般的ではない概念ですから，因子分析の行う多くの人の反応から共通成分を抜き出すという考え方には適さない概念です。より一般的なものに考え直す必要があるでしょう。

ほかにも，心理学的な態度なのか，パーソナリティのような行動の傾向なのか，抑うつ傾向のように心身両方の問題なのか，そしてそれらが紙とペンで測定可能なものなのかどうか，改めて考えましょう。何を当たり前のことを，と思うかもしれませんが，意外とおかしな問題設定に入り込んでないとも限らないのです。また，心理尺度や心理学的な概念はすでにさまざまなものが存在します。これらを見比べて，自分の測定したい概念が他の概念とどのように同じで，どのように違うのかを論理的に考えられなければなりません。どこかにある尺度とほとんど同じであればオリジナリティが存在しないだけでなく，車輪の再発明というムダな努力をすることになります。完全にオリジナルなものを考えるのは難しいかもしれませんが，新しい概念を使うことによってこれまでの概念で説明できたことを含み，さらにこれまでの概念で説明できなかったことも説明できるようになる，という利点が必要です。これらは**弁別的妥当性 (distincive validity)** にも関わる問題です。

■項目の選出　測定したい概念が明確になれば，これに関わる項目を作り出す必要があります。ある態度を強く持っている人はどのような振る舞いをし，どのような振る舞いをしないのか。どのような意見に賛成し，どのような意見には反対するのか，といったことをさまざまな角度から検証します。「目に見えないものを測定する」のが心理測定であり，質問紙調査というのはこの見えない的に向かって項目という矢を射掛けるようなものです。矢の数はなるべく多く，また人は嘘をついたり間違えたりする生き物ですから多角的に聞くことで，捉えるべき本質に迫っていく必要があります。言葉を使って聞くわけですから，古すぎる・新しすぎる表現は避

け，専門用語は使わずなるべく平易な言葉で聞く必要があります。ワーディング (言い回し) についても細部まで注意しましょう[*1]。

■予備調査の実施　ある程度項目が集まれば，予備調査を行います。用意した項目の 5 倍から 10 倍の人を集めるのが良い，と一般的に言われています (Grimm and Yarnold, 1994)。ここでの予備調査項目は，十分考えられたものではあると思いますが，分析してみると意外と不適切な項目というのが出てくるものですので，多めに用意されていることでしょう。必然的に，予備調査の対象サイズも百人以上の大きなものになるのが一般的です。

■探索的因子分析　この後詳しく説明しますが，ここが尺度作成に関わる分析のメインです。ここで因子数を決める (あるいは確認する) ことになり，因子的な妥当性みたり，不適切な項目は除外したりします。別の聞き方の方が良いような項目があれば，項目の入れ替えを行なって再調査することもあります。この探索的な因子分析と予備調査を繰り返して，何度行っても同じ因子構造になり，同じ項目が同じ因子に含まれるというのが確認できれば，項目セットは完成すると言えるでしょう。

■項目反応理論　ある因子に関連する項目とそのデータセットを抜き出して，項目反応理論 (段階反応モデル) を実行しましょう。なぜ一部を抜き出すかというと，項目反応理論モデルは 1 次元性を仮定したモデルであることが基本だからです。これを多次元に展開した多次元段階反応モデルもありますので，直接そちらを使っても構いません。ともかくこれを行うことで，反応段階数を確認でき，また項目情報曲線やテスト上表曲線を書くことで，この尺度がどのような領域を得意とする項目群からなるのかを記述できることになります。これに関しては次回より詳しく説明します。

■本調査へ　最終的に項目群が確定したら，大規模な調査を行ってテストの標準化を目指します。因子構造などはほぼ確定しているはずですから，このテストを使うとどのようなデータの散らばりが得られるのかを確定するわけです。この最後のステップの目的は標準化，つまりこのテストで測定される人の平均や散らばり，分布の形状を確認することにあります。使うときに逐一分析しなくてもいいように，項目回答パターンがどれぐらいであれば，全体のどのあたりに位置するかといったプロフィールが作れるとなおいいでしょう。

以上が大体の流れになります。ここにあるように，予備調査を何度も繰り返して因子構造を確定し，そこから本調査を行うことで，ほぼこの尺度はどれぐらいの点数で分布するか，といったことがわかるようになります。かつては因子分析を 1 回実行するだけでも多大な時間がかかりましたから，心理尺度を作るというのはそれだけでライフワークになるような作業でした。幸い最近は分析のスピードが飛躍的に向上しましたので，簡単に分析してやり直すということができます。しかしだからといって，構成概念妥当性を疎かにしてはいけませんし，「やったらこうなった」というようなやり逃げ研究になるのではなく，使うための尺度を作るのだということは忘れてはいけないでしょう。

10.2　共通性の推定

それでは実際にデータを使ってのやり方を説明します。改めて，因子分析モデルについて確認しておきましょう。正方行列を相関行列 $\boldsymbol{R}$ とし，$\lambda_1 \boldsymbol{x}\boldsymbol{x}' = \boldsymbol{a}\boldsymbol{a}'$ となるようにベクトルの大きさが整えられているとすれば，次のように表現できるのでした。

$$\boldsymbol{R} = \boldsymbol{a}_1\boldsymbol{a}_1' + \boldsymbol{a}_2\boldsymbol{a}_2' + \cdots + \boldsymbol{a}_m\boldsymbol{a}_m' + \boldsymbol{d}\boldsymbol{d}' \tag{10.1}$$

[*1] たとえば「テレビやラジオをよく見聞きしますか」という質問は悪い質問です。テレビしか見ない人，ラジオしか聞かない人がどう答えていいか悩むからです。これはダブルバーレルと呼ばれる悪手ですが，ほかにも色々ありますので調査法の専門書を参考にしてみてください。

このことから，相関行列を固有値分解すれば共通因子負荷量が得られることがわかります。しかしこれには 1 つ問題があります。この式の最後の項目，$\boldsymbol{dd'}$ がわからないのです。行列の形で書くと因子分析モデルは次のようになります。

$$\boldsymbol{R} = \boldsymbol{AA'} + \boldsymbol{D^2}$$

ここで $\boldsymbol{D}$ は対角項に d_j^2 をもつ対角行列です。共通因子を得るために分解するべき行列は，次のようになります。

$$\boldsymbol{R}\dagger = \boldsymbol{R} - \boldsymbol{D^2} = \boldsymbol{AA'}$$

$\boldsymbol{R}$ は相関行列であり，その対角は 1.0 ですが，分解すべき行列 $\boldsymbol{R}\dagger$ は対角項に $1 - d_j^2 = h_j^2$ が入った行列でなければなりません。それを固有値分解してはじめて，共通因子成分が得られるわけです。そしてこの d_j^2 の大きさは，事前にわかっていないので，計算の最初にはなんとかしてこれを埋める必要があります。これを**共通性の推定問題**といいます。

実際の計算においては色々な方法が考えられてきています。推定せずに 1.0 で分析しちゃう方法，その行における相関係数の最大値を入れる方法，その項目を他の項目から回帰分析した時の R^2 値を入れる方法 (SMC 法) などです。最初の「共通性を推定しない」という方法は，**主成分法 (principle component method)** と呼ばれます。**主成分分析**という分析法と同じやり方だからです。第二，第三の方法は，ひとまずその値で計算しますが，その後固有値分解をした行列から相関行列を再構成する，つまり $\boldsymbol{R}\dagger \to \boldsymbol{AA'} \to \boldsymbol{R}\dagger_2 \to \boldsymbol{AA'} \to \cdots$ という繰り返しをおこない，数字が変化しなくなるまで反復するという方法で，**主因子法**と呼ばれます[*2]。この主因子法と同じ結果になるとされているのが，**最小二乗法**です。回帰分析の時に聞いたことのある方法ですね。それと同じ考え方で，因子数が決まっていれば，因子構造の形とデータとの誤差が最も小さくなるように共通性を推定するという方法です。原理的に主因子法と最小二乗法は同じ結果になるとされています。最小二乗法が出てきたらひょっとして，と思うかもしれませんが，そうその通りで，**最尤法**もあります。得られたデータが多次元正規分布からのサンプルだと考え，多次元正規分布の形にぴったりフィットするように推定パラメータを定めていく方法です。

このように，探索的に因子分析をする場合は，共通性をどのように推定するかということを指定しなければなりません。

10.3 因子数の決定

先ほどの最小二乗法，最尤法の説明の時に「因子数が決まっていれば」とありました。そうです，探索的な方法の場合は，因子分析をするときに因子数を決めて，何因子の答えを出すかを定めてやらなければなりません。それには色々な方法が考えられています。ここでは古典的な方法 4 つと，洗練された方法 1 つを紹介します。

■スクリープロットを見る　古典的な方法その 1 です。「見る」と書いてあるように，目で見て判断します。**スクリープロット (scree plot)** とは，固有値を大きい順にならべ，横軸に大きさの順番，立て軸に固有値の値をとった折れ線グラフのことです。これを見て，大きな変化が見られたところを基準とし，因子の数を決めます。

■固有値 1.0 という基準　固有値を算出し，1.0 以上であれば共通因子，それ未満であれば誤差因子とまとめる基準のことを言います。1.0 というのは項目の分散の大きさであり，固有値はこの分散の大きさを再配

*2 反復せずに最初に推定した初期値でいく方法もあります。これは反復しない主因子法ですが，計算機能力が十分高い今では反復するのが基本であり，とくに断りなく主因子法とあれば反復して求めていると考えていいでしょう。

分する手続きなのでした。再配分した結果，1 項目分も情報がないような次元は誤差みたいなものでしょ，という判断をすることになります。

■累積寄与率をみる　固有値の大きさを項目数で割ることで，1 つの因子が全体分散の何 % を説明するかを算出できます。これをとくに**寄与率 (contribution ratio)** といいます。これをどんどん積み重ねていって (累積)，累積寄与率が 50% を超えるところまでを共通成分とみなす，という方法です。因子分析は項目情報の圧縮をしていることでもあり，共通因子に入れないものはゴミとして捨てるようなものですが，全体の半分以上をゴミとして捨てたらいかんでしょ，という基準です。

■解釈可能性を考える　因子数に迷った時は，とりあえず何パターンかで分析してみて，最も解釈しやすい数で OK にしましょう，というやり方です。なんと主観的な方法でしょうか。数学的な基準ではなく，研究者のカンに頼る方法であり，客観性や再現性の問題を考えると容認し難い方法ではありますが，実際にはよく使われているやり方です。

■並行分析による方法　ここまでの方法は幾分古く，見た目や感覚にたよるものでしたが，**並行分析 (parallel analysis)** は数学的な基準で分析を行うものです。これは分析するデータセットと同じサイズの乱数を発生させ，その固有値構造とデータの固有値構造を比較するというものです。乱数で作られる構造は，心理的な反応パターンに依らないのですから，当然無意味な因子ばかりが出てきます。実際のデータに基づく相関係数が持つ構造は，心理的なパターンを反映しているのですから，それとは違う有意味なパターンになっているはずです。この有意味性，無意味性を比較して，乱数で作る因子以上の意味ある次元がでていれば，共通因子として採用するという方法です。

以上，5 つほど説明してきましたが，これについては実際に見てもらったほうが早いと思いますので，今から演習を始めたいと思います。

10.4　探索的因子分析の実際

10.4.1　環境の準備 (確認)

まずは環境の準備です。すでに R や RStudio のインストールは終わっているものとして話を進めます。いつもの 3 つのステップをたどって，実行の準備をしてください。

■RStudio の起動　RStudio を起動してください。パッケージのインストールをするときなど，管理人権限が必要になるケースがあります。

■プロジェクトを開く　プロジェクトは前回作成した物で良いと思います。同じフォルダの中で，スクリプトファイルだけ変えれば良いでしょう。メニューからプロジェクトを開き (FIle→Open Project)，RStudio の右上などで，プロジェクトが開いていることを確認しましょう。

■R スクリプトを開く　今日のコードを書くための R スクリプトファイルを準備します。File > NewFile > R Script と進み，何も書いてない R スクリプトの画面を表示させてください。真っ白いファイルが開いたら問題ありません。

また今日は `psych` パッケージ (Revelle, 2021) を用います。準備がまだの人は Package タブからインストールするか，コンソールで `install.packages("psych")` と入力してパッケージを導入しておいてください。

データセットも `psych` パッケージが持っているものを使います。

code：10.1　サンプルデータを使う

```
1 rm(list=ls())
2 library(tidyverse)
3 library(psych)
4 dat <- psych::bfi %>% dplyr::select(-gender,-education,-age)
5 dat
6 help(bfi)
```

■コード解説

1 行目　環境の初期化
2-3 行目　パッケージの読み込み
4-5 行目　サンプルデータ bfi を使う。性別，教育，年齢の情報は因子分析には使わないので除外する。
6 行目　データセットのヘルプを表示

ヘルプ画面にあるように，これは 25 のパーソナリティについての調査項目からなる，2800 人分のデータです。A(Agreeableness, 同調性)，C(Conscientiousness, 几帳面さ)，E(Extraversion, 外向性)，N(Neuroticism, 神経質さ)，O(Opennness, 開放性) という Big 5 の各要素に対応した項目が 5 つずつあります。加えて性別 (gender)，最終学歴 (education)，年齢 (age) といった 3 つの変数も入っています。今回使うのは，5 つの因子それぞれに対応すると考えられる各 5 つの項目，計 25 項目分です。

10.4.2　並行分析による因子数の決定

code：10.2　データの構造を見る

```
1 corMat <- cor(dat,use="pairwise")
2 corMat
3 eigen(corMat)
4 fa.parallel(dat)
```

■コード解説

1-2 行目　データの相関行列を計算し，表示させる
4 行目　相関行列の固有値分解を実行
5 行目　並行分析の実行

因子分析は相関行列から分析を始めますので，まずその計算をしました。オプション use="pairwise"は欠損値が含まれるデータセットに対し，ペアで残っている箇所は使って分析するという指定です[*3]。4 行目で固有値分解をしてみました。結果として出力 10.1 のようなものが示されています。

[*3] これがなかったら，欠損値があるので計算できません，で終わります。

R の出力 10.1: 固有値の推移

```
eigen() decomposition
$values
 [1] 5.0369025 2.7440855 2.1076322 1.8318415 1.5356864 1.1131589 0.8462367 0.8114075
 [9] 0.7349482 0.6956449 0.6810036 0.6571432 0.6281130 0.5964129 0.5626284 0.5405207
[17] 0.5236707 0.4984540 0.4899130 0.4549419 0.4328324 0.4092023 0.4071932 0.3852111
[25] 0.2752152
```

この下に固有ベクトル ($values) も続いていますが，長くなるので省略しました。ここで固有値が 25 個算出されていることがわかります。また，これらを総和すると 25 になります ($tr(\boldsymbol{R}) = \sum_i = 1^{25} r_{ii}$)。このようにに標準化されたデータの分散を再構築し，第 1 固有値は 5.03，すなわち項目 5 個分の情報を持っているようになったのです。第 2 固有値が 2.744，第 3 固有値が 2.107，第 4 固有値が 1.183... と続きます。また，これを相対比率に変えると，$5.03/25 = 20.12$ ですから，第 1 固有値だけで全体の 20% を説明できることになります。第 2 固有値は 2.744 ですから，2.744/25=0.1096 で全体の 10% ほど，第 1 固有値と合わせて 31.12% の累積寄与率があることになります。累積寄与率は第 5 固有値までで 53.02% になりますから，25 項目すべてを使わずとも 5 因子だけで情報の半分は説明できることになります！

また，第 7 固有値の大きさは 0.846 であり，それ以降の固有値はすべて 1.0 未満，すなわち項目 1 つ分も情報を持っていない因子ということになります。であれば共通因子として掬い上げる必要はなく，少なくとも 7 番目以降 (基準 2)，あるいは 6 番目以降 (基準 3) は誤差と考えてまとめてしまってもいいかもしれませんね。

続く fa.parallel 関数は並行分析を行う関数で，同時に**スクリープロット**も表示してくれます (図 10.1)。

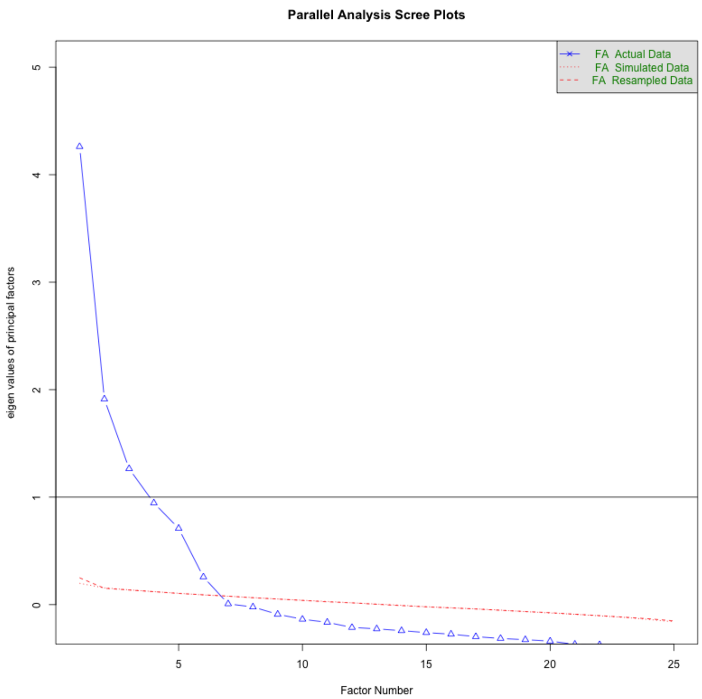


図 10.1　スクリープロットと並行分析

この図は先ほど算出されたような固有値[*4]を大きい順に並べ，線で繋いだものが示されています。青い線が実データ，赤い点線が乱数によるにデータサイズが同じだけの偽データです。基準 2 に該当する 1.0 のところで線が引かれていて，これよりも大きい固有値を持つのが 3 つあることがわかります。まだ，第 1 固有値から第 2 固有値にかけて大きな傾斜があります (4.26063957 → 1.91279965)。最初のうちはまあ当然として，第 5 と第 6 の間も少し大きなギャップがあるようです (0.70938377 → 0.25749866)。それ以降はだらだらと数値が減衰していますね。このような大きなギャップのところは，大きな意味の違いがあると考えてもいいでしょう。基準 1 の観点からいくと，第 5 固有値までを共通因子とし，第 6 因子以降は誤差としてまとめてしまってもいいでしょう。また，並行分析の観点では，赤い線よりも上にある青いラインは有意味ということですから，6 因子構造がいい，ということになりますね。

このように，さまざまな基準で考えても，因子数は 5 か 6 のどちらかになって決定打に欠けます。こういうときは解釈しやすい 5 因子にしよう (基準 4)，ということになったりします。

ちなみにこの固有値構造，最終的には負の値が出ていますので，あまり良いデータとは言えません。25 項目を用意していたのに，8 因子以降は 0 以下，つまり情報がないようなものです。これは項目同士が似かよっていて，あまり違った情報を持ってこなかったということでもあります。実際のデータを分析する時は，こうした点にも注意が必要です。

10.4.3　因子分析の実行

それでは因子数を 5 と定めて因子分析を実行してみましょう。

code : 10.3　因子分析の実行

```
1      result <- fa(dat, nfactors = 5, fm = "ml", rotate = "geominQ")
2      print(result, sort = T)
```

これは 1 行目で実行し，2 行目で表示しているだけですが，内容を少し説明します。psych パッケージの持つ fa 関数は，データの他に因子数 nfactors，共通性の推定方法 fm，回転方法 rotate などをオプションで定めることができます。共通性の推定方法は，ここでは ML(**最尤法**) を選択しました。回転方法とは**因子軸の回転 (rotation of factor axis)** について定めるところです。これについては少し説明が必要ですね。

■因子軸の回転　因子分析は，相関行列という項目間関係が作り出す空間に，その基礎となる軸を見出すものでした。これを図 10.2 のように表現してみます。項目同士はその相関関係から空間を作っていますが，ここでは 2 因子，F1 と F2 という軸をもとに座標のように書いています。項目同士が近くにあるのは，相関あっていることを意味します[*5]。ここで項目 1 の座標を見ると，第一因子 (F1) に 0.3，第二因子 (F2) に 0.6 とありますが，これが項目 1 の因子に対する因子負荷量ということになります。

因子分析では**因子負荷量**をもとに，「この項目と関係の深い因子はどれか」「因子はどういう項目から構成されているか」ということを考えていくのですが，項目 1 は第一，第二因子それぞれにちょっとずつ関係していることがわかります。そのほかの項目も，2 つの因子に負荷していますがなかなか決め手がない，というところですね。

そこで座標を回転させます。どの項目がどの因子に関係しているのかを，よりはっきりさせるように軸をグルリと回すのが因子軸回転の目的です。図 10.3 のように回転させますと，回転後の軸 ($F1', F2'$) に下ろした垂線が座標，すなわち因子負荷量になりますから，項目 1 は第一因子 ($F1'$) に 0.9，第二因子 ($F2'$) に

[*4] 正確には $\boldsymbol{R}\dagger$ の固有値です。

[*5] 原点からのベクトルと考えたときに，2 つの項目ベクトルが作る角度 θ のコサインを取った $\cos\theta$ が相関係数になります。

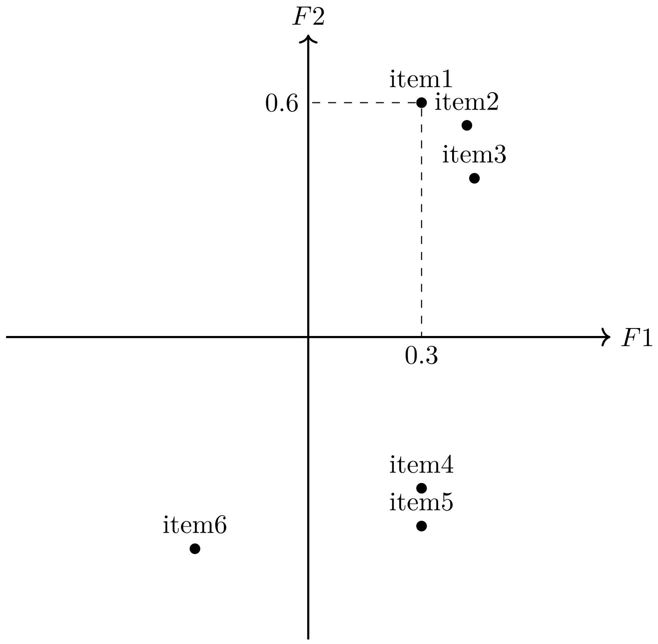


図 10.2　項目の空間にある基底

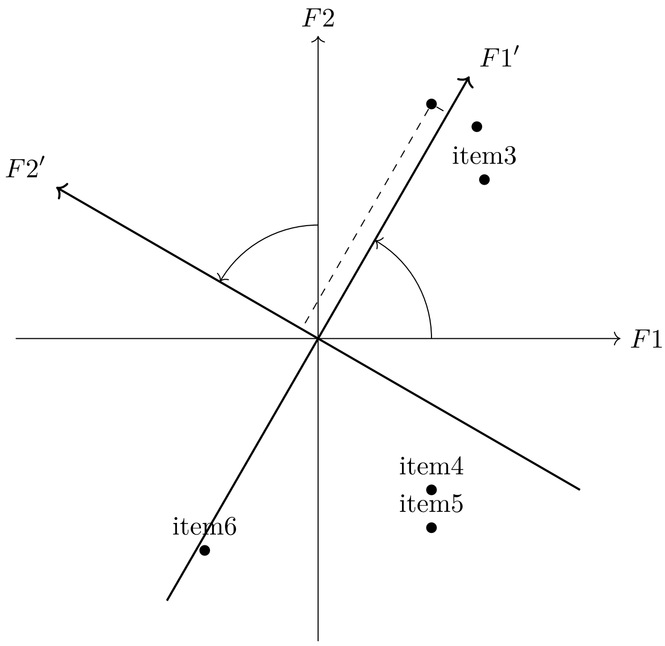


図 10.3　因子軸の回転。60 度回転させてみました。

-0.07 になりました。こうしてみると，項目 1 は第一因子と非常に深く関係しているし，第二因子とはほとんど関係がない，とみることができます。ほかの項目も一方の因子に大きく負荷し，他方の因子にほとんど負荷していないようになりますから，どの因子がどういう項目と関係するかを評価するのにメリハリが効いてやりやすくなります。このように，利用しやすく座標を変えるのが因子軸の回転，という技なのです。

ここで最初の因子軸 $F1, F2$ は直交していましたね。座標ですから当然です。直交している，すなわち因子間相関がないように回転することを**直交回転 (orthogonal rotation)** といいます。

しかし，因子軸の回転の目的は，わかりやすくするためなのですから，いっそ図 10.4 のように，角度をつけた方がさらにうまくメリハリをつけることができますね。この図では，項目 4,5 が明らかに第二因子に影響している，ということが見て取れるようになりました。

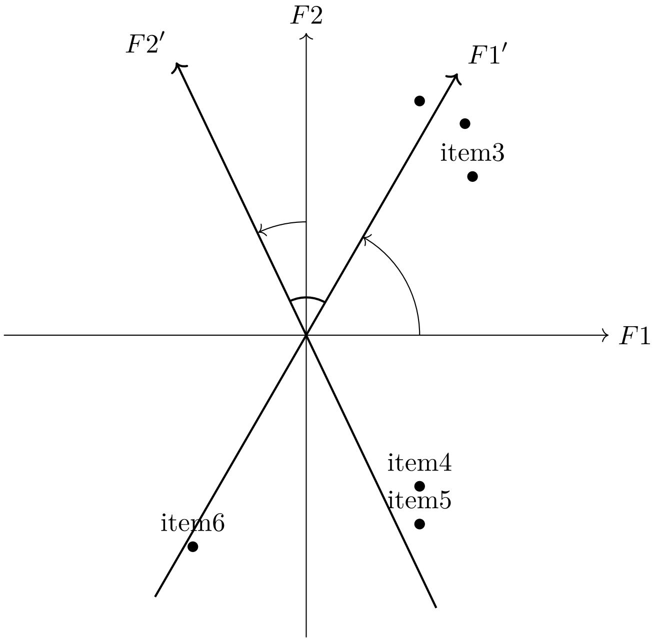


図 10.4　角度をつけて回転させる斜交回転の例。角度があるとは，因子同士が相関するということ。

このように，因子軸が直角で交わっていない場合の回転方法を**斜交回転** **(oblique rotation)** といいます。因子軸の回転は一般に，斜交回転した方がメリハリが効いて因子を解釈しやすくなります。斜交回転したとき算出される因子同士の相関を考えて，これが十分小さい，すなわち直角とほぼ見做せる程度だとおもわれたら直交回転を行う，とするのが良いでしょう。

■行列による説明　因子分析の行列モデルは次のようなものでした。

$$\boldsymbol{R} = \boldsymbol{A}\boldsymbol{A}' + \boldsymbol{D}^2$$

ここで $\boldsymbol{A}\boldsymbol{A}'$ に単位行列 $\boldsymbol{I}$ を挟んでも，$\boldsymbol{A}\boldsymbol{A}' = \boldsymbol{A}\boldsymbol{I}\boldsymbol{A}'$ で結果は変わりませんね。ここで $\boldsymbol{I} = \boldsymbol{T}\boldsymbol{T}'$ となるような行列 $\boldsymbol{T}$ があっても同じです。たとえば 2 因子構造の時に，次のような行列 $\boldsymbol{T}$ を考えたとすると，$\boldsymbol{T}\boldsymbol{T}' = \boldsymbol{I}$ であることがわかります。

$$\boldsymbol{T} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$$

$$\boldsymbol{T}\boldsymbol{T}' = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \boldsymbol{I}$$

このような $\boldsymbol{T}$ は回転行列と言われます。原点を中心に，座標を角度 θ で回転させるからです。θ の値は任意なので，因子負荷行列は，θ に応じて無数にあることになります。であれば，使い勝手の良い角度で回転してやればいいとも言えるわけです。

直交回転の場合は角度が直角，すなわち相関がないわけですから，数学的には回転行列が $\boldsymbol{T}\boldsymbol{T}' = \boldsymbol{I}$ のようになった状態を言います。一方で**斜交回転**の場合は相関があるわけですが，

$$\boldsymbol{I} = \boldsymbol{T}'\Phi\boldsymbol{T}$$

となるように置いておけば，$\boldsymbol{R} = \boldsymbol{A}\boldsymbol{T}'\boldsymbol{\Phi}\boldsymbol{T}\boldsymbol{A}' + \boldsymbol{D}^2 = \boldsymbol{B}\boldsymbol{B}' + \boldsymbol{D}^2$ の形が保たれているので因子分析モデルとしては違いがないわけで，このような $\boldsymbol{\Phi}$ を任意に定めることができます。この時 $\boldsymbol{\Phi}$ は**因子間相関** (**factor correlations**) と呼ばれます。

直交回転も斜交回転も，目的は**解釈をしやすくする**ということにあり，そのための基準はいくつか考えられます。それらの基準に応じて，たとえば直交回転では**バリマックス回転** (**varimax rotation**)，斜交回転では**プロマックス回転** (**promax rotation**) などがあり，ここでは**ジオミン回転** (**geomin rotation**) を選びました[6]。

10.4.4 因子分析の結果をみる

出力 10.2 に因子負荷行列を表示しました。あの $\boldsymbol{A}$ が計算されていますよ！

R の出力 10.2: 因子負荷行列

```
Factor Analysis using method =  ml
Call: fa(r = dat, nfactors = 5, rotate = "geominQ", fm = "ml")
Standardized loadings (pattern matrix) based upon correlation matrix
   item   ML2   ML1   ML5   ML3   ML4   h2   u2 com
N1   16  0.83 -0.07 -0.24  0.02 -0.04 0.71 0.29 1.2
N2   17  0.80 -0.02 -0.23  0.03  0.02 0.66 0.34 1.2
N3   18  0.68  0.15 -0.01 -0.03  0.03 0.53 0.47 1.1
N5   20  0.48  0.26  0.15  0.01 -0.13 0.34 0.66 1.9
N4   19  0.46  0.43  0.03 -0.12  0.09 0.48 0.52 2.2
E2   12  0.10  0.69 -0.08  0.00 -0.04 0.55 0.45 1.1
E1   11 -0.07  0.58 -0.07  0.12 -0.09 0.37 0.63 1.2
E4   14  0.00 -0.55  0.33  0.00 -0.07 0.52 0.48 1.7
E5   15  0.13 -0.43  0.03  0.25  0.22 0.40 0.60 2.4
E3   13  0.05 -0.38  0.27 -0.02  0.30 0.44 0.56 2.8
A3    3  0.02 -0.11  0.65  0.05  0.03 0.51 0.49 1.1
A2    2  0.03 -0.02  0.59  0.11  0.03 0.40 0.60 1.1
A5    5 -0.09 -0.21  0.58  0.01  0.04 0.48 0.52 1.3
A4    4 -0.03 -0.05  0.45  0.21 -0.15 0.29 0.71 1.7
A1    1  0.16 -0.12 -0.39  0.02 -0.03 0.15 0.85 1.6
C2    7  0.09  0.10  0.07  0.63  0.09 0.43 0.57 1.2
C4    9  0.18  0.06  0.05 -0.62 -0.05 0.47 0.53 1.2
C3    8  0.01  0.06  0.09  0.56 -0.04 0.32 0.68 1.1
C5   10  0.19  0.18  0.00 -0.55  0.08 0.43 0.57 1.5
C1    6  0.01  0.03 -0.02  0.52  0.18 0.32 0.68 1.2
O3   23 -0.01 -0.15  0.07 -0.01  0.62 0.47 0.53 1.1
O1   21 -0.03 -0.10  0.01  0.04  0.53 0.32 0.68 1.1
O5   25  0.14 -0.04  0.08 -0.03 -0.52 0.27 0.73 1.2
O2   22  0.20  0.00  0.18 -0.07 -0.44 0.24 0.76 1.8
O4   24  0.11  0.33  0.13 -0.02  0.39 0.26 0.74 2.4
```

出力 10.2 にあるように，各行に N,E,A,C,O の項目が並んでいます。この順番が入れ替わっているのは，表示オプションで `sort=T` を入れたからで，因子負荷量の大きい順に並べ替えて表示させたからです。現に左上から右下にかけて，因子負荷量の大きさの順に並んでいますね。列として ML1,2,3,4,5 とあるの

[6] ジオミン回転には直交モデルと斜交モデルがあり，それぞれ `geominT`，`geominQ` という名前で実装されています。

は最尤法 (ML) で抽出した因子の 1,2,3,4,5 を表しています。h2 は h^2，u2 は $1 - h^2 = d^2$ をあらわしています (**独自性 (uniqueness)** の u)。com は**複雑性指標 (index of complexity)** と呼ばれ，$\sum(a_j^2)^2 / \sum a_j^4$ で計算される指標です。共通性は高く，独自性，複雑性の低い項目がいい項目だといえるでしょう。ちなみに，出力オプションとして `cut=0.3` を追加すると，因子負荷量が 0.3 に満たないところの表示はなくなります。どの項目がどの因子と関係あるのかがわかりやすくなるのでおすすめです。

出力 10.3 にはその他の情報として，各因子の因子負荷行列の二乗和 (`SSloadings`)，分散説明率 (`Proportion Var`)，累積分散説明率 (`Cumulative Var`)，全体を 1 とした時の説明比率 (`Proportion Explained`)，その累積 (`Cumulative Proportion`) が表示されています。回転するまえは固有値と**因子負荷量**の二乗和が合致するのですが，回転した後の場合はそうはならないので，これらをみてどれぐらい説明できているかを考えます。とくに類遺跡分散説明率が，今回は 5 因子で 41% しかありません。59% をゴミとして捨てたようなものですから，これでよかったのかな，と思案するところです。

R の出力 10.3: 因子分析結果 2

```
ML2  ML1  ML5  ML3  ML4
SS loadings           2.50 2.24 2.05 1.95 1.61
Proportion Var        0.10 0.09 0.08 0.08 0.06
Cumulative Var        0.10 0.19 0.27 0.35 0.41
Proportion Explained  0.24 0.22 0.20 0.19 0.16
Cumulative Proportion 0.24 0.46 0.66 0.84 1.00
```

その下出力 10.4 には，因子間相関が表示されています。これをみると，第 2 因子 (ML1) と第 3 因子 (ML5) との間に -0.34 という中程度の負の相関がみられます。直交回転はこれを 0 として解釈してしまいますから，スッキリはするのですがデータには適合してないところかもしれません。このように，分析する時はまず斜交回転を行い，因子間相関がなべて低い場合は直交で分析をやり直す，というステップを経ます。

R の出力 10.4: 因子分析結果 3

```
 With factor correlations of
      ML2   ML1   ML5   ML3   ML4
ML2  1.00  0.11  0.06 -0.12  0.06
ML1  0.11  1.00 -0.34 -0.22 -0.16
ML5  0.06 -0.34  1.00  0.18  0.23
ML3 -0.12 -0.22  0.18  1.00  0.17
ML4  0.06 -0.16  0.23  0.17  1.00
```

10.5 因子分析の後で

因子数を定め，因子負荷量が明らかになると，ここから逆算的に**因子得点**を計算できます。`fa` 関数には `scores` オプションがあり，これをつけて実行すると因子得点行列を表示させることができます。因子得点は因子数 × 人数分 (ここでは 2800 人分) ありますので，大変大きなサイズのデータですから注意してください。ともあれ，そこに含まれているのは，各因子についての個々人の相対的な関与度ということになります。

code : 10.4 因子得点を算出する

```
result <- fa(dat, nfactors = 5, fm = "ml", rotate = "geominQ", scores = T)
result$scores
```

モデルの仮定にあった通り，因子負荷量は標準化されたスコアです。すなわち平均情報が取り払われていることに注意してください。たとえば 7 件法で「4. どちらでもない」を挟んで，ポジティブ・ネガティブな意味を持っていたとしても，ここでは個々人間の相対的な大小関係しか意味しないことになります。

そういう意味で，絶対的なスケールの情報を残したまま得点を考えてみたいということもあるでしょう。そういう時は**簡便的因子得点**といって，因子に関係する項目の素点から平均点を算出し，個々人の得点とする方法もあります。最もこのやり方は，平均点の情報は保持されますが，因子分析で除去できた d_j^2，すなわち独自性成分を再び取り込むことになっている点に注意が必要です。次のコード code:10.4 は，簡便法による因子得点と因子分析の結果から計算された因子得点との相関係数を算出する例です。高い相関を示すところ (ML2 と N で 0.9475742，ML5 と A で 0.83845417) もありますが，低いところもありますので，どちらの方法で計算するのか，その場合の長所短所をよく理解して使うようにしましょう。

code : 10.5　二種類の因子得点計算法比較

```
dat %>%
    dplyr::mutate(
        A = (A1 + A2 + A3 + A4 + A5)/5,
        C = (C1 + C2 + C3 + C4 + C5)/5,
        E = (E1 + E2 + E3 + E4 + E5)/5,
        N = (N1 + N2 + N3 + N4 + N5)/5,
        O = (O1 + O2 + O3 + O4 + O5)/5
    ) %>%
    dplyr::bind_cols(.,result$scores %>% as.data.frame) %>%
    dplyr::select(ML1,ML2,ML3,ML4,ML5,A,C,E,N,O) %>%
    cor(use="pairwise")
```

10.6　さいごに

今回は探索的因子分析の実行例をみてみました。いかがだったでしょうか。計算自体にはほとんど時間がかかりませんので，色々な因子数や回転方法，抽出法を試して見るのも良いかもしれません。

このように比較的簡単に計算できるようになってきましたが，大事なのはその目的と意味です。関数の名前を覚えて，機械的に XX 法 XXX 回転で答えが出たから正しいのだ，などと思わないことです。因子数の決め方の時にも薄々感じていただけたかと思いますが，数学的なモデルとは裏腹に，実際にやる時は職人的ノウハウも結構必要です。スクリープロットを「みて」とか，回転法を「選んで」とか，因子得点の計算の仕方を「比較して」，といったところは人間的要素であり，やり方が変われば結果が変わる可能性があります。論文などに示されているこれらの方法，指標も，変えようと思えば変得られるところだということです。科学的真実の探究が目的であれば，良い見栄えや自分に都合の良い結果のための手法選択は無意味であることは理解してもらえるところだと思います。皆さんも決して目的を見失わず，批判的に検証できるようになってください。

探索的因子分析は因子を見つけ出すことができます。が，それは決して人間の心の何かを取り出したわけではないのです。人間がそもそも持っている心の状態を取り出したのではなく[*7]，人間に「項目群」「紙とペン」という刺激を与え，反応を強いたのです。その結果，「項目群にみられる相関行列に潜む次元」が見いだせただけであり，それが人間の持つ特性だと考えるのは，あくまでの研究者の主観的主張にすぎません。実際にあり得る話ですが，「学校生活は嫌ですか」「学校は楽しくないと思うことがありますか」「学校の友達とは遊びたくないと思うことがありますか」といったネガティブな項目を山のように投げかけ，因子分析することで

[*7] 因子分析をすると「因子を抽出した」と論文などに記載される場合がありますが，このようにあたかも先にあるエッセンスを抜き出したのだ，という表現は適切ではないと批判されることもあります。

「学校を嫌う項目群＝因子」を抽出し，子供は学校が嫌いなのだという結論を出すのがおかしいことなのはわかりますよね。項目群のなかにネガティブなものしかなければ，因子もそれに応じたものしか出ません。そして因子得点が相対的なものでしかないので，素点を見ずに「学校嫌い得点が高い，低い」として議論を進めても仕方がないのです。そうした場合は簡便的因子得点を計算して，平均点がちゃんと中点「4. どちらでもない」を挟んでいるかどうかで嫌っているかどうかを判断しなければなりません。このように，自分で項目に埋め込んだ呪いを自分で取り出して見つけ出したと騒ぐ，**てっちゃんの手品**にならないようにすることは，常に心がけなければならないのです[*8]。

こうした批判的観点を持つためには，数理的にはどういうことをやっていて，どこからが人為的な問題なのかをしっかり把握しておくことが役に立つでしょう。

10.7 課題

今日の授業でおこなったすべての次の計算をする R コードを提出してください。ファイル形式は R スクリプトか Rmd とします。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

*8 てっちゃんの手品については，小杉 (2018) を参照してください。

第 11 章

R をつかった項目反応理論

今回も R を使った演習です。R を使っての行列計算，R を使っての因子分析と進めてきましたが，今回は R をつかって項目反応理論を実践的に理解していきましょう。

11.1　項目反応理論の実際

11.1.1　データの素描

項目反応理論は，テスト理論に関するモデルですから，サンプルデータとして二値データを使いましょう。表 11.1 に使うデータの一部を示しています[*1]。

表 11.1　IRT で使うサンプルデータ (一部)

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	1
0	1	0	1	0	0	0	0	0	1
0	1	1	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	0	1	1

このように，テスト理論で使うデータは 0/1 の二値データで，1 が正答，0 が誤答を表しています。

これらのデータを使って R で分析をします。IRT を実行するパッケージとして，`ltm`(Rizopoulos, 2006) や `irtoys`(Partchev, Partchev and Suggests, 2017) などがあります。ここでは `ltm` パッケージの方を使っていきましょう。

分析に先立って，データの記述統計量を確認しておきます。`code:11.1` を実行してください。

code : 11.1　データの読み込みから記述統計量の計算まで

```
rm(list=ls())
library(tidyverse)
library(ltm)
dat <- read_csv("IRTsample.csv")
ltm::descript(dat)
```

[*1] このデータ全体については，`IRTsample.csv` というファイル形式で配布していますので，それを読み込んで使ってください

■コード解説

1 行目　環境の初期化
2-3 行目　パッケージの読み込み
4-5 行目　サンプルデータ `IRTsample.csv` を読み込む。文字化けする人は，`read_csv` 関数のオプションとして `locale=locale(encoding="UTF-8")` を追加してください。
6 行目　ltm パッケージに含まれる `descript` 関数を実行

いろいろな情報が一気に出力されるので，順にみていきます。まず **R の出力** 11.1 のところです。ここには各変数の 0 と 1 の比率が入っています。この例では，V1 は 0 が 71.0%，1 が 29.0% 含まれているということです。最後の `logit` は，0 から 1 の間に入る数字 p にたいして，$logit(p) = \log\left(\frac{p}{1-p}\right)$ で計算される量です。ここでは p は正答率であり，この `logit` の値が負であれば 0 の比率が，正であれば 1 の比率が多かった，ということを意味します。0/1 がちょうど半々であれば，`logit` の値は 0 になる，そういう数字です。これを見るだけでも，V8 はずいぶんと正答率が低かったんだな，V3 は逆に正答率が高かったんだな，ということがわかります。

R の出力 11.1: 記述統計の出力 1

```
    Proportions for each level of response:
    0     1    logit
V1  0.710 0.290 -0.8954
V2  0.114 0.886  2.0505
V3  0.186 0.814  1.4762
V4  0.206 0.794  1.3492
V5  0.670 0.330 -0.7082
V6  0.832 0.168 -1.5999
V7  0.666 0.334 -0.6901
V8  0.884 0.116 -2.0309
V9  0.652 0.348 -0.6278
V10 0.254 0.746  1.0774
```

次のセクション **R の出力** 11.2 には，正答数の分布が示されています。10 問全部正解したのは 3 人だけ，6 問正解した人が 104 人だった，といったことがわかります。

R の出力 11.2: 記述統計の出力 2

```
        Frequencies of total scores:
        0  1  2  3  4   5  6  7  8  9 10
   Freq 4 19 41 62 97 104 66 54 36 14  3
```

次のセクション **R の出力** 11.3 には，**点双列相関係数 (Point Biserial correlation)** が計算されています。この相関係数は 2 値と連続値の相関係数であり，ここでは各変数 (二値) と合計得点 (0 から 10) の相関係数を計算しています。この合計得点の計算方法には，その変数自身を含める場合 (Included) と含めない場合 (Excluded) の二種類があります。具体的に説明したほうがわかりやすいでしょう。たとえば変数 V1 についていうと，「その変数自身を含める場合」の合計得点とは $V1+V2+V3+V4+V5+V6+V7+V8+V9+V10$ で計算しています。「含めない場合」は $V2+V3+\cdots+V10$ とするわけです。含める場

合の相関係数が 0.5207，含めない場合の相関係数が 0.2480 です。当然自身が含まれているほうが合計得点に深く関わっているので相関係数が高くなります。裏を返せば，含める場合と含めない場合の相関係数に大きな差があるとき，その変数はテスト全体に大きく貢献する重要な変数だといえるでしょう。

R の出力 11.3: 記述統計の出力 3

```
Point Biserial correlation with Total Score:
    Included Excluded
V1     0.4228   0.2118
V2     0.3959   0.2502
V3     0.5202   0.3567
V4     0.5209   0.3501
V5     0.5449   0.3470
V6     0.4635   0.2983
V7     0.5440   0.3452
V8     0.2901   0.1350
V9     0.4992   0.2890
V10    0.5890   0.4180
```

次のセクション R **の出力** 11.4 には，α **係数 (alpha coefficient)** が算出されています。これも全体の場合と，特定の変数を抜いた場合とを計算しています。心理尺度における信頼性係数としての α 係数は，0.8 以上であれば信頼性がある，といった目安がありますが，二値データの場合はここに示したようにそれよりもずいぶんと低くなります。これは二値データの分散が小さくなることから仕方のないことですし，項目反応理論で分析する上での信頼性は情報量関数として表されますから，問題ではありません。

R の出力 11.4: 記述統計の出力 4

```
        Cronbach's alpha:
        value
All Items     0.6341
Excluding V1  0.6303
Excluding V2  0.6195
Excluding V3  0.5978
Excluding V4  0.5986
Excluding V5  0.5982
Excluding V6  0.6100
Excluding V7  0.5987
Excluding V8  0.6380
Excluding V9  0.6130
Excluding V10 0.5817
```

最後にセクション R **の出力** 11.5 では，変数同士の関連がとくに弱いものが示されています。変数同士のペアについて度数分布表を作成し，χ^2 検定によって `p.value` を算出しています。この値が小さければ，2 つの変数の関係が強いことを示していると解釈できるのですが，ここにあげているように大きな数字になっているということは，変数間の関係が弱いと読み取ることができます。

R の出力 11.5: 記述統計の出力 4

```
Pairwise Associations:
   Item i Item j p.value
1       2      8   0.625
2       1      8   0.409
3       1      2   0.347
4       6      8   0.303
5       4      8   0.234
6       7      8   0.222
7       5      8   0.195
8       1      4   0.191
9       1      3   0.166
10      3      8   0.124
```

11.1.2 IRT モデルの実際

データの感じがわかったところで，IRT モデルを実行していきましょう。

今回は 3 つのモデルを試してみます。確認のために，少しモデル式に言及しておきます。困難度母数だけに注目するのが **1 パラメータ・ロジスティックモデル (1 parameter logistic model, 1PL)** で，モデル式は次の通りです。

$$p_j(\theta) = \frac{1}{1 + exp(-1.7(\theta - b_j))}$$

ここで $p_j(\theta)$ は能力が θ のひとが項目 j に正答する確率であり，b_j が**困難度母数** difficulty parameter と呼ばれます。これは開発者の名前を使って，別名**ラッシュモデル (rasch model)** とも呼ばれています。

項目の識別力にも注目するのが **2 パラメータ・ロジスティックモデル (2 parameter logistic model, 2PL)** で，モデル式は次の通りです。

$$p_j(\theta) = \frac{1}{1 + exp(-1.7a_j(\theta - b_j))}$$

ここで a_j は**識別力母数 (descriminant paramter)** と呼ばれます。

最後に **3 パラメータ・ロジスティックモデル (3 parameter logistic model, 3PL)** では，困難度，識別力に加えて**当て推量母数 (guessing parameter)** というのを推定します。モデル式は次の通りです。

$$p_j(\theta) = c + \frac{1 - c}{1 + exp(-1.7a_j(\theta - b_j))}$$

もちろん式ではピンと来ないという人もいると思いますので，実際にどのように表されるかみていきましょう。

1PL モデルの場合

`ltm` パッケージを使って 1PL モデルを実行します。`code:11.2` を実行してください。

code : 11.2 1PL モデルの実行

```
result.1pl <- rasch(dat)
```

```
2  print(result.1pl)
3  plot(result.1pl,type="ICC")
4  plot(result.1pl,type="IIC")
5  plot(result.1pl,type="IIC",items = 0)
```

■コード解説

1 行目　1PL モデル (ラッシュモデル) の実行

2 行目　結果の出力

3 行目　**項目特性曲線 (Item Characteristic Curve)** のプロット。`plot` 関数のオプションで表示する情報を ICC と指定。

4 行目　**項目情報曲線 (Item Information Curve)** のプロット。`plot` 関数のオプションで表示する情報を IIC と指定。

5 行目　**テスト情報曲線 (Test Information Curve)** のプロット。`plot` 関数のオプションで表示する項目番号をゼロにすると TIC になる。

2 行目で結果の出力を行っていますが，出力 11.6 のように表されます。

R の出力 11.6: 2PL モデルの結果出力

```
Call:
rasch(data = dat)

Coefficients:
 Dffclt.V1   Dffclt.V2   Dffclt.V3   Dffclt.V4   Dffclt.V5   Dffclt.V6
     0.992      -2.208      -1.615      -1.480       0.787       1.744
 Dffclt.V7   Dffclt.V8   Dffclt.V9  Dffclt.V10      Dscrmn
     0.768       2.187       0.699      -1.189       1.126
Log.Lik: -2468.743
```

1PL モデルで表現する項目の違いは**困難度母数**だけですから，この数字 (`Dffclt`) が小さければ簡単な項目，大きければ難しい項目だったことがわかります。たとえば V2 の困難度母数は $b_2 = -2.208$ ですが，記述統計のところで見た正答率 (88.6%) を考えると，簡単な問題だったことがわかりますね。**識別力母数 (は)**`Dscrmn` として 1.126 と推定されています。これは項目を通じて一定です。このことは図 11.1 の左にある ICC をみて，すべての曲線の傾きが同じことで確認できます。

図 11.1 の左が数値的出力結果を図示したものです。これを見た方がわかりやすいかもしれません。x 軸は θ，つまり測定する潜在変数の値になっており，テストで言えばある人の学力 θ_i があったときに，各問いにどれぐらいの確率で正答するかを表していることになります。これをみると V2 がもっとも簡単で，V8 が最も難しかったテストであることがわかります。

また図 11.1 の中央にあるのが IIC で，それぞれの項目がどのあたりの能力値 θ を測定するときにもっとも敏感になるかを表しています。これは項目反応理論でいう**信頼性**の表現であることを思い出してください。最後に図の左側にあるのは，この IIC を 10 問分足し合わせたテスト全体の情報曲線，TIC になります。これをみると，$\theta = 0$ よりやや右側のあたりにピークがきているので，このテスト全体としては平均より高い学力の人を測定するときに，最も鋭敏に働くことがわかるでしょう。

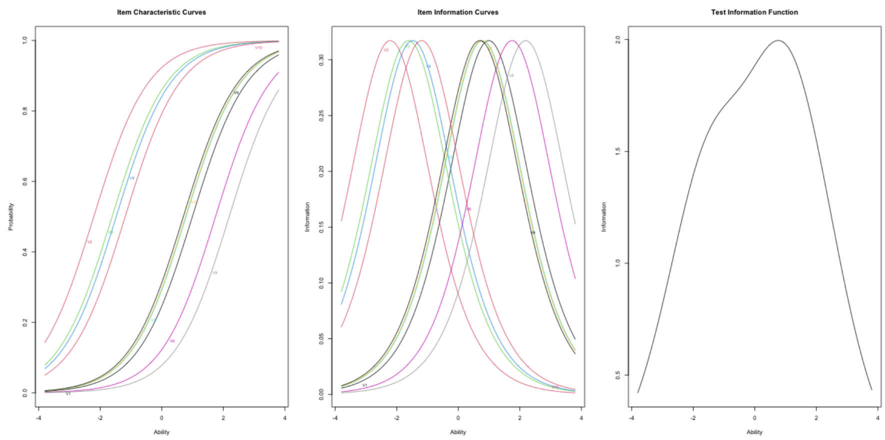


図 11.1　1PL のさまざまなプロット。左から ICC,IIC,TIC

2PL モデルの場合

ひきつづき `ltm` パッケージを使って，2PL モデルを実行していきましょう。`code:11.3` を実行してください。

code : 11.3　2PL モデルの実行

```
1  result.2pl <- ltm(dat~z1)
2  print(result.2pl)
3  plot(result.2pl,type="ICC")
4  plot(result.2pl,type="IIC")
5  plot(result.2pl,type="IIC",items = 0)
```

■コード解説

1 行目　1PL モデル (ラッシュモデル) の実行
2 行目　結果の出力
3 行目　**項目特性曲線 (Item Characteristic Curve)** のプロット
4 行目　**項目情報曲線 (Item Information Curve)** のプロット
5 行目　**テスト情報曲線 (Test Information Curve)** のプロット

2 行目で結果の出力を行っていますが, 出力 11.7 のように表されます。

R の出力 11.7: 2PL モデルの結果出力

```
Call:
ltm(formula = dat ~ z1)

Coefficients:
     Dffclt  Dscrmn
V1    1.602   0.603
V2   -2.078   1.228
V3   -1.213   1.892
V4   -1.199   1.620
V5    0.759   1.176
V6    1.698   1.175
V7    0.740   1.174
V8    4.133   0.516
V9    0.866   0.828
V10  -0.880   2.020

Log.Lik: -2446.104
```

2PL モデルで表現する項目の違いは，**困難度母数**と**識別力母数**の 2 つです。識別力母数を入れることで，IIC 曲線の傾きもデータに合わせて調整できるようになりました。識別力母数のデフォルトは (推定しないのであれば)1.0 ですから，ここから大きく逸脱するような項目には注意です。今回は `V8` をみると，困難度が非常に高いこともわかりますが，識別力が非常に低くなっているので，この項目に誤答したからと言ってすぐさま成績が悪い，と判断できるほどではないことがわかります。

最後の一行に `Log.Lik` とありますが，これは**対数尤度 (log likelihood)** を表しています。これが大きいほど，データとモデルの適合度が高いことを意味します。1PL のときの `Log.Lik` が −2468.743 でしたから，1PL モデルより 2PL モデルの方が当てはまりがよかったと言えるでしょう。

プロットによる出力も見ておきましょう (図 11.2)。識別力母数が入ったことで，ICC の傾きが項目によってさまざまになりますし，IIC や TIC の見た目もすっかり変わってしまいましたが，読み取り方は同じです。

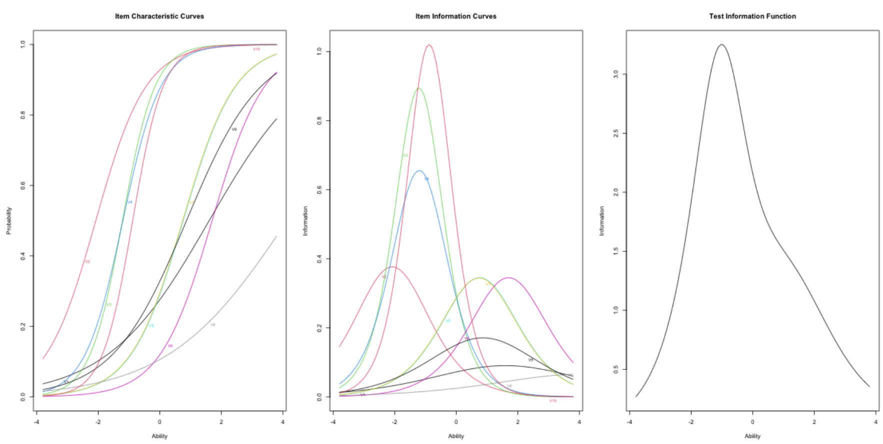


図 11.2　2PL のさまざまなプロット。左から ICC,IIC,TIC

3PL モデルの場合

最後に ltm パッケージを使って，3PL モデルを実行していきましょう。code:11.4 を実行してください。

code : 11.4　3PL モデルの実行

```
result.3pl <- tpm(dat)
print(result.3pl)
plot(result.3pl,type="ICC")
plot(result.3pl,type="IIC")
plot(result.3pl,type="IIC",items = 0)
```

コードは一行目の関数以外ほとんど同じですので，逐一解説は致しませんが，出力のところを見ておきましょう (出力 11.8)。

R の出力 11.8: 3PL モデルの結果出力

```
Call:
tpm(data = dat)

Coefficients:
     Gussng  Dffclt  Dscrmn
V1    0.230   1.407  15.123
V2    0.638  -0.610   2.232
V3    0.225  -0.825   2.907
V4    0.324  -0.637   2.324
V5    0.149   0.931   2.992
V6    0.043   1.613   1.706
V7    0.074   0.879   1.509
V8    0.000   4.490   0.471
V9    0.133   1.144   1.255
V10   0.278  -0.438   3.085

Log.Lik: -2431.371
```

3PL モデルは困難度，識別力に加えて**当て推量母数**が計算されます。これは ICC の形 (図 11.3 の左) を見ればわかりますが，曲線全体が底上げされたようになっているものがあります。この最低ラインの高さが当て推量母数の大きさです。図や数値から，V2 のそれが大きい数字であることが読み取れます。ICC は θ の関数である正答率ですが，これが θ の値にかかわらず一定の値を取るということは，どれだけ能力が低くてもその確率で正答してしまうことを意味しています。なので「適当に推測して答えを当てられる大きさ」という意味で当て推量母数と呼ばれているのです。

今回当て推量母数を含めて考えたことで，IIC や TIC がまた大きく変化しました。この数字が高すぎる V2 は，テストとしてはあまり良い項目ではなかったのかもしれません。また，モデルの適合度としては 3PL モデルが最もよかったのですが，2PL モデルとそれほど大きく違うわけでもありません。どのモデルでどのように推定するべきか，推定されたモデルはどういう仮定を持っているのかについて，しっかり理解した上で利用するようにしましょう。

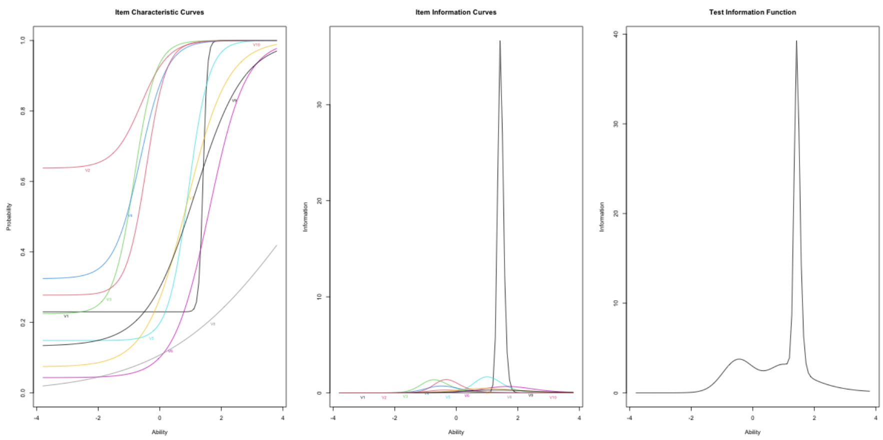


図 11.3　3PL のさまざまなプロット。左から ICC,IIC,TIC

11.2　段階反応モデルの実際

続いて**段階反応モデル**の練習に進みましょう。項目反応理論は正答/誤答の二値データに対して用いるものでしたが，これを多段階反応に拡張したモデルの 1 つが段階反応モデルです。多段階反応とは，「まったく当てはまらない」を 1，「非常に当てはまる」を 7 とする，といったような**リッカート尺度**のようなスタイルの調査項目です。反応させるカテゴリの段階数に応じて，5 件法とか 7 件法と呼ばれたりします。

リッカート尺度の場合は，シグマ法をつかって累積度数からカテゴリに該当する値を算出し，それを元に態度得点を作るという方法でした。とはいえこのやり方でやっても，いい尺度であればそのまま素点を尺度値としてもほとんど問題がないため，シグマ法が回りくどい方法だと思われたのか，使われなくなってしまいました。しかし実際には「いい尺度であれば」という条件を満たしているかどうか，しっかり確認しなければなりません。歪んだ分布や偏ったカテゴリ反応なのに，素点をそのまま尺度値とするのは不適切なことがあります。また，リッカート尺度で撮られたデータは因子分析をすることによって，解釈度に分類されて，因子ごとのスコアを算出して利用されることが一般的です。しかし，尺度値が適切でなければ，ピアソンの積率相関係数も適切ではなく，そこから計算が始まる因子分析のモデルにも疑義が生じます。

カテゴリ反応はあくまでも並びの問題であって，**順序尺度水準**の情報しか持たないと考えるのであれば，その背後にある態度のような連続体を仮定し，この態度 θ が大きくなることで徐々に反応カテゴリが変わる，という**段階反応モデル**のほうが適切な分析方法になるわけです。

項目反応理論 (の段階反応モデル) を使って確認することは決して難しくありません。今回は因子分析の時に使った bfi データの一部を使って段階反応モデルを実践してみましょう。一部を使うとしたのは，段階反応モデルは項目反応理論の中間ですから，潜在変数が 1 次元 (一因子) である，という仮定があるからです。ここでは N 因子 (Neuroticism，神経質さ) を使った例を示しています。

code : 11.5　GRM の実行

```
library(psych)
dat <- bfi %>%
  dplyr::select(-gender, -education, -age) %>%
  dplyr::select(starts_with("N"))
result.grm <- grm(dat)
result.grm
```

```
7  plot(result.grm, type = "ICC", items = 1)
8  plot(result.grm, type = "IIC", items = 1)
9  plot(result.grm, type = "ICC", items = 4)
10 plot(result.grm, type = "IIC", items = 4)
11 plot(result.grm, type = "IIC", items = 0)
```

■コード解説

1 行目　psych パッケージを読み込む。サンプルデータ bfi を用いるためです。

2-4 行目　データの加工。bfi データから，性別，最終学歴，年齢などの情報を取り除き，また N で始まる変数だけ選出 (select) しています。

5 行目　段階反応モデルの実行。

6 行目　結果の出力。

7 行目　項目 1 についての ICC，厳密には**項目反応カテゴリ特性曲線 (Item Response Category Characteristic Curve)** と呼びます。

8 行目　項目 1 についての IIC を表示させています。

9 行目　項目 4 についての ICC を表示させています。

10 行目　項目 4 についての ICC を表示させています。

11 行目　項目群についての TIC を表示させています。

code:11.5 の 6 行目で，分析結果の数値的出力をさせていますので，それをみておきましょう (出力 11.9)。

ここで，Dscrmn とあるのは識別力母数ですが，困難度母数に対応するのが Extrmt1 から Extrmt5 にあたります。この尺度は 6 件法ですので，反応カテゴリが変わる切れ目，すなわち**閾値**が 5 つあるのです。これらが 2PL モデルでいうところの困難度に対応すると言ってもいいでしょう。

R の出力 11.9: GRM モデルの結果出力

```
Call:
grm(data = dat)

Coefficients:
    Extrmt1  Extrmt2  Extrmt3  Extrmt4  Extrmt5  Dscrmn
N1   -0.810   -0.093    0.342    0.985    1.720   3.125
N2   -1.366   -0.555   -0.111    0.648    1.483   2.890
N3   -1.187   -0.298    0.123    0.876    1.767   2.026
N4   -1.564   -0.355    0.238    1.240    2.280   1.277
N5   -1.296   -0.125    0.494    1.479    2.530   1.112

Log.Lik: -21721.91
```

数値を見ていてもピンとこないかもしれませんので，図でこの各項目の特徴を確認しておきましょう。変数 N1 と N4 の IRCCC を図 11.4 に示しました。この図に表されている複数の曲線が，θ に対応した各カテゴリに反応する確率を表しています。図の左，N1 は綺麗なカーブが描かれており，各カテゴリのピークが見て取れますが，N4 は反応カテゴリ 3 の山が埋もれてしまっていることがみてとれます。この項目については，6 段

階ではなく 5 段階で反応を求めるのがよかったのかもしれません[2]。

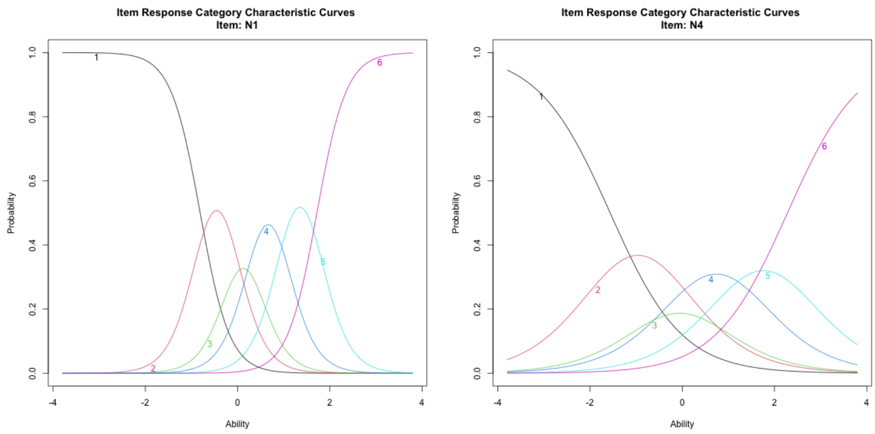


図 11.4 GRM の IRCCC

原理的には難しいことのように感じますが，コードを書いて実践する分にはほとんど苦労がなく，二値モデルと同じように考えることができます。コードには IIC や TIC を描くものもありますが，1PL,2PL,3PL モデルと変わらない感じで多段階モデルが実行できますね。みなさんも心理尺度を使う時は，盲目的にスコアリングし，因子分析するのではなく，どういう仮定の元でどういう計算をしているかを理解しながら，データや調査協力者にとって最適な分析をするように心がけてください。

11.3 カテゴリカル因子分析との対応

さて多段階反応による分析も簡単にできることがわかりましたが，気になるのは段階反応モデルが一因子構造しか仮定していないところです。bfi データはビッグファイブと呼ばれるように五因子 (5 次元) の仮定があるのですが，一因子ずつ grm を実行し，あとで統合するのは大変です。

でも大丈夫。簡単に多次元に展開する方法があります。いわゆる一般的な因子分析は，ピアソンの積率相関係数を元に (積率相関係数の相関行列 $\boldsymbol{R}$ の固有値分解から) 因子を求めていくのですが，この積率相関係数が**間隔尺度水準**を仮定したもので，シグマ法で計算された尺度値を使っていないのであれば疑義が残る，という話でした。この元になる相関係数が，潜在的連続体を仮定した順序尺度水準向けの相関係数である，**ポリコリック相関係数**であれば，その問題は解決します。反応段階が順序尺度を仮定した因子分析のことをとくに**カテゴリカル因子分析**と言います。カテゴリカル因子部な席は，前回紹介した psych パッケージに少しオプションを足すだけで簡単に実行できるものです。

code : 11.6 GRM の実行

```
1  dat <- bfi %>% dplyr::select(-gender, -education, -age)
2  result.fa <- fa.poly(dat, nfactors = 5, rotate = "geominQ")
3  print(result.fa, sort = T, cut = 0.3)
```

■コード解説

1 行目 サンプルデータ bfi の加工をしています。

[2] ちなみに N1 は「すぐ怒る (Get angry easily)」，N4 は「割と凹む (Often feel blue)」です。日本語訳は小杉の意訳で定訳ではありません。あしからず。

2 行目 ポリ個リック相関係数に基づく因子分析の実行

3 行目 結果の出力。

コードの 2 行目を見ていただくとわかる通り，ただの fa 関数ではなく fa.poly としただけで，あとは因子分析と同じです。この .poly のところがポリコリック相関係数を使うよう指定しているところで，これで実質的に測定尺度水準が順序尺度水準であると想定した因子分析をしたことになっています。

カテゴリカル因子分析は，段階反応モデルと数学的に等価であることが知られています。出自が違いますので，分析の数字や出力方法に違いがありますが[*3]，実質的には同じことをしていると理解してください。また，R ではたった数文字追加するだけで適切な分析方法になるのですから，使わない手はないと思います。

カテゴリカル因子分析は，因子分析の文脈から順序尺度水準の項目を扱えるように，と発展してきたものです，これに対して，項目反応理論の文脈から心理尺度を扱えるように，と発展してきたルートもあります。これは**多次元項目反応理論 (multi-dimensional item response theory)** と呼ばれるもので，これを提供する R パッケージもあります。それが mirt パッケージ (Chalmers, 2012) です。最後にこれを使ったコードの例を示しておきます。この計算には時間がかかりますが，完全情報最尤推定によって項目反応理論で算出するような，精度の高い**因子得点**を得ることができるのは利点だと言えるでしょう。

code : 11.7 mirt の実行

```
library(mirt)
result.mirt <- mirt(dat, 5)
print(result.mirt)
result.mirt %>% summary(rotate = "geominQ")
```

11.4 課題

今日の授業でおこなったすべての次の計算をする R コードを提出してください。ファイル形式は R スクリプトか Rmd とします。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

[*3] たとえば因子分析の文脈では，因子負荷量を提示して単純構造を目指すのに対し，項目反応理論の文脈では IRCCC を描いたり TIC を描いたりして各反応カテゴリの特徴を精査します。

第 12 章

構造方程式モデリング

本項では**構造方程式モデリング** (**Structural Equation Modeling, SEM**) について説明していきます[*1]。この手法はこれまで学んできた因子分析や回帰分析を統合したもので，分散共分散行列に方程式を作り込んでいくものです。我々が調査で得たデータから得られる情報は，分散共分散行列がすべてですから，その変数間関係を細かく作り込んでいき，行列の隅々まで考え尽くすという意味で，究極の手法といえるでしょう。実際，SEM は因子分析や回帰分析の他にも，多くのモデルをその下位モデルとして包含します。言い換えれば，SEM が登場する前に考えられてきたさまざまなモデルが，SEM の枠組みで理解・表現できるようになりました。であれば，この究極の技さえ知っておけば，非常に広範囲に拡張できるわけですから，こんなに便利なことはありません。

このように技術的には非常に高度なものでありますが，これを使うのは意外と簡単にできます。モデルをイメージで表現できる，パスダイアグラムの考え方から入っていきましょう。

12.1　パスダイアグラムの書き方

変数間の関係を表す図を**パス図**あるいは**パスダイアグラム** (**Path Diagram**) と言います。

分析する際の変数には**観測変数** (**Observed Variables**) と**潜在変数** (**Latent Variables**) があります。観測変数はその名の通り，観測された変数であり，データとして数値化されたものになります。これに対して潜在変数とは，モデルで仮定された変数のことです。たとえば因子分析やテスト理論では，因子や学力といったものを想定します。それが潜在変数です。潜在変数はデータとして数値化されているのではなく，抽象的な概念ですが，図にする場合はそれも表現しなければなりません。そこで観測変数を矩形 (長方形) で，潜在変数を楕円で表現することにします。

また，変数同士の関係を表現する必要がありますが，関係には**因果関係**と**相関関係**があります。調査データから因果関係を見出すのは原理的に不可能ですが，モデル上は一方が他方の原因になっている，と考えることがあります。回帰分析はその典型例で，説明変数が原因となって，被説明変数のあたいが結果的に変わる，という関係ですね。こうした関係を一方向の矢印で表現します。他方，相関関係は因果関係よりも緩やかで，何らかの関係がある，ということを意味しているに過ぎません。因果の向きが定まっているのではなく，どちらの向きにも影響しうる，ということで双方向矢印でもってこれを表現します。

図 12.1 にこの表記方法をまとめました。準備は実はこれだけです。この表記方法を使って，さまざまなモデルが統合的に表現できます。たとえば**回帰分析**を考えてみましょう。説明変数も被る説明変数も観測されている変数を使います。一方が他方の因果的関係にあるので，パスダイアグラムで表現すると図 12.2 の上図

[*1] この手法は別名**共分散構造分析**という名前でも呼ばれています。書籍や論文を検索する時は，こちらの名称もキーワードにしてみてください。

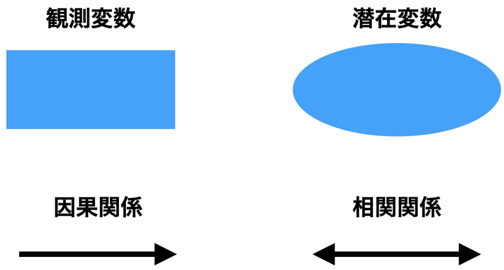


図 12.1　変数と関係の表記法

のようになります。ここで**残差**はデータになく，モデルの計算結果から想定される潜在的な変数なので楕円で描かれていることに注意してください。また**重回帰分析**は図 12.2 の下図のようになります。説明変数が複数に増えただけですので，表記上はこのように変わるわけですね。

なお，この因果関係を表す矢印の上に (偏) 回帰係数を書いて，影響力の大きさを表現することがあります。構造方程式モデリングでは，この矢印によって表される影響力の強さは**パス係数 (path coefficient)** という呼び方で統一されます。

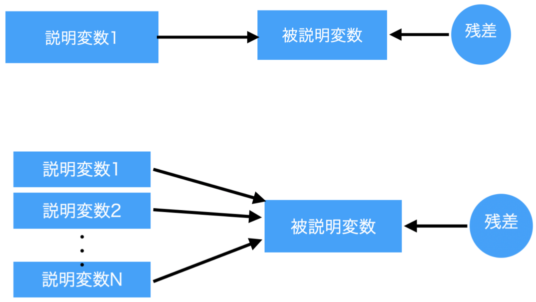


図 12.2　回帰分析と重回帰分析

では**因子分析**をパスダイアグラムで表現するとどうなるでしょうか。複数の項目の背後にある共通要因を潜在変数として仮定するので，パスダイアグラムで表現すると図 12.3 のようになります。これは 1 因子モデ

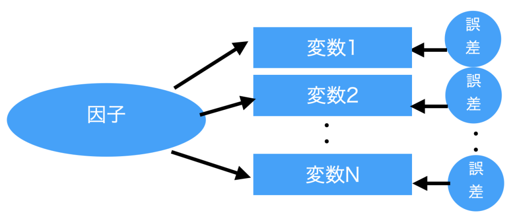


図 12.3　因子分析モデル

ルですが，多因子モデルは基本的に左側にくる潜在変数が複数になるだけです。潜在変数 (因子) からのパス係数は，**因子負荷量**なのです。また図 12.2 と図 12.3 を見比べると，回帰分析と因子分析も形はよく似て

いることがわかります。両者の違いは，説明変数が観測変数か潜在変数かであり，因子分析とは説明変数が潜在変数になった回帰分析モデル，ということもできるのです。

このように，今まで見てきた統計モデルをパスダイアグラムを使って表現できます。

12.2 パスダイアグラムによるさまざまなモデル

パスダイアグラムはさまざまな分析法を統合的に表現するツールであることがわかりました。この方法を使って他の統計モデルも見てみましょう。

■主成分分析　このコースではここまで取り上げてきませんでしたが，因子分析とよく似た手法で**主成分分析 (Principle Component Analysis)** というのがあります。主成分分析の目的は観測変数を重みつき線型結合させ，個々のデータをもっともよく説明するような主成分という合成変数を作り出すことです。たとえば体力テストなどで複数の競技の記録をつけた後，総合的に誰が一番運動能力があるのか，というときにこの分析が使われたりします。この分析方法をパスダイアグラムで書くと図 12.4 のようになります。

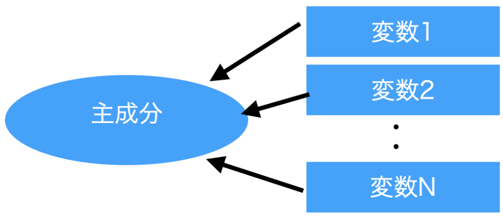


図 12.4　主成分分析モデル

これを見ると，因子分析と似ている，でもちょっと違う，というのがよくわかりますね。因子分析は潜在的な説明変数を作るのですが，主成分分析は潜在的な被説明変数を作る分析方法なのです。

その違いは，誤差の扱いにも関わってきます。これまでの例にあるように，パスダイアグラムでは基本的に矢印が入った変数には誤差 (残差) がつくのが決まりです。因子分析は説明変数が潜在的だったので，観測変数にはそれで説明できない残りが潜在変数として構成されました。主成分分析は観測変数が矢印の出所になりますので，そこには誤差が想定されません。作られた潜在変数は矢印が入ってきた方ですが，これはモデルで構成される変数なので誤差を考えることができません[*2]。主成分分析はデータに誤差を考えないときに有用なモデルですから，経済学などデータがはっきりしたものである領域でよく用いられてきました。これに対して心理統計の領域では，測定されたスコアに誤差が入ると考えるのが当然ですから，因子分析の方がよく用いられてきたのです。主成分分析と因子分析は数学的にはよく似た解法で答えを出すのですが，その背後にある考え方がまったく異なります[*3]。

■尺度水準の違いによるモデルの違い　たとえば因子分析において観測変数が**順序尺度水準**であれば，同じパスダイアグラムであってもそれはカテゴリカル因子分析をしたことになります。パスダイアグラムでは変

[*2] 成分も誤差もモデルから構成されるものなので，モデル上それらを区分する方法がないからです。

[*3] もう少し丁寧にいうと，主成分分析も因子分析も計算する上では正方行列の固有値分解をすることになります。ここで，因子分析の場合は項目に誤差を考えますから，固有値分解をするときに相関行列 $\boldsymbol{R}$ ではなく，その対角項に共通性を入れた $\boldsymbol{R}\dagger$ から計算を始めるのでした。主成分分析は誤差を考えませんから，$\boldsymbol{R}$ から計算したり，単位に意味がある場合が少なくないので分散共分散行列 $\boldsymbol{S}$ の固有値分解でもってパス係数を算出するという違いです。因子分析において共通性の推定を避け，$\boldsymbol{R}$ を分解する方法をとくに因子分析の主成分解というのが，初学者を混乱させるポイントになっています。

数の尺度水準を表現することはありませんが，言い換えると尺度水準が違うだけで別の分析モデルになるということです。

たとえば回帰分析において，被説明変数が離散的 (**名義尺度水準**) なモデルは，かつて**判別分析** **(Descriminant Analysis)** と呼ばれていました。同様に被説明変数が**順序尺度水準**で得られた場合は，**順序ロジットモデル (Ordinal Logit Model)** や**順序プロビットモデル** Ordinal Probit Model と呼ばれたモデルです (2 水準ならロジットモデル，3 水準以上はプロビットモデル)。

また回帰分析において，説明変数が離散的 (名義尺度水準) なモデルは，**分散分析 (ANalysis Of VAriance;ANOVA)** と呼ばれていたのでした。とくに説明変数が 2 カテゴリの場合は **t 検定 (t-test)** として知られています。これらは平均値差の検定という文脈で説明されてきたかと思いますが，いずれも回帰分析，もとい**線形モデル**の一種であるというのはご存知の通りです。

色々な分析法，分析モデル名がありますが，**パスダイアグラム**で表現すれば同じである，というのがポイントです。**構造方程式モデリング**はその下位モデルとしてこれらの分析をすべて含む，あるいは統合的な表現であるというのはそういう意味です。このように統合される以前は，それぞれの分析方法をそれぞれのソフトウェアや関数で実行する必要があったのですが，今は SEM が扱えるソフトウェア，関数 1 つで分析できるのです。大変ありがたいことではないですか！考えるべきポイントもすべてパス係数や適合度といった統合的指標でよいのですから。

■自由なモデルへ　因子分析と回帰分析，あるいはその他のさまざまな統計モデルが，パスダイアグラムによって統合的に表現できるようになりました。その利点は表現が統一化されただけではなく，同じプラットフォームで表現できるのですから，たとえば図 12.5 のような表現もできるということです。

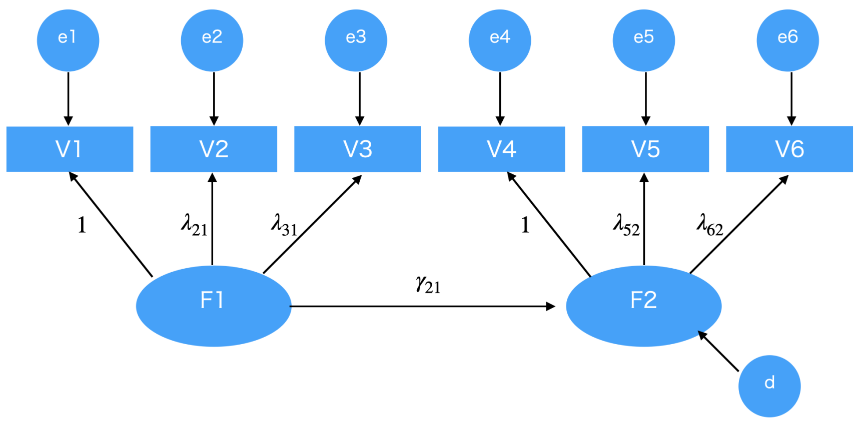


図 12.5　構造方程式と測定方程式。記号の意味は後述

図 12.5 には因子分析がふたつと，因子と因子の関係を矢印で結んだパスが描かれています。潜在変数の間に回帰分析を組み込んだようなものですね。イメージとしては，ある心理変数が別の心理変数に影響を与える (ex. 外向性が抑うつ傾向に影響する，など) といったものを表しているわけです。心理学者は心理的変数がどう影響しあっているか，関係しあっているかをみていきたいわけですから，やりたいことは要するにこういう複合的なモデルによる検証だったのだ，と言っても過言ではないでしょう。このような分析も，SEM のおかげで簡単にできるようになりました。ここで観測変数から潜在変数を構成する箇所をとくに**測定方程式**，潜在変数同志の因果関係などそれ以外の関係を表現する箇所を**構造方程式**と呼んで区別することがありますが，**構造方程式モデリング**はその両者からなるモデル全体のこと，ということができるでしょう。

構造方程式モデリングを使うと，調査などから得たデータ全体の見取り図を描いて分析できるのです。言い換えると，実際に調査を実施する前に，どの変数とどの変数がどのような関係にあるのか，という仮定を考えて調査票をデザインし，過不足なく測定してモデルを検証する，という**仮説検証型**の分析ができるのです。それまでの調査は，どういう変数がどういう関係になっているか細かくは分からないけれども，何か関係しそうだからまとめて調査項目にしておいて，因子分析や回帰分析を繰り返して，変数間の関係を**探索的に**研究するということがよく行われていました。もちろん今でも探索的研究はあるのですが，既存の尺度や先行研究がある場合はより具体的に調査を設計できるようになりました。

因子分析も，因子数がわからないのでスクリープロットなどを描いて探し出し，因子と項目の関係がわからないのですべての因子がすべての項目に影響すると考えて分析する**探索的因子分析 (Exploratory Factor Analysis,EFA)** がかつては主流でしたが，今ではどの因子がどの項目で測定されるかを明確にパスダイアグラムで表現して分析する，**検証的因子分析 (Confirmatory Factor Analysis, CFA)** が行われるようになってきています。

12.3 構造方程式モデルによる未知数の推定

12.3.1 未知数は本当に推定できるのか

さて，さまざまなモデルをパスダイアグラムで表現できる，ということがわかりましたが，「方程式」という言葉はどこからきているのでしょう？実は，統計モデルはパスダイアグラムでも表現できますが，同じことを方程式でも表現できます。

たとえば図 12.1 で表した回帰分析モデルは，次のような方程式で表せるのでした。

$$y = b_0 + b_1 x + e$$

図 12.2 に表した因子分析モデルも，同様に次のような方程式で表せるのでした (項目が 3 つの場合)。

$$\begin{cases} x_1 &= \lambda_1 f + e_1 \\ x_2 &= \lambda_2 f + e_2 \\ x_3 &= \lambda_3 f + e_3 \end{cases}$$

ここで f は因子得点，λ_j は因子負荷量，あるいはパス係数を表しています。

この因子分析モデルを例に，この方程式をどのように解くかをみていきましょう。そのために，いくつかの記号と特徴を確認します。まずベクトルの平均を関数 $\boldsymbol{E}$ で表すことにします。誤差の平均は 0，という特徴は $\boldsymbol{E}[\boldsymbol{e}] = \mathbf{0}$ と表すことができます。因子得点も標準化されていると仮定しますので，$\boldsymbol{E}[\boldsymbol{f}] = \mathbf{0}$ ですね。つぎにベクトルの分散を関数 $\boldsymbol{V}$ で表すとします。データベクトル $\boldsymbol{x}$ が平均 0，あるいは平均偏差，あるいは標準化されていると考えると，$\boldsymbol{V}[\boldsymbol{x}] = \boldsymbol{E}\left[(\boldsymbol{x} - \boldsymbol{E}[\boldsymbol{x}])^2\right] = \boldsymbol{E}[\boldsymbol{x}\boldsymbol{x}']$ は分散共分散行列 $\boldsymbol{\Sigma}$ だと考えることができます。また因子は標準化されているので $\boldsymbol{V}[\boldsymbol{f}] = 1$，誤差同士は相関しないものの，誤差には分散がありますので $\boldsymbol{V}[\boldsymbol{e}] = \boldsymbol{E}[\boldsymbol{e}\boldsymbol{e}'] = \Psi$ とします。ここで Ψ は対角に誤差分散が入った対角行列です。

さて，さきほどの 3 項目 1 因子モデルを行列で表現してみましょう。各要素を次のようにベクトルでまとめて表現します。

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix}, \boldsymbol{e} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix}$$

すると式は次のようにあっさり表現できます。

$$\boldsymbol{x} = \boldsymbol{\Lambda}\boldsymbol{f} + \boldsymbol{e}$$

ここからデータの分散共分散行列 $\mathbf{\Sigma}$ を考えてみましょう。

$$
\begin{aligned}
\boldsymbol{V}[\boldsymbol{x}] &= \boldsymbol{E}[\boldsymbol{x}\boldsymbol{x}'] \\
&= \boldsymbol{E}[(\mathbf{\Lambda}\boldsymbol{f}+\boldsymbol{e})(\mathbf{\Lambda}\boldsymbol{f}+\boldsymbol{e})'] \\
&= \boldsymbol{E}\left[(\mathbf{\Lambda}\boldsymbol{f}+\boldsymbol{e})(\boldsymbol{f}\mathbf{\Lambda}'+\boldsymbol{e}')\right] \\
&= \boldsymbol{E}\left[\mathbf{\Lambda}\boldsymbol{f}^{\mathbf{2}}\mathbf{\Lambda}'+\mathbf{\Lambda}\boldsymbol{f}\boldsymbol{e}'+\boldsymbol{e}\boldsymbol{f}\mathbf{\Lambda}'+\boldsymbol{e}\boldsymbol{e}'\right] \\
&= \mathbf{\Lambda}\mathbf{\Lambda}' + \Psi
\end{aligned}
$$

これをエレメントワイズで書き出すと次のようになります。

$$
\begin{aligned}
\begin{pmatrix} s_1^2 & s_1s_2 & s_1s_3 \\ & s_2^2 & s_2s_3 \\ & & s_3^2 \end{pmatrix} &= \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} \begin{pmatrix} \lambda_1 & \lambda_2 & \lambda_3 \end{pmatrix} + \begin{pmatrix} \sigma_1^2 & & \\ & \sigma_2^2 & \\ & & \sigma_3^2 \end{pmatrix} \\
&= \begin{pmatrix} \lambda_1 1^2 + \sigma_1^2 & \lambda_1\lambda_2 & \lambda_1\lambda_3 \\ & \lambda_2^2 + \sigma_2^2 & \lambda_2\lambda_3 \\ & & \lambda_3^2 + \sigma_3^2 \end{pmatrix}
\end{aligned}
$$

この式の左辺はデータから計算された分散共分散行列，右辺はモデルから計算された分散共分散行列です。行列の下三角が消えているのは対称行列で同じ情報が入るからで，データからオリジナルに得られるのは 3 つの分散 (s_1^2, s_2^2, s_3^2) と 3 つの共分散 (s_1s_2, s_1s_3, s_2s_3) の情報になります。一方右辺での未知数はパス係数 (因子負荷量) の $\lambda_1, \lambda_2, \lambda_3$ と誤差分散 $\sigma_1^2, \sigma_2^2, \sigma_3^2$ ということになります。未知数の数と既知数の数が同じなので，この方程式は解けます。とくに，未知数と既知数の数が一致しているので，**丁度識別 (just identification)**，あるいは単に**識別可能 (identifiable)** といいます。

分散共分散行列で提供される情報の増え方に比べて，モデルの未知数の増え方は小さいので，一般的な SEM のモデルのほとんどは**過剰識別**になります。つまりいくつかの答えがあり得る，ということになるので，その時は尤度が最も高くなるように，といった基準を加えて答えを一意に定めます。逆に未知数の方が多い場合**識別不可能** (解が求められない) ということになり，その場合はどこかのパラメータを値として入れてやる (**制約**をかける，といいます) 必要があります。

12.3.2　複雑なモデルの場合

では次に，図 12.5 のような複雑なモデルでも方程式で表せることを示しましょう。

ここで図 12.5 の係数について説明します。ギリシア文字 λ や γ がパス係数を表しています。パス係数の添字は，被説明変数の変数番号 j と，説明変数の変数番号 k をつかって λ_{jk} のようにかき，これで $k \to j$ の影響力の大きさを表していると思ってください。測定方程式の係数を λ で，構造方程式の係数を γ で表現しました。また $\lambda_{11}, \lambda_{42}$ になるべきところが 1 になっていますが，これは「潜在変数から出るパスのうち，1 つは必ず 1 にしないと推定できない」という数学的特徴があるからです。決まりごとだと思ってください。

それを踏まえて，モデルの方程式を書くと，次のようになります。

$$
\begin{cases}
V_1 &= f_1 + e_1 \\
V_2 &= \lambda_{21}f_1 + e_2 \\
V_3 &= \lambda_{31}f_1 + e_3 \\
V_4 &= f_2 + e_4 \\
V_5 &= \lambda_{52}f_2 + e_5 \\
V_6 &= \lambda_{62}f_2 + e_6 \\
f_2 &= \gamma_{21}f_1 + d
\end{cases}
$$

ややこしいだけで，書けるのは書けましたね！そして当然，これは行列表現した方が楽ではないか，というアイデアが浮かぶと思います。その通りで，エレメントワイズにまずは書いてみましょう。

$$\begin{pmatrix}V_1\\V_2\\V_3\\V_4\\V_5\\V_6\\f_2\\f_1\end{pmatrix}=\begin{pmatrix}0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&\lambda_{21}\\0&0&0&0&0&0&0&\lambda_{31}\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&\lambda_{52}&0\\0&0&0&0&0&0&\lambda_{62}&0\\0&0&0&0&0&0&0&\gamma_{21}\\0&0&0&0&0&0&0&0\end{pmatrix}\begin{pmatrix}V_1\\V_2\\V_3\\V_4\\V_5\\V_6\\f_2\\f_1\end{pmatrix}+\begin{pmatrix}e_1\\e_2\\e_3\\e_4\\e_5\\e_6\\d\\f_1\end{pmatrix}$$

左辺のベクトルの最後に f_1 が入っているところがポイントで，これは方程式的には $f_1=f_1$ というだけの式なのですが，これを入れたおかげで全体を整合的に表現できましたね。この左辺のベクトルを $\boldsymbol{v}$ として，式全体を $\boldsymbol{v}=\boldsymbol{G}\boldsymbol{v}+\boldsymbol{e}$ と考えてみましょう。すると次のように展開できます。

$$\begin{aligned}\boldsymbol{v}&=\boldsymbol{G}\boldsymbol{v}+\boldsymbol{e}\\\boldsymbol{v}-\boldsymbol{G}\boldsymbol{v}&=\boldsymbol{e}\\(\boldsymbol{I}-\boldsymbol{G})\boldsymbol{v}&=\boldsymbol{e}\\(\boldsymbol{I}-\boldsymbol{G})^{-1}(\boldsymbol{I}-\boldsymbol{G})\boldsymbol{v}&=(\boldsymbol{I}-\boldsymbol{G})^{-1}\boldsymbol{e}\end{aligned}$$

ここで $\boldsymbol{P}=(\boldsymbol{I}-\boldsymbol{G})$ とすると，

$$\boldsymbol{v}=\boldsymbol{P}^{-1}\boldsymbol{e}$$

ということになります。これをエレメントワイズで書くと次のようになります。

$$\begin{pmatrix}V_1\\V_2\\V_3\\V_4\\V_5\\V_6\\f_2\\f_1\end{pmatrix}=\begin{pmatrix}1&0&0&0&0&0&0&-1\\0&1&0&0&0&0&0&-\lambda_{21}\\0&0&1&0&0&0&0&-\lambda_{31}\\0&0&0&1&0&0&-1&0\\0&0&0&0&1&0&-\lambda_{52}&0\\0&0&0&0&0&1&-\lambda_{62}&0\\0&0&0&0&0&0&1&-\gamma_{21}\\0&0&0&0&0&0&0&1\end{pmatrix}\begin{pmatrix}e_1\\e_2\\e_3\\e_4\\e_5\\e_6\\d\\f_1\end{pmatrix}$$

さて，変数からなるベクトル $\boldsymbol{v}$ ではありますが，観測変数は V_1 から V_6 までしかありませんから，そこだけを取り出す行列 $\boldsymbol{F}=\{\boldsymbol{I},\boldsymbol{O}\}$ を考えます。ここでは次のような行列です。

$$\boldsymbol{F}=\begin{pmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\end{pmatrix}$$

これを使って，次のように関係を書き直すことができます。

$$\boldsymbol{g}=\begin{pmatrix}V_1\\V_2\\V_3\\V_4\\V_5\\V_6\end{pmatrix}=\boldsymbol{F}\boldsymbol{P}^{\boldsymbol{-1}}\boldsymbol{e}$$

さて，ではデータだけのベクトル $\boldsymbol{g}$ から分散共分散行列を構成するとしましょう。データベクトル $\boldsymbol{g}$ は平均が 0 だと仮定します[*4]。そうすると分散共分散行列 $\boldsymbol{\Sigma}$ は次のようになります。

$$\begin{aligned}\boldsymbol{\Sigma} &= \boldsymbol{g}\boldsymbol{g}' \\ &= \boldsymbol{F}\boldsymbol{P}^{-1}\boldsymbol{e}(\boldsymbol{F}\boldsymbol{P}^{-1}\boldsymbol{e})' \\ &= \boldsymbol{F}\boldsymbol{P}^{-1}\boldsymbol{e}\boldsymbol{e}'\boldsymbol{P}^{-1'}\boldsymbol{F}'\end{aligned}$$

ここで式の中にある $\boldsymbol{e}\boldsymbol{e}'$ は誤差などの未知数で作った分散共分散行列になっています。誤差は他の変数との共分散がないと考えられるので，これは結局対角行列です。

$$\boldsymbol{e}\boldsymbol{e}' = \begin{pmatrix} V(e_1) & & & & & & & \\ & V(e_2) & & & & & & \\ & & V(e_3) & & & & & \\ & & & V(e_4) & & & & \\ & & & & V(e_5) & & & \\ & & & & & V(e_6) & & \\ & & & & & & V(d) & \\ & & & & & & & V(f_1) \end{pmatrix}$$

さあややこしい話はここまでにして，何を考えていたのかをもう一度考えてみましょう。

この式の左辺は $\boldsymbol{\Sigma}$，すなわちデータからできる分散共分散行列であり，具体的に数字の計算のできるデータセットです。この式の右辺は $\boldsymbol{F}\boldsymbol{P}^{-1}\boldsymbol{e}\boldsymbol{e}'\boldsymbol{P}^{-1'}\boldsymbol{F}'$ であり，係数と潜在変数の分散からなる方程式です。左辺にデータ，右辺にモデルで表した分散共分散行列があり，これがイコールで結びついているのですから，この連立方程式を解けば，係数を求めることができます。

そりゃその計算はとても面倒なことになると思われますが，そこは機械がやってくれるのだ，と割り切ればいいでしょう。モデルがデータに最も合うように，未知数を決めていく方法は，**最小二乗法**や**最尤法**など，今まで学んできた手法を使えばよいのです！

こうして自由なモデリング技術とその解法を私たちは手に入れることができました。構造方程式モデリングとは，変数間関係を細かく作り込んでいき，分散共分散行列の隅々まで考え尽くす究極のモデル，と意味が伝わったでしょうか。

12.4 適合度によるモデルの評価

データ行列とモデル行列をイコールで結んだ方程式を解くことが，未知数を求める根本的な原理であるということがわかりました。方程式が多く未知数が少ない場合，値が一意に定まらな苦なってしまいます[*5]。これはパラメータに**自由度 (degrees of freedom)** がある，ということもできます。そのような場合は別の基準をつかって最も良いと考えられる数字を推定値として用いることになります。

さてなんとかモデルをデータに当てはめることができたとして，それがどの程度一致しているか，ということは別次元で評価しなければなりません。それが**適合度 (fit index)** です。SEM の文脈では複数の適合度が考えられており，それらを総合的に評価するというやり方がとられています。ここでは複数の適合度指標をご紹介しておきます[*6]。

[*4] ずいぶんと勝手な仮定だな，と思うかもしれませんが，分析の元になる変数ベクトルを標準化したと思っていただければそれほどおかしなことでもない，と考えてもらっても構いません。あるいは単に変数ごとの平均をとった平均偏差ベクトルを作って，それをデータだとしたと考えてください。

[*5] たとえば $x + 5 = 8$ と $x - 3 = 9$ という連立方程式があればどうでしょうか。未知数は 1 つ，式は 2 つなのでどちらが正解なのかわかりません！

[*6] 以下の記述は小杉・清水 (2014) をもとに加筆修正したものです。

■GFI・AGFI　**GFI(Goodness of Fit Index)** や **AGFI(Adjusted GFI)** は，0 から 1 までの値を取る適合度指標で，因果モデルがデータの何 % を説明したかの指標になります。1 に近いほど良いモデルで，GFI で 0.95,AGFI で 0.90 以上の数字であると良い適合と判断されます。

■情報量規準　**情報量規準 (Information Criteria)** とは，一般的な統計モデルを評価するための指標で，**AIC(Akaike's Infromation Criterion)** や **BIC(Bayesian I.C.)** などがそれにあたります。これは今検証しているモデルが，データによくフィットする「本当の」モデルからどの程度離れているかを示す指標です。本当のモデル，というのは分かり得ませんから，複数のモデル A,B,C を同じデータに当てはめたとき，AIC や BIC が相対的に小さいものが，より良く適合していると判断します。相対的な指標ですから，違う変数を扱うモデル間の比較はできないことに注意が必要です。

■RMSEA　モデルがデータとどの程度乖離しているか、を直接表現した指標が **RMSEA(Root Mean Square Error of Approximation)** です。近似した誤差に対する平方平均平方根，ということですね。一般に 0.05 より小さければ，そのモデルは当てはまりがよく，0.1 以上であれば当てはまりが悪いと判断します。

■CFI/TLI　**CFI(Comparatative Fit Index)** や **TLI(Tucker-Lewis Index)** は，観測変数間にまったく相関がないという非現実的なモデル（独立モデルという）に比べて，当該のモデルがどの程度よいものかを指標化したものです。いずれも 1.0 に近ければ近いほど良いモデルとされています。一般にこの数値が 0.9 以上になることを目指してモデルを改訂していきます。

■SRMR　**SRMR(Standardized Root Mean square Residual)** は標準化された残差平方平均平方根を表します。これはモデルで説明できなかったものがどれほどあったか，を表しますので，0 に近ければ近いほど当てはまりの良いことを表します。

■修正指数　最後に，**修正指数 (modification index)** を紹介しておきます。上で述べたように，検証しようとした統計モデルがどの程度当てはまっているか，ということを数値化できるようになったわけですが，常に最初にたてた仮説モデルが最適である，ということは滅多にありません。データから仮説の訂正を余儀なくされることがある，あるいはもっと良くなったり新しい発見があったりすることが，統計モデリングの醍醐味でもあります。修正指数は，どこをどう改良すれば指標がよくなりますよ，というヒントを与えてくれるものです。もっとも，適合度を上げることだけが目的なのではないことに注意してください。適合度を上げるために，意味のわからないモデルを作り上げたところで，モデルの意味がないのですから。

構造方程式モデリングは，非常に複雑な方程式を解いているんだな，ということは理解していただけたかと思います。次回はそんな複雑なモデルでも，R で簡単に分析できるよということを，演習を交えて理解していきましょう。

12.5 課題

2 項目 1 因子モデルの方程式，およびそれを行列形式で表したものを書きましょう。

また行列の要素で書いた場合，既知数と未知数はどちらが多くなるでしょうか。言い換えれば，このモデルは識別できるでしょうか。式を展開して確認してください。

第 13 章

R による構造方程式モデリング

これまでの流れと同じで，統計技術の理論を知っただけではなく，自分で実際に計算できる演習を経てこそ理解が深まります。本講では R をつかって実際に構造方程式モデリングを解くことを演習的に学んでいきましょう。構造方程式モデリングを実装するパッケージは複数あるが，最も応用範囲がひろい lavaan パッケージを用いることにします。授業を始めるにあたって，まずは lavaan パッケージをインストールしておいてください[*1]。

13.1　モデル式の入力

13.1.1　観測変数だけのモデル

まずは観測変数同士の関係をパスでつなぐモデルをみてみましょう。観測変数同士のパスですから，(重) 回帰分析をやるようなものと同じです。今回のデータとして lavaan パッケージに含まれている HolzingerSwineford1939 データセットを使います。これは (Holzinger and Swineford, 1939) のデータで 301 人に対して行われた 15 の能力テストスコアの一部が入ったデータです。一部を 13.1 に示しました。

表 13.1　Holzinger and Swineford(1939) のデータ

id	sex	ageyr	agemo	school	grade	x1	x2	x3	x4	x5	x6	x7	x8	x9
1	1	13	1	Pasteur	7	3.33	7.75	0.38	2.33	5.75	1.29	3.39	5.75	6.36
2	2	13	7	Pasteur	7	5.33	5.25	2.12	1.67	3.00	1.29	3.78	6.25	7.92
3	2	13	1	Pasteur	7	4.50	5.25	1.88	1.00	1.75	0.43	3.26	3.90	4.42
4	1	13	2	Pasteur	7	5.33	7.75	3.00	2.67	4.50	2.43	3.00	5.30	4.86
5	2	12	2	Pasteur	7	4.83	4.75	0.88	2.67	4.00	2.57	3.70	6.30	5.92
6	2	14	1	Pasteur	7	5.33	5.00	2.25	1.00	3.00	0.86	4.35	6.65	7.50

変数の意味は以下の通りです。x1 から x3 が空間的知覚，x4 から x6 が言語的能力，x7 から x9 が移動する物体の認識力を測るものになっています。

- id　被験者 ID
- sex　性別 (男性=1, 女性=2)
- ageyr　生まれ年

[*1] R のコンソールに install.packages("lavaan") と入力するか，RStudio のパッケージタブから入力しましょう。

agemo　生まれ月
school　所属校
grade　学年
x1　視覚的知覚
x2　立方体
x3　菱形
x4　段落の理解
x5　文章完成
x6　言葉の意味
x7　加速
x8　点を数える
x9　直線・曲線文字の区別

さて，ではまず回帰分析をやってみることにしましょう。変数 x4 を x5,x6 で回帰する重回帰モデルを `lavaan` を実行するには `code:13.1` のように書きます。

code : 13.1　重回帰モデル

```
1 model1 <- "
2 x4␣~␣x5␣+␣x6
3 "
4 result1 <- sem(model1, data = HolzingerSwineford1939)
5 summary(result1, fit.measures = T)
6 # 比較
7 result1.1 <- lm(x4 ~ x5 + x6, data = HolzingerSwineford1939)
8 summary(result1.1)
```

■コード解説

1-3 行目　モデルの記述
4 行目　関数による推定
5 行目　結果の出力
6-8 行目　従来の `lm` 関数による当てはめ例

ここでモデルを書くところは，ダブルクォーテーションで括られていることに注意してください。このようにすることで，R に複数行からなるモデルを渡すことが可能になります。実際のモデルは 2 行目だけで，これは `lm` 関数に書いているモデル式の形と同じであることがわかります。モデルを書いて，モデルとデータの組みを `sem` 関数に渡すと結果が帰ってくる，これだけです。結果を出力するときに，オプションとして `fit.measures` を書いてあるところだけが違います。

出力結果の一部を出力 13.1 に示します。実際はもっと色々でているのですが，CFI，TLI, AIC,BIC, RMSEA, SRMR などはそれぞれ適合度指標を荒らしています。推定結果は `Regression:` の `Estimate` のところを見てください。これらは，`lm` 関数と同じ係数になっていることが確認できます。

R の出力 13.1: 観測変数だけのモデル

```
    (中略)

    User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    1.000
  Tucker-Lewis Index (TLI)                       1.000

Loglikelihood and Information Criteria:

    (中略)

Akaike (AIC)                                   673.134
  Bayesian (BIC)                                 684.255
  Sample-size adjusted Bayesian (BIC)            674.741

Root Mean Square Error of Approximation:

  RMSEA                                          0.000

  (中略)

Standardized Root Mean Square Residual:

  SRMR                                           0.000

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  x4 ~
    x5                0.423    0.047    8.958    0.000
    x6                0.390    0.056    7.002    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .x4                0.537    0.044   12.268    0.000
```

このように，重回帰モデルをするのであれば記述方法は同じです。違いはさまざまな観点からの適合度指標モデルということ。加えて，このモデルはどんどん拡張していくことができる点にあります。

たとえば**パス解析 (Path Analysis)** というのがあります。これは影響力のルートが $x \to y \to z$ のように，次から次へと繋がっていくモデルです。SEM ができるまでは，まず $x \to y$ を回帰分析し，次に $y \to z$ を分析する，という方法でした。しかしこれでは R^2 など適合度を逐一確認する必要があり，またモデル全体の評価ができないという欠点があります。しかし SEM ではコード code:13.2 のように書くだけです。

code：13.2　パス解析のモデル

```
1 model2 <- "
2 x4␣~␣x5
3 x5␣~␣x6
4 x6␣~␣grade
5 "
6 result2 <- sem(model2, data = HolzingerSwineford1939)
7 summary(result2, fit.measures = T, standardized = T)
```

このモデル式 (2-6 行目) にあるように，$grade \to x6 \to x5 \to x4$ という一連の影響力を分析し，一気に適合度を表現してくれています。出力結果には適合度指標のほか，すべての変数を標準化した標準化係数を出すオプションを追加しました。

これを図にしたのが図 13.1 です。ちなみにこの図も R のパッケージで書きました。パス図を自動的に書いてくれるパッケージとして `semPlot` パッケージ (Epskamp, 2021) や `tidySEM` パッケージ (Van Lissa, 2019) などを使います。ここでは `tidySEM` パッケージの `graph_sem` 関数を使いました。

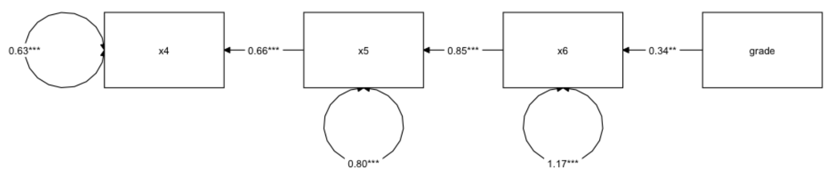


図 13.1　パス解析の図

13.1.2　測定方程式を入れたモデル

今度は測定方程式を入れたモデルを考えてみましょう。回帰モデルは従属変数と独立変数をチルダ (~) でつなぎましたが，潜在変数を作る場合は=~で右辺に観測変数，左辺に潜在変数をセットします。コード例を `code:13.3` に示します。

code：13.3　因子分析モデル

```
model3 <- "
␣visual␣=~␣x1␣+␣x2␣+␣x3
textual␣=~␣x4␣+␣x5␣+␣x6
␣␣speed␣=~␣x7␣+␣x8␣+␣x9
"
result3 <- sem(model3, data = HolzingerSwineford1939)
summary(result3, fit.measures = T, standardized = T)
```

これのモデル図は図 13.2 のようになります[*2]。モデル上とくに指定はしてありませんが，潜在変数同士の相関係数も自動的に推定されています。パス係数は潜在変数からでてきていますから，**因子負荷量**のように考えることができますね。

このモデルは因子分析の一種ですが**探索的因子分析**とはいくつかの点で違います。1 つは因子数が 3 である，と最初から決めていた点。2 つ目は因子負荷量ですが，探索的因子分析の場合はどの項目に対しても共通因子からの因子負荷量が計算されていました。今回の例では，x4 から x9 の変数にすいても visual 因

[*2] この図は小野島 (2021) の関数を使っています。tikz を使って綺麗なモデルが描ける素晴らしい関数の提供に感謝。

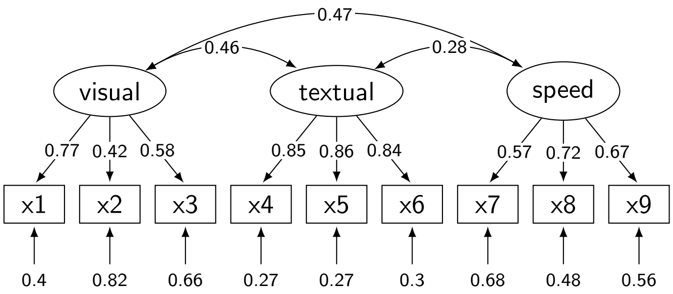


図 13.2　潜在変数モデル

子からのパスが出ている，というモデルだったのです。今回のモデルではそれらのパスはなく，visual 因子は x1,x2,x3 にしか影響していません。言い換えると，x4 から x9 に対する visual 因子のパス係数は 0 である，と特定したようなものです。

探索的因子分析の場合，理想的には「因子が関係する項目には十分影響しているが，関係しない項目にはまったく影響しない」という**単純構造の原理 (Principle of Simple Structure)** が理想とされていました。SEM を使うとこの理想的なモデルが合っているかどうかを検証する＝モデル上で制約をかけてその適合度を評価する，という使い方ができます。このような方針の因子分析のことを**検証的因子分析 (Confirmatory Factor Analysis)** と呼んで，探索的因子分析と区別します。

検証的因子分析をすることは，関係のない項目への影響を 0 とする，という強い仮定を置いていることにもなりますが，このことによって**因子的妥当性**や**収束的妥当性**，**弁別的妥当性**を検討している，と考えることもできます[*3]。もしこの理論的なモデルの当てはまりが悪ければ，異なる因子負荷のパターンを考えなければなりません。あるいは因子数を変えてモデルを組まなければならないかもしれないのです。因子数は同じだけど因子負荷量が違う，といったモデルを考えることもできますし，因子負荷量も固定して「この値に違いない」としたモデルを検証する，ということもできます。モデルを当てはめるデータは違っても構いません。むしろ違うデータに同じモデルが当てはめられ，十分な適合度があればそれはそのモデルの普遍性があるということで，いいことなのです。

このように，SEM では同じモデルを男性と女性，学生と社会人，日本人データと外国人データといった複数のデータに当てはめて，どこまで同じでどこから違うか，と言った比較をできます。こうしたモデル比較のことを**多母集団同時分析 (Multi-group analysis)** といいます。

13.1.3　構造方程式モデルへ

それでは潜在変数同士の関係に，更なる仮定を入れたモデルを考えてみましょう。

code : 13.4　構造方程式モデル

```
model4 <- "
 visual =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
  speed =~ x7 + x8 + x9
 textual ~ visual + speed
```

[*3] 因子的妥当性とは，因子が十分に大きく項目に影響していることです。収束的妥当性とは，因子が関係しない項目に影響しないこと，弁別的妥当性は因子同士が別のものであると区別できることを表しています。

```
6   speed ~ grade
7 "
8 result4 <- sem(model4, data = HolzingerSwineford1939)
9 summary(result4, fit.measures = T, standardized = T)
10 modificationindices(result4) %>%
11   as_tibble() %>%
12   arrange(-mi)
```

■コード解説

1-7 行目　モデルの記述。visual, textual, speed の 3 因子をつくり，textual は visual と speed から説明されるモデル。さらに speed 因子は学年によっても説明される。

8 行目　関数による推定

9 行目　結果の出力。適合度指標と標準化係数の出力オプションをつけて。

10-12 行目　分析結果の**修正指数**を出力。ただし指標 `mi` の大きい順に並べ替えるために 11 行目で出力を `tibble` 型にし，`arrange` 関数で並べ替えている。

まずは今回のモデルについても，図でみたほうがわかりやすいでしょう。結果を合わせて図 13.3 に示してみました。

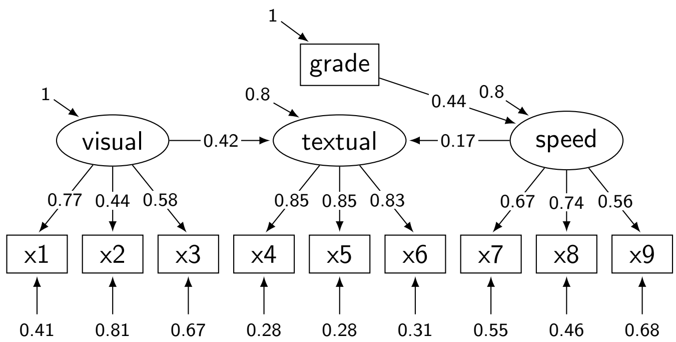


図 13.3　構造方程式モデルへ

この図の出力と，R での出力はどこが対応しているか，それぞれ確認しましょう。

まずは測定方程式のところです。因子負荷量，すなわち潜在変数からのパスはすべての変数を標準化したところの数字ですので，`Std.all` の列を参照します。この因子で説明できなかった独自成分の大きさは，`Varianvces` のところに表されています。

構造方程式として，textual 因子が visual 因子と speed 因子に説明されており，そのパス係数が `Regressions:` の `Std.all` 列に示されています。speed 因子はさらに grade 変数にも説明されているので，そのパス係数も確認できます。

SEM の基本として，パスが入った (影響を受けた) 変数は，影響が受けた部分と受けなかった部分に分割されます。影響を受けなかった部分が残りの分散として，`Variances:` のところに示されています。

visual 因子は説明変数になっていますが，パスが入ってきませんので，この分散は 1.00 です。図では visual 因子の楕円の上に 1 からの影響が入っていますが，これは分散が 1 だということの意味です。同じく

grade 変数も説明されない変数[*4]ですので，この分散が 1(標準化されています) になっているのです。

R の出力 13.2: 構造を入れたモデルの出力結果 (一部)

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual =~
    x1                1.000                               0.899    0.770
    x2                0.573    0.109    5.249    0.000    0.515    0.437
    x3                0.724    0.124    5.846    0.000    0.651    0.576
  textual =~
    x4                1.000                               0.978    0.849
    x5                1.109    0.067   16.646    0.000    1.085    0.851
    x6                0.924    0.056   16.349    0.000    0.903    0.834
  speed =~
    x7                1.000                               0.727    0.667
    x8                1.022    0.130    7.837    0.000    0.744    0.737
    x9                0.782    0.108    7.273    0.000    0.569    0.564

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  textual ~
    visual            0.453    0.094    4.841    0.000    0.416    0.416
    speed             0.226    0.093    2.435    0.015    0.168    0.168
  speed ~
    grade             0.645    0.105    6.147    0.000    0.887    0.443

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .x1                0.554    0.132    4.187    0.000    0.554    0.407
   .x2                1.121    0.103   10.841    0.000    1.121    0.809
   .x3                0.851    0.097    8.769    0.000    0.851    0.668
   .x4                0.370    0.048    7.706    0.000    0.370    0.279
   .x5                0.448    0.059    7.635    0.000    0.448    0.276
   .x6                0.358    0.043    8.268    0.000    0.358    0.305
   .x7                0.658    0.080    8.178    0.000    0.658    0.554
   .x8                0.466    0.072    6.478    0.000    0.466    0.457
   .x9                0.695    0.069   10.021    0.000    0.695    0.682
    visual            0.807    0.161    5.029    0.000    1.000    1.000
   .textual           0.764    0.096    7.918    0.000    0.798    0.798
   .speed             0.425    0.083    5.130    0.000    0.804    0.804
```

最後に出力 13.3 にある，**修正指数 (modification index)** の出力を見てみましょう。一行目にあるのは，visual 因子は変数 x9 へのパスをつけると，適合度がぐっと上がるよ，ということを意味しています。

[*4] 説明されない変数のことをとくに**外生変数 (exogenous variable)** といいます。これに対して説明されることがある変数は**内生変数 (endogenous variable)** といいます。

R の出力 13.3: 修正指数 (一部)

```
# A tibble: 64 x 8
   lhs     op    rhs        mi   epc sepc.lv sepc.all sepc.nox
   <chr>   <chr> <chr>   <dbl> <dbl>   <dbl>    <dbl>    <dbl>
 1 visual  =~    x9       41.9 0.442   0.397    0.394    0.394
 2 visual  ~     speed    25.3 0.507   0.411    0.411    0.411
 3 visual  ~     textual  25.3 2.24    2.44     2.44     2.44
 4 textual =~    x1       16.6 0.490   0.479    0.411    0.411
 5 visual  ~~    speed    13.6 0.184   0.314    0.314    0.314
 6 speed   ~     visual   13.6 0.228   0.282    0.282    0.282
 7 visual  ~     grade    12.8 0.445   0.495    0.247    0.495
 8 grade   ~     visual   12.8 0.137   0.123    0.247    0.247
 9 speed   =~    x1       11.1 0.313   0.227    0.195    0.195
10 x1      ~~    x9       11.0 0.169   0.169    0.272    0.272
```

実際に visual 因子が変数 x9 に影響している (x9 から因子が構成されている) というモデルを作り，適合度指標を比べてみましょう。表 13.2 に代表的な適合度指標の修正前の値と修正後の値を示しました。いずれの適合度指標においても，大きな改善が見られています。さて，これをどう考えたらいいでしょうか。

表 13.2　修正前後での指標の変化

index	before	after
CFI	0.895	0.945
TLI	0.856	0.923
AIC	7479.335	7433.224
BIC	7557.115	7514.707
RMSEA	0.100	0.073
GFI	0.919	0.944
AGFI	0.866	0.905

13.2　実践上の注意点

ここまでで，さまざまなモデルを表現できること，それを統一的に評価できることが理解できたかと思います。とくに，統一的に評価できることでさまざまな**モデル比較**も簡単になり，さらに適合度が良いモデルにするためにはどうすれば良いかについて，指標も出てくることがわかりました。

分散共分散行列の隅々まで記述できる大きな力を手に入れたことは間違いないのですが，大事なのは，我々は構造方程式モデルを使う側であって，それに使われる側であってはならない，ということです。学会誌に掲載されるような論文で，構造方程式モデルを見ると，その適合度は CFI,TLI,AGFI が 0.9 を超えているような，とても適合度の良いモデルがほとんどです。適合度が悪いモデルだと，これはデータとモデルがあっていないということですから，モデルの改善を求められたり掲載されなくなったりするということがあるでしょう。しかし大事なのは，モデルを使って主張したい心理学的内容のほうであって，適合度が良いだけでの中身のないモデルではないはずなのです。

今回も機械的に visual 因子が x9 変数から構成されるようにすると，適合度が上がるという提案を受けて

実施してみたところ，確かに適合度は上昇しました。ところでこの x9 変数というのは直線や曲線のスピードを認識するテストであって，(静的な) 空間認知能力とは違うものではないでしょうか？ そもそもそういうつもりで作ったものではないのに，機械に指摘されて「ひょっとしたらそういう側面があったのかも。そうだ，最初からそう思っていたに違いない」というように，自分を無理やり納得させてはいないでしょうか？

心理学の場合は変数の多くが関係しあっていますから，「移動する直線を見るというのは空間認知とも考えられるのだ。少なくともデータはそう示している」という理屈が成り立つかもしれません。しかし結果を見てから考え方を変えるのは，適切な研究実践法ではないでしょう。これらの問題は結果の再現性が担保されないという心理学の危機の引き金になりました[*5]。自分に都合の良い結果やモデルを出すことが目的ではなく，心がどのような状態になっているのかについての理論的積み重ねや発展こそ，目的であったことを忘れてはいけません。

とくに構造方程式まで使えるようになると，心理学的実体同士の関係を描写し，モデル化できるということが魅力的に映るかもしれません。心理学者は，これこそがやりたかったことなのかもしれないですね。しかしデータを超えての解釈はご法度ですし，何より構成された潜在変数が心理的な実在であるかどうかは，改めて考えなければならないのです。これらはあくまでも分散共分散行列から算出されるでしかなく，「文章読解力」「空間把握力」といった次元に貼り付けた自分たち自身の命名法に酔って，思考停止するようなことがあってはなりません。

この潜在変数同士の影響力が心理学的にどういう意味があるのか，をしっかり考えてから，モデルでの検証に進まなければならないことに注意して使ってください。

13.3 そのほかの統計パッケージ

構造方程式モデリングの利点の 1 つは，モデルを可視化したことにあります。皆さんもモデルの図を見た方が，出力結果を見るよりも理解が進んだ気がするでしょう。

こうした構造方程式モデリングを実行するソフトウェアは，R の専売特許ではありません。たとえば **Amos** という IBM 社が出しているソフトウェアでは，マウスをクリックしながら統計モデルを作り分析していくことができます。モデルの構成から GUI でできるのは大変な利点です。

また，構造方程式モデリングはさまざまな分析手法の統合的ツールです。言い換えるとかなり複雑なモデルであっても，ゴリゴリ計算し分析してくれます。現在考えられているさまざまなモデルの，ほぼすべてについて計算できるソフトウェアが **Mplus** です。このソフトウェアで分析できない SEM はない，と言っていいほどその用途は広く，また計算スピードも爆速です。カテゴリカルな変数にももちろん対応していますから，IRT/GRM のような出力もできます。

R の利点は商用ソフトと違ってフリーで手に入るところですが，有用・有償・商用パッケージでも良いのであれば，これらのソフトも活用することを考えてもいいでしょう。また R でも `lavaan` パッケージの他に，`sem` パッケージというのもあります。ツールは色々あって，ユーザがそれを選べるようにことが理想的ですから，皆さんも興味があれば色々試してみましょう！

13.4 課題

今日の授業でおこなったすべての次の計算をする R コードを提出してください。ファイル形式は R スクリプトか Rmd とします。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱い

[*5] さきほどの良い結果しか雑誌に掲載されない問題のことを，**出版バイアス (publication bias)** の問題といい，今も問題視されています。

になります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

第 14 章

双対尺度法

さて前回 SEM を学んだことで，分散共分散行列をとことんまで分析し尽くす方法が手に入ったことになります。SEM は統合的な表現方法ですから，観測変数でも潜在変数でもいいですし，**順序尺度水準**以上の大小関係が表現できる数値データであれば，あらゆる表現ができるわけです。

ではこれで多変量解析はすべて理解したことになるのでしょうか。いえ，もちろんそうではありません。分散共分散行列の分解はできるようになりましたが，変数間関係の表現は分散共分散行列だけではありません。ここまで，データの持つ情報は分散がすべて，変数間関係は共分散で表されるから分散共分散行列がすべてだ，と言わんばかりに話をしてきましたが，その枠を外すとどうなるでしょうか。

14.1 直線的ではない関係

ご存知の通り，分散共分散行列を標準化した行列は相関行列といいますが，相関係数が 1.0(あるいは -1.0) の状態を考えれば明らかなように，分散共分散行列 (や相関行列) で表現されるのは変数の直線的な関係性に限った話です。心理学の場合は中庸が良いようなシーンも少なくありません。たとえば血圧と健康度の関係で言えば，低血圧でも高血圧でも不健康であり，ちょうどいい血圧が一番健康的だというのはすぐにわかります。このように中庸が良い場合は，散布図が U 字型に現れてきます。散布図が U 字型であれば，相関係数としては 0 近くになりますが，血圧と健康が無関係だとは誰も言えないでしょう。

ごくシンプルかつ具体的な例をあげてみましょう。血圧と頭痛の頻度について調査したとします。血圧は高い，普通，低いの 3 段階。頭痛の頻度は「ない」、「たまに」、「ときどき」，「いつも」の 4 段階です。表 14.1 のような結果を見ると，この集計表に直線的な関係は確かになさそうですね。でも関係ないわけではない，と思いませんか。

表 14.1　血圧と頭痛

血圧	ない	たまに	ときどき	いつも
高い	0	0	3	2
普通	5	0	0	0
低い	0	2	3	1

このような場合は、「血圧が高い人か低い人は、頭痛がある」という傾向が見て取れるはずです。しかしこのデータ，機械的に血圧が高いを 1，普通を 2，低いを 3 とし，同様に頭痛もない〜いつもを 1,2,3,4 として相関係数を計算すると，$r = 0.165$ になります。相関関係では傾向をうまく読み取れていないのです[*1]。

[*1] 相関係数でお話ししましたが，分散共分散行列でも同じです。共分散で表現すると単位のせいで関係が直接的には理解できま

その根本的な原因は，もちろんスコアリングの方法にあります。共分散を計算するには間隔尺度水準以上の情報が必要で，今回のような 3,4 件法では相関係数を計算して良い数字ではありません。ではポリコリック相関係数のように順序尺度水準の相関係数なら良いか，といいたいところですが，それでもまだ不十分です。なぜなら，血圧が高い方から低い方まで，順番が保存されてしまっているからです。

関係を導き出すには抜本的な改革が必要です。たとえば表 14.2 のようにすればどうでしょうか。こうすると左下から右上にかけての直線的な関係がまだ見えてきます。これは血圧の順番を「高い」「低い」「普通」に並べ替えたものです。また，頭痛の順番も「いつも」と「ときどき」をひっくり返しました。順番を並び替えてしまっ

表 14.2　並び替えられた「血圧と頭痛」

血圧	ない	たまに	いつも	ときどき
高い	0	0	2	3
低い	0	2	1	3
普通	5	0	0	0

たというのは，尺度水準的には数字を無視して「高い」「低い」「普通」というカテゴリーとして扱ったということになります。つまり**名義尺度水準**レベルにまで落として考えたのです。

このように並べ替えて，血圧のスコアを高い $\rightarrow 3$, 低い $\rightarrow 2$, 普通 $\rightarrow 1$ とし，また頭痛の頻度をいつも $\rightarrow 3$, ときどき $\rightarrow 4$ と付け替えて相関係数を計算すると，今度は $r = 0.810$ になりました。こちらの方が線形性は高いことが数字でも確認できました。

ここで行ったのはこのように，すべての反応カテゴリをただの言葉だと考えて，付与された数字を無視して並べ替えたことになります。そうすることで，左下から右上にかけて線形性を高めることができました。しかし「高い」と「低い」,「低い」と「普通」の配置も等間隔である必要はなく，もっと直線性がはっきりするように配置してやってもよいのでは，というアイデアが浮かびます。

ここで考え方が反転したことに注意してください。一般的なリッカート尺度では，反応カテゴリに (シグマ法などで) 数字をつけて，変数間の線形性を算出して意味を考えるのでした。ここでは逆に，変数間の線形性を最大にするように反応カテゴリに数字をつけてやろう，という考え方です。データの直線性というのはデータの特徴を最も強調し解釈しやすい形です。そのようにデータを整えるためには，反応カテゴリにどういう数字を付与すれば良いか，と考えるのです。この方法を**数量化の理論 (Quantification Methods)** といいます。

反応カテゴリに数字を与えるとき，名義尺度水準にまで落として，すなわち数字の意味を無くして並べ替えるような作業をします。もちろん分析対象が，初めから名義尺度水準であっても構いません。たとえば出身県を調査したようなデータがあったとして，他の変数との線形関係を最大にするように並べ替え，数字を付与することもできます。数量化の考え方は，名義尺度水準の変数に解釈しやすい数値を与えることとも言えるのです。

せん。

14.2 林の数量化理論

数量化の研究をしたのは，日本の偉大な統計学者，林知己夫 (はやし ちきお)[*2]という人です。その名を冠して林の数量化理論と呼ばれることがあります。

林はさまざまな研究成果を挙げており，論文ごとにデータに必要な分析方法を開発するというようなスタイルでした。その膨大な研究業績を，弟子である飽戸弘[*3]が分類して，同じような分析方法ごとに I 類，II 類，III 類，IV 類，と呼んでいきました。

表 14.3 林の数量化

手法	外的基準	データ	目的	関連する手法
数量化 I 類	量的変数	質的変数	外的基準の予測	重回帰分析
数量化 II 類	質的変数	質的変数	外的基準の判別	判別分析
数量化Ⅲ類	なし	質的変数	変数間の関係の要約と記述	正準相関分析
数量化Ⅳ類	なし	対象間の類似度	対象間の関係の要約と記述	多次元尺度法

表 14.3 にあるように，基本的には質的変数，すなわち名義尺度水準や**順序尺度水準**のデータが得られたときに，それを解釈するためにはどのような数値を割り振ってやれば良いか，という発想から生まれたものになっています。

さきほどの表 14.1 のようなデータの並べ替えについては，数量化でいうところの III 類に該当します。ただこの数量化 III 類はおもしろいもので，同時期に独立にこの手法がフランス，カナダでも発展しており，2 つの別名を持っています。フランス学派が開発した手法は**対応分析 (correspondence analysis)** といい，カナダ在住の日本人，西里静彦[*4]が開発した手法は**双対尺度法 (Dual Scaling)** といいます。どの分析も，基本的にはカテゴリカルな変数について直線性を最大にする値を割り振ることを目的にします。

カテゴリカル変数なので，分析の応用範囲は多岐にわたります。どのような変数でも名義尺度水準に落とすことができるからです。表 14.2 には一般的なデータセットの形を示しました。ID があって，Q1,Q2... と項目が列方向に並ぶ形です。一行が一人の反応を表しています。Q1 がたとえば性別などの名義尺度水準の変数であっても，男性 → 1，女性 → 2 のようにコード化するルールを決めて入力します。Q2 はたとえばリッカート法で当てはまる，やや当てはまる･･･などの 5 件法だったとしましょう。その場合はシグマ法によるスコアを入れる，あるいは簡便的に当てはまるを 5，やや当てはまるを 4，といったように数字を割り振って入力しますね。

これをカテゴリカル変数と見なしてデータセットにした例が表 14.2 です。ID は分析対象ではありませんから横に置くとして，Q1 が 2 列に増えています。ID=1 の人は男性なのですが，この人は Q1:Male=1 かつ Q1:Female=0 というようにコード化されています。同様に，Q2 には「どちらとも言えない」を選択しているのですが，これを 3 とするのではなく 00100 と 5 列にわたってコード化しているのです。このようにすることで，

[*2] 林 知己夫 (はやし ちきお、1918 年 6 月 7 日 - 2002 年 8 月 6 日) は、日本の統計学者。正四位勲二等、理学博士。統計数理研究所第 7 代所長。社会調査･世論調査におけるサンプリング方法の確立を始め、数量化理論 (Hayashi's Quantification Methods) の開発とその応用で知られる。1990 年代以降、データの科学 (Data Science) を提唱し、その研究･思想は現在へと引き継がれている。Wikipedia より。

[*3] 飽戸 弘 (あくと ひろし、1935 年 3 月 14 日 -) は、日本の社会学者、東京大学名誉教授。専門は、社会心理学、コミュニケーション論。Wikipedia より。

[*4] 西里 静彦 (にしさと しずひこ、1935 年 6 月 9 日 -) は、カナダの行動計量学者（計量心理学）。学位は Ph.D.（ノースカロライナ大学･ 1966 年）。トロント大学名誉教授、アメリカ統計学会フェロー、日本行動計量学会名誉会員。北海道十勝郡浦幌町出身 (札幌市生まれ)。Wikipedia より。

すべての変数をカテゴリーとして扱うことができるようになります。

表 14.4 一般的なデータセット

ID	Q1	Q2	···
1	1	3	···
2	2	4	···
⋮	⋮	⋮	

表 14.5 カテゴリカル化したデータセット

ID	Q1:M	Q1:F	Q2:5	Q2:4	Q2:3	···
1	1	0	0	0	1	···
2	0	1	0	1	0	···
⋮	⋮	⋮				

また表 14.1 を並べ替えた時のように，こうした名義尺度のデータの線形性が最大になるように，行だけでなく，列も並べ替えます。データの中で線形性が最大になるように，反応カテゴリ**と**回答者に数字を割り振るのです。ちなみに表 14.1 は集計されたデータであり，今回の表 14.2 は集計前のデータです。カテゴリカルな変数の分析の場合は，どちらでも良いのです。行と列の関係が表されている数字であれば，「ある/ない」のような二値反応でも，集計された度数でも構いません。さらにいえば，行と列の関係が記述されていればなんでもいいのです。

これまでの回帰分析，因子分析から SEM に至るまでの流れは，変数間関係だけを考えてきました。N 行 M 列のデータ行列の計算の途中で，行に関する情報は平均化して潰されてしまい，最終的には $M \times M$ サイズの正方行列だけを扱うことになったのでした。そして M 個の変数に**因子負荷量**などの重みをつけて考察してきました。今回のカテゴリカルなデータは，$N \times M$ 行の矩形行列をそのまま分析し，行と列の両方に重みをつけます。正方行列でも矩形行列でも，変数と変数，変数と回答者がどのように関係しているかというところを見るという意味では同じです。SEM では分散共分散行列や相関行列が，双対尺度法ではクロス集計表や素データがその分析対象になります。数学的な面から説明すると，正方行列の場合は**固有値分解**をして，**固有値**と**固有ベクトル**を求め，固有ベクトルが変数の重みになるのでした。矩形行列の場合は**特異値分解 (Singular value decomposition)** と呼ばれ，行および列に対応する**特異ベクトル**をその重み，座標と考えることになります。本質的には同じようなものだと思っていただければと思います。

14.3 双対尺度法による分析

それでは実際の分析例を見てみましょう。次のコード code:14.1 は，MASS パッケージに含まれるサンプルデータ caith を対応分析で分析するものです。

code : 14.1 双対尺度法による分析

```
1 library(MASS)
2 caith
3 result <- corresp(caith,nf=min(nrow(caith),ncol(caith)-1))
4 result
5 plot(result)
```

■コード解説

1 行目　パッケージ MASS のよみこみ

2 行目　サンプルデータ caith を表示させる

3 行目　対応分析関数 corresp で分析した結果を result オブジェクトに入れる

4 行目　結果の出力

5 行目　結果のプロット

データ `caith` はスコットランドの Caithness 地方に住んでいた人に対してなされた調査で，髪の毛の色と目の色の関係を集計したものです。列方向に髪の毛の色，行方向に目の色が入っていますね (表 14.6)。

表 14.6　データ `caith` の中身

	fair	red	medium	dark	black
blue	326	38	241	110	3
light	688	116	584	188	4
medium	343	84	909	412	26
dark	98	48	403	681	85

この行・列のカテゴリにはとくに順序性などありませんが，分析することで最も線形性の高いウェイトをつけることができます。もちろん 1 次元で表現できない可能性があり，その場合は第二，第三と次元数を増やして行きます。データの行数あるいは列数の小さい方マイナス 1 次元まで求めることができます。それを表現しているのが，関数 `corresp` のオプション `nf` で，行数 `nrow(caith)`，列数 `ncol(caith)` の小さい方 (`min` 関数) マイナス 1 を指定しています。

結果は出力 14.1 のようになります。行および列にスコアがついていますね。この値を尺度値として使うと，データの相関係数が最大になるというような指標化がなされたのです。

R の出力 14.1: 対応分析の結果

```
First canonical correlation(s): 4.463684e-01 1.734554e-01 2.931691e-02 1.134031e-
16

 Row scores:
              [,1]       [,2]       [,3] [,4]
blue   -0.89679252  0.9536227  2.1884132    1
light  -0.98731818  0.5100045 -1.0837859    1
medium  0.07530627 -1.4124778  0.1894089    1
dark    1.57434710  0.7720361 -0.1482208    1

 Column scores:
              [,1]       [,2]       [,3]       [,4]
fair   -1.21871379  1.0022432  0.4271282 -0.8692696
red    -0.52257500  0.2783364 -4.0268545 -1.3400421
medium -0.09414671 -1.2009094  0.1103959 -0.8453208
dark    1.31888486  0.5992920  0.3450676 -1.2251588
black   2.45176017  1.6513565 -1.5736976  1.1609621
```

結果はプロットされたものを見た方がわかりやすいかもしれません。図 14.1 にプロットを示しました。たとえば横軸，dim1 にそって目の色を見ていくと，light – blue – medium – dark の順に並べた方が良い，ということがわかります。とくに light と blue は近いのであまり大きな意味的違いはないことがわかります。同様に髪の毛の色は，fair – red –medium –dark –black の順に並べられることになります。具体的な数字はそれぞれ出力の一列目の通りです。この dim1 だけでは表現できない違いが，dim2 で表されており，それが図では上下の広がりとして示されていますね。

この並びが，新しく構成された次元だと考えることができます。そしてこれを見ると，たとえば髪の毛と目の

色がどちらも dark である場合，この二点は近くにプロットされています。この二点が近くにあるということは，両者の結びつきが強いということです。髪の毛が暗い人は目の色も暗い人が多い，といった関係の強さが見て取れます。

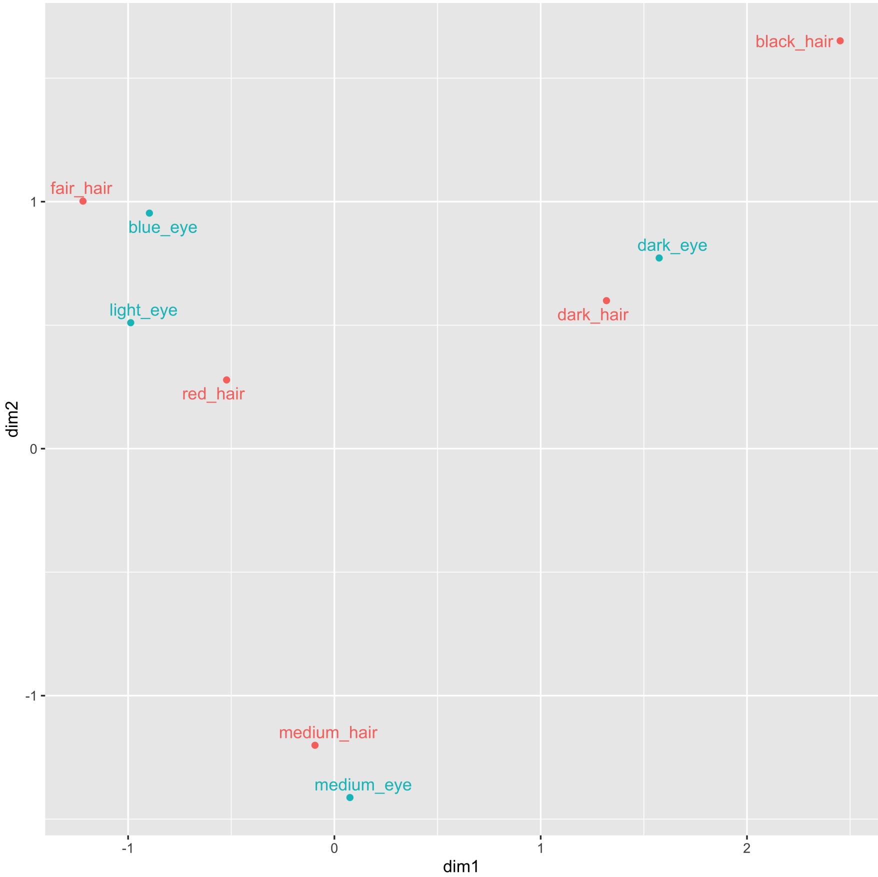


図 14.1　分析結果のプロット

ちなみにこれは対応分析独特の表示法で，行と列の変数が 1 つの画面にうつされていますね。厳密には，行ベクトルが作る次元，列ベクトルが作る次元は別物です。ですから，列の要素を行の空間に写像するためには変換が必要で，逆もまた然り，なのです。双対尺度法の場合はこの違いを厳格に捉えます。対応分析の場合は，相互に写像しあった座標を一枚の図にすることで情報を圧縮しています。数学的には同じモデルであっても，表現の仕方や表示の仕方に，モデル作成者の考え方が反映されているとも言えるでしょう。

14.4　テキストマイニングへの応用

さて，本講で紹介した分析方法は，名義尺度水準を対象にしていて，かつ外的な基準がなく内的な構造を明らかにしようとするものだということがわかってきました。いわば名義尺度水準における因子分析ですね。名義尺度水準を対象にしたということは，およそ言語化できたものはすべて分析の対象にできる，とも言えます。最も低次元で一対一対応した言葉の世界の数字だからです。

応用例として，**テキストマイニング (Text Mining)** を紹介しておきましょう。別名自然言語処理とも言

われますが，これはテキスト，すなわち普段の「言葉」に潜む関係を分析する手法です[*5]。文章を対象にしますから，小説や新聞記事などはもちろん，日記や逐語録なども分析の対象にしてしまおうというものです。心理学の分野では面接法などにおいて，クライアントがどのように語るかをすべて記録することがありますが，これをみて「うーん，こういうことを考えているんだな」と読み取るのは主観的な判断によるものです。ここにテキストマイニングを用いれば，機械的にどう言った言葉の使われ方をしているかを分析できます。あるいはツイッターやフェイスブックなどの SNS でどのような言葉，記事が流行しているかと言ったことを分析する，というような使い方もできます。分析対象が一気に広がりますね。

このテキストマイニングがやっていることは二段回に分かれ，第一段階が**形態素解析 (morphological analysis)**，第二段回が多変量解析です。第一段階は，「今日はいい天気ですね」といった平文を「きょう」「は」「いい」「てんき」「です」「ね」といった要素に分解することを指します。英語のような分かち書きがされる言語であればこれは簡単なのですが，日本語の場合は分かち書きされていないことに加え，漢字，ひらがな，カタカナなど表記方法もさまざまですから，この分析をするための特別な解析エンジンが必要です。幸い，すでに開発されているものがフリーソフトウェアとして利用できます。時間がかかりますがこれらを使うと，品詞ごとに分解し，その原形 (活用する前の形) や活用形は何か，と言ったことを一覧にしてくれます。

自然言語ですので大量の分割がなされますが，それを言葉同士の関係を表す行列の形で表現します。使われている品詞の原形ごとに誰が何回発言したかとか，各要素が 1 回の文章の中で同日に使われた回数をカウントするなどして，言葉と人，言葉と言葉の関係をデータ化するのです。データ化できればあとはこっちのものです。数量化できるのですから，その言葉群のなかでどの言葉とどの言葉が近いのか，他のどの変数と関係するのか，と言った分析をできます。

テキストマイニングについては R でもできますし，専門的なソフトウェアがあります。詳しくは樋口 (2020) を参照してください。

14.5 課題

今日の授業でおこなったすべての次の計算をする R コードを提出してください。ファイル形式は R スクリプトか Rmd とします。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

[*5] マイニングとは鉱脈を掘る，という意味です。

第 15 章

多次元尺度構成法

今回は**多次元尺度構成法 (Multi-Dimensional Sacling; MDS)** について解説します。ここまで分散共分散行列を分解するところから，クロス表のような**名義尺度水準 (の)** データを分解するところまでやってきました。

多次元尺度構成法は，分散共分散行列を扱う線形モデルよりはやや仮定が緩く，また数量化のように解析者が数字を与えることを目的にするというよりは，その名の通り尺度を作ろうとしているという意味で心理学的・心理測定的なモデルだといえるでしょう。

私たちが単位もない不確かなものを対象にしながらもそれを測定するモノサシをつくることができるのは，2 つの対象にたいしてその比較をして一方が他方よりも大である，ということが言えるからでしょう。記号を使って言えば，x と y を比べて $x \succeq y$ である，ということから，尺度上の値 $p \geq q$ を対応させるということが，尺度を作るということです[*1]。このとき比較する 2 つが重さや広さのような物理的なものであればわかりやすいですし，「痛み」や「喜び」といった心理的な要素であっても構いません。あるいは「より賛成」といった態度表明のようなものでも良いかもしれません。この比較から出てくる関係は，分散共分散や相関係数のように強い線形の過程を置かなくても，より緩やかに**距離 (distance)** という考え方で表現されます[*2]。

MDS は距離行列を固有値分解するモデルです。それではこの方法について内容を見ていきましょう。

15.1　多次元尺度構成法

R にはサンプルデータとして `eurodist` というヨーロッパ各地の都市間距離のデータが用意されています。表 15.1 にその一部を示します。

元のデータは 21×21 のサイズです。表から，たとえばアテネ (Athens) とバルセロナ (Barcelona) の距離が 3313km，と言うことが読み取れます。表の右上が空白になっていますが，これは**距離行列**が**対称行列**なので，ムダな情報を表示しないようにしているからです。アテネとバルセロナの距離は，バルセロナとアテネの距離に等しいですからね。

さて距離行列はこのように，**正方行列**ですから，**固有値分解**をすることで**基底**を求めることができます。基底は座標を形成する基本単位ですから，それを使って各変数の位置にあたる座標を計算できます[*3]。このように距離行列から座標を求めて，変数をプロットすることで変数間関係を可視化する手法のことを**多次元尺**

*1 経済学では学問の最初の段階で，財を源とした集合に対する二項関係 $x \circ y$ を定義し，それを効用関数 u で実数領域に写像して $u(x) \geq u(y)$ を考える，といった基本的な原理を抑えます。心理学ではなぜか，「集合」「元」「二項関係」という基本的な比較の要素について考えることなく，その分析ツールだけがどんどんと発展してきています。

*2 後述しますが，相関係数も距離の一種と考えられます。

*3 厳密には距離行列 $\boldsymbol{D}$ そのものではなく，それを二重中心化した行列の固有値分解になりますが，本質的には変わりません。詳しくは岡太・今泉 (1994) などを参照のこと。

表 15.1 ヨーロッパ都市間距離データの一部

	Athens	Barcelona	Brussels	Calais	Cherbourg	Cologne
Athens	0					
Barcelona	3313	0				
Brussels	2963	1318	0			
Calais	3175	1326	204	0		
Cherbourg	3339	1294	583	460	0	
Cologne	2762	1498	206	409	785	0

度構成法 (Multi-Dimensional Scaling: MDS) と言います。

試しに eurodist データを MDS で 2 次元プロットしてみましょう。これを実行するコードは簡単で，eurodist データのようにデフォルトで組み込まれている cmdscale 関数を使います。

code : 15.1 計量 MDS の実践と描画

```
1  library(ggrepel)
2  # MDSを実行
3  result.MDS1 <- cmdscale(eurodist,k=3)
4  # y軸反転させつつ描画
5  g <- result.MDS1 %>%
6    as.data.frame() %>%
7    dplyr::mutate(label = rownames(.)) %>%
8    ggplot(aes(x = V1, y = V2, label = label)) +
9    geom_point() +
10   geom_text_repel() +
11   xlim(-2500, 2500) +
12   ylim(2500, -2500) +
13   xlab("dim␣1") +
14   ylab("dim2")
```

■コード解説

1 行目 ggplot2 でラベルをプロットするときに，綺麗な配置にしてくれる ggrepel パッケージを使います。このコードを実行するときに持っていない人は，インストールしておいてください。

3 行目 cmdscale 関数に eurodist データを与えています。k=3 は 3 次元解を出すように指定しています。地球は球体ですから，地球上の地理は 3 次元で表現できますよね。

5-14 行目 ggplot2 による描画です。結果オブジェクトである result.MDS をデータフレームにし，行の名前になっていた都市名を変数として格納したのち，散布図のようにプロットしています。

図 15.1 をみると，上の方にストックホルム (Stockholm) があって，右下にアテネが，中央にパリ (Paris) が･･･といった配置になっています。地球上の位置と完全に一致しているとは言えませんが，それでも概ねうまくプロットできていますね[*4]。このように，距離関係だけから地図を作ることができるというのが，MDS という手法なのです。

[*4] 完全に一致しない理由は，地球が球面であるのに対し平面にプロットしたから，と言うのもあります。

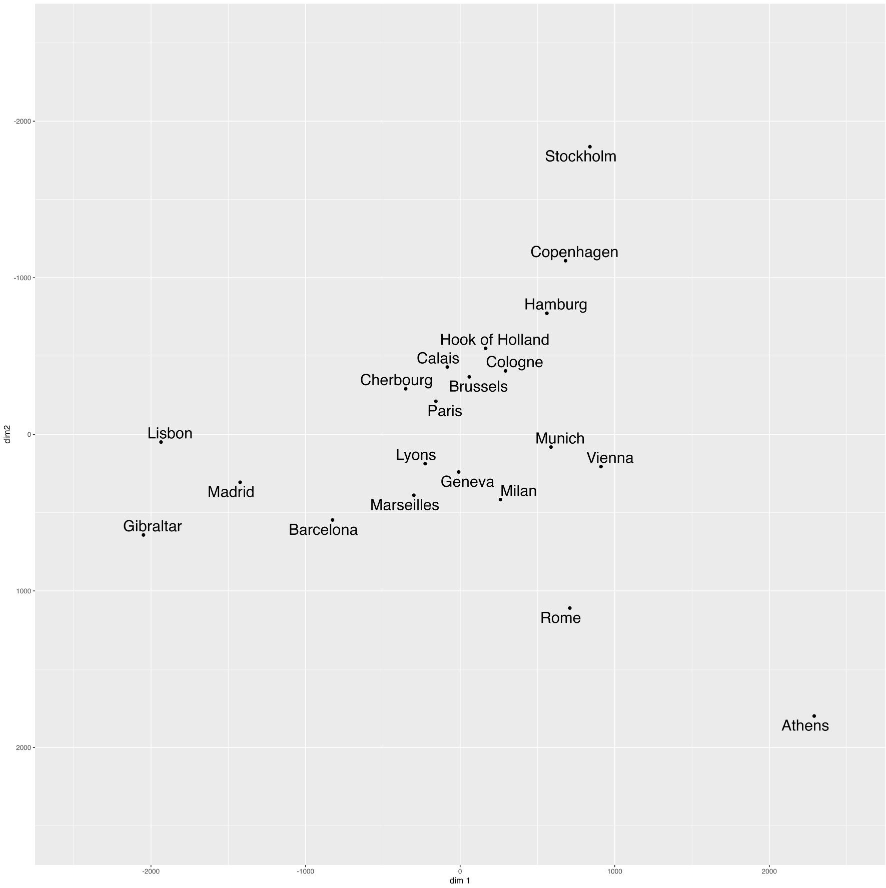


図 15.1 MDS のプロット

15.2 距離と心理学のデータ

MDS は距離関係だけから地図を作る方法でした。因子分析は相関関係から次元を作る方法でした。この 2 つの手法はとても似ています。というのも，相関関係と言うのが変数と変数の近さ･遠さ，つまり距離を表しているとも言えるからです。あらためて**距離 (distance)** とは何かを考えてみましょう。2 点 x, y の距離を $d(x, y)$ とすると，距離とよばれる数字の条件は次のようになります。

非負性 $d(x, y) \geq 0$
非退化性 $d(x, y) = 0 \Leftrightarrow x = y$
対称性 $d(x, y) = d(y, x)$
三角不等式 $d(x, z) + d(z, y) \geq d(x, y)$

もっとも一般的に使われるのはユークリッド距離で，2 次元座標 $(x_1, y_1), (x_2, y_2)$ があった時のユークリッド距離は次のように計算します。

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

3 次元座標 $(x_1, y_1, z_1), (x_2, y_2, z_2)$ の場合は項を増やすだけです。

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

このようにして計算される距離ですが，上の条件を考えると何も二乗してルートを取らなくても絶対値を足し合わせるような方法でも構いません。たとえば 2 次元座標の場合は，次のような計算でもいいのです。

$$|x_1 - x_2| + |y_1 - y_2|$$

現にこのような距離のことを**マンハッタン距離 (Manhattan distance)** といいます。二乗ではなく n 乗して n 乗根をとる，という形で一般化することもできます。

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - x_j|^p \right)^{\frac{1}{p}}$$

これは**ミンコフスキー距離 (Minkowski distance)** という名前がついています。他にもマキシマム距離，バイナリ距離，チェビシェフの距離，キャンベラ距離などがあり，いずれも R の dist 関数のオプションで選ぶことができますので，ヘルプなどを参照してみてください。

また，相関係数 r_{ij} は $-1 \leq r_{ij} \leq +1$ の範囲にありますが，この絶対値から $1 - |r_{ij}|$ とするとこれも距離の条件に当てはまります。どの程度類似しているかというのも，距離と考えることができるのです。

ここまでは数学的な距離のバリエーションでしたが，次に心理学的な意味に目を向けてみましょう。距離とは類似度でもありますから，何を距離と見なすかによって，さまざまな心理学的刺激がデータを形作ることになります。

たとえば評定尺度で，n 個の項目で何らかの評価をしてもらったとします。x_{ij} を i 番目の項目における対象 j の評定値だとすると，$d(j,k) = \sqrt{\sum_{i=1}^{N}(x_{ij} - x_{ik})^2}$ とすれば対象の類似度が計算できます。ほかにも何らかの刺激 A,B について，A か B かの判断をさせた時に混同してしまった混同率や，単語と単語の連想価，刺激の汎化勾配，反応潜時，ソシオメトリックなデータなど，いろいろなものが「類似しているかどうか」の指標として使えます[*5]。尺度評定よりも，具体的な 2 つの刺激が似ているかどうかの反応の方がやりやすい，というのは誰しも実感としてわかることではないかと思います。

類似度のデータが得られれば，それを距離と見做して MDS にかければ，変数間の関係を地図に描くことができるわけです。このように応用可能な領域が非常に広いことも，MDS の利点であると言えるでしょう。

15.3　非計量多次元尺度法

さて心理学的なさまざまな刺激が，MDS によって可視化できるということがわかってきました。距離行列が構成できれば，後の分析は何とでもできるわけです。ただし，この場合の距離行列とは，間隔尺度水準以上であることが必要です。しかし心理学的な刺激に対する反応をデータ化する時は，同時に被験者はそこまで鋭敏に反応しているのか，いいかえれば本当に間隔尺度水準以上の精度で判断できているのかな，という人間側の問題が気になりますね。そこまで人間は鋭敏無反応をしていないかもしれません。

でも大丈夫。MDS は仮定を緩めたモデルがあります。**非計量的多次元尺度構成法 (Non-Metric Multi-Dimensional Scaling)** と呼ばれる手法がそれです。非計量 MDS では，対象 j と k の類似度を δ_{jk} とし，分析によって埋め込む多次元空間での距離を d_{jk} とすると，

$$\delta_{jk} > \delta_{lm} \text{ならば} d_{jk} \leq d_{lm}$$

のように，イコールではなく順序関係だけ保持して座標を求めます。この手法では，元データが順序尺度水準程度の情報しか持っていなくても地図を描くことができるのです。人間の判断はせいぜいが順序尺度ぐら

*5 これらデータの例に関しては高根 (1980) の Pp.14–27 を参照してください。

いですから，「より類似している ($\delta_{jk} > \delta_{lm}$)」のであれば「より近くにある ($d_{jk} \leq d_{lm}$)」というぐらいの配置の方が良いかもしれません。

表 15.1 に示したような物理的距離であれば，間隔尺度水準の情報であることは間違いありません。その場合には，最初に示した固有値分解による手法を使います。これは非計量 MDS に対して，**計量的多次元尺度構成法 (Metric Multi-Dimensional Scaling)** と呼ばれています。

15.3.1 非計量多次元尺度法の例

心理学ではデータの性質上，非計量 MDS のほうが便利なことが多いでしょう。計量 MDS でも非計量 MDS でも，R では簡単な関数で実行できます。ここでは `MASS` パッケージに含まれる `isoMDS` 関数を使って実践してみます。

使うデータは `M1score2021.csv` とします[*6]。ファイル名からお察しいただけるように，M-1 グランプリの評定をデータ化したものです[*7]。2021 年度のスコアは表 15.2 でした。

表 15.2 M-1 グランプリ 2021 の採点結果

演者	巨人	富澤	塙	志らく	礼二	松本	上沼
モグライダー	91	93	92	89	90	89	93
ランジャタイ	87	91	90	96	89	87	88
ゆにばーす	89	92	91	91	93	88	94
ハライチ	88	90	89	90	89	92	98
真空ジェシカ	90	89	92	94	94	90	89
オズワルド	94	95	95	96	96	96	93
ロングコートダディ	89	90	93	95	95	91	96
錦鯉	92	94	94	90	96	94	95
インディアンス	92	91	93	94	94	93	98
もも	91	90	91	96	95	92	90

M-1 の採点は審査員各自の主観に基づいて行われ，得点の絶対値はそれほど重要ではないかもしれません。すなわち，松本人志の 80 点が上沼恵美子の 80 点と同じぐらいの面白さを評価しているか，ということについては真偽判断ができないでしょう。それでも各審査員の中での相対的評価には，一貫性がありそうです。すなわちある審査員が漫才師 A に 80 点，漫才師 B に 85 点をつけたのなら B の方が面白かったということでしょうし，漫才師 C が 83 点なら $A < C < B$ という順序はあると思われます。つまり**順序尺度水準**程度の質はあると仮定することに無理はなさそうです。

そこでこのデータをもとに，10 組の漫才師の類似度を計算します。類似度はユークリッド距離を用いることにします。ユークリッド距離は既に述べたように差分の二乗を総和して平方根を取ったものですが，具体的な数字で見た方がわかりやすいかもしれませんので，表 15.3 を用意しました。表 15.3 にモグライダーとランジャタイの二組だけ取り出し，これで計算例を見てみます。それぞれの得点の差分，その二乗を計算し，それ

[*6] ファイルはシラバスのサイトからダウンロードできます。

[*7] 念のために解説しておきますが，M-1 グランプリとは 2001 年から始まった漫才の賞レースの 1 つで，年末に年間チャンピオンが決定します。開催年ごとにルールが少し変わることもありますが，基本的には予選を勝ち抜いた 10 組の漫才師が 4 分間のネタを披露し，6-7 名の審査員が 100 点満点で採点します。点数の上位 3 組が決勝戦を行い 2 本目のネタを披露，投票によりチャンピオンが選出されるという流れです。2021 年はオール巨人，富澤たけし，塙宣之，立川志らく，中川礼二，松本人志，上沼恵美子が審査員で，最終的には錦鯉がチャンピオンになりました。このデータセットは 1 本目のネタについての採点を 2001 年から集めたものになります。

表 15.3　距離の計算

	巨人	富澤	塙	志らく	礼二	松本	上沼	総和
モグライダー	91	93	92	89	90	89	93	637
ランジャタイ	87	91	90	96	89	87	88	628
差分	4	2	2	-7	1	2	5	9
差分の二乗	16	4	4	49	1	4	25	103

を総和したところ 103 という値になっています。これの平方根を取ったもの，すなわち $\sqrt{103} = 10.14889$ がこの二組の距離，すなわち非類似度ということになります。この計算を全ての組み合わせについて計算してくれるのが，dist 関数なのです。

code : 15.2　距離行列の計算

```
1  dat <- read_csv("M1score2021.csv")
2  dat.mat <- dat %>%
3    dplyr::filter(年代 == 21) %>%
4    arrange(ネタ順) %>%
5    dplyr::select(-年代, -ネタ順) %>%
6    pivot_longer(-演者) %>%
7    na.omit() %>%
8    pivot_wider(id_cols = 演者,
9                names_from = name,
10             values_from = value) %>%
11   as.matrix()
12 rownames(dat.mat) <- dat.mat[, 1]
13 dat.mat <- dat.mat[, -1] %>%
14   dist()
```

■**コード解説**

1 行目　データファイルを読み込み，dat オブジェクトに格納します。

2-9 行目　必要なデータだけに絞り込む操作です。流れを解説しますが，他のやり方でもいいですしできあがったものが何かだけわかれば結構です。

- **3 行目**　2021 年のデータだけに絞り込みます。
- **4 行目**　ネタ順に並び替えています。
- **5 行目**　年代変数とネタ順変数はもういらないので削除してしまっています。
- **6 行目**　ロング型に変換しています。これで演者-審査員-採点のデータセットができます。
- **7 行目**　欠損値を除外しています。実はこれがこの操作の目的で，というのも過去の審査データも入っているものですから，過去の審査員も大量に変数として含まれていて，それらが欠損値になってしまっていたのです。
- **8-10 行目**　元のワイド型に戻しています。
- **11 行目**　以下の行列処理のため，data.frame 型から matrix 型に変換しています。

12 行目　変数として一列目に演者名が入っていますが，これを行列の行名に入れています。matrix 型は行名･列名をデータの外に持つのです。

13-14 行目　変数としての演者名を除いて，パイプで dist 関数に入れ，距離行列を作っています。

できた距離行列は，表 15.4 のようになっています。モグライダーとランジャタイの距離が，先ほどの例で計算した値と一致していることを確認してください。またこれは**正方対称行列**ですから下三角だけ表示しておいます。また，対角が 0 になっています。自分自身との距離はゼロだからです。

表 15.4 演者の非類似度行列 (演者名は略記)

	モグ	ラン	ゆに	ハラ	真空	オズ	ロン	錦鯉	インデ	もも
モグ	0.000									
ラン	10.149	0.000								
ゆに	4.583	9.165	0.000							
ハラ	7.937	12.806	7.616	0.000						
真空	8.660	7.483	7.071	11.832	0.000					
オズ	12.490	15.652	12.207	14.933	11.000	0.000				
ロン	9.381	11.446	6.403	9.110	7.416	9.487	0.000			
錦鯉	8.485	15.264	8.307	10.909	10.247	7.071	7.874	0.000		
インデ	9.381	14.107	8.062	8.660	9.950	8.124	4.472	6.325	0.000	
もも	10.100	9.110	8.307	12.207	3.606	8.718	6.782	9.592	8.718	0.000

あとはこの距離行列を `isoMDS` 関数に渡すだけです。`isoMDS` 関数は引数として，何次元の解を求めるかを設定できます。地理データであれば 2,3 次元から作られていることは明らかですが，この評価が何次元かは事前にわかりません。何次元にするかの指標として，Kruskal (1964b) は `Stress` と呼ばれる値を次のように定義しました。

$$Stress = \sqrt{\frac{\sum(d_{ij} - \delta_{ij}^2)}{\sum d_{ij}^2}}$$

つまり実際のデータの距離 d_{ij} と，MDS で作られる空間上の座標から計算される距離 δ_{ij} の全体的な距離を当て嵌まりの指標と考えていることになります。これを目安に，次元数を 1 から 7 まで変化させながら，Stress 値がどうなるかを表したのが図 15.2 になります[*8]。

まるで因子分析の**スクリープロット**のようですね。横軸に次元数，縦軸に Stress 値を置いた折れ線グラフですが，当然のことながら反映させる MDS 空間の次元数が増えるとデータとの距離が縮んでいきます。Kruskal (1964a) の基準によれば，Stress 値は表 15.5 のように評価できます。この基準でいくと，今回は 2 次元で 4.9%(0.0049534) ですから，2 次元解で OK としましょう。

演者をプロットしたのが図 15.3 です。東西南北というか，上下左右の軸に特に意味はありませんから，因子分析のように因子軸の解釈や命名をすることはありません。近くにプロットされた対象は評価が似ていたんだな，ということがわかりますし，相対的な位置関係から，「ハライチとももは芸風が真逆だな」とか「ロングコートダディは中心に近いから中庸的な笑い，悪く言えばキャラが立ってないんだな」といったことが読み取れます。この図 15.1 や図 15.3 のように MDS で作られた座標のことを特に**布置 (configure)** といいます。特に非計量 MDS は優劣・大小関係という順序尺度水準の評定だけからでも，その空間的な特徴を描くことができますから，心理尺度のもつ仮定や測定モデルを考える必要がないため，今後その重要性が再発見されていくのではないかと思います。

[*8] どうして 7 までか，というと評定者が 7 名だからです。7 名それぞれの評価次元があると考えると，この距離空間の中には最大 7 次元あるはずですから。あるいは 10 組の演者がいますから，組み合わせの自由度から考えて 9 次元まで試してもいいと思います。

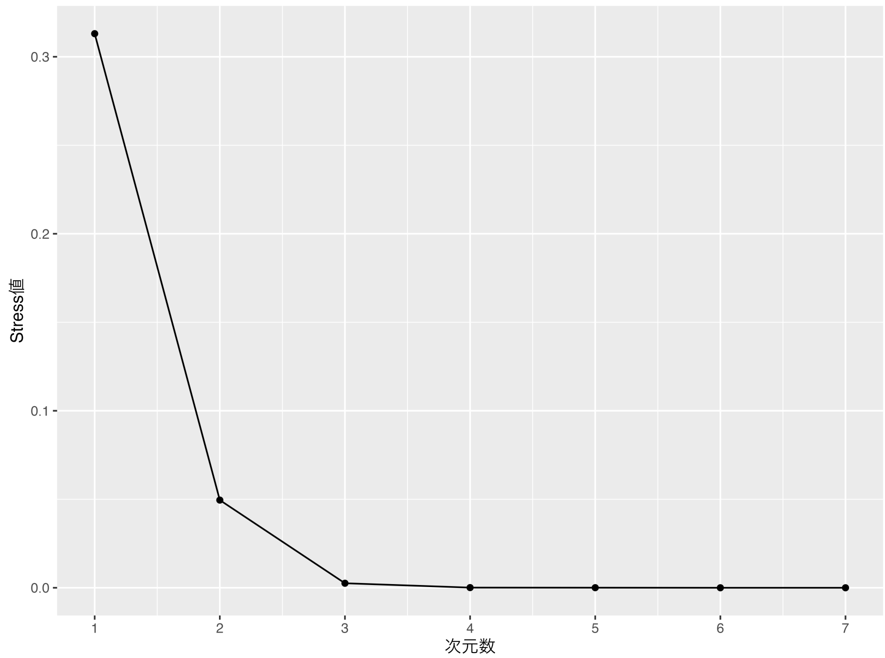


図 15.2 Stress 値の減衰

表 15.5 Stress 値の評価

Stress	Goodness of Fit
20%	poor
10%	fair
5%	good
$2\frac{1}{2}$%	excellent
0%	“perfect”

15.4 多次元尺度法の展開

多次元尺度構成法で作られた地図は，対象をプロットした地図です。地図には，その上に何か書き込んだり，地形の図に天気図を重ねるように複数の地図を重ねて表現したりできます。多次元尺度構成法にも，このような応用モデルがいろいろ考えられています。ここでいくつかの発展的な MDS モデルを見てみましょう。

■prefmap 類似度空間の上に，個々人の理想点を追加する方法です。個人の好み preference をマッピングする方法なので，**Preference Mapping()** と呼ばれています。個人 i が対象 j について，好みの程度を s_{ij} と評価したとします。対象 $1, 2, ...j..., M$ は別途類似度評定によって，MDS のつくった地図上にプロットされているとします。ここで個人 i と対象 j との地図上の距離 d_{ij}^2 に対して，次の回帰式を考えます。

$$s_{ij} = a_i d_{ij}^2 + b_i + e_{ij}$$

つまり，個人 i と対象 j の距離を使って，好みの程度 s_{ij} を予測するモデルを作り，誤差がもっとも小さくなる点に個人 i をプロットするのです。こうすることで，対象とそれを評価する人を一枚の地図に表現すると言

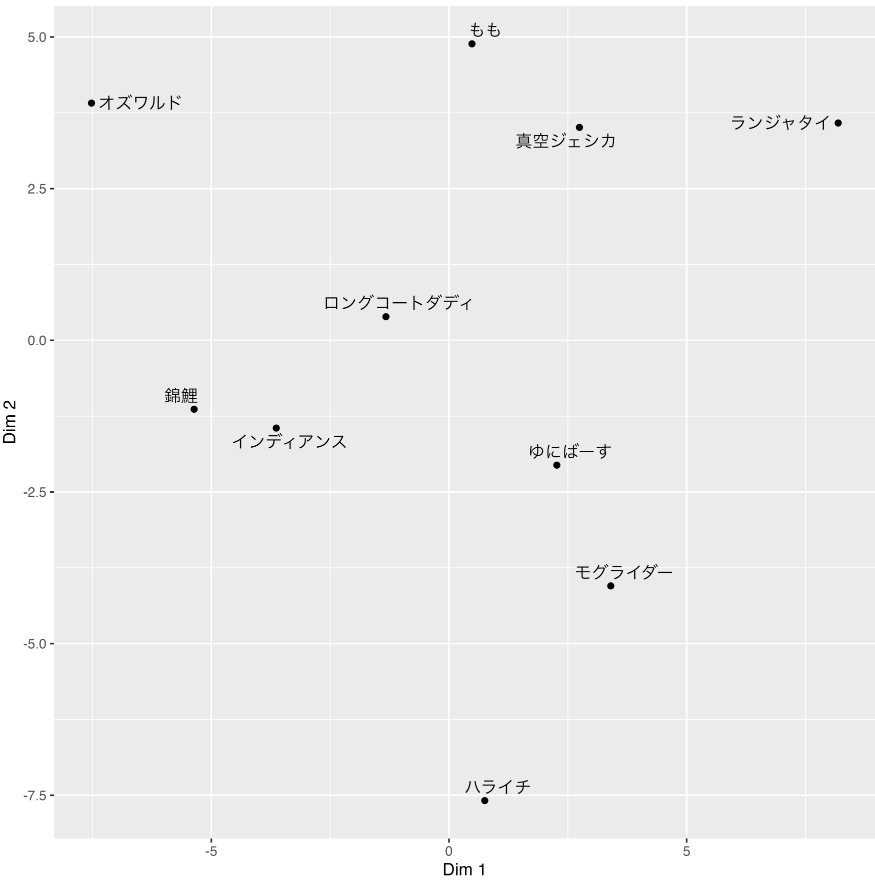


図 15.3 非計量 MDS のプロット

3254 うことができます。

例えば先ほどの M-1 の例ですが，私の採点では表 15.6 のようになりました[*9]。この評定値を使って著者

表 15.6 著者の評定

演者	採点
モグライダー	90
ランジャタイ	60
ゆにばーす	85
ハライチ	92
真空ジェシカ	83
オズワルド	89
ロングコートダディ	85
錦鯉	83
インディアンス	82
もも	88

*9 ランジャタイは何が面白いのかわからなかった。モグライダーはもっと評価されるべき。ハライチも良かったですね，ちょっと時間オーバーしたっぽいけど。

の理想点を書き足したのが図 15.4 です。低く評価したランジャタイからは遠く，高く評価したハライチやオズワルドに近いところに著者の理想的な笑いの点があり，錦鯉やインディアンスも近くにありますから，今回の優勝にはまぁ納得，と言ったことがわかります。皆さんも自分の理想の点を書き加えてみませんか?

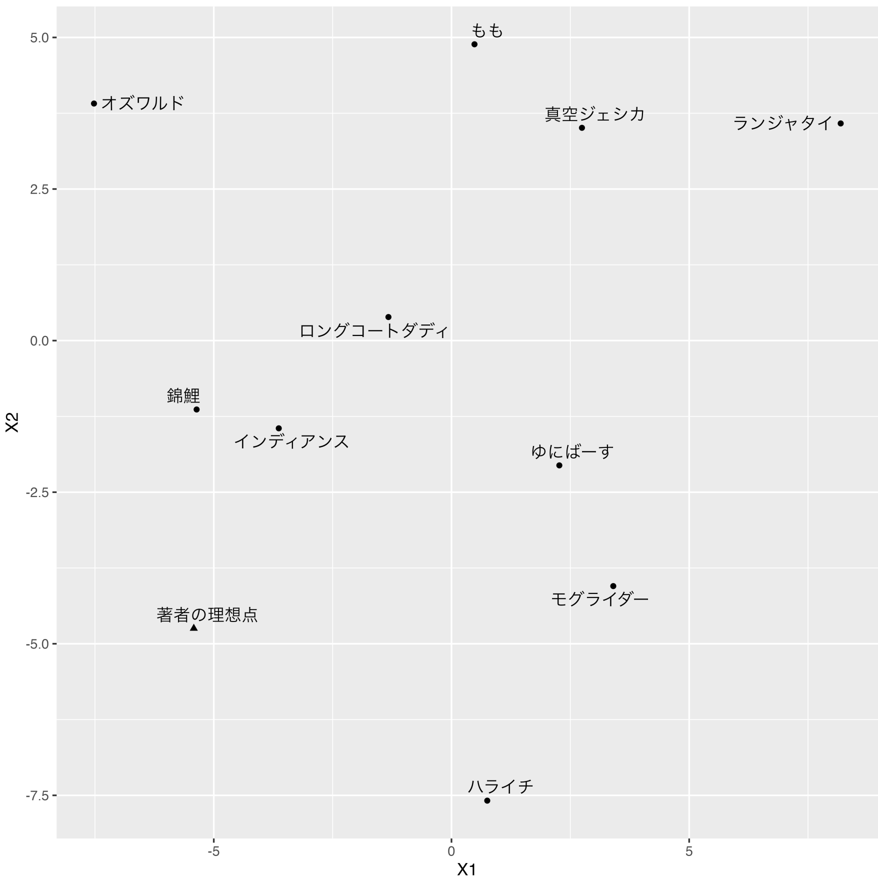


図 15.4　著者の理想点プロット

■Abelson Map　これは (Abelson, 1954) の考えた手法で，これも prefmap と同じく布置される対象に別の力 (選好度でもなんでもよい) があると考え，地図空間上に力の場をプロットして等高線を引くことで表現するモデルです。

この方法では，各点 P にかかる力 $V(P)$ を次のように定式化します。

$$V(p) = \sum_{j=1}^{M} \frac{V(j)}{1 + d_{pj}^2}$$

ここで j は各点，ここで言えば漫才師のことで，$V(j)$ が漫才師 j に与えられた評価点です。d_{pj}^2 は任意の点 p と対象 j の距離ですから，任意の点 p にかかる力は各漫才師が発する笑い力ですね。

この方法で，先ほどの M-1 プロットに，著者の評価を Abelson Map の方法で描き加えてみましょう。図 15.5 の右にある，ランジャタイの周りにたくさんの線が引かれていますが，いわばこれは低気圧のようにここのパワーが弱い (と著者は思っている) ところになります。実は，真空ジェシカの左側に通っているラインが平均点のラインですから，ここを境に著者は「面白い」と「面白くない」を区分しているとも言えます。ハライチや

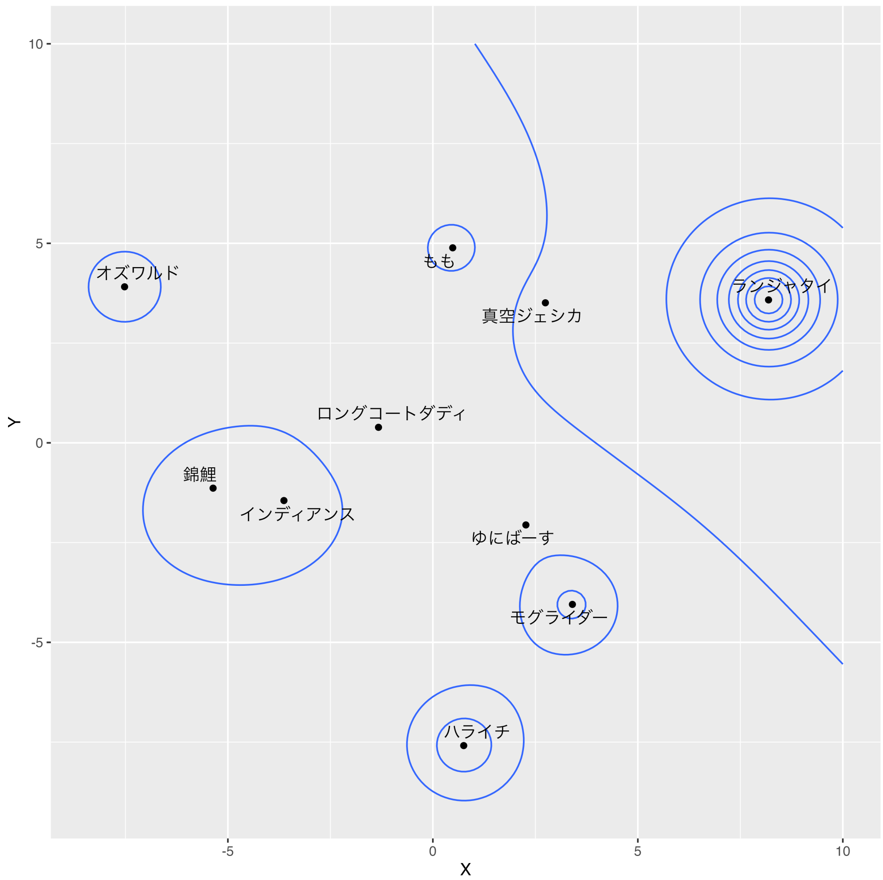


図 15.5　Abelson Mapping によるお笑い力の等高線図

オズワルドの周りにある等高線は山の高さ，パワーの強さを表しているラインですね。

　力のメタファーにすぎないと言われたらそうですが，このように色々なものを可視化できるのも MDS の面白いところです。

■非対称多次元尺度構成法　距離データは対称でなければならない，というものの，たとえば好きな人に振り向いてもらえない＝片想いとか，都市間の人口の流入・流出，国際貿易の赤字・黒字などを考えると，対称間の関係が対称出ないことは少なくないわけです。そうした非対称な関係を，図に加えようと言うのが非対称 MDS です。非対称情報を対象部分 ＋ 歪対象部分に分割し，この対称でない要素を図の中に書き加えるモデルが一般的で，プロットする対象の周りに縁や楕円で表現するとか，矢印で表現するとか，vonMises 分布と呼ばれる**確率分布**で表現するとか，さきほどの Abelson Map で高さとして表現するなどのモデルがあります。より数学的なモデルとして，プロットする空間を複素空間にするモデルもあります。詳しくは千野・岡田・佐部利 (2012) を参考にしてください。

■多次元展開法　リッカート法は心理尺度でもっともよく使われる方法で，「非常に当てはまる」から「まったく当てはまらない」までの段階的な反応を被験者に求める方法です。これは**段階反応モデル**で分析されることからも明らかなように，項目に対する反応段階が順番的に強くなっていくことを仮定しています。しかし，実際に被験者が丸をつけるときは，「自分の感覚にもっとも近いカテゴリーを選ぶ」ということをしているはずです。つまり，被験者と反応カテゴリーの距離が，尺度に反映されているはずなのです。この考え方から，

尺度に対する反応を距離とし，項目とそれに回答した被験者の両方をプロットする地図を書く方法があります。Coombs が考えた**多次元展開法 (Multi-dimensional unfolding method)** と呼ばれる手法がそれで，この手法から態度測定の尺度化を考えることもできます。発想としては数量化理論に近く，また清水 (2018) はこれを確率モデルに展開しています。

ここであげたいくつかの例にあるように，多次元尺度法もさまざまな角度から人の判断や反応から，**尺度 (scale)** を作る方法です。多次元尺度構成法は因子分析モデルよりも制約や仮定が少なく，より直接的に心理的反応をモデル化しようとしています。

本講で学んだように，心理学ではさまざまな角度から「数値化するルール」を考えてきていますが，それぞれ仮定や目的，何を良い尺度と考えるかという観点がことなります。我々心理学者は，統計的なツールのユーザに過ぎません。データは分析できればそれでいい，分析は機械がやってくれるから深く考えなくていい，という割り切り方もあるかもしれませんが，「そもそも何をデータとするか」「どのように数値化するか」という点については，そのような態度は取れないはずです。なぜならそもそも「心はいかにして数値化できるか」ということが明らかでないならば，その後の分析はすべて嘘っぽい数字を使った統計ごっこに過ぎないからです。**心理学者であるために，われわれは何をどのように数値化しているかについて，しっかりと理解する必要があるのです。**

15.5 課題

計量 MDS, 非計量 MDS をそれぞれ一例ずつ，実践してみてください。データはどのようなものでも構いません。提出ファイル形式は R スクリプトか Rmd とします。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

第 II 部

心理学データ解析応用 2

第 16 章

プログラミングの基礎

さて，ここから始まる統計の話は，統計学の中でも最近使われるようになってきたベイズ統計を駆使し，色々な心理学的現象を分析していこうというものになります。

このアプローチはデータに合わせて分析モデルを考えていくことになるので，ボタンをポチポチ押すと答えが出るような定型パターンの分析ではありません。個別のケースに合わせて分析モデルを考えていくことになります。その時に必要になってくるのがプログラミング技術ですので，まずはウォーミングアップということでプログラミングの話を始めていきたいと思います。中にはうんちくのようなところもありますので，興味があるところだけ飛ばして読んでいただいて構いません。最後にあげた課題ができるのであれば，知識は後からでもいいと思います。

16.1　プログラミングの基礎

プログラミングとは，コンピュータに計算をさせるその仕様書を作ること，だと思ってください。お料理のレシピのように，計算レシピを書くわけですね。ただ相手は計算機ですので，言葉の端々まで正確に伝えないと理解してくれません。よく「思った通りに動いてくれない！」と不満を訴える初心者がいますが，それは当然で，プログラムは**思った通りに動くのではなく，書いた通りに動く**のです。思った通りに動かないのであれば，それは仕様書・レシピの方に間違いがあります。

今の第一原則に加えて，プログラミングを進める上での注意点をあげておきます。簡単なことだと思うかもしれませんが，基本的なルールをしっかり守ることが上達への近道です。

1. プログラムは思った通りには動かない。書いた通りに動く。大文字，小文字，スペルの違いに注意する。
2. 書き間違えないための工夫は美しさ。綺麗に書くことが大事。
3. 一言一句すべてに意味がある。ただの写経ではなく，意味を考えながら書く。
4. 「遊び心」をもって！ここを変えたらどうなるか，を少しずつ「やってみる」が大事
5. 変更は少しずつ。一気に変えるとどこが変わったかわからなくなるから。

16.1.1　綺麗に書こう

綺麗に書く，というのはコードの可読性を上げるために必要です。綺麗な書き方として，松浦 (2016) は次の 5 点をあげています。

- インデントは必ずする
- データを表す変数の先頭の文字は大文字。パラメータを表す変数の先頭の文字は小文字。

- 各ブロックの間は1行あける
- camelCase ではなく `snake_case` で。
- 「 」や「＝」の前後は 1 スペースあける

1 つめのインデントとは，字下げのことです。行の頭を少し凹ませることで，違う人まとまりであることを明示するのです。たとえば次の 2 つのコードは同じ働きをしますが，16.2 の方が見やすいと思いませんか？

code : 16.1　インデントのないコード

```
data {
int<lower=0> N;
array[N] y;}
parameters {
real mu;
real<lower=0> sigma;}
model {
for(i in 1:N)
y[i]~normal(mu, sigma);}
}
```

code : 16.2　インデントのあるコード

```
data {
  int<lower=0> N;
  array[N] y;
}

parameters {
  real mu;
  real<lower=0> sigma;
}

model {
    for( i in 1:N ){
        y[i] ~ normal(mu, sigma);
    }
}
```

インデントのある 16.2 は，`data` というブロック ({ と} で囲まれている領域) の中に，2 つの行が入っている，ということが明確にわかります。また `model` というブロックの中には，`for` から始まる分がありますが，これも複数行に渡るブロックを構成する文なので，その中身はインデント (字下げ) されています。このように，インデントすることでどこの列がどこまでブロックを組んでいるのかがわかりやすくなるのです。ちなみにこのインデントを作るのは TAB キーを使います。TAB キーは普段，どう使うのかわからないものだと思われがちですが，このインデントをしてくれるためのものです[*1]。コードエディタによっては，カッコで括ったときに自動的に閉じるカッコを用意し，また改行ごとに字下げ位置を合わせてくれるものがあります。これらの機能を使って是非わかりやすいコードを書いてください。

松浦 (2016) の指摘の 2,3 番目についてはお好みで，4 番目もそれほど強い・広く浸透した決まりではあ

*1 ここには流派があって，TAB でインデントする派とスペースキーを 4 回または 8 回押してインデントする派がいます。どちらでも働きは同じなのですが，個人的には数回の字下げを 1 回のキーで行ってくれる TAB のほうが良いと思っています。

りません[*2]。ただし，5 番目のスペースの前後に少し空白を取るのは，見やすさのためにも是非実行してもらいたいところです。

これらの書き方は，**可読性**を上げるためのものです。プログラムは仕様書・レシピですから，あとで読み直した時やほかの人が読むときも，意味がわかるようにしておいた方が良いのです。自分だけわかれば良い，と思うかもしれませんが，その自分ですら何があるのかわからなくなってしまう可能性があります。そのような読みにくさはすぐにバグにつながりますから，綺麗に書くことを常々心がけておいて欲しいのです。

16.1.2 意味を考えて書こう

プログラミング言語は，ほとんど英単語のようなものです。プログラミング上の都合から，短い言葉に省略されていたり，アンダースコアやピリオドでつながって書いてあったりしますが，比較的わかりやすい言葉であることに違いはありません。

プログラムを習得し上達するためにすべきことは，最初は写経と呼ばれる，テキストや指示にしたがって書き写すことです。もちろんネット上にあるサンプルコードなどをコピー＆ペーストしても良いのですが，自分用の分析コードを書くためには手を入れる必要があったりします。ですから，最初はなるべく自分でキーボードを叩いて，コピー＆ペーストではなく入力する経験を積んでください。もちろん間違えてしまうことが多いでしょうが，転ばずに歩けるようになった人が一人もいないように[*3]，ミスを犯すことでどういうミスだったかを考え，自覚することではじめて，すこしずつですが b ミスが減っていきます。失敗する経験も時には必要なのです。

ということで，時折自分なりにコードを書いてもらうのですが，その時に「ただの記号列」と思わないでください。すでに書いたように，英語あるいはそれを省略した表現になっているので，見慣れないものかもしれませんが必ず意味があります。プログラムを始める最初の頃は，ミススペルが多く，あちこちでエラーが出て気持ちが萎えることもあるかもしれません。エラーが出る時は目を凝らして，どこに問題があるかを考えて修正することになります[*4]。ここで難しいのが，x や s などの大文字と小文字の区別がつきにくいもの，1 と l のように違いがわかりにくいものがあるということです。私は職業柄，多くの人に教え，多くのバグを見つけてきましたが，その中でも特別見つけるのに苦労したのが，`norma1` というミススペルでした。みなさん，これのどこがミススペルかわかりますよね？でもこれ，書く時に「正規分布のことだな」と思っていれば，そもそも入力時にそんな入れ方はしないと思うのです。ただの字だ，と心を無にしてしまうと，後々見つけにくいエラーを作ることにもなりますから，この文字・この名前・この関数はどういう意味だろう，と少し考えながら取り組んでみてください。

[*2] ちなみにキャメルケース Camel Case，スネークケース Snake Case とは変数名のつけ方のルールです。R などプログラミング言語ではあらゆるものに名前をつけて管理しますが，名前のつけ方は任意です。命名はわかりやすい方がいいですが，長いものになると区切りを入れたくなるかもしれません。さて Camel とはラクダの意味で，ラクダのコブのように区切りのところだけ大文字にする，という記法です。Snake は蛇のことで，区切りのところにアンダースコア_を使うというものです。たとえば野球チーム，という変数名をつけたい時に，キャメルケースのやり方だと `BaseballTeam` のように書きますし，スネークケースのやり方ですと `Baseball_team` のような書き方になります。ちなみに R は日本語での命名も許しますが，そのほかの言語では一般的ではありませんし，何より全角と半角を切り替える時のミスやエラーが多くなるので，半角英数字での命名をすべきという点はどちらでも共通です。

[*3] お釈迦様は除く。

[*4] ちなみにプログラミング歴 30 年以上の私でも，簡単な英単語，たとえば library のスペルを間違えるなんてことはしょっちゅうです。

16.1.3 遊び心が大事

プログラミングを進める上では，遊び心が大事です。うまく動くコードがかけたら，もうおかしなことになりたくない，と手をつけないままにしてしまう人がいます。

ところで，今まで教えてきた経験からいって，プログラミングが上達する人は，うまくコードが書けたらすぐに「ここをこう変えたらどうなるんだろう」と試してみる人だと思います。最初はプログラムの意味がまったくわからないので，どこをどういじっていいのか見当もつかないかもしれません。しかしたとえば `blue` という文字が出てきたら，これは色の青のことじゃないか，と思いますよね。ではその `blue` を `red` に変えたらどうなるんだろう？やってみよう！となるような，そういう遊び心が大事なのです。

もちろん変えてしまったことでエラーになって，動かなくしてしまうことがあるかもしれません。その場合は，元に戻して元通り動くかどうか，さらに確認すれば良いのです。数字や文字を少しいじっただけで，PC が爆発してしまうようなことはありませんから，ちょっと遊び心を出して，どうなるのかな？と思う気持ちを大事にしてください。

16.1.4 変える時は少しずつ

プログラムを色々いじって遊ぶことが成長につながる，という話をしました。ただし遊び方には気をつけましょう。まずうまくいくコードができれば，それは保存しておいて，また同じ内容のものを別名で保存し，「遊んでいい方」「壊れてもいい方」を作って，そちらで色々変えて遊ぶといいでしょう。最悪なことがあっても，うまくいくバージョンは常に残っているわけですから。

そして色々試す時のコツは，一箇所ずつ変更していくことです。たとえば `blue` を `red` に変え，`line` を `box` に変え･･･と複数箇所を同時に変更して，うまくいかなかった場合，どちらが原因になっているのか把握できませんよね。これは心理学実験でいうところの交絡です。操作が交絡してしまって，原因が特定できなくなるのです。

また，実行するときも 1 行ずつやりましょう。とくに R は一問一答型，つまり 1 つの命令について 1 つの反応を随時返してきます。複数行をまとめて実行することもできますが，まとめて実行している途中でエラーが出ていて，その後の計算がすべて空回りしているということも少なくありません。エラーがどこで生じているのか，しっかり特定することが重要です。そのためにも焦らず，1 つずつ確実にできることを積み重ねていくという姿勢が重要です。

統計分析も複雑で細部まで設定し尽くしたモデルを作り上げていくことができますが，いきなり全体が完成するのではなく，小さなピースの積み重ねなのです[*5]。

16.2 プログラミング言語

それでは具体的に，プログラミング言語としての R の機能をいろいろ見ていきましょう。

[*5] R やプログラミングで統計を学ぶ意義はここにあります。GUI で操作できる統計パッケージも少なくありませんが，それらは画面が出てきた時にデフォルトで設定されている値があるのがほとんどで，自分が何をやっているのか細部まで気づかないまま実行できてしまうのです。そうすると当然，エラーが出たり，うまくいっても誤用している，ということにもなりかねません。自分で理解できていない技術に振り回されないようにするために，しっかりとわかるピースを積み重ねていく必要があるのです。

16.3 プログラミング言語の基本的な働き

プログラミング言語はさまざまありますが，それらのやっていることは基本的に計算であり，その表現方法が違うだけです。どの言語にも共通する，プログラミング言語の特徴的な働きは「代入」と「反復」と「条件分岐」です。以下，この 3 つの働きについて R を使ってみていきます。

16.3.1 代入

料理でよくあるシーンとして，「野菜を切ってザルにあげておく」など，いったん作業の途中経過を横に退けて別の作業をする，というのがあります。計算プロセスでも，「計算結果をいったん横に置いておく」という操作をしたいことがあります。これが**代入**で，コンピュータとしてはあるメモリ番地に値を書き込んでおく，という操作をすることになります。横に置いておく時に名前をつけることができ，これが R のなかでは**オブジェクト (Object)** と呼ばれるものになります。すでにみなさんもやったことがあると思いますが，計算結果をオブジェクトに代入しておくコードの例をみてみましょう (code:16.3)。

code：16.3　代入操作

```
1  a <- 1
2  b <- 2
3  a + b
4  a <- 3
5  a + b
```

■コード解説

1 行目　オブジェクト a に 1 を代入

2 行目　オブジェクト b に 2 を代入

3 行目　オブジェクト a と b に $+$ という操作。すなわち $1+2$ の計算を指示していることと同じ。

4 行目　オブジェクト a に 3 を代入。もともと入っていた情報は上書きされる。

5 行目　オブジェクト a と b に $+$ という操作。すなわち $3+2$ の計算を指示していることと同じ。

非常に単純な例で，今更何を，と思った人もいるかもしれませんが，これが基本中の基本なので改めて示しました。R での代入は<-あるいは=を使います。前者は矢印のイメージ，後者は a=3 のようにして，「a は 3 だよ」という意味で代入を表しています。ポイントは 3 行目の「オブジェクト名で値を指定していること」と，4 行目の「値の上書き」にあります。オブジェクト名で計算を表現することで，操作の一般化ができます。すなわち，3 行目と 5 行目が同じ式であるのに結果が違うように，「どんなに値が違っても同じ動作をする」ようにできるわけです。コンピュータがすごいのは，この手の代入がどれほど複雑に，どれほど繰り返されても決して計算ミスをせず，指示通りに動くことができるというところです。人間は「ベクトルの 2 番目の要素と 3 番目の要素を足す」という操作を数回やるだけでも計算ミスをすることがあり得ますが，コンピュータは同じ計算を 100 回やっても 1000 回やってもミスなく動作します！

ただし注意が必要なのは，長いプログラムを書いていて，途中で「さっきの分析をやり直そう」と思って部分的に実行し，さらに続けて分析をする，ということをやっている時に上書きが生じることがあることです。途中の部分的なやり直しでできた計算は，改めて冒頭からやり直すとうまくいかないことになったりします。上書きをする時は注意して，また完成版として清書する時は一度プログラムの冒頭から走らせてみて，処理が間違いなく進んでいるかを確認するようにしましょう。

また, ここでの代入例は 1 つの数字だけでしたが, 数字がセットになったベクトルや行列もオブジェクトに代入できます。セットではなく, 別々の要素だけどまとめて持っておきたいという時は list 型を使います。さらに一般的なデータセットは行に個体が, 列に変数がある矩形行列ですが, R では list 型の特殊ケースである data.frame 型と呼ばれます。このように, 形式による違いをデータの**型**といいます。

code : 16.4 さまざまなデータの型

```r
# 数字をセットで持つvector型, matrix型
x <- c(1, 2, 5, 8, 9)
A <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 2, nrow = 3)
# 型にこだわりなくなんでも収納list型
dataSet <- list(name = c("kosugi","koji"),
                v1 = c(1,2,4,5,3,5,6),
                v2 = matrix(c(1,2,3,4),ncol=2))
# 矩形に整えられたlist型であるdata.frame
df <- data.frame(list(name=c("kosugi","suzuki"),
              V1 = c(176,173),
              V2 = c(83,55)))
```

R の出力 16.1: データの型による持ち方の違い (コード 16.4 の作ったオブジェクト)

```
> x
[1] 1 2 5 8 9
> A
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> dataSet
$name
[1] "kosugi" "koji"

$v1
[1] 1 2 4 5 3 5 6

$v2
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> df
    name  V1 V2
1 kosugi 176 83
2 suzuki 173 55
```

16.3.2 反復

エビフライをたくさん作ることを考えてみましょう。「えびの殻を剥いて, 衣をつけてあげる」という動作を何回も繰り返すことになると思います。このような反復捜査もコンピュータは得意とするところです。なにせ, 疲

れを知りませんので。反復操作を指示するコードは `for` 文という書式を使います。反復計算コードの例は次のようなものです (code:16.5)。

code : 16.5　反復操作 1

```
1  a <- 0
2  for (i in 1:4) {
3    a <- a + 1
4    print(a)
5  }
```

■コード解説

1 行目　オブジェクト a に 0 を代入 (初期化)

2 行目　`for` 文開始。中括弧で括られている 5 行目までの中身を反復計算する。

3 行目　オブジェクト a に，もとの a に 1 を加えた値を代入 (上書き) している。

4 行目　オブジェクト a を出力させている。

5 行目　`for` 文ここまで。

ここでは 3 行目の「1 を加える」，4 行目の「表示する」という操作を繰り返していることになります。ポイントは 2 行目の書き方ですね。`for` の後ろの小カッコ () の中身が，繰り返しに使う要素の指定です。後ろの 1 : 4 は R 言語特有の書き方で，「1 から 4 まで」すなわち `1,2,3,4` を意味しています。変数 i がこの順番で変わるよ，ということを意味しているので，i は 1 回目，2 回目，3 回目，4 回目，と進んでいくことになります。code:16.6 のような書き方をすると，その振る舞いがよくわかるかと思います。

code : 16.6　反復操作 2

```
1  a <- 0
2  for (i in c(1, 3, 5, 15, 12)) {
3    print(paste(a, "に", i, "を加えます"))
4    a <- a + i
5    print(a)
6  }
```

4 行目には `a <- a + i` という代入があります。「a に $a+i$ を代入せよ」という意味であり，数学のイコール記号だと意味がわかりませんが ($2 = 2 + 1$ なんて式はおかしいですもんね)，ここでは $a + i$ の計算をしたものを，新たにオブジェクト a}**に上書きせよという意味になります。またここでのポイントは，反復用インデックスiが$1,3,5,15,12$の順に変わっていくということと，変わっていくインデックスiの値そのものも計算に使えるということです。2 点目については注意が必要で，反復用インデックスが$1,2,3,4$と変わるような例の場合，途中で**\verbi <- 1|などとしてしまうと，いつまでもインデックスが前に進まず永遠に計算が終わらないことになります。そのようなミスが生じないように，注意してくださいね。

また，この `for` 文は入れ子にして使うことができます。たとえば i が 1 から 5 まで変わり，j が 1 から 3 まで変わりながら計算をする，ということを考えて，次のような表現が可能です。

code : 16.7　反復操作 3

```
A <- matrix(1:15, ncol = 3, nrow = 5)
for (i in 1:5) {
  for (j in 1:3) {
```

```
4       print(paste("Aの",i,"行",j,"列目の要素は",A[i,j]))
5       A[i, j] <- A[i, j] + (i * j)
6     }
7   }
8   print(A)
```

■コード解説

1 行目　オブジェクト A は 3 行 5 列の行列で，1 から 15 までの数字が順に入っている。

2 行目　for 文開始。i が 1 から 5 まで変わる。

3 行目　for 文その 2 開始。j が 1 から 3 まで変わる。

4 行目　オブジェクトの中身を表示させる。

5 行目　行列 A の i 行 j 列目の要素に対し，$i \times j$ の計算結果を加えたものを代入 (上書き) させる。

6 行目　for 文その 2 がここまで。

7 行目　for 文その 1 がここまで。

8 行目　最終計算結果の表示。

行列の行，列それぞれについて順番に，各要素を指定しながら代入していくという計算です。挙動を確認しておきましょう。

16.3.3 条件分岐

「卵が 200 円より安かったら買ってくる」というようなお使い指示，ありますよね。これは「200 円以上の時は買わない」という意味でもあります。この 200 円かどうか，という条件に対して，その後の動きが「買う」「買わない」に分岐するので，条件分岐と呼ばれます。これをプログラムで書くと次のようになります。

code : 16.8 条件分岐

```
1  egg <- 250
2  if (egg < 200) {
3    print("卵を買います")
4  } else {
5    print("卵を買いません")
6  }
```

■コード解説

1 行目　オブジェクト egg に 250 を代入。

2 行目　if 文開始。() 内が条件で，条件が該当すれば次の中括弧で括られている領域までを実行する。

3 行目　「卵を買います」と画面表示させる。

4 行目　条件が該当した時の実行内容を閉じつつ，該当しない場合の実行内容を書く領域を展開する。

5 行目　「卵を買いません」と画面表示させる。

6 行目　条件が該当しない場合の実行内容を閉じる。

一行目の egg にさまざまな数字を入れて検証してみてください。思った通りの振る舞いができているでしょうか。この if 文は小括弧の中身が**条件節**ということになります。条件が成立していることを**真**または

TRUE と表現し，成立しないことを**偽**または **FALSE** と表現します[*6]。今回は egg 変数が 200 未満であることを条件としています。もし 200 円も含めたいのであれば (200 円以下にしたいのであれば)，`egg <=200` と書く必要があります。

ところで，ちょうど 200 円のときに，という一致を表す条件はどう書けば良いでしょうか？ `if(egg=200){...}`としたいところですが，このままでは括弧の中身が「egg オブジェクトに 200 を代入せよ」という命令と同じになってしまいます。条件節で使う「同じかどうか判定する」の記号はとくに，`==`で表します。「卵がちょうど 200 円であれば」という条件節は，`if(egg==200){...}`と書くのが正しい表記です。逆に「同じでないとき」を表現したい場合は，`!=`と書きます。他にも条件節は，「A かつ B のとき」,「A または B のとき」のような表現をしたくなることがよくあります。このような条件節の表記や計算のことを**論理演算**といい，「かつ」は`&&`,「または」は`||`で表現します。コード 16.9 に論理演算の例を示しました。これを実行して，R が `TRUE` か `FALSE` のどちらで返答してくるか，確認してみてください。

code : 16.9 論理演算

```
1  X <- 2
2  X > 3
3  X < 3
4  X == 3
5  X != 3
6  X > 1 && X < 3
7  X < 1 || X > 3
```

■コード解説

1 行目 オブジェクト X に 2 を代入しておきます。

2 行目 X は 3 より大きい?

3 行目 X は 3 より小さい?

4 行目 X は 3 と等しい?

5 行目 X は 3 と異なる?

6 行目 X は 1 より大きく，かつ，X は 3 より小さい?

6 行目 X は 1 より小さい，または，X は 3 より大きい?

条件分岐をする場合は，条件が思った通りに設定できているか，分岐した後のルートが間違いなく書かれているかなどに注意が必要です。というのも，我々が日常言語で使っている「もし〜なら XXX する」といった表現は曖昧なことがあり，書いた通りにしか解釈しないコンピュータは，思った通りに動いてくれないということがあるからです。Twitter 上で有名になったジョークに，次のようなものがあります[*7]。

> ある妻がプログラマの夫に「買い物にいって牛乳を 1 つ買ってきてちょうだい。卵があったら 6 つお願い」と言った。夫はしばらくして、牛乳を 6 パック買ってきた。妻は聞いた「なんで牛乳を 6 パックも買ってきたのよ！」夫いわく「だって、卵があったから……」

これはプログラミング的思考の例として示されており，気の利かない夫とされていますが，言外の意味を含めすぎた妻の条件分岐指示がまずかったといえます。より適切な指示の例は，次のようなものです[*8]。

*6 `TRUE,FALSE` が大文字なのは重要で，真偽を表す R の特別な用語です。こうした用語のことを予約語といいます。
*7 https://twitter.com/beamtetrode350b/status/1406773935069229057
*8 https://twitter.com/takl/status/1127065591380971521

> プログラマを夫にもち長年苦労してきた妻「1 パック 8 個以上入った 190 円以上 210 円以下の鶏卵のパックがあったら買ってきて。それがなかったら 1 パック 8 個以上入った 189 円を超えない最も高い鶏卵のパックを買ってきて。売り場について 15 分以内に見つけられなかったときは私に電話してから指示にしたがって」

これでもまだ指示が厳密でない！条件漏れの可能性がある！というコメントがありますが，「書いた通りにしか動かない」ことの重要さがお分かりいただけるかと思います。

16.4 まとめ

ここで説明したのは，レゴのピースのようなものです。ピース 1 つ 1 つは小さくて，それだけではなんの形も作り上げることができませんが，細かなピースでも組み合わせ次第で大きなオブジェを作ることができます。プログラミングにはたった 1 つの正解というのはなく，どういう形であれいと通りに動くものができればそれで構いません。書き方は人それぞれでよく，結果が伴うかどうかがポイントです。どのピースをどのように組み上げてもいいので，思い通りのものが作れるようにトレーニングをしておきましょう。

16.5 課題

■FizzBuzz 課題　1 から 15 までの数列に対して，次の条件にあった出力をするプログラムを書きなさい。なお，割り算の余りを計算する R の関数は `%%`，文字を出力する関数は `print` です。

- その数が 3 で割り切れる場合には、その数の代わりに「Fizz」を出力する
- その数が 5 で割り切れる場合には、その数の代わりに「Buzz」を出力する
- その数が 3 でも 5 でも割り切れる場合には、その数の代わりに「FizzBuzz」を出力する

■行列のかけ算　行列 $\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 2 & 0 \end{pmatrix}$ と $\boldsymbol{B} = \begin{pmatrix} 1 & 5 \\ 2 & 3 \\ 3 & 8 \end{pmatrix}$ を用意し，$\boldsymbol{AB}$ の掛け算をしたいとします。R の行列の積を求める記号ではなく，`for` 文をつかって，この計算をするプログラムを書きなさい。

第 17 章

データ生成モデリング

17.1 データ生成モデリング

「心理学統計法」と名のつく講義のほとんどは，記述統計や推測統計，とくに帰無仮説検定を扱うことが中心であるのが現状です[*1]。心理学において帰無仮説検定を行う理由は，心理学実験の効果を検証することと，それも手元の標本についての記述ではなく母集団においてもその効果があると言えるかどうかを検証するためです。手元のデータに基づいて，目に見えない母集団全体のことに推論するというのは，基本的に不良設定問題です。つまり，確実な正解を導き出すにはヒントが少なすぎる，圧倒的に不利な状況で知恵を絞る必要があるのです。こうした不利な状況下ですから，いくつかの仮定をおいて，確率的に推論するのでした。その推論の方法として，**モーメント法** (**moment method**)，**最尤法** (**Maximum Likelohood estimation**)，**ベイズ法** (**Bayesian estimation**) があるのでした[*2]。

これらの推定方法が**実験計画** (**Experimerntal Design**)(要因計画ともいう) と組み合わせで用いられ，帰無仮説検定となるのでした。心理学における実験的なアプローチは，実験群と統制群に無作為に割り当てた集団に対し，介入・処置のあとでの群平均を見ることでその効果を検証します。人間を相手にする研究ですから，当然誤差や個人差がデータには含まれますが，無作為割り当てと平均化によってそれらはキャンセルアウトされ，平均の比較をすることで効果を見ることができると考えるのでした。また標本の平均ではなく，それを用いて母平均を推測することで，結果の一般化を考えます。母平均の推定には点推定と区間推定とがあり，確率的表現を用いた区間推定をつかって慎重に結論を導き出すのです。ただし区間推定は判断基準が明確ではありませんから，帰無仮説と対立仮説という 2 つのモデルを戦わせて決着を見る，というやり方をして一応の決着を見るのでした。こうした「実験計画」＋「推測統計学」＝「帰無仮説検定」は，長らく心理学のスタンダードとして君臨しています。

こうしたアプローチについて，昨今批判があることについては，今は触れないでおきましょう[*3]。それよりもこうした問いの立て方や解決策に注目してみたいと思います。心理学は物理学を範として，科学 science の仲間入りをしたい，というモチベーションが強く根付いている世界であり，また人間というのは嘘をついたり間違えたりするものだ，ということが骨身に染みてわかっていますから[*4]，データを分析する際にも**客観性**を大事にすることが殊更重視されます。客観性の反対は主観性，つまり本人の思い込みや考え方の癖がもっとも

[*1] とくに公認心理士に必要な授業として，科目名「心理学統計法」を名乗る必要があり，そこで教える基本的な内容としてこれらが含まれています。なお測定論やそれに関する多変量解析法はそれほど中心的話題ではありません。みんな使うのになあ。

[*2] これら 3 つの方法について，十分に思い出せない人は一年時の授業資料を振り返ってみてください。

[*3] Amrhein, Greenland and McShane (2019) などが指摘するように，誤った使い方をされることが多く心理学研究の意義そのものが疑われるほどになっている現状があります。また豊田 (2020) ではさらに辛辣に批判がなされています。

[*4] というか心理学の研究というのは，いかに人間がダメで偏った考え方をする生き物であるかを，滔々と明らかにしていくという側面もあります。

邪魔になるのです。そこでデータに対しては真っ白な気持ちで向き合うものだ，という姿勢をとることになります。

言い方を変えると，分析をする前はデータについて何も考えてないよ，という態度をとるわけです。もちろんデータは要因計画の結果として得られるものですから，計画を立てる際は主観的な誤謬が微塵も入り込むことないように緻密に練り上げるのですが，出てきた結果は結果，数字の羅列に過ぎないと考えるのです。その結果を統計的に分析するときは，それが記憶実験の数字であろうが臨床実験の結果であろうが気にすることはなく，淡々と帰無仮説検定の俎上に乗せていくことになります。こうしたアプローチができるからこそ，そしてそれを許す統計ソフトウェアがあるからこそ，心理学の研究を積み重ねていくことができたのだという一面があります。こうした研究アプローチは，データ駆動型分析と言えるでしょう。つまりデータができてから，分析が始まるという考え方です。

これに対して，データがどのように生まれてきたのか，そのメカニズムを考え，その仕組みを明らかにしていこうというのが**データ生成モデリング**です。松浦 (2016) は統計モデリングを「確率モデルをデータに当てはめて現象の理解と予測を促す営みのこと」と定義しています。すなわちこのアプローチは，まずデータがどのようなメカニズムで生まれてきたのかを，簡潔な数式を使って表現します。そのモデルをデータに当てはめ，このモデルからデータが出てきたと言えるかどうか，他のモデルの方が今のデータをうまく説明するのではないか，といった比較検証をしていくことになります。データ駆動型分析では，こうしたメカニズムが実験計画の中に暗黙理に埋め込まれていたと言えるかもしれません。データ生成モデル駆動型の分析は，メカニズムを明示して検証する，というところが違います。

データが作られるメカニズムを考えるアプローチの利点は，予測にも向います。メカニズムが正しい，あるいはうまく現状のデータを再現できるのであれば，おそらく今後も同じようにデータが作られていくでしょう。であれば将来の予測ができる，という考え方です。たとえば市場の動向の予測，消費者の傾向の予測ができればそのメリットは想像に難くありません。昨今はデータサイエンスという言葉が流行していますが，そこにはこうした研究アプローチの隆盛があるのです。もちろん，心理学においてもモデリングのアプローチは有用ですし，従来通りの平均値の比較をする上でも，さまざまなメリットがあります。

この授業では，データ生成モデリングのアプローチをとって，心理学的にもどういう意義があるのかを考えていきましょう。

これを考える上で重要なのが**確率モデル (stochastic model)** です。言葉の通り，データ生成過程を確率を使って表現します。なぜ確率を？と思うかもしれませんが，手元のデータは誤差や個人差を含んで微小に変わる，偶然の成分が必ず入っているからです。こうした偶然をハンドリングし，偶然の中にも理論的な予測をするという点で確率が必要になるのです。また確率モデルで考えるときに，最尤法よりもベイズ法の方がよく使われます。確率モデルが複雑になっていくにつれ，最尤法では計算コストが非常に高く，実質的に解が得られないということも少なくありません。ベイズ法によるアプローチはこの問題をクリアしてくれるからです。それでは改めて，ベイズ法について復習しておきましょう。

17.2 ベイズ推定の基礎

データ生成モデルが確率の言葉で記述される，というのはすでに述べたところです。推測統計というのがそもそも不良設定問題，すなわち少なすぎるヒントから未知なるものを当てるという状況に置かれた学問であり，この「わからないこと」を確率で表現するからです。この確率についての考え方として，ベイズの公式というのがあるのでした。ベイズの定理は次のように書き表されます。

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

ここで $P(X)$ は X についての確率，という表現です。$P(A|B)$ は条件付き確率というもので，B が与えられた時の A の確率，という意味です。

ベイズの定理の用語は次の通りです。まず右辺の分子に注目しましょう。$P(D|\theta)$ とありますが，ここで D はデータを表しています。θ はデータを生む確率のパラメータです。$P(D|\theta)$ はパラメータ θ のもとでのデータ D が得られる程度を表すもの，という意味になります。これは**尤度 (likelihood)** と呼ばれるもので，パラメータの関数になっているのでした。たとえば正規分布からデータが生成されると考えるのであれば，手元のデータがパラメータ θ から出てくる可能性がどれぐらいあるのか，を表現していると言えるわけです。この尤度関数は，データを生み出すメカニズムの表現そのものです。

右辺の分子の第二項，$P(\theta)$ はパラメータの確率です。正規分布からデータが生成される例で言えば，尤度はあるパラメータの値からデータが得られる程度を表現しているのですが，そのパラメータが出てくる確率はそもそもどの程度であるか，を考えているのです。これは別名**事前分布 (prior distribution)** とも呼ばれます。実際にデータが出てくる前の段階で，そもそもそのパラメータがどれほど出やすいかという確率を表現しており，これは今回のデータを取る前までの事前のデータや経験に基づいている，ということができます。

右辺の分母，$P(D)$ はデータ全体が得られる確率であり，**周辺尤度 (marginal likelihood)** とか**エビデンス (evidence)** と呼ばれます。**正規化定数 (normalized constant)** ということもあります。これについては理解しにくいところもあるかもしれませんが，後に述べる理由によって，ひとまず深く考える必要はありません。

左辺はこの計算の結果得られる**事後分布 (posterior distribution)** を表しています。$P(\theta|D)$ はデータ D で条件づけられたパラメータ θ の確率です。我々は確率モデルを作るわけですが，そのパラメータがどうなっているかは事前にはわかりません。が，データを取ることで「データが与えられたのであれば，パラメータはこうだよ」というのがわかるわけです。あるいは事前にパラメータはこのあたりにあるのではないか，と経験上考えていたとしても，それがデータを取ることによって更新される (確信が強くなったり，違うかもしれないと思い直したり)，ということを意味しています。

この式を言葉で表現し直すと，次のようになります。

$$\text{事後分布} = \frac{\text{尤度} \times \text{事前分布}}{\text{周辺尤度}} = \text{尤度} \times \frac{\text{事前分布}}{\text{周辺尤度}}$$

尤度がメカニズムそのものだ，ということはすでに書きました。たとえば**回帰分析**においても，尤度を計算できます。普通の回帰分析は，誤差が確率的に生じると考え，誤差以外のところは線形モデルで表現します。すなわち，従属変数 Y_i に対して，独立変数 X_i があったとすると，

$$\begin{aligned} Y_i &= \hat{Y}_i + e_i = \beta_0 + \beta_1 X_i + e_i \\ e_i &\sim N(0, \sigma) \end{aligned}$$

より，

$$Y_i \sim N(\beta_0 + \beta_1 X_i, \sigma)$$

となるのでした。ここで $N(\mu, \sigma)$ は平均 μ，標準偏差 σ の正規分布を表し，データ Y_i が正規分布から生成されているというモデルになっています。回帰分析の場合は**最尤法**で未知数 β_0, β_1, σ を求めたりしましたが[*5]，その名の通りこのモデルが示す尤度を最大にする未知数の求め方を最尤法というのでした。

ですが，尤度は確率を表すものではありません。確率とは非負の実数で，すべての確率を足し合わせる (あるいは積分する) と 1.0 にならなければなりませんが，尤度は足し合わせても 1.0 にならない数字なので

[*5] あるいは**最小二乗法 (Least Square method)** で求めるのですが，これは幸い最尤法の結果と一致します。正規分布を使った線形回帰モデルの場合，データの記述統計的値と確率モデルによる推定値が一致するので，気づかないまま推測統計学に足を踏み入れてしまえるのでした

す。ですから，「手元のデータがパラメータからでてくる**可能性**」という変な言い回しをしていました。「出てくる**確率**」とは言えないのですね。そこで，尤度を確率に置き換えたい，と考えたときに便利なのがベイズの定理なのでした。ベイズの定理は尤度に事前分布をかけ，周辺尤度で割るという操作によって，事後分布が得られる式，と読むこともできます。事後分布は**確率分布**です。尤度にある数字をかけて (事後の) 確率分布にしていると考えるといいでしょう。ちなみに事前分布も確率分布ですが，**周辺尤度**は確率ではありません。たとえば度数を総度数で割った相対頻度は確率と考えることができますが，それと同じように，分子をとある数字で割って，全体を 1.0 に整えるための定数なのです。**正規化定数**というのはそういう意味ですね。定数倍 (正確には定数の逆数) をかけて大きさの調整をしているだけですので，ベイズの式はさらに次のように書き直すことができます。

$$\text{事後分布} \propto \text{尤度} \times \text{事前分布}$$

ここで $\propto$ は比例する，という関係を表しています。最終的には事後分布の形が知りたいのですが，その形を決めているのは分子の尤度と事前分布だけだ，ということになります。

ここで事前分布が**一様分布 (uniform distribution)** であれば，事後分布の形状には影響せず，事後分布は尤度そのものの形を反映することになります。従来のデータ駆動型分析では，データに対して事前の仮定を一切置かないということでしたが，ベイズ流の分析でもそのことは同様に表現できるわけです。

以上がベイズの定理についての簡単な復習でした。しかしベイズの定理自体は，1740 年代には明らかになっていたことであり，それがなぜ 21 世紀の今になって見直されてきたのでしょうか。それには色々な理由がありますが，その 1 つは最近まで「ベイズ流の分析は絵に描いた餅」だったからです。すなわち理屈ではこのような形になることは明らかだったのですが，実際に計算するのは難しいことが多々ありました。その問題を解決したのが**マルコフ連鎖モンテカルロ法 (Markov Chain Monte-Carlo method; MCMC)** と呼ばれる方法です。

17.3 マルコフ連鎖モンテカルロ法

ベイズの定理から，事前分布と尤度が分かれば，計算式を解いて事後分布の形を算出できます。しかし**確率分布**の式が複雑になればなるほど，その計算はとても難しくなります。確率関数を複数組み合わせたり，求めるべきパラメータの数が増えて行ったりすると，とてもじゃないけど計算して答えを出すことができない，ということになります。そのせいもあって，ベイズ統計学は長らく実践には向かない手法だったのですが，**マルコフ連鎖モンテカルロ法 (Markov Chain Monte-Carlo method)**，略して **MCMC** が出てきてからその状況が一変しました。

MCMC 法は近似的な答えを探す 1 つの方法です。MCMC という名前は「マルコフ連鎖」と「モンテカルロ法」の 2 つのパートからできあがっています。マルコフ連鎖は，目標となる確率分布状態になるような推移規則を作る方法であり，モンテカルロ法は乱数を発生させるアルゴリズムです。この 2 つが合体することで，確率分布の形がわからなくてもサンプルが得られるような，乱数発生器を作ることができるようになったのです。

ベイズ統計のゴールは事後分布を作ることです。事前分布を一様分布にして，尤度はベルヌーイ分布にする，といった簡単な場合ですととくに問題ないのですが，ある確率分布のパラメータがあって，さらにそれを生成する確率分布があるといった，階層的に入れ子になっているようなモデルが出てくると，「最終的な事後分布の形がわからない」という状況になります。正規分布とかポアソン分布といった，名前がついている確率分布であればその特徴がはっきりわかるのですが，それらを組み合わせて作られるものは名もなき合成関数になり，どんな形状をしているのか，まったく想像つかないことがあります。マルコフ連鎖を使うと，そのなもなき

合成関数の形状をとにかく作り上げることができる，そんな数学的技術です。

モンテカルロ法は乱数発生技術です。乱数というのは規則性のない数ですが，これを作るのはなかなか難しいものです。人間が 0-9 の数字を適当に書いていくと，知らず知らずのうちに均等でないパターンができてしまいます[*6]。コンピュータに (たとえば R に) 乱数があるじゃないか，と思われるかもしれませんが，コンピュータはあくまでも計算機でどこまでも合理的です。乱数によって動いているように見えますが，乱数に見えるような数字を生成するアルゴリズムがその中には埋め込まれていますから，正確には擬似乱数でしかありません。コンピュータの作る乱数は，もちろん人間が適当に思いつく乱数よりもより規則性が少ないですが，それでも基本的に

- 何らかの関数 g によって，内部状態 S_t をアップデートする；$S_{t+1} = g(S_t)$
- 内部状態 S_t から何らかの関数 h によって，実現値が生成される；$x = h(S_t)$

というステップを反復 ($t = 1, 2, 3 \ldots$) することで生成するのです。こうした乱数列ができれば，それを正規分布の形だとか二項分布の形に当てはめて出力することは簡単なのです。このステップ・バイ・ステップの計算法としてある確率分布に従う乱数発生技術があり，これがマルコフ過程とひっついてできたのが MCMC 法ということになります[*7]。

MCMC 法は，ですから，どんな形の確率分布であっても乱数のサンプリングは生成できる，という技術なわけです。事後分布の関数の形，形状がわからなくてもそこからの乱数は作れるという技術であり，コンピュータ技術の性能が発展した今では大量の乱数を生成することも瞬時に行われます。

乱数を用いるアプローチはいくつかの利点があります。たとえば**確率分布**の平均である期待値など，分布の特徴を記述するための計算は積分を含むので解析的に解くのは知識も技術も必要です。しかしその確率分布から乱数を発生させると，その平均値を求めることでその近似値を得ることができます。確率分布の計算が記述統計の計算に置き換えられるのであり，また統計ソフトウェアにとって大量のデータの記述統計量を描くのは造作もないことなのです。乱数による近似値は，あくまでも近似値，近くて似ている値に過ぎませんが，精度を上げるためにはその乱数の数を増やしてやるだけで良いことになります。この点もいいですね。

また求めたいパラメータが複数ある場合，たとえば正規分布だと平均と標準偏差が未知数ですし，回帰分析では平均の中に切片と傾きといった未知数が入っているわけですが，このような場合の事後分布は同時確率空間ということになります。すなわち平均と標準偏差という 2 つの変数の場合でも，可視化するなら 3 次元空間が必要です (x 軸に平均，y 軸に標準偏差，z 軸に確率密度)。こうした多次元空間において，あるパラメータだけについて期待値を計算したい，という場合はそれ以外のパラメータについては**周辺化**といって積分して全部の可能性をつぶしてしまう必要があるのですが，この計算も解析的にやるには実に大変なものです。しかし多次元の事後分布空間から生成された乱数があるなら，注目したい変数だけについての記述統計をすれば，他の変数を周辺化したこと同じになります。なんと便利なのでしょう。

このように，乱数を使ったアプローチは計算上非常に有利な特徴を揃えています。乱数を大量に発生させられるような計算機のパワーは，最近の PC でしたら十分持っていますから，最近になってベイズ統計学やモデリングアプローチが生きてきたわけですね。さらにありがたいことに，事後分布から乱数を発生させるためのプログラミング言語ツールが登場したのも大きいでしょう。古典的には BUGS というソフトウェアが，最

[*6] 嘘の 538(ゴサンパチ) という標語があって，人間がふと思いつきで数字を作ろうとすると 5，3，8 が多くできてしまうと言われています。エビデンス (出典) がわからないので本当かどうかわかりませんが・・・。

[*7] 余談ですが，スマホアプリなどで「ガチャを引く」というのも内部では乱数が生成されていて擬似的に「偶然あたりが出た」のを装っているに過ぎません。私はゲームなどをする時，擬似乱数に思いを馳せ「どこかの誰かに遊ばされている」と思うとやる気がなくなるので，ガチャを引くようなシーンは興醒めしてしまいます。やはりマルチプレイヤーゲームのように，人間が背後にいたほうがよっぽど意外な行動が見られますし，ゲームよりもリアルの世界の方が擬似的でない本物の偶然を楽しむことができておもしろいと思います。皆さんはどうお考えですか。

近では JAGS や Stan といったのがそれで，**確率的プログラミング言語 (stochastic programming language)** と呼ばれたりします。これらの言語では，尤度と事前分布をモデルとして表記し，それにデータを与えてやることで，事後分布からの乱数を生成します。いわば万能乱数発生器なのです。こうした環境が整ったことで，簡単に分析できるようになりました。

17.4 乱数によるアプローチの例

それでは MCMC を使った実践に入る前に，乱数を使って何ができるのか，R で確かめてみましょう。

17.4.1 乱数による近似の例

まずは馴染み深い**正規分布 (Normal distribution)** の例からいきましょう。

正規分布に感する R の関数は dnorm,pnorm， qnorm ,rnorm などがあります。この d,p,q,r は他の確率分布を表す関数の前につける接頭語で，d は確率密度，p は累積確率,q はある累積確率になるときの確率点，そして r が乱数発生を意味しているのでした。

code：17.1 乱数のコードの例

```
1 rm(list = ls())
2 set.seed(12345)
3 dnorm(0, mean = 0, sd = 1)
4 pnorm(0, mean = 0, sd = 1)
5 qnorm(0.6, mean = 0, sd = 1)
6 rnorm(10, mean = 0, sd = 1)
7 set.seed(12345)
8 rnorm(5, mean = 0, sd = 1)
```

■コード解説

1 行目　環境の初期化
2 行目　乱数発生開始点の設定
3 行目　標準正規分布の，確率点 $x = 0$ における確率密度の計算
4 行目　標準正規分布の，確率点 $x = 0$ までの累積確率
5 行目　標準正規分布の，累積確率 60% になるときの確率点
6 行目　標準正規分布から乱数を 10 個出力する
7 行目　乱数発生開始点の再設定
8 行目　標準正規分布から乱数を 5 個出力する

このコードを使って確認しておいて欲しいところは，3 つあります。まず d,p,q,r をつけたときの意味です。それぞれ正規分布のどういう数字を返しているのか，しっかり確認しておいてください。次に 7 行目にある乱数の発生です。これを実行すると，標準正規分布にし違う乱数が 10 個出力されます。乱数ですので，数字の並びに規則性はありません。バラバラの数字が出ていることを確認してください。最後に 8 行目，9 行目の内容です。8 行目は 2 行目と同じく，乱数の開始点を定めています。R で出力される乱数は，規則性がない数字とは言え，計算によって算出している数字ですから規則性は拭いきれず，再現してしまいます。どこから計算を始めるか，という開始点は**シード値 (seed value)** と呼ばれ，ここに入力した数字が開始点になります。シード値の設定にとくに意味はなく，好きな数字にしていただいて結構です。ポイントは，任意の数字で

あっても同じ数字に設定すると，乱数はそこから計算して算出されますので，9 行目で実行した 5 つの乱数は 7 行目の最初の 5 つと同じ数字が再現されているというところです。規則性のない数字なので再現されては困ると思うかもしれませんが，科学計算のアプローチという意味では再現性が担保できることも重要なのです。

では乱数を使う例をもう少し見てみましょう。次は**ベルヌーイ分布 (Bernoulli distribution)** について考えてみたいと思います。これはコイントスをして表が出るか，裏が出るかという 0/1 の離散的結果を生み出す分布です。残念ながら R にはベルヌーイ分布の関数はなく[*8]，二項分布の特殊例として使うことにします。**二項分布 (Binomial distribution)** とは，N 回コイントスをして K 回表が出る確率を表す分布ですが，これのコイントス回数が 1 回であればベルヌーイ分布と同じことになります。

code : 17.2 乱数のコードの例 2

```
1 rbinom(10, size = 1, prob = 0.5)
2 rbinom(10, size = 1, prob = 0.3)
3 rbinom(10, size = 1, prob = 0.7)
4 # パッケージの利用
5 library(extraDistr)
6 rbern(10, prob = 0.5)
```

■コード解説

1 行目 二項分布の乱数を 10 個発生させる。確率は 0.5 で。

2 行目 二項分布の乱数を 10 個発生させる。確率は 0.3 で。

3 行目 二項分布の乱数を 10 個発生させる。確率は 0.7 で。

4-6 行目 `extraDistr` パッケージによるベルヌーイ乱数例

ここで確認して欲しいのは，コイントスを 10 回するとして，確率 0.5 の場合はほぼ半々，0.3 の場合は半分以下，0.7 の場合は半分以上表 (1) が出ている，ということです[*9]。ここで確率を幾つにするかは，われわれが自由に設定できます。また乱数を幾つ発生させるかも自由です。これらの数字を色々変えて遊んでみましょう。

たとえば乱数の数を 100 回，1000 回，10000 回と増やしたとします。その結果をすべて表示するのは大変ですが，その平均値を計算してみるとどうなるでしょうか。

code : 17.3 乱数のコードの例 3

```
1 N100 <- rbinom(n = 100, size = 1, prob = 0.5)
2 N1000 <- rbinom(n = 1000, size = 1, prob = 0.5)
3 N10000 <- rbinom(n = 10000, size = 1, prob = 0.5)
4 mean(N100)
5 mean(N1000)
6 mean(N10000)
```

■コード解説

1-3 行目 二項分布の乱数を 100 個，1000 個，10000 個発生させる。確率は 0.5 で。

[*8] `extraDistr` パッケージを用いると，`bern` という関数名でベルヌーイ分布が使えます。

[*9] 正確には二項分布なので，1 が出た回数が出力されているのです。試行数 (`size`) を 1 にしているので，1 回中 1 回表が出た＝表 (1) が出た，と解釈しても，結果的に同じことであるというだけです。

4-6 行目　それぞれの平均値を計算する

これの実行結果は，私の環境では出力 17.1 のようになりました。

R の出力 17.1: 乱数出力の結果

```
> mean(N100)
[1] 0.62
> mean(N1000)
[1] 0.52
> mean(N10000)
[1] 0.5003
```

もともと確率を `prob = 0.5` と設定しましたから，理論的には 0.5，つまり半々の確率で表 (1) が出たり裏 (0) が出たりするはずですが，最初の 100 回では 62 回なのでやや表が多めに出たようです。乱数ですので，そういうことはあります。ただ，この傾向も，1000 回，100000 回と繰り返していくと，ほぼ偶然による偏りはなく，半々の比率になっていくのが確認できると思います。このように，乱数が十分多ければ，確率分布の近似値として使うことができるというわけです。

ちなみに今回は平均をとりましたが，これは確率分布の**期待値 (Expectation)** を計算したこと同じになります。

17.4.2　乱数を用いた確率分布のプロット

乱数を使って近似できる，という例を可視化で見てみましょう。また正規分布を例にしてみたいと思います。

code : 17.4　確率分布の可視化 1

```
1  x <- seq(-4, 4, 0.05)
2  plot(x, dnorm(x))
3  library(tidyverse)
4  ggplot(data.frame(x = c(-4, 4)), aes(x = x)) +
5    stat_function(fun = dnorm)
```

■コード解説

1 行目　-4 から 4 までの範囲で，0.05 刻みの数列を作る

2 行目　低水準プロット。各確率点の確率密度をプロットする

3 行目　パッケージの読み込み

4-5 行目　`ggplot` による美しい描画。`stat_function` は関数の結果を図にするもの

これは理論的な正規分布の形を図示するものです。美しいですね。でも同様のことが，乱数を使ってもできます。

code : 17.5　確率分布の可視化 2

```
1  N <- 1000
2  X <- rnorm(N, mean = 0, sd = 1) %>% as.data.frame()
3  ggplot(data = X, aes(x = .)) +
4    geom_histogram()
```

■コード解説

1 行目　発生させる乱数の数
2 行目　乱数を発生させ，`data.frame` 型に組み上げる
3-4 行目　`ggplot` による美しい描画。`geom_histogram` になっているところに注意

このコードの結果描かれるカーブは，それほど美しいものではないかもしれません。しかし発生させる乱数の数が増えればどうでしょうか。どんどん形が似ていくことがわかると思います。ここでのポイントは，ヒストグラムを描いたらその稜線が**確率分布**の形になること，乱数の数が増えれば十分な近似になることです。よく確認しておいてください。

17.4.3　乱数を用いた確率分布の要約

最後に，確率分布の特徴を記述するコードの書き方について練習しましょう。乱数発生によるアプローチは，記述統計量で確率分布の特徴を記述できます。確率分布の期待値は，その算術計算の平均で良いのですが，中央値やパーセンタイル，分散や標準偏差も分布を記述する重要な指標です。さらに確率分布における，確率密度が最も高くなる点は，「最も生じる確率が高い点」という意味で重要ですが，それを求めるためには特別な関数を書く必要があります。

まずは `data.frame` 型にした正規乱数を使って，記述統計量を算出する例を見てみましょう。

code : 17.6　確率分布の要約

```
1  N <- 10000
2  X <- rnorm(N, mean = 0, sd = 1) %>%
3    as.data.frame() %>%
4    dplyr::rename(val = 1)
5  X %>%
6    summarise(
7      Exp = mean(val),
8      SD = sd(val),
9      Median = median(val),
10     U50 = quantile(val, prob = 0.50),
11     U90 = quantile(val, prob = 0.90),
12     L90 = quantile(val, prob = 0.10)
13   )
```

■コード解説

1 行目　乱数を 10 万個作ることにします。
2 行目　標準正規分布の乱数発生
3 行目　データフレーム型に整形
4 行目　変数名がつけられていないので，`val` という名前をつけることに。ここでの 1 は 1 列目を意味している。
6 行目から　データフレーム `X` を使って計算
7 行目　`summarise` 関数は要約計算。変数名 = 計算式で表す。
8 行目　期待値を `mean` 関数で算出

9 行目　確率分布の標準偏差を sd 関数で算出[*10]
10 行目　中央値を median 関数で計算
11-13 行目　パーセンタイル点を算出。50 パーセンタイル点は中央値に同じ。

また，確率密度が最も高くなる点を算出するための関数を自分で作ってみることにします。次のようにコードを入力してください。

code : 17.7　確率分布の要約

```
1 map_estimation <- function(z) {
2   density(z)$x[which.max(density(z)$y)]
3 }
4 # 試してみる
5 X %>%
6   summarise(
7     MAP = map_estimation(val)
8   )
```

■コード解説

1 行目　関数の宣言。map_estimation という関数名を作る。この関数は引数 z を取る
2 行目　density 関数は与えられた数列の関数を考え，そのカーネル密度を計算します。which.max 関数はその中でも最も密度の高い値を返すものです。
3 行目　関数の終わり
5-8 行目　さきほどの例で試してみる。

このようにして，確率密度関数の近似値が簡単な関数で求められます。

次回以降は，モデルに基づいて計算された事後分布から得られる乱数について，これらの関数を使ってその特徴を記述していくことになります。関数や指標の意味をしっかりと理解しておいてください。

17.5　課題

次の計算をする R コードを記述し，提出してください。提出は R スクリプトファイルでも Rmd ファイルでも構いませんが，どの課題に対するコードなのかがわかるよう，コメントや説明文を記入しておくこと。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

1. 平均 50，標準偏差 10 の正規分布に従う乱数を 10 万点生成し，その平均値，中央値，標準偏差，15 パーセンタイル，75 パーセンタイル，2.5 パーセンタイル，97.5 パーセンタイルを算出してください。
2. さきほど求めた正規乱数の記述統計量が，理論値の近似値になっていることを確認します。dnorm,pnorm,qnorm などを使って理論値を算出してください。
3. 正規乱数のデータセットのうち，30 以上 60 未満の値が含まれる割合を計算してください。またそれが理論値の近似値になっていることを確認するため，dnorm,pnorm,qnorm などを使って理論値を算出してください。

[*10] R の sd 関数は不偏分散の平方根を計算しており，厳密には標本標準偏差の計算で良いが，サンプルサイズが十分に大きいのでほとんど影響しません。

第 18 章

いんたーみっしょん；Stan の概略と環境の準備について

このセクションでは，確率的プログラミング言語 Stan を導入するにあたっての，周辺知識を解説します。授業中に解説するものではありませんし，ここに書かれていることのすべてを理解していないと Stan を使えないというわけではありませんが，導入や利用にはトラブルやエラーも多く，その際に前提知識，周辺知識があるとないとでは理解度が大きく異なります。「わからなくても，なんとなく動いた」という状態で受講し続けるのは自身のためにならないだけでなく，**面白くない**です。知識があってはじめて価値がわかるということもありますので，共用としてご一読いただければと思います。

18.1　はじめに

この授業では統計言語としての R，それを使う統合開発環境としての RStudio，そして**確率的プログラミング言語 (stocastic programming language)** としての Stan を利用します。大学での PC ルームの利用にはこれらの環境がすでにある程度準備されていますが，最新バージョンではありませんので，自分の PC に環境を準備することを強く推奨します。今後の研究室配属やその後の卒業研究などでも活用することになると思いますので，まずは自身の PC 環境に準備することを考えてみてください。

この付録資料は，これらの環境を準備する方法について解説するものですが，提供されるソフトウェアのバージョンや対応する環境などは日々発展するものですので，必ずしも最適な情報提供になり得ません。基本的な情報は提供いたしますが，執筆時[*1]での情報であることも多く，より新しい情報についてはインターネットなどでキーワード検索を行なって，より良いものを探してみてください。

PC の操作が苦手に思っている人のために付言しますが，うまくいかなかったからといって癇癪を起こしたり，諦めたりしてはいけません。相手は機械ですから，命じられたことを愚直に実行しているだけです。また文房具のようなものですから，恐れて嫌うのではなく，愛して可愛がってあげることが肝要です。もしみなさんの中に，お持ちの PC に名前をつけていない人がいるようでしたら，今すぐ命名することをオススメします。パソコンが，と思うと腹が立ちますが，わたしの XXX ちゃんが，と思うとエラーも「ちょっとご機嫌斜めなのかしら」と受け入れやすくなります[*2]。

また，困った時は教員，TA，先輩などに相談することが重要ですが，その際には症状や経緯を正確に伝えることが必要です。ペットを病院に連れていくときに「何もしてないのに勝手に病気になった。治してほしい」

[*1] この記事は 2022 年 11 月 10 日に加筆修正されました。

[*2] ちなみに私が初めて手に入れたノート PC には「秀吉」という名前をつけました。Mac に変わってからは代々数学者の名前をつけることにしています。私はいま，Gauss ちゃんや Hermite ちゃんを持ち歩いています。

といわれても獣医さんは困ると思います。**普段の様子や症状についての丁寧な報告**を心がけてください。とくに PC は機械ですので，何もしていないのに壊れるということはありません。自分が自覚していないことであっても，「なにかをした」から様子がおかしくなるのです。自分はそんな大それたことはしていない[*3]，と思っても必ず何かトリガーがあったはずなのです。どこをクリックしたか，どういうコマンドを入れたかという**履歴をしっかりと把握し，丁寧に報告する**ことを心がけてください。

18.2 Stan の位置付け

確率的プログラミング言語 (stocastic programming language) とは，確率モデルを記述し，データと合わせることで統計的推論をしてくれるコンピュータ言語のことです。Stan ができる前は，**BUGS** や **JAGS(と)** いうものがありましたが，BUGs は開発が終わってしまいました[*4]。今，こうした言語は Stan が最先端だといって間違い無いでしょう。これらの言語は，具体的には確率モデルとデータの関係を記述することで，事後分布からの乱数を生成するというものです。以下ではこの言語の利用形式について解説します。

18.2.1 コンパイラとインタプリタ

プログラミングはコンピュータを動かすための仕様書を書くことです。ここで「コンピュータ」というのは計算機という意味だと思ってください。もちろんコンピュータは数字の計算だけでなく，音楽を奏でたりゲームを楽しませてくれたり，と色々なことができるのですが，その背後にあるのはとにかく 0/1 の数値演算です。0 か 1 かという数字がどうして映像や音声，通信になるのか，と疑ってしまうほど異なっているようですが，それでもすべて数字の計算から構成されています。

コンピュータができ始めたごく初期の頃は，これを動かすのに 0 と 1 からなる数字の羅列で仕様書を用意していました。流石にそれではわかりにくく表現力にも乏しいので，次にできたのがマシン語とよばれる 16 進数による表現でした。この段階でも，知らない人にとっては暗号か，意味のある連なりに思えない文字列にすぎません。画面に線を引いて欲しい時に `line` という命令で伝えられるようになって，やっと普通の人間にも意味がわかるレベルになってきます。このように，人間がわかるレベルでコンピュータに命令が伝えられるような言語のことを**高級言語**と言います。高級言語は人間寄りで，直接コンピュータが理解できませんので，この高級言語を機械の言葉に翻訳するためのアプリケーションを介在させます。それがいわゆる**プログラミング言語 (programming language)** と呼ばれるものです。

プログラミング言語は，古くは BASIC，PASCAL，FORTAN などと呼ばれる書式のものがあり，その後出てきた C 言語，その改良版である C++ 言語[*5]などが有名です。他にも JAVA や Object C，Python などが有名ですね。R も言語の一種で，統計解析に特化したものです。また，この授業で扱う Stan という**確率的プログラミング言語 (stocastic programming language)** は，確率モデルの分析に特化したものです。みなさんがデータサイエンス業界に進むのであれば，Python や R を使えるようになっておくといいでしょう。これらの言語の特徴の 1 つは，パッケージを追加することで機能が拡張できることにあります。とくに**機械学習**系 (AI など) のパッケージが豊富なのが Python です。またこれらの言語は基本的にフリーソフトウェアであり，導入にお金がかからないところもポイントですね。タダで始められるおもちゃみたいなもの

*3 たとえば計算途中だけど時間が来たので電源をオフにした，といったことでも，内部ファイルにアクセスしているときの中断であれば，十分に PC を破壊することができます。

*4 JAGS はまだ開発が続いているようで，R から使うこともできます。

*5 この ++ という書き方は，C 言語特有の「変数に 1 加える」という表記法であり，C 言語の改良版という意味で C++ と命名されています。読み方はシープラスプラスですが，シィプラプラの愛称でも知られています。

です！[6][7][8][9]

プログラミング言語の分類には，その設計思想や書き方などを基準にすることもできますが，実行方法を基準にするものとして**インタプリタ**型と**コンパイラ**型とに分けることができます。インタプリタ型は，interpret，つまり翻訳型です。これは毎回の命令を機械語に翻訳して実行していきます。R はインタプリタ型言語で，毎回の命令を逐一翻訳して実行していきます。R を実行するとき，コンソールに>というマークが出ていますよね。これは R が聞き耳を立てている状態，入力待ちの状態なのです。ここに命令を書くと (たとえば 2+3 と書く)，R はすぐさま答えを返してきます (たとえば [1] 5 と返す)。基本はこのように一問一答，ひとつひとつの命令を毎回機械語に翻訳して実行し，その答えを出力するという手続きをとっています。もちろん私たちの分析プログラムは，一行で書ける簡単なものではありませんから，複数行に渡る長々としたファイルを書くことが多いでしょう。こうしたファイルはプログラムともいわれますし，一行一行の命令のことを**スクリプト (script)** ということもあります。プログラムファイルあるいはスクリプトファイルを開いて，一行ずつ実行するのが R の基本的なスタイルであり，RStudio はスクリプトファイルを編集する**エディタ (editor)** がセットになっているのが便利な点なのでした。

これに対して，インタプリタ型は，スクリプトファイルをまとめて機械語に翻訳 (コンパイル) し，実行ファイルというのを別途作ります。その上で，実行ファイルを実行すると計算が進められるのです。どうしてこんな手間がかかることをするのでしょうか。ひとつには，ひとつひとつの命令分を逐一翻訳しているのでは，実行スピードが遅くなるということが挙げられます。命令を逐一聞き耳を立てて待ち，全ての命令を聞き終わるまで途中の指示を記憶しておいて，命令文が終わってはじめて行動に移す，というのでは記憶容量も時間もかかるわけですね。これに対して，一連の計算命令文書を一括で渡すことができれば，機械は内部的に効率よく計算手順を配置して実行できます。コンパイラ型は計算スピードが速いのです。ちなみに実行ファイルは機械がわかる言葉に書き換わっているので，人間が見ても意味がわかりません。また，OS や CPU に理解できる形に書き換えていますので，MacOS の実行ファイルを Windows で実行する，ということはできません。翻訳後の言語が違うからです。スクリプトファイルは環境を超えて共有できますが，それを翻訳したり最適化したりする仕組みは，個別の環境に依存します[10]。

Stan はコンパイラ型です。Stan のファイルを読み込んで，機械にわかるように「事後乱数生成命令文」に翻訳して実行します。確率モデルから乱数を生成するのは非常に高度な計算機能ですから，コンパイルしてまとめておくことでスピードアップの効率が断然良くなるわけです。コンパイラ型の利点はこのスピードにありますが，欠点は「命令に変更やミスがあれば翻訳し直さなければならない」というところです。あり得ない命令を出しているようであれば，コンパイルの段階で「おかしな文法だよ」とエラーを出して翻訳を止めてくれます。翻訳が通れば良いかというとそうではなく，翻訳はあくまでも文法上の手続きであって，数値や内容に間違いがあっても翻訳自体は完了させることができるのです。翻訳 (コンパイル) には少し時間がかかりますか

[6] 余談ですが，Scratch や任天堂 Switch の「ナビつき！ つくってわかる はじめてゲームプログラミング」などにみられる，GUI ベースのプログラミング言語もあります。これはプログラミングの要素がアイコンやキャラクターになっていて，それらを組み合わせることで計算させるという方法をとっています。その操作のしやすさ (ミススペルがない！) から，小学生などにも導入できる素晴らしい取り組みです。同様の発想は，LOGO 言語などにもみられました。

[7] Perl や Java Script など，Web ブラウザ上で動くプログラミング言語もありますが，これはブラウザ上での操作に限定されており，数値計算にはあまり向かないのでここには取り上げていません。さきほどあげた JAVA と JAVA Script は別物なので注意してください。

[8] GUI アプリケーションを作ることに特化した言語もあります。数値計算と違って，マウスがどこでクリックされたか，と言った「動き」を契機としてプログラムが走る，イベントドリブン型の言語で，Microsoft 社の Vsial Basic などが有名です。これは Microsoft Office 製品の中に埋め込むこともでき (VBA)，これを使うと Excel でも高度な統計解析ができたりします。Excel を使った統計ソフトウェアとして代表的なものに HAD があります (清水, 2016)。他にも Delphi などの言語がありました。

[9] (シ, 2016) には，数え方にも癖がありますが，117 ものプログラミング言語が紹介されています。

[10] ちなみに R や Rstudio はもちろん，Word や Excel などのオフィスアプリ，果ては OS そのものも，コンパイルされた実行ファイルを PC が計算して実行しているに過ぎません。

ら，文法上も内容上も間違いのない命令分をしっかり準備しておくことが重要になってきます。

18.2.2 ファイルのやり取りについて

ここで，R/RStudio をつかって Stan を使う際の，ファイルのやり取りを見ておきましょう (図 18.2)。

Stan のコードは R のコードの中に書くこともできますが，別のファイルに保存しておくのがベターです。Stan ファイルは拡張子を`.stan`とすることが一般的です。このファイルの中身はプレーンテキストでコードを書いていますから，一般的なエディタ[*11]で編集できます。RStudio のエディタ画面も必要十分なエディタ機能を持っていますので，RStudio をエディタとして利用するのがいいでしょう。

RStudio で File > New File として新しいファイルを開くときに，Stan ファイルとして開くことが可能です。Stan ファイルとして開くと，初心者向けの配慮からか，最初からちょっとしたコードがすでに書かれています。コードの書き方を示すサンプルコードなのですが，実際にこのコードを使って何かすることはありませんので，中身は全部削除してしまいましょう。それでも RStudio はその画面が Stan ファイルのものだということは認識していますから，そこで書いたファイルは Stan ファイルとして扱ってくれます。ここでまちがって新しく R Script で開いてしまった，というときに，保存するときだけ`.stan`をつけておけばいいか，という対応をすると失敗します。RStudio では拡張子をあえて表示していませんので，ファイル名として`hoge.stan`と書くと実際には`hoge.stan.R`というファイル名になっています。拡張子は最後のピリオドで判断されますので，これは R ファイルなのです。このミスを防ぐために，ファイル名にピリオドは使わないこと，ファイルの種類を変える時はエディタ画面の右下でファイル種別を選択してください (図 18.1)。

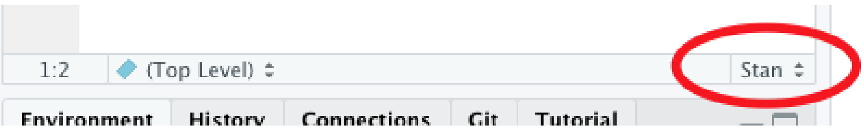


図 18.1 ファイル種別の変更

RStudio を使って Stan を書くと便利なことがいくつかあります。ひとつは強調表示機能で，ブロックや関数名など Stan で使うことが決まっている専門用語は色やフォントが変わって表示されます。`transformed parameters`のように長い専門用語を書くときはミススペルが心配ですが，書いた後で強調されなければスペルミスがある，ということがわかります。もうひとつの利点は文法チェックの機能です。コンパイル型なので，Stan に書かれた命令文は実行前に一括で翻訳されますが，スペルミスや型の違いなど，文法的に間違っているところがあればこれも自動的にチェックして警告記号がでます。また Stan ファイルが開かれているペインの左上に`Check`というボタンがあり，これを押すことでファイル全体の文法チェックをしてくれます。もし文法上の問題がなければ，「hogehoge.stan is syntactically correct.(hogehoge.stan というファイルは文法的には正しいですよ)」というメッセージがコンソールに表示されます。逆にいうと，これが表示されないというときは何か間違っているので，コンパイルに進む前に修正しましょう。

さてこのようにして Stan ファイルを準備します。また分析用のデータセットが外部ファイルにある場合などは，これまで同様，当該プロジェクトフォルダ内に置いておくと良いでしょう。これらはいずれも，R のコードで

[*11] エディタとは ASCII ファイル，いわゆるプレーンテキストを編集するアプリケーションの総称で，OS にデフォルトで「メモ帳」などの名称で含まれています。デフォルトのアプリは文字を読み書きできるだけですので，もう少し発展的，あるいは便利な拡張機能を持っている専用のアプリを使うことが一般的です。筆者は最近もっぱら VS Code というエディタを利用しています。他にも MacOS でしたら，「mi」というアプリを好んで使っていました。Windows でしたら「秀丸」というアプリが有名です。Ubuntu には gedit というアプリがデフォルトで入っていますし，歴史やユーザの多さでは vim や emacs などがあります。Linux 界隈で，vim 派か emacs 派かの話題は，きのこたけのこ戦争ばりに激しい戦いになります。

呼び出して使うことになります (図 18.2 の 1. と 2. のステップ)。R では，Stan ファイルやデータファイルを読み込んで，これを PC 内部にある Stan に渡します (図 18.2 の 3. のステップ)。Stan は R と違う言語であり，Stan 独自の計算機能で計算してくれるわけですから，R と Stan の橋渡しが必要になります。R ではこれを `cmdstanr` や `rstan` というパッケージを経由して行います。これらのパッケージが Stan を呼び出してくれるわけです[*12]。

データと確率モデルの関係を記した Stan ファイルと，データのそのものを受け取った Stan は，コンパイルして PC 内に乱数発生装置を作ります (図 18.2 の 4. のステップ)。事後分布の関数を書き出すのではなく，その形状とそれを代表する乱数を作る状態をもつわけです。R の指示をうけて，そこから必要な数だけ乱数を作り出します (図 18.2 の 5. のステップ) 。R はこれを受け取って，統計解析を始めるわけです。R は統計に特化した言語ですから，集計や可視化などの作業はお手のもの。事後分布から得られた大量の乱数は，事後分布の特徴を反映した大量のデータセット，事後分布の具体例たちになっているのでした。

分析の進め方は以上のようになります。R でデータを整形し，Stan のファイルも別途用意して，R から Stan を呼び出す，Stan か返してきたデータを R で解析する，ということです。R の拡張子は`.R` で，Stan の拡張子は`.stan`，データファイルの拡張子は`.csv` などでしょうか。いずれにせよ，複数の種類のファイルをやり取りしながら進めるので，相互の関係についてしっかり理解しておく必要があります。

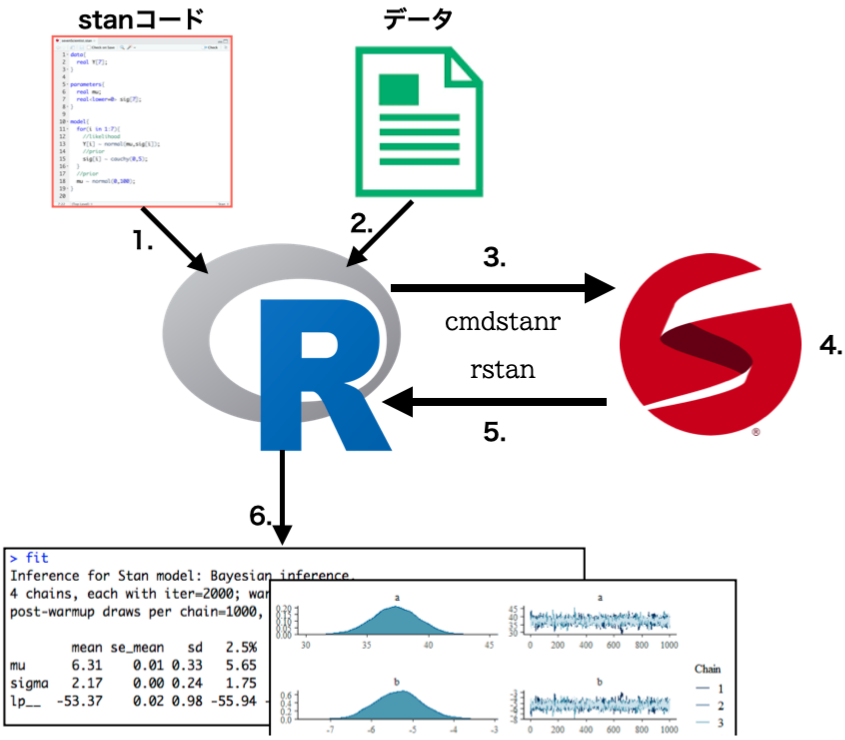


図 18.2 R/RStudio と Stan ファイルのやり取り

*12 このように，Stan は独立した計算環境であり，R 以外のアプリケーションから呼び出して使うことが可能です。Python から使いたい場合は，PyStan というパッケージ経由で，Julia から使いたい場合は StanJl, コマンドラインから使いたい場合は cmdstan といったように，いろいろなルートがあります。

18.2.3　2 つのルート

ここでは R から Stan を呼び出す 2 つのルート，具体的には `rstan` パッケージと `cmdstanr` パッケージについて説明します。この 2 つのパッケージは，いずれも Stan を呼び出してつかう，R と Stan の間を取り持つインターフェイスの役割を持ったパッケージです。間を取り持つだけなので，Stan ファイル自体は同じであっても構いません。同じ Stan ファイルを `rstan` から呼んでも，`cmdstanr` から呼んでも，結果は同じです。ただしインターフェイスが違うので，結果の扱い方が変わってきます。他にも違うところがいくつかありますので順に説明していきますが，結論からいうとこれから[*13]は `cmdstanr` のほうを使うほうがお勧めです。

歴史的には `rstan` のほうが古くから存在します。このパッケージも最初の頃は導入に一苦労するものでしたが，2016 年ごろから CRAN を通じて配布されるようになりました。要するに，他のパッケージと同じように，`install.packages("rstan")` と R で書くだけで導入できるようになったのです。このようにして導入するとわかりますが，`rstan` は多くの依存パッケージがあります。Stan はコンパイルするのに OS に応じた C++ コンパイラを借用しますが，R から C++ を呼び出すパッケージや R と C++ の間に入る StanHeader と呼ばれる緩衝材など，関連する多くの環境を整えてやっと R から Stan へのルートが通じるのです。ユーザにとってはそのような苦労は知ったことではない，という感じですが，「間に多くの調整が入る」ということがエラーの温床になっていました。つまり，いくつかの内部的ステップのどこかでバグがある，互換性がない，というようなことがあると，Stan ファイルが同じでも，バージョンが変わるとコンパイルできないということがよくあったのです。Stan 本体の方も開発が盛んですから，Stan の新しいバージョン，新しい機能を使いたいと思っても，パッケージが対応するのを待たなければなりません。パッケージが対応しても，途中の媒介パッケージが追いついていなければバグになったりします。これらを使う R や OS もアップデートしていきますから，その度に「動かなくなる」ということはよくありました。研究の再現性が問題になる昨今ですが，自分の研究とデータであっても，自分の環境で再現できないということが少なくなかったわけです[*14]。そうなると，今動いているからそれでいい，と環境のアップデートを控えるようになりますが，ご存知の通り OS やアプリのアップデートはバグやセキュリティホールの修正など，安全につかうための基本的な機能に関わってきますから放置もできない，という問題があるわけです[*15]。

このような不安定な環境になる原因は，R と Stan の間にあるさまざまな障壁と，それをなんとかして調整しようという媒介パッケージの存在でした。そこでなるべくシンプルに R と Stan をやりとりする方法はないものか，ということで考えられたのが `cmdstanr` です。これはコマンドライン[*16]から Stan を呼び出す `cmdstan` を R から呼び出して使おう，というものです。Stan を R になんとか取り込んで動かす，というのではなく，Stan の部分は OS のプリミティブな環境にお任せ，R は結果をもらうだけ，というシンプルな設計にしようというわけです。たとえば `rstan` は Stan の計算結果を R オブジェクトとして取り込みますが，`cmdstanr` は実は MCMC の結果は csv ファイルのような形で吐き出しており，それを読み込んで使います。御用聞きや仲介業者を介して情報をもらうのではなく，現地で直接買い付けしてくるようなイメージでしょうか。このよう

---

*13 執筆時点，2022-11-14 現在

*14 実際 R や OS のアップデートに関わるエラーは大変で，筆者は一度 OS をアップデートした後半年ほど Stan が使える環境が作れず，研究が頓挫したことがあります。そのために別途，PC を購入し直したぐらいです。

*15 OS のアップデートなど，通知が来ても「今支えているからいいや」と無視する人もいますが，できれば OS やアプリは最新版であるほうが良いのです。セキュリティホールの問題などを放置しておくと，インターネットを介して自分の PC の中身が全部見られて公開される危険性があります。Stan しか使わないネットに繋がない PC というのがあればいいのかもしれませんが，文房具としての PC はもっと一般的な用途がありますから，そうもいってられませんよね。また，一昔前の OS アップデートは，アップデートの失敗で PC が動かなくなるというようなこともありましたから，アップデートが公開されてすぐに対応するのは「人柱」などと揶揄されていましたが，最近はそういったこともほとんどありませんので，安心して常に最新の状態にしておいてください。

*16 MacOS や Linux では端末，ターミナル，Windows ではコマンドプロンプトなどと呼ばれる，PC に直接コマンドで命令を出すプリミティブな実行環境です。

に間に挟むものを減らすことによって，不安定な要素をなくしたわけです。

cmdstanr は Stan と R の複雑な絡まりを排していますので，Stan の開発が進めばすぐにそれを R に届けることができます。たとえば 2022/11/14 現在で，公式にリリースされている rstan のバージョンは 2.21.1 ですが，cmdstanr が呼び出す stan のバージョンは 2.30.1 です。cmdstanr のほうが新しいものに対応できていますね。このように，cmdstanr がでてきてからはこちらの対応，反応が早く，また動作も安定的ですから，よりお勧めしやすくなっています。

これまでに出ている R で Stan を使う方法を解説したテキストの多くは，rstan をつかっていますので，今それを使うためには多少の読み替えが必要です。とはいえ，Stan のコード自体はそのまま使えるものがほとんどで，R とのインターフェイス，やりとりの方法が違っているだけです。cmdstanr には，結果を rstan で得られたオブジェクトに変換する関数も含まれていますから，これらを使って変換すれば，以後のコードは rstan 準拠のものでもエラーになることはありません。このテキストでは第 2 版から cmdstanr を中心にするように切り替えましたが，以前のコードでも本質的に問題はありません。

18.3 導入の概略

環境の導入は，OS の種類によって変わります。Widows なのか Mac なのか，あるいは Linux なのかといった違いに合わせて導入を考えてください。OS の種類で言えば，Linux は全体的に CUI 操作が主であり，堅牢かつ合理的な使い方ができるものです。如何せんマイノリティですので，ググってもあまり情報が出てこないところが玉に瑕です。Mac は Linux ベースの OS になっているので，比較的堅牢かつ合理的な使い方ができます。ハードウェアも OS も 1 つの会社 (Apple) が作っているので，サポートもしやすいという利点がありますが，日本では少数派なのが残念なところです。Windows は多数派の OS ですが，OS メーカはソフトウェア会社でハードウェアが別会社なことが多く，利用される範囲は広いのですが，その分問題が生じたときにケースバイケースになりがちです。多くのハードウェアの違いを吸収するために最大公約数的な設計をするせいか，ソフトウェアデザインの統一性がないところが欠点です。ユーザ数は最も多いので，一生懸命調べたら同じような問題で困っている同じような機体の人に出会える可能性が高いのがせめてもの救いです。また光明として WindowsOS も Linux ベースの仕組みを取り入れ始め (WLS)，比較的合理的な振る舞いができるようになったことが挙げられますが，OS のグレードによって導入のしやすさが異なります。

ちなみに ChromeOS や iOS, Android などタブレット・スマートフォンでの統計環境の利用は限界があります。これらは計算結果を利用するフロントエンドとしては非常に高機能なのですが，インタラクティブに使うことには不向きです。いわば書籍のように多くの情報を一方的に与えてはくれるのですが，ユーザが計算するといった働きかけの補助になるような (ノートとペンのような) 使い方ができませんので注意してください。

さて，環境の導入として 2 つのルートを紹介しましょう。第一のものが最も基本的なアプローチで，自分の手元の環境を作り上げるというものです。これを**ローカル環境**での構築と呼びます。ローカル環境は自分だけの環境ということなので，自分の責任でもってしっかりと環境構築をできます。第二のアプローチとして，ある程度できあがった環境を丸ごと取り込む，**仮想環境**での構築という方法があります。PC の OS の中に別のマシン・別の OS をさらに憑依させて使うようなやりかたで，これができると取り込む環境は完成しているので細かい設定をする必要がありません。問題は「憑依させる」準備がいろいろ大変であるということです。また憑依させた PC はブラウザ経由でアクセスすることになることにも注意してください。

18.3.1 導入方法 1；ローカル環境での構築

ローカル環境での構築は，R，RStudio の導入から入ります。ここは既にできている人もいるかもしれませんが，念のためバージョンが最新のものかどうかを確かめておいてください。ローカル環境構築の方法がわからない人は，「RStudio」「インストール」などのキーワードでウェブ検索し，自分の環境にあった例を見つけてそれをみながら進めると良いでしょう。参考までにいくつかあげておきますと，新しいものでは，高知工科大学柳井先生のこちら http://yukiyanai.github.io/jp/resources/ とか，Quiita の記事 https://qiita.com/hujuu/items/ddd66ae8e6f3f989f2c0 が良いでしょう。書籍では，手前味噌で恐縮ですが小杉 (2019a) などが良いでしょう[*17]。

次に確率的プログラミング言語，Stan の導入を行います。Stan を R から使うためのパッケージとして `cmdstanr` と `rstan` という 2 種類があるという話はしました。かつては `rstan` のほうがよく使われていたのですが，最近は `cmdstanr` のほうが安定的に動作し，開発も盛んになっているので，2022 年度後期からは `cmdstanr` をこの授業のデフォルトパッケージとしたいと思います。このパッケージの導入には，公式サイト https://mc-stan.org/cmdstanr/articles/cmdstanr.html を参考にしましょう。「cmdstanr インストール」で検索すると色々な紹介サイトが出てきますので，自分と同じ環境でなるべく日付の新しいものから参考にしていくといいでしょう。

インストールにあたっては，Stan がコンパイルされるときに利用する C++ 言語環境が必要です。これは MacOS であれば Xcode の導入が，Windows であれば Rtools での導入が必要になります。

■MacOS ユーザの場合　AppStore で Xcode と検索し，Xcode をインストールすれば OK です。

■Windows ユーザの場合　Rtools という R 周辺のツールをインストールする必要があります。https://cran.r-project.org/bin/windows/Rtools/ にいって，自分の R のバージョンにあった RTools をインストールしてください。この他にも Windows ユーザは導入にあたって色々障壁にあたることがありますので，こちらのサイト https://norimune.net/3609 なども参考にしてみてください。

18.3.2 導入方法 2；仮想環境での構築

第二のルート，仮想環境を構築する場合ですが，これについては我らが国里先生が大変詳しいサイトを作ってくださっています。日本心理学会のチュートリアルワークショップ，「再現可能な日本語論文執筆入門：jpaRmd で実現する再現可能で低コストな日本語論文執筆のはじめの一歩」で使われた時の資料集がそれで，このサイト https://ykunisato.github.io/jpa2021-tws-jpaRmd/ の事前準備編をよく読むと導入できます。通信環境にもよりますが，慣れているとものの 10 数秒でシステム導入できるという優れものです。面倒な設定が怖い人は，是非試してみてください。

18.4 導入方法 3；外部サーバの利用

最近，Google 社がオンラインで利用できる分析環境，Google Colaboratory を提供してくれています。これは Google 社が提供する，ブラウザで実行できるプログラミング環境です。Google が教育，研究用に提供しているもので，90 分間は無料で利用できます。90 分を超える場合でも，書いたプログラムを保存して

[*17] インストールに際しては，RStudio Desktop の Free edition を選んでください。RStudio Cloud や RStudio Server は別のものです。

おけば，新しいセッションを開始することで続けて利用できます。これを用いる利点は，どのユーザにも同じ環境を提供できることです。

基本言語環境は Python で提供されており，Jupyter Notebook を使いますが，次のアドレス (に含まれるコード) を使えば言語環境を R に変えて利用することができます。

https://colab.research.google.com/notebook#create=true&language=r

ここでウィンドウに R のコードを入力し，実行していきます。cmdstanr や rstan もインストールできます。インストールに際して，install.packages("rstan") のように入力することもできますが，この方法ですとインストールに随分と時間がかかります。そこで少し例外的ですが，システムコマンドを直接利用して，system("apt install -y r-cran-rstan") のようにすることで，大幅にスピードを短縮することができます。

18.4.1 導入後の確認

インストールが終わったらサンプルコードを実行して「動くかどうか」を確かめてみてください。サンプルコードは Getting started with CmdStanR のサイト，あるいは RStan Getting Started のサイトに掲載されています。どちらのパッケージから実行しても，文字列がズラズラっと出力されればとりあえず動いたと思っていただいて結構です。赤い文字やエラーなどが表示されうる，あるいは全く表示されない場合は，インストールがうまくいっていないことを疑いましょう。上に戻って，丁寧に説明に目を通して一歩ずつ進めてください。機械に命令することですので，自分勝手にショートカットしたり変更したりしないことが重要です。

18.5 Stan を使ってみよう

具体的な Stan コードの書き方や統計モデルへの応用については，本書の次の章から順に説明していきます。それに先立って，全体的な注意をしておきたいと思います。具体的には，バージョンによる違い，パッケージによる違い，そして本書の以下の章で使う関数の紹介です。

18.5.1 バージョンによる書き方の違い

cmdstanr と rstan はパッケージの使い方以上に，それらが呼び出す Stan のバージョンの違いがあります。上で解説したように，cmdstanr のほうが最近は開発が進んでいて，より新しい Stan の機能を使っていることになります。バージョンが違っても，基本的な書き方などに違いはないのですが，バージョン 2.26 から Stan の文法に変更が予定され，導入されました。

データ X が N 人分あったとします。数学的フォームでは，$X_1, X_2, X_3, \cdots, X_N$ というようなものですね。これはコンピュータ上では X と名付けられた配列，サイズ N というように考えます。これは，Stan では次のように書くものでした。

code : 18.1 Stan2.26 以前の書き方

```
int n[5];
real a[3, 4];
real<lower=0> z[5, 4, 2];
vector[7] mu[3];
```

これらは上から順に，サイズ 5 の整数変数 n，サイズ 3×4 の実数変数 a，下限 0 のサイズ $5 \rightarrow, es4 \times 2$

の配列 z，長さ 7 のベクトル mu が 3 つ，ということを意味しています[18]。

この書き方が，Stan のバージョン 3.32 以降は次のような書き方になります。

code : 18.2　Stan2.32 以降の書き方

```
array[5] int n;
array[3, 4] real a;
array[5, 4, 2] real<lower=0> z;
array[3] vector[7] mu;
```

このコード 18.2 はさきほどのコード 18.1 と同じ意味で，書き方が変わっただけです。一瞥してわかるように，サイズを先に array という言葉で宣言しましょう，というだけですね。

さて，今現在は Stan2.26 と 3.32 の間，過渡期になります。ですから，どちらのコードで書いてもエラーにはなりません。ただし，(ここからが少し面倒なのですが)

- RStudio のコードチェック機能は新しいコードの書き方に対応しておらず[19]，新しい書き方をするとエディタ上でエラーの警告 (赤いバッテンがつきます) が出ますし，チェックボタンを押しても SYNTAX ERROR が出ます。
- rstan パッケージは使っている Stan が古いこともあって，新しい書き方のコードをコンパイルしようとするとエラーになります。RStudio と同じ，SYNTAX ERROR が出ます。
- cmdstanr パッケージは逆に，新しい Stan の書き方を推奨していますので，「その書き方は古いよ」という警告を出してきます。Declaration of arrays by placing brackets after a variable name is deprecated and will be removed in Stan 2.32.0. Instead use the array keyword before the type. This can be changed automatically using the auto-format flag to stanc. という警告がでますが，これは「配列のカッコを変数の後ろに置く書き方はもうダメで，Stan2.32 以降は無くなります。型の前に array と書くようにしてください。これは auto-format フラグを立てることで自動的に変更するようになります。」という意味です。

つまり，本書は cmdstanr を推奨していますが，そうすると RStudio のエディタ上ではエラーが出て文法チェックができず，コンパイルした後でいざサンプリング，というときにエラーが出たりします。チェックをするには，コンパイルしたオブジェクトを使ってサンプリングをする前に，model$check_syntax() 関数を実行する必要があります。rstan パッケージを使うとスマートに文法チェックもできていいのですが，rstan そのものが不安定ですし，今後徐々に開発・発展をしなくなっていくことが明らかですので，どこかで新しい方に舵を切る必要があります。

本書のコードは新しい書き方に統一していますが，最初のうちは，そして今しばらくは，実際にコードを書くときは古い方で書いたほうが便利かもしれません。

18.5.2　パッケージによる指示と出力の違い

さて今度はパッケージごとの違いを見ていきましょう。

[18] ベクトルは数字をセット，人まとめとして演算するものです。最後の変数は，7 つの数字のセットが 3 つという意味で，7 つセットで 1 つのまとまりですよ，ということを明示的に宣言していることになります。

[19] RStudio のバージョン 2022.07.2 Build 576 で確認しました。

指示の仕方の違い

まずは実行方法です。Stan では事後分布からの乱数を生成しますが，それにあたって「乱数の数」「ウォームアップ期間の長さ」「チェインの数」などオプショナルに指定できるものがあります。これは見てもらったほうが早いかと思いますので，同じことをそれぞれのパッケージで実行するときにどのように指定方法を変えるかを見てみましょう。まずは `rstan` パッケージの書き方からです。

code : 18.3 rstan のスタイル

```
fit <- rstan::sampling(model,
  data = dataSet,
  chains = 4,
  iter = 6000,
  warmup = 1000
)
```

ここで指定しているのは，データを `dataSet` として設定し，同時に 4 本のチェインを発生させ，6000 個の乱数を作って，そのうち 1000 個はまだ機械が温まっていないので候補から除外する，というものです。複数のチェインを作ること，温まっていない部分を捨てる理由などは後ほど説明しますが，結果的に合計 20000 個の乱数が作られることを知っておいてください。この数字は $(6000-1000)\times 4 = 20000$ という計算からでてきます。また，乱数の発生は並列計算させた方が良いので，`rstan` パッケージを使う場合は事前に，`options(mc.cores = parallel::detectCores())` という一行を入れておきます。

さて，同じことを `cmdstanr` パッケージでやると次のようになります。

code : 18.4 cmdstanr のスタイル

```
fit <- model$sample(
  data = dataSet,
  chains = 4,
  parallel_chains = 4
  iter_sampling = 5000,
  iter_warmup = 1000,
)
```

ここにあるように，並列するチェインの数をオプションに書き込む必要があります。またサンプルの数は捨てる部分を別にして (含めずに) 計算させることができます。このコードで 20000 個のサンプルが得られます。細かいことですが，多少の違いがあるわけです。

出力の違い

さて，今度は出力結果の使い方についての注意です。`rstan` パッケージは，出力結果を `stanfit` オブジェクト型という形にします。`rstan` のもっているさまざまな関数，たとえば要約した結果の出力やプロットなどは，`stanfit` オブジェクトだとわかるとそれに対応した出力に変えて表現してくれるわけです。`cmdstanr` の出力はまた別物[20]です。

[20] `class` 関数で `rstan` の出力がどういうクラスなのかを表示させると，`stanfit` と出てくるのですが，`cmdstanr` の出力を同様にチェックすると`"CmdStanMCMC" "CmdStanFit" "R6"`という答えが返ってきます。クラスというのはデータの形を定義する方法で，オブジェクト指向プログラミングに必要な概念なのですが，ここではややこしすぎるのでパス。違うということだけわかってもらえれば結構です。RStudio のばあいは Environment タブを見るだけでも違いがわかると思います。`rstan` パッケージの出力はちゃんとしたクラス (Formla Class stanmodel) なのですが，`cmdsatnar` パッケージの出力は Environment(グローバル環境の要素) として剥き出しの出力が出ている感じです。

同じモデルをそれぞれのパッケージで出力させて，違いを見てみましょう。まずは rstan パッケージの出力から。

rstan の出力 1: rstan パッケージの出力

```
Inference for Stan model: ttest01.
4 chains, each with iter=6000; warmup=1000; thin=1;
post-warmup draws per chain=5000, total post-warmup draws=20000.

       mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
mu1   59.96    0.04 5.58  49.01  56.33  59.99  63.60  70.91 16043    1
mu2   40.04    0.05 5.62  28.98  36.38  40.03  43.67  51.17 14330    1
sig   15.54    0.03 3.12  10.86  13.35  15.08  17.23  22.92 12118    1
lp__ -51.58    0.02 1.36 -55.12 -52.18 -51.24 -50.60 -50.05  7115    1

Samples were drawn using NUTS(diag_e) at Tue Nov 15 09:50:17 2022.
For each parameter, n_eff  a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

rstan パッケージが出力しているのは次のような情報です。

- モデル名，チェイン数，反復回数，最終的に得られた MCMC サンプル数 (draws) などの説明
- パラメータの事後分布の記述統計量，すなわちベイズ推定による事後分布の情報。順に平均値 (mean)，平均値の標準誤差 (se_mean)，標準偏差 (sd)，2.5%,25%,50%,75%,97.5% パーセンタイル。
- MCMC サンプルについての特徴量。有効サンプルサイズ (n_eff) と Rhat(Rhat)
- サンプリングに関するあとがき

それぞれの内容についてはあとの章で説明するとして，続けて cmdstanr パッケージの出力をみてみます。

cmdstanr の出力 1: cmdstanr パッケージの出力

```
variable   mean median   sd  mad     q5    q95 rhat ess_bulk ess_tail
lp__ -51.56 -51.20 1.34 1.07 -54.18 -50.12 1.00     7799    10071
mu1   59.99  60.00 5.63 5.47  50.82  69.16 1.00    16325    12764
mu2   40.05  40.05 5.48 5.18  31.09  49.05 1.00    16511    12466
sig   15.48  15.03 3.07 2.75  11.41  21.07 1.00    14489    12137
```

同じような情報ですが，ちょっと違うところもあるようです。前から順に，事後分布の平均値 (mean)，中央値 (median)，標準偏差 (sd)，平均絶対偏差[*21](mad)，5%,95% パーセンタイルなど分布の特徴[*22]と，Rhat や有効サンプルサイズに関する情報 (ess_bulk,ess_tail) が示されています。しかし，まえがき・あとがきのようなものがなく，結果だけがシンプルに出力されていますね。

[*21] **中央値絶対偏差 (Median Absolute Deviation)** とは，中央力のばらつきを表す指標で，各データ点の中央値からの偏差の絶対値をとり，その中央値を計算したものです。標準偏差とは違う分布の幅を表現する方法で，中央値を使って計算しますから外れ値に強いという特徴があります。

[*22] rstan パッケージは 2.5% から 97.5% のパーセンタイル，すなわち囲まれる区間の面積が 95% です。cmdstanr パッケージは 5% と 95% のパーセンタイル，すなわち囲まれる区間の面積が 90% です。なぜだかわかりませんが，ちょっとした違いがありますね。

出力としては rstan パッケージの方が丁寧で良いかもしれません。さて，cmdstanr の出力は実は csv ファイルとして一時フォルダに保存されているので，これを rstan パッケージの関数を使って stanfit オブジェクトに変換することができます。関数の使い方の例は次のようになります。

code : 18.5　cmdstanr の出力を rstan のオブジェクトに変える

```
fitR2 <- fitC$output_files() |> rstan::read_stan_csv()
```

コードの中身ですが，まず fitC と書いてあるのが cmdstanr の出力オブジェクトです。このオブジェクトがさまざまな情報を持っているという形になっており，ここから出力ファイルの場所を聞き出すのが output_files() です。このファイルを rstan パッケージの read_stan_csv() 関数に渡してやると，csv ファイルを読み込んで rstan のオブジェクトに変えてくれます[*23]。変えたものをここでは，fitR2 オブジェクトに保存しています。この方法を使うと，先ほどの cmdstanr の出力が rstan の出力と同じになります (同じなので再掲しません。)。

このコードを覚えておけば，rstan パッケージ向けに書かれたものであっても，cmdstanr パッケージで実行したあと同じ関数を適用できるので，使いやすいですね。

18.5.3　本書で利用する準備関数

どちらのパッケージであっても，欲しいものは事後分布からの乱数をデータセットにしたものであり，そのデータセットさえ持っていれば，既存の関数を使わなくとも自分で工夫して加工すれば良いでしょう。

そこで本書では，MCMC サンプルを抜きだしてデータセットに変換する関数を作り，これを利用して話を進めることにします。ここではその関数の中身を解説しておきます。

MCMC サンプルをデータフレームにする関数

まずは MCMC サンプルをデータセットにして渡してくれる関数です。この関数は伴走サイトのコードの冒頭に必ず含まれており，以後はこれを使って結果の要約を示すといった使い方をしていますので，中身を知っておいてもらった方が良いかもしれません。自分の使っているパッケージの方だけでも目を通してください。この関数を経由すると，出てくるオブジェクトは同じ形式になっているところがミソです。

■stanfit オブジェクト (rstan パッケージ) の場合　rstan パッケージを使って stanfit オブジェクトとして MCMC サンプルを得た場合，そのオブジェクトから乱数部分だけを抜き出す関数は rstan::extract 関数です。これで取り出したものをデータフレーム型にして返す関数として，次のようなものを作りました。

code : 18.6　MCMCtoDF 関数 (rstan 版)

```
MCMCtoDF <- function(fit) {
  fit %>%
    rstan::extract() %>%
    as.data.frame() %>%
    tibble::as_tibble() %>%
    tibble::rowid_to_column("iter") %>%
    dplyr::select(-lp__) %>%
    tidyr::pivot_longer(-iter) -> MCMCsample
  return(MCMCsample)
```

[*23] 当然のことながら，rstan パッケージがないとこの関数も呼び出せませんので，こちらもインストールしておく必要があります。なおこのコードの，|>は tidyverse パッケージのパイプ演算子，%>% と同じ機能を持つ R の演算子で，ネイティブパイプと呼ばれています。ネイティブパイプは R4.1.0 以降に導入されたものです。

```
10  }
```

■コード解説

1 行目　関数を作る宣言。引数として stanfit オブジェクトをとります。

2 行目　引数で引き受けたオブジェクトを加工していきます。

3 行目　まずは MCMC サンプルを抜き出します。

4 行目　data.frame 型にします。

5 行目　tibble 型にします。別にしなくてもいいんですが，著者の好みです。

6 行目　行番号を表す変数 iter を作ります。

7 行目　変数の中から lp__を除外します。

8 行目　行番号変数は残して，tidy なデータにし，MCMCsample というオブジェクトに保存します

9 行目　戻り値として MCMCsample を返します。

■cmdstanr パッケージの場合　cmdstanr パッケージを使って MCMC サンプルを得た場合，そのオブジェクトから乱数部分だけを抜き出す関数は draws 関数であり，これをオブジェクトに直接作用させます。これで取り出したものをデータフレーム型にして返す関数として，次のようなものを作りました。

code : 18.7　MCMCtoDF 関数 (cmdstanr 版)

```
1  MCMCtoDF <- function(fit) {
2    fit$draws() %>%
3      posterior::as_draws_df() %>%
4      tibble::as_tibble() %>%
5      dplyr::select(-lp__, -.draw, -.chain, -.iteration) %>%
6      tibble::rowid_to_column("iter") %>%
7      tidyr::pivot_longer(-iter) -> MCMCsample
8    return(MCMCsample)
9  }
```

■コード解説

1 行目　関数を作る宣言。引数として stanfit オブジェクトをとります。

2 行目　引数で引き受けたオブジェクトに draws 関数を作用させ，サンプルだけ取り出します。

3 行目　取り出したサンプルをデータフレーム型にする posterior パッケージの as_draws_df 関数を適用します。

4 行目　tibble 型にします。別にしなくてもいいんですが，著者の好みです。

5 行目　変数の中から lp__や他の隠し変数を除外します。

6 行目　行番号を表す変数 iter を作ります。

7 行目　行番号変数は残して，tidy なデータにし，MCMCsample というオブジェクトに保存します

8 行目　戻り値として MCMCsample を返します。

■MCMCtoDF 関数の結果　出力結果は，どちらも同じく以下のような形式になります。

R の出力 18.1: MCMC サンプルをデータセットにしたもの

```
# A tibble: 60,000 × 3
    iter name  value
   <int> <chr> <dbl>
 1     1 mu1    60.3
 2     1 mu2    41.0
 3     1 sig    12.5
 4     2 mu1    62.9
 5     2 mu2    37.6
 6     2 sig    11.5
 7     3 mu1    68.7
 8     3 mu2    46.4
 9     3 sig    13.7
10     4 mu1    67.7
# … with 59,990 more rows
# ☒ Use `print(n = ...)` to see more rows
```

ここで `iter` とあるのは MCMC のステップ番号，`name` とあるのが変数名，`value` とあるのがその値です。tidy な形になっていますから，このまま分析や描画に用いることができます。

MCMC サンプルの結果を要約する関数

MCMC の結果を分析するにあたっては，`bayestestR` パッケージなど既存のパッケージを使うと便利でしょう。ですが，こうしたパッケージはどんどん開発が進んでアップデートされたり，`rstan` か `cmdstanr` かで書き方が変わったりするなど，テキストで紹介するには難しいところがあります。そもそも MCMC サンプルを加工して記述統計を示したり，描画したりするわけですから，上で紹介したように MCMC サンプルを抜き出すことができれば自力でもできるはず。ということで，自作の関数を用意しました。本書では以下，この関数をつかって解説しますので中身を知っておくと良いでしょう。

■MAP 推定関数　確率分布の特徴を報告するときに，その確率分布に従う乱数を使って，期待値や中央値を計算するのは簡単です。記述統計としての平均値やパーセンタイルでよいからです。しかし確率分布の確率密度が最も高くなるところ，すなわち **MAP 推定値 (MAP Estimaiton)** の計算はちょっと難しいですね。というのも，関数であれば微分して極値を求めれば良いのですが，MCMC サンプルは関数ではなくて値なので，ヒストグラムを書くしかなく，最大の値を求める計算がむずかしいからです。

でも大丈夫，R の描画関数を駆使して対応することができます。MAP 推定値を計算する関数は，次のように書きます[*24]

code : 18.8　MAP 推定する関数のコード

```
map_estimation <- function(z) {
  density(z)$x[which.max(density(z)$y)]
}
```

これは R の `density` 関数でヒストグラムに密度関数を当てがい (カーネル密度推定)，その関数の特徴から値を返すようになっています。

*24 このコードは関西学院大学社会学部の清水裕士先生に教えてもらったものです。記して謝意を表します。

■MCMC サンプルの要約関数　先ほどの MAP 推定する関数を含めつつ, また MCMC サンプルをデータフレームにする関数を使いつつ，MCMC サンプルの要約を表示する関数を次のように作りました。

code : 18.9　MCMC の要約を報告するコード

```
1  MCMCsummary <- function(MCMCsample) {
2    MCMCsample %>%
3      dplyr::group_by(name) %>%
4      dplyr::summarise(
5        EAP = mean(value),
6        MED = median(value),
7        MAP = map_estimation(value),
8        SD = sd(value),
9        U95 = quantile(value, prob = 0.975),
10       L95 = quantile(value, prob = 0.025)
11     ) %>%
12     mutate(across(where(is.numeric), ~ num(., digits = 3)))
13 }
```

■コード解説

1 行目　関数を作る宣言。引数として `MCMCtoDF` 関数の戻り値をとります。

2 行目　引数で引き受けたオブジェクトを加工していきます。

3 行目　変数名でデータをグループ化します。

4-10 行目　記述統計の計算を通じて，EAP,MED,MAP,SD,95% 区間を返します。

11　　行目 出力する数値データは, 少数下 3 桁まで表示させるようにします。

この関数は, 先ほどの `MCMCtoDF` とセットで使い, 次のような結果を得ます。

R の出力 18.2: MCMCsummary の結果

```
> fit %>% MCMCtoDF() %>% MCMCsummary()
# A tibble: 3 × 7
  name         EAP        MED        MAP         SD        L95        U95
  <chr> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 mu1       59.897     59.888     58.867      5.547     48.672     70.847
2 mu2       39.958     39.935     39.748      5.523     28.973     50.943
3 sig       15.519     15.062     14.646      3.074     10.877     22.788
```

ここにあるように, 各変数の EAP 推定値，MED 推定値，MAP 推定値，95% 確信区間が表示されます[*25]。

*25 この区間はコードから明らかなように，`quantile` 関数で計算しています。下から 2.5%, 上から 2.5% を取り除くと 95% の区間が残ることになりますが, このように分布の両端を均等に切った区間を厳密には**等裾区間 (Equal-tailed interval;ETI)** というのでした。これに対して, 分布の最も信頼できる部分, かつ分布の大部分をカバーし, 区間内部のすべての点が, 区間外の任意の点よりも高い密度を持っているように推定するものを**最高密度区間 (Highest-Density Interval;HDI)** と呼びます。これらは左右対称の単峰分布であれば同じになるのですが, 事後分布の形がぐちゃっとしている時は必ずしもそうなりません。両者の違いについては Kruschke (2014) を参考にしてください。また，HDI を出力する関数は `bayestestR::hdi()` です。

これらの推定値や，そもそも事後分布が何をどのようなことを表しているのか，といった点については今後の授業で説明していきます。以下は結果をしれっとこの関数で表現していきますので，何をやっているのかわからなくなったらここに立ち戻ってきてください。

第 19 章

ベイジアンアプローチと確率的プログラミング 1

それではいよいよ，**データ生成モデリング**とベイズ推定による実践例を見ていくことにしましょう。

19.1　7 人の科学者

ここでは Lee and Wagenmakers (2013) より「7 人の科学者」の例を紹介します。そこでのカバーストーリーは次のようなものです。

> 実験スキルが大きく異なる 7 人の科学者が，全員同じ量について測定を行う。彼らが得た数値結果は次の通り。
> $$Y = \{-27.020, 3.570, 8.191, 9.808, 9.603, 9.945, 10.056\}$$
> 直感的には，最初の二人の科学者はひどく適性を欠いた測定者であり，この量の真の値はおそらく 10 をわずかに下回るくらいであるように思えるのだが・・・？？

さあ，これをみて「あれ，どこが統計的な問題なんだ？」と思った人もいるかもしれません。なんらかの測定をして，データのばらつきがあるんだけど，それが測定者によって違うらしい，というのはわかったかと思いますが，これをどうやって統計的な問いにするのでしょうか。まず，ストーリーの背後に「正確な測定値 (真の値) はわからないけど，定数のはず」という前提があることを確認しましょう。測定の基本として測定誤差があるので真の値を特定出来はしませんが，測定を繰り返すと真の値に近づいていくはずではあります。また，今回の測定結果は 7 人それぞれ別の人による測定であることから，測定誤差の出方が測定毎すなわち測定者毎に異なるだろう，という仮説もあります。

これらを踏まえて謎解きをしていきましょう。わからない量を確率の言葉で表現し，データを使って少しでも正解に近づこうとするのが統計のやり方です。まずここでわからない量は，「正確な測定値」と「個人毎の誤差の大きさ」です。誤差は正規分布に従うと思われますから，データ Y は $Y \sim N(\mu, \sigma)$ と表すことができます。データ生成メカニズムという意味では，「正規分布に従ってデータが作られる」と言ってもいいかもしれません。また，正確な測定値はここでは μ ということになります[*1]。また，データは 7 点あって，それぞれ $Y_1, Y_2, \cdots, Y_7$，あるいは一般に Y_i と書くことにしましょう。ここで添字の i は第 i 回目の測定でもありますし，測定者 i のことでもあります。この測定者 i 毎に誤差の出方の大きさが変わります。誤差の大小，言い換

[*1] 正確な測定値 μ に誤差 $N(0, \sigma)$ がついてデータになる，すなわち $Y = \mu + N(0, \sigma)$ ということですが，誤差の平均は 0 で μ の分だけ正規分布がズレる，そして「分布に従う」を表現する $\sim$ を使いたいので，$Y \sim N(\mu, \sigma)$ となるのです。

えれば測定の精度は誤差の SD で表現されていますから，σ が個人毎に変わる，すなわち σ_i と書くことができます。これが今回のデータ生成モデルです！

ここまでのことをイメージ図であらわすと図 19.1 のようになります。この図の書き方のポイントは，まずデータを一番下に置くことです。次にこのデータがどこから来たのかな，ということを矢印で表現して書きます。データは誤差を伴うなどで，毎回定数にならないでしょうから，**確率分布**をその上に書くことになります。確率分布にはその形状を定めるパラメータがあるはずですから，それを書き加えれば OK です。

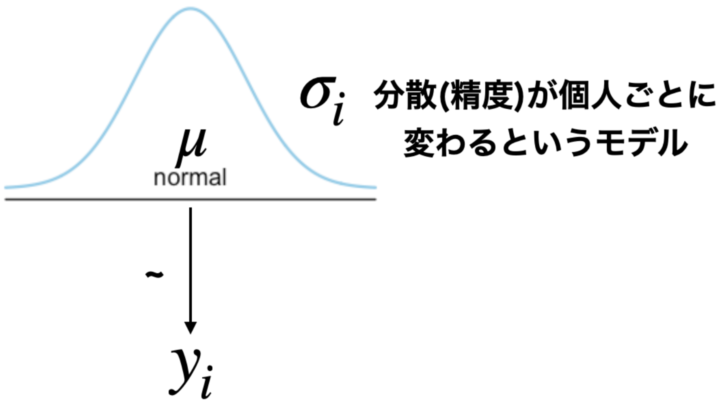


図 19.1　データが生成されるプロセス

しかしここでも，μ と σ_i は結局なんなんだ，どういう数字なんだということがわからないままです。わからない量は確率の言葉で表現するのがベイジアンの生き様。ここは未知のパラメータがどうやって出てくるかなんてわからないところなので，「わからない」ことを確率で表現する必要があります。パラメータに対して「この辺りにあるだろう」と事前に想定する確率分布のことを**事前分布 (prior distribution)** というのでした。

ここでは μ は平均 0，SD100 の正規分布からきていることにしましょう。正規分布というのは左右対称で平均値・中央値・最頻値が一致する単峰の分布です。これは正確な測定値を表しているのでしたから，なんらかの値を取るとしても 1 つの値だろう，というのは無理のない仮定でしょう。複数の値を取る可能性があるのなら多峰性の分布を仮定したらいいと思いますが，今回はそうではないだろう，と仮定するのです。平均 0 で SD100 ということは，-300 から +300 の範囲にある可能性が 99.7% だということでもあります[*2]。実際のデータが -27 から 10 ぐらいの範囲に入っているので，-300 から $+300$ の間というのも十分過ぎるぐらい広めの範囲にしてある数字と言えるでしょう。このように，うっすら情報を持っているけどほとんど意味がない程度に縛りをかけるのを，**弱情報事前分布 (weakly informative prior distribution)** と言います。たとえばこれが身長のデータであれば，± 3m の範囲は十分すぎ，なんなら負の数は取るはずがないので過剰な安全策を置いているとも言えます。それでも「なんらかの事前分布を持つのは，主観的な思い込みだ」というのであれば，$\pm\infty$ の一様分布を考えるか，ベイズ法をやめるかということになります。

次に σ_i です。これは標準偏差パラメータなので負の数を取ることはあり得ません。ここでは**半コーシー分布 (half-cauchy distribution)** を事前分布におくことにします。コーシー分布とは正規分布によくに

[*2] 正規分布は，平均 ±1SD の範囲に全体の 68% が，±2SD の範囲に 95% が，±3SD の範囲に 99.7 が入ることが理論的にわかっているのでした。わからない人は R で計算して確認すること！

た形ですが，正規分布より裾が重く[*3]，標準偏差 (分散) の事前分布に適していると言われている分布です (Gelman et al., 2006)[*4]。このコーシー分布は正規分布のように左右対称ですが，0 を中心に半分に折りたたんで正の値しか取らないように考えたのが半コーシー分布です。今回はスケールパラメータに 5 を置いていますが，とくにこの数字に深い意味はありません。気になるようでしたら 10 でも 100 でも変えていただいて結構です[*5]。

これらをおくことで，モデルの設計図ができあがりました (図 19.2)。あとはこれをもとに，**確率的プログラミング言語 (stocastic programming language)** に書き起こしていけば良いでしょう。

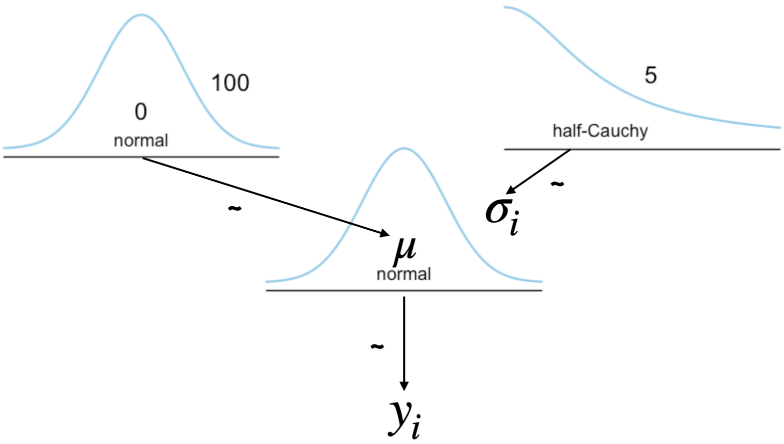


図 19.2　モデルの全体像

19.2　Stan コードの書き方

では図 19.2 という設計図に基づいて，Stan のコードを書いていきましょう。

19.2.1　Stan コードの文法

Stan は事後分布からの乱数を生成するための言語です。その言語仕様は C や C++ ににているところがあります。R で書くのと違うところとして「ブロック単位の記述」「セミコロンによる行の区切り」「変数の宣言」の 3 つを押さえておきましょう。

■ブロック単位の記述　Stan に書くべきことは**ブロック**の中に記述します。ブロックは中括弧 ({ ... }) で括ったもので，次の 6 つの種類があります。

1. `data` ブロック
2. `transformed data` ブロック
3. `parameters` ブロック

[*3] より極端な値が出る可能性が多いという意味です
[*4] この論文では他にも，student の t 分布などを提案していますが，パラメータ数がより少ないコーシー分布を選びました。
[*5] 半コーシー分布がどんな数字になるのか気になる，という人は R で `rcauchy` 関数を使って乱数を発生させ，その挙動を見てみればいいでしょう。乱数発生によって大体の分布感 (分布の感覚?) を得ることができる，ということを思い出してください。

4. `transformed parameters` ブロック
5. `model` ブロック
6. `generated quantities` ブロック

Stan のプログラムに含まれるブロックは，この順番でなければならないと決まっているので注意してください。この中で，1,3,5 番目，すなわち `data,parameters,model` ブロックが基本的に必要なもので，2.4.6 番目のブロックはその派生や応用です。まずは基本ブロックから説明します。

`data` ブロックは Stan と外部とのやりとりをするブロックです。外部からデータを取り込んで，それに基づいて乱数を発生させます。Stan で計算せずに外部から与えるものはすべてこのブロックの中で書く必要があります。`parameters` ブロックは，パラメータ，すなわち Stan で推定したいものを書くところです。Stan は事後分布からの乱数発生機であり，事後分布は**確率分布**ですからその形状はパラメータによって特徴づけられます。事後分布の形が分からないというのは，事後分布のパラメータがどのような値になっているのか分からない，ということと同じ意味です。その「分からない」「知りたい」パラメータをリストアップしておくブロックです。`model` ブロックは，確率モデルすなわち尤度と事前分布を書くところです。ここで書かれた尤度と事前分布が，`data` ブロックのデータと組み合わさって，事後分布が Stan 内部で計算され，そこからの乱数が次々取り出されてくることになるのです。

■セミコロンによる行の区切り　このブロックの中に，Stan 言語の文を書いていくことになるのですが，ここで 1 つ覚えておいて欲しいのが，**Stan の句点はセミコロン**ということです。句点がない文章は読みにくいどころか，機械にとっては意味不明な文字列になりかねませんので，必ず 1 つの命令・一文がおわったらセミコロンで閉じるようにしてください。これを忘れてエラーになるのが初心者にもっとも多いミスのひとつです。

■変数の宣言　たとえば R では数字でも分析結果でも，オブジェクトに代入でき，オブジェクト名は任意につけることができました。またオブジェクトはいつどのタイミングで発生させても OK で，思いついたらすぐオブジェクトに代入，ということができました。Stan をはじめ，いくつかの高級言語ではこれに厳格な制限をかけるものがあります。すなわち，どんなオブジェクトを作るかを最初に**宣言 (declaration)** する必要があるのです。Stan では使われるオブジェクト (変数) が，どういう数字になるものなのか，どの範囲の数字になるのかをブロックの最初に宣言します。この数字の種類のことを**型 (type)** といい，その宣言された数字の型に適合しない数字はエラーとなって弾かれます。なんでそんな窮屈な，と思うかもしれませんが，逆に「ありえない数字は許さない」という意味で，エラーを未然に防いでくれることにもなるのです。

変数の形として，Stan は整数 (**int 型**)，実数 (**real 型**)，ベクトル (**vector 型**)，行列 (**matrix 型**) などがあり，また範囲を `lower` や `upper` で指定します。たとえば標準偏差は負の数を取らないので，標準偏差の変数を宣言したいときは，`real<lower=0> sig;` のように書きます。

19.2.2　Stan コードを書いてみよう

されこれらを踏まえてコードを書いてみるわけですが，設計図を書いたときにデータを下に，それに紐づく確率分布を上に上に，と下から上に書いていったように，`model` ブロック →`parameters` ブロック →`data` ブロック，と書き進んでいったほうがわかりやすいでしょう。

Stan のコードはメモ帳など，エディタで開くことのできるテキストファイルでいいのですが，拡張子を `.stan` にしておきましょう。RStudio で書く場合は，`.stan` にしておくと Stan のファイルだと認識してくれるので，ブロックや型宣言の用語などを強調表示してくれるようになります。RStudio でコードを書き始めるときは，

ファイル (File) ＞新しいファイル (New File) と進むと Stan というファイルがあります。これを選ぶとすでに Stan のサンプルコードが書かれているファイルが出てきますが，そのファイルの中身は全部消して，次のコード 19.1 のように書き，ファイル名をつけて保存しましょう。ここでは sevenScientist.stan とファイル名をつけたものとします。

code : 19.1 7 人の科学者コード

```
data{
  array[7] real Y;
}

parameters{
  real mu;
  array[7] real<lower=0> sig;
}

model{
  for(i in 1:7){
    //likelihood
    Y[i] ~ normal(mu,sig[i]);
    //prior
    sig[i] ~ cauchy(0,5);
  }
  //prior
  mu ~ normal(0,100);
}
```

■コード解説

model **ブロック** 尤度と事前分布を書くところ。確率分布からのサンプリングは変数を左に，～で**確率分布**につなぎます。normal とか cauchy が確率分布の名前なので RStudio ではハイライト表示されているはず。一行の終わりはセミコロンで。複数のデータや変数，Y_i や σ_i は，i ごとに変わるということをあらわしていますが，プログラム的には Y[i] のように書いて，大括弧で変数を括って添字であることを表現します。データは 7 件あって，それぞれについて i さんの測定値 Y_i と測定誤差 σ_i があるので，for 文で i を繰り返しています。変数 sig[i] についての事前分布も同様に繰り返しています。ちなみに変数名 mu，sig，Y などは任意で，予約語でなければ好きな名前にしてもらって構いません。

parameters **ブロック** model ブロックで，変数 mu や sig という任意に命名して使っていたわけですが，それは何なのかを宣言してやらねばなりません。これらは求めたいパラメータなのですから，このブロックに型と共に宣言するわけです。変数 mu は実数型，変数 sig も実数型ですが 7 つの要素を持つ配列，しかも負の数を取らないので下限はゼロですよ，という宣言をしているところになります。範囲の設定に<>の記号を使っていること注意してください。配列については次の data ブロックで説明します。

data **ブロック** 最後にデータブロックです。モデルの変数でもパラメータでもない，外部から与えられるものなので，変数名 Y で宣言します。宣言のときに 1 つの変数に複数の要素がある (**配列**といいます) ことを明示するため，array[size] と書きます。この大括弧 ([ ]) の中身は整数で，今回は 7 件のデータなのでサイズは 7 としてあります。そしてその配列の数字が整数 (Integer) なのか，実数

(Real) なのか，といった型宣言をしたうえで，変数名を書きます[*6]。

いかがでしょうか。書き方 (文法) がわかると比較的単純，あるいは率直な書き方をしていることがお分かりいただけるかと思います。尤度と事前分布とデータを書くだけで，Stan は事後分布からの乱数を生成してくれます。確率密度関数を書いたり解いたりしなくても，結果の近似計算ができるというのはとても便利なことではないでしょうか。

19.3 Stan を使った MCMC の実践

さて Stan ファイルが準備できましたから，これはいったん保存しておいて，今度はこれを R の方から利用してやることになります。R からは `rstan` パッケージあるいは `cmdstanr` パッケージを使って，Stan を呼び出して結果を返してもらう，という流れです。Stan の言語は一度機械語に翻訳 (コンパイル) されるので，その間はしばらく待つ必要がありますが，それが終わると Stan から事後分布の乱数が次々生成され，R に返ってきます。R はそれら乱数からのサンプルを事後分布の代表値からなるデータセットとして扱って，記述統計や可視化を通じて確率分布の解釈に使うのですね。

それではどのように R から呼び出すのか，実際のコードを見てみましょう。呼び出し方は `rstan` パッケージと `cmdstanr` パッケージのどちらを使うかによって少し異なります。共通部分のあと，両方の呼び出し方について記述しますので，自分の環境にあった方を試してください。

■共通部分 どのパッケージを使うにしても，環境をクリアしたり，必要なパッケージを読み込んだりする部分は共通です。今回は次の 5 行を共通部分として読み込んでおいてください (code19.2)。

code : 19.2 環境の準備と共通部分

```
rm(list = ls())
library(tidyverse)
library(bayesplot)
# データ
x <- c(-27.020, 3.570, 8.191, 9.898, 9.603, 9.945, 10.056)
```

■cmdstanr パッケージによる呼び出し その上でさっそく `cmdstanr` パッケージを使って，Stan を呼び出し事後乱数生成機を作ってみましょう。これは次のようにコードを書きます (code19.3)。

code : 19.3 cmdstanr パッケージによる呼び出し

```
library(cmdstanr)
model <- cmdstan_model("sevenScientist.stan")
fit1cmdstan <- model$sample(
  data = list(Y = x),
  chains = 4,
  parallel_chains = 4
)
sample <- fit1cmdstan$draws() %>%
  posterior::as_draws_df() %>%
```

[*6] Stan の以前のバージョンでは `Y[i]` や `sig[i]` のように，大括弧 (`[ ]`) を後ろにつけて書いていました。しかしバージョン 2.32.0 はその表記は誤りとされ，ここで解説しているような `array` 表記がスタンダードになります。古い記法のままだと警告が出たり，RStudio も古いバージョンだと新しい記法に変更要請 (赤い波線がコードについたり) が出たりしますが，今すぐ問題になるというものではありません。とはいえ，なるべく新しいバージョンを使ったほうがよいので，表記法も新しいものに心がけましょう。

```
10    as_tibble()
```

■コード解説

1 行目　パッケージの呼び出し

2 行目　モデルのコンパイルを実行して `model` オブジェクトに入れます。コンパイルには少し時間がかかります。

4-8 行目　乱数発生。ここで R のデータを Stan に渡したり，並列化の指定をしたりします。実行すると画面に色々表示されるかと思います。

9-11 行目　得られた乱数をデータセットに整形。まず `posterior::as_draws_df` で MCMC サンプルだけを取り出し，`as_tibble` で tibble 型に変換します。

■rstan パッケージによる呼び出し　`rstan` パッケージを使って Stan ファイルを呼び出す場合は，少し書き方を変えて次のように書きます (code19.4)。

code : 19.4　rstan パッケージによる呼び出し

```
1  library(rstan)
2  options(mc.cores = parallel::detectCores())
3  rstan_options(auto_write = TRUE)
4  model <- stan_model("sevenScientist.stan")
5  fit1rstan <- sampling(model, data = list(Y = x))
6  sample <- fit1rstan %>%
7    rstan::extract() %>%
8    as.data.frame() %>%
9    as_tibble()
```

■コード解説

1 行目　パッケージの呼び出し

2-3 行目　CPU の並列化や上書き保存など，パッケージの設定

4 行目　モデルのコンパイルを実行して `model` オブジェクトに入れます。コンパイルには少し時間がかかります。

5 行目　乱数発生。ここで R のデータを Stan に渡しています。実行すると画面に色々表示されるかと思います。

6-9 行目　得られた乱数をデータセットに整形。まず `rstan::extract` で MCMC サンプルだけを取り出し，`as.data.frame` でデータフレーム型に，`as_tibble` で tibble 型に変換します。

どちらのパッケージで実行してもらっても結構ですが，コンパイル → サンプリングという流れは同じです。

またサンプリングの際に，R からデータを渡すことになりますが，データは R 環境では `x` というオブジェクト名だったものを，Stan の `data` ブロックで宣言した `Y` という変数にして渡すのだ，ということが書かれています。データは複数のものを与えることがあるので，`list` 関数で並べて渡すことにします。そのほかの設定については，順次説明します。

最終的に `tibble` 型でえられた MCMC サンプルのデータセットができあがります。これはパッケージによって少し変数名が異なりますが，内容は同じで，今回の設定だと 4000 行のデータになっているはずです。つまり事後分布からの乱数を 4000 個発生させたことになります。

そしてこの各行が 1 回のサンプリングで得られた**事後分布からの代表値**です。一部を見てみましょう (出力 19.1)。1 回目のサンプリングで，$9.98, 23.4, 4.63, 1.13, 0.123, 5.17, 0.0264, 1.24, -9.10$ という数字のセットがあります。変数名にあるように，前から $\mu, \sigma_1, \sigma_2, \cdots, \sigma_7$ の代表値と，`lp__`の値が得られています。最後の `lp__`は**対数尤度 (log posterior)** の略で，ここでは正確には事後対数尤度と呼ばれるものです。今回の代表値から計算された尤度の対数ということで，パラメータの推定値とは違いますのでここでは気にしないことにしましょう。ということで改めて，1 回目のサンプリングで $9.98, 23.4, 4.63, 1.13, 0.123, 5.17, 0.0264, 1.24$ という数字のセットが得られました。このように，今回求めたかったパラメータは 8 つあったわけですが，その 8 次元からなる空間の 1 つの座標点を取り出した 1 回目，ということです。2 回目は $9.90, 27.2, 13.1, 2.86, 1.23, 0.779, 0.0449, 0.446$ というセットでした。以下 3,4,.. と続き，4000 サンプルを得たわけです。このように結果は 8 次元からなるパラメータの**同時確率空間 (joint probability space)** から得られており，一行一行が 1 回 1 回のサンプリングステップなのです。

R の出力 19.1: MCMC サンプルの結果 (ごく一部)

```
    # A tibble: 4,000 × 9
    mu sig.1 sig.2 sig.3 sig.4 sig.5   sig.6 sig.7   lp__
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl>  <dbl>
1  9.98  23.4  4.63  1.13 0.123 5.17   0.0264 1.24   -9.10
2  9.90  27.2 13.1   2.86 1.23  0.779  0.0449 0.446  -7.72
3  9.99  20.9  5.12  1.11 0.132 4.48   0.0397 0.937  -8.97
```

19.4 MCMC 結果の診断

さて，MCMC が実行できましたが，その中身を少しみてみましょう。サンプリング結果を代入したオブジェクト名をそのまま入力すると，要約された結果が示されます。パッケージによって出力例は少し異なりますが，ほぼ同じ情報が含まれています。

cmdstanr の出力 2: MCMC の結果出力 (cmdstanr パッケージの場合)

```
variable  mean median     sd   mad     q5    q95 rhat ess_bulk ess_tail
  lp__   -9.70  -9.35   1.91  1.69 -13.28  -7.30 1.00      914     1694
  mu      9.64   9.81   0.67  0.36   8.28  10.36 1.01      879     1101
  sig[1] 54.19  31.91 105.83 16.95  15.41 142.60 1.01      663      215
  sig[2] 10.48   6.71  20.30  3.96   2.88  25.82 1.00     1272      935
  sig[3]  4.30   2.76   6.19  2.01   0.67  12.16 1.01     1281     1104
  sig[4]  2.52   1.31   3.88  1.47   0.11   8.58 1.01      603      252
  sig[5]  2.68   1.37   5.65  1.39   0.19   9.02 1.00     1291     1535
  sig[6]  2.66   1.26   5.11  1.47   0.09   9.46 1.01      492      312
  sig[7]  2.87   1.42   5.64  1.56   0.15   9.49 1.01      904     1049
```

rstan の出力 2: MCMC の結果出力 (rstan パッケージの場合)

```
Inference for Stan model: sevenScientist.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

         mean se_mean    sd   2.5%    25%   50%   75%  97.5% n_eff Rhat
mu       9.74    0.02  0.60   8.11   9.62  9.89  9.98  10.52   607 1.02
sig[1]  38.61    1.37 43.68  14.36  23.33 27.16 41.93 123.28  1016 1.01
sig[2]   9.44    0.57 12.88   2.86   4.63  6.22 10.57  33.54   515 1.00
sig[3]   4.64    0.40 14.66   0.63   1.33  2.49  4.98  17.98  1354 1.01
sig[4]   1.96    0.18  4.14   0.04   0.17  0.79  2.18  10.41   511 1.01
sig[5]   3.35    0.23  7.54   0.11   0.73  1.87  4.56  13.23  1044 1.01
sig[6]   2.00    0.25  3.65   0.03   0.22  0.77  2.44  10.55   216 1.02
sig[7]   2.50    0.17  5.71   0.09   0.44  1.24  2.49  12.28  1133 1.01
lp__    -9.45    0.07  1.67 -13.80 -10.19 -9.10 -8.39  -7.07   570 1.01
```

ここで確認して欲しいのは，**Rhat** です。これは「複数チェインの収束の程度」を表しています。まずMCMC によってサンプルを得る方法は乱数だったことを思い出してください。乱数ですから毎回同じ数字からスタートするわけではありません。そしてある乱数は次の数字を生むステップになっており，一歩一歩，事後分布の形に近づいていくものです。しかし違う数字からスタートしても，目標とする事後分布は同じはずですから，最終的にはこのステップの鎖はよく絡まったもの，つまりどこからスタートしても同じあたりをウロウロするものになるはずです。ですから，MCMC がきちんとできているかどうか，事後分布を正しく代表した数字になっているかどうかを確認する必要があります。この **Rhat** という数字は，1.0 であればピッタリ一致という数字で，1.1 よりも小さければ十分「絡まっている」と判断できます。今回はいずれも十分な収束が得られたと言えるでしょう。

このことを視覚的に確認することもできます。**トレースプロット (trace plot)** というのがそれで，MCMC のステップを可視化し，十分に絡まっていることを見ておくのです (図 19.3)。

ちなみにトレースプロットの描画のコードは,code19.5 の通りです。パッケージによって少し作法が違うので注意してください。`cmdstanr` で推定したものを使う場合は，`bayesplot` パッケージの力を借りて描画します。

code : 19.5 トレースプロットの描画

```
# cmdstanr パッケージのオブジェクトからトレースプロットを描く場合
fit1cmdstan$draws("mu") %>% bayesplot::mcmc_trace()
# rstan パッケージのオブジェクトからトレースプロットを描く場合
traceplot(fit1rstan)
```

これについては悪い例を見た方が早いかもしれません。図 19.4 に，チェインが絡まない場合のトレースプロットを示しました。このように，複数のチェインがそれぞれバラバラ，迷子になっている場合は，手元に生成された乱数が事後分布の適切な代表になっていないと判断することになります。

次に，**有効サンプルサイズ (effective sample size)** を見ておきましょう。`rstan` パッケージを使っている例の場合は `n_eff`，`cmdstanr` パッケージを使っている例の場合は `ess_tail` を見ましょう。これはMCMC サンプルがうまく取れたかどうかを診断した結果で，ここが少なくとも 3 桁ほどなければ不十分なのだ，と考えます。いくらあったらいいかについては，どれだけ MCMC サンプルを発生させたかに依存するので一概には言えませんが，少なくとも効果的に得られているのが 100 サンプルぐらいないとダメなんだ，とい

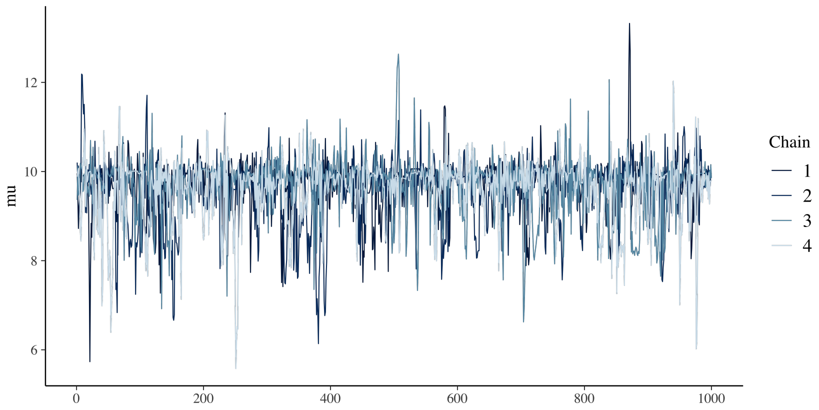


図 19.3　トレースプロットの例

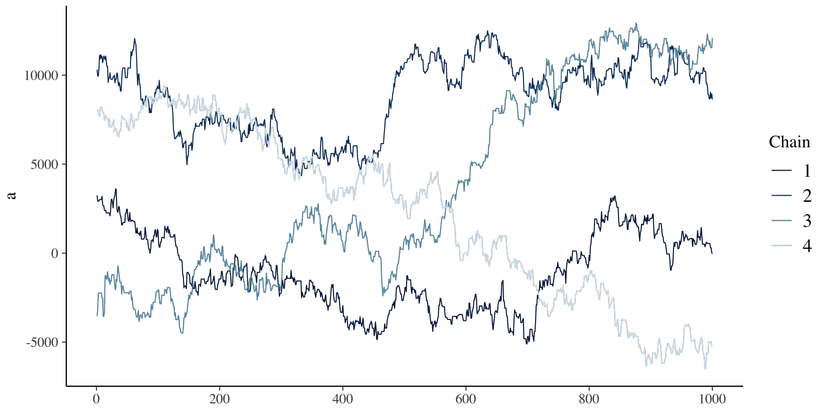


図 19.4　チェインが絡まない場合

うことです。

ちなみに，MCMC サンプルのチェイン (発生させた系列数) や，MCMC サンプルの数は，関数のオプションで指定します。今回は指定なしだったので，デフォルトの値で 4 チェイン，各々 1000 サンプルずつ作るようになっていました。これらを明示的に設定するには次のようにします (code19.6,code19.7,)。

code : 19.6　cmdstanr パッケージによるオプション指定

```
fit1cmdstan <- model$sample(
  data = list(Y = x),
  chains = 4,
  parallel_chains = 4,
  iter_warmup = 1000,
  iter_sampling = 4000
)
```

code : 19.7　rstan パッケージによるオプション指定

```
fit1rstan <- sampling(model,
  data = list(Y = x),
  chains = 4, iter = 5000, warmup = 1000
)
```

ここにある chains というのが生成する MCMC サンプルの連鎖系列の数です。「どこからスタートしても同じ事後分布にたどり着く」ことを検証するために，少なくとも複数のサンプルが必要で，デフォルトでは 4 本のチェインを作っていることになります。cmdstanr パッケージの方は，並列化する本数も別途 parallel_chains 関数で指定します。

また生成する乱数の数ですが，ここに**ウォームアップ (wamup)** という単語が出てきています。これは MCMC サンプルの際，最初の数ステップは調整のための準備運動期間中なので，この期間に得られた MCMC サンプルは代表値として使わない，という意味です。cmdstanr パッケージの場合，捨てるステップ数とサンプルするステップ数を明示的に指定していますからわかりやすいですね。最終的に得られる MCMC サンプルの数は $4000 \times 4 = 16000$ で計算できます。rstan パッケージの場合，全体の反復 (iter) 回数からウォームアップの分を引き算しなければなりません。引き算した残りの期間をチェイン数発生させることになります。今回の例では $(5000 - 1000) \times 4 = 16000$ サンプルできあがることになります。

19.5 MCMC の結果の解釈

さあ長々と話をしてきましたが，どうやら Rhat や有効サンプルサイズにも問題がなかったので，MCMC は事後分布からの適切な代表値を産んでいたと考えられますから，その結果を考えてみましょうか。

ここで，前の章 (セクション 18.5.3,Pp.197) で準備した関数を使ってみましょう[7]。これらをコード 19.8 のように使うと結果が得られます。私の結果では出力 1 のようになりました。

code : 19.8 stan の結果を準備し関数に与える

```
fit %>% MCMCtoDF() %>% MCMCsummary()
```

MCMC の結果 1

```
# A tibble: 8 × 7
  name         EAP       MED       MAP        SD       L95       U95
  <chr>  <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 mu         9.689     9.863     9.929     0.647     7.982    10.602
2 sig[1]    45.886    31.238    26.266    70.414    14.011   158.906
3 sig[2]    10.037     6.781     5.206    13.746     2.635    36.035
4 sig[3]     4.609     2.872     2.067     8.394     0.494    18.808
5 sig[4]     2.442     1.060     0.208     6.055     0.029    12.608
6 sig[5]     2.856     1.421     1.085     9.126     0.109    12.860
7 sig[6]     2.688     1.175     0.443     6.511     0.058    13.277
8 sig[7]     2.715     1.251     0.234     6.232     0.037    13.120
```

ここでは**事後確率最大値 (Maximum A Posterior)** を見ることにしましょう。これによると，μ すなわち真の測定値は 9.596 ぐらいじゃないか，という推測がなされています。10 より少し小さいくらいではないか，と思っていたようですが，こうして計算してみると確かにそれぐらいのようですね。そして 7 人それぞれの測定精度が σ_i として算出されています。これをみると，最初の測定者は 23.101，2 番目の人は 6.849 になっています。これが大きいということは，誤差の幅広さを表しているのですから，測定精度が悪いということを意

[7] 伴走サイト https://kosugitti.github.io/psychometrtics_syllabus/ からは，これらの R コードをダウンロードできます。Stan ファイルは自分で準備し，読み込み先やファイル名などは適宜変更してください。

味します。確かに最初の二人は非常に精度が悪く，他のベテラン測定者が 0.3 とか 0.9 ぐらいの誤差であることに比べれば，圧倒的に悪いということが言えますね。

もちろんこれは点推定値で，真の測定値が 9.596 に違いないとか，最初の人の測定精度が 23.101 だと断言すると，おそらくほぼ確実に外れた予測ということになるでしょう。そこで幅を持った予測，すなわち**区間推定 (interval estimation)** を行えば良いのです。区間を考えれば，真の測定値は 10.547 から 7.753 の範囲にあるだろうとか，最初の測定者の精度も最悪 151.573，最善なら 14.259 ぐらいである可能性があるわけです。

ベイズ推定のポイントとして，これらは事後分布，すなわち**確率分布による予測**であるということが挙げられます。真の測定値 μ のありそうな範囲が 10.547 から 7.753 にある確率が 95% だ，という言い方ができるところです。**モーメント法**による**信頼区間 (confidential intervals)** とは違い，ベイズ法の予測は**確信区間 (Credible Intervals)** という言い方になります[*8]。

いかがでしたでしょうか。今回はモデリングとベイズ推測の実際，Stan コードの書き方を実践してみました。この「7 人の科学者」がおもしろい点は，少ないサンプル (たった 7 件のデータ！) で推測ができること，簡単なコードで検証できること，というのもありますが，なにより標準偏差 (分散) を検証対象とすることにあると私は思います。心理学の実験のほとんどが，平均の比較，操作の効果の話になっているのですが，データの散らばりというのも測定精度のように心理学的に意味のあるものとして考えられ，検証の対象にしたっていいのです。もちろん正規分布でなくてもいいですし，さまざまな分布のさまざまなパラメータに意味があるなら，それをわからないものとして検証しようと言える，というのはとても可能性が広がる話ではないでしょうか。

19.6 課題

次の計算をする R/Stan コードを記述し，提出してください。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

1. 8 人目の科学者が現れて，測定値 18.25 だったと報告してきました。この人のデータも含めた分析モデルにするために，R および Stan のコードを書き換え，MCMC 推定してください。また 7 人の時とくらべて結果がどう変わったでしょうか？気づいたことを報告してください。
2. MCMC サンプルの精度を上げるため，5 つのチェインで，warmup 期間を 10000，最終的な MCMC サンプルを 10 万点得られるように R のコードを書き換えてください。
3. MCMC サンプルを使って，μ と σ の分布の形を可視化するコードを書いてください。

[*8] 信頼区間の場合は，ここにあると予測したとして，その予測が当たる確率を表しています。すなわち当たるか外れるか，0/1 の話をしているのであって，この「範囲に存在する」という大きさ・幅の話をしているわけではないことに注意です。

第 20 章

モデリングの目から見た検定 1；二群の平均値の差

前回はデータ生成メカニズムという観点で考えることで，標準偏差 (分散) を「測定精度」と考えた検討ができることを示しました。今回は，心理学でもよく使われている平均値の差を検討対象とするモデルを考えることにします。

皆さんは帰無仮説検定のことを覚えているでしょうか。正規分布を仮定した母集団からの標本統計量は，正規分布に従うことを利用して，母平均の区間推定を行い，群間に差があるかないかといった判断を確率的に行うのが帰無仮説検定でした。心理学では要因計画と帰無仮説検定が合体し，群間の差の形でデータが得られるようにすることで結論を導き，考察を重ねてきました。また帰無仮説検定という手法は，誤用や誤解が多く，ひどい時には悪用もされるということについてもみてきたと思います。ところで，帰無仮説検定の考え方はデータが得られたところから考え始めるデータ駆動型の分析モデルでした。ではデータがどのように出てきているのかを考える，**データ生成モデリング**の考え方をつかうと，この平均値の差についての検討はどのように姿を変えるのでしょうか。

ここでは t **検定**を例に話を進めていきたいと思います。

20.1　t 検定の過程と実際

t 検定は，二群の平均値の差を比較し，結論を下すという最も典型的な帰無仮説検定の例の 1 つです。とくに独立した二群の検定は，最も単純な要因計画 (一要因二水準 Between デザイン) ということができるでしょう。これについて，よく思い出せない人はデータ解析基礎の資料や，山田・村井 (2004)，清水 (2021) などを読んで，分析の流れや仮定をもう一度確認しておいて欲しいと思います。

非常に駆け足ながら概略を説明してみますと，次のようになります。

1. 標本の母集団が正規分布していると考える。母集団から無作為に選ばれた標本が二群あるとする。
2. 二群の一方に何らかの処置を加え，他方には何もしない (あるいは検討したい処置と同等の効果のない処置を施す[*1])。前者を実験群，後者を統制群という。データ (標本) は正規分布するはずだから，その平均値を見ることで誤差や個人差は相殺される。つまり群間の平均値の差が効果の大きさであるということができる。

[*1] たとえば「単語リストは声に出して覚えたほうが記憶の定着度が高い」ということを検証したい場合，声に出す群と出さない群で比較することになりますが，声に出さないことが (頭の中で反芻するなど) 従属変数に与える別の効果を持っていることがあるので，音のない映像を見せるなどして効果のない同等の処置をした群を比較対象にする，ということを考えたりするわけです。

3. ところで標本平均値などの**標本統計量**も正規分布に従うことがわかっている。母集団が $N(\mu, \sigma)$ であれば，標本平均は $N(\mu, \frac{\sigma}{\sqrt{n}})$ に従う (ここで n はサンプルサイズ)。
4. 標本平均が従う分布 (散らばりの位置と幅) が分かれば，母平均がどのあたりにあるのかは区間推定することが可能。実験群・統制群ともに母平均の値を推定する。これが異なっていれば標本を超えて母集団で効果があった，と結果を一般化できる。もちろん点推定値は母平均の値とピッタリ同じとは思えないし，標本平均もピッタリ同じになるはずがないから，点推定値ではなく区間推定で考えたい。
5. ところで標本平均の従う分布の中には，σ つまり母 SD がパラメータとして入っている。母数がわからないという前提のもとでは，どの程度の幅で散らばるのかがわからないことと同じ。そこで σ の代わりに $\hat{\sigma}$ を用いて推定することを考える。この場合，標本平均は正規分布ではなく t 分布に従うことがわかっている。
6. t 分布を使った区間推定をしても，差があるのかないのか判断するために何らかの基準が必要である。そこで「差がない」という主張と「差がないとは言えない」という主張を戦わせて，判定すると言う形式を取る。前者の主張を**帰無仮説**，後者の主張を**対立仮説**という。また判断も確率的になるので，勝敗を決める基準を 5% とする。この確率は**有意水準**とか**危険率**と呼ばれる。
7. データから t 分布に従う統計量を計算し，その値が出てくる確率を算出する。この確率は p **値**と呼ばれる。これが有意水準よりも小さいようであれば，帰無仮説の仮定の下で算出した数字が滅多に生じ得ないことを意味するから，帰無仮説が間違っていたのだと判断してこれを**棄却**し，対立仮説を**採択**する。

さて，この二群の平均値を検定することの流れですが，とくに「帰無仮説のもとでの t 値を算出して判定する」というあたりがややこしかったかもしれません。数式で書くと嫌われそうですが，帰無仮説というのは二群の平均値に差がない，という仮定だったので，実験群から考えられる母平均 μ_A と統制群から考えられる母平均 μ_B に差がない，つまり $\mu_A = \mu_B$，からの $\mu_A - \mu_B = 0$ という位置母数が 0 の理論分布のを考えている，というところがポイントです。検定統計量である t 値は次のように計算できるのでした。

$$t = \frac{\bar{X}_A - \bar{X}_B}{\sqrt{\frac{(n_1-1)\hat{\sigma_A}^2+(n_2-1)\hat{\sigma_B}^2}{n_1+n_2-2}(\frac{1}{n_1}+\frac{1}{n_2})}}$$

とっても複雑な式に見えますが，複雑そうなのは分母で，分子は標本平均の差を表しているに過ぎません。参照する t 分布は，標準正規分布が $N(0,1)$ だったように，位置と幅のパラメータを持ち，$t(0, df)$ で考えられる分布にこの統計量を照らし合わせると，差 0 で自由度に応じて変わる幅 df の分布から「どの程度極端な値が出てきたのか」の確率を計算できるわけです。ちなみに分母は，母分散ではなく標本から計算される分散 s^2 からバイアスを除いた不偏推定量 $\hat{\sigma}^2$ を使っています。この計算はサンプルサイズ n ではなく $n-1$ で割ることによって計算されるのですが，2 つの群それぞれについて $n_j - 1$ で割ったものを足し合わせるために，いったん $n_j - 1$ 倍して二群のサンプルサイズ -2 で割り直す，という作業をしているため，分母がとくに複雑に見えているだけです。

ともあれ，こうして検定するんだったという流れを思い出したところで，データ生成モデリングの観点からこれを考え直してみましょう。仮定としておいてあるのは，両群ともに同一の正規分布から得られた標本であり，操作によってその平均値が異なっているはず，ということだけです。つまり実験群のデータは $X_{i,A} \sim N(\mu_A, \sigma)$，統制群のデータは $X_{i,B} \sim N(\mu_B, \sigma)$ という分布が前提とされているのです。

これをそのまま設計図にし，コードにしてみましょう。設計図として図 20.1 が，これをもとにコードが `code:20.1` のようにかけていればいいでしょう。コードが設計図をほぼそのまま文字起こししていることを確認してください。

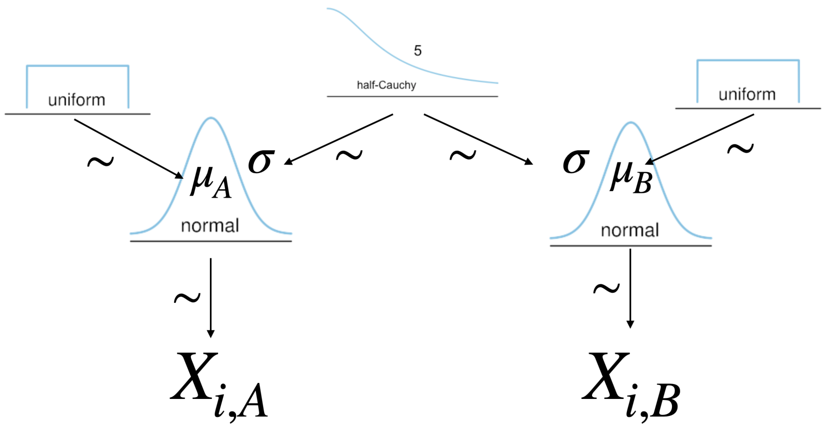


図 20.1　平均値の差を比べるときの設計図

code : 20.1　二群の平均値のコード

```
data{
    int<lower=0> N1; // Number of Subjects in Group 1
    int<lower=0> N2; // Number of Subjects in Group 2
    array[N1] real X1; // Data in Group 1
    array[N2] real X2; // Data in Group 2
}

parameters{
    real mu1;
    real mu2;
    real<lower=0> sig;
}

model{
    // likelihood
    X1 ~ normal(mu1,sig);
    X2 ~ normal(mu2,sig);
    // prior
    mu1 ~ uniform(0,100);
    mu2 ~ uniform(0,100);
    sig ~ cauchy(0,5);
}
```

ここで Stan コードにちょっとした工夫を 2 点入れています。1 つ目は data ブロックにある，変数 $N1, N2$ の存在です。二群のデータについて，サンプルサイズがとくに決まっていませんから，ここで外部から入力することにしています。2 つの群それぞれのサンプルサイズをデータとして取り込み，その数と同じだけデータ数を配列として宣言しているところがポイントです。もう 1 つのポイントは，尤度のところです。丁寧に書くならば，次のようにするべきでしょう (コード 20.2)。

code : 20.2　丁寧な尤度の記載

```
    ...
```

```
model{
    for( i in 1:N1){
        X1[i] ~ normal(mu1 , sig);
    }
    for( i in 1:N2){
        X2[i] ~ normal(mu2 , sig);
    }
    ...
}
```

このように，それぞれの群にいて for 文を回し，各データ点が正規分布から出てきているよ，ということを明示するのです。しかしそうしなかったのは，Stan が「分かりきったことは書かなくていいよ」という優しい設計になっているので，変数 X1 の要素すべてがある分布に従うのであれば，まとめて `X1 ~ normal(mu1, sig);` と書いても良い，つまり X1 の配列要素を逐一指定しなくても同じ尤度関数をあてがってくれるというのを利用しています。

ともかくこれでできるのでした。では少し具体例をつかって，これがどういう結果をもたらしているのかを確認しましょう。

20.1.1　t 検定の具体例

> 統計学の新しい指導法の教育効果を見るため，全国の大学の心理学科から学生を無作為に 16 名選び，従来の指導法グループ（統制群）と，新指導法のグループ（実験群）にランダムに 8 名ずつわりつけました。プログラム終了後，心理統計のテストを行いました。統制群のスコアは $20, 40, 60, 40, 40, 50, 40, 30$，実験群のスコアは $30, 50, 70, 90, 60, 50, 70, 60$ となりました。新たしい指導法は，心理学科の学生に効果があると言えるでしょうか？ 5% 水準で検定してください。

これを t 検定するのは簡単ですね。計算式は複雑でも，機械がやってくれるから楽なのが NHST[*2]のいいところです。

code : 20.3　帰無仮説検定のコード

```
groupA <- c(30, 50, 70, 90, 60, 50, 70, 60)
groupB <- c(20, 40, 60, 40, 40, 50, 40, 30)
## t検定
t.test(groupA, groupB, var.equal = TRUE)
```

結果は出力 20.1 のようになります。検定統計量である t 値，検証のための自由度 df，帰無仮説のもとでの出現確率 p 値，が表示されています。5% より小さいので有意差あり，という判断ができます。もっとも，t 値とか自由度とかは，データに関係のない数字なのでピンと来ないというところもあるかと思います。

[*2] **帰無仮説検定 (Null Hypothesis Significance Test)** の略です

R の出力 20.1: 検定の結果

```
> t.test(groupA, groupB, var.equal = TRUE)

 Two Sample t-test

data:  groupA and groupB
t = 2.6458, df = 14, p-value = 0.01919
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  3.786937 36.213063
sample estimates:
mean of x mean of y
       60        40
```

20.1.2 モデルで推定してみる

つづいて，先ほどのコードを使ってモデリングによる推定をしてみたいと思います。私の環境下では，次のような数字になりました。

MCMC の結果 2

```
# A tibble: 3 × 7
  name         EAP        MED        MAP         SD        L95        U95
  <chr> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 mu1       60.006     59.976     59.700      5.608     48.869     71.420
2 mu2       39.990     39.997     39.650      5.679     28.654     51.128
3 sig       15.534     15.048     14.070      3.111     10.893     22.877
```

これを見ると，未知のパラメータ μ_1, μ_2, σ の値がそれぞれ推定されていますが，「判定」のような結果は出てきていません。というのも，勝負するような形で考えているのではなく，パラメータがどこにありそうか，ということを示す**確率分布**を求めているからです。それでも，**EAP** をみると 60006 と 39.990，95%CI をみると実験群は `[48.869, 71.420]`，統制群は `[28.654, 51.128]` とあり，実験群の平均値が 51 より小さい値になる確率はほとんどなく (2.5% 以下)，統制群の平均値が 49 以上より大きくなることもまたほとんどないわけですから，確実に差があると言ってもほぼ過言ではない，と言い切れるでしょう。

モデリングの利点は，t 値や p 値のような直接のデータと関係ない値を出すのではなく，データに直接関係する数字で考えさせてくれるので，イメージしやすいところもあるかもしれませんね。

20.1.3 分散の等質性

ところで，t 検定の場合は「分散が同じであるかどうか」という条件によって，算出方法を補正するという考え方があるのを覚えていますでしょうか。**Welch の補正**というやつで，こちらの方が一般に条件が緩いものですから，t 検定では普通こちらの方が使われます。

R の出力 20.2: Welch の補正の例

```
> t.test(groupA, groupB, var.equal = FALSE)

 Welch Two Sample t-test

data:  groupA and groupB
t = 2.6458, df = 12.274, p-value = 0.021
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  3.570406 36.429594
sample estimates:
mean of x mean of y
       60        40
```

出力 20.2 にあるように，オプションとして指定するだけで瞬時に答えが出てきます。t 値，自由度，p 値が先ほどと異なって出ていますが，結果はほぼ変わりがありません。

モデリング上ではこれをどう表現すれば良いでしょうか。これは実は簡単で，分散が実験群と統制群で異なるというのですから，別々に推定してやれば良いのです。

code : 20.4　分散が異なる場合の推定モデル

```
        ...
parameters{
    real mu1;
    real mu2;
    real<lower=0> sig1;
    real<lower=0> sig2;
}

model{
    // likelihood
    X1 ~ normal(mu1,sig1);
    X2 ~ normal(mu2,sig2);
    // prior
    mu1 ~ uniform(0,100);
    mu2 ~ uniform(0,100);
    sig1 ~ cauchy(0,5);
    sig2 ~ cauchy(0,5);
}
```

20.2　差の分布

さて，二群のデータから考えられるそれぞれの母平均が推定されました。今回は実験群と統制群の平均値差が大きく離れており，一方の 95% 上限が他方の 95% 下限を上回っているという状況でしたので，「差がある」と判断できましたが，そうでないこともありそうです。つまり，微妙な差のとき，ですね。次のようなデータで試しに推定してみましょう。まずは 5% 水準で検定をしてみます。

code : 20.5　帰無仮説検定のコード

```r
groupA <- c(30, 50, 70, 90, 60, 50, 70, 60)
groupB <- c(30, 45, 60, 40, 60, 50, 40, 30)
## t検定
t.test(groupA, groupB, var.equal = FALSE)
```

今度は $t(12.175) = 2.0761, p = 0.0597$ で有意とは言えなくなりました[*3]。このデータを，先ほどのモデルで推定してみましょう。目にも分かりやすくするために，推定された平均値の事後分布をプロットしてみたいと思います (図 20.2)。

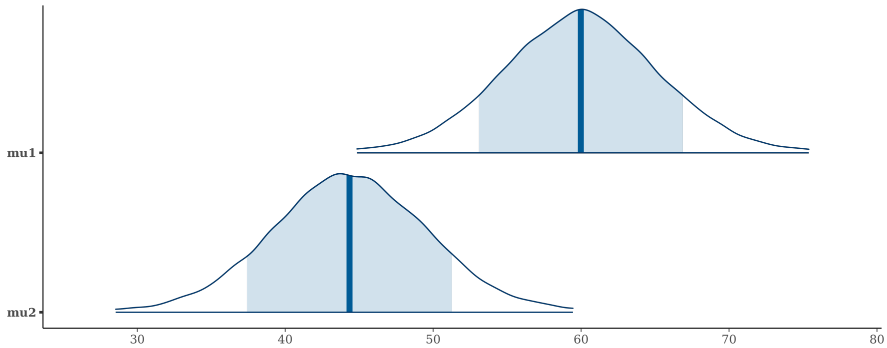


図 20.2　二つの平均値パラメータの事後分布

今回は 80% 確信区間と，確率分布の両端を 99% のところまで描画してみました。2 つの平均値パラメータは，分布の代表値で見ると異なっているのですが，重複している領域が多く一概に「一方が他方より大きい」と言いにくい結果になっています。帰無仮説検定のようにズバっと「差がある」「ない」と言い切れればいいのですが，このような場合はどうしたらいいでしょうか。

正解は「このままにしておく」です。急いで差があるとかないとか言い切るのではなく，これが区間推定の結果そのものですから，この程度の重複があり得るもの・それほど明確な答えは出ないもの，として考え続けるしかありません。

しかしまあ，このままでは考えるヒントが少ないので，ここで少し工夫をしてみましょう。帰無仮説検定では，$\mu_A = \mu_B$ という帰無仮説が反駁できるかどうかが問題なのでした。今回，μ_A や μ_B を代表する数字の候補が色々ありますので，これを使って「実際どの程度差があったのか」を計算してみれば良いのです。MCMC サンプルは，毎回推定したいパラメータの同時分布からの代表値を持ってきているという話をしました。たとえば 1 回目のサンプルは $\mu_A = 62.4, \mu_B = 25.7$，2 回目のサンプルは $\mu_A = 60.4, \mu_B = 36.8$ と言った具合に，です。それらは事後分布からのサンプル，事後分布を代表した実現値の 1 つですから，これを使って $\mu_A - \mu_B$ の計算をできます。すると**差の分布**の代表値が MCMC サンプルで計算できることになります。

これを使って，$\mu_A - \mu_B = 0$ になる確率はどれぐらいか，というのを考えれば良いのかもしれません。もっとも，連続変数におけるある一点の確率はゼロですし[*4]，代表値といってもピッタリゼロになることはほぼあり得ないでしょうから，「どの程度ならゼロと見做すか」ということを考える必要があります。この範囲は**実質的に等価な範囲 (Region Of Practical. Equivalence; ROPE)** と呼ばれ，たとえばこのテストの例では ±5 点ぐらいは誤差みたいなものだけど，5 点以上変わったら意味のある違いだ，ということであれば「5 点以上点差が出た確率」を計算できます！

この計算は，MCMC サンプルを取り出した R のオブジェクトを使って計算することもできますが，実は

[*3] 5.9% だから惜しい！とか「有意傾向にある」なんて変なこと言い出したらダメですよ。勝負は 5% だと決めて始めたのですから。

[*4] 確率は相対的な面積で表される数字ですから，面積を持たない点については確率が定義できないのです。

Stan の中で計算させてしまうこともできます。`generated quantities` ブロックが，この MCMC サンプルを取り出したあとの処理をするブロックに該当します。ここで計算される量のことを**生成量 (generated quantities)** と呼びます。実際にどのように使うのかをみてみましょう。

code : 20.6 生成量ブロックの利用

```
    ...
generated quantities{
    real diff;
    int<lower=0, upper=1> FLG;
    diff = mu1 - mu2;
    if(diff > 5){
        FLG = 1;
    }else{
        FLG = 0;
    }
}
```

コード 20.6 には生成量ブロックを使う例を示しました。ここもブロックですので，変数の宣言が必要です。今回はまず `diff` という変数を実数型で宣言してみました。これはその下で $\mu_1 - \mu_2$ として定義されていることからわかるように，差の分布 (の代表値) を生成す流のです。これを見ると，差の生じる確率を考えることができます。次に整数型 (int 型) で `FLG` という変数を宣言しました。上限が 1 で下限が 0，というか実質的には 0 か 1 の数字しか入らない，ある/なし，成立する/しないといった情報しか持たない変数です。いわゆる「フラグが立ったかどうか」の指標であり，その実態は下の `if` 文で表現されています。ここでは先ほど計算した `diff` が 5 よりも大きいのであれば 1，そうでなければ 0 を代入する，というコードになっています。

これを実行すると，パラメータのサンプルに加えて，そのパラメータから計算されるさまざまな量も同時に算出されます。出力例を見てみましょう。

MCMC の結果 3

```
# A tibble: 6 × 7
  name        EAP        MED        MAP         SD        L95        U95
  <chr> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 diff     15.650     15.605     14.940      8.156     -0.516     31.706
2 FLG       0.911      1.000      1.000      0.285      0.000      1.000
3 mu1      60.035     60.035     60.112      6.737     46.602     73.575
4 mu2      44.385     44.374     44.291      4.594     35.249     53.696
5 sig1     18.509     17.472     15.960      5.369     11.414     31.792
6 sig2     12.496     11.752     10.761      3.683      7.629     21.594
```

ここにあるように，差の 95% 確信区間は `[-0.516, 31.706]` であることがわかります。差が大きければ 31.70，小さければ逆転して-0.52 になる可能性すらあるのです。この確信区間は 0 を含んでいますから，差がない可能性は否定できません。そういう意味で，帰無仮説検定的表現を借りるなら，5% 水準で差がないとは言い切れない，ということができるでしょう。

とはいえ，せっかく分布として情報が得られているのに，点推定にして勝負をするのはもったいないともいえるわけです。結論を急がずに，どの程度の差があるのかをじっくり考えてみるのがいいでしょう。また変数 `FLG` を見ると，その平均が 0.908 となっています。これは変数 `diff` が 5 よりも大きい時は 1，そうでなければ 0 という条件の数字で，その平均は 1 になった数を総 MCMC サンプル数で割った相対度数になっている

のですから，確率を表す数字だと考えても良いことになります。つまり，差が 5 よりも大きくなる確率は 90.8% である，と考えることもできるのです。これを応用すると，さまざまな仮説を考えることができそうですね。

20.3 帰無仮説検定を省みる

さてこのように考えてくると，帰無仮説検定について異なる視点で見ることができるようになったのではないでしょうか。ここでは，帰無仮説検定の独特さを 3 つの点から考えてみたいと思います。

20.3.1 不平等な対決

帰無仮説検定は帰無仮説 vs 対立仮説という勝負に持ち込んで判断する方法である，ということを冒頭に述べました。平均値の差の検定の文脈で言えば，帰無仮説は $H0 : \mu_1 = \mu_2$ であり，対立仮説は $H1 : \mu_1 \neq \mu_2$ です。このように，帰無仮説は「差がない」の一点張りであり，対立仮説は「それ以外ならなんでもあり」という状態を指しているのでした。そもそも，帰無仮説というのは非常に限定的な仮説だったのです。無に帰してほしい仮説とは言え，あまりにも不利な勝負なのでした。その上で，勝負は「差がない」「ある」のどちらかにしかなりません。ごく微妙な差であっても，判定によれば差が「あった」といえますし，ギリギリでもなければ「なかった」というしかないのです。このような過度に結果を際立たせる報告は，科学的な研究実践の文脈では拙速なことになりかねませんので，注意が必要なのでした。

データ生成モデリングのやり方で同じデータを分析してみると，差も分布の形で描かれますから，「あったか，なかったか」という単純な問題にせずに色々考えてみたくなるのではないでしょうか。

20.3.2 量的な判断を

もっとも，出力 20.2 の t 検定の結果をよくみてみると，t 値や p 値の下に 95% 信用区間が示されており，ここでは `[3.570406, 36.429594]` という数字が出ています。そうです，帰無仮説検定をやっているようでも，しっかり区間推定した結果も表示されていたのです。これは差の信用区間で，この区間が 0 を跨いでいないということは，差がないとは言えない $(\mu_A - \mu_B = 0)$，ということを意味しています。この区間の幅をみると，3 から 36 と随分ひろくとられているように思えます。つまり，今回の検定結果は差があるとなったけれども，それほど確かなものではないかもしれないぞ，と慎重に判断することもできたはずなのですね。

この差についての量的な評価については，**効果量 (effect size)** という指標で表現されるのでした。効果量とは，標準化された差の大きさです。標準化するというのは，標準偏差で割って幅を整える，あるいは標準偏差何個分という単位で表現することでもあります。比べられるスコアはさまざまですから，その標準偏差で表現することで一般的に議論できるのです。平均値の差の検定においては，**Cohen の d(Cohen's d)** などが指標としてもいいられますが，それは $\frac{\bar{X}_1 - \bar{X}_2}{\sigma}$ で計算されるのでした。

これは生成量を使って簡単に計算できます。今回の場合，計算する場合は，生成量に次のようにすれば良いのです。

code : 20.7 生成量ブロックで効果量の算出

```
    ...
generated quantities{
    real diff;
    real cohen_d1;
    real cohen_d2;
    diff = mu1 - mu2;
```

```
        cohen_d1 = diff / sig1;
        cohen_d2 = diff / sig2;
}
```

どちらの群の標準偏差を基準にするか，ということを考える必要がありますが[*5]，相対的な大きさの比較も簡単にでき，しかもその結果も分布の形で得られる，すなわち効果量の確信区間 (効果が少なくともどの程度ありそうかとか，大きければどの程度ありそうかとか) を考えることもできるのです。

もちろんこれらの計算は，帰無仮説検定の文脈，言い換えれば伝統的な統計手法 (**モーメント法**による推論) であってもできたことなのですが，データ生成モデルという観点からみるといっそう理解がしやすいのではないかと思います。

20.3.3 方向性を持った仮説も

今回は最後に，「実験群が統制群よりも 5 点以上大きくなる確率」というのを考えました。この「一方が他方よりも大きい」という仮説は，帰無仮説検定の文脈では**片側検定 (one-tailed test)** と呼ばれます。一般的に「差がある」という対立仮説は，プラスであれマイナスであれ違いが生じている，という意味であり，統計量は分布の右側でも左側でも，とにかく極端な値になりさえすれば良いという判断でした。そうではなく「$A > B$ のはずだ」という仮定であれば，小さくなる可能性を考えなくていいのですから，統計量のどちらか一方の極について考えれば良いことになります。

とはいえ検定ですから，ある有意水準で一方が他方より大きいと判断すると間違える確率が云々，という判定に落とし込むことという点では同じでした。

今回は生成量を使って，5 点以上大きくなる**確率**を求めることができました。これは検定統計量や p 値のように架空の値ではなく，もとのデータに直結した数字や確率になっていますから，解釈が比較的自然にできるというのが利点です。プログラム上の表現も，たとえば今回のように `if` 文を使って表現するのは非常に直感的で，「もしあれがこうなってそれがこうなったらどうなる？」といろいろな仮説を考えていくこともできますね。帰無仮説検定のロジックは，結果判定のために仮説を限定的な型にはめ込んでしまいます。しかしその型を飛び越えていろいろなことをやってもいいのだ，できるんだというのは，データがどうやって生まれてきているかを考えている「創造主」としての特権かもしれません。

ただし注意してもらいたいのは，これらの検証の仕方は，今回のデータと仮定されたモデルという前提の上で成立する割合を確率とみなしているに過ぎない，ということです。データが変わればその数字も変わるでしょうし，そもそもデータが正規分布から出てきていないのかもしれません。本当は XX 分布から出てきているのに，YY 分布を仮定したモデルで計算したら，仮説が正しい可能性が 100% !といっても虚しい (明らかに間違っている) ことになりますね。心理学のデータの場合はえてして，どういう分布やメカニズムになるのかがはっきりわからないものです。ただ，あまりにもわからなさ過ぎて，さまざまな影響が混ぜ合わさっているので，色んな要素が相殺しあって結局ほとんど正規分布とみなしていいよ，ということだったりします。心という目に見えないものに，不良設定問題を解くためのツールで立ち向かうという，無理に無理を重ねるような推論の世界であるということに，自覚は持っていたいですね[*6]。

[*5] 二つの標準偏差の平均をとった，プールされた標準偏差で計算することもあります。

[*6] いいんです，それでも。だってそんなよくわからない人間というのが好きなんですから。

20.4 今回のまとめ

我々がデータを分析する時は，そのデータの数字がどうであったか，ということを真っ先に考えたいはずです。身長の差は何センチあったのか，最終学歴によって生涯年収は何円差がつくのか，といった具体的な単位に基づく**実質的な差**が一番大事なはずです。心理学をやっていくと，尺度をはじめとした「絶対的なスコア」が出てきませんから，尺度で何ポイント差があったのか，ということがあまり意味のある実態と対応しないことも少なくありません。そう言った時のために，**標準化された差**を考えるようになったのです。その計算は非常に面倒ですが，機械がすぐに計算してくれることによって結論を急ぎすぎ，単位も実際のデータとも関係のない統計量を参照して**有意差**を求めるようになってしまったのでは意味がありません。

データがどういうメカニズムで生まれてきたかを考え，母数をベイズ推定するというやり方は，いちいちコードを書かなければならないこともあって，非常に手間がかかるようにも思えます。しかし自分が何を仮定していたのか，どういうメカニズムの元で考えていたのかを自覚し，またその推定値を使って自由に仮説を考えることで，決まりきった分析手続きに落とし込むという単調な作業から解放され，クリエイティブにデータに向き合えることでもあります。

20.5 課題

次の計算をする R/Stan コードや回答を記述し，提出してください。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

■フライドポテトの研究 あるコンビニで，ホットスナックのフレンチフライを 2 つ買ったところ，どうも一方より他方の方が長くて大きい気がした。そこでそれぞれの袋から取り出して長さを測定してみた (単位 cm)。測定結果が表 20.1 である[*7]。このデータを用いて次の問に答えなさい。なおこの数値はシラバスのサイト上

表 20.1 二つの店舗のポテトの長さ

A	8.4	11.3	8.1	11.2	5.8	6.3	7.1	10.9	7.1	6.5	5.0	3.0	7.2	6.5	6.4	6.4	9.3	8.3
B	6.7	7.2	4.2	11.0	7.5	8.9	7.0	8.0	7.2	4.2	6.0	9.0	8.6	9.0	5.0			

のコードで取得可能です。

1. ポテトの長さは正規分布に従うと仮定します。また両群の分布の幅は同じであると仮定します。t 検定で二群の平均値に差があるかどうか，判断してください。t 値や自由度，p 値を参照しながら説明すること。効果量も算出するとなお良いです。
2. ポテトの長さは正規分布に従うと仮定します。また両群の分布の幅は同じであると仮定します。その上で，群 A と群 B の平均値を推定するモデルを作り，ベイズ推定してください。その上で，結果の平均値，中央値，MAP 推定値，90% 確信区間を報告してください。
3. 両群のポテトの長さの平均が，3cm 以上違うようであればクレームをつけに行こうと思います。3cm 以上の差がある確率はどれぐらいですか。推定してください。

[*7] このデータは実際に筆者がコンビニの二つの店舗でポテトを購入し，長さを測定した結果に基づいています。

4. よく考えてみれば，両郡の分布の幅が同じであるという仮定はおかしいような気がしてきました。そこで，それぞれの群毎に分布の幅を推定するモデルに作り替え，ベイズ推定してください。その上で，結果の平均値，中央値，MAP 推定値，90% 確信区間を報告推定してください
5. 異なる分散を指定したモデルで，両群のポテトの長さの平均差が 5cm 以内であれば，クレームをつけにいくのはやめておこうと思います。差が 5cm 以内である確率はどれぐらいですか。推定してください。

第 21 章

モデリングの目から見た検定 2；パラメータの世界とデータの世界

前回は，二群の平均値差の検定を行うデータ生成モデルを考えました。平均値差の検定には正規分布が仮定されますから，データ生成モデルも正規分布からデータが作られていると考えれば良かったわけです。仮定に忠実に設計図を書き，それに対応した Stan コードを書くと母平均の推定値が出てきます。検定の時も母平均の推定値を使って考えるのですが，検定統計量にしてしまうので実感が湧きにくいところがあったかもしれません。データ生成モデルの場合は，直接パラメータを考えるのでイメージがしやすいという側面がありました。

さらに，MCMC による推定ですから事後分布からの代表値が，実現値として手元のオブジェクトに代入されています。これをつかった計算をすることで，たとえばパラメータの差を計算でき，その分布を考えることができるのでした。こうした計算は Stan の `generated quantities` ブロックを使うことで，MCMC の結果と同時に考えることもできるのでした。

さて今回も，この `generated quantities` ブロックをつかって，生成量からいろいろ考えてみましょう。

21.1　事後予測分布

前回，二群のデータ $X_{i,A}$ および $X_{i,B}$ に対して，次のようなデータ生成モデルを考えました。

$$X_{i,A} \sim N(\mu_A, \sigma), X_{i,B} \sim N(\mu_B, \sigma)$$

このパラメータ μ_A, μ_B, σ の推定値，$\hat{\mu_A}, \hat{\mu_B}, \hat{\sigma}$ の分布からの代表値が MCMC によって得られるのでしたね。MCMC サンプルは iteration ごとに 1,2,3...M 個得られたとすると，その期待値すなわち **EAP** は，次の式で求めていることと同じです[*1]。

$$\hat{\mu_A}_{eap} = \frac{1}{M}\sum \mu_A^i = \frac{1}{M}(\mu_A^1 + \mu_A^2 + \cdots + \mu_A^M)$$

$$\hat{\mu_B}_{eap} = \frac{1}{M}\sum \mu_B^i = \frac{1}{M}(\mu_B^1 + \mu_B^2 + \cdots + \mu_B^M)$$

$$\hat{\sigma}_{eap} = \frac{1}{M}\sum \sigma^i = \frac{1}{M}(\sigma^1 + \sigma^2 + \cdots + \sigma^M)$$

*1 ここで上つきの数字は MCMC のステップ番号を表しているもので，冪乗の数字ではありません。また M は Stan のデフォルトでは 4000 です。

さて, 今回は得られたデータ $X_{i,.}$ から μ, σ を推定したわけですが, これはいわばデータ生成メカニズムの設定値を見つけたことと同じですね。例え話で考えるなら, こんな感じです。ある企業 A が製品 a を量産しているとします。ライバル会社 B が, この製品 a の類似品を作ろうと考えたとします。そこで B 社は A 社の製品 a を作っている製造機械と同じ型番の機械を購入します。この機械には製品を生成するに当たっていくつかの設定をしなければならないとします。アナログな例えで恐縮ですが, ツマミが 2 つ 3 つ付いていて, それをひねれば何らかの製品ができるといった感じです。でも A 社が製品 a をつくるのに, どういう設定にしているのかはわからない。そこで製品 a をいくつかサンプルとして入手します。入手したサンプル a1,a2... と見比べながら, この辺りかな？この辺りかな？とツマミを調節し, サンプル a1,a2... と同じような製品 b1,b2... ができるのはこの辺の設定だな, というのを見つけたという感じ。

この例え話を MCMC の用語を使っていうと, 入手したサンプル a1,a2... がデータです。このデータをつくる製造機械が**確率分布**です。機械には設定のツマミがついているんですが, そのツマミは**パラメータ**のメタファーになっています。サンプルを参考に, この辺りかな, この辺りかな, とツマミの値を探るステップが MCMC の各段階で, まずは $\mu_A^1, \mu_B^1, \sigma^1$, 次にそれからちょっと動かして $\mu_A^2, \mu_B^2, \sigma^2$, またちょっと動かして・・・と 4000 回試すことで「だいたいこのデータを作るための設定はこの辺り」というのが決まってくるわけですね。「この辺り」というのを点推定する場合は **EAP** や **MAP** などを使いますし, 幅を持って推定したい場合は**確信区間**で表現するのでした。

ということで, いくつかの手元のデータから, 製造メカニズムの設定を手に入れました。このとき B 社はきっと,「じゃあこの推定値をつかって (模造品である) 製品をジャンジャン作っていこう」となるでしょう。この推定値から作られる新しい製品 (模造品, 新しいデータとも言えます) のことを MCMC ではとくに**事後予測分布 (posterior predictive distribution)** と言います。分布となっているのは, 作られる製品も散らばりますので, その散らばり方を分布で考えるからです。

事後予測分布は, 点推定値を使って計算することもできます。つまり, データが $X_i \sim N(\mu, \sigma)$ というメカニズムで作られていて, それから推定値 $\hat{\mu}, \hat{\sigma}$ を得たわけですから, これを使って $X_{new} \sim N(\hat{\mu}, \hat{\sigma})$ とすれば良いのです。これはただの乱数生成でもありますから, たとえば R で `rnorm(mu,sigma)` とするときに `mu` と `sigma` に推定値を入れてやればいいでしょう。でも点推定値では決め打ちが過ぎますので, 区間推定した上限と下限も入れますか？いや, それならいっそ, MCMC で得られた事後分布からの代表値を全部入れてしまえばいいのではないでしょうか。

それを実現するために,`generated quantities` ブロックを使うといいでしょう。コード 21.1 にその例を書いてみました。データやモデルは前回の二群の平均値差の検定と同じものです。

code : 21.1　事後予測分布を作るコード

```
data{
    int<lower=0> N1; // Number of Subjects in Group 1
    int<lower=0> N2; // Number of Subjects in Group 2
    array[N1] X1; // Data in Group 1
    array[N2] X2; // Data in Group 2
}

parameters{
    real mu1;
    real mu2;
    real<lower=0> sig1;
    real<lower=0> sig2;
}

```

```
model{
    // likelihood
    X1 ~ normal(mu1,sig1);
    X2 ~ normal(mu2,sig2);
    // prior
    mu1 ~ uniform(0,100);
    mu2 ~ uniform(0,100);
    sig1 ~ cauchy(0,5);
    sig2 ~ cauchy(0,5);
}

generated quantities{
    real Xpred1[N1];
    real Xpred2[N2];
    for(i in 1:N1){
        Xpred1[i] = normal_rng(mu1,sig1);
    }
    for(i in 1:N2){
        Xpred2[i] = normal_rng(mu2,sig2);
    }
}
```

ここで，generated quantities には，normal_rng という関数が入っています。この_rng は確率分布の後ろにつけて，乱数を発生させろという意味です。R でいう r****みたいなもんですね。このコードでは各群と同じサイズだけの事後予測データセットを作っています。まあ同じ μ と σ からの乱数ですから，データの数だけ作ったりしなくてもいいんですがイメージしやすくするためにそうしてみました。この μ と Xpred を図にしてみたのが図 21.1 です。この図の左側は μ_1 の**事後分布**が，右側には X_1 の**事後予測分布**が描かれ

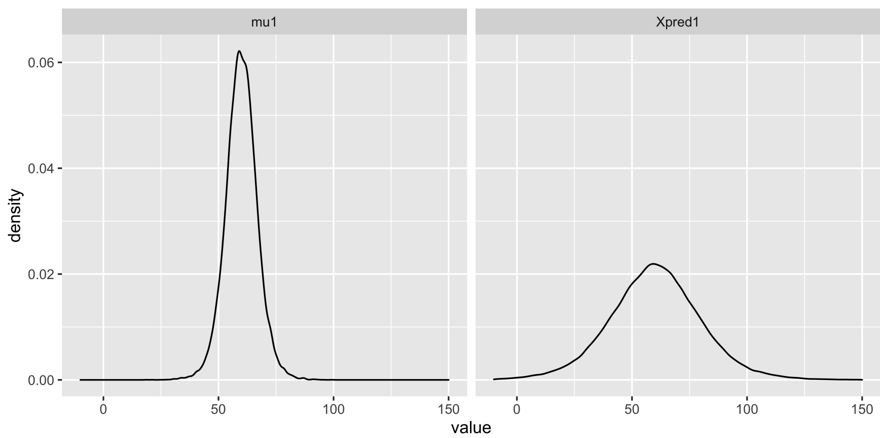


図 21.1　事後分布と事後予測分布

ています。左側は**パラメータの世界**の話であり，右側は**データの世界**の話になっていることに注意してください。

パラメータ μ_1 は，EAP で 59.96,95%CI[48.81,71.22] となっています。そこからでてくるであろうデータは，25 から 100 弱の幅に分布していますね。平均が 60 であっても，SD が 20 弱ありますから，データのレベルではその散らばる幅が大きくなるのも当然ですよね。でもこれはとても大事なことで，我々は「群間に差があった」となるとその効果について色々考えますが，それはパラメータの話，あるいは**平均因果効果**の話で

あって，個々のデータではまた事情が違うのです。たとえば，男女差があると言っても平均的な男性，そうでない女性などさまざまなケースがあるように。たとえば，平均的に 80% 成功する手術があると言われても，自分の身に置き換えると生きるか死ぬかは割合の問題ではないように。

また，この事後予測分布の形は，もとのデータの分布の形と似ているかどうかを視覚的に確認するために使われます。図 21.2 に，事後予測分布ともとのデータのヒストグラムを合わせて表示してみました。事後予測分布は MCMC サンプルの数だけありますが，ここでは一部だけ取り出しています。これを見ることで，もと

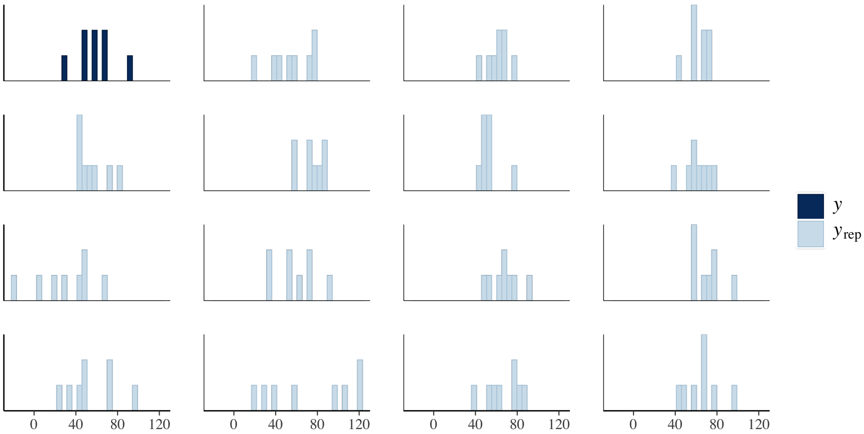


図 21.2　データの分布と事後予測分布の視覚的な確認

のデータと事後予測分布の形が似ているかどうかがわかります。もし模倣した機械の設定値が正しいのであれば，手元にある商品サンプルと同じような分布をするデータ分布が得られるはずですね。視覚的な確認ですから，見て「うーん似てるな，似てないな」というのを見て考えるだけではあります。今回はデータの数が 8 件と少ないので何ともイメージしにくいところですが，データが多くなるとデータの分布の形状で比較することもしやすくなるでしょう。もし分布の形が似ていなければ，購入した製造機械が違うものだったのかもしれない，ということになります。というのも実際のデータ分析の場合は，製造機械 (確率分布，モデル) が正しいかどうかはわからないので，それすらも仮定，検証すべき対象だからです。

21.2　データレベルの仮説

事後予測分布の利用方法は他にもあります。事後予測分布は，確率モデルと推定されたパラメータを利用した「新しいデータの生成」だったわけですが，この新しく生まれるであろうデータについての，データの世界での仮説を考えることができるのです。

帰無仮説検定で扱っていたのは，パラメータの世界の仮説でした。たとえば二群の平均値差の検定について言えば，帰無仮説は $\mu_1 = \mu_2$ であり，対立仮説は $\mu_1 \neq \mu_2$ です。これはギリシア文字 μ を使って書いてあることから明らかな通り，母数の仮説であり，$\bar{X}_1 = \bar{X}_2$ のような，データについての仮説ではありませんね。ではデータについての仮説というのはどういうものがあり得るでしょうか？たとえば次のようなものが考えられます。

- 無作為に選んだ統制群のデータと実験群のデータを比べたとき，実験群が統制群を上回っている確率はどれぐらいあるだろうか？
- 無作為に選んだ統制群のデータと実験群のデータの差が，基準点 c より大きくなる可能性はどれぐらいあるだろうか？

第 1 の点は**優越率 (probability of dominance)** という指標です。これはパラメータの世界の比較ではなく，実データの比較ですからイメージしやすいのではないでしょうか。たとえば男女差が平均的にはあるとされていたとしても，それはパラメータの違いであって，実際これから新しく出会う男性 (女性) が自分より優れている (劣っている) 可能性は，と考えた方が実質的な意味がありそうです。性差の話の多くは，平均値差だけでしか議論されませんが，効果量[*2]で考えると小さいことが少なくありません。それでも有意差が検出されてしまうのは，非常に大きなサンプルサイズをとっているから，ということもあったりします。サンプルサイズを大きくとるのは研究者の工夫や努力なのですが，小さな効果を取り上げて針小棒大に騒ぎ立てるのは科学的に良い態度とは言えません。そんな時に優越率で考えてみると，差があると言っても無作為に二群から新しい情報を得た場合の優越率が 50% ちょっとしかない，つまり超えるか超えないかが半々ぐらいですから「何の情報ももたらさない」のと同じこと，ということになったりします。くどいようですが，実データのレベルでの比較になるので，群間の差の表現としてよりわかりやすいということが言えるでしょう。第 2 の点は，優越率に一定の違いの差 c を含めて考えるもので，**閾上率 (probability beyond threshold)** といいます。これもデータレベルでの仮説の 1 つで，たとえば新しいトレーニング方法を使えばタイムが 3 秒小さくなる確率は X%，などということが分かれば具体的にイメージしやすい伝達方法になると思いませんか。

ちなみにこの優越率，閾上率などの指標は古くから指摘されてきていたものであり[*3]，ベイズ推定を使わない，**モーメント法**による推定でも計算可能な量でした。まぁしかし，このような統計量が実際に使われているケースってほぼ見かけないんですけどね。しかし，これらの指標はベイズ推定を使うことによって，というより乱数による近似計算をすることによって，遥かにイメージしやすくなっています。ここで示された数値はいずれも，`generated quantities` ブロックの中に `if` 文をつかって書いてやれば表現できることだからです！

実際に考えてみましょう。これら「条件が成立する確率」については，成立すれば 1，成立しなければ 0 というフラグをたてて MCMC サンプルを生成し，その平均値すなわち相対頻度で考えてやれば良いのでした。

code : 21.2　優越率や閾上率を計算するコード

```
data{
    int<lower=0> N1; // Number of Subjects in Group 1
    int<lower=0> N2; // Number of Subjects in Group 2
    array[N1] real X1; // Data in Group 1
    array[N2] real X2; // Data in Group 2
    int<lower=0> C; //constant
}

...

generated quantities{
    real Xpred1;
    real Xpred2;
    int<lower=0,upper=1> FLG1;
    int<lower=0,upper=1> FLG2;
    Xpred1 = normal_rng(mu1,sig1);
    Xpred2 = normal_rng(mu2,sig2);
    //probability of dominance
    if(Xpred1 > Xpred2){
        FLG1 = 1;
    }else{
```

*2 標準化された平均値差のことでしたね。

*3 たとえば南風原・芝 (1987) による優越率の解説は 1987 年の論文です。

```
            FLG1 = 0;
        }
        //probability beyond threshold
        if(Xpred1 - Xpred2 > C ){
            FLG2 = 1;
        }else{
            FLG2 = 0;
        }
    }
```

コード 21.2 は，generated quantities ブロックで 2 つの FLG 変数を用意しています。FLG1 が優越率，FLG2 が閾上率のために用意したものです。このブロックで，推定されたパラメータ mu1,mu2,sig1,sig2 からそれぞれ生成される二群の「新しいデータ」を作り (Xpred1,Xpred2)，成立するかどうかを検証します。FLG1 は if 文の中にあるように，Xpred1 > Xpred2 が成立すれば 1，しなければ 0 になる数字です。FLG2 は，同じくその差分が一定の値 C を超えていれば成立，超えていなければ不成立というフラグです。ちなみにこの値 C は data ブロックで外部から読み込むことにしてあります。

　これと前回のデータを使って推定させてみましょう。

code : 21.3　優越率，閾上率を推定するコード

```
groupA <- c(30, 50, 70, 90, 60, 50, 70, 60)
groupB <- c(20, 40, 60, 40, 40, 50, 40, 30)
dataSet <- list(X1 = groupA, X2 = groupB, N1 = 8, N2 = 8, C = 3)
#### rstanの場合
sampling1 <- sampling(modelR, dataSet, warmup = 1000, iter = 4000, chains = 4)

### cmdstanrの場合
sampling2 <- modelC$sample(
  data = dataSet,
  chains = 4,
  iter_sampling = 5000,
  iter_warmup = 1000,
  parallel_chains = 4
)
```

MCMC の結果 4

```
# A tibble: 8 × 7
  name          EAP        MED        MAP         SD        L95        U95
  <chr>   <num:.3!>  <num:.3!>  <num:.3!>  <num:.3!>  <num:.3!>  <num:.3!>
1 FLG1        0.800      1.000      0.999      0.400      0.000      1.000
2 FLG2        0.764      1.000      1.001      0.425      0.000      1.000
3 mu1        59.933     59.936     59.833      6.914     46.147     73.895
4 mu2        40.006     40.002     40.141      4.656     30.869     49.362
5 sig1       18.598     17.495     15.846      5.480     11.310     32.054
6 sig2       12.640     11.941     11.071      3.725      7.623     21.903
7 Xpred1     59.880     59.828     58.098     20.590     18.660    100.984
8 Xpred2     39.887     39.978     40.457     13.984     11.642     67.691
```

　ここにあるように，優越率は 80.0%，閾上率は 76.4% ということになりました。実験群 (GroupA) のほうが

統制群 (GroupB) よりもスコアが高くなる確率が 80%, スコアが 3 点以上大きくなる確率は 76% もあるのですから，実験群の効果は結構おおきいぞ，といったことがわかるようになります。

こうしたデータに基づく仮説は，単位がもとのデータに対応しているから想像しやすく，より実践的な意味を想像しやすいですね。

まとめておきます。パラメータの世界に関する仮説は，推定されたパラメータはどれぐらいの値なのか，どれぐらいの「差」「違い」があるのかを考えることでした。あるいは，推定されたパラメータはどれぐらいの幅に分布するのか，という差の分布を考えることで，効果の有無の確からしいさを量的に評価できるのでした。データの世界に関する仮説は，データ生成機がどういうデータを作るのか，ひいてはそれが今のデータと同じ形をしているかを見ることで，モデルが正しいかどうかを検証できます。また未来のデータの分布はどんな形になるのか，という意味では，データという具体的な数字でもって未来予測をできます。

ただしいずれの世界においても注意しておいて欲しいのは，データ生成機すなわち確率モデルが正しい場合であり，これが間違っていると `generated quantities` などで作られる数字もまったく意味のないものになってしまう，ということです。もちろん **t 検定**など帰無仮説検定を行う時も，データが正規分布していなければ意味のない検定結果になるので，「そもそも統計的な仮定が間違ってた」というのでは意味がないのはいずれも同じです。推測統計学は手元の少しのデータから，未知の世界を予測検証するという不良設定問題に対応するためのものですから，仮定や前提が正しいかどうかについては常に敏感でなければなりません。

21.3 パラメータ・リカバリ

そもそもの仮定が間違っていた，というのを考えすぎて何もできなくなっても良くないですから，そこはいったん OK だとしましょう。そう考えると，仮定・前提をつけた上での話になりますが，未知の世界や未来を予測できるというのは，とんでもなく強力なツールであるような気がします。株価や競馬の予想ができれば，大儲けできる薔薇色の人生が待っているかも。ということで，ベイズ推定の勉強にも身が入るというものです。推定を可能にしてくれた MCMC に感謝！

半ば冗談，半ば本気なのですが，Stan や JAGS などの**確率的プログラミング言語 (stocastic programming language)** が登場したおかげで，事後分布の計算が可能に，簡単便利にできるようになったというのは大変ありがたいことです。これのおかげでどんなモデルでも，設計図が書ければ答えが出せちゃう。複雑な確率の式とか微分積分を考えなくても，なんとかなるような気がします。しかし簡単なツールが出てきたときの常で，ちょっと注意が必要なのですが，MCMC は優秀すぎて正しくない計算式からでも答えが出てしまう，という可能性があるのです。

MCMC がうまく行ったかどうかは，**Rhat** や**有効サンプルサイズ**，**トレースプロット**などをチェックすればよい，というのはすでにお話しした通りです。しかしここで指摘したいのはそうではなく，これらの MCMC がうまく行っていたとしても，推定結果が正しいとは限らない可能性です。MCMC は「あり得そうな答えの可能性」を大量に出力し，その分布で我々は考えるのですが，その答えの可能性群が全部的外れなところに行ってるかもしれないのです。実際にデータを分析していると感じるのですが，MCMC はかなり複雑なモデルを考えても答えを出してくれます。とにかく答えてくれる機械が手に入ったとしても，その信憑性をチェックすることを忘れてはいけません。

ではこれをチェックするためにはどうすればよいでしょうか。ひとつは，答えが先にわかっている問題を作って，機械が正解を当てられるかどうかを検証するということが考えられます。この「わかっている正しい値」を当てることができるのであれば，実際のデータを用いた「わからない値」を推測する方法として有用だということがわかります。正解を当てることができないようであれば，この計算機械は当てにならないので，実際のデータを使っても正しく推測できているとは言えないでしょう。

この検証方法のことを**パラメータリカバリ** (parameter recovery) と言います。実際のデータ分析をする場合は，いきなり未知の答えに推測機をあてがうのではなく，推測機が事前に設定したパラメータをリカバリ (回復) できることを確認してから利用するようにしましょう。実際の研究実践の場合，その分析手順は，以下のようになります。

1. データ生成メカニズムを考える (紙とペンによる設計図の作成)
2. モデルを式で表現する
3. モデルの式からデータ生成をシミュレーション
4. Stan でコードを書く
5. シミュレーションデータを Stan が再現していることを確認する
6. 実際のデータでやってみる
7. 結果の解釈（図による確認も）

ここで第 3 のステップ，仮想データの生成から第 5 のステップ再現の確認が，パラメータリカバリという作業になります。推定したい平均値などのパラメータを事前に仮に入れてみて，その答えが復元できるかどうかを見るのですね。

なんだかまだるっこしい！と思う人もいるかもしれません。自分で設定したパラメタ通りになるのって当たり前じゃないか，何がおもしろいのか？というわけですね。ですが意外と当たり前でないことがあるんです。間違えた答えであれば，実践では役に立ちません。あるいは，いきなりデータでいいじゃない，固いこと言うなよ，と思う人もいるかもしれません。これが設定ミスの場合は答えを出さない・答えられないという機械であればそれでもいいのかもしれませんが，MCMC は大概なんでも答えてしまうので，いきなりの運用は怖いと思いませんか。またあるいは，基礎実験とかそのほかの授業でも，今までそんなのやったことないよ，という意見もあるかもしれません。大学の授業で行うような演習・実習は，そもそも効果がはっきりあること，再現する現象であることがある程度確からしいものを選んでやっていますからいいのですが，これから皆さんは研究者の卵として，新しい現象，十分に確認されていない現象を研究対象にすることになります。そのとき，いきなり出てきた結果がどれほど信用できるでしょうか。実際，最近では心理学実験が再現できないケースも数多く指摘されており，その理由のひとつとして統計モデルの乱用 (悪用・誤用) にあったことは何度もお伝えしてきた通りです。さらに，ごく微小な効果を偶然検出してしまった，という可能性もあります。事前に推定機の精度がわかっていればこうした問題を回避できるのです。この話はベイズ推定をする場合だけでなく，帰無仮説検定など従来のやり方にも通じ，**効果量**や**例数設計**の話に直結します。

21.3.1 二群の平均値差の例

ここまで扱ってきた，二群の平均値差を検証する例で考えてみましょう。もともとの設計図 (図 20.1) がここでも役に立ちます。データがどういうメカニズムで出てきたかを考えていたわけですから，これを使えばデータ生成を考えることができますね。乱数生成のアプローチはデータを取り始める前にも有用なのです！

今回はパラメータを仮に設定してやることができます。二群の平均とその差を次のように決めてしまいましょう。

code : 21.4　平均値差の設定

```
mu1 <- 50
diff <- 10
mu2 <- mu1 + diff
sig1 <- 5
sig2 <- 8
```

第一の群の平均は `mu1 <- 50` としています。第二群の平均は `mu2 <- 60` というようにしてもいいのですが，差を考えるという意味を強調するためにいったん変数 `diff` を作って差の量を定め，それを `mu1` に加えるという方法をとっています。2 つの群は正規分布に従いますから，それぞれの標準偏差を `sig1,sig2` として設定しました。ここから仮想データを作ります。データの生成は乱数に従うので，`rnorm` 関数の出番です。

code : 21.5　仮想データの生成

```r
set.seed(12345)
N <- 10
X1 <- rnorm(N, mu1, sig1)
X2 <- rnorm(N, mu2, sig2)
```

これで実際に生成された仮想データは次のようになっています。

R の出力 21.1: 検定の結果

```
> X1
[1] 52.92764 53.54733 49.45348 47.73251 53.02944 40.91022 53.15049
[8] 48.61908 48.57920 45.40339
> X2
[1] 59.07002 74.53850 62.96502 64.16173 53.99574 66.53520 52.90914
[8] 57.34738 68.96570 62.38979
```

ちなみにこの 2 つのデータセット，平均は $\bar{X}_1 = 49.33528, \bar{X}_2 = 62.28782$ となっています。理論的にはそれぞれ $50, 60$ であるはずなのですが，乱数による実現値なので少し誤差が出ていますね。これは実際の研究状況でもある話です。理論的な値と，目の前の実現値というのは必ず少しずれがあるもの。その中で，きちんと理論値を推定できるかどうかが問題なのですから。さて，このデータを使って分析していきましょう。

code : 21.6　パラメータリカバリ·検証

```r
## t検定(モーメント法による推定と判定)
t.test(X1,X2)
## ベイズ推定(MCMCによるベイズ推定と差の分布)
dataSet <- list(X1 = X1, X2 = X2, N1 = N, N2 = N, C = 3)
modelC <- cmdstanr::cmdstan_model("ttest03.stan")
sampling2 <- modelC$sample(
  data = dataSet,
  chains = 4,
  iter_sampling = 5000,
  iter_warmup = 1000,
  parallel_chains = 4
)
```

まずは t 検定の結果ですが，ここでは $t(14.797) = 5.2059, p = 0.0001113$ で有意差ありと判定されました。実際 `diff <- 10` で差がある設定なのですから，正しく判定できて良かったな，というところです。ベイズ推定の方はどうでしょうか。結果は次のようになりました (ここでは前回のコード 20.6 を再利用しました)。

MCMC の結果 5

```
# A tibble: 6 × 7
  name         EAP         MED         MAP          SD         L95         U95
  <chr> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 diff     -12.936     -12.957     -13.250       2.762     -18.458      -7.445
2 FLG        0.000       0.000      -0.001       0.000       0.000       0.000
3 mu1       49.349      49.350      49.499       1.477      46.381      52.302
4 mu2       62.285      62.285      62.256       2.355      57.600      66.977
5 sig1       4.482       4.257       3.868       1.194       2.842       7.406
6 sig2       7.189       6.870       6.363       1.821       4.615      11.579
```

これを見ると，mu1 <- 50 と設定したときの EAP 推定値が 49.349，95%CI で [46.381, 52.302] ですから，ほぼ正しい点推定値，また確信区間の中に真値を含んでいます。推定はうまく行っている，ということです。mu2 についても同様で，真値が 60 であり，EAP 推定値が 62.285，95%CI で [57.600, 66.977] です。点推定値は真値から 2 点ほどずれていますが，95% の区間推定にすれば真値を含んでおり，推定はうまく行っているようです。標準偏差の推定値についても，それぞれうまく行っており，差についても EAP 推定値が-12.936，95%CI で [-18.458, -7.445] です[*4]。確かに確信区間に真値は含んでいるのですが，17.45 は真値 10 から少し外れすぎかな... というようなことがわかりますね。

このように，今回の結果をみると概ね正しく推定できていると言えるでしょう。その精度についても「どの程度誤差が生じうるものなのか」が具体的にわかったことと思います。しかし次のような設定の場合はどうでしょう。

code : 21.7　仮想データの生成その 2

```
mu1 <- 50
diff <- 18
mu2 <- mu1 + diff
sig1 <- 10
sig2 <- 15
N <- 3
```

この設定でデータを作って，推定した結果は次のようなものです (出力 6)。

MCMC の結果 6

```
# A tibble: 6 × 7
  name         EAP         MED         MAP          SD         L95         U95
  <chr> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 diff      -5.434      -5.630      -5.104      12.855     -31.789      21.966
2 FLG        0.171       0.000       0.000       0.376       0.000       1.000
3 mu1       53.962      53.972      54.117       4.147      45.756      62.294
4 mu2       59.396      59.671      59.608      12.124      32.811      84.278
5 sig1       6.194       5.034       3.778       4.332       2.356      17.443
6 sig2      20.651      17.148      13.108      12.966       8.682      53.683
```

MCMC の収束は問題なく，推定値は出力されていますが，本来 $\mu_1 = 50$ であるところが 53.962，95% 確信区間は [45.756 ,62.294] であり，$\mu_2 = 68$ であるべきところに至っては EAP 推定値が 59.396，95%

[*4] 生成量は diff = mu1 - mu2 で作っているので，正負が逆転しているのはご容赦ください。

確信区間は [32.811 ,84.278] です。確信区間はなんとか平均値を挟み込みましたが，38 から 84 までと 46 点も幅がある推定はあまり有効な予測とは言えませんね。差は 18 あるという設定なのですが，EAP 推定値はわずか 5.434 であり，95% 確信区間は [-31.789, 21.966] です。確信区間にゼロをふくんでいますから，「差がないかも」ということになります。実際，t 検定の結果は帰無仮説を棄却できませんでした ($t(2.2337) = 0.52828, p = 0.6451$)。**タイプ 2 エラー (Type II Error)** を犯してしまっています。

どうしてこんなにうまくいかなかったのでしょうか。今回は正規分布という正しい確率モデルを使っていたのに，です。その答えは明らかに，`N <- 3` という点にあります。つまり，データが 3 件しかないので予測の精度が悪くなったのです。このように，モデルが正しくても推定精度が十分ではない，あるいは判定をするときに間違った判断をする可能性がある，ということがわかります。推定精度や効果の判定については，サンプルサイズ，データの散らばり (群内分散)，データ間の差の大きさ (群間分散) などの要因が影響してきます。ここで設定値を色々変えてみて，どういう関係にあるのかをみておくのも良いでしょう。

このように仮想データを使ったパラメタリカバリを通じて，推定や検定の精度を設計したりチェックしたりできることがわかりました。これを踏まえて考えると，前回の例では $N = 8$ のデータで分析をしましたが，$N = 8$ だと平均値差を 8〜12 点ほど外した予測をしてしまうかもしれない，ということがわかります。逆に「5 点差までの推定誤差に収めたい」というような希望があった場合，データは何件ぐらい取れば良いでしょうか？このような問いについても，N の設定値をさまざまに変えてみることで，実験や調査を実際に始める前に予想を立てておくことができます。これが**例数設計**です。例数設計の重要さを確認するとともに，こうした乱数を作るアプローチで簡単に実践できますので，実際の研究の前にはぜひ一度試してみてください。

21.4 今回のまとめ

生成量をつかうことで，次のような検証をできるのでした。

- 事後予測分布を作ることで，想定した確率モデルが正しかったかどうか検証する。
- 事後予測分布を利用して，優越率や閾上率などデータに基づく仮説を検証する。

さらに，データ生成モデルを応用して，そもそも擬似データをつくることで推定モデルの精度や，どれぐらいのサンプルサイズが必要かといったことを，実践の前に検証できます。

データ生成アプローチと乱数発生技術の組み合わせで，より慎重かつ確実な研究アプローチができることを理解してください。

21.5 課題

次の計算をする R/Stan コードや回答を記述し，提出してください。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

■フライドポテトの研究その 2　あるコンビニチェーンの 2 つのお店 A,B で，それぞれホットスナックのフレンチフライを買ったところ，どうも一方より他方の方が長くて大きい気がした。そこでそれぞれの袋から取り出して長さを測定してみた (単位 cm)。測定結果が表 21.1 である[*5]。このデータを用いて次の問に答えなさい。なおポテトの長さは正規分布に従い，また両群の分布の幅は同じであると仮定します。ちなみに，この数値はシラバスのサイト上のコードで取得可能です。

[*5] このデータは実際に筆者がコンビニの二つの店舗でポテトを購入し，長さを測定した結果に基づいています。

表 21.1 二つの店舗のポテトの長さ

A	8.4	11.3	8.1	11.2	5.8	6.3	7.1	10.9	7.1	6.5	5.0	3.0	7.2	6.5	6.4	6.4	9.3	8.3
B	6.7	7.2	4.2	11.0	7.5	8.9	7.0	8.0	7.2	4.2	6.0	9.0	8.6	9.0	5.0			

1. 店舗 A のポテトの平均が, 店舗 B のポテトの平均より長い確率はどれぐらいあるか。
2. 今後, 店舗 A で購入するポテトが店舗 B で購入するポテトよりも長い確率はどれぐらいあるか。
3. 今後, 店舗 A で購入するポテトが, 店舗 B で購入するポテトよりも 2cm 以上ながい確率が 30% あれば店舗 B では買わないようにしようと思う。店舗 B を利用すべきかどうか, 推定に基づいて判定してください。
4. 今後, 店舗 A で購入するポテトと店舗 B で購入するポテトの長さの違いが 5cm 未満である確率が 80% であれば, 家により近い店舗 B を利用してもいいように思う。店舗 B を利用すべきかどうか, 推定に基づいて判定してください。
5. このコンビニチェーンのポテトの長さ平均を, できるだけ正確に推定したいと思う。標準偏差は今回の MAP 推定値を使うこととして, 何本ぐらいポテトのサンプルを集めればその 95% 確信区間を 0.05cm 以内に収めることができるだろうか。シミュレーション結果に基づき, おおよその数字を答えてください。

第 22 章

モデリングの目から見た検定 3；多群の平均値差モデル

22.1　要因計画モデル

　さてここまでは二水準の平均値差を考えるモデルでしたが，ここからそれを三水準に増やしてみましょう。水準数が多くなると**要因計画** (**Factorial Design**) や**実験計画** (**Experimental Design**) といった話になってきます。**分散分析** (**ANalysis Of VAriance;ANOVA**) は要因計画と帰無仮説検定の合わせ技で，効果がある/ないの判定を分散の比をつかって行うことでした[*1]。

　今回は要因計画をデータ生成モデルから考え，**ベイズ推定** (**Bayesian Inference**) をすることになります。**帰無仮説検定**のような Yes/No 判断はせず，平均値や差の大きさを直接見積もることになるのは，二群の時と同じです。

　まずは簡単な**群間計画** (**Between Design**) の話から進めましょう。対応のない二群の時のように，それが三群になるモデルですので，モデルの設計図は非常に単純です (図 22.1)。設計図を見れば，コードはそ

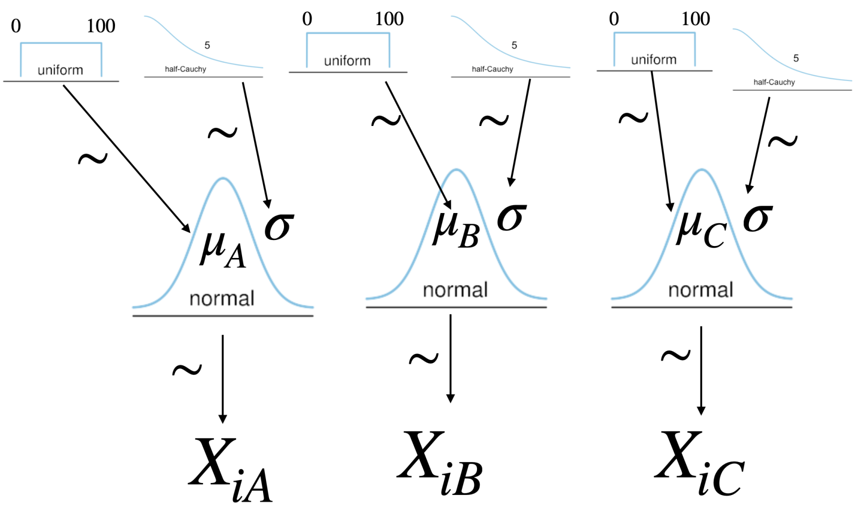


図 22.1　多群の平均値の差を比べるときの設計図

*1 忘れた人は，データ解析基礎のテキストに戻って確認してください。

のまま置き換える形でかけますね。設計図の通りに書いてみたコードが code:22.1 にあります。

code : 22.1　三群の平均値のコード

```
data{
  int<lower=0> N1;
  int<lower=0> N2;
  int<lower=0> N3;
  array[N1] real X1;
  array[N2] real X2;
  array[N3] real X3;
}

parameters{
  real mu1;
  real mu2;
  real mu3;
  real<lower=0> sig;
}

model{
  // likelihood
  X1 ~ normal(mu1,sig);
  X2 ~ normal(mu2,sig);
  X3 ~ normal(mu3,sig);
  // prior
  mu1 ~ uniform(0,100);
  mu2 ~ uniform(0,100);
  mu3 ~ uniform(0,100);
  sig ~ cauchy(0,5);
}

generated quantities{
  real diff12;
  real diff13;
  real diff23;

  diff12 = mu1 - mu2;
  diff13 = mu1 - mu3;
  diff23 = mu2 - mu3;
}
```

最後の generated quantities ブロックにあるように，各群の差の分布も計算して出せるようにしてあります。これをみながら，群 A と B，A と C，B と C などどこに差があるのか，その大きさはどれぐらいかということを考えることができます。左の大きさを標準偏差 sig で割ることで効果量を出すこともできます。

二群の平均値差の時もそうでしたが，検定と違って差の大きさを直接検証できているところがポイントです。また，検定の場合は分析によって有意差が確認できた場合，そのあと**多重比較 (multiple comparison)** へと進むのでした。これは帰無仮説が $H_0 : \mu_1 = \mu_2 = \mu_3$ となっていて，これを否定することから「どこに差があったか」をさらに詳しくみる必要があったからです。この時，統計的検定の繰り返しによって**タイプ 1 エラー (Type I Error)** が増えること，すなわち $\alpha = 0.05$ と 5% 水準で検定していても，検定を繰り返すと 5% を大きく上回る問題が生じるため，有意水準を調節するなど工夫が必要だという話で

した[*2]。

ところが，ベイズ推定の場合はこうした問題が生じません。なぜなら，確率を伴う判断をしていないからです。帰無仮説検定の場合は，「帰無仮説が正しいという条件のもとで」計算された検定統計量 (F 値や t 値) が，帰無仮説の条件下ではどれほど生じやすいかということを考え，その確率 p 値を計算していましたが，この p 値はパラメータがどこにあるかといった確率ではありません。判断のための指針としての確率に過ぎないのです。一方ベイズ推定する場合，推定されたパラメータの**事後分布**は，パラメータがこの辺りにあるだろう，という確率であり，生成量である差の分布もさの大きさがこれぐらいだろうという範囲，大きさに直接関わる確率です。ですから何 % 区間で考えてもいいですし，その判断が間違うかどうかの確率というのは問題に含まれないのです[*3]。検定の多重性の問題が生じない，というのは複雑な要因計画を考える場合には非常に助かりますね。

22.2 パラメータの変形と制約

22.2.1 パラメータの変形

ところで，先ほどは差の分布を `generated quantities` ブロックで表現しましたが，違う表現も可能です。差分も未知なるパラメータと考えて推定するのです。

順を追って説明しましょう。まず，最初の群の平均値を μ_1 とします。第二の群平均 μ_2 は，μ_1 と δ_1 だけ違う，つまり $\mu_2 = \mu_1 + \delta_1$ と考えるのです[*4]。同様に，第三の群平均 μ_3 は $\mu_3 = \mu_1 + \delta_2$ と考えます。こうすると，推定するべきパラメータは $\mu_1, \delta_1, \delta_2$ であり，`generated quantities` ブロックを使わなくても，直接差分パラメータを推定できるようになります。これを書いてみたのが，code:22.2 になります。

code：22.2　差分を直接推定するコード

```
...
parameters{
  real mu1;
  real delta1;
  real delta2;
  real<lower=0> sig;
}

model{
  // likelihood
  X1 ~ normal(mu1,sig);
  X2 ~ normal(mu1+delta1,sig);
  X3 ~ normal(mu1+delta2,sig);
  // prior
  mu1 ~ uniform(0,100);
  delta1 ~ normal(0,100);
```

[*2] お前は何をいってるんだ，と思った人は，データ解析基礎のテキストに戻って確認しておいてください。大雑把に解説しますと，5% 水準というのは一回の判断で間違いが含まれる可能性を 5% にしましょう，という判定基準のことでしたが，二回判定を繰り返すと $1-(0.95^2)=0.0975$ と 9.75% 水準になってしまうということでした。3 群の平均値を比べるときは，3 箇所比較する必要があり，**t 検定**を 3 回も繰り返すと 5% 水準ではなくなる，という問題です。

[*3] もっとも，すべて研究者の仮定したモデルのもとで，という話ではありますから，その仮定が間違っていたら全部意味のない数字である，ということにはなるかと思います。とはいえ，帰無仮説検定もデータが正規分布に従うという仮定で行うのですから，これについてはどっこいどっこい，というところでしょうか。

[*4] ここで δ はデルタと読む，ギリシア文字の d のことです。先ほどは生成量として `diff` と書いていましたが，ここでは未知のパラメータなのでギリシア文字に書き換えました。

```
  delta2 ~ normal(0,100);
  sig ~ cauchy(0,5);
}
```

data ブロックに違いはありませんので省略してあります。変更点として，まず parameters ブロックで推定するべきパラメータが変わっていることを確認してください。次に model ブロックで，3 つのデータがそれぞれ normal(mu1,sig),normal(mu1+delta1,sig),normal(mu1+delta2,sig) から出てきているように書き変わっています。正規分布の位置パラメータをずらすことで書き換えているのです。最後にその下，事前分布の設定ですが，差分 δ_1, δ_2 は正負どちらにも出てくる可能性があり，その大きさがわかりませんので，平均を 0 にした正規分布としてあります。これで推定した結果は，先ほどの生成量を使った結果と変わりはありません。数式の展開で書き方が変わっただけで，モデルの形は同じだからです。

ここで transformed parameters ブロックの紹介をしておきましょう。code:22.2 は悪くはないのですが，モデルのところ，normal(mu1+delta1,sigma) としているのがちょっと美しくないですね。元のコードにあった，normal(mu2,sigma) のほうがわかりやすいのに，と思った人もいると思います。まあ今の所，そこまで複雑ではないのでそれほど問題にはならないでしょうが，これからもしパラメータが複雑な構造をするようになったとき，尤度のところはもう少しスッキリ書きたいということも生じてくるでしょう。そういうときに transformed parameters ブロックを使います。このブロックは，parameters ブロックの次に書くもので，言葉の通りパラメータを変形 (トランスフォーム) するためのものです。

これを使って書いたのが code:22.3 です。

code : 22.3　パラメータ変換を入れたコード

```
...
parameters{
  real mu1;
  real delta1;
  real delta2;
  real<lower=0> sig;
}

transformed parameters{
    real mu2;
    real mu3;
    mu2 = mu1 + delta1;
    mu3 = mu1 + delta2;
}

model{
  // likelihood
  X1 ~ normal(mu1,sig);
  X2 ~ normal(mu2,sig);
  X3 ~ normal(mu3,sig);
  // prior
  mu1 ~ uniform(0,100);
  delta1 ~ normal(0,100);
  delta2 ~ normal(0,100);
  sig ~ cauchy(0,5);
}
```

ここで注目すべきは，まず transformed parameters ブロックで mu2,mu3 という変数名が宣言され，それが式によって表現されていることです。ここでパラメータ mu1,delta1 が mu2 に変形されていることがわかります。このようにして作られたパラメータであれば，model ブロックの中で使ってやることができます。現に，尤度のところが X2 ~ normal(mu2,sig) のようにスッキリした形になっていますね。ただし注意すべきは，事前分布のところが変わっていない，ということです。事前分布はパラメータに対しておかれますので，ここで mu2 ~ uniform(0,100); などと置くべきではありません。

このように変換することで，コードが多少なりともスッキリしたと思います。面倒だな，と思うかもしれませんが，コードがスッキリすることは自分にとっても誰にとっても「読みやすい」ことにつながります。そして読みやすいコードは，間違い (バグ) があったときに修正しやすいのです。あまりごちゃごちゃしたコードを書くと，問題が生じたときに対応が難しくなりますから，なるべくこうした作法を活用して，コードは単純明快なものにしておくと良いでしょう。

22.2.2 パラメータの制約

さてここでこのモデルをもう一段階，発展させてみましょう。今のコードは第一の群，μ_1 を基準に，μ_2, μ_3 を構成していました。別に第二，第三の群を基準にしても良かったですよね。つまり基準点は恣意的になっています。それで悪いということではないのですが，μ_1 だけ特別扱いです。そうではない作法として，全体平均 μ を考えて，次のように定式化してみましょう。

$$\mu_1 = \mu + \delta_1 \tag{22.1}$$
$$\mu_2 = \mu + \delta_2 \tag{22.2}$$
$$\mu_3 = \mu + \delta_3 \tag{22.3}$$

こうすることで，全体平均からの差分としてモデルを表現できます。ここで全体平均 μ はどういう意味があるでしょうか。群の違いを独立変数，結果のスコアを従属変数とした線形モデルの一環として考えると，全体平均は群を通じて変化のない水平線 (図 22.2) という関係を表していることになります。横一線で群ごとに違いがないのですから，いわばこれは**帰無仮説**のモデル，すなわち**帰無モデル (Null Model)** とでもいうべきものです。

あるいは，群ごとの平均値の違いはこの全体平均からの差分で表現されます。この群ごとの違いは，その群に入ったからこそ生じた**効果**であるとも言えます。つまり，δ は効果の大きさなのです。翻ってかんがえると，全体平均は「なんの処置もしなければ理論的にこの値になるはず」という操作なしの状態，天然の状態とでもいえるでしょう。**ランダム化比較実験 (Randomized Control Trial)** は，被験者をランダムに割り付けることによって，平均的に見れば個別の効果が相殺し合うことを用いています。平均化することで誤差が相殺するので群ごとの比較ができますし，群の平均値が変化したということは，その群に属したことの効果が現れたと考えるわけです。帰無モデルの状態は，この個別の効果を相殺させた状態であることを表しています。

さてこのように考えて，これを数式の形，モデルの形に書き起こしていくことを考えましょう。ところがこのままだと，未知のパラメータは $\mu, \delta_1, \delta_2, \delta_3$ と 4 つあることになります。今までは μ_1, μ_2, μ_3，あるいは $\mu_1, \delta_1, \delta_2$ の 3 つだったのに，です。同じ数式を書き直しただけなのに，パラメータの数が変わるのはおかしいですね。実はこれには 1 つ，制約が欠けているのです。今回は全体平均 μ からの相対比較になっていますから，$\delta_1 + \delta_2 + \delta_3 = 0$ という関係が隠れていることになります[*5]。そこで，この制約をモデルの中に書き込まな

[*5] もしこれが $= 0$ にならなければ，全体平均の計算に矛盾が生じますよね。全体平均 μ は，$\mu = \frac{1}{3}(\mu_1 + \mu_2 + \mu_3)$ ですから，

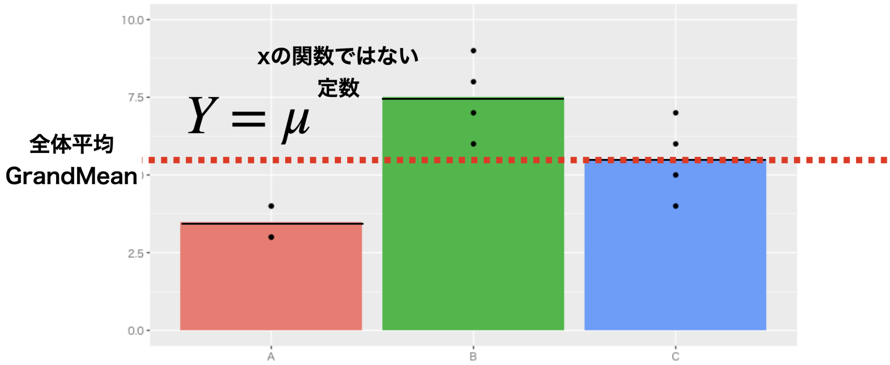


図 22.2　全体平均は水平線

ければなりません。$\delta_1 + \delta_2 + \delta_3 = 0$ という式から，変数を移項することでこれを表現します。すなわち $\delta_3 = 0 - (\delta_1 + \delta_2)$ のようにします。

これを使って書いたのが code:22.4 です。

code：22.4　制約を入れたコード

```
...
parameters{
  real gm;
  real delta1;
  real delta2;
  real<lower=0> sig;
}

transformed parameters{
    real delta3;
    real mu1;
    real mu2;
    real mu3;
    delta3 = 0 - (delta1 + delta2);
    mu1 = gm + delta1;
    mu2 = gm + delta2;
    mu3 = gm + delta3;
}
...
```

このようにすることで，先ほどと等価なモデルを別の形で書き表すことができるようになりました。

ここまでをまとめておきます。三群の平均値差を求めるモデルを，3 つの方法で書いてみました。

1. 3 つの群平均 μ_1, μ_2, μ_3 を個別に推定するモデル
2. ある群とそこからの差分，$\mu_1, \delta_1, \delta_2$ を推定するモデル
3. 全体平均とそこからの差分，$\mu, \delta_1, \delta_2, \delta_3$ を推定するモデル。ただし $\sum \delta = 0$ という制約をつける

$\mu = \frac{1}{3}(\mu + \delta_1 + \mu + \delta_2 + \mu + \delta_3) = \frac{1}{3}(3\mu + \sum \delta_j) = \frac{1}{3}3\mu + \frac{1}{3}\sum \delta_j$ となり，最初の項が $\frac{1}{3}3\mu = \mu$ ですから，第二の項は $\frac{1}{3}\sum \delta_j = 0$ でないと計算があいません！

この方法はいずれも同じ推定結果を返します。第一のモデルの生成量から差分を計算すれば，δ_1, δ_2 を計算したことと同じです。

同じことを異なる方法で実装するのは，生産的でないように思うかもしれません。しかし第 3 の形式にしておくと，モデルの一般化が容易になります。すなわち，多群の平均値差の比較をするときに，群の数が変わるたびにコードを書き換える手間がなくなるのです！

22.3 モデルの洗練

ここまで，二群，三群の平均値差を求めるモデルを書いてきました。書き方はわかっているので，比較する群が四群，五群と増えても書き足していくだけでなんとかなりそうです。

しかし，たとえば四群に増えたときに $\delta_1, \delta_2, \delta_3$, 五群に増えたときに $...\delta_3, \delta_4$ と手作業で増やしていると，N 群までふえたときは $...\delta_{N-2}, \delta_{N_1}$ と何行も書き足していかなければなりません。しかもその度にコンパイルしているようでは，とても面倒なことはすぐに想像がつきます。

そこで，群の数が変わっても同じコードで対応できるように，一般化を目指してコードを改良しましょう。そのためには，何群の比較なのかを示す群の数を，`data` ブロックで外部から取り込むようにすれば良いでしょう。その変数を使って差分の数，効果の数を計算してやれば良いのです。たとえば比較する群の数を `G` とすると，全体平均 μ と，$G-1$ 個の差分をパラメータとして推定すれば良いことになります。

その考え方を使って書いたコードは，たとえば code:22.5 のようになります。

code : 22.5　群の数を一般化したコード

```
data{
  int<lower=0> Lv;             // 水準数
  int<lower=0> N;              // 各群のサンプルサイズ
  array[Lv,N] real X;          // 変数の値
}

parameters{
  real gm;                     // 全体平均
  array[Lv-1] real raw_delta;  // 全体からの差。水準数マイナス1個
  real<lower=0> sig;           // 誤差の分散
}

transformed parameters{
  array[Lv] real delta;          // 差の大きさを作り直す
  array[Lv] real mu;             // 再構成される群ごとの平均

  for(i in 1:(Lv-1)){
    delta[i] = raw_delta[i];   // ほとんどコピー
  }

  delta[Lv] = 0 - sum(raw_delta); // 総和が0になるように最後だけ書き換える

  for(i in 1:Lv){
    mu[i] = gm + delta[i];       // 群ごとに再構成
  }

}

```

```
model{
  // Likelihood
  for(l in 1:Lv){
      for(i in 1:N){
          X[l,i] ~ normal(mu[l],sig);

      }
  }

  // Prior
  gm ~ uniform(0,100);
  raw_delta ~ normal(0,100);
  sig ~ cauchy(0,5);
}
```

■コード解説

data ブロック　群の数 Lv，各群のサンプルサイズ N，各データ点 X を入力します。ここで X が**二元配列**になっていることに注目してください。X[1,1] は第一群の第 1 番目のデータ，X[1,2] は第一群の第 2 番目のデータ，X[2,1] は第二群の第 1 番目のデータ，同様に X[i,j] は第 i 群の第 j 番目のデータを表すことになります。この形式は二元配列 (2 次元の変数セット) になっていると言います[*6]。宣言の時に，上で取り込んだ変数 Lv や N を使うことができます。最大 Lv 個の群，各群 N 人のデータがあるという前提です。

parameters ブロック　全体平均 gm と，差分，誤差の SD をパラメータとしています。差分はここでは raw_delta とし，上で宣言した Lv をつかって Lv-1 個の要素を持つ配列にしています。raw_delta って変な名前，と思うかもしれませんが，この後これを変換して δ を作りますので，作る前の生 (raw) の δ という名前にしてみました[*7]。

transformed parameters ブロック　以後の話をわかりやすくするために，各群の平均 mu を構成することにします。mu の数は水準数と同じ Lv です。そして第 i 群の平均 mu[i] を作るために，mu[i]=gm+delta[i] という書き方をしたいので，水準数と同じ数だけ delta[i] を作りたいと思います。もとの raw_delta が 1,2,...,Lv-1 個ありますので，delta も Lv-1 番目まではそれと同じものをコピーします (for 文で繰り返し代入していってます。)。そして最後に Lv 番目の要素を，0 からこれまでの要素をすべて足した (sum(raw_delta)) ものを引くことで作っています。ちなみに sum は stan の持っている総和の関数で，配列要素のすべてを足し合わせるというものです。こうして delta が Lv 個できあがりましたので，各水準の平均値を mu[i]=gm+delta[i] という数式で一般的に書くことができるようになりました。

model ブロック　尤度のところに，群それぞれについて巡回する for 文，その中で 1,2,3...N 人まで巡回する for 文を組み込んで二重にぐるぐると回しています。事前分布は設定の通りです。

このコードを使って，実際に推定してみましょう。データは表 22.1 のものを使います。
推定してみましょう (R コードは 22.6, 結果は 7)。各群の平均値，全体平均からの偏差などが，事後分布

[*6] 行列でもいいじゃないか，と思われるかもしれません。それでも結構です。Stan は行列の時，real ではなく matrix で宣言することができます。しかしこのように real 型にしておいて，後ろのカッコで配列の次元を表現すると，何次元でも拡張することができるので今回はそのようにしました。

[*7] 変数名は任意ですので，著者の命名法がわかりにくいなあと思ったら，自分で好きな変数名にしてもらって構いませんよ。その場合は，以後すべての変数を読み替えてくださいね。

表 22.1　独立した三群から得られたスコア

groupA	6	6	5	5
groupB	7	4	7	6
groupC	4	3	4	6

として出力されているのがわかると思います。これをみて，たとえば差の事後分布の 50% 確信区間でみると δ_2, δ_3 はその区間に 0 を挟んでないので，一定の効果はあるだろうなどと判断できるわけです。もちろん生成量をつかって，ある程度の差がある確率がどのくらいとか，優越率などデータレベルの仮説を立てることも可能です。

code : 22.6　三群の平均値の比較

```
Example <- matrix(c(6, 6, 5, 5, 7, 4, 7, 6, 4, 3, 4, 6),
                  ncol = 4, byrow = T)
dataSet <- list(Lv = 3, N = 4, X = Example)
```

MCMC の結果 7

```
# A tibble: 10 × 7
   name               EAP        MED        MAP         SD        L95        U95
   <chr>        <num:.3!>  <num:.3!>  <num:.3!>  <num:.3!>  <num:.3!>  <num:.3!>
 1 delta[1]         0.257      0.256      0.227      0.569     -0.880      1.394
 2 delta[2]         0.749      0.754      0.804      0.566     -0.387      1.878
 3 delta[3]        -1.005     -1.009     -1.114      0.576     -2.158      0.162
 4 gm               5.251      5.251      5.242      0.400      4.452      6.039
 5 mu[1]            5.507      5.507      5.562      0.699      4.114      6.902
 6 mu[2]            5.999      6.001      5.986      0.697      4.603      7.386
 7 mu[3]            4.246      4.238      4.182      0.695      2.860      5.647
 8 raw_delta[1]     0.257      0.256      0.227      0.569     -0.880      1.394
 9 raw_delta[2]     0.749      0.754      0.804      0.566     -0.387      1.878
10 sig              1.339      1.261      1.134      0.391      0.823      2.301
```

22.3.1　データサイズの一般化

さてこのモデルは，比較する群の数が増えても外部からデータとして群の数を与えることができますので，毎回コンパイルする必要がありません。やったね！これで完璧，と言いたいところですが，1 つ気になるのはサンプルサイズです。各群のサンプルサイズ N をデータとして与えるようになっていますが，たとえば A 群は 10 人，B 群は 12 人，C 群は 14 人と言ったように，群ごとにサンプルサイズが異なるとこのコードでは対応できません。実際に実験をやる場合，各群のサイズを整えて人を集めたいところですが，調査研究などでは群ごとに同じ人数を与えるような調整ができないこともあり，そういう意味ではこのコードではうまく対応できそうにありません。

そこで，群の人数が異なる場合でも対応できるように，さらにこのコードを拡張していきたいと思います。そのためにまず，データを**整然データ (tidy data)** の形に整形しましょう。

先ほどのサンプルデータ (表 22.1) は，3×4 の長方形をしていました。これはデータのサイズが整っているからできることで，サイズが変わってしまうと表の中に欠損ができてしまうことになります。また，たとえば B

群の 3 番目の人の値を見る時，われわれは 2 行 3 列目のデータにさっと目が行きますが，機械的には群の情報が行名に，人の群内整理番号が列名にあるので，1 つのデータを見る時に行･列それぞれを参照することになります (図 22.3 左)。ここで「どの群か」「群の中の何番目の人か」という情報はいずれも変数で，人によって変わるものですから，データの持つ情報の 1 つなのです。データの持つ情報，変数なのに参照先が変わっていることになります。

　このデータの情報をまったく損なうことなく，縦長に並べ直したのが図 22.3 の右側です。これは 1 つの行がそれぞれの観測に対応しています。このような形式にすると，何行目かということを指定するだけで，どの群の何番目の人なのかという情報が手に入ります。またデータの中に欠損があっても，整然データの形にする時にこれは観測に含まれないので，データの一部が欠けることがないという利点があります。

群	群内整理番号 1	2	3	4
A	6	6	5	5
B	7	4	7	6
C	4	3	NA	6

群	番号	値
A	1	6
A	2	6
A	3	5
A	4	5
B	1	7
B	2	4
B	3	7
B	4	6
C	1	4
C	2	3
C	4	6

図 22.3　一般的なデータと整然データ

　このようなデータ形式にして，これを与える形でデータ分析をするとデータサイズに依存しないコードが書けます。具体的には code:22.7 のようにします。

code : 22.7　整然データに対応したコード

```
data{
  int<lower=0> Lv;                              // 水準数
  int<lower=0> L;                               // データ数
  array[L] int<lower=0,upper=Lv> idx;           // データのID
  array[L] real X;                              // 変数の値
}
...(中略)...
model{
  // Likelihood
  for(l in 1:L){
    X[l] ~ normal(mu[idx[l]],sig);
  }

  // Prior
  gm ~ uniform(0,100);
  raw_effect ~ uniform(-100,100);
  sig ~ uniform(0,100000);
}
```

parameters ブロックと transformed parameters ブロックは code:22.5 と同じなので省略しました。まず data ブロックをみてください。群の数を Lv で入力するのはそのままですが，次にデータの総数 L を撮るようにしています。図 22.3 の例で言えば，C 群 3 列目のデータがありませんから，データ長は全部で $L = 11$ になります。次に L 行のデータがどの群に属するのかを指示するインデックス変数 idx を L 個用意しています。L 行目のデータが第何群に属しているのかの数字が入ります。たとえば 1 行目は A 群なので 1，5 行目 (L=5) は B 群なので 2，と言った数字が入るようになっています。さいごにデータの値そのものである X も，データ長 L と同じだけの 1 次元配列を用意してやります。

技術的に注目すべきは model ブロックの尤度のところで，for 文がデータ長 L の反復をしているだけです。つまり 1,2,3...,L 行目のデータを順に参照していくのです。そして l 行目のデータがどの群に属するかは，変数 idx[l] が指し示してくれますから，mu[idx[l]] としてやることで指定できていることになります。idx[l] は，l 行目のデータが属している群の数字が代入されていますから，たとえば 1 行目 (l=1) の場合は mu[idx[1]] = mu[1] としていることと同じ，5 行目の場合は mu[idx[5]]=mu[2] としていることと同じ，ということになります。ここでは入れ子になった参照が行われています。ちょっとテクニカルですが，この技術を使うと表現力も広がりますので，仕組みをしっかり理解しておいてください。

22.4 パラメータリカバリ

さあ，最後にパラメータリカバリをして，このコードが正しく推定できるか，あるいはどの程度のサンプルサイズがあればどの程度の精度で推定できるのかを検証しておきましょう。

私たちはデータがどのように造られるか，というそのメカニズムの方からアプローチしているわけです。これはリバースエンジニアリングと呼ばれる考え方ですね。そして今から仮想データを作ろうという時も，そのデータ生成モデルをそのまま利用すればいいのです。仮想データの生成は，データ生成メカニズムをリバースエンジニアリングで明らかにしていくアプローチのちょうど正反対，リバース・リバース・エンジニアリングです。

具体的には，たとえば次のようなコード (code:22.8) でデータを作ることができるでしょう。

code : 22.8 リバースエンジニアリング

```
N <- 100
Lv <- 5
gm <- 50
sig <- 3

raw_effect <- runif(Lv - 1, -10, 10)
effect <- c(raw_effect, 0 - sum(raw_effect))
mu <- gm + effect

X <- rnorm(N * Lv, mu, sig)
dat <- data.frame(
  Idx = rep(1:Lv, N * Lv),
  value = X
)

modelC <- cmdstanr::cmdstan_model("BetweenAnova2.stan")
dataSet <- list(Lv = Lv, L = NROW(dat), idx = dat$Idx, X = dat$value)
```

■コード解説

1 行目　各群のサンプルサイズ N を設定します。各群共通のサイズになります。

2 行目　水準数です。ここでは 5 群の平均値の比較をすることにしました。

3 行目　全体平均の設定です。事前分布に [0,100] の一様分布を置いてますから，真ん中ぐらいにしてあげました。

4 行目　誤差の散らばりを設定します。

6 行目　水準数 -1 の効果を設定します。手入力でもいいのですが面倒なので，-10 から 10 までの範囲で一様乱数によって生成しました[*8]。

7 行目　水準数 Lv の効果にするため，先ほど作った水準数 -1 の `raw_effect` に `0-sum(raw_effect)` の計算結果をつけくわえています。関数 `c` は結合させる combine という意味です。

8 行目　各群の平均値です。全体平均に効果を足しています。全体平均は 1 つの数字，効果は水準数の要素を持つベクトルですから，計算ができないように思えますが，このような場合 R は自動的にサイズ合わせのため値の再利用を行います。つまり，$50 + (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ を $(50, 50, 50, 50, 50) + (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$ と解釈してくれるのです。

10 行目　水準数 × 各群のサイズの乱数を発生させています。乱数は正規分布によるもので，平均や SD は上で指定した通りです。ここでも値の再利用が行われていて，ベクトル `mu` は $\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_1, \mu_2, ...$ とくり返して当てはめられていきます。

11-14 行目　数値を `data.frame` 型にくみ上げていってます。`rep` 関数は，指定した回数だけ繰り返すもので，ここでは `1:Lv` すなわち 1,2,3,4,5 という数字をデータサイズ分繰り返していることになります。

16 行目　ここでは `cmdstanr` パッケージを使ってコンパイルしています。

17 行目　推定に使うデータセットです。データ長は `data.frame` の長さを返す関数 `NROW` を使って与えています。インデックス変数は `dat` オブジェクトの `Idx` 変数，データは同じオブジェクトの `value` 変数を指定しています。

いかがでしょうか。作ったデータ，オブジェクト，ベクトルなどの意味がわからない場合は，その都度 R でオブジェクト名を入力し，何が格納されているかを確認しながら進めると良いでしょう。ここで作った仮想データを，先ほどの Stan オブジェクトに与えて乱数を生成し，パラメータリカバリがどの程度の精度でできるかを考えてみてください。サンプルサイズや誤差の大きさなどを変えながら確認してみると良いでしょう。

22.5　課題

今回は，基本課題と発展課題の 2 つを用意します。基本課題は必須ですが，応用課題は提出者に加点されるだけで必須ではありません (提出しなかったからと言って減点されることもありません)。

■基本；整然データに対応した多群比較のコードを書く　最後に紹介した，整然データに対応した多群比較のコードを書き，パラメータリカバリのコードとデータを使って正しく動くかどうか確認してください。作った Stan ファイルや R コードを提出してください。

■発展；自分のデータを使って分析する　基礎実験 1 や基礎実験 2 など，他の授業でとったいくつかの群の平均値を比較するようなデータを用意し，今回のモデルを適用して平均値の比較を行なってください。とくにそのようなデータセットを持っていない場合は，こちらから提供するサンプルデータを使ってください。

[*8] 一様乱数の関数は，R では `runif` です。

第 23 章

モデリングの目から見た検定 4；対応のある群の比較

23.1　対応のある群

23.1.1　多次元正規分布

ここまでは**群間計画 (Between Design)** のモデリングをしてきましたが，今回は**群内計画 (Within Design)** のモデリング的アプローチになります。

Within モデルは別名として，**反復測定 (Repeated Measure)** と呼ばれることがあり，とくに二群の場合は**対応のある t 検定 (Paired t-test)** と言ったりします。同じ人から 2 回データを取って，その変化をみる**事前・事後デザイン (Pre-post Design)** がその典型例です。2 回以上とる場合もあるので，その場合は反復測定とよばれるわけです。

この場合のデータ生成モデルは，何が違うのでしょうか。**群間計画**の t 検定は別名**独立した二群の** t 検定と呼ばれるように，2 つの群のデータがバラバラに出てきているのですが，今回は**対応のある二群**ですから，独立してないということになります。独立していない，つまり関係がある。データ分析上は，2 つのデータに相関がある場合を考える必要があります。同じ人の事前・事後のデータであれば，当然変化のもとになる「その人」という共通要因があるわけで，同じ人からデータを得ていますから当然相関していることを前提に考えなければなりません。

統計モデル上，独立した二群であれば別々の正規分布からデータが出てきていると考えますが，今回は 1 つの分布から 2 つのデータが出てきていることになります。ここで使うのは**二次元正規分布 (Two-Dimensional Norma Distribution)**，さらに群の数が多い場合に一般化した**多次元正規分布 (Multidimensional Normal Distribution)** あるいは**多変量正規分布 (Multivariate Normal Distribution)** と呼ばれる分布を使います。この正規分布の特徴は，それぞれの変数 (次元) に注目すれば正規分布なのですが，各次元に相関関係があることを組み込まなければなりません。

これまでの 1 次元の正規分布は，平均 (位置)μ と標準偏差 (幅)σ で特徴付けられました。多次元の正規分布も，各次元について平均と標準偏差があります。平均も複数の次元のセットになっているので，平均ベクトル $\boldsymbol{\mu}$ として表現することになります。標準偏差もベクトルで考える必要がありますし，さらにすべての組み合わせにおける相関関係がありますので，分散共分散行列 $\boldsymbol{\Sigma}$ を考えることになります。丁寧に数式で表現すると，1 次元正規分布に従う確率変数は次のように書きます。

$$X \sim Normal(\mu, \sigma)$$

これに対して多次元正規分布に従う確率変数ベクトル $\boldsymbol{X}$ は次のように書きます。

$$\boldsymbol{X} \sim MultiNormal(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

この分散共分散行列の中身を見てみますと，次のようになっています。

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1m} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{m1} & \sigma_{m2} & \cdots & \sigma_m^2 \end{pmatrix}$$

この行列の**対角 (diagonal)** 要素には分散 σ_j^2 があり，非対角要素には共分散 σ_{jk} が入っている**正方対称行列**になっていますね[*1]。またこうしてみると，変数 j の分散 σ_j^2 は j と j の共分散，すなわち σ_{jj} であることもわかりやすいですね。ところでこれを考えることでどこに相関の要素が入っているのでしょう？改めて，分散や共分散，相関係数の定義式から考えてみたいと思います。

$$\text{分散}: \sigma_j^2 = \frac{1}{n}\sum(x_{ij} - \bar{x_j})^2 \tag{23.1}$$

$$\text{共分散}: \sigma_{jk} = \frac{1}{n}\sum(x_{ij} - \bar{x_j})(x_{ik} - \bar{x_k}) \tag{23.2}$$

$$\text{相関}: \rho_{jk} = \frac{\sigma_{jk}}{\sigma_j \sigma_k} \tag{23.3}$$

この関係から逆に，共分散 σ_{jk} は次のように表すことができます。

$$\sigma_{jk} = \sigma_j \sigma_k \rho_{kj}$$

ということで，共分散の式は分布の幅を表す標準偏差と，相関係数からできあがっていることがわかりましたね。

23.1.2 対応のあるデータの生成モデル

ではこれを使ってモデルの設計図を書いてみましょう。設計図は図 23.1 のようになるでしょう。

あとはこれを使ってコードを書くだけです。この時のポイントは，データの渡し方にあります。

ベクトル型とマトリックス型

2 次元正規分布に従うデータは，必ず 2 つセットになっています。1 回の観測で 2 つの数字を持つものです。これを Stan で表現するときには，変数を `vector` 型で表現しておく必要があります。今回の場合は `vector[2] mu;` などとします。これで `mu` という変数が 2 つセットで準備されることになります。これまで同じ変数名で複数のデータを持つ場合，たとえば `real mu[2]` のようにしていましたが，なぜ今回もこの形式，つまり**配列**にしないのか，と思われるかもしれません。配列でも 2 つの数字を扱ったりすることは可能なのですが，ベクトル型にしておく利点が別にあります。ベクトルにはベクトルを使った演算がありますが，Stan にはベクトルや行列専用の関数がありますので，変数をベクトルで宣言しておいてやるとこれらの関数を使って高速化できるのです。

また Stan には，`matrix` 型という行列の型もあります。`matrix[N,M] X;` のように宣言してやると，サイズ $N \times M$ の行列変数 X が 1 つ用意されます。`matrix[N,M] X[L];` のように宣言すると，サイズ $N \times M$

[*1] これら行列の要素や名称については，第 6 講でやったことを思い出してください。

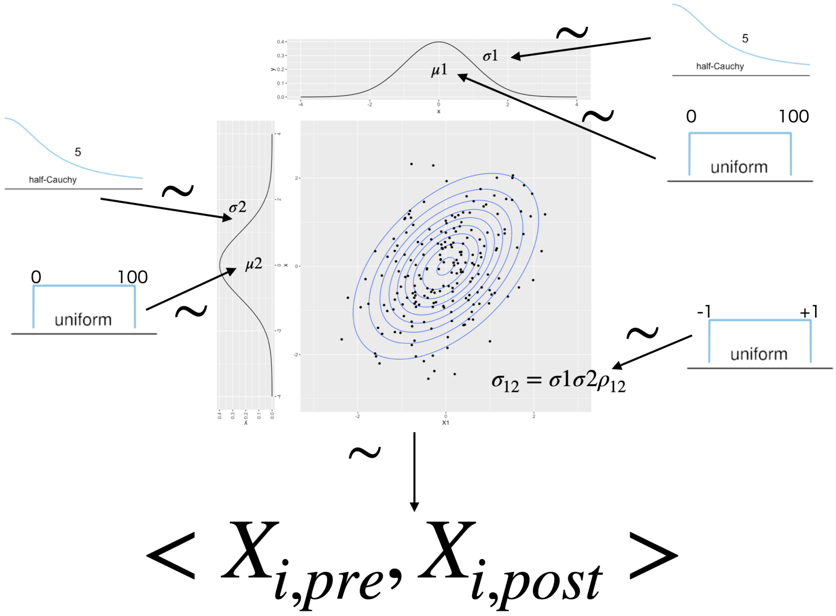


図 23.1　対応ある二群のデータ生成モデル設計図

の行列変数 X を L 個用意することになります。他にも特殊な型として，`cov_matrix` 型とか `corr_matrix` 型などがあります。これらは正方行列ですから，サイズの指定は 1 つでよく，たとえば `cov_matrix[2]` とすれば 2×2 の正方行列を用意したことになります。Stan における変数や配列の型について，松浦 (2016) を参考にまとめたものを表 23.1 に，イメージ図を図 23.2,23.3,23.4 に示します。

表 23.1　Stan の配列と型

型の例	宣言例	解説
整数	`int X`	整数ひとつ
実数	`real X`	実数ひとつ
範囲つき実数	`real<lower=0> X`	0 より大きい数字の入る X
整数の配列	`int X[N]`	N 個の整数
実数の配列	`real X[N,M]`	$N \times M$ 個の実数からなる 2 次元配列
実数の配列	`real X[N,M,L]`	$N \times M \times L$ 個の実数からなる 3 次元配列
ベクトル	`vector[K] X`	長さ K のベクトルひとつ
ベクトル	`vector[K] X[N]`	長さ K のベクトルが N 個
ベクトル	`vector[K] X[N,M]`	長さ K のベクトルが $N \times M$ 個
行列	`matrix[J,K] X`	サイズ $J \times K$ の行列がひとつ
行列	`matrix[J,K] X[N]`	サイズ $J \times K$ の行列が N 個
特別な行列	`cov_matrix[J] X`	サイズ $J \times J$ の正方対称行列がひとつ
特別な行列	`corr_matrix[J] X`	対角が 1 のサイズ $J \times J$ の正方対称行列がひとつ

これを踏まえて，今回の対応のある二群のコード例を見てみましょう (Code::23.1)。

code : 23.1　対応ある二群のデータ生成モデル

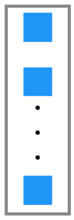
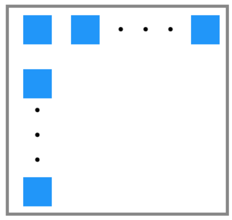
型	宣言例	サイズ
整数	`int X`	
実数	`real X`	
範囲つき実数	`real<lower=0> X`	
整数の配列	`int X[N]`	N 
実数の配列	`real X[N,M]`	M N 

図 23.2　変数の型とサイズ (1)

```
data{
  int N;
  array[N] vector[2] X;
}

parameters{
  vector[2] mu;
  real<lower=0> sd1;
  real<lower=0> sd2;
  real<lower=-1,upper=1> rho;
}

transformed parameters{
  cov_matrix[2] SIG;
  SIG[1,1] = sd1 * sd1;
  SIG[1,2] = sd1 * sd2 * rho;
  SIG[2,1] = sd2 * sd1 * rho;
  SIG[2,2] = sd2 * sd2;
}

model{
  X ~ multi_normal(mu,SIG);
  //prior
  mu[1] ~ uniform(0,100);
  mu[2] ~ uniform(0,100);
```

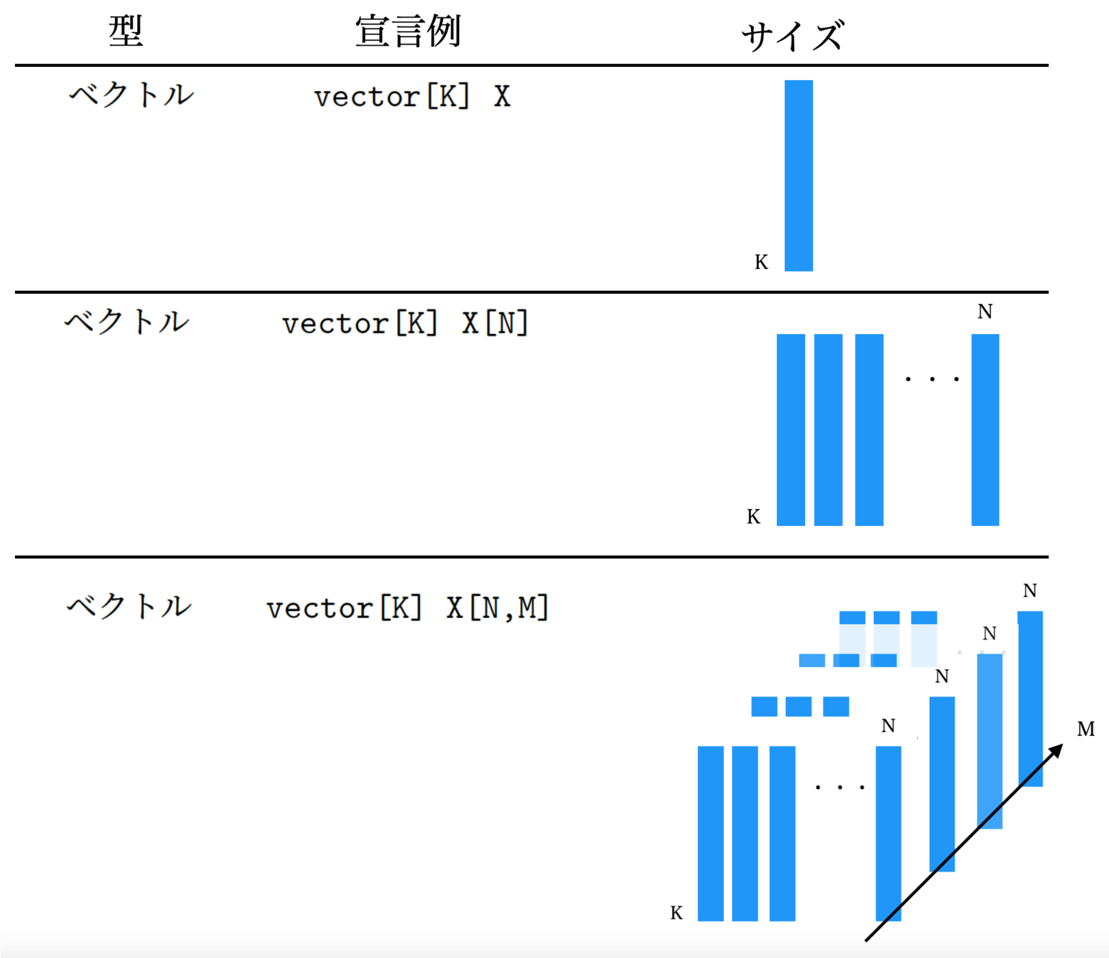


図 23.3 変数の型とサイズ (2)

```
    rho ~ uniform(-1,1);
    sd1 ~ cauchy(0,5);
    sd2 ~ cauchy(0,5);
}
```

■コード解説

data **ブロック** サンプルサイズ N のデータですが，ペアになっている数字なので `vector[2]` で宣言した変数 X を N 個用意しています。

parameters **ブロック** パラメータとして，平均をベクトル，標準偏差や相関係数はスカラで宣言しています。標準偏差や相関係数の範囲指定にも注意してください。

transformed parameters **ブロック** 各標準偏差，相関係数を使って分散共分散行列 SIG を構成しています。わかりやすくするために，各要素について書き下して書いています。

model **ブロック** 尤度はデータベクトルが多次元正規分布関数 `multi_normal` から出てきていることを示しています。多次元正規分布関数の引数は，ベクトルと行列です。

この書き方ですと，三群に拡張した時に大変になるのは目に見えていますね。こんな時のために，ベクトルと行列用関数をつかって書いておく方法があります。Code::23.2 には，多次元にまで拡張したコードを示し

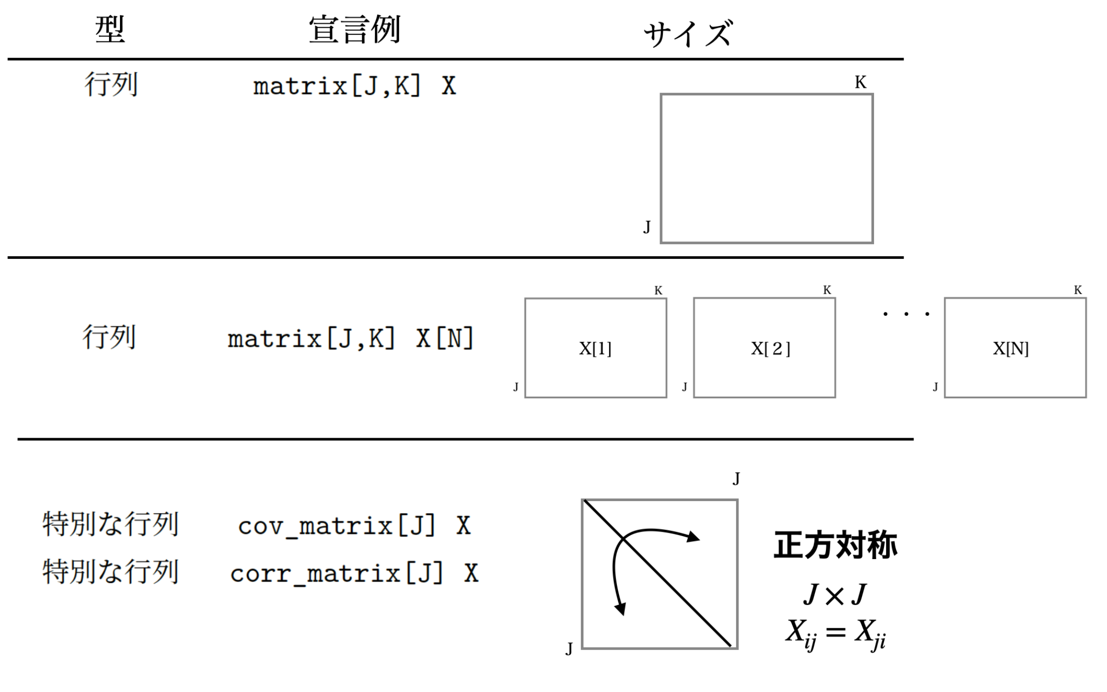


図 23.4 変数の型とサイズ (3)

ています。

code : 23.2 コードの一般化

```
data{
  int N;
  int K;
  array[N] vector[K] X;
}

parameters{
  vector[K] mu;
  vector<lower=0>[K] sds;
  corr_matrix[K] rho;
}

transformed parameters{
  cov_matrix[K] SIG;
  SIG = quad_form_diag(rho,sds);
}

model{
  X ~ multi_normal(mu,SIG);
  //prior
  mu[1] ~ uniform(0,100);
  mu[2] ~ uniform(0,100);
  rho ~ lkj_corr(1);
  sds ~ cauchy(0,5);
}
```

■コード解説

data ブロック　データのサイズ K も外侮から指定できる変数とし，それを使ってサイズ K のベクトル X を N 個用意しています。

parameters ブロック　パラメータとして，平均と標準偏差をベクトルで，相関係数も行列として宣言しました。標準偏差ベクトルの範囲指定の方法に注意してください。

transformed parameters ブロック　Stan の関数 `quad_form_diag` を使って分散共分散行列を作成しました。この関数は，正方行列 $\boldsymbol{X}$ とベクトル $\boldsymbol{v}$ から，$diag(\boldsymbol{v})\boldsymbol{X}diag(\boldsymbol{v})$ という計算をするものです。ここで `diag` は k 個の要素を対角項に持つ $k \times k$ の対角行列を作ることを意味します。今回の例で具体的に説明しましょう。ここでの正方行列は相関行列で，ベクトルは標準偏差を要素に持つ `sds` ですから，

$$\begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}\begin{pmatrix} 1 & \rho_{12} \\ \rho_{12} & 1 \end{pmatrix}\begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} = \begin{pmatrix} \sigma_1 & \sigma_1\rho_{12} \\ \sigma_2\rho_{12} & \sigma_2 \end{pmatrix}\begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_{12} \\ \sigma_1\sigma_2\rho_{12} & \sigma_2^2 \end{pmatrix}$$

となり，分散共分散行列が構成されていることがわかりますね。

model ブロック　注目すべきポイントは相関行列の事前分布です。相関行列の事前分布には，Stan では `lkj_corr` という関数を使うのが一般的で，この引数が 1 になっているのは無情報分布であることを意味しています。

このコードで推定される平均値，`mu` ベクトルの 2 つの要素が，事前・事後の平均値に該当します。ですからこの差分を計算すれば「平均的にどれぐらい変化があったか」という**平均因果効果**を見積もったことになります。この差分については，**生成量**のブロックで算出しても構いません。もちろんその差分から標準化した効果の大きさを算出することもできますし，パラメータについての仮説，事後予測などデータについての仮説など，さまざまに検証できることはこれまでの通りです。

23.2 ID をもったデータ構造

さてでは今回の乱数生成機の精度を検証するために，パラメータリカバリをしてみましょう。

仮想データの作り方は，データ生成モデルの裏返しです。2 群の平均値差を検証するためのデータでしたら，次のようにして作ることができます。

code : 23.3　対応ある二群の仮想データ

```
mu <- c(50,50)
sd1 <- 10
sd2 <- 5
rho <- 0.7
SIG <- matrix(ncol=2,nrow=2)
SIG[1,1] <- sd1 * sd1
SIG[2,2] <- sd2 * sd2
SIG[1,2] <- sd1 * sd2 * rho
SIG[2,1] <- sd1 * sd2 * rho
N <- 100
library(MASS)
X <- mvrnorm(N,mu,SIG)
# Stanに与えるデータセット
dataSet <- list(N=N, X=X)
```

ここでは二群の平均をいずれも 50, 標準偏差を 10 と 5 にし, 相関係数を 0.7 としたサンプルを 100 個作ることにしました。発生させる乱数は `MASS` パッケージの `mvrnorm` 関数を使います。この関数は, 発生させるサンプルサイズと, 平均ベクトル, 分散共分散行列を引数に取ります。これを使って推定させた結果の例が出力 8 になります。

MCMC の結果 8

```
 # A tibble: 9 × 7
 name          EAP       MED       MAP        SD       L95       U95
 <chr>    <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 mu[1]      47.590    47.594    47.608     0.913    45.808    49.356
2 mu[2]      49.506    49.508    49.570     0.500    48.523    50.493
3 rho         0.693     0.697     0.703     0.052     0.582     0.784
4 sd1         9.087     9.059     9.071     0.640     7.919    10.434
5 sd2         4.980     4.960     4.930     0.352     4.345     5.724
6 SIG[1,1]   82.975    82.063    81.989    11.761    62.715   108.872
7 SIG[1,2]   31.597    31.189    30.790     5.515    22.086    43.574
8 SIG[2,1]   31.597    31.189    30.790     5.515    22.086    43.574
9 SIG[2,2]   24.923    24.599    24.207     3.550    18.883    32.761
```

中央値による点推定値 (MED 推定値) を見ると, ほぼ真の値の通りになっていますね。

ところで, 今回与えたデータ X は行列の形をしています。一行が 2 つの数字を持つペアになっているのですが, これは**整然データ**の形にはなっていませんね。整然データであれば, 次のようなデータセットになっているはずです (R の出力 23.1)。

R の出力 23.1: 対応ある二群の仮想データ・整然版

```
# A tibble: 200 × 3
      ID name  value
   <int> <chr> <dbl>
 1     1 pre    45.1
 2     1 post   45.8
 3     2 pre    45.1
 4     2 post   51.2
 5     3 pre    52.4
 6     3 post   53.7
 7     4 pre    46.8
 8     4 post   48.6
 9     5 pre    45.5
...
```

このようなデータ形式に対応できるように, Stan のコードを書き直してみましょう。ポイントは, 個体識別インデックスと, 事前・事後を表す変数のインデックスの両方を準備する, というところにあります。

まず R から与えるデータの方から考えましょう。整然データの形に並び替えるのに加えて, 事前・事後といった水準を表す変数も数字で与えなければなりません。先の例 Code::23.4 では pre,post となっていたところを, pre=1,post=2 と置き換えた別の変数を用意しました。

code : 23.4　対応ある二群の仮想データ・整然版

```
1 ...
2 X <- mvrnorm(N, mu, SIG)
3 tidy_data <- X %>%
4   as.data.frame() %>%  as_tibble() %>%
5   rename(pre = V1, post = V2) %>%
6   rowid_to_column("ID") %>%
7   pivot_longer(-ID) %>%
8   mutate(cond = if_else(name == "pre", 1, 2))
```

■コード解説

2 行目 真値の設定をして，行列 X を作っています。

3 行目 X を加工して，`tidy_data` という変数に作り替えます。加工のプロセスは下記の通りです。

4 行目 `data.frame` 型を経て `tibble` 型に変形します[*2]。

5 行目 この段階で，変数名が `V1,V2` となっていますが，わかりやすくするために `pre,post` に書き換えました。

6 行目 行番号を ID という変数名を持つものに変えています。

7 行目 ID をキーに，横長だったデータを縦長に (tidy に) 変換する関数です。

8 行目 縦長になった変数 (Code::23.4 はこの段階の出力です) から，`name` 変数の値が `pre` であるものを 1 に，そうでないものを 2 にした新しい変数 `cond` を作りました。

このコードでどういうデータセットができたのか，R のコンソールで `tidy_data` として確認すると良いでしょう。1 つ 1 つのステップを確認したい場合は，途中で止めて一歩ずつ進むのも手です。

さて Stan にデータセットを与えるときは，このデータセットから 1. データ全体の長さ，2. 被験者の数，3. 被験者 ID，4. そのデータがどの水準なのかを示す識別子，5. 実際の値を取り出して渡すことになります。続いて Stan コードの例を見てみましょう。

code：23.5 整然データ対応番

```
data{
  int L;
  int N;
  array[L] int<lower=0,upper=N> IDindex;
  array[L] int<lower=1,upper=2> Condition;
  array[L] real val;
}

transformed data{
  array[N] vector[2] pairX;
  for(l in 1:L){
    pairX[IDindex[l],Condition[l]] = val[l];
  }
}
...(中略)...
model{
  pairX ~ multi_normal(mu,SIG);
```

[*2] `tibble` 型にする必要は別にありません。著者の趣味です。`tibble` 型は `data.frame` 型の拡張版で表示した時に変数の型などがわかりやすいという利点があります。`matrix` 型からいきなり `tibble` 型に変更すると警告が出るので，いったん `data.frame` 型を経由しました。

```
    //prior
    mu[1] ~ uniform(0,100);
    mu[2] ~ uniform(0,100);
    rho ~ uniform(-1,1);
    sd1 ~ cauchy(0,5);
    sd2 ~ cauchy(0,5);
}
```

■コード解説

data ブロック　1. データ全体の長さ L，2. 被験者の数 N，3. 被験者 IDIDindex，4. そのデータがどの水準なのかを示す識別子 Condition，5. 実際の値 val を与えています。最後の 3 つはデータ長と同じサイズです。

transformed data ブロック　データを変形する transformed data ブロックの登場です。ここではペアとしての情報がある vector 型にした変数を作っています。その変数 pairX は，N 人分の情報があり，それぞれ要素番号 1,2 がベクトルの要素に対応しています。ここでも入れ子になった識別子を使っていて，pairX[IDindex[l],Condition[l]] は l 行目の個人 IDindex[l]，l 行目の条件 Condition[l] から，たとえば IDindex[l] = 15，Condition[l]=2 であれば 15 番目の人の事後の値 pairX[15,2] を値として代入する，ということをしています。

model ブロック　作った pairedX というベクトルに対して多次元正規分布からのデータを与えています。

少しややこしくなったように思えますが，データの構造が複雑化すると整然データの方がスッキリすることの方が一般的です。コードをよく読んで，どのような動きをするのかイメージをしっかり掴むようにしましょう。

23.3　個人差と変化量のモデルへ

さて，コードは随分と複雑になってきましたが，やっていることは「事前・事後で変化はあるか」ということだけです。それなのに多次元正規分布が出てくるなんて，面倒ですねえ！もう少し簡単な表現はできないものでしょうか。たとえば事前 + 効果=事後のように。

もちろん可能です。むしろその方が自然なモデリングと言えるかもしれないですね。ただその時，事前・事後が同一人物であることを忘れてはいけません。小杉の事前の状態 + 効果=山田の事後の状態，では意味がわからないからです。記号を使って書くと，$X_i^{pre} + effect = X_i^{post}$ のように，添字 i で同じ個人であることを明記しておく必要があります。

これを踏まえて，少し話を拡張した 3 水準モデルを考えてみましょう。たとえば次のようなカバーストーリーはいかがでしょう。

> 臨床心理学者がある介入法をつかってメンタルケアに取り組んでいます。4 人のクライアントにこの手法を適用し，抑うつ度が改善されていくかどうかチェックし，この介入法に効果があるのかどうかを検証したいと思っています。

チェックの結果が表 23.2 のようになっていて，ここの数字が抑うつ度スコア (低ければ低いほど健康) だと思ってください。記号で書く時のノーテーション (表記法) は，値 X_{ij} が i さんの第 j 期のスコア，ということにしたいと思います。

これを使って毎回の変化の大きさを見積もりたいと思います。前回やった，多水準モデルのことを参考に見ていきましょう。ポイントは，前回は全体平均からの差分で効果を考えていたところが，今回は個人差がある

表 23.2　介入時期とスコアの推移

ID	1 期	2 期	3 期
1	10	5	9
2	9	4	5
3	4	2	3
4	7	3	5

ので個人ごとの平均 μ_i を基準におくところです。

最初の人のスコア，$X_{11} = 10, X_{12} = 5, X_{13} = 9$ はそれぞれ，その人の平均 μ_i から考えて，$\mu_1 + \delta_1, \mu_1 + \delta_2, \mu_1 + \delta_3$ となるはずだと考えましょう。理論通りにならない誤差を含めて考えれば，先ほどの表 23.2 は表 23.3 のようになっているはずなのです。

表 23.3　介入時期とスコアのモデル式

ID	1 期	2 期	3 期
1	$X_{11} \sim N(\mu_1 + \delta_1, \sigma)$	$X_{12} \sim N(\mu_1 + \delta_2, \sigma)$	$X_{13} \sim N(\mu_1 + \delta_3, \sigma)$
2	$X_{21} \sim N(\mu_2 + \delta_1, \sigma)$	$X_{22} \sim N(\mu_2 + \delta_2, \sigma)$	$X_{23} \sim N(\mu_2 + \delta_3, \sigma)$
3	$X_{31} \sim N(\mu_3 + \delta_1, \sigma)$	$X_{32} \sim N(\mu_3 + \delta_2, \sigma)$	$X_{33} \sim N(\mu_3 + \delta_3, \sigma)$
4	$X_{41} \sim N(\mu_4 + \delta_1, \sigma)$	$X_{42} \sim N(\mu_4 + \delta_2, \sigma)$	$X_{43} \sim N(\mu_4 + \delta_3, \sigma)$

つまり，$X_{ij} \sim N(\mu_i + \delta_j, \sigma)$ ですね。個人のベースライン μ_i に，各時期の効果 δ_j が加わったものを中心に，正規分布に従う誤差を纏ってデータになる，という考え方です。ただしここでも注意が必要で，パラメータがいろいろありますが，効果は相対的なもの，すなわち $\delta_1 + \delta_2 + \delta_3 = 0$ という総和ゼロの制約を考える必要があります。実質的には自由度 2，すなわち $\delta_3 = 0 - (\delta_1 + \delta_2)$ となることを忘れないでください。

そしてもう 1 つ大事なことが。μ_i は人によって違う個人差成分ですが，この散らばりも正規分布に従うものとして考えることができますね。すなわち平均と散らばりをもった集合体が従う分布としての正規分布です。この平均を ψ，標準偏差を τ として[*3]，

$$\mu_i \sim N(\psi, \tau)$$

と考えることにしましょう。これを設計図に書き込んだのが，図 23.5 です。

このように，相関があることを個人 i に紐づけられた変数として表現し，個人差からの差分として対応のあるモデルを考えることができます。ここで，個人差に関係なく影響を与えている効果のことを**固定効果 (Fixed Effect)** と呼び，個人差のように交換可能な要素のことを**変量効果 (Random Effect)** と呼びます。また図 23.5 に示されているように，データ生成分布の中に個人差の分布が入れ子になって含まれていますね。このようなデータは階層モデルといい，とくに今回のように線形な階層モデルは**階層線形モデル (Hierarchical Linear Model)** とよばれます。この展開については，次回以降にお話しすることになるでしょう。

ではこのモデルを実装していきましょう。設計図に沿えば Stan コードは書けるはずです。ここには Stan に与える R のコード例を用意しましたので，自分なりのコードを書いて検証してみてください[*4]。

[*3] ψ はギリシア文字でプサイと読みます。τ も同じくギリシア文字でタウと読みます。

[*4] これに限らず，コードに正しい書き方はありません。自分なりの書き方で結構です。判断すべきは正しく機能するかどうかであって，書き方はあくまでも一例にすぎません。

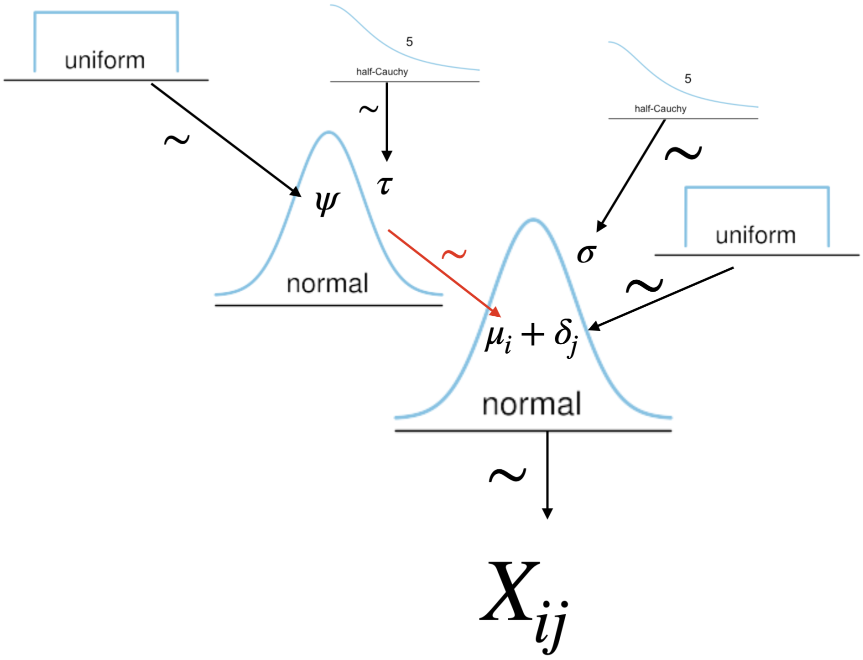


図 23.5　階層的データ生成モデル

code : 23.6　整然データ対応番

```
dat_raw <- data.frame(ID = 1:4,
                      period1 = c(10,9,4,7),
                      period2 = c(5,4,2,3),
                      period3 = c(9,5,3,5))
# 整然データにする
tidy_dat <- dat_raw %>%
  pivot_longer(-ID) %>%
  mutate(name = as.factor(name) %>% fct_relevel("period1","period2")) %>%
  mutate(cond = as.numeric(name))
```

23.4　課題

Code::23.6 を参考にしながら, 表 23.2 の事例を検証するための Stan コードを書いてください。作った Stan ファイルや R コードを提出してください。

第 24 章

モデリングの目から見た検定 5；カテゴリカル分布をつかって

ここまでさまざまなモデルを使って，パラメータ間あるいはデータの間の差を検討するということを考えてきました。ここで差分を計算できているということは，数値が**連続的**であったということもできます。いいかるならば，**間隔尺度水準**以上の尺度水準で得られたデータに対する検証だったわけです。

ところがデータの種類によっては**順序尺度水準**や**名義尺度水準**でしか得られないこともあります。試験に合格したのか，失敗したのか。男性か，女性か。47 都道府県のどこ出身なのか。あるいはたとえば，正解率や成功率，相対的な比率などでデータが現されたり考えたりすることもあります。比率は連続的な数字ではありますが，$\dfrac{X}{Y}$ のように数え上げによる度数の比較ですから，元は**離散的**な，質や種類を区別するための数字による (度数に該当するかどうか) 情報だと言えるでしょう。たとえば比率のデータに対して t 検定のような分析をするのは適切ではなく[*1]，カテゴリカルな分布を考えて検証する必要があります。

また，複数のカテゴリカル変数の組み合わせによって**クロス集計表** (**cross-tabulation table**) あるいは**分割表** (**contingency table**) を利用することもあります。男性の X 割が A 党に，女性の Y 割が B 党に投票した，といった表は，性別 $\times$ 支持政党のクロス集計表ですが，これを見ることで性別による支持政党の偏りがあるのかないのか，といったことがわかります。クロス集計表は社会科学的なデータにも多く見られ，カテゴリカルな性質の組み合わせについての情報を提供してくれます。この組み合わせに偏りがあるのかないのか，と言ったことを考える際にも，確率モデルとしてはカテゴリカルな分布を必要とします。

24.1　離散的な分布

それでは代表的な離散変数についての**確率分布**を見ていきましょう。離散分布の場合，確率関数は密度 density ではなく，質量 mass，つまり**確率質量関数** (**Probability Mass Function**) と呼ばれることに注意してください。カテゴリーの中に含まれるかどうか，どれぐらいの量がそこに含まれているかということを直接表しているからです。

■ベルヌーイ分布　離散的な分布の基本は**ベルヌーイ分布** (**Bernoulli Distribution**) です。この分布から出てくる数字は 0 か 1 であり，確率 θ で 1 が，$1-\theta$ で 0 が出る分布です。たった 2 つの状態しか区分しませんが，正答と誤答，賛成と反対，男性と女性，生と死などさまざまなもののメタファーとして利用できるため，応用範囲はむしろ広いと言えるでしょう。パラメータは θ ひとつですので，わかりやすいですね。Stan

[*1] t 検定は正規分布するデータを前提にしている検定方法･確率分布であり，比率のデータはとうてい正規分布とは考えられないからです。正規分布の定義域は $-\infty$ から ∞ なのに対し，比率は 0 から 1 でしかないことからも明らかです。

での関数は bernoulli と書き[2]，y ~ bernoulli(theta); のように使います。

今後出てくることになりますが，ロジスティック関数と組み合わせて使うことが多い分布です。ロジスティック関数は，連続的な数字を 0 から 1 の範囲に変換する関数で，変換したものをベルヌーイ分布のパラメータにする合わせ技が非常に便利だからです。たとえば学力などは正規分布すると考えられ，$-\infty$ から ∞ の値を取りうることになりますが，その能力を反映してテストに正答するか誤答するか，ということを考えると学力 θ をロジスティック関数にいれた $(logistic(\theta))$ ものをベルヌーイ分布にいれる (X ~ bernoulli(logistic(theta))) からです。この合わせ技はよく使われるので，Stan では bernoulli_logit という関数があるほどです。

■二項分布　パラメータ θ をもつベルヌーイ分布に従うコイントスを N 回やったときに，何回表が出るか，というときに使うのが**二項分布 (Binomial Distribution)** です。10 問中何問正解するかとか，N 個提示した刺激のうち覚えていたのはいくつかとか，N 回シュートして入ったのは何回かといった，割合や比率に関係する分布です。パラメータは N と θ の 2 つあり，Stan での関数は binomial です。y ~ binomial(N, theta); のように書きます。

■カテゴリカル分布　ベルヌーイ分布や二項分布はコイントスの「表」のことしか考えていませんでした(裏は「表ではない」という考え方)。これに対してカテゴリカル分布はサイコロの出目のように，1,2,3,4,5,6 のどれが出るか，という多段階の乱数発生を考える分布です。こん関数のパラメータはサイズ K のベクトルで，たとえばサイコロの場合は $\boldsymbol{\theta} = (1/6, 1/6, 1/6, 1/6, 1/6, 1/6)$ のようになります。Stan での関数は categorical と書き，使う時は y ~ categorical(theta); のようにします。このとき theta はベクトルで宣言されていなければなりません。

■多項分布　カテゴリカル分布と似ているようですが，この分布は N 回サイコロを降った時の各出目の回数であり，1 が k_1 回，2 が k_2 回，$\cdots$，6 が k_6 回出た，というようにカウントしたものを出力します。ABO 型の血液分類で，サンプルの中に A 型が何人，B 型が何人‥というときに，この母比率を推定する時は多項分布のパラメータを推定する，ということになります。二項分布の多変量版だと考えてもいいかもしれません (松浦, 2016)。この分布に与えるパラメータはベクトル $\boldsymbol{\theta}$ であり，Stan では y ~ multinomial(theta); のように使います。総数 N はデータベクトル $\boldsymbol{y}$ の要素の総和 $(N = \sum_{i=1}^{K} y_i)$ ですから指定しなくても構いません。ここで注意すべきは，theta はサイズ K のベクトルであり，すべての要素を足し合わせると 1 になる必要があります。Stan では合計 1 のベクトル専用の型があり，それは simplex と呼ばれます。

24.2　χ^2 検定

カテゴリカルなデータ，度数分布などをどうやって研究の時に使うのか，具体的な例とともに見ていきましょう。ベイズ推定の話に入る前に，帰無仮説検定の話から先に考えてみたいと思います。

表 24.1 には，3 つの携帯電話キャリアのユーザ数をとある大学で調査した例です。合計 123 名から回答を得て，それぞれ 51 名，45 名，27 名ということが判明しました。これを見て「A 社は多いね，学生人気だね」などと解釈してもいいのですが，これでは推測統計学の域を出ていませんね。違うサンプルを対象にすれば違う度数になる可能性があるからです。そこで推測統計学的な発想をします。

ここで検証したいのは「どこかのカテゴリが多く，どこかのカテゴリは少ない」，つまり「散らばりに偏りがある」ということです。逆に考えると帰無仮説が出てきます。つまり「散らばりに偏りはない」が帰無仮説であり，

[2] 対数尤度で書く場合は bernoulli_lpmf となります。lpmf は log probability mass function の略です。

表 24.1　携帯キャリア調査

キャリア	A 社	D 社	S 社	合計
観測度数	51	45	27	123
期待度数	41	41	41	

対立仮説は「偏りがある」ということになります。散らばりに偏りがない，という帰無仮説の世界の元では，期待度数は総数を均等に割った値になるはずです。表 24.1 には**期待度数 (Expected frequency)** という行を設けました。ちなみに得られたデータそのものは**観測度数 (Observerd frequency)** と言います。

今回のデータが帰無仮説の元ではどれぐらいあり得ない数字だったのでしょうか。これを考えるために，χ^2 値を計算します。χ はギリシア文字のカイであり，x(エックス) とは違うので注意してください。χ^2 値はカイ二乗値と読みます。この値は次のように計算します。

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

ここで $_i$ は観測度数，E_i は期待度数です。今回の場合は次のような数字になります。

$$\chi^2 = \frac{(51-41)^2}{41} + \frac{(45-41)^2}{41} + \frac{(27-41)^2}{41} = 2.439 + 0.390 + 4.780 = 7.609$$

計算式から明らかなように，期待度数からのずれの大きさを表現していますから，期待通り＝帰無仮説通りであればこの値は 0 になります。今回は $\chi^2 = 7.609$ ですが，この統計量がどれぐらい出てきやすいかということは，自由度 2 の χ^2 分布をつかって検証できます。ここでの自由度は選択カテゴリ数 -1 です[*3]。自由度 2 の χ^2 分布から 7.609 という実現値が出てくる確率は $p = 0.022$ で 5% より小さいですから[*4]，統計的に有意な偏りがある，と判断できます。

χ^2 は偏りを検証するための数字です。今回は均等である，という帰無仮説から算出しましたが，母集団比率が事前にわかっているのであればそれを用いて，手元のサンプルがどれぐらい偏っているかを検証するということも可能です。たとえば日本人の血液型はおよそ A 型が 40%，O 型 30%，B 型 20%，AB 型 10 %といわれていますから，期待度数をこの比率で割り振るとサンプルが偏っていたかどうかの検証になります[*5]。他にも (比率によらず) 理論的な度数がわかっていることがあれば，どれぐらい合致しているかを検証できます[*6]。

また同様のロジックで，クロス集計表の検定を行うこともできます。クロス集計表の場合は**独立性の検定**と呼ばれることもあります。偏りがあれば関係がある，独立ではないということですね。

例として，表 24.2 には，先ほどの携帯キャリア調査の結果に男女の区別もつけてみました。これを見て「男性は D 社が好きで，女性は A 社が好き」といった偏りがあると判断できるかどうかを検証できます。

各セルの期待度数は行の周辺度数 × 列の周辺度数 ÷ 総度数で計算でき，各観測度数から期待度数を引いて二乗したものを，さらに期待度数で割ったものを足し合わせることで χ^2 にするのは同じです。これが (行の数 -1)×(列の数 -1) の自由度をもつ χ^2 分布に従いますので，今回は $\chi^2 = 6.4326, df = 2, p =$

[*3] 合計数が 123 と決まっていますから，最初の二つのカテゴリに入る数字が決まれば残りの一つは自動的に決まります。自由に値が変えられるのは二つまでなのです。

[*4] この値は R で `1-pchisq(7.609,df=2` とすることで得られます。もっと直接的にするなら，`chisq.test(c(51,45,27))` とすると χ^2 検定が実行されます。

[*5] 母比率の検定と言います。

[*6] **適合度**の検定といいます。モデルフィットの指標としても使われ，たとえば第 13 講で説明した**構造方程式モデリング**の指標などでも用いられています。

表 24.2 携帯キャリア調査と性別の関係

キャリア	A 社	D 社	S 社	合計
男性	21	30	13	49
女性	30	15	14	74
合計	51	45	27	123

0.0401 であることから有意 (に偏っている), と判断できます[*7]。これは**分散分析**の時と同様に, どこかに偏りがあるという全体的な傾向を示しただけであることに注意が必要です。

このようにして度数の検定を行うことができます。これに対して, データ生成メカニズムを考える場合は, ある比率に沿って度数が出てきていると考えますから, もっと直接的に母比率の検定を考えることができますし, MCMC によるアプローチを使うと「どこが, どの程度大きいと言えるか」, といったことについても検証できます。次にその方法を見ていきましょう。

24.3 カテゴリカル分布のモデリング

ではまず, 表 24.1 にあげた携帯キャリア調査の結果を, データ生成モデルから考えてみましょう。設計図にするまででもない感じで, 51:45:27 というデータの比率から考えられる, 母比率ベクトル $\boldsymbol{\pi} = \{\pi_1, \pi_2, \pi_3\}$ を考え, そこからデータベクトル $\boldsymbol{y}$ が多項分布に従って出てくる, と考えるわけです。

Stan のコードは Code::24.1 のようになります。ポイントは特殊なベクトル `simplex` で $\boldsymbol{\pi}$ を宣言しているところでしょうか。またデータベクトルは `vector[K] X;` としたいところですが, モデルからいってここは `int` 型である必要があり, 型指定をしたベクトルというのがないので, 配列にしました。

code : 24.1 多項分布からのデータ生成モデル

```
data{
    int K;
    array[K] int X;
}

parameters{
    simplex[K] pi;
}

model{
    X ~ multinomial(pi);
}
```

これを次のような R コードで実行してみましょう (Code:24.2)

code : 24.2 多項分布のパラメータ推定

```
# rstanの場合
model <- rstan::stan_model("categorical1.stan")
fit <- sampling(model,
  data = list(K = 3, X = c(51, 45, 27))
)
```

[*7] この計算は `chisq.test(matrix(c(21,30,13,30,15,14),ncol=3,byrow=T))` で算出しました。

```
# cmdstanrの場合
model <- cmdstanr::cmdstan_model("categorical1.stan")
fit <- model$sample(
  data = list(K = 3, X = c(51, 45, 27)),
  chains = 4,
  parallel_chains = 4,
  refresh = 500
)
```

構造も何もないモデルですから，推定はすぐに終わると思います。$51 : 45 : 27 = 0.4146 : 0.3658 : 0.2195$ ですから，ほぼ標本分布と同じような形で事後分布が推定されました (図 24.1)。これで終わり，といった

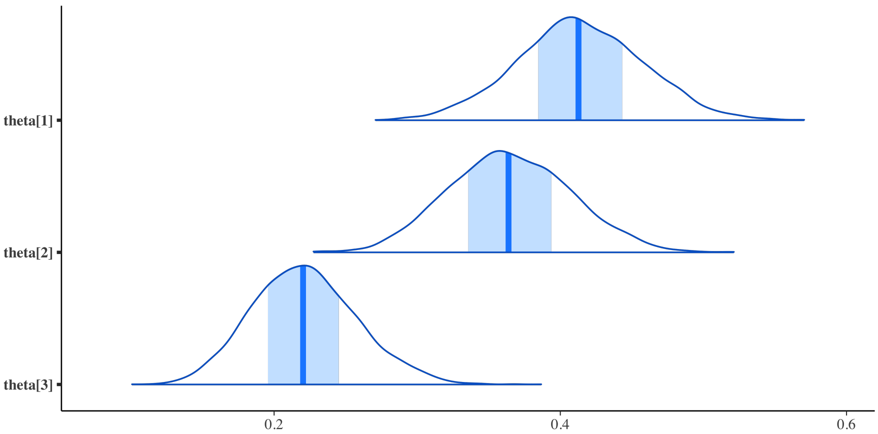


図 24.1　比率ベクトルの推定結果

らそれまでなのですが，仮説検定のように考えてみましょう。三社それぞれが均等に選ばれている，つまり $\pi_1 = \pi_2 = \pi_3$ というのはあり得そうにないですが (ピッタリ一致するはずはない)，たとえば

- A 社よりも D 社の方が好まれている
- S 社よりも D 社の方が好まれている
- D 社が最も好まれている

といった仮説は生成量を使って検証できそうです。1 つ目は $\pi_1 > \pi_2$，2 つ目は $\pi_2 > \pi_3$ が成立する割合から考えれば良いからです。3 つ目の仮説はこの 2 つと同じじゃないか，と思うかもしれませんが，正確には $\pi_2 > \pi_1$ かつ $\pi_2 > \pi_3$ という両方の可能性が成立していること，と考える必要があります。こうした「A かつ B」のような命題は**連言命題**といいます。

これを検証する生成量の作り方は，次のようになります。他にもいろいろ考えられそうですね。

code : 24.3　多項分布からのデータ生成モデル

```
...前略...
generated quantities{
    int<lower=0,upper=1> FLG1;
    int<lower=0,upper=1> FLG2;
    int<lower=0,upper=1> FLG3;

    if(pi[1] > pi[2]){ FLG1 = 1; } else { FLG1 = 0; }
    if(pi[2] > pi[3]){ FLG2 = 1; } else { FLG2 = 0; }
```

```
    if(pi[2] > pi[1] && pi[2] > pi[3]){ FLG3 = 1; } else { FLG3 = 0; }
}
```

24.3.1 対応のない二変数の場合

続いてクロス表の検定に行きましょう。表 24.2 にあるように，携帯キャリアの選び方と性別の関係をみたいと思います。男性と女性は独立したカテゴリですから，先ほどの多項分布モデルが 2 つ同時にあることと同じです。

男性は 21:30:13 という観測度数が得られていますが，これは $\boldsymbol{\pi_M} = c(\pi_1^M, \pi_2^M, \pi_3^M)$ という母比率を持っており，また女性は 30:15:14 という観測度数から，$\boldsymbol{\pi_F} = c(\pi_1^F, \pi_2^F, \pi_3^F)$ という母比率だったのではないか，と推測することになります。

「男性と女性とで携帯キャリアの選択率に違いがあるか」ということが検証したい命題になるかと思いますが，何を持って違いがあるとするか，ということを厳密に考えて生成量を作らなければなりません。たとえば，

- 男性は女性よりも A 社を好む → $\pi_1^M > \pi_1^F$
- S 社は女性の方が多く選んでいる → $\pi_3^M < \pi_3^F$
- 男性は D 社が一番好きで女性は A 社が一番好き → $\pi_2^M > \pi_1^M$ かつ $\pi_2^M > \pi_3^M$ かつ $\pi_1^F > \pi_2^F$ かつ $\pi_1^F > \pi_3^F$

というように考えられます。とくに 3 番目の命題は，「一番好き」というのを「他のどの群と比べても大きいという条件が同時に成り立っている」と考える必要がありますし，「男性は・・・で，女性は・・・」というときの「で」を論理的には「かつ」，確率的には「同時に」という意味で捉えないといけないところがポイントです。

このように，生成変数を使うとさまざまな仮説を検証することが可能です。集計表に関するデータは，官公庁が公開しているものなどを含め，身の回りのさまざまなところで目にできます。これらの資料を見た時に，標本統計量だけでなく母比率にまで思いを馳せて，検証可能な仮説を色々考えてみると良いでしょう[*8]。

ところでこの生成量を使った連言命題の検証については，今回のデータから考えられる事後分布に依存していることを忘れないようにしてください。今回のデータから言える「命題が成立する確率」は，事前分布とデータに基づいている結果ですから，「仮説が正しい確率」とまで言い切るのは危険です。あくまでもモデルに基づく推定であることを忘れないようにしましょう。

24.4 κ 係数の算出

先ほどのクロス集表の例では，男性と女性という独立した群から作られていましたので，2 つの母集団を別々に考えることができました。しかしたとえば，「法案 A,B それぞれに対して賛成か反対か」といった調査をした場合には，「A にも B にも賛成」「A には賛成，B には反対」「A には反対，B には賛成」「A にも B にも反対」という 2×2 のクロス集計表が得られますが，これは一人の人間が 2 つの質問に答えていますので**対応のある**データになります。対応のあるデータの場合は，先ほどと同じように独立した確率分布を考えるわけには行きません。

[*8] たとえばテレビのバラエティ番組や情報番組においても，該当アンケート結果についてコメンテータが色々物申すことが少なくありません。「エビフライのしっぽは食べるのか，残すのか」とか「焼き鮭の皮は食べるのか残すのか」といった些細なことについて，無作為抽出とは思えないような該当インタビューをし，僅差でも多数派を勝者と見立てて騒ぎ立てる様をみる度に，筆者は「せめてちゃんと検定，推定してから結論づけるべきだ」と思い，常々腹立たしく感じていました。しかし妻に「そういう些細なことを，僅差でも大袈裟にいうことで，お茶の間に盛り上がる話題を提供しているのであって，学術報告ではない」と諭されたことがあって，なるほどそういうものかと納得した次第です。

また「ある病院にいったら健康だと診断されたが，別の病院に行ったら異常が発見された」とか，「ある専門家の見立てでは問題のある行動といわれたが，別の専門家の見立てでは問題ないといわれた」，「ある映像を見て判断する課題で，審査者 A と B の判断が一致したりしなかったりする」といったシーンを考えてみてください。これらのシーンでは，2×2 のクロス集計表が表 24.3 のようになっていると考えられます。ここでは Yes/No としてありますが，Hit/Miss でも健康/異常でもいいのですが，ともかく 2 つの判断が合致したかどうか，ということが問題になるシーンです。

表 24.3　対応のあるカテゴリ判断

		A		
		Yes	No	
B	Yes	a	c	
	No	b	d	
				n

このような時も，行と列の判断が独立しているとは言えません。むしろどの程度重複しているかのほうが問題になるわけです。こういったときは，判断が一致した程度を**カッパ係数 (kappa coefficient)** で表現できます[*9]。この係数は，観測された一致度，すなわち

$$p_o = \frac{(a+d)}{n}$$

と，偶然の一致度

$$p_e = \frac{(a+b)}{n}\frac{(a+c)}{n} + \frac{(b+d)}{n}\frac{(c+d)}{n} = \frac{(a+b)(a+c)+(b+d)(c+d)}{n^2}$$

から，次のように計算されます[*10]。

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

この式の分子は「合致した割合から偶然の一致をひいたもの」であり，分母は「偶然ではない割合」になっていることから，一致の程度を表す指標として考えられるのです。この κ 係数は -1 から $+1$ の範囲の値を取ります。

このように，2 つの群わけが独立ではない場合の確率はどう考えればいいでしょうか。$a:b:c:d$ を $\pi_1:\pi_2:\pi_3:\pi_4$ と考えるのでは 4 つのセルが独立だと考えていることになってしまいます。そうではなく，順番にまず A が確率 α で Yes と判断する，ということを考えましょう。No と判断する確率は $1-\alpha$ です。次に B の判断ですが，A が Yes と判断した時に B が Yes と判断する確率を β としましょう。また A が No と判断した時に B も No と判断する確率を γ と考えます。

そうすると，両者が Yes と答える確率 $\pi_a = \alpha\beta$，A が No で B が Yes と答える確率 $\pi_b = (1-\alpha)\beta$，A が Yes で B が No と答える確率 $\pi_c = \alpha(1-\gamma)$，両者が No と答える確率 $\pi_d = (1-\alpha)\gamma$ のようにあらわすことがで切るようになります (図 24.2)。κ 係数はこれら $\pi_a \sim \pi_d$ の組み合わせで計算できますから，生成量を使えば算出できますね。

それではこれをコードにしていきましょう。確率ですから，変数の範囲は 0 から 1 に制限し，事前分布もこの範囲の一様分布にしています。

[*9] カッパ係数は河童ではなく，ギリシア文字の k にあたる κ です。

[*10] A が Yes と判断する割合 ×B が Yes と判断する割合を，A が No と判断する割合 ×B が No と判断する割合に足している数字が p_e です。割合 (=確率) の積なので二つの条件が偶然に一致する割合を，Yes-Yes のケースと No-No のケースで算出し，合わせたものと理解すればいいでしょう。

		Aの判断 Yes	Aの判断 No
Bの判断	Yes	$\pi_a = \alpha\beta$	$\pi_c = (1-\alpha)(1-\gamma)$
	No	$\pi_b = \alpha(1-\beta)$	$\pi_d = (1-\alpha)\gamma$

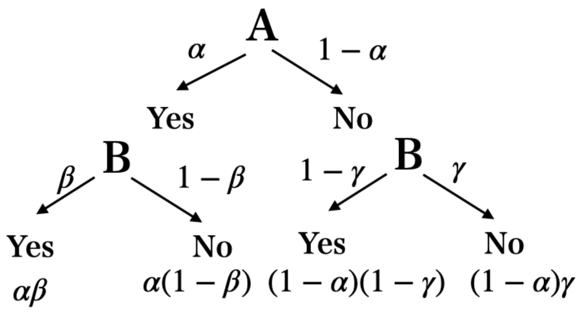


図 24.2　対応のあるカテゴリ判断の確率

code : 24.4　κ 係数を計算するモデル

```
data{
    array[4] int Y;
}

parameters{
  real<lower=0,upper=1> alpha;
  real<lower=0,upper=1> beta;
  real<lower=0,upper=1> gamma;
}

transformed parameters{
  simplex[4] Pi;
  Pi[1] = alpha * beta;
  Pi[2] = (1-alpha) * (1-gamma);
  Pi[3] = alpha * (1-beta);
  Pi[4] = (1-alpha) * gamma;
}

model{
  Y ~ multinomial(Pi);
  alpha ~ uniform(0,1);
  beta ~ uniform(0,1);
  gamma ~ uniform(0,1);
}

generated quantities{
  real po;
```

```
  real pe;
  real kappa;
  po = Pi[1] + Pi[4];
  pe = (Pi[1]+Pi[2])*(Pi[1]+Pi[3]) + (Pi[2]+Pi[4])*(Pi[3]+Pi[4]);
  kappa = (po-pe)/(1-pe);
}
```

カテゴリカルな分布はこのように度数に関するモデルを考えることができますし，この分布を応用して潜在的な変数がカテゴリカル分布に従うと考えると，分類わけを表現することもできます。さまざまな応用例が思いつくのではないでしょうか。

24.5 課題

次の 2 つの課題について，考察を導くための計算をする R/Stan コードとともに，回答を提出してください。Rmd ファイルでの提出が望ましいですが，メモやコメントアウト，Word ファイル，Google ドキュメントなどでの提出も可とします。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

■多項分布と連言命題 とある棒状のお菓子にはさまざまな味のバリエーションがあるが，コーンポタージュ味，チーズ味，めんたい味，野菜サラダ味，たこ焼き味あたが人気上位に入るらしい。そこで実験を行なって，最も美味しい味に投票してもらったところ，コーンポタージュ味が 105 票，チーズ味が 80 票，めんたい味が 75 票，野菜サラダ味が 60 票，たこ焼き味が 45 票であった。この時，次の仮説を検証するコードを書き，考察してください。

- コーンポタージュ味がチーズ味よりも好まれているといえるかどうか
- コーンポタージュ味がめんたい味よりも好まれているといえるかどうか
- コーンポタージュ味が他のどの味よりもこのまれてるといえるかどうか

■一致率の判断 新型コロナウイルスに対するスピード臨床検査が開発された。新型コロナに罹っていることがわかっている 40 名のうち，新しいスピード検査法では 34 名を陽性と判断できたが，6 名はコロナであると判断できなかった。また新型コロナに罹っていないことがわかっている 200 名の，新しいスピード検査法では 190 名はコロナでないと正しく判断できたが，10 名はコロナであると間違って判断してしまった。この検査法のデータから一致率を計算し，新しいスピード臨床検査法が有用であるといえるかどうか，事後分布に基づいて考察してください[11]。

[11] この課題は Lee and Wagenmakers (2013) の Pp.59，練習問題 5.3.1 を参考にしています

第 25 章

一般化線形モデル

25.1　一般線形モデル

ここまで検定と線形モデルが同じ形をしていることについては，何度か指摘してきました。検定モデルは，

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

という形をしており，ここでとくに X_i が統制群に属するか，実験群に属するかを $0/1$ で表現したものであったことを再確認しておきましょう。たとえば二水準モデルの場合，統制群の場合 $X_i = 0$ ですから，データは $Y_i = \beta_0 + 0 \times \beta_1 + \varepsilon_i = \beta_0 + \varepsilon_i$ であり全体平均に誤差がついただけの値になります。実験群の場合は $Y_i = \beta_0 + 1 \times \beta_1 + \varepsilon_i = \beta_0 + \beta_1 + \varepsilon_i$ になりますから，この β_1 の部分が効果の大きさを表しているのでした。

水準数が増えても同じことです。3 水準モデルの場合はベクトルを使って表記した方がわかりやすいので，データセットを $\boldsymbol{Y}$ と表すと，

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon} \tag{25.1}$$

であり，詳しく書くと

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 1 & -1 & -1 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

のようになります[*1]。ここで行列 $\boldsymbol{X}$ はその群に属しているかどうか (その群の効果が発動するかどうか) を $0/1$ で表している**デザイン行列 (design matrix)** であり，有無という状態に一対一対応した**名義尺度水準**の数字です。説明変数が名義尺度水準になっているだけで，これが間隔尺度水準以上の**量的な**データであれば，**回帰分析 (regression analysis)** になるわけです。回帰分析は一般に，

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \varepsilon$$

あるいはベクトルで書くと

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon} \tag{25.2}$$

となります。この通り式で書くと 25.1 と 25.2 にはなんら違いがないことがわかりますね。

このように**要因計画と線形モデルは同じである**，ということを含意して，これをまとめて**一般線形モデル (General Linear Model)** と言います[*2]。

---

*1 $\beta_3 = 0 - (\beta_1 + \beta_2)$ となっていることに注意。

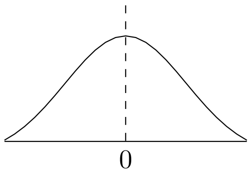
*2 一般，というのは同じだという意味があって，辞書には「私は彼女と同じい罪を犯したも一般だ」といった用法が載っています。

25.1.1　回帰分析の確率モデル

ところで，この一般線形モデルでは誤差が正規分布に従うという仮定を置いていました。すなわち，$\varepsilon \sim N(0,\sigma)$ というわけです[*3]。線形モデルの部分，すなわち $\hat{Y}_i = \beta_0 + \beta_1 X_i$ のところは確率的ではありませんから，データ全体では次のようになっていることになります。

$$y_i = \underbrace{\boxed{\beta_0 \quad + \quad \beta_1 x_i}}_{\text{確率的でないところ}} + \underbrace{\boxed{e_i}}_{\text{確率的}}$$

誤差 e_i は平均 0 を中心に確率的に散らばりますが，その前のところは確率的ではありませんので，これを組み合わせた式全体としては，最後に確率的な散らばりがひっついているような形になります。数式ではありませんが，イメージでいうと次のような形でしょうか。

$y_i = \beta_0 + \beta_1 x_i +$ 

このことから，モデル全体の確率的挙動は，

$$Y_i \sim N(\beta_0 + \beta_1 X_i, \sigma) \tag{25.3}$$

ということになります。これが正規分布の確率モデル的表現です。一般線形モデルですから，説明変数 X が連続的でも離散的でも同じモデルです。また，この正規分布は平均と SD でその形状が定められるのでした。平均は分布の位置を表すので，別名**位置母数** (**location parameter**) といいます。ちなみに SD のことは別名**尺度母数** (**scale parameter**) と呼ぶことがあります。一般線形モデルは平均的な位置を線形で予測して，データはそれに付随した誤差が幅 σ でぶれる，と考えます。そういう意味では，図 25.1 のように，回帰線にそって誤差分布があり，結果として実現値が得られているイメージ図のほうがわかりやすいかもしれません。

これを確認した上で，では回帰分析をベイズ的に推定するコードを書いてみましょう。設計図も一応準備しておきます。図 25.2 では，μ が $\beta_0 + \beta_1 X_i$ で作られているので，矢印のところは $\sim$ ではなく $=$ になっています。また，未知数 β_0, β_1 については平均 0，SD100 のとても幅の広い正規分布を置いています。一様分布でもいいのですが，原理的に $\pm\infty$ までの幅があり，どこか 1 つの値に収束するだろう＝単峰性があるだろう，ということで正規分布を選んでいます。誤差 SDσ については，半コーシー分布[*4] 設計図をそのままコードに起こせば一丁上がりです。事後予測分布も生成するコードを書いてみました。

code : 25.1　線形モデルの Stan コード

```
data{
  int N;
  array[N] real X;
  array[N] int Y;
}
```

[*3] 誤差の平均はゼロであるのは，偶然誤差の仮定から必然的です。

[*4] 正の値しか取らないように，コーシー分布を 0 のところで半分に折り畳んだ形にした分布です。Stan 上では `lower=0` の制限をかけることで正の範囲からしかサンプリングしない，ということで表現します。

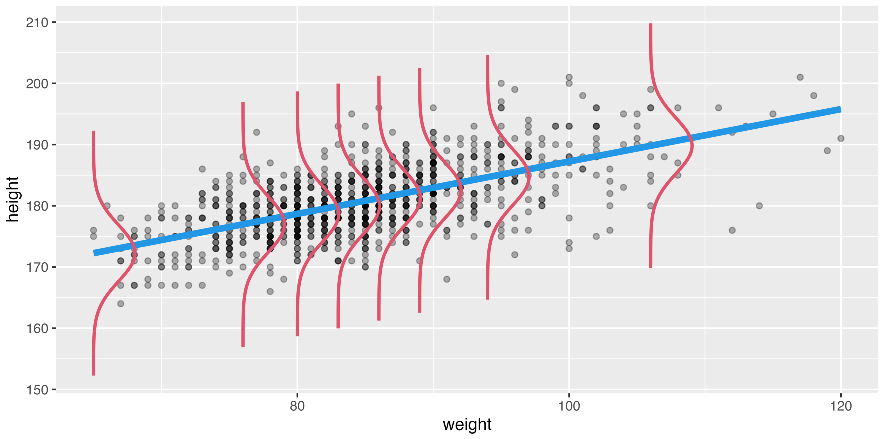


図 25.1 回帰線に伴う誤差分布

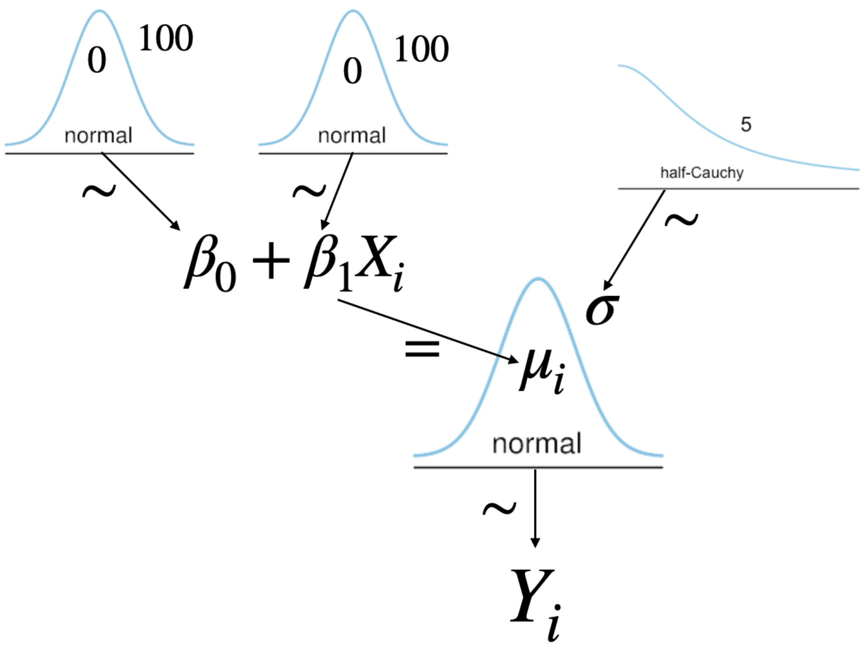


図 25.2 線形モデルの設計図

```

parameters{
  real beta0;
  real beta1;
  real<lower=0> sig;
}

transformed parameters{
   array[N] real mu;
   for(i in 1:N){
      mu[i] = beta0 + beta1 * X[i];
```

```
      }
  }

  model{
    // model
    for(i in 1:N){
      Y[i] ~ normal(mu[i],sig);
    }
    // prior
    beta0 ~ normal(0,100);
    beta1 ~ normal(0,100);
    sig ~ cauchy(0,5);
  }

  generated quantities{
    real predX[N];
    for(i in 1:N){
      predX[i] = normal_rng(mu[i],sig);
    }
  }
```

これがどのような挙動をするのか，実際のデータセットを使って検証してみましょう。サンプルデータとして，`baseball2020.csv` を使います。このデータセットはプロ野球データ Freak[5]さんから取ってきたものです[6]。

この野球データセットにはさまざまな情報が含まれていますが，身長と体重のデータだけ取り出して回帰分析をしてみましょう。身長を体重から予測することにします。ここで身長は正規分布に従っていると考えられますから，一般線形モデルが適切ですね。データの散布図を書いて，線形関係にありそうかどうか確認しておきます (図 25.3。

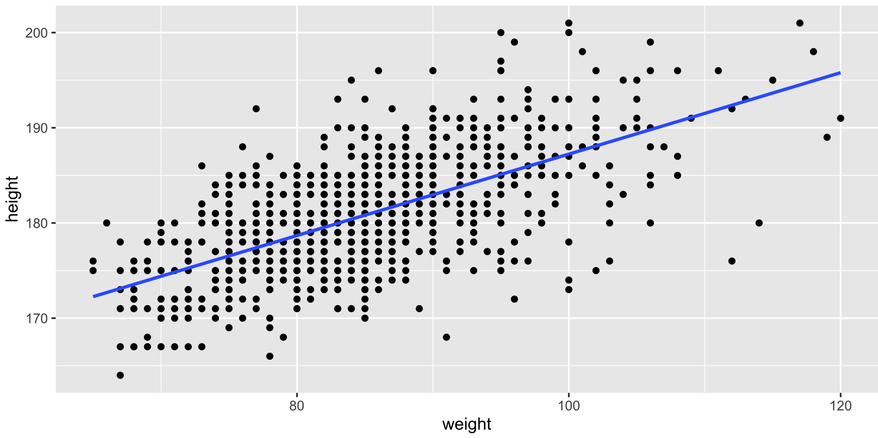


図 25.3　身長を体重で回帰するモデル

そこそこ相関がありそうですね。では回帰モデルを当てはめてみましょう。

[5] https://baseball-data.com/

[6] データセットはシラバスのサイトでも提供していますし，著者の個人サイト (https://kosugitti.github.io/kosugitti10/index.html) にはスクレイピングのコードを公開しています。

code : 25.2　一般線形モデルを実行するコード

```
1  dat <- read_csv("baseballDecade.csv") %>% filter(Year=="2020年度")
2  dataSet <- list(N = NROW(dat), Y = dat$height, X = dat$weight)
3  ## cmdstanrで実行する場合
4  modelC <- cmdstan_model("LM.stan")
5  fit <- modelC$sample(
6    data = dataSet,
7    chains = 4,
8    parallel_chains = 4,
9    iter_warmup = 1000,
10   iter_sampling = 2000
11 )
12 ## 実行後のファイルをstanfit形式に置き換えておく
13 fit.stanfit <- fit$output_files() %>% rstan::read_stan_csv()
14 ## 事後予測を取り出す
15 predY <- rstan::extract(fit.stanfit)$predY
16 # 事後予測分布の描画
17 bayesplot::ppc_dens_overlay(y = dataSet$Y, yrep = predY[1:10, ])
18 bayesplot::ppc_intervals(
19     y = dataSet$Y,
20     yrep = predY,
21     x = dataSet$X,
22     prob = 0.5,
23     prob_outer = 0.95
24 )
```

■コード解説

1 行目　データを読みこみます。

2 行目　stan に与えるデータセットをリストで作ります。

3-11 行目　`cmdstanr` でコンパイルする例です。`rstan` パッケージを使っていただいても構いません

13 行目　実行後のファイルを stanfit 形式に置き換えています。`rstan` パッケージを使っている人はこの行を実行する必要がありません。

15 行目　事後予測分布の一部です。予測値は MCMC サンプル数 (ここでは 8000 点)× サンプルサイズ (野球選手 959 人) ありますが, このうち最初の 10 人についての実際のデータ y と予測値 `yrep` を重ねて表示しています。全員分表示させると非常にサイズが大きくて重たいからです。

16-22 行目　データの散布図に予測値とその 50% および 95% 確信区間をプロットしました。

今回の分析結果を単純に表示するなら, 次のようになります (出力 3)。

cmdstanr の出力 3: ベイズ推定の結果

```
variable   mean median   sd  mad     q5    q95 rhat ess_bulk ess_tail
beta0    144.43 144.42 1.46 1.46 141.97 146.83 1.00     2344     2554
beta1      0.43   0.43 0.02 0.02   0.40   0.46 1.00     2339     2507
sig        4.57   4.57 0.11 0.10   4.40   4.75 1.00     3171     2952
```

これまで習ってきた**最尤法 (Maximum Likelihood method)** の結果と出力の違いを比較しましょう (出力 25.1)。

R の出力 25.1: ML 推定の結果

```
Call:
lm(formula = height ~ weight, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max 
-16.3680  -2.8245  -0.0963   3.0368  14.9037 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 144.45986    1.44682   99.85   <2e-16 ***
weight        0.42775    0.01693   25.27   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.573 on 957 degrees of freedom
Multiple R-squared:  0.4002, Adjusted R-squared:  0.3996 
F-statistic: 638.6 on 1 and 957 DF,  p-value: < 2.2e-16
```

最尤法では，切片が 144.45986，傾きが 0.42775 という**点推定**だけが表示される形になっていますが，MCMC を使ったベイズ推定では切片の平均 (**EAP 推定値**) が 144.43，中央値 (**MED 推定値**) が 144.42，95% 確信区間が [141.97,146.83] となっています。同じく傾きの平均が 0.43，中央値も 0.43，95% 区間が [0.40,0.46] となっています。結果が分布として得られますから，初めから幅を持った推定になっていることが確認できます。

事後予測分布 (図 25.4) をみると，いくつか予測幅から外れているデータ点はありますが，概ね回帰線の範囲に捉えられており，線形モデルの当てはまりがそこそこ良いことが確認できます。事後予測分布が元データとよく当てはまるということは，データ生成メカニズムとして正規分布や線形モデルを想定したわれわれのモデルが悪くないんだ，と解釈していいでしょう。

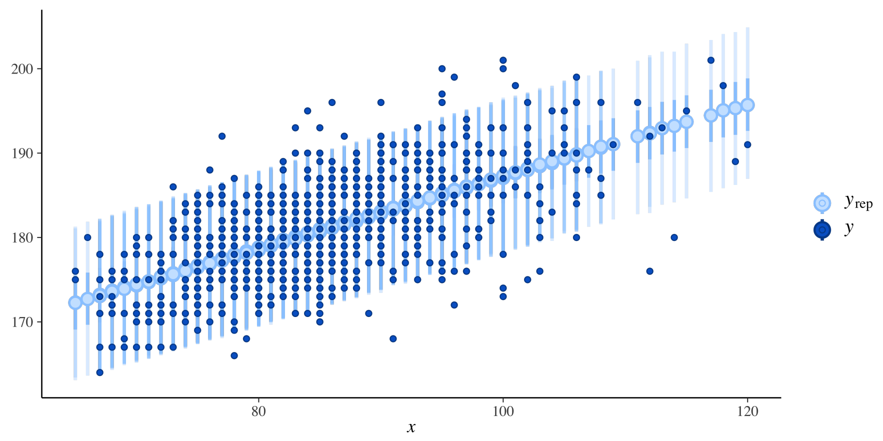


図 25.4　事後予測分布

25.2 データに合わせた確率分布

今回は身長が従属変数，体重が独立変数でした。ここで独立変数を**名義尺度水準**のもの，すなわち離散変数だと考えると要因計画のようになります。Stan のコードは書き換える必要がありますが，線形モデルとして表現できるところは同じです[*7]。

では**従属変数**を離散変数に置き換えたらどうなるでしょうか。野球のデータの例で考えます。野球は野手と投手に大きく分けることができます。投手は野手と違って毎日登板するということはありませんから，当然のことながら出場試合数は野手と比べ物になりません[*8]。図 25.5 には，年収 5000 万円を超える一流の選手たちに限定し，試合数で野手か投手かを予測するような分析ができないかと，回帰直線を引いたものを示してあります。この直線はどうみても，データにうまく当てはまっているとは言えませんね。

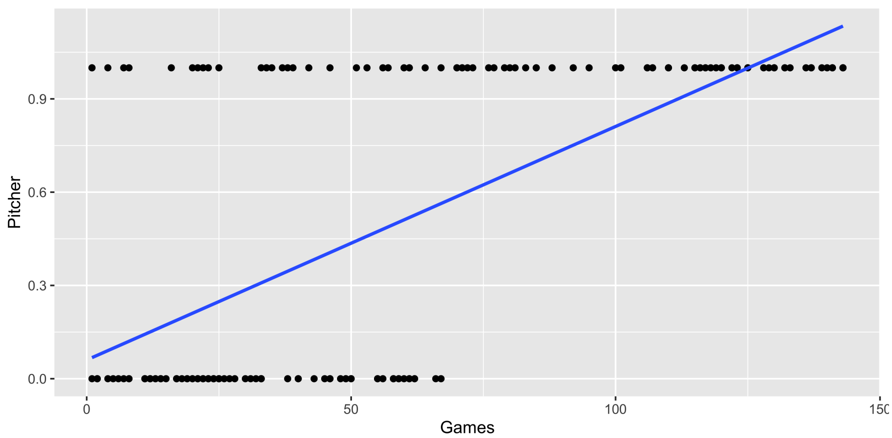


図 25.5 離散変数が目的変数だとおかしなことになる

従属変数が離散的，とくに今回は野手か投手か，という二択の問題になっていますから，データとしては 0/1 しかないわけです。この二種類の数字が正規分布から出てくると考えるのは，流石におかしなことになってしまうでしょう。

そこで前回学んだ**離散確率分布**の登場です。データが二値しか取らないというのは，コインの表と裏のようなものですから**ベルヌーイ分布 (Bernoulli Distribution)** に従うと考えるのが自然です。ベルヌーイ分布はパラメータ θ で表 (1) が，$1-\theta$ で裏 (0) がでるというように，パラメータが 1 つで，このパラメータがそもそも「表が出る確率」を表しているわけです。今回は野手であれば 1，投手であれば 0 というように考えることにしましょう。野手か投手かを表す変数で，野手になる確率 θ が試合数によって変わる (予測できる) ということを考えるのです。この予測の仕方については，「試合数が多いのは野手である確率が高い」という線形的な関係が考えられますから，$\theta_i = \beta_0 + \beta_1 X_i$ という式を考えることにします。ここで X_i は試合数を表す変数ですね。

ところがここで，注意すべきことが 1 つあります。θ は野手になる (表が出る) 確率ですから，0 から 1 の範囲の数字しか取れません。プロ野球の試合数は 1 試合から 140 試合ぐらいありますので，この数字をその

[*7] 実際，R の最尤推定関数の `lm` は，説明変数が `factor` 型であれば関数を書き換えることなく，分散分析のような結果出力をしてくれるのでした。

[*8] メジャーリーガーの大谷選手は特殊すぎるケースです。投手は一試合で 100 球程度投げますが，当然徐々に握力が弱くなり，身体的な疲労も翌日，翌々日まで残り続けるものなのだそうです。彼は投げた次の日に DH で打席に立ちますから，とんでもないことです。

まま伸ばしたり (β_1 倍) ずらしたり (β_0 を加える) しても，0 から 1 の範囲に収めるのは難しそうです。しかも上手くその範囲に収めたとしても，そもそも直線なのでどうあってもデータと合致しないことになるでしょう。そこで，この数字をロジスティック関数で変換することを考えます。

ロジスティック関数とは次のような式で表されるものです。

$$y = \frac{1}{1+\exp(-x)}$$

ここで $\exp(x) = e^x$ であり，正規分布の式の中にも出てくる関数です[*9]。関数の形は図にするとよくわかります (図 25.6)[*10]。

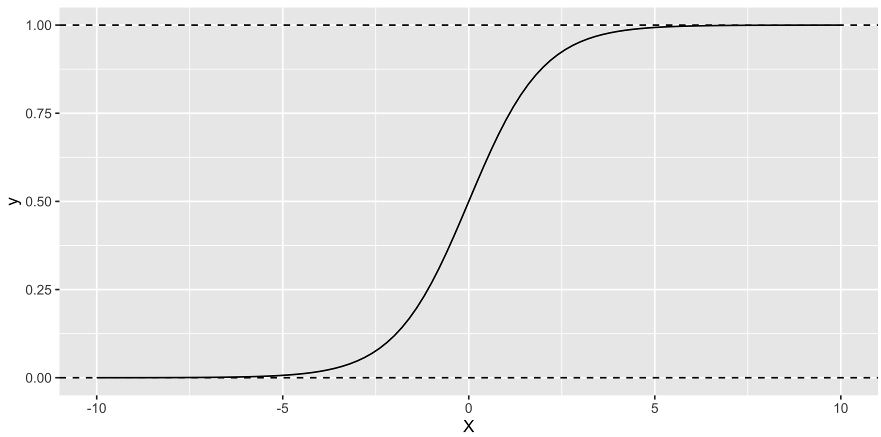


図 25.6　ロジスティック関数

この S 字カーブは，x がどのように変わっても y は必ず 0 から 1 の範囲に入りますので，今回の目的にもってこいです。つまり，今回「野手か投手か」を判断する結果変数 Y_i は，次のような分布に従っていると考えます。

$$Y_i \sim Bernoulli(\theta) = Bernoulli(logistic(\beta_0 + \beta_1 X_i))$$

これを使ってモデルを書いてみましょう。まずは設計図です (図 25.7)

ここまで書ければ，Stan コードに起こすことも簡単ですね。Stan には `inv_logit` という関数がありますのでそれを使うと良いでしょう。

code : 25.3　ロジスティック回帰分析のコード

```
data{
  int N;
  array[N] int<lower=0,upper=1> Y;
  array[N] real X;
}

parameters{
```

[*9] e はネイピア数とも呼ばれる自然対数の底，約 2.7182... の実数です。

[*10] すでに第 4 講で出てきた関数ですが，あの時は累積正規分布に近似する関数として紹介していました。今回は $\pm\infty$ の範囲の数字を 0 から 1 に変換するものとして改めて紹介しています。なお，今回は累積正規分布への近似を目的としているわけではないので，係数 1.7 は外してあります。

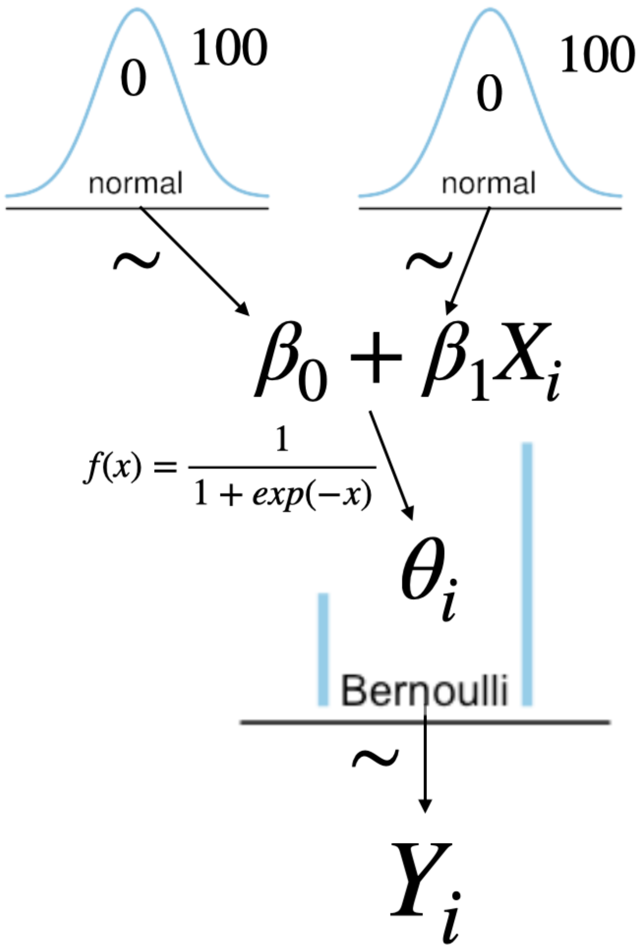


図 25.7 ロジスティック回帰分析の設計図

```
  real beta0;
  real beta1;
}

transformed parameters{
  real theta[N];
  for(i in 1:N){
    theta[i] = inv_logit(beta0 + beta1 * X[i]);
  }
}

model{
  // model
  for(i in 1:N){
    Y[i] ~ bernoulli(theta[i]);
  }
  // prior
```

```
  beta0 ~ normal(0,100);
  beta1 ~ normal(0,100);
}
```

Code25.3 がこのモデルになります。ベルヌーイ分布を使っているのでベルヌーイ回帰と読んでもいいのですが，一般には**ロジスティック回帰 (logistic regression)** と呼ばれています。これに適切なデータセットを与えて (R コード 25.4)，推定した結果が出力 3 になります。

code：25.4 ロジスティック回帰分析を実行する R コード

```
1  dat2 <- dat %>%
2    dplyr::mutate(Pitcher = if_else(position == "投手", 0, 1)) %>%
3    dplyr::filter(salary > 5000) %>%
4    dplyr::select(Games, Pitcher) %>%
5    na.omit()
6
7  model <- cmdstanr::cmdstan_model("cmdstan/logistic.stan")
8  dataSet <- list(N = NROW(dat2), Y = dat2$Pitcher, X = dat2$Games)
9
10 fit.logistic <- model$sample(
11   data = dataSet,
12   chains = 4,
13   parallel_chains = 4,
14   iter_warmup = 1000,
15   iter_sampling = 5000
16 )
17
18 ## 簡易表示
19 fit.logistic$print(c("beta0", "beta1"))
```

■コード解説

1-5 行目 先ほど読み込んだデータに `Pitcher` 変数を作ります。これは投手であれば 0，野手であれば 1 になる変数です。その後，サラリーが 5000 万円以上のデータにし，変数 `Game`，`Pitcher` と必要なものだけ抜き出します。最後に `na.omit` で欠損値のあるデータは除外しました

7 行目 `cmdstanr` でコンパイルする例です。`rstan` パッケージを使う場合は適宜変更してください。

8 行目 stan に与えるデータセットをリストで作ります。

10-13 行目 サンプリングするコードです。

19 行目 結果の出力です。出力 3 のように得られます。

rstan の出力 3: ロジスティック回帰分析の結果

```
 variable  mean median   sd  mad    q5   q95 rhat ess_bulk ess_tail
    beta0 -2.77  -2.75 0.40 0.41 -3.47 -2.13 1.00     4978     5537
    beta1  0.05   0.05 0.01 0.01  0.04  0.07 1.00     4832     5417
```

出力結果はグラフにした方がわかりやすいでしょう。図 25.8 にあるように，上下の黒い点がデータ点，真ん中を通る S 字カーブが回帰線になります。回帰線は **MAP 推定値**で描きましたが，50%，95% の幅であり得る可能性の範囲についても薄く書き足しています。これを見ると，ちょうど 50 試合目ぐらいの時にカーブが

Y 軸 0.5, すなわち表裏半々の点を通りますから，50 試合以上出ている人はより野手と判断されがち，と考えれば良いでしょう。

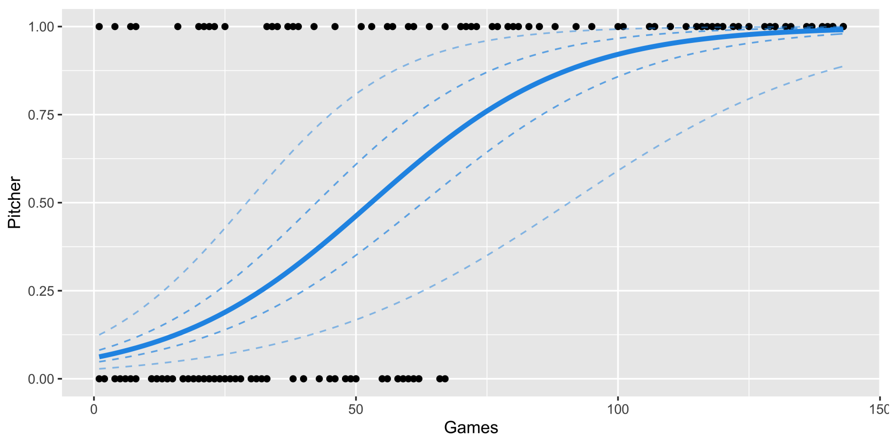


図 25.8　ロジスティック回帰分析の結果

さて，このように従属変数が離散的である場合に，離散確率分布をつかって線形モデルを当てはめることができるのをみてきました。図 25.8 に示されているように，出力される関数は S 字のカーブ型をしていますが，予測モデルとしては**線形**です。今回 $\theta_i = \beta_0 + \beta_1 X_i$ としたように，θ は線形モデルであり，予測変数 X_i が線型結合されているので，これも広い意味で線形モデルになるわけです。線型結合したものを，そのまま確率分布に入れることができなかったので，変数を変換してモデルの中に組み込むという工夫が必要でした。この変換関数のことをとくに**リンク関数 (Link function)** と言います。線型結合とパラメータの形をつなげる (link する) 関数だからです。

このように，リンク関数をかませてさまざまな確率分布に対応させる線形モデルを総称して，**一般化線形モデル (Genrealized Linear Model;GLM)** といいます。この章の初めに出てきた**一般線形モデル**とは違いますので注意してくださいね。日本語では「化」があるかないか，英語では “‘$-ed$ がつくかどうかだけの違いですが，一般線形モデルが正規分布一般だったものに対し，それ以外の確率分布にまで一般化したのがこちらです。

ちなみに今回はベイズ推定で一般化線形モデルを実践してみましたが，最尤法による推定も可能です。可能ではあるのですが，ベイズ推定の方が拡張性が高いこと，確率モデルとしてまとめて理解しやすいと筆者は考えています。久保 (2012) は線形モデルの発展段階を推定法とともに，図 25.9 のように表しています。一般線形モデルは**最小二乗法**による推定でも十分だったのですが，確率モデルとして考えたときに**最尤法**による推定が必要になってきました。さらに複数の分布を混ぜるような**一般化線形混合モデル (Generalized Mixed Linear Model)** にまで発展すると，最尤法でカバーしきれないところが出てきました (網掛けが部分的になっていることでそれを表しています)。分布の上に分布がくるような，幾重にも層が厚くなるモデルのことを階層モデルと言いますが，そうなってくるともはや最尤法では推定できません。ここで「形はわからなくても乱数は発生させられる」という **MCMC** の方法が出てきます。MCMC はすでに説明した通り (→Pp.178)，乱数による事後分布の近似法です。この方法はベイズ統計学に基づいた方法ですから，**ベイズ推定**ともいわれるわけです。ベイズ推定は最尤推定の結果と解釈の仕方が少し異なりますから，そこに注意する必要がありますが，ほぼ無敵の推定法と言ってもいいでしょう。

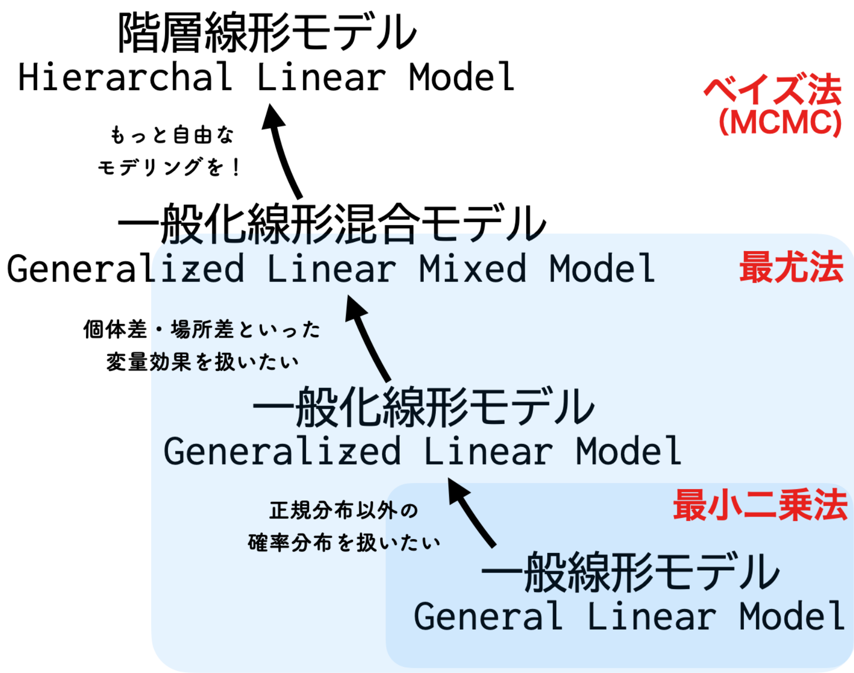


図 25.9　久保 (2012) による線形モデルの発展の図

25.3　リンク関数とパラメータの解釈

さてここでもう一度，先ほどのロジスティック回帰分析の結果をみてみましょう。EAP 推定値でいくと，$\beta_0 = -2.77, \beta_1 = 0.05$ でしたから，$\hat{Y}_i = -2.77 + 0.05X_i$ という関係だったことがわかります。

ところで一般線形モデルのうちは，この傾きは説明変数が一単位増えた時に，従属変数がどれぐらい増えるのかを表す数字でした。これが正規分布に従うモデルであれば，出場試合数が一試合増えれば野手に 0.05 ポイント近づく，という考え方をするのでした。しかし今回は事情が違います。この $\hat{Y}_i$ はこのあと，$\dfrac{1}{1+exp(-\theta_i)}$ と変換されてしまいますので，X_i が $X_i + 1$ になった値も変換されて確率の数字に変わっていくはずです。ここで少し，数式的な展開を追って何が起こっているか考えてみましょう。

まずは**ロジスティック関数**の表記を，少し改めます。同じことなのですが，$\exp(x) = e^x$ に書き改めて

$$\frac{1}{1+\exp(-x)} = \frac{1}{1+e^{-x}} = \frac{1}{1+\frac{1}{e^x}}$$

とします[*11]。ここで最後の形に $\dfrac{e^x}{e^x}$ をかけまして，

$$\frac{1}{1+\frac{1}{e^x}} = \frac{e^x}{e^x+1}$$

としましょう。さらに，$\hat{Y} = \beta_0 + \beta_1 X$ で，ベルヌーイ分布のパラメータ θ はこれによって変換されたものを代

[*11] 指数計算のルールで，マイナス乗は逆数になります。$a^{-n} = \dfrac{1}{a}$ です。

入することになりますから，

$$
\begin{aligned}
\theta &= \frac{e^{\hat{Y}}}{e^{\hat{Y}}+1} \\
(e^{\hat{Y}}+1)\theta &= e^{\hat{Y}} \\
\theta e^{\hat{Y}} + \theta &= e^{\hat{Y}} \\
\theta &= e^{\hat{Y}} - \theta e^{\hat{Y}} \\
\theta &= e^{\hat{Y}}(1-\theta) \\
\frac{\theta}{1-\theta} &= e^{\hat{Y}} \\
\log\left(\frac{\theta}{1-\theta}\right) &= \hat{Y}
\end{aligned}
$$

という関係が成り立ちます[*12]。つまり，

$$\log\left(\frac{\theta}{1-\theta}\right) = \beta_0 + \beta_1 X$$

ということですね。

この式の右辺は線形モデルです。左辺は**ロジット関数 (logit function)** と呼ばれるものです。線形モデルはロジット関数を意味し，ベルヌーイ分布のパラメータ θ は確率を表す数字でした。ですから係数の解釈は，線形モデルの方で一単位増えたことは，ロジット関数が一単位増えたことを意味するわけです。ちょっとややこしいですが，線形モデルを確率のパラメータにするには，ロジット関数の逆をすることですから，**ロジスティック関数は逆ロジット関数**とも呼ばれています。そう言えば，Stan の関数も `inv_logit` でしたね[*13]。

- 確率の数字 $\rightarrow$ 線形モデルにつなげる；ロジット関数；$\log\left(\frac{\theta}{1-\theta}\right)$
- 線形モデル $\rightarrow$ 確率の範囲に収める；ロジスティック関数あるいは逆ロジット関数；$\frac{1}{1+exp(-x)}$

また，線形モデルが表していたのは確率の比ですが，それの対数 (log) をとったものでした。対数の中身は確率 θ と確率 $1-\theta$ の比率ですから，裏が出る確率に比べて何倍表が出やすいか，といったことを表しているのです。これをとくに**オッズ (Odds)** と言います。賭け事の勝率などを表す言葉ですね。オッズの対数がロジットです。ロジットが正の数であれば分子の方が分母より大きい，つまりより表が出やすいことになり，負の数であれば分母の方がより大きい，つまり裏の方が出やすいということになります。

さて話は戻って，線形モデルで説明変数が一単位増えた時のことを考えてみましょう。わかりやすくするために，増える前を A，増えた後を B とします。

$$
\begin{aligned}
A &= \frac{1}{1+exp(-(\beta_0+\beta_1 X)} \\
B &= \frac{1}{1+exp(-(\beta_0+\beta_1 (X+1))}
\end{aligned}
$$

これはロジット関数を使って元に戻せますから，

$$\log\left(\frac{A}{1-A}\right) = \beta_0 + \beta_1 X$$

[*12] 最後の一行は，指数の反対が対数で，$\log_e^x = a$ は $e^a = x$ をあらわしていますから，指数を取るために両辺に log をかけています。

[*13] `inv` は inverse，「逆」の意味です。

$$\log\left(\frac{B}{1-B}\right) = \beta_0 + \beta_1(X+1) = \beta_0 + \beta_1 X + \beta_1$$

ここで傾き β_1 のことを考えたいのですが，β_1 はこの式の関係から，

$$\beta_1 = \log\left(\frac{B}{1-B}\right) - \log\left(\frac{A}{1-A}\right)$$

ということになりますね。ここで対数の差分は割り算ですから[*14]，

$$\beta_1 = \log\left(\frac{\frac{B}{1-B}}{\frac{A}{1-A}}\right)$$

となります。右辺が非常にややこしい形をしていますので，log を外してあげましょう。

$$\exp(\beta_1) = \frac{B}{1-B}\Big/\frac{A}{1-A}$$

右辺は A のオッズに対する B のオッズになりました。オッズの比なのでこれを**オッズ比 (Odds ratio)** と言います。左辺はロジスティック回帰分析の係数を exp 関数に入れたものですが，これが説明変数の増加前のオッズと増加後のオッズの比率です。つまり一単位増加した時にどれぐらい生じやすくなるかの比の上昇率が，$\exp(\beta_1)$ だというわけです。

ややこしいですね！具体的な数字で考えてみますと，今回 $\beta_1 = 0.05$ だったわけですから，$\exp(0.05) = 1.051271$ です。つまり，一試合多く出ている選手はそうでない選手に比べて，1.05 倍野手と判断される確率が高い，ということになります。

25.4 まとめ

今日の話を簡単にまとめておきます。

- 一般線形モデルは，正規分布を使った線形モデルで，説明変数が連続変数でも離散変数でも同じ形で表すことができることを示している。
- 一般線形モデルの回帰係数をベイズ推定するモデルは，正規分布の位置母数に線形モデルを入れるだけ。
- 従属変数が離散変数になると，確率モデルは離散確率分布を使う必要がある。そのためには，説明変数の線型結合をリンク関数を使って変形し，適切な形にして推定する必要がある。これを一般化線形モデルといいます。
- 従属変数がベルヌーイ分布に従う例として，ロジット関数をリンク関数としたロジスティック回帰分析の実例をやってみました。
- 一般化線形モデルの場合は説明変数がリンク関数で変換されているから，そのまま解釈するのではなく，変換の意味を理解しながら注意深く解釈しよう。

一般化線形モデルを導入したことで，要するにリンク関数がわかっていれば色々な確率分布が使えるんだな，ということが見えてきたのではないでしょうか。

[*14] $\log_a \frac{M}{N} = \log_a M - \log_a N$ です。たとえば $8 = 2^3, 16 = 2^4$ ですから，$\log_2 168 = \log_2 2 = 1$ は $\log_2 16 - \log_2 8 = 4 - 3 = 1$ と同じことです。

これは従属変数の形に関わることですので，心理学的実践の上でも注意が必要です。たとえば「記憶のトレーニングをすると思い出せる単語の数が増える」といった研究をする時，従属変数は「思い出した単語の数」になると思いますが，実験群と統制群とでこれを **t 検定**する，というのが間違っていることはもうお分かりですね。「思い出した数」のようにカウントできる数字はマイナスになることはなく，0 以上の正の整数しか取りません。これは正規分布ではなく，**ポアソン分布 (poisson distribution)** に従うことになります。またたとえば「記憶のトレーニングをすると，50 の単語リストのうち思い出せる単語の割合が増える」と言った研究をする時，従属変数は今度は「思い出せた率」ということになると思います。これを実験群と統制群で t 検定する，というのが間違っているのも明らかですね。「思い出せた率」という比率のデータは 0 から 1 の範囲の実数で，正規分布ではないからです。これは前回やりました**二項分布 (binomial distribution)** に従うことになります。

ポアソン分布に従うような変数の場合は，リンク関数として対数関数 log が用いられます (逆リンク関数は指数関数 exp)。また二項分布に従う変数の場合はリンク関数としてロジット関数 (逆リンク関数はロジスティック関数) を用います。何気ない「度数」「比率」のようなデータを分析しようという時に，とりあえず数字が得られているから正規分布でいいや，と考えるのは間違いです。データの性質に応じたモデルを描くように心がけましょう。

25.5 課題

野球選手のデータの中で，とくに野手に限って考えます。

年俸の高い選手は，成績が良いから高い契約金が得られるのでしょう。言い換えると，年俸で成績がある程度予測できるかもしれません。そこで打率を従属変数に，年俸を独立変数にした一般化線形モデルを考えたいと思います。選手によっては試合への出場回数が違いますので，打席数 N で安打数が K 本である，という情報を用いましょう。これは二項分布に従うモデルということになります。

また，独立変数の年俸 (変数 `saraly`) は個人差が非常に大きいので，標準化してから使うようにしましょう。元のデータを加工するコードは提供しますので，二項分布に従う一般線形モデルの Stan コードを書いてください。また結果から，年俸が 1 単位上昇するとどういうことが言えるでしょうか。併せて報告してください。

考察を導くための計算をする R/Stan コードとともに，回答を提出してください。Rmd ファイルでの提出が望ましいですが，メモやコメントアウト，Word ファイル，Google ドキュメントなどでの提出も可とします。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

■ヒント　打席数と安打数とのデータがありますから，打率は計算したら出るように思えます。しかしここでは，打率を直接データとして与えるわけではありません。安打数が打席数に伴う二項分布から出現したものとし，打率は推定するべきパラメータとして考えます。その打率に線形モデルの構造を入れるようにしてください。

打率を安打数と打席数から計算するのは，確率変数の実現値から計算する標本平均を使うことと同じです。つまり，標本平均が母平均の推定値になるという**モーメント法**によるアプローチなのです。この方法では，シーズン初日，一打席だけ出場して 1 本ヒットを打つと，打率 100% だというのと同じです。あるいはその初打席でヒットが出なければ，その人は打率 0% なのでしょうか？本当にそんなことを信じている人はいませんよね。つまり 1 回の実現値だけで全体の推定値とするには偏りがある，ということはわかっているわけです。モーメント法による推定，あるいは**頻度主義統計学**による考え方では，無限回の試行の平均値，無限に打席に立ち続けた時に収束していく先の値として確率が定義されています。これに対して**ベイズ統計学**では，確実

さの指標として確率を用いますから，初打席の結果がどうであれ「本当はこんなもんじゃないよな」という事前分布との組み合わせで，真の打率を推定することになります。

第 26 章

階層線形モデル

前回は**一般線形モデル**から，**一般化線形モデル**へ，すなわち正規分布以外の確率分布を扱えるモデルへと展開しました。今回はさらにモデルを展開し，**一般化線形混合モデル (Generalized Linear Mixed Model;GLMM)** へと展開します。混合 mixed という言葉があるように，複数の確率分布の混ぜ合わせが含まれることが，GLMM の特徴です。

とはいえ，皆さんはすでに第 23 講でこの問題を扱っています。Within 計画・多水準のモデルがそれに当たります。Within 計画のモデルでは，個人差 μ_i に効果 δ_j が加わるという形でモデルを形成しました。ここで個人差 μ_i も正規分布に従うものと考えていました。個人差が従う正規分布，誤差が従う正規分布が混じりあってデータになりますから，これも Mix されたモデルと考えることができるのです。

ただしこの時は，いずれも正規分布のモデルでした。このモデルを正規分布以外のデータに対応させたい，それが今回のモデルになります。

26.1　一般化線形混合モデル

線形モデルを正規分布以外のモデルに対応させるためには，確率分布のパラメータの形にあうように**リンク関数 (link function)** で接続してやる必要があるのでした。たとえばベルヌーイ分布の場合，線形モデルは次のような確率の比 (**オッズ**) を表すものと考えます。

$$\log\left(\frac{\theta}{1-\theta}\right) = \beta_0 + \beta_1 X$$

このリンク関数の逆関数を考えてやると，

$$\theta = \frac{1}{1+exp(-(\beta_0+\beta_1 X))}$$

となり，**ロジスティック関数**と呼ばれるこの変換を通じて $0 \leq \theta \leq 1$ の範囲に入るようにしてから，**ベルヌーイ分布**に入れてやるのでしたね。

リンク関数と**逆リンク関数**，この違いがちょっとわかりにくいかもしれません。基本となる線形モデルは $\beta_0 + \beta_1 X$ ですので[*1]，これが確率分布とセットになる方がリンク関数です。あるいは，線形モデルを変えてしまう方が逆リンク関数だ，と考えてください。

[*1] より正確に，より一般的にいうと，説明変数は複数あっても構いませんから，$\beta_0 + \beta_1 X_1 + \beta_2 X_2 \cdots \beta_n X_n$ と書く方がいいのかもしれませんし，もちろんベクトルを使って $\boldsymbol{X\beta}$ とすればいいでしょう。この時，係数ベクトル $\boldsymbol{\beta}$ には切片項 β_0 も含まれていることを思い出してください。また，説明変数が複数になった場合は，**交互作用 (interaction)** 項を考えることもありますが，本書では簡略化のために含めずに考えています。

リンク関数の例をもう 1 つ。**ポアソン分布 (Poisson distribution)** という離散分布があります[*2]。これは 0 以上の整数を取る**カウント変数**に使われる関数です。たとえば野球選手のホームランの数，たとえば青年期における親友の数や交際経験の人数，記憶実験で思い出せた単語の数などは，負の数をとることも小数点を持った実数になることもありません。こうした数を数えるような変数が従うのがポアソン分布ですが，このパラメータ λ は正の実数です。ということは，線形モデルで算出される値を必ず正の数にしなければなりませんから，逆リンク関数は exp を使うことになります[*3]。

$$\lambda = \exp(\beta_0 + \beta_1 X)$$
$$\log(\lambda) = \beta_0 + \beta_1 X$$

ですから，リンク関数は log になります。

表 26.1 に確率分布，逆リンク関数，リンク関数の関係をまとめました。さらに Stan ではそれぞれの確率分布に加え，リンク関数とセットになった関数がすでに用意されています。`transformed parameters` ブロックで逐一変換しなくても，この関数を使うと `model` ブロックで直接線形モデルを書き込むことができますので，便利です。ここで線形モデルで表されるものを μ, θ, λ などさまざまなギリシア文字を使っていますが，このギリシア文字の違いにとくに意味はありません。一般的に正規分布の**位置母数**には μ が，ベルヌーイ分布のそれには θ が，ポアソン分布のそれには λ が使われることが多く，その慣例に習っただけです。いずれも確率分布の中心的な位置を表す母数であるということだけが重要です。また正規分布のところを見ると，リンク関数も逆リンク関数も同じ形をしていることがわかります。こうした**恒等式**で済んだので複雑なことを考える必要がなかったんですね。

表 26.1　確率分布とリンク関数の関係

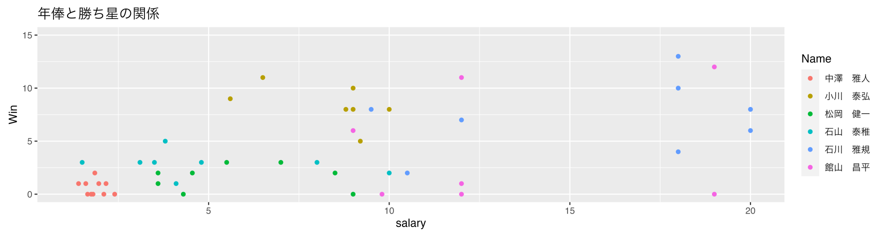
確率分布	逆リンク関数	リンク関数	Stan の専用関数
正規分布	$\mu = \beta_0 + \beta_1 X$	$\beta_0 + \beta_1 X = \mu$	`normal`
ベルヌーイ分布	$\theta = \dfrac{1}{1 + exp(-(\beta_0 + \beta_1 X))}$	$\beta_0 + \beta_1 X = \log\left(\dfrac{\theta}{1-\theta}\right)$	`bernoulli_logit`
ポアソン分布	$\lambda = \exp(\beta_0 + \beta_1 X)$	$\beta_0 + \beta_1 X = \log(\lambda)$	`poisson_log`

では一般化線形モデルに「個人差」を混合するモデルを考えてみましょう。ここでは前の章でも扱った野球のデータを例に説明します。図 26.1 には Swallows で 2011 年から 2020 年の間，8 年以上在籍した投手の年俸 (単位は千万円) と勝利数の関係を表しています。年俸が高いと勝ち星が増える，という関係にあると考えてモデルを組みます。

線形モデルですから，勝ち星を Y_i，年俸を X_i として $\hat{Y}_i = \beta_0 + \beta_1 X_i$ という関係を考えます。とはいえ，選手の年俸というのはそれまでの業績によるところも大きく，そもそも個人差があり得るところです。そこで個人差 μ_i を考えて，$\hat{Y}_i = \beta_0 + \beta_1 X_i + \mu_i$ とした線形モデルを考えましょう。また，勝利数は 0 以上の整数しか取りませんから，これはポアソン分布に従うと考えられます。そこでポアソン分布に合わせるために，逆リンク関数を使って $\lambda = \exp(Y_i) = \exp(\beta_0 + \beta_1 X_i + \mu_i)$ とした上で，$Y_i \sim poisson(\lambda_i)$ という確率モデルに入れることにしましょう。このモデルの設計図は図 26.2 のようになります。

[*2] 後学のためにきちんと定義式を書いておくと，変数 X が k の値を取る確率を $P(X = k) = \dfrac{\lambda^l e^{-\lambda}}{k!}$ とした分布です。ここで e はいつもの**ネイピア数**であり，$k!$ の ! は階乗を表す記号です。

[*3] この関数はよく出てきますから周知のことと思いますが，$exp(x) = e^x$ でこの e もいつものネイピア数です。この関数で x が負の数になっても，$x^{-1} = \dfrac{1}{x}$ という指数計算の決まりがありますから，結果が負になることはありません。



年俸の推移(単位;千万円)

	Name	2011年度	2012年度	2013年度	2014年度	2015年度	2016年度	2017年度	2018年度	2019年度	2020年度
1	石川　雅規	18.0	20.00	20.00	18.0	18.00	20.00	18.0	12.00	9.5	10.5
2	館山　昌平	12.0	19.00	19.00	NA	9.00	12.00	12.0	9.80	2.4	NA
3	松岡　健一	8.5	9.00	7.00	5.5	4.55	3.60	4.3	3.60	NA	NA
4	中澤　雅人	2.4	1.85	1.65	1.4	1.96	2.16	1.8	1.75	2.1	1.6
5	石山　泰稚	NA	NA	1.50	3.5	3.80	4.10	3.1	4.80	10.0	8.0
6	小川　泰弘	NA	NA	1.20	5.6	6.50	10.00	8.8	9.00	9.2	9.0

勝ち星の推移

	Name	2011年度	2012年度	2013年度	2014年度	2015年度	2016年度	2017年度	2018年度	2019年度	2020年度
1	石川　雅規	10	8	6	10	13	8	4	7	8	2
2	館山　昌平	11	12	0	NA	6	1	0	0	0	NA
3	松岡　健一	2	0	3	3	2	2	0	1	NA	NA
4	中澤　雅人	0	2	0	1	1	1	0	0	0	1
5	石山　泰稚	NA	NA	3	3	5	1	3	3	2	3
6	小川　泰弘	NA	NA	16	9	11	8	8	8	5	10

図 26.1　一流選手の年俸と勝利数

それではこれをもとに，GLMM をやってみましょう。実行するための Stan のコードはコード 26.1 のようになります。

code：26.1　個人差を含んだポアソン回帰モデル

```
data{
  int L;
  int N;
  array[L] real X;
  array[L] int Y;
  array[L] int index;
}

parameters{
  real beta0;
  real beta1;
  array[N] real mu;
}

model{
  // model
  for(l in 1:L){
    Y[l] ~ poisson_log(beta0 + (beta1 * X[l]) + mu[index[l]]);
  }
  // prior
  beta0 ~ normal(0,10);
  beta1 ~ normal(0,10);
  mu ~ normal(0,10);
}
```

■コード解説

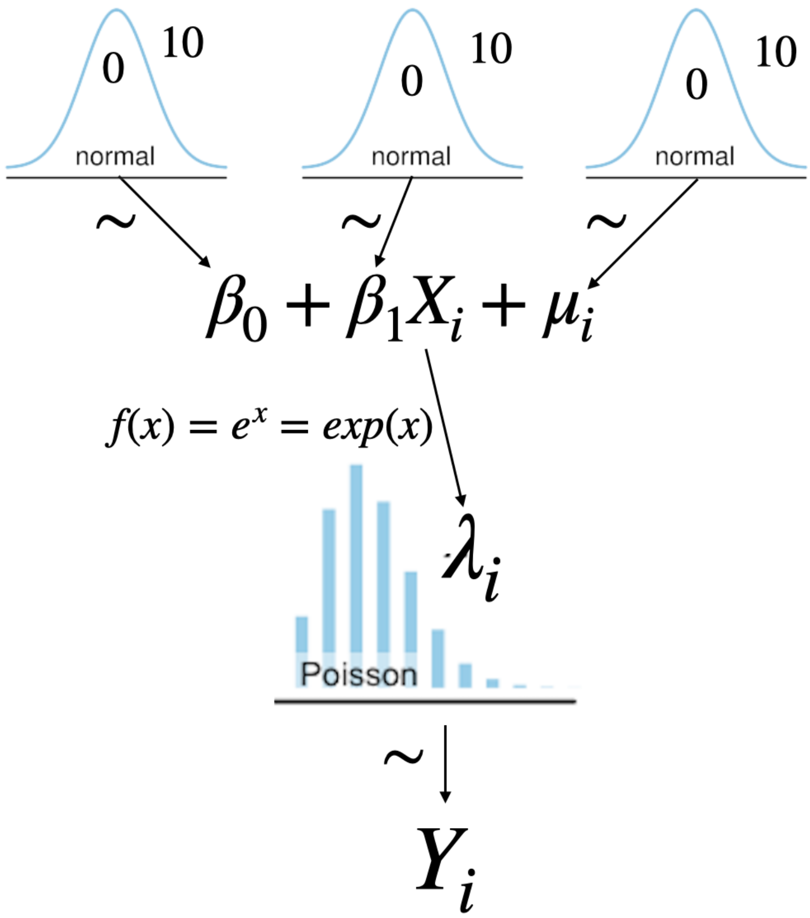


図 26.2 ポアソン分布を使った GLMM モデルの設計図

data ブロック データ長 L と扱う選手数 N, 独立変数 X, 従属変数 Y, 個人差インデックスを宣言しています。ポアソン分布に従う確率変数ですので従属変数は `int` 型にしてあります。

parameters ブロック 線形モデルの係数と個人差 μ_i を人数分用意しました。

model ブロック モデル尤度のところは `poisson_log` を使っているので, 線形のコードをそのまま書き込むことができています。事前分布は正規分布にしました。

これを R で呼び出すコードは Code::26.2 の通りです。少しテクニカルなコードになっていますので自分で書けなくても構いませんが[*4], 各ステップで何をしているか読めるようになっておくといいでしょう[*5]。

code : 26.2 ポアソン回帰のコード

```
baseball <- read_csv("baseballDecade.csv")

dat <- baseball %>%
  filter(position == "投手") %>%
  filter(team == "Swallows") %>%
```

[*4] コードの書き方は単一の正解というものがありません。ここに書いてあるのは著者流のやり方だというだけであって, 同じ結果が出るのであれば別のコードの書き方であっても構いません。今回のこのコードはシラバスのサイトで配布しています。

[*5] パイプ演算子 `%>%` を使って変換プロセスをつなげて言ってますが, その途中でデータがどのように加工されているか知りたい場合, `%>% print %>%` と `print` 関数を挟むといいでしょう。ステップごとのデータの変形プロセスがよくわかると思います。

```
6     group_by(Name) %>%
7     nest() %>%
8     mutate(
9       n = purrr::map_dbl(data, ~ NROW(.)),
10      FLG = purrr::map_lgl(data, ~ anyNA(.$Win))
11    ) %>%
12    filter(n > 7) %>%
13    filter(!FLG) %>%
14    unnest(data) %>%
15    select(Year, Name, salary, Win)
16
17    dat.tmp <- dat %>%
18    mutate(salary = salary / 1000) %>%
19    mutate(ID = as.factor(Name)) %>%
20    mutate(ID = as.numeric(ID))
21
22  dataSet <- list(
23    L = NROW(dat.tmp),
24    X = dat.tmp$salary,
25    Y = dat.tmp$Win,
26    index = dat.tmp$ID
27  )
28
29  model_pois <- cmdstanr::cmdstan_model("glm_poisson.stan")
30  fit <- model_pois$sample(
31    data = dataSet,
32    chains = 4,
33    parallel_chains = 4,
34    iter_warmup = 1000,
35    iter_sampling = 3000
36  )
```

■コード解説

1 行目 ファイルからデータを読み込みます。このデータファイル `baseballDecade.csv` には野手，投手の 10 年分のデータが入っています。

3-15 行目 最初のブロックとして，データの加工・変改を行い，それを `dat` オブジェクトに格納します。

- **4 行目** フィルターをかけて，投手だけのデータに絞り込みます。
- **5 行目** フィルターをかけて，Swallows の選手に限ります
- **6 行目** 選手名でグルーピングします。
- **7 行目** `nest` 関数でグループ化変数に沿ってデータを畳み込みます。引数をとくに指定しなければ，変数はすべて `data` という変数名に含まれます。この段階でデータフレームは出力 26.1 のようになっています。`nest` 関数のイメージを図 26.3 に示しました。
- **8 行目** 畳み込まれたデータセットを使って，そのデータの行数 (何年分のデータがあるか[*6])，勝利

[*6] ここで `data` とあるのはまとめられたミニ・データフレーム全体を引数にしています。それに対して `NROW` という関数をあてがうことで行数を数えています。`NROW` 関数の引数は `.` ですが，これは `map` 関数の第一引数全体を再度参照するときの略記号です。`purrr::map` というのは `purrr` パッケージの `map` 関数で，同じ関数操作を繰り返し行うときに使います。

数情報の欠損値の有無を作る変数[*7]を作っています。

9 行目 フィルターをかけて，データの行数 n が 7 より大きいもの，つまり 8 年以上登板している一流の選手だけに絞ったのです。

10 行目 フィルターをかけて，欠損値がないデータだけに絞っています。

11 行目 畳み込みを解除します。

12 行目 年度，選手名，年俸，勝利数だけ変数をセレクトして取り出します。

17-20 行目 このブロックでは Stan で使うためにさらに一時的なデータ加工をし，`dat.tmp` オブジェクトに格納しています。

18 行目 年俸のデータは数字が大きいので，1000 万円単位にします。分析の際，数字が大きすぎると計算がオーバーフローしてしまうことがありますから，適当な単位にしておくことは実践上の有効なテクニックの 1 つです。

19 行目 名前の変数は文字型ですが，そのまま Stan に渡すことはできないので，Factor 型にした `ID` という変数に作り替えます。

20 行目 先ほど作った `ID` は Factor 型ですが，Stan では質的な違いを表す数字だけあれば良いので，数字型に変形します。

22-27 行目 Stan に与えるためのデータセットにするブロックです。

29-36 行目 cmdstanr でサンプリングするためのコードです。rstan の場合は `sampling` 関数を使うことになります。

R の出力 26.1: データをグループ化変数でネストする

```
# A tibble: 295 × 2
# Groups:   Name [295]
   Name       data
   <chr>      <list>
 1 永川　勝浩 <tibble [3 × 16]>
 2 前田　健太 <tibble [5 × 16]>
 3 シュルツ   <tibble [1 × 16]>
 4 大竹　寛   <tibble [8 × 16]>
 5 横山　竜士 <tibble [2 × 16]>
 6 ジオ       <tibble [2 × 16]>
 7 バリントン <tibble [5 × 16]>
 8 藤川　球児 <tibble [7 × 16]>
 9 久保　康友 <tibble [7 × 16]>
10 小林宏     <tibble [1 × 16]>
# … with 285 more rows
```

さて，このコードを実行し，結果からモデルを図にしたのが図 26.4 です。図中の赤い転線は，$+\mu_i$ をおかずに全体を混ぜ合わせたときのポアソン回帰です。個人差を表す項目を追加することで，より個々のデータに沿った回帰プロットが実践できていることがわかると思います。

今回は個人差 μ_i だけを考えましたが，たとえば野球の弾の規定が変わってピッチャーが有利になる年とそうでない年があったとか，登板する球場のサイズによって有利不利がある，といったさまざまな個別事情に対し，それを表す変数の項を付加することで，よりデータの詳細を細かくモデルに組み込んでいくことができま

[*7] `purrr::map` の使い方は先ほどと同じで，`.$Win` は `data$Win` と同じ意味です。`anyNA` は NA があるかどうかを `TRUE/FALSE` で返す関数です。

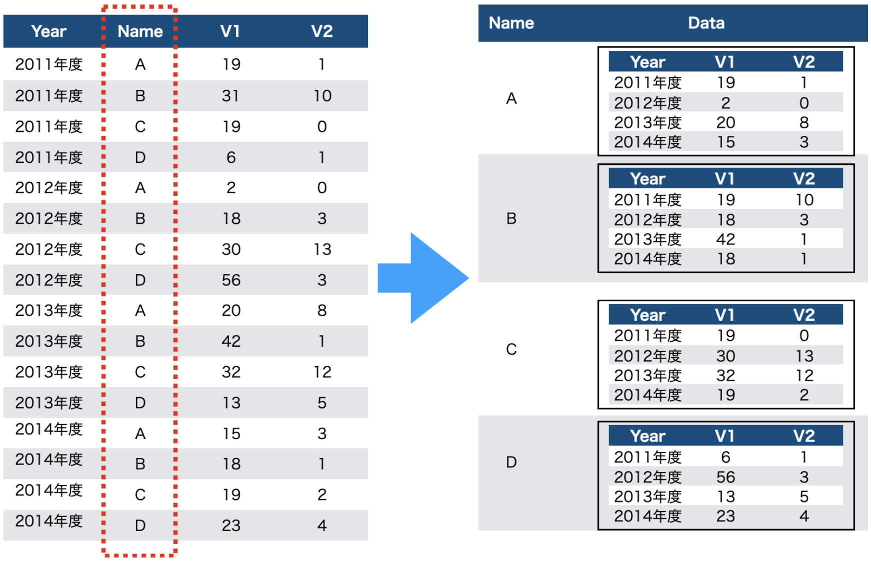


図 26.3　データの畳み込み関数の挙動。左側がもとのデータで，グループ化変数を使って畳み込むと，データフレームの中にグループごとのミニ・データフレームが畳み込まれる。

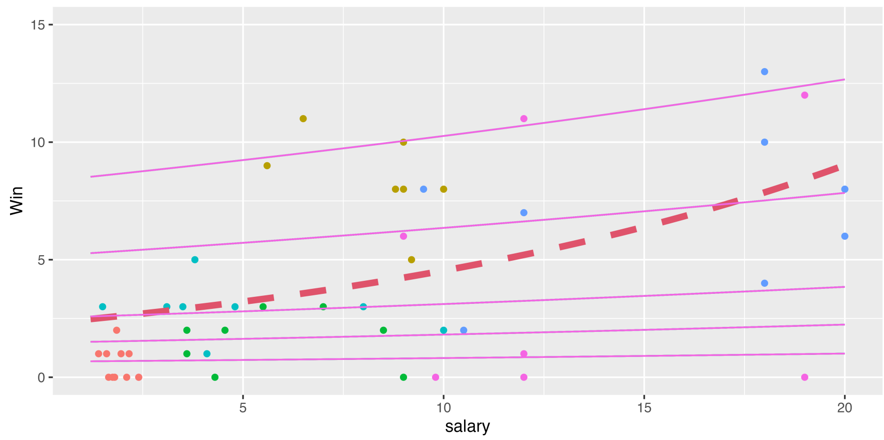


図 26.4　ポアソン分布を使った GLMM の結果

す。線形モデルは**線形である**という制限から逃げることはできませんが，確率分布や個別の事情を組み込むことで，より上手く個々のデータの特徴を表現できるようになることが，イメージできたのではないでしょうか。

26.2　ネストされたデータ

さて先ほど，`nest` 関数を使ってデータをネストするという例を見ました。

ネストとは「入れ子にする」という意味で，マトリョーシカのようにデータフレームの中のデータフレーム，データフレームの中のデータフレームの中のデータフレーム，というように，同じ形のものを入れていくイメージです。このネストされたデータは，別名**階層性**を持ったデータだ，ということができます。

たとえば今回の野球データの場合，選手はどこかのチームの一員ですから，チームという上位階層の中に選手がいることになります。野球チームは 12 球団ありますが，それぞれセ・リーグ，パ・リーグに分かれて戦い

ます[*8]。つまりリーグという上位階層があるわけです。

逆に，チーム傘下の選手も，年間 140 試合ぐらいありますから，試合ごとに調子が良かったり悪かったりするでしょう。試合の中でも，野球は 9 回のターン制バトルですから，毎回の活躍というのがあるかもしれません。すなわち，ある選手の中に試合や回がネストされていると考えることもできます。私たちはデータのある側面で区切ってみていることになります。

こうしたデータの階層性は，何も野球選手のデータだけではありません。たとえば調査研究を行う場合，ある小学校でデータをとったというときに，児童は学年でネストされ，学校でネストされ，地区でネストされ，市区町村でネストされ，都道府県でネストされている，ということになります。もちろん 1 つの小学校だけでデータを取るのであれば比較検証はできませんが，ある程度規模が大きくなってくると何らかの階層的な情報を持っていると考えるべきでしょう。

またあるいは，心理学の実験のような少人数からしかデータを得ないという場合であっても，条件 A で n 回試行，条件 B で m 回試行する，といった Within デザインの場合は**反復測定 (repeated measure)** であり，これも被験者ごとに各データがネストされている，と考えることができるわけです。Within データの場合は個人差を考え，GLMM で個人差以外の要因もモデルに組み込むことができるようになったわけですから，データの階層性について考えないわけにはいきません。

何がそんなに問題なのでしょうか。図 26.5 をみてください。左右にあるのは同じデータの分布ですが，左側は階層の区別をしない場合で，右側はその区別をした場合です。線形回帰の直線を引いておきましたが，これを見ると明らかなように階層性を考えなければ，私たちは正の相関があるデータだと判断してしまうでしょう。しかし右側，群ごとに区別してモデルを書いてみると，回帰直線はすべて右下がり，つまり負の相関があるデータだったことがわかります。階層性や群ごとの特徴を無視して分析してしまうと，結果が真逆になってしまう可能性を孕んでいるわけです。

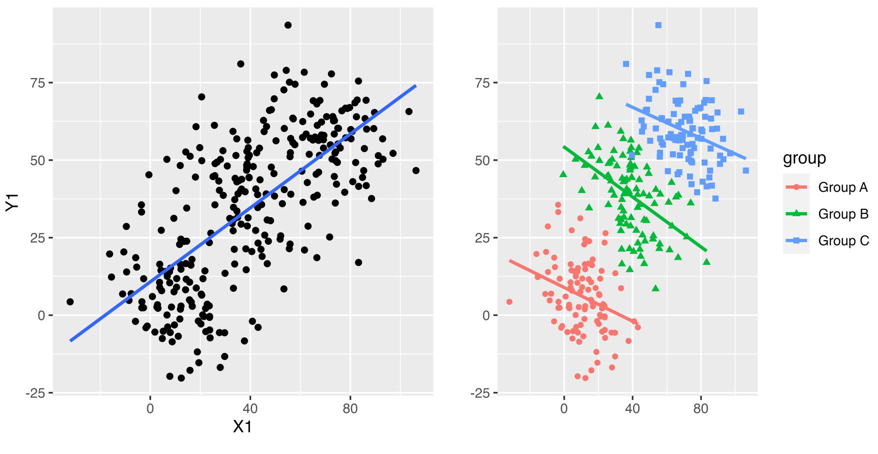


図 26.5　階層データの分析が必要な理由

データは図にすることの重要性は何度指摘してもしすぎることはないと思いますが，ことほど左様にデータの持っている性質はなるべく丁寧に見定めなければなりません。そして階層に区分して分析する必要がある

[*8] セントラル・リーグ Central League とパシフィック・リーグ Pacific League の略であり，セ・リーグには巨人 (Giants)，阪神 (Tigers)，広島東洋カープ (Carp)，中日 (Dragons)，ヤクルト (Swallows)，DeNA(Baystars) が含まれます。パ・リーグにはソフトバンク (Hawks)，西武 (Lions)，日本ハム (Fighters)，ロッテ (Marines)，オリックス (Buffaloes)，楽天 (Eagles) からなります。2021 年現在のチーム名であり，かつては大洋ホエールズとか，阪急ブレーブスとか，南海ホークスなどのチームがありました。

のであれば，モデルもそのようにレベルアップさせなければなりません。そこで出てくるのが**階層線形モデル (Hierarichal Linear Model)** です。

26.3 階層線形モデル

階層線形モデルは，ネストされたデータに対して群ごとの特徴を記述できるよう，線形モデルを拡張したものです。

野球の例で考えてみましょう。階層化せずにデータを眺めて，「グラウンドにはゼニが落ちてるんだ」とばかりに，たくさん年俸をもらっているバッターはホームランを打つし，ピッチャーは勝利数を稼ぐ，という傾向を分析したいとします。年俸が説明変数で，成績が従属変数になるモデルですね[*9]。しかし，ソフトバンクや読売巨人のように，資本力のある親会社のもとでの球団であればそういう傾向もあるかもしれませんが，広島カープのような市民球団[*10]ではそこまで大盤振る舞いできるほどではないよ，といった球団ごとの差異があるかもしれません。

年俸が成績に影響する，というのが全体的な傾向としてあり，影響の程度は群ごとの特徴が多少入り込む，ということこの状況をモデルで表現することを考えましょう。変数の影響力の大きさは，β_0, β_1 などの回帰係数で表現されます。全体的な傾向はこれで表現できるとして，個別の効果はこの係数が群 j ごとに違っている，ということになります。この群ごとの違いを，群平均からの差異で表現しましょう。

球団 j に属するプレイヤー i の成績を Y_{ij} と表現したとします。データ全体としては，$\hat{Y_{ij}} = \beta_{0j} + \beta_{1j}X_{ij}$ なのですが，ここでたとえば切片 β_{0j} に群 j ごとの効果があるとすれば，$\beta_{0j} = \gamma_{00} + u_{0j}$ のように，切片の平均 γ_{00} と，切片に加わる群の効果 u_{0j} という，**係数を説明する線形モデル**を想定することと同じです。切片だけでなく傾き β_{1j} の方も，$\beta_1 = \gamma_{10} + u_{1j}$ と線形モデルで考えましょう。個人差 $u_{\cdot j}$ は平均 0 の正規分布に従うと考えます。ここで全体的な傾向，すなわち個人差の平均である γ_{00}, γ_{10} のことを**固定効果 (Fixed Effect)** といい，個別の効果 u_{0j}, u_{1j} のことを**変量効果 (Random Effect)** と呼ぶことがあります。

式を整理するとこうなります。

$$
\begin{array}{lcl}
\hat{Y_{ij}} & = & \beta_{0j} + \beta_{1j}X_{ij} \\
\beta_{0j} & = & \gamma_{00} + u_{0j} \\
\beta_{1j} & = & \gamma_{10} + u_{1j} \\
\\
\begin{pmatrix} u_{0j} \\ u_{1j} \end{pmatrix} & \sim & MN\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \tau_{00} & \tau_{01} \\ \tau_{10} & \tau_{11} \end{pmatrix}\right)
\end{array}
\tag{26.1}
$$

あらまあ。何だか訳がわからなくなりそうですね。いやもちろん，じっくり考えればお分かりいただけると信じていますが，直感的にわかりにくいかもしれません。そこで同じことを違う角度から説明してみたいと思います。

階層性のあるデータとはいえ，たかだか 12 球団ですから，12 回ぐらいの繰り返し分析は頑張ればできますよね。実際それをやったのが図 26.6 にあります。えっへん。この結果を解釈するときは，球団ごとにバラバラの分析をしていることと同じですから，12 球団ごとの切片 β_{0j} と傾き β_{1j} が出てきます。ちょっと面倒ですが，データがそういうものなんだから仕方ありません。

[*9] ちょっと待って，逆じゃないか，と思った人もいるかもしれません。つまり成績がいいから年俸が高くなるのでは，ということですよね。その場合は，ある年 X の年俸が翌年 $X+1$ 年の成績を予測する，という形にデータを変えなければなりません。今回は年度ごとの年俸と成績からなるデータセットになっていますから，年俸が成績に影響する，という仮説になっています。

[*10] 広島カープは戦後復興の名目で公的資金が投入されて作られた球団であり，その意味で市民の資本から作られた球団と言えるでしょう。現在も，特定の親会社はなく複数の企業が連携して運営するスタイルをとっています。

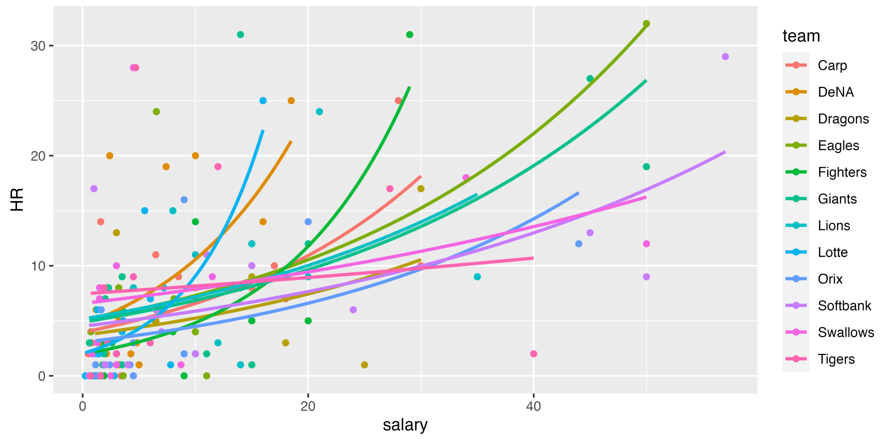


図 26.6　チームごとにバラバラの回帰分析を実施する。できなくはないけど，要するにどういうこと，と言いたくもなる。

でもこれが，12 球団ではなく，100 球団，1000 球団あったらどうしますか。1000 個の切片と傾きをずらーっと数表に並べられても，目が痛くなりますね。辛いだけです。せっかく大量のデータを手に入れているのに，分析結果を細かくみてくれなかったら悲しい。かといって「で，結局どういうことなの」と言われた時に，「えー，第 1 球団の切片と傾きは･･･で，第 2 球団の切片と傾きは･･･で，第 998 球団の切片と傾きは･･･」と読み上げることもないでしょう。ではどうするか。傾きと切片の平均と SD で報告するんではないでしょうか。傾きの平均はこれぐらいです，球団による違いは ± これぐらいでます，といえば，1000 球団の情報を要約して言えることになります。それが $u_{0j} \sim N(0, \tau_0)$ や $u_{1j} \sim N(0, \tau_1)$ が言っていることなんですね。散らばりの平均は γ_{00}, γ_{10} であり，それが τ_0, τ_1 ぐらい散らばりますよ，ということです。

これは見方を変えると，階層線形モデルを使う場合は，階層性をまとめる必要があるかどうかを考えてからでも構わないということです。12 球団しかなくて，それぐらいならじっくりみましょうというのであれば，無理に階層モデルにする必要はありません。また 100 球団，1000 球団ある世界であっても，ほとんど球団ごとの違いがないようであれば，全部まとめて分析してしまっても構いません。階層線形モデルで表現する場合は，ネストされているデータに一定の傾向はあるものの，考慮する必要がある程度に散らばりがある場合，ということになります[*11]。

またさらに別の角度から考えてみましょう。今度はこの線形モデルの，設計図を書いてみるのです。図 26.7 にそれを書いてみました。

図 26.7 では，話を簡単にするために係数の共分散を想定しない ($\tau_{10} = \tau_{01} = 0$)，1 次元正規分布から係数が来ているモデルにしています。これをみると，

1. データはポアソン分布からきている；$Y_{ij} = poisson(\lambda_{ij})$
2. ポアソン分布は線形モデルから逆リンク関数で接続されてきている；$\lambda_{ij} = exp(\beta_{0j} + \beta_{1j} X_{ij})$
3. 線形モデルの切片が別の分布からきている；$\beta_{0j} \sim N(\gamma_{00}, \tau_{00})$
4. 線形モデルの傾きも別の分布からきている；$\beta_{1j} \sim N(\gamma_{10}, \tau_{11})$
5. γ_{00} がどういう数字かわからないので，確率分布 (事前分布) でそのわからなさを表現；$\gamma_{00} \sim N(0, 10)$

[*11] これも分散分析的発想で，群内の類似性と群間の類似性の比率から，十分に大きな群間変動があるようなら階層モデルにするべき，と判断する基準があります。この比率のことを**級内相関 (Intra-Class Coefficients)** と言います。詳しくは清水･荘島 (2017) を参考にしてください。

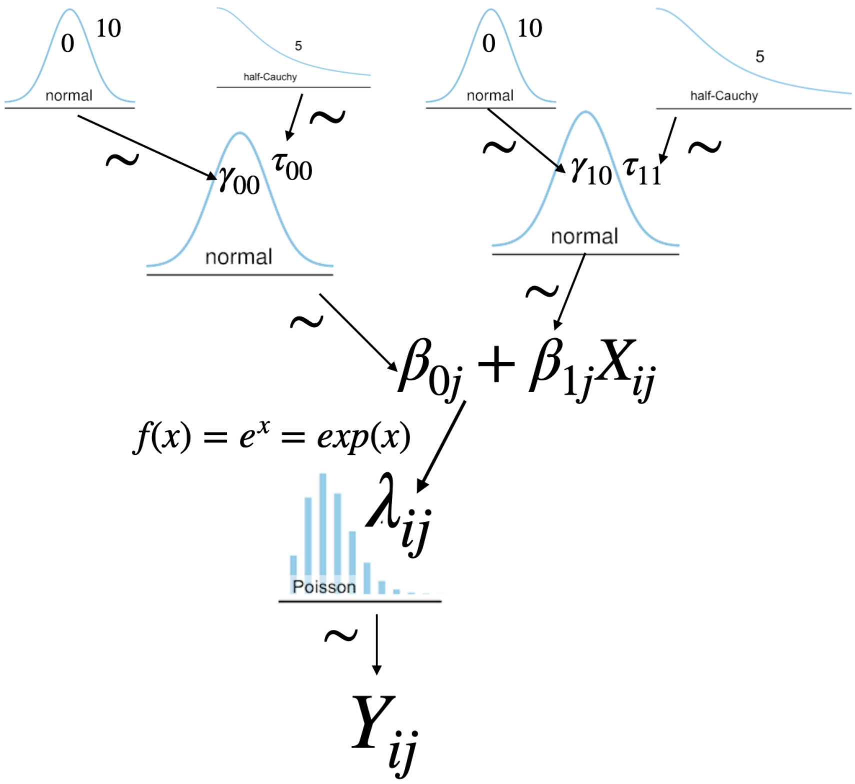


図 26.7　チームごとにバラバラの回帰分析を実施する

6. γ_{10} がどういう数字かわからないので，確率分布でそのわからなさを表現；$\gamma_{10} \sim N(0, 10)$
7. τ_{00} がどういう数字かわからないので，確率分布（事前分布）でそのわからなさを表現。SD なので cauchy 分布にしてみた；$\tau_{00} \sim cauchy(0, 5)$
8. τ_{11} がどういう数字かわからないので，確率分布（事前分布）でそのわからなさを表現。SD なので cauchy 分布にしてみた；$\tau_{11} \sim cauchy(0, 5)$

という順番で読み解くことができます。データに分布を仮定する，これがベイズ推定のスタートでしたが，仮定された分布のパラメータに，線形モデルという数学的**構造**が入っています。このパラメータに数式を入れることを**モデリング (modeling)** というわけです。そしてそのモデルに含まれている未知数（パラメータ）に，さらに別の分布が入っています。分布に分布が入っているから，混合 mix しているわけです。しかもこの混合加減は，上に上にとレベルが上がっていきますので，階層的なモデリングになっています。パラメータのパラメータは，**ハイパーパラメータ (Hyper parameter)** と呼んだりします。かっこいい。

さて，これをみればわかるように，階層線形モデルは，12 の球団にある違いに正規分布というカバーをかけ，平均とそこからの散らばりという形で表現することでもあります。このように，係数たちに正規分布というカバーを上からかけると，12 球団バラバラで推定した時よりも，その係数が少し小さくなることがあります。上位階層の正規分布の平均値に，少し引っ張られて推定されてしまうわけです。これを係数の**縮小 (shrinkage)** と言います。そうした問題はありますが，事前分布や階層性のおかげで，下位レベルの推定値が安定して算出できることにもなります。

さあ設計図が書けたら，もうコードに落とすことはできますね。ポアソン分布を使った階層線形モデルのコード例を Code::26.3 に挙げておきます。

code : 26.3 階層ポアソン回帰モデル

```
data{
  int L;
  int G;
  array[L] int Gindex;
  array[L] real X;
  array[L] int Y;
}

parameters{
  array[G] real beta0;
  array[G] real beta1;
  real gamma0;
  real gamma1;
  real<lower=0> tau0;
  real<lower=0> tau1;
}

transformed parameters{
  array[L] real<lower=0> lambda;
  for(l in 1:L){
    lambda[l] = exp(beta0[Gindex[l]] + (beta1[Gindex[l]] * X[l]));
  }
}

model{
  // model
  for(l in 1:L){
    Y[l] ~ poisson(lambda[l]);
  }

  for(g in 1:G){
    beta0[g] ~ normal(gamma0,tau0);
    beta1[g] ~ normal(gamma1,tau1);
  }

  //prior
  gamma0 ~ normal(0,10);
  gamma1 ~ normal(0,10);
  tau0 ~ cauchy(0,5);
  tau1 ~ cauchy(0,5);
}
```

■コード解説

data ブロック データは整然データの形で渡します。データの長さ L, 説明変数 X, 被説明変数 Y の他に，群の数 G, どの群に属するのかを表すインデックス変数 G[L] を用意しています。

parameters ブロック 群の数だけ係数 β_0, β_1 が必要です。加えて，これら切片と傾きの平均値 γ と散らばり τ を用意しています。

transformed parameters ブロック `poisson_log` を使ってもいいのですが，話をわかりやすくするために

いったん λ_{ij} を作っています。係数 `beta0[g]` のグループを表す g のところは，L 行目のデータの所属先を表すインデックス，`Gindex[l]` で代入しています。`beta1` についても同様です。

model ブロック　データの各行は，行ごとの線形モデルで推定されますのでスッキリしたものです。`beta0[g]` や `beta1[g]` はハイパーパラメータを持つ上位の正規分布にカバーされています。パラメータの事前分布は設計図通りです。

これを使って，年俸でホームランの数を予測する階層線形モデルを推定させましょう。結果を出力 9 に示してあります。

MCMC の結果 9: ハイパーパラメータの推定値

```
  # A tibble: 4 × 7
  name          EAP       MED       MAP        SD       L95       U95
  <chr>   <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 gamma0      1.446     1.448     1.449     0.120     1.200     1.680
2 gamma1      0.046     0.046     0.044     0.010     0.026     0.068
3 tau0        0.367     0.348     0.324     0.114     0.201     0.642
4 tau1        0.033     0.031     0.028     0.010     0.018     0.057
```

これをみると，12 球団全体で考えると，平均的に年俸が 1000 万上がると $exp(0.046) = 1.047074$，つまり 1 本ホームランが増えるわけですね。ただ年俸の効果は球団ごとの違いがありますから，±1SD の範囲だと係数は $0.046 + 0.033 = 0.0793$ から $0.0461 - 0.033 = 0.0127$ までぶれることがあるようです[*12]。それよりは切片，平均 4000 万ではありますが，2900 万から 6000 万までの差がありますので，やはりどの球団に属するかの方が効果が大きいかなあ，と思ったりもするわけです。

最後にこの結果をプロットしてみました。球団ごとの 50% 確信区間も合わせてプロットしてあります。これをみると傾きや切片の球団ごとの違いがイメージしやすくなるかと思います。

今日は分布を混ぜるというところから，線形モデルを階層的に組み上げていくことを説明してみました。数式的には難しいように感じるかもしれませんが，設計図を書いてコードにできれば，分析イメージは掴めるのではないでしょうか。データがどういうところから出てきて，どういう潜在的な影響があるのか，個人差や群の差はどの程度あるのか，といった状況に応じて，モデルを柔軟にカスタマイズしていくことができます。よりデータに適切なモデルへ，より柔軟で複雑な表現を目指すことは，とてもクリエイティブで楽しいことではありませんか！

26.4　課題

以下のモデルを分析する R/Stan コードを提出してください。結果の解釈などを，スクリプトのコメントアウトや別添ファイルなどで提供してもらえるとなお良いです。もちろん Rmd ファイルでの提出であれば完璧です。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

■個人差を入れたモデルを考えてみよう　野球のデータで，野手の長打率を年俸で予測するモデルに個人差を入れたモデルを考えてみましょう。データはソフトバンクの，年俸 2500 万円以上の人を対象にします。今回，長打率は安打数 `Hit` のうち，ホームランの本数 `HR` から考える割合とします。割合情報の分析ですの

[*12] 傾き β_1 の平均が γ_1，標準偏差が τ_1 であり，それぞれの **EAP 推定値**を見ています。γ_1 の SD，0.0120 は推定値の振れ幅，すなわち標準誤差ですのでご注意ください。

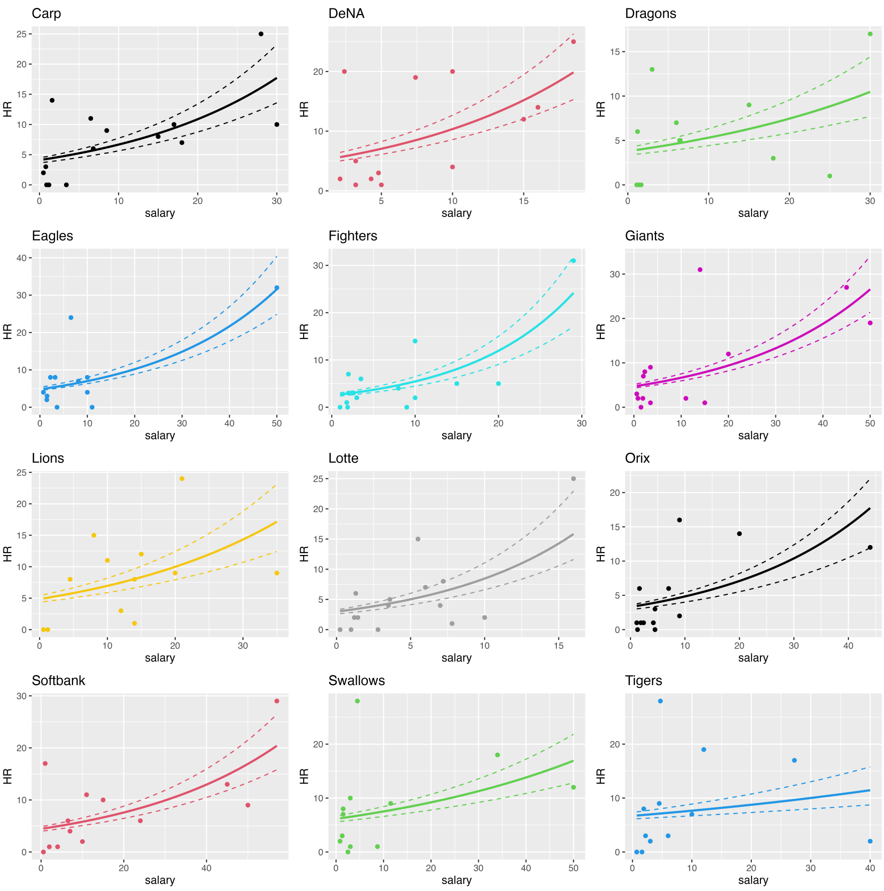


図 26.8　階層線形モデル；チームごとのポアソン回帰分析

で，二項分布で表現できます。二項分布の確率 θ の線形モデルに個人差を入れ，分析してみましょう。コード 26.1 が参考になるはずです。

■階層線形モデルのコードを書いてみよう　野球のデータで，年俸でピッチャーの勝利数を予測する階層線形モデルを考えてみましょう。12 球団の平均や，球団ごとの散らばりはどれぐらいになるでしょうか。ピッチャーの勝利数はポアソン分布に従うと考えられますので，コード 26.3 が参考になるでしょう。データ加工のコードはシラバスのサイトで提供しますが，ご自身でコードを書いても一向に構いません。

第 27 章

混合分布モデル

さて LM→GLM→GLMM→HLM と，線形モデルを展開させてきました。後半の GLMM や HLM では分布を混ぜるということをしてきたわけですが，今回も分布を混ぜるモデルについて見ていきたいとおもいます。

今回の分布の混ぜ方は，パラメータの構造が複雑化するのではなく，そもそも違うパラメータの分布が混ざり合っているというパターンです。具体的な例から見ていきましょう。

27.1　混合分布モデル

図 27.1 を見てください。これは野球選手データから Tigers の年俸をヒストグラムにしたものです。年俸のデータはそのままだと左に歪んでいるので，対数をとったものを表示しています。ヒストグラムを見ると，どう

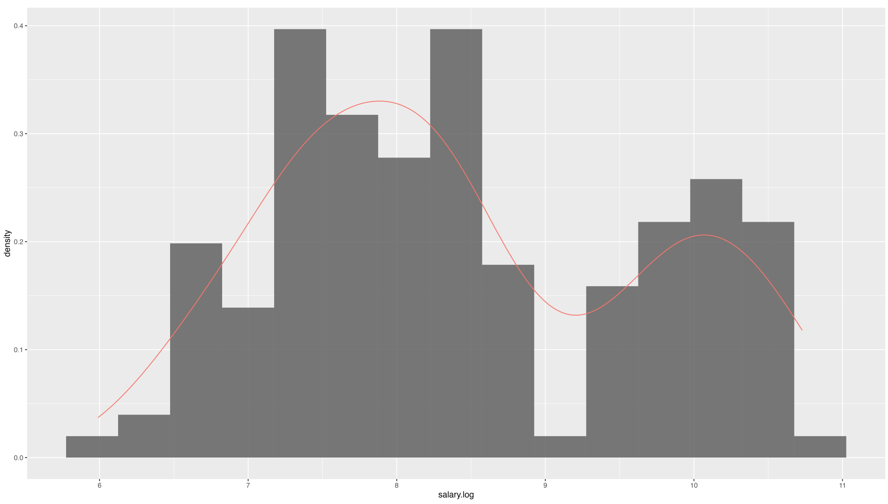


図 27.1　Tigers の選手年俸 (対数) の分布

も 2 つの山があるように見えますね。赤いラインは密度のカーブを書いたものですが，これはどう見ても 2 つのピークを持っています。このようなデータに対して，正規分布のモデルを当てはめるのはおかしなことです。**正規分布**はご存知の通りベルカーブ，すなわちピークは 1 つ (単峰) で，両裾になだらかに下がっていくものですから，このデータと違う形になっていることは間違いありません。

赤い密度カーブの方が暗示するように，この分布は 2 つの正規分布が混じり合って出てきているのではないか，と思われます。つまり，出どころや種類の違う分布があるわけです。これまで分析してきたデータは，基

本的に同じ分布からデータが出てきたと考えられてきたわけですが，そうでもない分布の場合はどうするか，という問題です。

こうしたモデルを考えるのが**混合分布モデル**と呼ばれるものです。ここでは複数の正規分布を混ぜ合わせた**正規混合モデル (normal mixture model)** を扱います。

データが複数の出自を持つという時に，その違いを示す変数が明確なのであれば困ることはありません。たとえば男性と女性で分布が違うとか，東日本と西日本で分布が違うという既有知識があって，それがデータに含まれていれば，その変数でデータセットを分割して階層モデルにすれば良いだけです。しかし今回のように，阪神の選手という意味で同じ種類のメンバーであるはずなのに，データの表れ方が違う場合，「この 2 つの分布に分かれる潜在的な変数は何か」ということを探す必要があります。言い換えれば，データの表面的な特徴からグルーピングをしなければなりません。これはデータの類似性だけを用いて，外的基準を持たずして分類する，**クラスタリング (Clustering)** という分析方法と関係します。そこで少し寄り道になりますが，クラスタリングの手法について見ておきましょう。

27.1.1 クラスタリングいろいろ

クラスタリングという分析方法は，先に述べたように別の変数 (外的基準) を使わずに[*1]，データの分類を行う方法です。分類は科学の基本で，まず同じものか違うものか，何が同じで何を違うものと考えるか，ということがすべてのスタートになります。クラスタリングではこれをデータから行う必要があり，データの類似性をもとに分類することが基本です。

データの**類似度**あるいは同じことですが**非類似度**を表す指標の基本は**距離 (distance)** です。距離と聞くと普通は 2 点間を直線で結んだ**ユークリッド距離 (Eucredian distance)** を想像されると思いますが，実は距離には他にもいろいろな種類があります (詳しくは第 15 講，セクション 15.2，Pp.153 を参照)。たとえば**相関係数**も絶対値を取ればデータの類似度を表す距離情報の一種と考えることができます。この距離を使って，距離が短い＝類似している＝同じクラスター，という考え方から分類していくことになります。

こうした分類，クラスタリングの手法はいくつかあって，大きく分けると**階層的クラスタリング (hierarchal clustering)** と**非階層的クラスタリング**の二種類に分かれます。階層的というのは HLM の発想と同じで，クラスターが積み重なっていくようなイメージです。たとえば要素 $\{a, b, c, d, e\}$ をクラスタリングする時，距離が近いものを使ってまず $\{a\}, \{b, c\}, \{d, e\}$ と分割したとします。次に $\{b, c\}$ を 1 つの新しいデータ点だと考えて，a と $\{b, c\}$，$\{b, c\}$ と $\{d, e\}$ の距離を考え，近い方をさらに次のクラスターにまとめます。すなわち $\{a, \{b, c\}\}, \{d, e\}$ というようにです。それをさらに上位階層でまとめ $\{a, \{b, c\}, \{d, e\}\}$ とすべてが 1 つのクラスターにまとめられたら終わり，とより包括的なクラスターへ，より上位のクラスターへとまとめていくのが階層的クラスタリングという手法です。この方法で分類した例を図 27.2 に示します。階層的クラスタリングは，階層が上がる時にクラスター化されたもの同士の距離をどう考えるかによって，いくつかの方法があります。先ほどの例ではまず $\{b, c\}$ が 1 つのクラスターになりましたが，これを A とすると，a と A の距離をどう考えるか，がいろいろあるわけです。大文字の A の方は，2 つの要素から合成されたもの ($A = \{b, c\}$) ですから，a と b の距離を取るのか，a と c の距離を取るのか，はたまた b, c の平均的な値と a との距離を取るのか等々，基準が必要なわけです。複数の要素を持つクラスター同士の距離の考え方として，その最大値を取る**最大距離法**，最小値を取る**最小距離法**だけでなく，データの重心を取る**セントロイド法**，平均を取る**平均法**，クラスター内分散とクラスター間分散の差が最小となるように取る**ウォード法 (Ward's method)** など

[*1] **機械学習**の文脈では，こうした基準となる変数を置かずに分析する手法のことを**教師なし学習**，と呼びます。

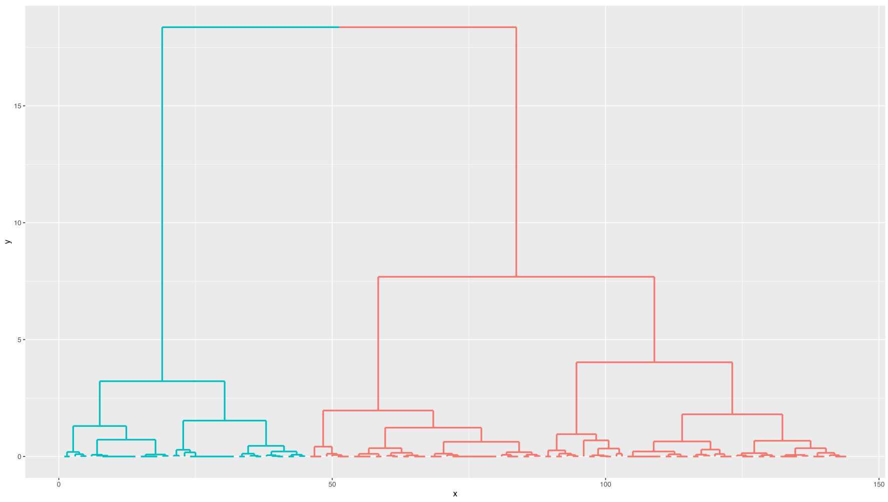


図 27.2　Tigers の選手を年俸で階層的クラスタリング (ウォード法による)

があります[*2*3]。最後のウォード法は弁別後の結果がわかりやすいため，よく用いられます。

一方，階層的でないクラスター分析で有名なのが **K-平均法 (k-means 法)** です。これは変数間の距離に k 個の基準点をランダムに置き，その点に近いものをクラスターとします。次に作られたクラスターの重心を新しい基準点として，再度分類します。それの重心を次の基準点とアップデートして再分類 $\cdots$ と繰り返して，分類結果が変わらなくなる前で何度も反復するのです。だいたい数回の反復で分類できること，データのサイズが大きくても効率よく部類できることからよく利用される方法の 1 つです。最初に幾つのクラスターに分類するか，明確な基準がないことが欠点でしたが，その後のモデルの展開でクラス数の適合度などを算出して適切なクラス数を求めるようになりました。

クラスター分析のもう 1 つの分類基準が，クラスターの境界強度によるものです。**ハードクラスタリング (Hard clustering)** では，ある個体がどのクラスターに含まれるかがはっきりしています。この個体はクラスター A，この個体はクラスター B，と決められたらそれで決まりです。これに対して**ソフトクラスタリング (Soft clustering)** は，ある個体がクラスター A に含まれる確率が XX%，クラスター B に含まれる確率が YY%，のように分類が確率的で固定的ではない方法です。

このように，データの特徴から分類をしていくクラスター分析はいろいろな手法があるのですが，今回扱う混合分布モデルはモデルに基づいたクラスタリング，Model based clustering とも呼ばれる手法です。すなわち，各個体が潜在的な正規分布のどちらに含まれる可能性があるか，その確率を考えながらソフトに分類していく非階層的な方法になります。

*2 厳密に書くと，クラスター P に含まれる要素と重心との距離の二乗和を $E(P)$ とすると，二つのクラスター P,Q を併合して作られる新しいクラスター T を考えた時，$E(T) - E(P) - E(Q)$ を P,Q の距離と考える方法です。

*3 R では標準関数 `hclust` で階層的クラスター分析ができ，ウォード法を使うときは `method='ward.D2'` とします。この `.D2` にはいわれがあって，かつては `ward.D` でウォード法の指定ができたのですが，この関数にバグが含まれていることが判明，修正版を `ward.D2` としたのです。R は無償のフリーソフトウェアですから，バグが含まれているなんてやっぱり使い物にならない！という批判もあるのですが，オープンなソフトウェアであるからこそソースコードが誰にでも見られて指摘することができ，よくなっていくものでもあります。バグがあったこと，それを自浄作用でフィックスしたことを記録するために，2 を残すことになりました。プロプライエタリな商用ソフトの場合，間違いがあってもこっそり修正して，同じ関数名で提供されてしまえば，ユーザは分析が間違っていたことに気づくことすらできません。科学技術に関するソフトウェアはオープンであるべきだ，という R の思想がここに現れていると言えるでしょう。

27.1.2 混合分布モデルの考え方

混合分布モデルの考え方は，アルクラスターに含まれるかどうかが確率的に決まるというものです。たとえばコイントスをして，表が出たらクラスター A，裏が出たらクラスター B，というように分類するようなものです。コイントスはベルヌーイ分布に下がいますから，確率 θ でクラスター A，確率 $1-\theta$ でクラスター B，と言っていることと同じです (図 27.3)。

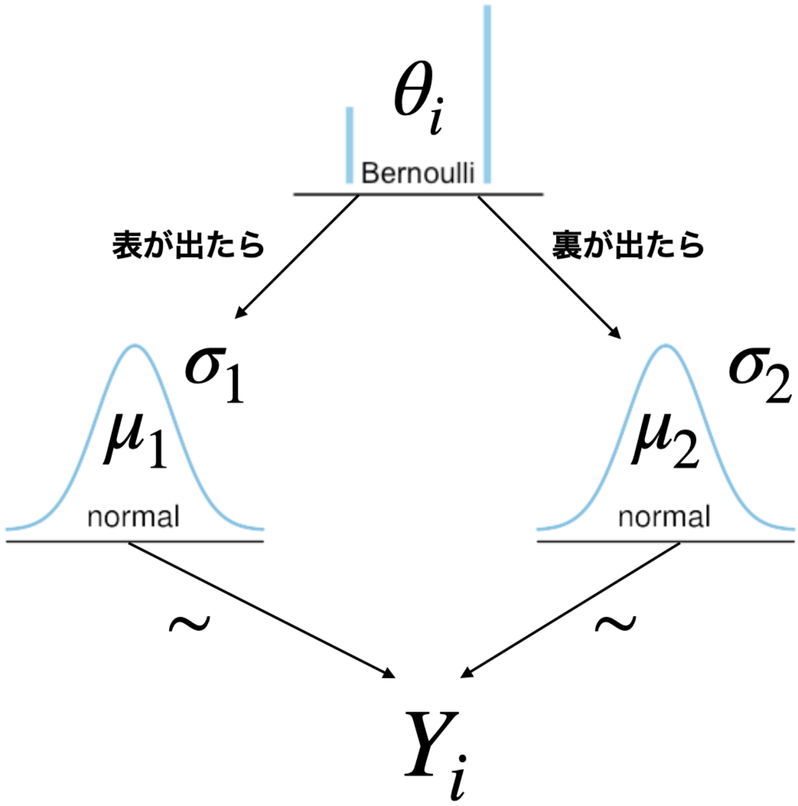


図 27.3　二つの正規分布からデータが出てくるモデル

もちろん 3 つ以上のクラスターに分類することもあるでしょう。この場合は**カテゴリカル分布** (セクション 24.1，Pp.263 を参照) に従う確率変数 z を考えることになります。

データが K 個の正規分布，いずれかから得られていると考えるとしましょう。ここで各正規分布はそれぞれの位置母数 μ_k とスケール母数 σ_k を持っているとします。個々のデータ i がどの正規分布からきているかを表す混合確率 λ_{ik} を考えると，$\lambda_{ik} \geq 0$ で $\sum_{k=1}^{K} \lambda_{ik} = 1$ になります。つまり，i ごとの K-simplex ベクトルを考えることになります。個々のデータ Y_i の出自は，$z[i] \sim Categorical(\lambda_i)$ で決まり，データ $Y_i \sim N(\mu_{z[i]}, \sigma_{z[i]})$ からきている，と考えることになります (図 27.4)。

データ生成メカニズムの設計図は図 27.3，27.4 の通りです。具体的なプログラミングに進む前に，イメージをしっかり掴んでおきましょう。これらの設計図の一番上にある，どのクラスターからデータが出てくるかを確率的に決めるというところがポイントです。ですがこのモデルを実装できるようになれば，データ発生の条件分岐を確率モデルで表現できるようになるのです。確率 θ で左，$1-\theta$ で右のルートを取る，というように，

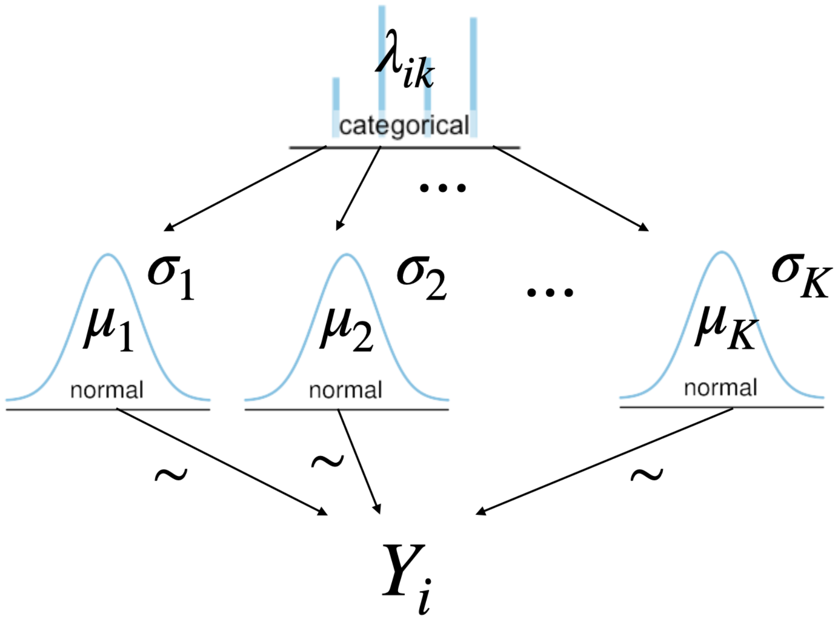


図 27.4　K 個の正規分布からデータが出てくるモデル

データが出てくるメカニズムの背後に潜在的な条件を設定できますから，分析の可能性がグッと広がることになります。実践的にはたとえばマーケティングにおいて，顧客がどういうクラスター (主婦層，独身者，大家族 etc...) に属するかはわかりませんが，買い物の傾向から分類するというようなことができます。あるいは心理学的なシーンでも，潜在的に異なる種類 (抑うつ傾向があるタイプ，ないタイプなど) が，反応傾向の違いから読み取れるようになるかもしれません。

クラスター分析はデータから分類するので，分類後にそれがどう言った変数と関係しているのか，あるいは分類されたクラスターにはどういう特徴があるのかを探索することが一般的です。それでもデータだけから分類できるのは魅力的な手法であると言っていいでしょう。

27.2　ターゲット記法と周辺化消去

さてこの条件分岐をどのようにコードに書いていくか，というところに話を進めましょう。ここで新しく，2 つの技術を導入する必要があります。1 つはターゲット記法，もう 1 つは**周辺化消去 (marginalizing)** です。

27.2.1　ターゲット記法

これまで Stan で，確率変数に従う分布の記法は ∼ を使って表現していました。すなわち，コード 27.1 のような書き方です。

code : 27.1　sampling 記法

```
for(i in 1:N){
    Y[i] ~ normal(mu,sigma);
}
```

これと同じ動きをする, 別の表記方法があります。それが**ターゲット記法**というもので, コード 27.2 のように書きます。

code : 27.2 target 記法

```
for(i in 1:N){
    target += normal_lpdf( Y[i] | mu, sigma);
}
```

何やら奇妙な書き方に見えますが, 順番に見て行きましょう。まず +=の右側, noraml が norma_lpdf になっているところに注目してください。ここで lpdf は log probability density function, すなわち対数確率密度関数という意味です。normal というのがついていますから, 正規分布の対数確率密度関数ということになりますね。ちなみに離散変数の場合は**確率密度 (probability density)** ではなく**確率質量 (probability mass)** になりますから, lpmf, すなわち log probability mass function(対数確率質量関数) になります。具体的には, bernoulli_lpmf とか poisson_lpmf となりますので注意してください。

さて対数確率密度関数ってのがなぜ出てきたのでしょうか。ここで**尤度 (likelihood)** のことを思い出してほしいのですが, 尤度とはデータが**確率分布**から出てくるときの尤もらしさ, 出てきやすさのような指標なのでした。確率分布はパラメータがわかっている時にデータが出てくる確率を記述する関数ですが, 同じ関数でパラメータが未知, データが既知の場合 (多くの研究シーンはこちらですが) に, それを尤度関数とよぶのでした。データが得られた時の尤度を計算し, その尤度が最も大きくなるところをパラメータの推定値とするのが**最尤推定法 (Maximum likelihood method)** であり, 尤度を使って事後分布を算出, 事後分布の形から推定値を考えるのが**ベイズ推定法 (Bayesian inference method)** なのでしたね。つまり MCMC で計算するにはデータ全体の尤度を考えているのです。そしてデータ全体尤度は各データ点の尤度の総積 $\prod$ で考える必要がありますが, 確率の掛け算は数字が小さくなるので**対数尤度 (log-likelihood)** の総和で計算するのでした。

具体的に書くと, データ $\boldsymbol{Y} = Y_1, Y_2, Y_3$ がパラメータ θ のベルヌーイ分布から出てきている場合, データの尤度は

$$L(\boldsymbol{Y}|\theta) = \prod_{i=1}^{3} bernoulli(Y_i|\theta) = bernoulli(Y_1|\theta) \times bernoulli(Y_2|\theta) \times bernoulli(Y_3|\theta)$$

であり, 対数尤度は

$$LL(\boldsymbol{Y}|\theta) = \sum_{i=1}^{3} \log(bernoulli(Y_i|\theta)) = \log(bernoulli(Y_1|\theta)) + \log(bernoulli(Y_2|\theta)) + \log(bernoulli(Y_3|\theta))$$

となるわけです。ここで $bernoulli(Y|\theta)$ の縦棒 | は, θ が与えられた時の Y の出る確率, という条件付き確率の記号になります。

プログラミング的には, $Y_1, Y_2, \cdots, Y_n$ は for(i in 1:N){ ~ }という形で 1 から N までの繰り返し計算を意味しますから, コード 27.1 や 27.2 でやっていることは, 各データ点についての対数尤度を足し合わせて行っている作業そのものを意味しています。

ここでターゲット記法にある +=ですが, これはプログラミング独特の表記方法で, 数学表記ではありません。プログラミングでは代入を使って, たとえば変数 A に 1 を加えたものを新しい A とする (上書きする), ということを A = A + 1 と表記できます。この +=はこうした上書きを一言で書く記号で, この A の例だと A += 1 とすれば 1 を加えて上書き, と同じ意味になります。これを踏まえてターゲット記法を見ると, target += normal_lpdf(Y[i] | mu, sigma) ですから, target = target + normal_lpdf(Y[i] | mu, sigma) としていることと同じです。つまり, 対数尤

度を足しあげて行ってね，ということを明示的に書いていることになります。`target` は足し合わされたもの，というだけで「目標はこれ」を意味するぐらいの予約語だと思ってください。

サンプリング記法とターゲット記法は，書き方が違えど同じ働きをするのですからどちらでもいいじゃないか，と思うかもしれませんが，ターゲット記法でないと表現できないようなことがあるので，このような代替手法が用意されているのだと思ってください。ターゲット記法でないと表現できないようなこと，というのはもちろん今回の，分布を混ぜ合わせる時がそれです[*4]。

27.2.2 周辺化消去

さてここまでさまざまな問題を解決してくれた Stan ですが，Stan には離散型のパラメータを扱うことができない，という欠点があります[*5]。今回は $z[i] \sim Categorical(\lambda_i)$ と，どの目が出たかというパラメータが離散的ですから，このままではコード化できません。ではどうするか。仕方がないので，「起こりうる可能性をすべて数え上げて考える」という方法を取ることにします。最終的な事後分布は対数尤度関数を足し合わせたものになりますから，条件分岐したルートすべてについて対数尤度関数を考えて，それを足し合わせてやれば良いのです。

簡単な例として図 27.3 にあるような，2 つの正規分布モデルを考えてみましょう。コイントスで表が出れば $N(Y_i \mid \mu_1, \sigma_1)$，裏が出れば $N(Y_i \mid \mu_2, \sigma_2)$ のルートに行けば良いのです。表が出る確率は θ ですから，表が出るルートを通ってデータ Y_i が出てくる確率は $\theta \times N(Y_i \mu_1, \sigma_1)$ になります。同じく裏が出るルートを通ってデータが出てくる確率は，$(1-\theta) \times N(Y_i \mid \mu_2, \sigma_2)$ ということになります。

ですから，最終的にデータ Y_i が出てくる確率は $\theta \times N(Y_i \mid \mu_1, \sigma_1) + (1-\theta) \times N(Y_i \mid \mu_2, \sigma_2)$ だ，ということができるわけです。

さて，これらは確率をそのまま使った表現になっていますが，Stan の中では対数で考えるのでした。つまり表が出るルートの対数尤度は

$$\log(\theta) + \log(N(Y_i \mid \mu_1, \sigma_1))$$

ですし，裏が出るルートもまた同様に log をとって考える必要があります。対数を取ると積が和になるので，$\times \to +$ に変わっていることに注意してください。

ところがこの表ルート，裏ルートは足し合わせないといけません。対数をとったものを直接足し合わせるわけにはいきませんから，対数尤度をただの尤度に戻す，つまり log の逆関数である exp 関数を通す必要があります。ややこしいですが，

$$\exp\left(\log(\theta) + \log(N(Y_i \mid \mu_1, \sigma_1))\right) + \exp\left(\log(1-\theta) + \log(N(Y_i \mid \mu_2, \sigma_2))\right)$$

が尤度であり，これの対数をとったものを `target` に加える必要があるのです。

このように，exp を通して，足し合わせ ($\sum$) て，さらにその log をとるという作業をひとまとめにした関数が，Stan にはあります。それが `log_sum_exp` 関数です。この関数は，`log_sum_exp(x,y)` とすると $\log(\exp(x) + \exp(y))$ という操作をします。この関数に渡すものがベクトルであれば，

$$\texttt{log\_sum\_exp}\left(\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}\right) = \log\left(\sum_{i=1}^{n} exp(x_i)\right)$$

[*4] そのほかにもターゲット記法は，サンプリング記法では省略されている事後対数確率も記録するという特徴があり，ブリッジサンプリングによる事後対数尤度の計算などにはこちらの表記が必要になってきます。もっとも非常に専門的なことなので，ここでは条件分岐の際に必要な記法，と割り切っていただいても構いません。

[*5] JAGS という確率的プログラミング言語でしたら，これは可能です。

という操作をすることになります。

これを使って混合分布モデルの実装をしていきましょう。

27.2.3 混合分布モデルのコード

先ほどの，阪神タイガースにおける選手の年俸分布を 2 つの正規分布から出てきているもの，と考えてコード化したものは，Code27.3 のようになります。

code : 27.3 混合正規モデル

```
data {
  int<lower=1> K;
  int<lower=1> L;
  array[L] real Y;
}

parameters {
  array[L] simplex[K] theta;
  ordered[K] mu;
  array[K] real<lower=0> sigma;
}

transformed parameters{
  array[L] vector[K] lp;
  for (l in 1:L) {
    for (k in 1:K) {
      lp[l,k] = log(theta[l,k])+ normal_lpdf(Y[l]|mu[k],sigma[k]);
    }
  }
}

model{
  for(l in 1:L){
    target+=log_sum_exp(lp[l]);
  }
  sigma ~ cauchy(0,5);
  mu ~ normal(0,10);
}

generated quantities{
  array[L] vector[K] prob_class;
  array[L] int<lower=1,upper=K> pred_class;
  for(l in 1:L){
    prob_class[l] = softmax(lp[l]);
    pred_class[l] = categorical_rng(prob_class[l]);
  }
}
```

■コード解説

data ブロック クラスター数 K，データ長 L，アウトカム変数 Y を宣言しています。

parameters **ブロック** 合計が 1 になる K の長さを持つベクトルである，`simplex` 型ベクトルを，データ数だけ用意します。また，K 個の正規分布からデータが出てきているのですから，平均と標準偏差も K 個用意します。ここで大事なポイントとして，平均 μ が `ordered` 型としてあるところです。このベクトルは，要素 $\mu_1 > \mu_2 > \cdots > \mu_k$ と，要素が大きさの順に並んでいることです。このような制約をかけておかないと，K 個の正規分布がどの位置にあるのか判然としな苦なるという問題が生じます。たとえばあるチェインで $\mu_1 > \mu_2$ となっていて，データ X_k が 1 番目の正規分布から出てきていると考えられたとしましょう。しかし別のチェインで $\mu_1 < \mu_2$ となっていれば，X_k は平均値が大きい方から出てきているのですから，こちらのチェインでは 2 番目の正規分布から出てきたものとして，MCMC サンプルが進むことになります。複数のチェインから得られた MCMC サンプルは最終的に統合されますから，このようになっていると X_k は正規分布 1 からも 2 からも同じように出てきていることになります。各チェインでは正しくサンプリングできているのですが，チェインごとに分布のラベルが違うので，統合すると訳がわからなくなる，この問題をとくに**ラベルスイッチング (Label Switching)** 問題といい，混合分布モデルではよく生じる問題なのです。そこで分布の位置母数に順番の制約を入れて，ラベルの変動を止める必要があります。

transformed parameters **ブロック** 長さ K のベクトルをデータの数 L だけ用意した `lp` 変数を作りました。これは log-probability の略でこのような名前にしましたが，任意の名前で結構です。l 行目のデータがクラスター k に含まれる可能性は，$\theta_{lk} \times N(Y[l] \mid \mu_k, \sigma_k)$ ですから，それの対数をとった形で表現しています。

model **ブロック** モデル尤度のところは，対数 → exp 関数で確率の形 → ベクトルの要素をすべて足し合わせる → log をとって対数尤度にする，という計算をまとめて `log_sum_exp` 関数で行い，それをすべて `target` に追加していくという方法で書いています。事前分布のところは普通のサンプリング記法で書きました。ちなみに μ, σ はそれぞれ K 個ずつありますが，Stan は添字の省略を許してくれます。

generated quantities **ブロック** 久しぶりに生成量ブロックの登場です。ここでは分析結果を使って，l 行目のデータが結局何番目の正規分布から出てきたのかを逆算して出しています。まず，対数尤度で表現されている K 個の要素を持つベクトルを，確率の形に計算したものを `preb_class` 変数にしています。ここで使われている `softmax` という関数は，

$$softmax(x) = \frac{exp(x)}{\sum_{k=1}^{K} exp(x_k)}$$

という計算をするものです。分母はベクトルの総和で分子がその要素ですから，要するに各クラスに入る確率を表すベクトルになっている訳です。これを使って `pred_class` 変数を作っています。今回のデータから予想される所属クラスなので，K 面サイコロを振った出目という形で表現します。`categorical_rng` はカテゴリカル分布からの乱数発生で，その発生確率が先ほどの `prob_class` になっているので，この乱数で出た目が予測されるクラスということです。

これを実行するための R コードをコード 27.4 に用意しました。

code：27.4 混合分布モデルのコード

```
dat <- read_csv("baseballDecade.csv")
dat.tmp <- dat %>%
  filter(position != "投手") %>%
  filter(Games > 50) %>%
  filter(team == "Tigers") %>%
```

```
6     mutate(salary.log = log(salary)) %>%
7     dplyr::select(salary.log, Name) %>%
8     rowid_to_column("ID")
9
10  dataSet <- list(K = 2, L = NROW(dat.tmp), Y = dat.tmp$salary.log)
11  model <- cmdstanr::cmdstan_model("latent.stan")
12  fit <- model$sample(data = dataSet, chains = 4, parallel_chains = 4)
```

■コード解説

1 行目　データファイル baseballDecade.csv を読み込みます。

2-8 行目　分析用のデータに加工しています。

- **3 行目**　フィルターをかけてデータを野手だけのものにします。
- **4 行目**　フィルターをさらにかけて，50 試合場出ている選手に限定します。
- **5 行目**　チームが Tigers の選手だけに絞っています。
- **6 行目**　年俸のデータは歪んでいるので，対数をとって正規分布の形に近似させました[*6]。
- **7 行目**　使う変数だけに絞っています。
- **8 行目**　行番号を ID という変数名にして追加しました

10 行目　データセットを作っています。

11 行目　cmdstan でコンパイルしています。rstan パッケージを使う場合は rstan のコンパイル関数を使ってください。

12 行目　cmdstan のサンプリング関数です。rstan パッケージを使う場合は rstan のサンプリング関数を使ってください。

推定した結果を見てみましょう。私の環境では出力 10 のようになりました。

MCMC の結果 10

```
# A tibble: 4 × 7
  name          EAP       MED       MAP        SD       L95       U95
  <chr>     <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 mu[1]         7.624     7.624     7.638     0.107     7.415     7.826
2 mu[2]         9.702     9.699    10.017     0.284     9.164    10.138
3 sigma[1]      0.656     0.653     0.646     0.063     0.544     0.788
4 sigma[2]      0.762     0.786     0.901     0.237     0.380     1.192
```

対数をとった時の平均値が 7.624 と 9.702 ですから，指数関数を入れて読み直せば $\exp(7.624) = 2046.733$,$\exp(9.702) = 16350.28x$ となります。2000 万円クラスの選手と，1 億 6500 万円クラスの選手の 2 つのグループに分かれることが見て取れますね。事後予測分布から，所得平均の高い「超一流」選手と「一般的」な選手とに分けて分布を色分けしてみました (図 27.5)。データから，この選手がどちらのクラスに所属しているかも想像できるところがおもしろいですね。

[*6] データを変形させずに確率分布の方で対応するのがよい，と常々話をしてきましたが，ここではデータを変形させています。実は左に歪んだような分布は対数正規分布というのがあり，そこからのデータ生成を考えても良いのですが，対数正規分布はデータの対数が従う正規分布というだけなので，データの方を変えてしまった方が説明しやすいのです。比率や度数など，データの生成メカニズムがそもそも違うばあいは，データを変換せずに使うのが正しいやり方です。

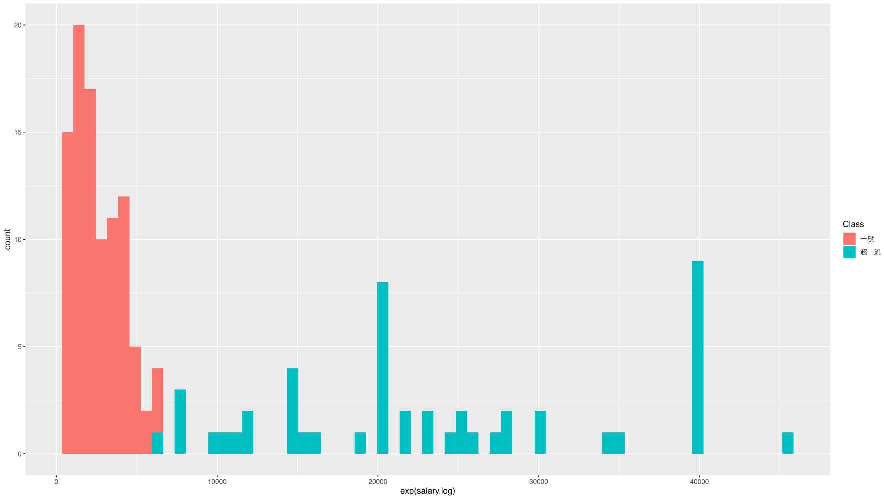


図 27.5　選手のクラスタリング

27.3　ゼロ過剰ポアソン分布モデル

　さて，異なる分布の混ぜ合わせ，離散確率分布による条件分岐の例として，もう 1 つ別の例を示しましょう。次の図 27.6 を見てください。

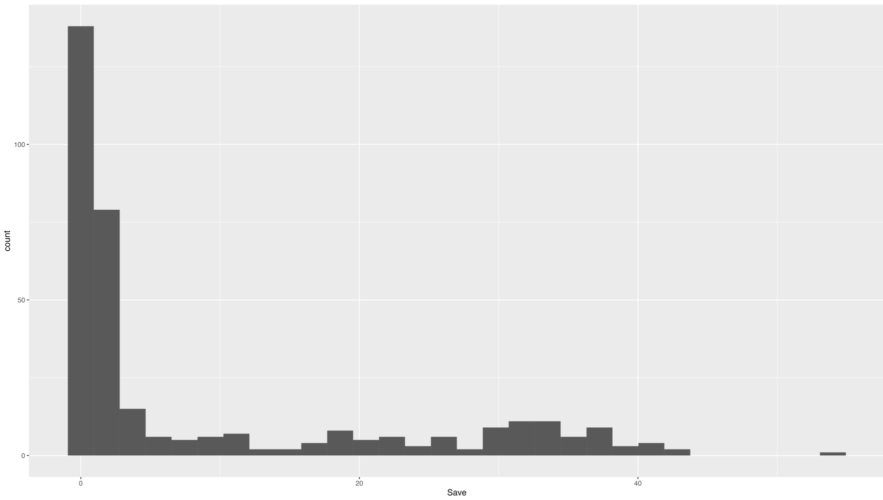


図 27.6　投手のセーブ数

　これは投手のセーブ数の度数分布です。野球は 9 回のターン制で，1 回の最初から投げる投手は先発投手と言います。先発投手が 9 回まで投げ切れればいいのですが，疲労も溜まるし戦術によっては投手の打席に代打を投入することもあります。投手が降板した場合は，次の投手が代わって続きから投げますが，「勝っている試合で後半まできたので，最後の 1，2 回は確実に勝ち逃げできるようにしたい」というときに投入されるのが抑え (クローザー) と呼ばれる投手です。最近のプロ野球は分業が進んでいますから，先発投手が完投することは少なく，100 球程度で交代し，中継ぎ – 抑えと投手リレーしていくのが一般的な戦術です。さて，先発投手は最初から出てきて 6，7 回まで投げますから (途中でボロボロに打たれたり怪我で降板などが

なければ，ですが)，毎日のように登板することはなく，一度登板すると 3，4 日休んで次の試合にでる，ということになります。日本プロ野球は年間 140 試合近くありますが，通年で 2-30 回も登板することはありません。一方，抑えの投手は最後の 8，9 回を投げる程度で，しかも負けている試合などでは出番がありません。必然的に中 4 日の休憩なども入りませんから，試合数はたくさん出ることができます。そして抑えに成功するとセーブポイントというのがつきますが，このセーブポイントは優秀なクローザーであれば年間数十ポイント取ることができる訳です。

さて，図 27.6 にあるようにこのセーブポイントを見てみますと，ほとんどの投手がゼロになっています。以下データをみますと，1 セーブが 53 人，2 セーブが 26 人･･･となって，20〜40 セーブある投手が 10 人弱ずつぐらいいる，という分布をしています。これは先ほども言った分業制のせいで，先発投手にセーブポイントがつくことはありませんし，先発投手は 1 チームに 10 人弱ぐらいは揃えているでしょうから，投手全体で見るとセーブポイントは 0 の人が多いのです。

このセーブポイント，0 以上の整数を取りますからポアソン分布に従うと考えられるのですが，ポアソン分布のパラメータ λ をどう変えても (図 27.7)，こんな形にはなりません。

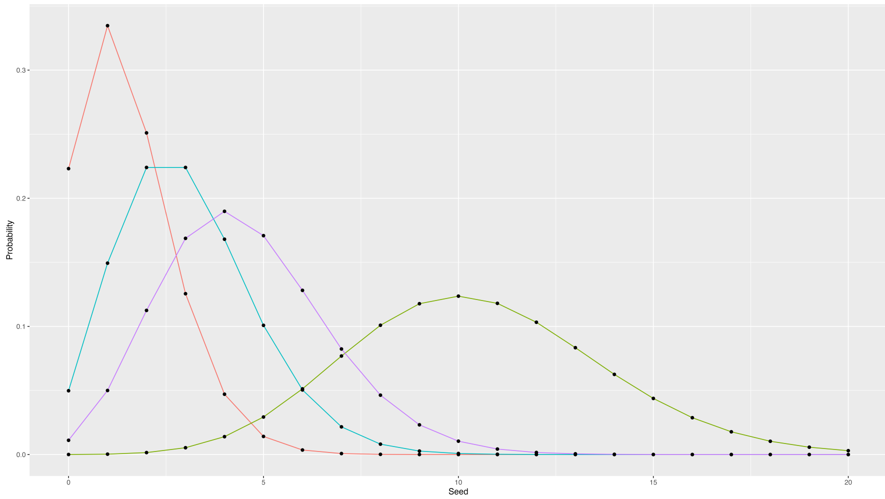


図 27.7　さまざまなポアソン分布。λ は 1.5，3，4.5，10.5 で描画

そこで混合分布です。まずセーブ数が 0 かどうかを考えます。セーブ数 0 というのは，そもそもクローザーではないか，クローザーなのに 0 セーブポイントのへっぽこか，の二択です。セーブ数が 0 でない限り，ポアソン分布からセーブ数が生成される，と考えるのです。設計図のイメージは図 27.8 の通りです。

このモデルがちょっと特殊なのは，ご覧の通り 0 というデータの値一箇所に極端な偏りがあるところです。この分布のことを**ゼロ過剰ポアソン (Zero-Inflated Poisson)** といいます。

注意してほしいのですが，ポアソン分布から 0 の度数が出てくることもあり得ます。逆に考えると，データ $Y_i = 0$ であれば，これはクローザーでないからそうなっているのか，クローザーなのにヘッポコでセーブ数がつかなかったのかのどちらか，という判断をしなければならないということです。つまり，データからの分岐を考えることになります。クローザーかどうかは確率 θ で決まるとして，θ ならクローザー，$1-\theta$ ならクローザーじゃないとすると，

$$\begin{cases} \text{データが 0 ではない} & \rightarrow \theta \times Poisson(Y_i \mid \lambda) \\ \text{データが 0 だ} & \rightarrow 1-\theta \text{ or } \theta \times Poisson(0 \mid \lambda) \end{cases} \tag{27.1}$$

このデータが 0 のときに生成メカニズムが条件分岐していますから，ここに `log_sum_exp` 関数を入れる必

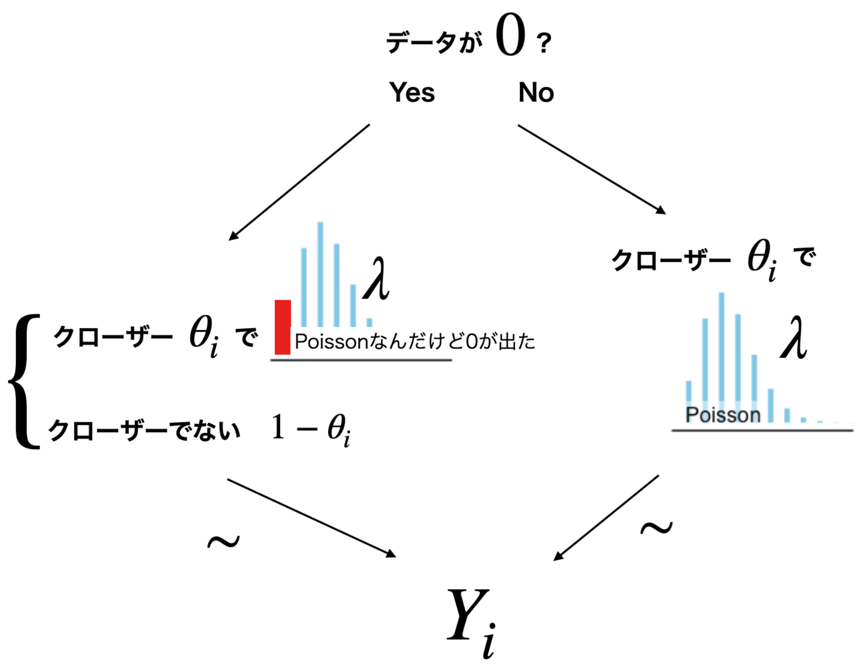


図 27.8　ポアソン分布との混ぜ合わせ

要があります。

$$\texttt{log\_sum\_exp}\begin{pmatrix} \log(1-\theta) \\ \log(\theta) + \texttt{poisson\_lpmf}(0 \mid \lambda) \end{pmatrix}$$

このことを念頭において，コードを書いてみましょう (コード 27.5)。

code : 27.5　ゼロ過剰ポアソン分布

```
data{
  int L;
  array[L] int Y;
}

parameters{
  real<lower=0,upper=1> theta;
  real<lower=0> lambda;
}

model{
  for(l in 1:L){
    if(Y[l]==0){
      target += log_sum_exp(log(1-theta),log(theta)+poisson_lpmf(0|lambda));
    }else{
      target += log(theta) + poisson_lpmf(Y[l]|lambda);
    }
  }
}
```

■コード解説

data ブロック　データ長 L，アウトカム変数 Y を宣言しています。

parameters ブロック　クローザーかどうかを決める θ と，ポアソン分布のパラメータ λ がパラメータです。

model ブロック　データの中で，プログラミング言語としての if 文による条件分岐が出てきています[*7]。データが 0 であれば，クローザーでないか，ポアソン分布からゼロが出ているかです。そうでなければ，確率 θ 経由でのポアソン分布です。

これを実行するための R コードはコード 27.6 のようになります。

code : 27.6　ゼロ過剰ポアソンモデルのコード

```r
dat.tmp <- dat %>%
  dplyr::filter(position == "投手") %>%
  dplyr::filter(Games > 50) %>%
  dplyr::select(Save)

model <- cmdstan_model("ziPoisson.stan")
dataSet <- list(L = NROW(dat.tmp), Y = dat.tmp$Save)
fit <- model$sample(
  data = dataSet,
  chains = 4,
  parallel_chains = 4
)
```

推定した結果を見てみましょう。私の環境では出力 11 のようになりました。

MCMC の結果 11

```
# A tibble: 2 × 7
  name          EAP       MED       MAP        SD       L95       U95
  <chr>   <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 lambda     14.666    14.665    14.661     0.261    14.154    15.185
2 theta       0.606     0.606     0.608     0.025     0.556     0.655
```

これを見ると，6 割の確率でクローザー判定され，クローザーであれば $\lambda = 14.7$ ぐらいのポアソン分布でデータが生成されるということですね。事後予測分布のように，この確率モデルで出てくるデータを図 27.9 に示してみました。単一の分布では表現できないモデル化ができているのではないでしょうか。もっとも，今回は 1，2 回セーブポイントがついている人のところをうまく表現できませんでした。しかしコードを書き換えることで，セーブ数 0 の先発投手と，本職のクローザーに加え，セーブ数が一桁ぐらいの中継ぎ投手の三種類に分割する，ということもできるでしょう。

ここでみた分布の混ぜ方は，たとえば「お店に初来店した人とリピーターの来店数」とか，「20 歳までの恋愛経験の数」のようなデータがあったときに，その特徴から分割してモデルを調整できます。研究のシーンでも，たとえば「質問紙で何も考えずにどちらとも言えないと答える人と，ちゃんと考えてどちらとも言えないに丸をつけた人」というように回答者のバイアスを除去してその特徴を考えるなど[*8]，積極的な分析をできるのです。

さらに言えば，このポアソン分布の λ に線形モデルを入れて，ゼロ過剰ポアソン回帰分析にすることだってできますね。たとえばクローザーのセーブ数が年俸によって説明される，というモデルにするなら，コード 27.7

[*7] プログラミングでの条件分岐については第 16 講，セクション 16.3.3，Pp.172 でやりましたね。忘れた人は戻って再確認しましょう。

[*8] この場合は中点の 3 が多い，というようなケースでしょうから，3 過剰正規分布とでも言えば良いでしょうか。応用例として清水 (2018) などがあります。

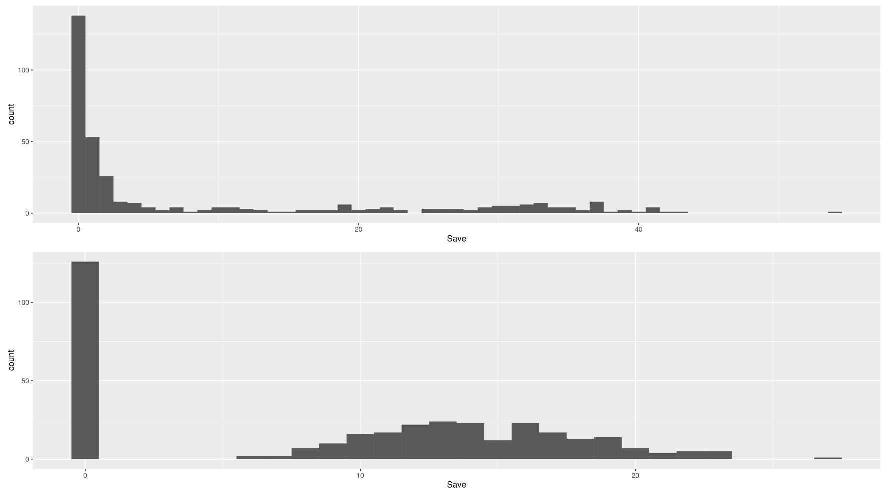


図 27.9　ゼロ過剰ポアソン分布による当てはめ。上がロウデータ, 下が事後予測分布

のようにすれば良いのです。

code : 27.7　ゼロ過剰ポアソン回帰

```
data{
  int L;
  array[L] int Y;
  array[L] real X;
}

parameters{
  real<lower=0,upper=1> theta;
  real beta0;
  real beta1;
}

transformed parameters{
  real<lower=0> lambda[L];
  for(l in 1:L){
    lambda[l] = exp(beta0 + beta1 * X[l]);
  }
}

model{
  for(l in 1:L){
    if(Y[l]==0){
      target += log_sum_exp(log(1-theta),log(theta)+poisson_lpmf(0|lambda[l]));
    }else{
      target += log(theta) + poisson_lpmf(Y[l]|lambda[l]);
    }
  }
}
```

これは一般化線形モデルの応用ですから, ポアソン分布に限らず使える技であることは, すぐにお分かりいた

だけることと思います。

データの分布や特徴を考えて，それにあった分析モデルを作っていくのは，モデラーとしての階段を一歩登ったことになります。図 27.10 には統計分析家の成長ステップの模式図を描いてみました[*9]。最初はデータの記述や線形モデルの当てはめなど，得られたデータに振り回されるような形でモデルを作り上げてきましたが，今回のように分布を混ぜ合わせると，さらにその表現力が豊かになることが実感できたのではないでしょうか。図にあるように，確率モデルのパラメータが心理事象の (数学的) 記述から導出され，それがデータと合致する可動かで検証するようになれれば，さらにもう一段階研究のレベルが上がることになるでしょう。最終的には，心理学理論の方から新しい確率分布が導出される日が来るかもしれませんね。

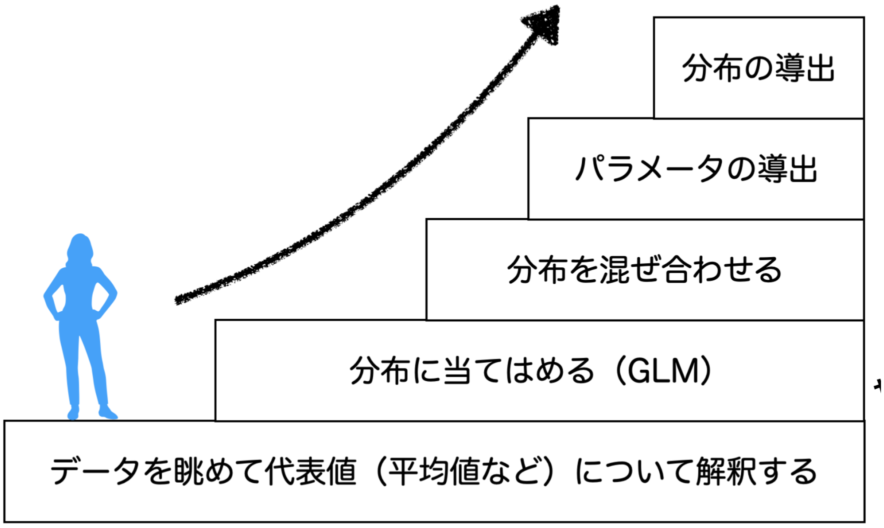


図 27.10　統計モデラーの道。浜田宏先生 (東北大学) のアイデアをもとにイラスト化したもの

27.4　課題

以下のモデルを分析する R/Stan コードを提出してください。結果の解釈などを，スクリプトのコメントアウトや別添ファイルなどで提供してもらえるとなお良いです。もちろん Rmd ファイルでの提出であれば完璧です。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

■混合分布モデル　今回は阪神 Tigers のデータセットで行ったが，他の球団ではどうだろうか。別の球団データの年俸 (対数をとったもの) のヒストグラムを描き，混合分布モデルが適用できそうなものを見つけたら，実際に当てはめて推定してみましょう。

■ゼロ過剰ポアソン回帰モデル　セーブポイントが年俸で説明できるとしたゼロ過剰ポアソン回帰を実行してみましょう。年俸データは標準化して単位を整えておくと良いでしょう。

[*9] この図は東北大学大学院文学研究科の浜田宏先生が，専修大学社会知性開発研究センター主催の研究会 (2019.2.26 開催，於専修大学サテライトキャンパス) にてご講演いただいた際の，資料をもとに作図したものです。

第 28 章

確率的プログラミング；項目反応理論

さてここまで, 線形モデルを中心に**モデリング**のステップを一段一段登ってきました。この後は, これまでの技術や考え方を応用し, より実践的なプログラミングへと進んでいきます。さまざまなモデルの確率的表現や, そのコーディング技法を学ぶことで, ご自身の研究にも応用できる技術的ヒントが得られるかもしれません。

今回は第 4 講, 第 5 講および第 11 講で学んだ項目反応理論を，Stan を使って実装する例を見ていきたいと思います。

28.1 ロジスティックモデルの復習

28.1.1 パラメータモデル側から

ここで簡単に前期の復習をしておきます。

目に見えないものを測定するという意味では, 心理尺度や学力検査は同じ技術であるというところから, 測定モデルの話を導入しました。とくに学力検査で想定されている測定対象, 学力は, 正規分布していると考えられること, 累積的な能力の蓄積の程度を測定していると考えられることから, 正答率が累積正規分布をつかった潜在能力 (学力) の関数と考えられるというところからモデル化がすすみました。累積正規分布を直接扱うモデルもありますが, 一般にはそれによく近似する**ロジスティック関数**, すなわち

$$f(x) = \frac{1}{1 + exp(-1.7x)}$$

を使って近似することで, 項目の特徴を描写することが考えられたのでした。

x 軸が潜在能力 θ を表していると考えれば, 縦軸は能力に応じた**通過率**になると考えられます。この関数を左右に動かす位置パラメータ，b_j を加えたモデルが **1 パラメータ・ロジスティックモデル (One Parameter Logistic model)** であり, 次の式 28.1 で表現されるのでした。

$$p_j(\theta) = \frac{1}{1 + exp(-1.7(\theta - b_j))} \tag{28.1}$$

ここで $p_j(\theta)$ は項目 j に対する θ の通過率であり，b_j は**困難度 (difficulty)** 母数と言われるものです。

さらに項目を特徴づけるために, 傾きの大きさを表すパラメータ，a_j を加えたモデルが **2 パラメータ・ロジスティックモデル (Two Parameter Logistic model)** です。モデル式は式 28.2 で表されます。

$$p_j(\theta) = \frac{1}{1 + exp(-1.7a_j(\theta - b_j))} \tag{28.2}$$

a_j はのことをとくに**識別力 (discriminant)** と呼ぶのでした。a_j の値によって傾きがどのように変わる

かを，図 28.1 の二段目をみて確認しておきましょう。

最後に，第 4 講では紹介していませんでしたが，3 つ目のパラメータである c_j，**当て推量母数 (guessing parameter)** を入れたモデルも紹介しておきます。モデルは式 28.3 のように表されます。

$$p_j(\theta) = c_j + \frac{1 - c_j}{1 + exp(-1.7a_j(\theta - b_j))} \tag{28.3}$$

ここには切片のように定数 c_j が入っていますから，関数のベースラインが上に上がるイメージです。これはつまり，学力 θ にかかわらず一定の正答率を有するということで，「適当に答えても当たる確率」ということから当て推量とよばれています。

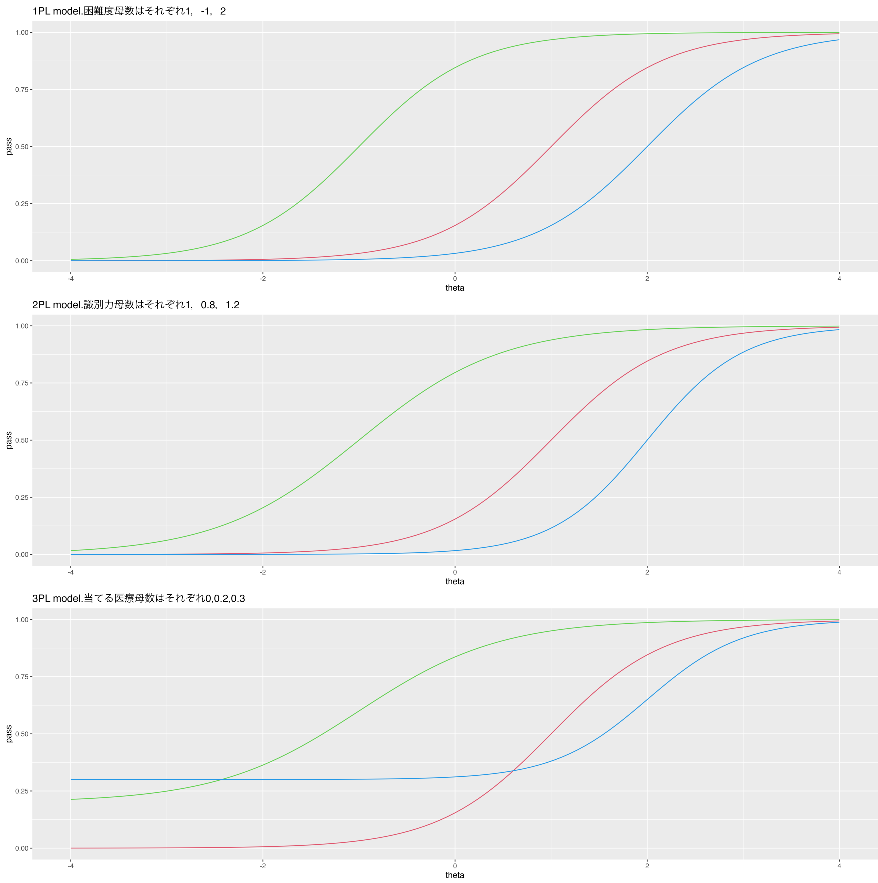


図 28.1　上から順に，1，2，3 パラメータロジスティックモデル。どこがどう変わっていくか確認しておこう。

28.1.2　確率モデル側から

さて今度は同じモデルをデータとの対応の観点から見てみましょう。テストのデータは 0/1 の二値であり，ベルヌーイ分布から出てきていると考えることができます。またロジスティックモデルはその言葉の通りロジスティック関数を使って，$-\infty$ から $+\infty$ まであり得る θ の値を，0 から 1 の範囲に入るように変換しているわけです。これは一般化線形モデルの文脈でいうところの，**ロジスティック回帰**分析であり，**リンク関数**がロジット関数，**逆リンク関数**がロジスティック関数になっているモデルだと考えることができます。

データ生成のモデル設計図として考えると，図 28.2 のようになります。パッケージを使っての分析の時は**最尤法**による推定結果でしたが，ここで被検者母数に標準正規分布を，項目母数に正規分布を事前分布として

おいただけで，ベイズ推定モデルに置き換えることができました。

これまで利用してきた技術の応用で考えることができますから，すぐにでも実装できそうですね！

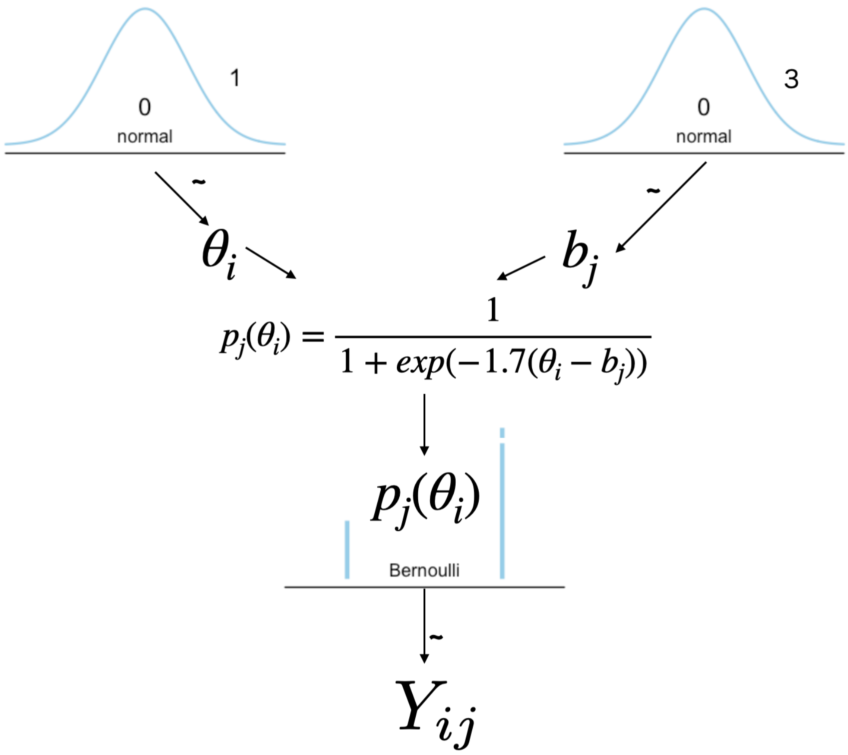


図 28.2　確率モデルとしての設計図。ロジスティック関数とベルヌーイ分布の組み合わせでテスト理論のモデルが表現できる。

28.2　ロジスティックモデルでの実装

それでは設計図をもとにモデルをコードにしていきましょう。

項目数が M，被検者数が N の $N \times M$ サイズの行列でデータが与えられていると考え，行列の各要素に対して通過率 p_{ij} を計算し，それをベルヌーイ分布に当てはめるという形で実装したのがコード 28.1 になります。

code：28.1　1PL モデル

```
data{
  int<lower=0> M;
  int<lower=0> N;
  array[N,M] int<lower=0,upper=1> resp;
}

parameters{
  array[M] real<lower=-5,upper=5> b;
  array[N] real theta;
}

transformed parameters{
```

```
  array[N,M] real<lower=0,upper=1> prob;
  for(n in 1:N){
    for(m in 1:M){
      prob[n,m] = inv_logit(1.7*(theta[n]-b[m]));
    }
  }
}

model{
  for(n in 1:N){
    for(m in 1:M){
      resp[n,m] ~ bernoulli(prob[n,m]);
    }
  }
  //prior
  b ~ normal(0,3);
  theta ~ normal(0,1);
}
```

今回はわかりやすくするために，transformed parameters ブロックを使いましたが，直接 model ブロックに書き込んでも構いませんし，bernoulli_logit 関数を使えばさらに高速に安定した推定結果が得られます。

transformed parameters ブロックにパラメータを追加するだけで，2PL,3PL モデルに拡張することも容易にできます。2PL モデルに拡張した例がコード 28.2，3PL モデルに拡張した例がコード 28.3 です。

code : 28.2　2PL モデル

```
...(前略)...
parameters{
  array[M] real<lower=0> a;
  array[M] real<lower=-5,upper=5> b;
  array[N] real theta;
}

transformed parameters{
  array[N,M] real<lower=0,upper=1> prob;
  for(n in 1:N){
    for(m in 1:M){
      prob[n,m] = inv_logit(1.7*a[m]*(theta[n]-b[m]));
    }
  }
}
...(後略)...
```

code : 28.3　3PL モデル

```
...(前略)...
parameters{
  array[M] real<lower=0> a;
  array[M] real<lower=-5,upper=5> b;
  array[M] real<lower=0,upper=1> c;
  array[N] real theta;
```

```
}

transformed parameters{
  array[N,M] real<lower=0,upper=1> prob;
  for(n in 1:N){
    for(m in 1:M){
      prob[n,m] = c[m] + (1-c[m])*inv_logit(1.7*a[m]*(theta[n]-b[m]));
    }
  }
}
...(後略)...
```

第 11 講で使ったサンプルコードをつかって，このモデルで推定するためのコードがコード 28.4 です (cmdstanr での例)。

code : 28.4　IRT のコード

```
dat <- read_csv("IRTsample.csv")
model_1pl <- cmdstan_model("oneParameter.stan")
model_2pl <- cmdstan_model("twoParameters.stan")
model_3pl <- cmdstan_model("threeParameters.stan")

dataSet <- list(N = NROW(dat), M = NCOL(dat), resp = as.matrix(dat))
fit1 <- model_1pl$sample(data = dataSet, chains = 4, parallel_chains = 4)
fit2 <- model_2pl$sample(data = dataSet, chains = 4, parallel_chains = 4)
fit3 <- model_3pl$sample(data = dataSet, chains = 4, parallel_chains = 4)
```

分析結果を見てみましょう。**EAP 推定値**はそれぞれ表 28.1, 図 28.3 のようになりました。

表 28.1　それぞれのパラメータ推定値

	1PL	2PL		3PL		
Qid	b_j	a_j	b_j	a_j	b_j	c_j
1	0.727	0.347	1.716	2.865	1.567	0.221
2	-1.605	0.724	-2.140	1.818	-0.761	0.557
3	-1.175	1.124	-1.229	2.900	-0.626	0.338
4	-1.077	0.955	-1.222	2.033	-0.535	0.362
5	0.577	0.691	0.771	2.451	0.946	0.159
6	1.271	0.694	1.742	1.100	1.659	0.050
7	0.564	0.692	0.753	1.027	0.941	0.099
8	1.590	0.360	3.650	1.581	3.355	0.084
9	0.514	0.484	0.893	0.836	1.205	0.142
10	-0.864	1.214	-0.887	2.547	-0.395	0.298

28.3　整然データでの分析

このように，IRT モデルを簡単に実装できました。

ところで，このモデルをもう少し使いやすくするために，データを**整然データ (tidy data)** にすることを考

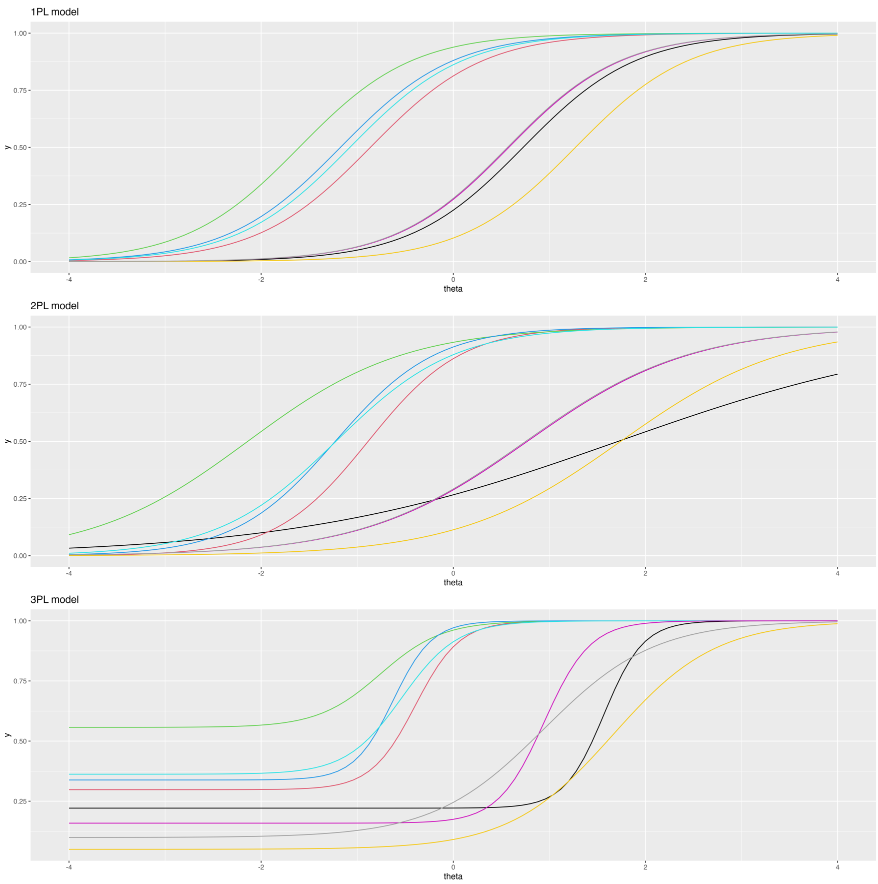


図 28.3　推定値を使って描いた各モデルの ICC

えてみましょう。というのも，今はデータを行列形で渡していますが，これだと欠損値があった場合にうまく機能しないからです。Stan はデータに欠損値を取ることができず，与えられるデータに `NA` が入っていることは許されません。

整然データは，1 行に 1 つの数字が入っているようなデータで (セクション 22.3.1，Pp.247 参照)，その行を見るだけですべての情報が手に入るようなデータです。行列型のデータは，人の ID を行ラベルで，変数の ID を列ラベルでみなければなりませんので，参照方向が 1 つに定められていません。具体的には次のような形のデータになります (**R の出力** 28.1)。この形式ですと，一行で誰の (`Pid`)，どの問いに対する (`Qid`) 反応か (`value`) ということがわかりますね。

R の出力 28.1: 整然データになったテストデータ

```
# A tibble: 20 × 3
     Pid   Qid value
   <int> <dbl> <dbl>
 1     1     1     0
 2     1     2     0
 3     1     3     1
 4     1     4     0
 5     1     5     0
 6     1     6     0
 7     1     7     0
 8     1     8     0
 9     1     9     0
10     1    10     0
11     2     1     1
12     2     2     1
13     2     3     0
```

もし欠損値があればその行を削除してしまえば，完全データになりますから，Stan に欠損値を与えなくて済むことになります。

もちろんこれに対応する形で，Stan のコードを書き換える必要があります。こんどはデータも長くなりますから，何行目に誰のど問いに対する反応が入っているかを，識別変数を使いながら指定することになります。具体的には次のようなコード例になるでしょう (コード 28.5)。

code : 28.5 Tidy Data に対応した 2PL モデル

```
data{
  int<lower=0> L;
  int<lower=0> N;
  int<lower=0> M;
  array[L] int<lower=0,upper=N> Pid;
  array[L] int<lower=0,upper=M> Qid;
  array[L] int<lower=0,upper=1> resp;
}

parameters{
  array[M] real<lower=0> a;
  array[M] real<lower=-5,upper=5> b;
  array[N] real theta;
}

transformed parameters{
  array[N,M] real<lower=0,upper=1> prob;
  for(n in 1:N){
    for(m in 1:M){
      prob[n,m] = inv_logit(1.7*a[m]*(theta[n]-b[m]));
    }
  }
}

```

```
model{
  for(l in 1:L){
    resp[l] ~ bernoulli(prob[Pid[l],Qid[l]]);
  }
  //prior
  a ~ normal(0,3);
  b ~ normal(0,3);
  theta ~ normal(0,1);
}
```

■コード解説

data ブロック　データ長 L, 最大被検者数 N, 最大項目数 M, 被検者識別変数 Pid, 項目識別変数 Qid, L 行目の反応 resp としてデータを受け取ります。

parameters ブロック　これまでと同じです。

transformed parameters ブロック　ここも手を加える必要がありません。

model ブロック　L 行目の反応に対して, 識別変数で特定された確率をつかってモデルを当てはめます。

これの結果はこれまでのモデルと変わりませんが, 欠損値に対応できたはずですので, 元の完全データに技と欠損値を与えて改変し, 結果がどう変わるかを確認してみましょう。

code : 28.6　tidy データにして技と欠損値を与える

```
1  dat.tmp <- dat %>%
2    rowid_to_column("Pid") %>%
3    pivot_longer(-Pid) %>%
4    mutate(Qid = str_extract(name, pattern = "\\d+") %>% as.numeric()) %>%
5    dplyr::select(Pid,Qid,value)
6
7  # わざと欠損値を与える
8  dat.tmp$value[1] <- NA
9  dat.tmp$value[11:13] <- NA
```

■コード解説

1 行目　dat に入っている行列型・完全データを変形していきます。

2 行目　一行ごとに被検者の反応が入っていますので, 行番号を被検者識別変数 Pid として作ります。

3 行目　データを縦長にします。その時のキーとなるのは, 先ほど作った Pid で, この変数は除いて縦長にする関数が pivot_longer です。

4 行目　少し技巧的ですが, ここまでの段階で変数は Pid,name,value という 3 つになっています。なかでも name 変数には元データの変数名が入っていますので, これを加工して問題番号を取り出します。str_extract 関数の str とは string, すなわち文字列を扱う関数であるという意味です。str_extract は文字列から条件に合ったものを抜き出す extract というもので, 条件を pattern で指定しています。ここで\\d+ とあるのは**正規表現 (regular expression)** というもので, 文字列を一定の規則に従って特定の文字列を表現する方法です。ここでは数字だけを抜き出しています[*1]。

[*1] 少し丁寧にいうと, \d は正規表現で数字を意味する記号です。ただ, バックスラッシュ (\) がそのままでは正規表現の記号だと認識されず特殊文字だという宣言をしている (エスケープシーケンス, といいます) だけになってしまいます。そこで,「特殊文字

また，このようにして取り出せた数字は文字列としての数字ですので，これを `as.numeric` 関数に送ることで，数値であることを教えています。

5 行目 被検者識別変数，項目識別変数，反応の値だけにデータを限定しています。

7 行目 ここでこのデータセットの 1 行目に欠損値 NA を上書きしています。わざと欠損させたのです。

8 行目 同じく 11 行目から 13 行目までも欠損値に上書きしてしまいました。

こうしてできたデータセットは次のようになります (R **の出力** 28.2)。

R の出力 28.2: 整然データになったテストデータ

```
# A tibble: 20 × 3
     Pid   Qid value
   <int> <dbl> <dbl>
 1     1     1    NA
 2     1     2     0
 3     1     3     1
 4     1     4     0
 5     1     5     0
 6     1     6     0
 7     1     7     0
 8     1     8     0
 9     1     9     0
10     1    10     0
11     2     1    NA
12     2     2    NA
13     2     3    NA
14     2     4     0
15     2     5     0
16     2     6     0
17     2     7     0
18     2     8     0
19     2     9     0
20     2    10     1
```

確かに数カ所，欠けているところができましたね。これをそのまま渡すことができませんから，欠損のあるところは削除してデータセットを作り，推定してみましょう。

code : 28.7 データセットを作って推定する

```
# 欠損値を消したデータセットにする
dat.tmp <- na.omit(dat.tmp)

dataSet <- list(
  L = NROW(dat.tmp), N = max(dat.tmp$Pid), M = max(dat.tmp$Qid),
  Pid = dat.tmp$Pid, Qid = dat.tmp$Qid,
  resp = dat.tmp$value
)
```

のバックスラッシュだよ」を表すために重ねて\\としています。また，+ は正規表現でいうところの「直前の文字が 1 回以上繰り返される」という意味です。今回は 10 という数字が出てくる可能性があり，\\d だけだと 1 しか抜き出せないので，数字が連なっていたら全体を取り出すようにしています。要するに，数字を全部取り出しましょう，が正規表現では\\d+ になるわけです。

```

model_2pl_ver2 <- cmdstan_model("twoParameters2.stan")
fit2.2 <- model_2pl_ver2$sample(
  data = dataSet,
  chains = 4,
  parallel_chains = 4
)
```

欠損があっても，データを整形して欠損を取り除いた形で分析し，問題なく推定できたと思います。欠損が含まれているから，そのデータはすべて使い物にならないと考えるのではなく，データの存在するところ・利用できるところは利用しつくすという有効活用ができたと思います。

最後に，推定結果を確認しておきましょう。1 人目の被検者は 1 つ，2 人目の被検者は 3 つの欠損がありました。つまり他の被検者は 10 問分の情報を持っているのに，この 2 人はそれぞれ 9 問，7 問分しか情報が得られなかったことになります。そのことが結果の推定値にどう変わるのかというと，出力 12 の通りです。

MCMC の結果 12

```
# A tibble: 3 × 7
  name          EAP      MED      MAP       SD      L95      U95
  <chr>     <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 theta[1]   -1.403   -1.392   -1.438    0.548   -2.530   -0.366
2 theta[2]   -0.774   -0.778   -0.809    0.606   -1.982    0.372
3 theta[3]   -0.656   -0.659   -0.620    0.508   -1.654    0.331
```

これを見ると，10 問分の情報を持っている 3 人目の被検者の能力値が $\theta_3 = -0.656$ と推定されていますが，その SD が 0.508 です。これに比べて，9 問しか情報のない被検者 1 の SD は 0.548，7 問しか情報のない被検者 2 の SD は 0.606 と，情報が少なくなると SD が大きくなっていくことがわかります。推定値の SD が大きいということは，幅が広い，すなわちわからないことがより多くあることを意味します。得られる情報が少なければ，絞り込みが難しくなるというのがデータにも表れていることがわかりますね。

28.4 課題

本講で扱った，1PL，2PL，3PL モデルそれぞれを実行する，Stan ファイルや R コードを提出してください。ただし，いずれのモデルも整然データに対応したコードになっている必要があります。データや R コードはシラバスのサイトを通じて提供されています。不明な点がありましたら，TA あるいは小杉まで連絡して指導を受けてください。

第 29 章

確率的プログラミング；変化点と折線回帰

さて今回は，今までの線形モデルとはちょっと違うモデリングになります。その名も変化点検出，そして折線回帰です。タイトルだけでも面白そうではありませんか？

さらに今回は次のようなデータを扱います (図 29.1)。何を隠そうこのデータは，私の体重の推移のデータなのです。私のモーニング・ルーティンとして，朝目が覚めるとトイレに行きます。そして体重計にのり，体重と体脂肪を測定します。その後顔を洗って[*1]，計測値を iPhone のアプリに書き込むというのがあります[*2]。このルーチンも早いもので 10 年近く続けていることになります (一時期やめていたこともありましたが，最古のレコードは 2012 年 06 月 08 日です)。ともかく，このデータを使って，いろいろ遊んでみようと思います。

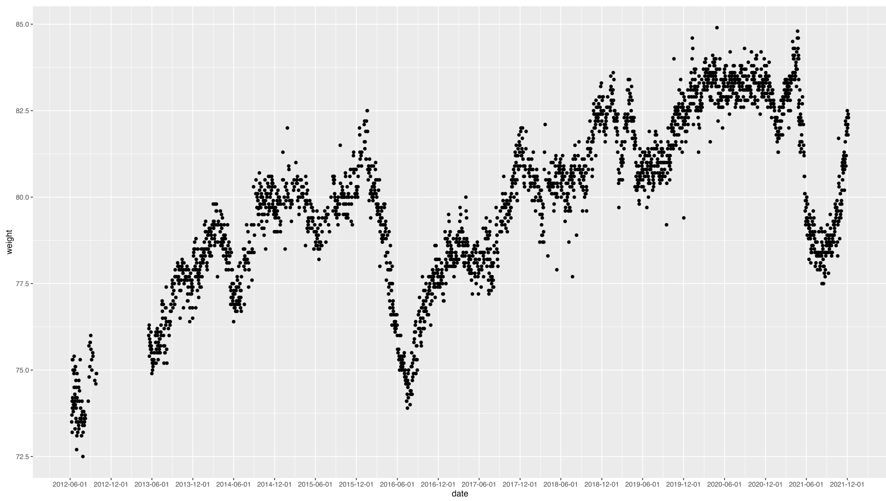


図 29.1　体重の時系列的推移

[*1] 体の内側の水分を出し，表皮に水分をつけないようにすることで，しっかりと肉の量を測ろうとしているのでこの順番になります。

[*2] ついでにいうと，書き込んだ結果はツイートします。フォロワーの中には私の体重の推移を楽しみにしている人がいるのです。おかしな世の中です。

29.1 混合分布モデルの応用

データが 10 年分というのはちょっと多いですから，少し時期を区切ることにします。2019 年から 2020 年の 2 年間に限定したデータが，図 29.2 になります。

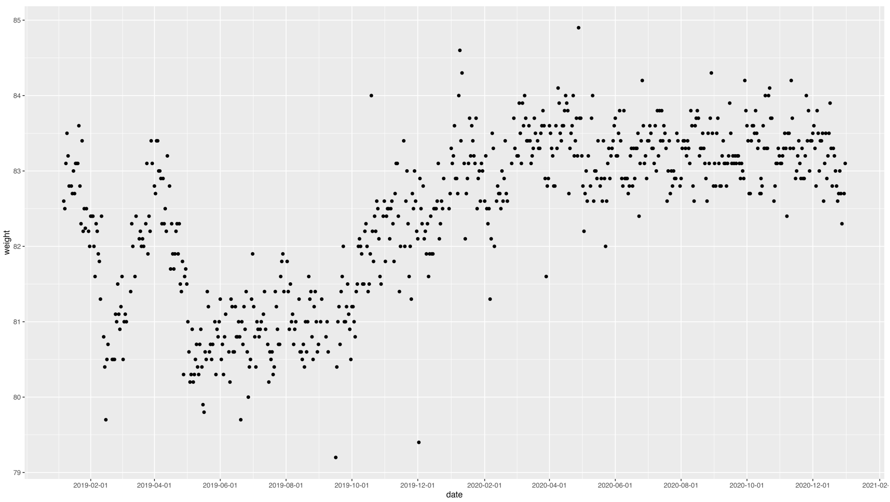


図 29.2 2019-2020 年のデータ

これをみると，うーん残念なことに，2020 年になると体重が増えてしまっているようですね。2019 年は 81kg 台をうろうろしていたようですが，2020 年になると 83kg 台になっているようです。毎回の体重の計測に偶然の測定誤差がついているとしても，19 年と 20 年とではそもそも体重が違ってしまっているようです。これは平均値の違う正規分布が混合している，混合分布モデルを考えてみることができそうですね。

第 27 講を参考に，混合分布モデルを考えてみましょう。筆者の体重には 2 つの状態があり，軽い方の状態なのか重い方の状態なのかは，確率 θ で変わると考えてみます。するとデータは，確率 θ で μ_1 を平均とする正規分布から，確率 $1-\theta$ で μ_2 を平均とする正規分布から得られるわけですから，確率モデルは次のようになります。

$$p(W_j) = \theta_j \times N(\mu_1, \sigma) + (1-\theta_j) \times N(\mu_2, \sigma)$$

ここで θ_j としたのは，観測時点 j ごとに θ が変わるという想定です。どちらのモードになるのかは一定の数字というより，毎回違っているように思えたのでそのようにしてみました。されてこを Stan で実装するにはどうすれば良いのでしょうか。そう，`log_sum_exp` 関数を使って，2 つの状態をベクトルで表記し，それを足し合わせる必要があるんでしたね。実際にコードにしてみたのがコード 29.1 です。

code : 29.1 2 つの体重モードモデル

```
data{
  int L;
  array[L] real W;
}

parameters{
  array[L] real<lower=0,upper=1> theta;
```

```
  ordered[2] mu;
  real<lower=0> sigma;
}

model{
  for(l in 1:L){
    target += log_sum_exp(
      log(theta[l]) + normal_lpdf(W[l]|mu[1],sigma),
      log1m(theta[l]) + normal_lpdf(W[l]|mu[2],sigma)
    );
  }

  mu ~ normal(80,10);
  sigma ~ cauchy(0,5);
}
```

これを使って，体重のデータを分析してみましょう。コード 29.2 を実行し，図 29.3 のような結果を得ます。

code : 29.2　混合分布モデルのコード

```
dat1 <- dat %>%
  dplyr::filter(date > "2019/01/01") %>%
  dplyr::filter(date < "2021/01/01")

model <- cmdstanr::cmdstan_model("changePoint1.stan")
dataSet <- list(L = NROW(dat1), W = dat1$weight)
fit <- model$sample(
  data = dataSet,
  chains = 4,
  parallel_chains = 4,
  seed = 12345)
```

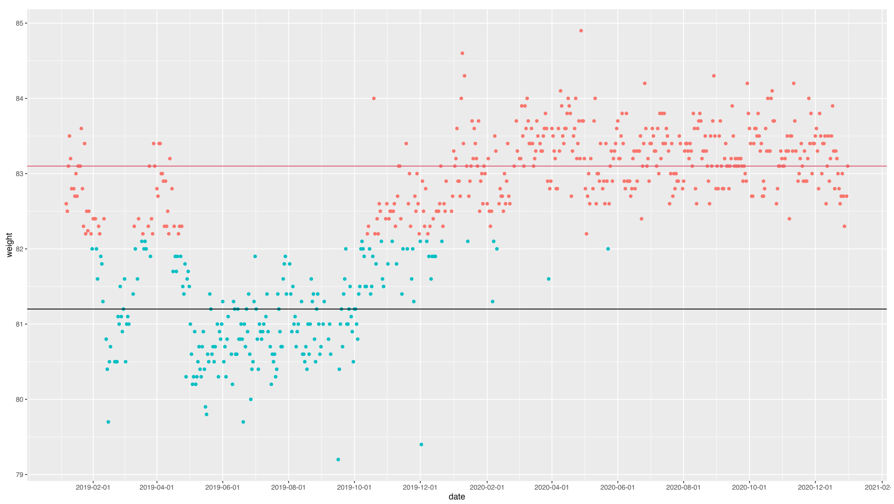


図 29.3　2 つの体重のモード

結果の図を見ると，どうやら 2019 年の初期にも 2 回ぐらい平均値が高い，「重いモード」が混在してお

り[*3]，2019 年の 10 月ごろからそちらの比率が増えています。もっとも，2020 年の 1 月にも「軽いモード」に入っている点はあるようですね。

とまあ，このような分析結果になったわけですが，これをみるとデータが 82kg ぐらいのラインを超えたかどうかで分割されているな，というのがわかります。当然，平均値の違う 2 つの分布を混ぜたわけですから，平均値の違いによって分かれるわけです。それにしても 2019 年の初期はどうしたんでしょうね。1 月や 4 月は重いモード，2-3 月と 5 月以降は軽いモードと，コロコロ入れ替わっています。ここでは 2 つのモードのどちらからデータが出てきているのか，ということだけを考えているのでこれでいいのですが，体重というのはそもそも時系列的な変化をするものですから，どこかで重くなった，どこかで軽くなった，というような時系列的なつながりについての情報が，うまくモデル化されていないように思えます。そこで，横軸が時間的な連続であるということに注意して，今度は 2021 年 1 月から 11 月までのデータを見てみましょう。その区間を取り出したのが図 29.4 になります。

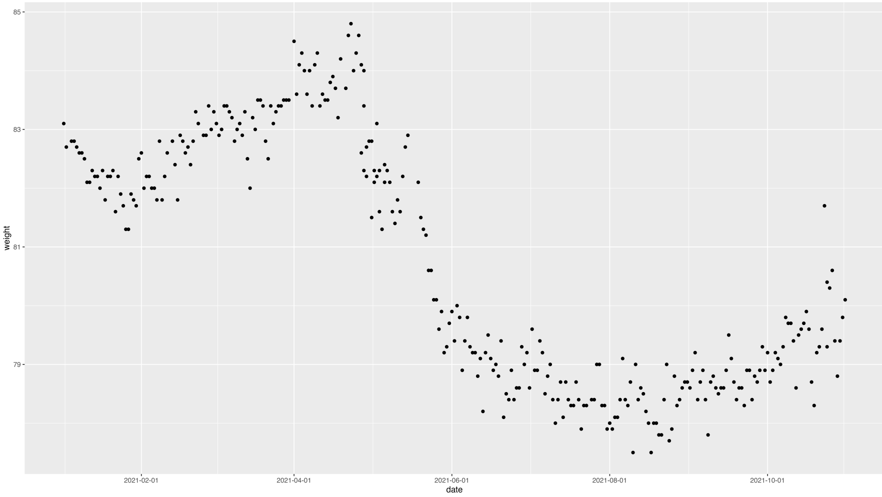


図 29.4　大きく変化したデータ

これは 2 つのモードがあるというより，途中で大きく変化したというべきではないでしょうか。とくに 2021 年 5 月ごろから 4kg ほど体重がぐぐぐっと下がることがあり，6 月以降は下がった体重のレベルに落ち着いているようです。このように，横軸が時系列だと考えると，5 月に何か変化があったのではないか，ということが推察されます。これをモデルで表現してみましょう。

29.2　変化点検出

何かのきっかけでデータの様相がガラリと変わってしまった，その変化したところを変化点と呼び，ここでの課題はその変化点を見つけ出す**変化点検出 (Change point detection)** ということになります。たとえばセンサーが検出するデータの針が急に違うレベルに変化すると，測定している対象の状態がガラリと変わったのではないか，と考えることができますね。

他にもたとえば，心理学の応用領域として，科学捜査研究所が担当する**ポリグラフ検査**というのがあります[*4]が，これなども針が大きく触れたことで変化をみることになります。

[*3] 誰がデブモードやねん

[*4] 皮膚電気活動や心拍，呼吸など複数の整理指標を同時に測定するので**ポリ**グラフであり，熟練の検査官が反応パターンから被疑者の特別な反応を検出するものです。嘘発見器と呼ばれることがありますが，正確には**虚偽検出**と読んだほうが良いでしょう。

変化点検出はその名の通り「いつ」変化したのかを検出するものです。今回のデータも，だいたい 5 月ごろに大きな変化があったというのはわかるのですが，いつなのかを特定するのは難しいところです*5。これをデータとモデルから明らかにしようというのです。

ベイズ統計はわからないことを確率で表現し，データでその確率情報をアップデートしていくというものです。今回はいつ変化したのがわかりませんから，その変化した時期を τ とし，データ区間の中にその日があると考えて，その区間の一様分布を事前分布とします。その時期 τ を境に，データが出てくる分布の位置パラメータがずれると考えるのです。設計図は次のようになります (図 29.5)

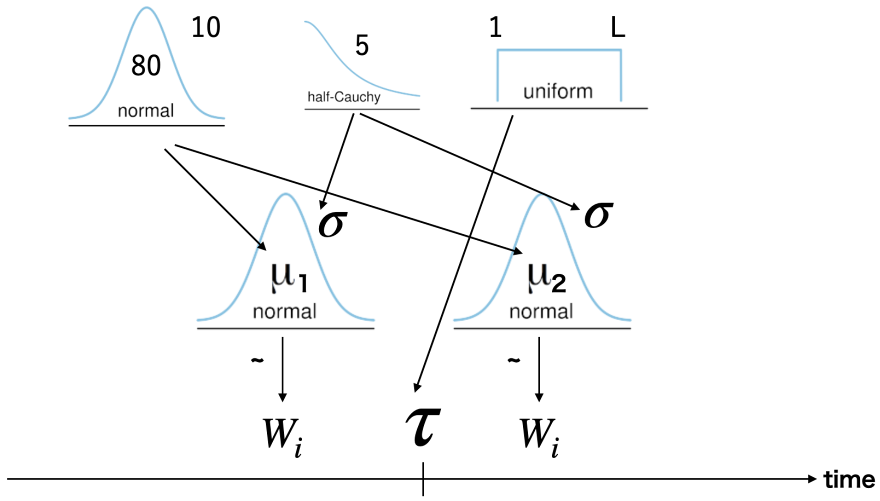


図 29.5　変化点検出のモデル設計図

設計図をもとに，コードにしてみたのがコード 29.3 になります。

code : 29.3　変化点検出のモデルコード

```
data{
  int L;
  array[L] real W;
}

parameters{
  real<lower=1,upper=L> tau;
  ordered[2] mu;
  real<lower=0> sigma;
}

model{
  for(l in 1:L){
    if(l < tau){
      W[l] ~ normal(mu[2],sigma);
```

最近では**隠匿情報検査 (concealed information test)** ということもあります。ちなみに科学捜査研究所，通称科捜研は，心理の他にも物理，化学，法医，文書 (筆跡鑑定) などの専門領域があり，各都道府県に 1 つずつ設置されています。

*5 とはいえこのデータは筆者自身の体重変化ですので，何があったのかは実はわかっています。5 月 11 日，筆者が帰宅途中に自転車で転倒する事故を起こし，右肩の鎖骨を骨折しました。17 日に手術した後，利き手が使いにくいものですから食事の量が減り，結果的に体重 (筋肉?) が落ちたというのが真相です。

```
      }else{
        W[l] ~ normal(mu[1],sigma);
      }
    }

    tau ~ uniform(1,L);
    mu ~ normal(80,10);
    sigma ~ cauchy(0,5);
  }
```

日々のデータが順に 1 から L 行目まで並んでいるとします。パラメータ tau があり，1 行目のデータが変化点 tau より前にあるときは μ_2, σ の正規分布から，tau より後になれば，μ_1, σ の正規分布からデータが出てくると考えるのです。今回は後半の平均値が小さいことが明らかですから，μ のベクトルを ordered 型で宣言してあります。ベクトル要素の小さい順に並びますから，前半が μ_2，後半が μ_1 としています。

これを実行して，変化点がいつになるのかを推定してみましょう。推定結果は出力 13 のようになりました。

MCMC の結果 13

```
# A tibble: 4 × 7
  name          EAP        MED        MAP          SD        L95        U95
  <chr> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 mu[1]    82.770     82.769     82.762      0.062     82.648     82.893
2 mu[2]    78.891     78.892     78.896      0.060     78.775     79.008
3 sigma     0.750      0.749      0.747      0.031      0.693      0.812
4 tau     144.642    144.578    144.431      0.646    143.321    146.190
```

データの 145 行目がちょうどその変化点だといって，ほぼ間違いないようですね。145 行目はといいますと，データから 5 月 23 日だったことがわかります。

R の出力 29.1: いつでしょう

```
> dat2[145, ]
# A tibble: 1 × 3
  date                    weight bodyFat
  <dttm>                   <dbl>    <dbl>
1 2021-05-23 06:50:57       80.6       26
```

違いがわかるようにデータとモデルの推定値を合わせたのが，図 29.6 です。このようにして，ここで様相が変わったのだなということを，データから見出すことができました。

29.3　折線回帰

先ほどは，変化点を機に平均値が変わる，というモデルでした。では次のような分布の場合はどうなるでしょうか。今度は 2016 年に注目してみました (図 29.7)。この年はダイエットを志した時期で，最初の半年ぐらいでぐんぐん体重が落ちていってるのがわかります。そして残念なことに，8 月ごろでしょうか，ダイエットが終わりリバウンドが始まったのがよくみて取れますね[*6]。

[*6] このときのダイエット法は，1. ラーメンのスープは飲まない，2. 夕食と晩酌は分ける (食べながら飲むのではなく，食べ終わって，お酒だけ飲む),3. お酒を飲むときにおつまみは食べない，という 3 か条を守るというものでした。お酒が飲みたいので食事の量

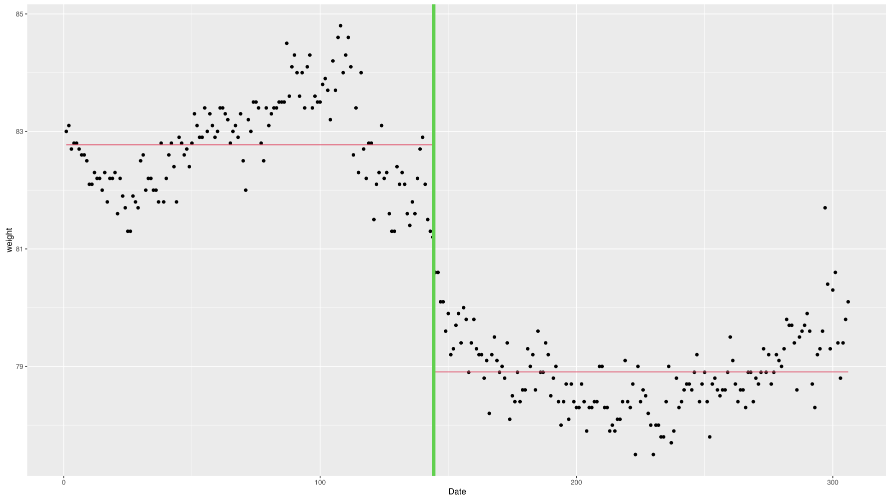


図 29.6　検出された変化点と平均値の違い

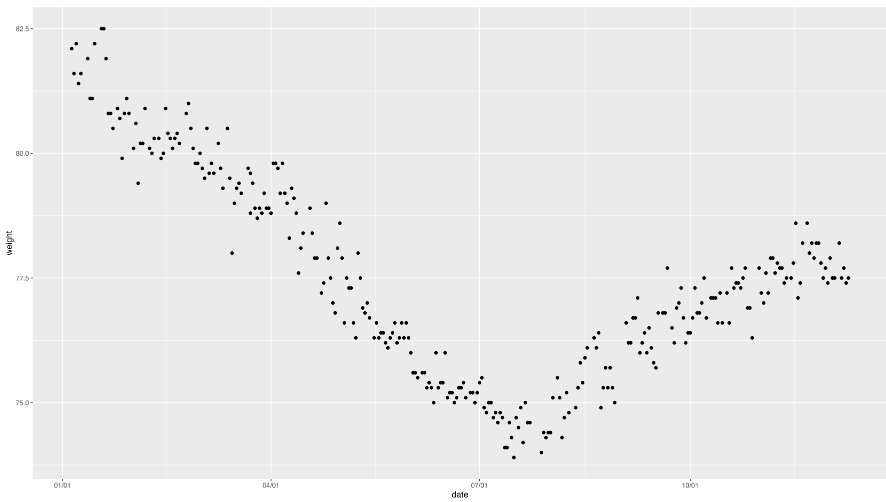


図 29.7　2016 年の変化

さてこのデータに対して，先ほどの変化点検出もできるのですが，変化点の前後で平均値が違う，というような簡単な話ではなさそうです。変化点の前はどんどん数字が減っていき，変化点のあとはどんどん数字が大きくなっていく，というのが実態です。回帰分析をしたら，変化点前は負の傾きが，変化点後は正の傾きが推定されそうな，そんなデータになっていますね。ということで，そのイメージができたのであれば，そのままモデルを描いてしまいましょう。設計図が少しごちゃっとしてしまいましたが，ポイントは傾きの係数を前半は負，後半は正に限定しているところでしょうか (図 29.8)。また，変化点は大体このあたり･･･ということでデータの 100 行目 (2016/04/29) から 250 行目 (2016/10/15) の間に置いてみました。

設計図をもとに，コードにしてみたのがコード 29.4 になります。

code：29.4　折線回帰のモデルコード

を減らし，すぐに飲むことにしたので体重が減り，食事の前に飲んでしまえばいいんじゃないかという裏技を見つけてダイエットが崩壊したのを覚えています。

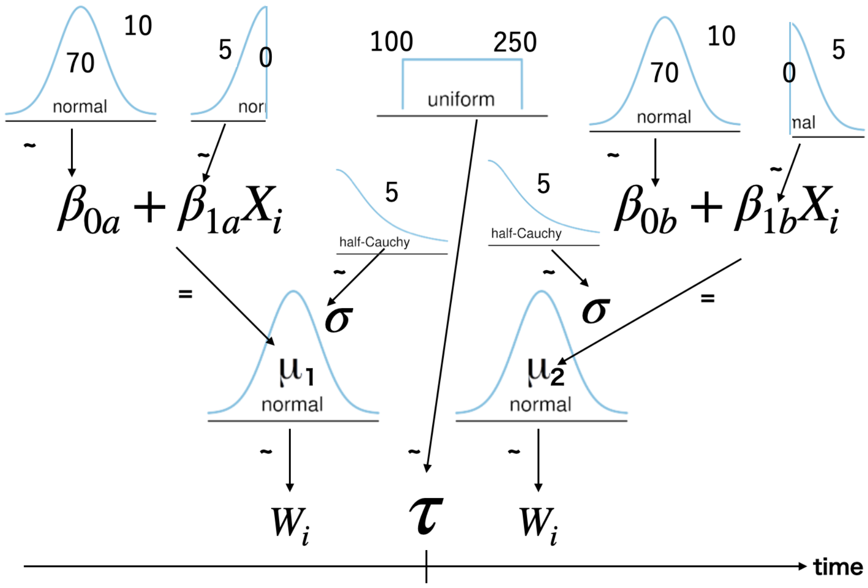


図 29.8 折線回帰モデル設計図

```
data{
  int L; // data length
  array[L] real W;
  array[L] real X;
}

parameters{
  real<lower=100,upper=250> tau;
  array[2] real beta0;
  real<upper=0> beta1a;
  real<lower=0> beta1b;
  real<lower=0> sigma;
}


model{
  for(l in 1:L){
    if( l < tau ){
      W[l] ~ normal( beta0[1] + (beta1a * X[l]),sigma);
    }else{
      W[l] ~ normal( beta0[2] + (beta1b * X[l]),sigma);
    }
  }

  beta0 ~ normal(70,10);
  beta1a ~ normal(0,5);
  beta1b ~ normal(0,5);
  sigma ~ cauchy(0,5);
}
```

回帰係数を正･負に限定するところは，パラメータの宣言で<lower=0>や<upper=0>としていることで表現しています。あとはデータが出てくる正規分布の平均に，線形モデルが入っているだけですね。これを実行すると，出力 14 のような結果が得られます。データの 187 行目 (2016/08/01) あたりが変化点ですかね。95% 区間で言うと 185 行目 (2016/07/30) から 188 行目 (2016/08/02) の間に変化点があると言えそうです。

MCMC の結果 14

```
# A tibble: 6 × 7
  name          EAP       MED       MAP        SD       L95       U95
  <chr>    <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!> <num:.3!>
1 beta0[1]   81.871    81.872    81.874     0.075    81.721    82.014
2 beta0[2]   70.548    70.551    70.549     0.380    69.793    71.274
3 beta1a     -0.043    -0.043    -0.043     0.001    -0.044    -0.041
4 beta1b      0.026     0.026     0.026     0.002     0.023     0.029
5 sigma       0.499     0.498     0.497     0.021     0.461     0.543
6 tau       187.135   187.326   187.460     0.832   185.057   188.215
```

そして回帰係数をプロットしてみました (図 29.9。なかなかデータにフィットしていそうです。

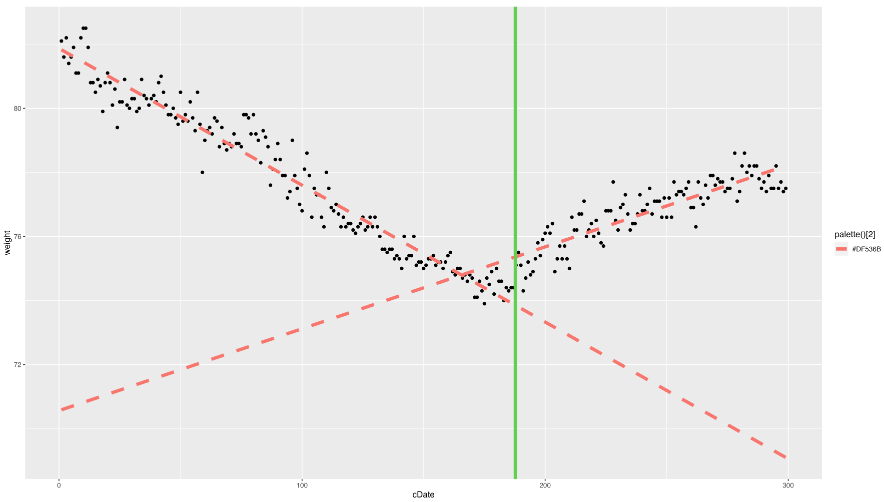


図 29.9　折線回帰モデル推定結果

このままでもよいのですが，変化点と折れたポイントが合致しないのはなんだか気持ち悪いですね。これを合わせることを考えてみたいと思います。変化点 τ のあるところで 2 つの回帰線は交わる，つまり同じ値になるはずですから，$\beta_{0a} + \beta_{1a}\tau = \beta_{0b} + \beta_{1b}\tau$ という式が成り立つはずです。ここから逆算して，

$$\beta_{0a} + \beta_{1a}\tau - \beta_{1b}\tau = \beta_{0b}$$

と考えることができますから，推定するパラメータを 1 つ減らすことができます。

code : 29.5　折線回帰のモデルコード 2

```
...(前略)...
parameters{
  real<lower=100,upper=250> tau;
```

```
  real beta0a;
  real<upper=0> beta1a;
  real<lower=0> beta1b;
  real<lower=0> sigma;
}

transformed parameters{
  real beta0b;
  beta0b = beta0a + ((beta1a-beta1b) * tau);
}
...(後略)...
```

このコード 29.4 にあるように，beta0b を計算式から出すようにして，改めて推定をした結果が図 29.10 になります。これだと変化点が前にずれて，データの 169 行目 (2016/07/10) で心がポッキリ折れていることがわかります。

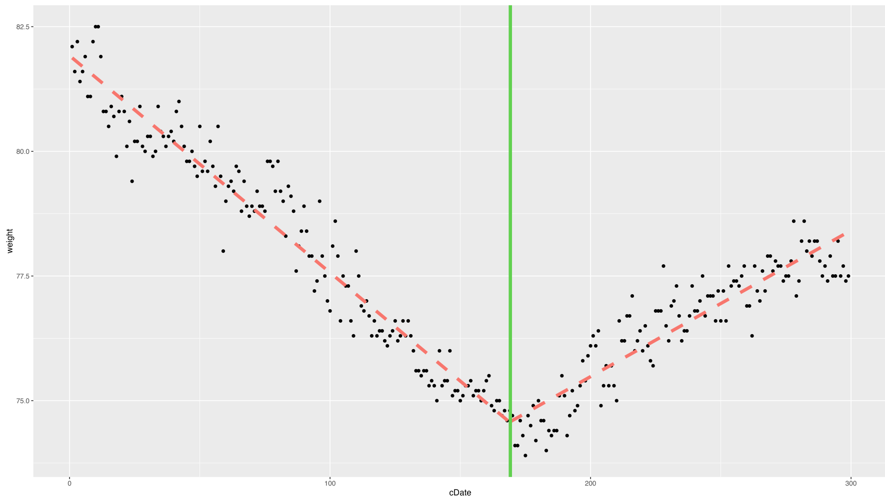


図 29.10　変化点を合わせた折線回帰モデル

このように，モデリングの力を借りるとわからなかった変化点がどこにあるのか，そしてその変化点の前後で予測モデルを変えるといったようなことが簡単に表現できます。ただの回帰分析であっても，データにあったモデルを考えることでさまざまな応用可能性が出てくるのではないでしょうか。

ところで，今回は時系列的なデータに対して回帰分析を行いました。この時の説明変数 X_i は日付，あるいは「何日目か」というデータだったわけです。しかし回帰分析には重要な仮定として，各データ点は (同じ分布から) 独立に得られていると言うものがありました。たとえば身長と体重の回帰分析，という話をするときに，各データ点にあたる個々人の身長･体重は，互いに影響し合わない独立だったはずであり，だからこそ尤度の計算の時に各データの尤度を掛け合わせていくことができたのでした。それに対し，体重のデータは明らかに違います。すなわち，今日の体重が明日の体重に影響している，それどころか今日の体重は明日の体重と大いに関わっているはずですし，明日明後日とその後の日々にも少なからず影響しているはずなのです。

このように考えると，独立していないデータ点，時系列的なデータに対しては異なる分析をする必要がありそうです。次回はこの時系列的なデータを扱うモデルを紹介していくことになります。

29.4 課題

2015 年の 1 月 1 日から 2015 年の 12 月 1 日の間にも，体重の減少と増加が切り替わったところがありそうです。変化点を合わせた折線回帰を実行し，何月何日に変化点があったのか推定するモデルを分析する R/Stan コードを提出してください。結果の解釈などを，スクリプトのコメントアウトや別添ファイルなどで提供してもらえるとなお良いです。もちろん Rmd ファイルでの提出であれば完璧です。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

第 30 章

確率的プログラミング；状態空間モデル

前回は体重の変化データを例に，データの平均点が変わるモデル，変化点を検出したり変化点を境に傾きが変わる線形モデルをみてきました。

同じように，今回も時系列的なデータを扱います。前回の最後に案内したように，時系列データに普通の回帰直線をそのまま当てはめることは適切ではありません。どのような注意点が必要なのかについて，少し考えてみましょう。

30.1　時系列データの特徴

時系列的なデータはどのようなときに得られるでしょうか。前回導入した時のように，誰か 1 人が日々の生活を記録し続けている，それも立派な時系列データです。心理学には**日誌法**と呼ばれるデータ収集法があります。一言で行ってしまえば日記なのですが，日々多くの出来事や感情が到来するのを後から振り返って考えるのでは，正しく思い出せなかったり記憶が歪んでしまうこともあります。日々の状態を細かく記録し積み重ねることは，数字になっていなくても貴重なデータなのです。また最近は，**経験サンプリング**と呼ばれる調査法もあります。これは決まった間隔で質問紙がスマートフォンなどに送られてきて，定期的に心情を心理尺度に反映させていく，というものです。こうしたことができるようになったのには，私たちの周りに電子的なデバイスがたくさんあることが理由の 1 つです。さらに最近ではウェアラブル端末といって，身につける電子端末がありますね。Apple Watch など携帯電話と同じような機能を持ったこれらのデバイスは，GPS 機能がついていたり，万歩計による歩数の計測，血圧・脈拍のリアルタイムな計測が行われていたりします[*1]。これを蓄積したデータとすると，かなりの時系列的なデータが手に入ることになります。

あるいはまた，社会的なデータも時系列的に得られるものが多いです。一番わかりやすいのは株価の指標などでしょうか。日経平均株価など，時事刻々と変化する株価がグラフになって表されているのをみなさんもみたことがあると思います。また，Twitter などの SNS で，今どのようなキーワードが使われているかといったことが，リアルタイムに計測されます。これらの指標は個々人の特徴というより，社会全体のうねりのようなものを表現していることになりますが，これを研究することも立派な社会心理学的テーマになり得ます。時系列データは，人文社会科学領域ではこれまで経済学やマーケティングなどが専門的に扱ってきたところがありますが，これからは心理学の領域でもこうしたデータを分析することが流行してくるかもしれません[*2]。

さて時系列的なデータはどういった特徴があるかと言うと，端的に言えば**自己相関 (auto correlation)** がある，ということでしょう。すなわち，ある時点のデータはその前の時点と相関している，ということです。体重のデータの場合でもそうですが，明日の体重は突然ランダムな値になるのではなく，今日の体重に応じ

[*1] こうしたデータは**ライフログ (Life-log)** と呼ばれることもあります。

[*2] もちろん生理心理学や動物心理学では，古くからこうしたデータを扱ってきています。

て変化するはずだからです。また，データの変動に周期的なパターンが存在することもあります。脳波などのデータは基本的に波ですから，大きな波，小さな波がパターンを描いています。株価など社会的なデータも，季節による一定の変動など全体的な揺らぎがある上で，目的とする意味のある変化を見つけ出さなければならないという課題に取り組んでいます。たとえば周波数のデータの平均はゼロになりますから，基本的な記述統計では太刀打ちできないところがあります。回帰分析もその前提として，各データが独立に得られていると言うものがありますから，自己相関するデータは当然この仮定に違反していることになります。こうしたデータを分析するためには，周波数の大きさをスペクトル解析するとか，多次元の行列であるテンソルを使ってさまざまな要素を分解する，といった特殊な応用数学を駆使する必要があります。

今回はこうしたさまざまなモデルの中でも，**状態空間モデル (State Space Model)** を紹介します。

30.2 状態空間モデル

時系列的なデータは，自己相関すなわち前の時点の状態が，次の時点に影響しているという特徴があり，これをうまく表現するのが状態空間モデルです。状態空間モデルはこれまでの**モデリング**の技法を使うと簡単に実装できます。というか，これまでの時系列的なデータ解析は，自己相関を減らすために一時点前のデータとの差分をとってそれをモデルにする，時点ごとの変化をスムージングするなど，さまざまな工夫の上に成り立っていました。状態空間モデルはそれを非常にすっきり表現してくれているので，それまでの分析ノウハウがなくとも誰でも簡単に利用できるモデルだといえるかもしれません。

状態空間モデルは，まず観測されたデータと，その背後にある**状態**を区別します。体重変化のデータの例で考えてみましょう。計測された体重は，当然体の重さを反映したものですが，測定の時に多少の誤差が生じるでしょう。例に使っている筆者のデータでも，毎朝全裸で計測しているわけではありませんから，季節の変わり目で薄いパジャマからジャージにしたり，冬用の厚手のパジャマに変えたりすると，それだけで計測された数字には違いができます。本来知りたいのは，体の重さ，肉の重さそのものなはずですね。ここでいう体重そのもの，計測誤差のない肉の重さが，ここでいう状態のことになります。

そして t 時点の状態 μ_t は，次の時点の状態に影響します。服の重さなどを取り除いた t 時点の肉の状態に，食事や運動といった変化が加わって明日 $t+1$ 時点の肉の状態になるわけですから，$\mu_{t+1} = f(\mu_t)$ と考えることができるわけです。

もし翌日の体重が，今日の体重にちょっとした誤差がつくような変動しかしない，と言うのであれば $\mu_{t+1} \sim N(\mu_t, \tau)$ と表すことができるでしょう。あるいは，運動量 P_t の関数だというのであれば，$\mu_{t+1} \sim N(\mu_t + \beta_1 P_t, \tau)$ のように回帰分析の確率モデルのように表現すれば良いでしょう。もちろん説明変数が増えたり，一次関数ではないものを考えても構いません。

一方，計測値 W_t は，この μ_t に誤差 σ がついて得られる，すなわち $W_t \sim N(\mu_t, \sigma)$ という関係になります。これらの関係を表現したのが，図 30.1 になります。このようにして，状態が時間 $t, t+1, t+2, \cdots$ を通じてつながっており，状態に応じて観測値が得られると考えます。観測値を手に入れる際には偶然誤差 σ が生じますし，状態の変化は確率的に変わるのであれば幅 τ で，系統的に変わるのであればそれをモデリングしてやれば良いことになります。

この発想に基づいて，設計図を書いてみましょう。ここでは体重の変化に特別な傾向を考えず，ただ幅 τ で確率的にノイズが加わる**ホワイトノイズモデル (white-noise model)** で考えてみることにします。少しイメージしにくいかもしれませんが，たとえば次のように表現できます (図 30.2)。

それではこれをコードにしてみましょう。少し長くなりましたが，ホワイトノイズのモデルはコード 30.1 のようになります。

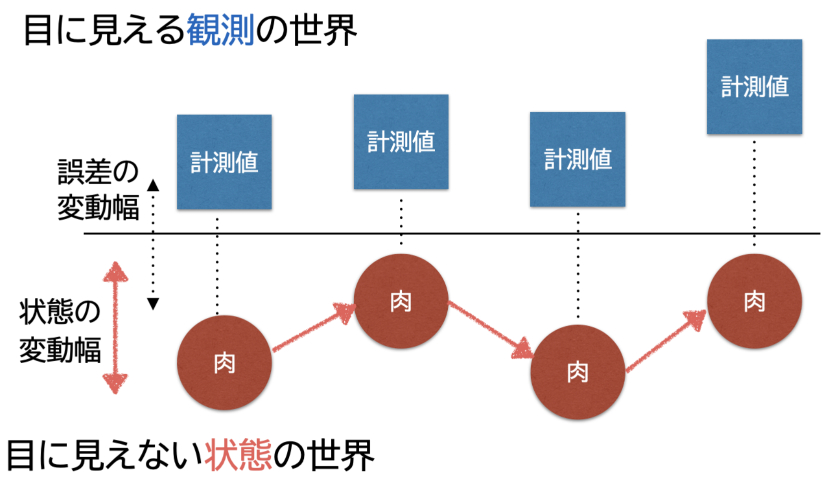


図 30.1 状態空間モデルのイメージ (体重変化の例)

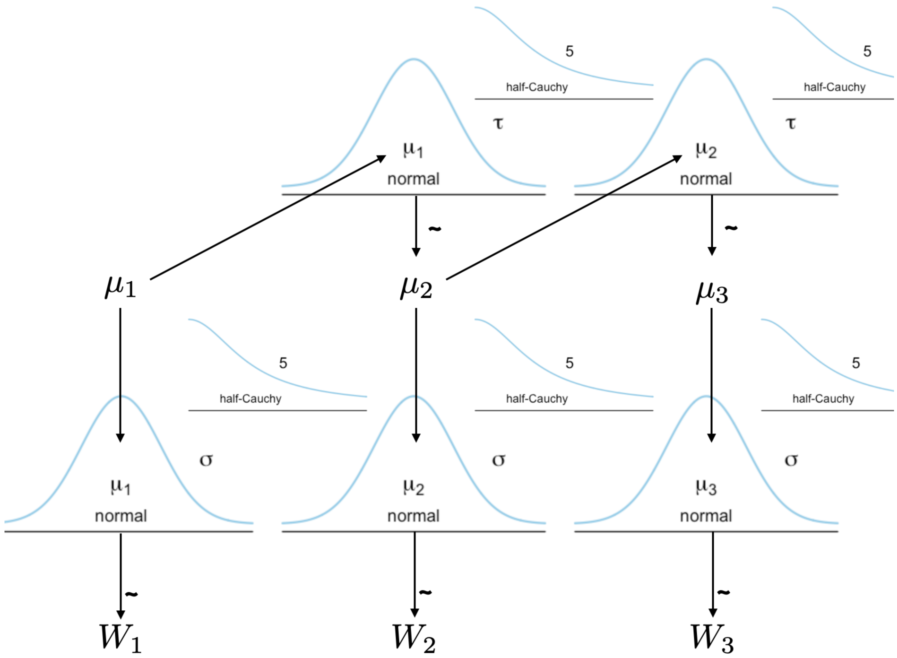


図 30.2 状態空間モデルの設計図

code : 30.1 ホワイトノイズモデル

```
data{
 int L;
 array[L] real W;
}

parameters{
  real muZero;
  array[L] real mu;
```

```
  real<lower=0> sig;
  real<lower=0> tau;
}

model{
  mu[1] ~ normal(muZero, tau);

  for(l in 1:L){
    W[l] ~ normal(mu[l],sig);
  }

  for(i in 2:L){
    mu[i] ~ normal(mu[i-1],tau);
  }

  muZero ~ normal(80,10);
  sig ~ cauchy(0,5);
  tau ~ cauchy(0,5);
}
```

■コード解説

data ブロック　データ長 L, 各時点の体重 W がデータになります。

parameters ブロック　データ長と同じだけの状態を表す mu と, 状態の変動幅 tau, 状態に付加される計測誤差の変動幅 sig を宣言しています。それに加えて, 最初の状態 μ_0 をパラメータとしました。これは 2 時点目の状態 μ_2 は $\mu_2 \sim N(\mu_1, \tau)$ で，3 時点目は $\mu_3 \sim N(\mu_2, \tau)$ で･･･と表現できるのに対し, 最初の点は $\mu_1 \sim N(\mu_0, \tau)$ となって, 計測される前の状態を考えなければならないからです。わからないものは確率で表現する, というベイズの流儀に則って, これはパラメータとして推定してやることにします。

model ブロック　まず 1 時点目の状態は，0 時点目の状態から出てくるものですので，$\mu_1 \sim N(\mu_0, \tau)$ をモデル化しました。次に計測された値 W_l は状態 μ_l から出てくるものですから，1 時点目からデータ長 L まで順に $W_l \sim N(\mu_l, \tau)$ として尤度を書きます。次に状態のモデルですが, 2 時点目以降は前の時点からの影響で表現できるので，$\mu_i \sim N(\mu_{i-1}, \tau)$ としています。最後に事前分布として, これまでの経験から $\mu_0 \sim N(80, 10)$ とし, 変化の幅はいずれも SD ですから半コーシー分布で表現しました。

これを使って, 体重のデータを分析してみましょう。データは 2020 年からのものを使います。図 30.3 をみると，2021 年まではあまり変化がなく，21 年初頭に少し体重が下がって, また上がって, その後 5 月から謎の下降[*3], そしてその反動[*4], とおそらく線形モデルでは表現できないような複雑な動きをしています。

さて, コード 30.2 を実行し, 図 30.4 のような結果を得ます。複雑な動きについても, モデルがしっかりと予測しているのがみて取れますね。状態空間のパワフルな表現力をお楽しみいただけましたでしょうか。

code : 30.2　状態空間モデルのコード

```
dat1 <- dat %>%
  filter(date > "2020/01/01") %>%
```

[*3] 自転車事故による鎖骨骨折と, それに伴ってお箸持ちにくくて食事量が減る, 運動も減るといったのが原因でしょう。
[*4] 悔しいです。

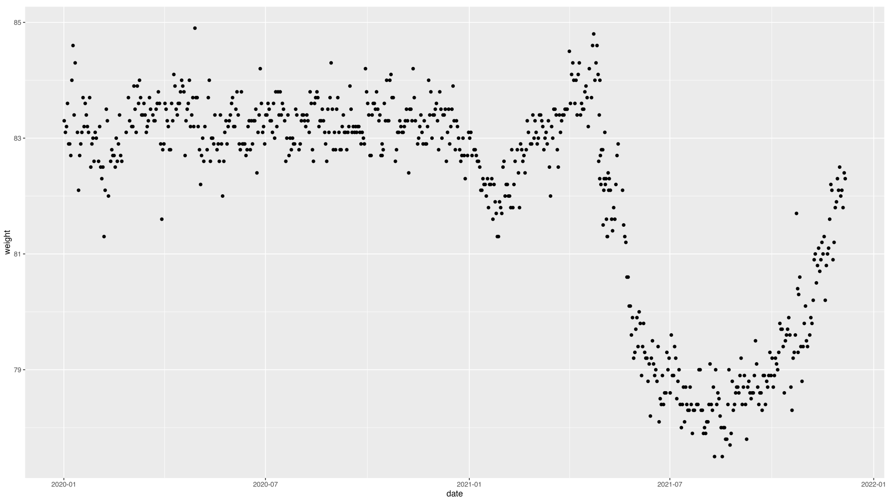


図 30.3　2020 年の体重変化

```
  mutate(date = as.Date(date))

model <- cmdstan_model("StateSpace.stan")
dataSet <- list(L = NROW(dat1), W = dat1$weight)
fit1 <- model$sample(
  data = dataSet,
  chains = 4,
  parallel_chains = 4
)

fit1.stanfit <- fit1$output_files() %>% rstan::read_stan_csv()
fit1.df <- fit1.stanfit %>% as.data.frame() %>%
  as_tibble() %>%
  rowid_to_column("iter") %>%
  pivot_longer(-iter, names_to = "Varname") %>%
  group_by(Varname) %>%
  summarise(
    EAP = mean(value),
    MED = median(value),
    MAP = map_estimation(value),
    SD = sd(value),
    L95 = quantile(value, probs = 0.025),
    L50 = quantile(value, probs = 0.25),
    U50 = quantile(value, probs = 0.75),
    U95 = quantile(value, probs = 0.975)
  )

Est1 <- fit1.df %>%
  dplyr::filter(str_detect(Varname, "mu")) %>%
  dplyr::mutate(ID = str_extract(Varname, pattern = "\\d+") %>%
  as.numeric()) %>%
  arrange(ID)

```

```
36  g <- dat1 %>%
37    rowid_to_column("ID") %>%
38    left_join(Est1, by = "ID") %>%
39    ggplot(aes(x = ID, y = weight, ymin = U95, ymax = L95)) +
40    geom_point() +
41    geom_point(aes(x = ID, y = MAP), color = palette()[2]) +
42    geom_ribbon(fill = palette()[3], alpha = 0.2)
43  plot(g)
```

■コード解説

1-3 行目　データファイルから該当する日程だけ抜き出します。日付を Date 形式に変換しています。

5-11 行目　cmdstanr でコンパイルしてサンプリングするところです。

13 行目　cmdstanr で作ったファイルを stanfit オブジェクトに変換しています。rstan パッケージを使っている人はこの行を実行する必要がありません。

14-28 行目　MCMC サンプルを集計し，**EAP**，**MED**，**MAP** 推定値，SD, 確信区間などを出しています。

30-34 行目　推定された状態 μ_i を取り出しています。

36-43 行目　もとのデータと推定値を合体させ，プロットしています。

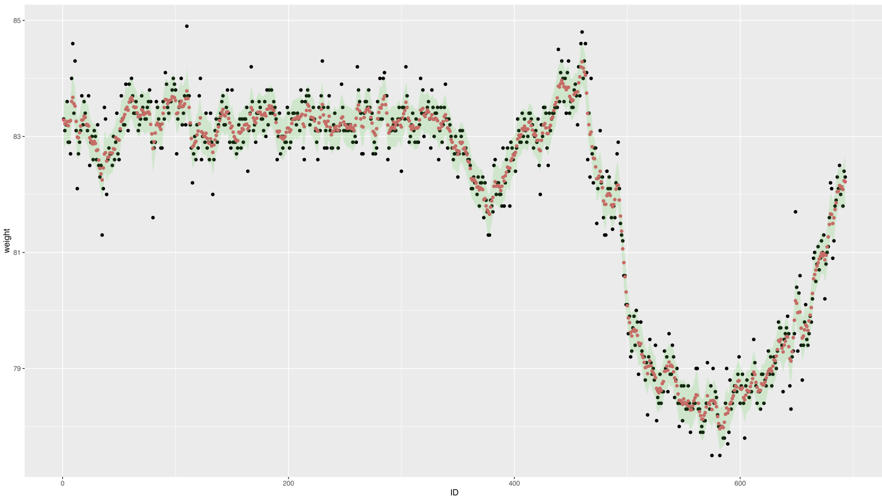


図 30.4　2020 年の体重変化をモデルで追跡してみた結果

30.3　欠損値の補間

ところで，データをよくみてみると，実は毎日のデータになっていないところがあることに気づきます。いや，気づかないかもしれないですね。次のコード 30.3 を実行してみてください。

code : 30.3　日付は連続かな?

```
1  dat1 %>%
2    mutate(lag = lag(date)) %>%
3    mutate(date = as.Date(date), lag = as.Date(lag)) %>%
```

```
4    mutate(FLG = date - lag) %>%
5    dplyr::filter(FLG > 1)
```

■コード解説

1 行目　先ほどの推定に使ったデータファイルです。
2 行目　`lag` 関数で，データを 1 行ずらした列を作ります。
3 行目　体重を記録した日の変数 `date` と，先ほど作った一行ずらした変数 `lag` はいずれも日付に関する変数ですので，日付型に変更します。
4 行目　日付変数の引き算をして，一行下のデータと何日ずれていたのか算出しています。
5 行目　1 日以上ずれている日をフィルタリングで抜き出しています。

これをみると，ちょくちょく抜けていて，時には 4 日も空いていたりすることがわかります[*5]。ともかく，これでは正確にデータを分析できているとは言えません。

R の出力 30.1: 抜けている日

```
     # A tibble: 21 × 5
     date       weight bodyFat lag        FLG
     <date>      <dbl>   <dbl> <date>     <drtn>
   1 2020-01-13   83.1    26.8 2020-01-11 2 days
   2 2020-02-01   82.6    26.7 2020-01-30 2 days
   3 2020-02-12   82.6    26.7 2020-02-10 2 days
   4 2020-02-26   83.1    26.8 2020-02-22 4 days
   5 2020-02-28   83.7    27   2020-02-26 2 days
   6 2020-03-02   83.2    26.9 2020-02-29 2 days
   7 2020-03-21   83.4    26.9 2020-03-19 2 days
   8 2020-05-09   82.8    26.8 2020-05-07 2 days
   9 2020-05-18   82.8    26.7 2020-05-16 2 days
  10 2020-07-31   82.8    27.7 2020-07-29 2 days
  # … with 11 more rows
```

しかし計測はできていなくても，出張中にも状態の変化は続きいているはずで，次の計測時点はその日の状態の関数になっているはずです (図 30.5)。

この欠損した状態のデータをどのように考えれば良いでしょうか。実はこのことのヒントはすでに，最初のコードの中に入っています。状態空間モデルの 1 時点目，μ_1 は μ_0 から影響されているはずですが，データから μ_0 はわかりません。しかしわからないことは確率で表現するのがベイジアンのやり方です。今回も，欠損している測定値はわからないものとして確率で表現し，推定してやれば良いのです！このようにして間を埋めることをとくに**補間 (interpolating)** と言います。

補間をするためには，R コードの方でも Stan コードの方でもすこし工夫が必要です。まず R のほうで，データを加工するところから見てみましょう。Stan には欠損値 `NA` を与えることはできませんから，欠損しているところには特別なあり得ない数字，そうですね，999 という数字でも入れておきましょう。

code : 30.4　連続したデータを作り欠損値に借りの値を代入する

[*5] 2020 年 2 月 22 日から 24 日は岡山県にベイズ塾合宿 (出張) でした。コロナがまだ豪華客船の中だけに抑え込まれている時期で，ギリギリ出張できた時期です。このあと大学の卒業式がなくなったり，前期オンライン授業になったりと言う，波乱の 2020 年度が始まったのでした。

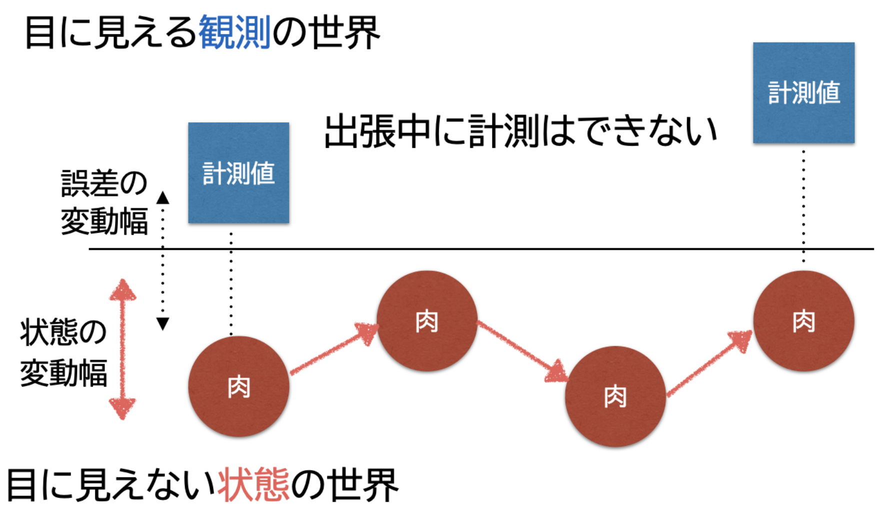


図 30.5　計測できない時点も変化は続く

```
1 fullDays <-
2   data.frame(date = as.Date("2020/01/01"):as.Date("2021/12/01")) %>%
3   mutate(date = as.Date(date, origin = "1970-01-01")) %>%
4   left_join(dat1, by = "date") %>%
5   tidyr::replace_na(list(weight = 999, bodyFat = 999))
```

■コード解説

2 行目　2020 年 1 月 1 日から 2021 年 12 月 1 日までにデータを限定します。この期間，連続した日付を `date` 変数に追加します。

3 行目　日付変数を日付型に変更しています。

4 行目　もとのデータを日付変数に結合しています。`left_join` 関数は左にもとのデータセットを置き，右から `dat1` を引っ付けていきます。結合する際は，変数 `date` をキーにし，キー変数が同じものは同じ行に合わせるという湯やり方です。このやり方だと，該当する行がない場合 (計測データセットの日付がない場合)，もとのデータセットは残して，体重や体脂肪変数は `NA` になります。

5 行目　`replace_na` 関数で，欠損のところに 999 という数字を入れています。

このようにすることで，次のようなデータセットができました (出力 30.2)。

R の出力 30.2: 抜けている日を無くした完全データセット

```
> fullDays
           date weight bodyFat
1   2020-01-01   83.3   26.70
2   2020-01-02   83.1   26.80
3   2020-01-03   83.2   26.90
4   2020-01-04   83.6   27.00
5   2020-01-05   82.9   28.70
6   2020-01-06   82.9   26.80
7   2020-01-07   82.7   26.70
8   2020-01-08   84.0   27.10
9   2020-01-09   84.6   27.30
10  2020-01-10   83.4   26.90
11  2020-01-11   84.3   27.20
12  2020-01-12  999.0  999.00
13  2020-01-13   83.1   26.80
14  2020-01-14   82.1   65.00
15  2020-01-15   82.7   26.70
```

これをみると，2020 年 1 月 12 日は欠損だったのですが，データ上は 999 という数字が入って完全データのように見えます[*6]。このデータを Stan に与えてやり，体重が 999 というあり得ない数字の場合は別の処理をする，という 0 過剰ポアソンの技術を応用します (セクション 27.3，Pp.313)。

code : 30.5 欠損値補間のコード

```
data{
 int L;
 array[L] real W;
 int<lower=0> Nmiss;
}

parameters{
  real muZero;
  array[L] real mu;
  array[Nmiss] real<lower=0> Miss_W;
  real<lower=0> sig;
  real<lower=0> tau;
}

model{
  mu[1] ~ normal(muZero, tau);

  {
    int j = 0;
    for(l in 1:L){
      if( W[l] != 999){
          // こっちは尤度
          W[l] ~ normal(mu[l],sig);
```

[*6] ちなみに 2020/01/11 から 12 にかけては，犬会という研究会があったため関西に出張していました。宿に泊まったので，12 日の朝のルーティンができなかったのです。

```
        }else{
          j = j + 1;
          // こっちはパラメータ
          Miss_W[j] ~ normal(mu[l], sig);
        }
      }
    }

    for(i in 2:L){
      mu[i] ~ normal(mu[i-1],tau);
    }

    muZero ~ normal(80,10);
    sig ~ cauchy(0,5);
    tau ~ cauchy(0,5);
  }
```

data ブロック　データ長 L, 各時点の体重 W がデータです。加えて, 推定すべき欠損値の数もデータとして受け取ります。

parameters ブロック　データ長と同じだけの状態を表す mu と, 状態の変動幅 tau, 状態に付加される計測誤差の変動幅 sig を宣言しています。また最初の状態 μ_0 を表すパラメタと, 欠損の数だけある推定用パラメータ Miss_W を用意しました。

model ブロック　先ほど同様, 0 時点目の状態から出てくるものですので, $\mu_1 \sim N(\mu_0, \tau)$ をモデル化しています。次に各回のデータについて, W_l が 999 でなければ (同じでない, という論理式は!=で表現します), 普通の状態空間モデルのように尤度として計算します。そうではない, すなわちデータ $W_l = 999$ である, つまり欠損値であるはずということなら, $W_{miss} \sim N(\mu_l, \sigma)$ で推定してやります。ここで特殊な変数 j が出てきています。これは「何番目の欠損値か」を表す変数です。ブロックの中の一部だけで変数を宣言するため, まず for 文全体を中括弧で括り, はじめに $j = 0$ という数字を宣言しています。ここで欠損値推定のシーンになると, j のカウターを 1 つ増加させ, j 番目の欠損値の推定をさせる, というやり方をしています。欠損値の数と行番号が合致しないので, こうした工夫が必要なのですね。その後のコードは先ほどと同じです。

このようにして, 推定してみましょう。Stan に与えるデータセットは次のようにして作ります。

code : 30.6　Stan に与えるデータセット

```
dataSet <- list(L = NROW(fullDays), W = fullDays$weight,
                Nmiss = sum(fullDays$weight == 999))
```

欠損値の数は, オブジェクト fullDays の weight 変数が 999 かどうかを判定させ (==でイコールかどうかという論理判断になります), 条件が合致した数を数え上げるという方法をとっています。

コードが走り出すと, 推定は順調に進むと思います。が, 結果の出方が少しややこしいです。状態 μ_i はすべての日程について推定されますが, これに加えて欠損があったデータも推定されていきますので, 描画する際は「実測値なのか推定値なのか」,「補間した欠損の推定値は何番目の欠損値だったか」を判断しながら結合していき, データにする必要があるからです。Stan の中で作ったカウント変数 j を R の方でも考えてやらないといけないわけですね。

code : 30.7　プロット用にデータを整形する関数とプロットのコード

```
Est2 <- fit2.df %>%
  dplyr::filter(str_detect(Varname, "mu")) %>%
  dplyr::filter(!str_detect(Varname, "muZero")) %>%
  dplyr::mutate(ID = str_extract(Varname, pattern = "\\d+") %>%
  as.numeric()) %>%
  arrange(ID) %>%
  select(ID, MAP, U95, L95)

Est2miss <- fit2.df %>%
  dplyr::filter(str_detect(Varname, "Miss_W")) %>%
  dplyr::mutate(ID = str_extract(Varname, pattern = "\\d+") %>%
  as.numeric()) %>%
  arrange(ID) %>%
  select(ID, MAP, U95, L95)

### plot用の関数を準備
plotFunction <- function(fullDays, Est, MissEst) {
  tmp <- fullDays %>%
    rowid_to_column("ID") %>%
    left_join(Est, by = "ID") %>%
    rowwise() %>%
    mutate(FLG = if (weight != 999) {1} else {2})
  misJ <- 1
  tmp$weight2 <- NA
  tmp$weight2U <- NA
  tmp$weight2L <- NA
  for (i in 1:NROW(tmp)) {
    if (tmp$FLG[i] == 2) {
      tmp$weight2[i] <- MissEst$MAP[misJ]
      tmp$weight2U[i] <- MissEst$U95[misJ]
      tmp$weight2L[i] <- MissEst$L95[misJ]
      misJ <- misJ + 1
    } else {
      tmp$weight2[i] <- tmp$weight[i]
      tmp$weight2U[i] <- tmp$weight[i]
      tmp$weight2L[i] <- tmp$weight[i]
    }
  }
  return(tmp)
}

plot.tmp <- plotFunction(fullDays, Est2, Est2miss)
g <- ggplot(data = plot.tmp) +
  geom_point(aes(x = date, y = weight2)) +
  geom_errorbar(aes(x = date, y = weight2, ymin = weight2L,
                    ymax = weight2U, color = palette()[2])) +
  geom_point(aes(x = date, y = MAP, color = palette()[3])) +
  geom_errorbar(aes(x = date, y = MAP, ymin = L95,
                    ymax = U95, color = palette()[4])) +
  scale_x_date(date_breaks = "1␣month",
               limits = as.Date(c("2020-01-01", "2020-05-01"))) +
```

```
    theme(legend.position = "none")
```

■**プログラム解説**

Est2 を作るブロック　MCMC サンプルをデータフレーム化したものの中から，状態 μ に関するデータだけを抜き出したものを作る。

Est2miss を作るブロック　欠損値を補間するための推定値を，MCMC サンプルから取り出した欠損値補間だけのデータセットを作る。

plot 用の関数　プロットはこの後にも行いますので，もう関数を作ってまとめてしまうことにします。もとのデータセット `fullDays` と，状態の推定値データセット `Est`，欠損値の推定値データセット `MissEst` を引数にとる関数です。

- **関数 3 行目**　まずもとデータに行番号を `ID` として変数化します。
- **関数 4 行目**　状態の推定値は，すべての行について推定しているはずですので，行番号をキーに結合 `left_join` します。
- **関数 5 行目**　もとの観測値が実測値なのか欠損値なのかを表現する変数 `FLG` を用意しておきます。これらはすべて，一時的なオブジェクト `tmp` に納められます。
- **関数 6 行目**　欠損値のインデックスを表す変数 `misJ` をつくり，カウントを 1 に設定しておきます。
- **関数 7-9 行目**　先ほどの一次オブジェクトにいまから体重の推定値を入れていきますが，この推定値は実測値で得られている時は不要で `NA` になるはずですので，いったんすべてに `NA` を代入しています。
- **関数 10-21 行目**　オブジェクト `tmp` を一行ずつ見ていって，もし `FLG` 変数が欠損値であることを教えてくれたら，`misJ` 番目の推定値を代入します。代入が終わったらカウンタを 1 つ追加しておきます。欠損値でなければ，もとの体重をそのまま移し入れます。ちなみに U や L がついているのは 95% 区間の上限と下限なのですが，実測値の場合は推定ではありませんのですべて同じ数字になります。
- **関数 22 行目**　作ったオブジェクト `tmp` を戻り値として返します。

関数の結果を受けとる　`plot.tmp` は描画用の一時データオブジェクトです。ここに全期間のデータと，今回の状態推定値，欠損推定値を入れておきます。

描画する　x 軸は全日程です。まずは体重 W_j を `geom_point` と `geom_errorbar` でプロットします。欠損の場合は推定値が入っています。次のポイントとエラーバーの geom は，状態 μ のものになります。

さあどうでしょう。コードはともかく，図 30.5 を見てみましょう。実際に観測しているのは黒点，状態の推定値は緑点と赤いエラーバーで表示されています。青いバーは欠損した時の推定値で，値がないのですから当然触れ幅は大きいですが，おそらくこの体重が測定できなかった日はこれぐらいだったんじゃないか，という値が推定されているわけです。数字で見ると，たとえば 2020 年 1 月 12 日の MAP 推定値は 83.4kg，95% 区間でいうなら [82.6,84.2] です。前日が 84.3kg で翌日が 83.1kg ですから，そんなもんかなあという気もします。

ところで，測定していないはずの日も補間できるよ，ということを考えてみると，これはすごいことだと思いませんか。そうです，未来の値ももちろん「測定していない日」ですから，このモデルを使うと未来予測ができるのです！すごい！

早速同じコードで，日程を未来に飛ばしてみましょう。データは 2021 年 12 月 06 日までしかありません[*7]

[*7] この原稿は 2021 年 12 月 10 日 17 時に執筆しています。

図 30.6　欠損していても推定できる

から，2021 年 12 月 31 日までデータを伸ばし，推定してみようではありませんか。コードは省略しますが，実行した結果は次のような図 30.7 として得られます。

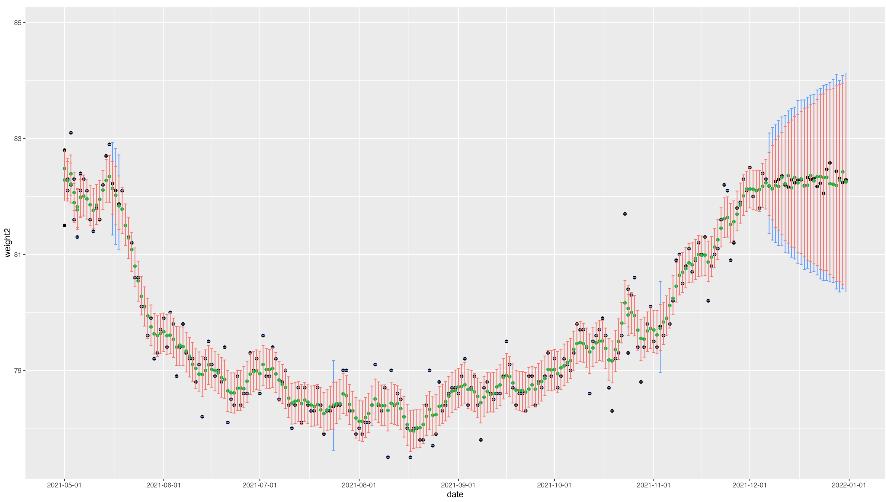


図 30.7　未来もある意味欠損値

　これをみると，データが得られなくなった次の日から，確信区間が日を追うごとに広がっていく様がみて取れます。当然そうですね。μ_{t+1} は μ_t にプラスかマイナスか，いずれとも言えない振幅幅で揺れ動くのですから，先に行けば行くほどその可能性は不確かなものになっていくのです。ちなみに描画用のデータから 2021 年最後の数日間の予測値を見てみると，次のようになっています。

code : 30.8　予想される 2021 年末の体重 (一部略)

```
> plot.tmp %>% tail
# A tibble: 6 × 11
# Rowwise:
    ID date         weight bodyFat   MAP   U95   L95   FLG weight2
  <dbl> <date>       <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
```

```
6     738  2021-12-26     999        NA   82.2   83.8   80.5       2      82.1
7     739  2021-12-27     999        NA   82.3   83.9   80.4       2      82.1
8     740  2021-12-28     999        NA   82.2   83.9   80.4       2      82.2
9     741  2021-12-29     999        NA   82.2   84.0   80.4       2      82.0
10    742  2021-12-30     999        NA   82.3   84.0   80.3       2      82.3
11    743  2021-12-31     999        NA   82.3   84.0   80.3       2      82.2d
```

未来の状態 μ は 82.2 のあたりでほぼ固定，それに伴って想定される体重 `weight2` も 82.2 程度ですが，幅がどんどんと広くなっていってます。可能性として，80kg に落ちているかもしれませんし，84kg になっているかもしれない，というのが現時点での予測になります。

予測ができるとは言え，なんだかほとんど変わり映えしない数字で面白くないですね。それは当然，「明日の体重は今日の体重 + 誤差だろう」という，とくに何らメカニズムや仮定を入れなかったものですから，ほぼ横一線の推定にしかならないわけです。

では少し欲張って，構造を入れみましょうか。ここまでは，μ_t は μ_{t-1} にのみ影響される，というモデルになっていましたが，もう 1 日前の影響が入っていることを考えます。たとえば体重が増加傾向にあるとか，減少傾向にあるといった，変化を考えてみることにしましょう。すなわち，$\mu_t - \mu_{t-1} = \mu_{t-1} - \mu_{t-2} + \varepsilon$ です。これは昨日から今日への変化 $(\mu_t - \mu_{t-1})$ は，一昨日から昨日への変化 $(\mu_{t-1} - \mu_{t-2})$ と同じようなものだろう (誤差 ε はあるけど)，と考えていることになります。このようなモデルを **2 階差分のトレンド**と呼びます。

この式を変形すると，

$$\begin{aligned}\mu_t &= \mu_{t-1} + \mu_{t-1} - \mu_{t-2} + \varepsilon \\ &= 2\mu_{t-1} - \mu_{t-2} + \varepsilon\end{aligned}$$

となることはすぐにわかりますね。これを確率モデルで表現するなら，

$$\mu_t \sim N(2\mu_{t-1} - \mu_{t-2}, \tau)$$

と考えていることと同じですから，そのように Stan のコードも変形します (コード 30.9)。

code : 30.9　2 階差分トレンドのコード

```
...(前略)...
model{
  mu[1] ~ normal(muZero, tau);
  mu[2] ~ normal(mu[1] , tau);

  {
    int j = 0;
    for(l in 1:L){
      if( W[l] != 999){
          // こっちは尤度
          W[l] ~ normal(mu[l],sig);
      }else{
        j = j + 1;
        // こっちはパラメータ
        Miss_W[j] ~ normal(2*mu[l-1]-mu[l-2], sig);
      }
    }
  }

  for(i in 3:L){
```

```
      //2階差分
      mu[i] ~ normal(2*mu[i-1]-mu[i-2],tau);
    }

    muZero ~ normal(80,10);
    sig ~ cauchy(0,5);
    tau ~ cauchy(0,5);
  }
```

コードは 2 階差分になっているので，状態は 3 時点目からしか推定できません。最初の 2 行で μ_1, μ_2 をメカニズムに沿って推定し，変化が取れるようになってから，は 2 階差分のコードで推定します。for 文が 3 から始まっていることに注意してください。

このコードと年末までのデータ，そしてプロット関数を使って推定した結果が次の図 30.8 になります。

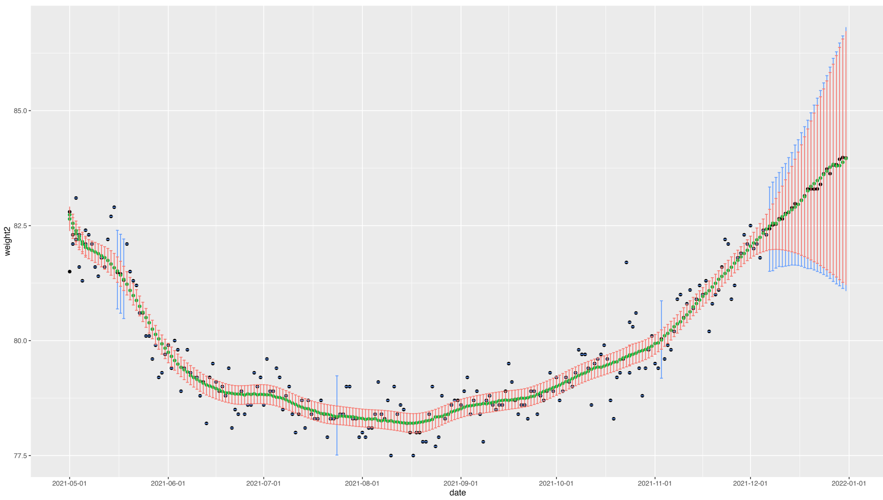


図 30.8　2 階差分トレンドの予測

これを見ると，それまでの増加傾向を反映して，年末には 84kg，最悪の場合 86.6kg まで増えている可能性があることがわかります[*8][*9]。

code : 30.10　2 階差分トレンドで予想される 2021 年末の体重 (一部略)

```
> plot.tmp %>% tail()
# A tibble: 6 × 11
# Rowwise:
     ID date         weight bodyFat   MAP   U95   L95   FLG weight2
  <dbl> <date>        <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
    738 2021-12-26      999      NA  83.9  85.6  81.5     2    83.6
    739 2021-12-27      999      NA  83.9  85.8  81.5     2    83.8
    740 2021-12-28      999      NA  84.0  85.9  81.4     2    83.8
    741 2021-12-29      999      NA  84.1  86.1  81.4     2    83.9
    742 2021-12-30      999      NA  84.2  86.3  81.3     2    84.1
```

[*8] なんてこった，まだ執筆中は答えが出ていませんが，この予測通りになったらとんでもないことですよ。だって私は 20 代 66kg だったんです。あの頃から 20kg も増えている自分の未来なんて想像したくないですよ。またダイエット始めるかなぁ。

[*9] 2021/01/15 追記。2021/12/31,6:07AM の記録は 82.3kg，体脂肪率 26.6% でした。ちゃんと確信区間の中に入ってますね！

```
11    743 2021-12-31     999       NA  84.2  86.5  81.2     2    84.0
```

今回は 2 階差分のトレンドを入れての予測となりましたが，この他にも周期的に影響してくる季節項をいれるとか，たとえば「週末は体重が増加するだろう」と考えて週末効果の項を入れるとか，さまざまな工夫を思いつくことができるのではないでしょうか。

30.4 状態空間モデルの展開

最後になりましたが，状態空間モデルは表面に現れている値の背後に潜在変数を仮定し，それを滑らかに繋ぎ合わせた**スムージング (smoothing)** の技術であるとも考えることができます。時系列を通じて，その前後の「状態」を，柔らかなバネで結合し，データ全体を通じて変化する関数として相互に調整させあっている，と考えることもできるでしょう。

また，時間というのは過去から未来へ進む，1 次元の道のりです。ところで私たちは 2 次元の地図を見たり三次元の空間を見たりしていますよね。たとえばこの状態空間が，時系列的一次元ではなく，X 軸と Y 軸に広がる網のような二次元結合していることを考えると，空間的な統計分析が可能になります。たとえば地価というのは，駅からの距離や大型商業施設，コンビニエンスストアなどの利便性に応じて変化しますが，おそらくそれらは連綿と繋がっていて，徐々に変化していくものであるはずです。であれば徐々に変化する要素をつなぎ合わせて，面の状態空間モデルを作ってやることもできるわけです。さらにその地価の変動を考える，つまり二次元平面 $\times$ 時系列に潜在変数をつなげてスムージングしたモデルを作れば，時系列的な地価の変動を考えたり予測したりできるかもしれません。

心理学はこれまで，調査法で一時点を切り出してその構造を把握する，というアプローチが多くありました。時系列を扱うとしても，プレ・ポストの変化を見るとか，あまり長期にわたって追いかけるようなことはしてきませんでした。しかし冒頭でお話ししたように，これからの心理学データは時間的にも空間的にも広がるものを計測し，利用できるようになるかもしれません。そのためにはより深い人間の観察や推察によるモデルの作り込みと，状態空間モデルとベイズ推定のような数理モデルと強力な推定法のタッグが，必要になってくるでしょう。

30.5 課題

体重変化のデータには，体脂肪の記録もあります。体重 $\times$ 体脂肪率から，筆者の筋肉量を計算できます。2020 年以降のデータを使って，筋肉量の変化を状態空間モデルで推定してみてください。2 階差分トレンドモデルで，未来のデータも予測できればなお結構です。これらのモデルを分析する R/Stan コードを提出してください。結果の解釈などを，スクリプトのコメントアウトや別添ファイルなどで提供してもらえるとなお良いです。もちろん Rmd ファイルでの提出であれば完璧です。なお提出されたコード単体でバグがなく動くことが確認できないものは，未提出扱いになります。コードの書き方などわからないところがあれば，曜日別 TA か小杉までメールで連絡し，指導を受けてください。

第 31 章

モデル比較

さて，今回でこの授業も最終回となりました。今回は授業全体のまとめとして，これまでの流れを俯瞰的に捉え直すとともに，今後みなさんがこの講義を通じて学んだことが，どのように利用できるのかに言及していきたいと思います。

本書の前半，心理学データ解析応用 1 として前期にお話ししてきた内容は，心理学で利用されるデータ，とくに心理尺度を用いたものが，どういう根拠で「心」を測定していると言えるのかについて解説しててきました。後半，心理学データ解析応用 2 の方は，プログラミングやベイズ統計といった目新しい研究手法の方が目についたかもしれませんが，その背後にあったメッセージは，どういうメカニズムでデータが生成されてきたのかという，**データ生成メカニズム**というアイデアをもつことでした。その意味で，本書全体のメッセージは一貫しています。すなわち，心理学者がデータを取るという時の，データに込められた意味や想定されるメカニズムをどのように表現し抽出し解釈するかという，心理学的営みそのものをお伝えしたかったのです。ここに無知・無自覚なまま，心理尺度で何かが測定できると考えたり，検定の結果から意味を解釈したりできるはずがないのですから。

そのための準備として，確率統計に関する理論や，コンピュータ，プログラミングに関する技能などの習得が必要だったわけです。心理学を学ぼうと思って入ったはずの大学で，どうして数学やプログラミングをしているのか，こんなはずじゃなかったと思った人もいるかもしれませんが，逆にどうして既知の知識や技術だけで，この摩訶不思議な心，人間，社会のありようが理解できると思っているのかと言いたいほどです。もちろん他のアプローチや，他にも学ぶべきことがたくさんありますし，時間は有限ですから苦手な手法は後回しにしたくなる気持ちはわかりますが，実は他の教養を身につけるよりもこちらの方が直接的で近道なルートだといえるかもしれません。

31.1　ベイジアンモデリング

Stan を使った統計的アプローチは，**ベイジアンモデリング (Bayesian Modeling)** と呼ばれることがあります。Stan が事後分布からの乱数発生機であり，事前分布と尤度を記述するだけで事後分布を得ることができるというところが，ベイズの定理に根拠を持ったベイズ統計学ですから，「ベイジアン」という冠がつくのは当然です。しかし後半の「**モデリング**」については，なにもベイズ統計学に限った話ではありません。モデルを立てて考えるということだけを考えれば，広い意味では心理学だって「心」というモデルを立てて考えているのですから。もっというと，学問というのは一般的に抽象化 (あるいは単純化) した理想の世界での論理的構造を扱うものですから，すべてモデリングアプローチだと言えるかもしれません。

心理学はその研究方法として，調査，実験，観察，介入といったものがあります。これらを通じて心とはなにかということを考えていくわけです。とくに実験や介入においては，条件の制御や統制を行って，それに伴って

生じた結果の違いを**効果**と考えるのでした。そのことと**帰無仮説検定**は非常に相性が良かった事は，想像に難くないでしょう。帰無仮説検定は，出てきたデータと変数の関係について，機械的に判断を下すことができる方法論でした。帰無仮説検定が農学から生まれてきたことからもわかるように，その方法はどの研究分野に対しても適用できる，共通のルール・御作法なのです。心理学者は実験計画に心理学的要素を埋め込み，結果を統計的に判断することで，心理学的な結果の解釈を進めるてきました。いいかえれば，実験の計画を立てることそのものが心理学のエッセンスであったのです。

しかし皆さんはすでに，実験計画が統計的に見れば**線形モデル**にすぎないことを知っていますね。そしてごく単純な線形モデルによってしか，心理学のエッセンスを表現する方法がなかった時代にあっては，想定しうる心のメカニズムが平均値の差として出てくるように置き換えるほかなかったのです。方法論に自由がない分，その他のところで工夫するしかなかったとも言えます。そうした工夫の中には芸術的なまでに作り込まれているものがあったりしますから，そこに痺れる憧れる，という人も少なからずいるのは当然ともいえるでしょう。

線形計画の結果は，操作/介入の効果があったかなかったかという判断になりがちです。結果がそれしか示しておらず，その結果を生むための工夫はすでになされており，得られるデータの精度も平均値差を示すことができれば良い程度にしかない，ともいえるかもしれません。しかし測定のツールは時事刻々と進歩してきましたので，より精緻に違いを見て，よりリッチな解釈を許す意味合い豊かなデータも得られるようになってきました。そのような時，伝統的な方法論だけで立ち向かうのはもったいないのではないでしょうか。方法論の呪縛から自由になり，さまざまな表現ができるようになれば，心理学ももっと発展させることができるでしょう。

モデリングは，変数間関係を数式で表現することでもあります。「心」や「気持ち」といった表現を始め，「幸せ」とか「恋愛感情」といった日常用語で考えるべきところを，$X_i, Y_{ij}, \alpha, \beta...$ といった記号で置き換えて表現するのには慣れが必要です。ひょっとしたら抵抗を感じる人もいるかもしれません。しかし，数式で表現する事は，他の解釈を許さない一意的表現をすることでもあります。人間って色々だよね，当てはまることもあれば当てはまらないこともあるよね，という日常的な理解で留まりたいのであれば – そしてそこで止まっていられるのは幸せなことでもあるのですが – それでいいのですが，学問を進める以上はより明確に，より誤解のない表現をしなければなりません。個別のケースにしか当てはまらないこと，その程度を確率で表現して，理論的に全体的な傾向を「こうなっているに違いない」と厳密に表現するには，日常用語ではなく数式用語が必要なのです。モデリングを進める上で，そうした表現ができるようにトレーニングする必要はありますが，使えるようになると折線回帰や状態空間モデルのように，さまざまな表現ができることも見てきた通りです。

この授業を受けてきたからと言って，さあ今すぐ自分で心理学的モデルを作れと言われても難しい，と思うかもしれません。実際これまで私が教えてきた中で，よく質問されるのは「やり方はわかるけれども，自分でやれる気がしない。どうやってモデルを思いついたらいいのか」ということです。これに対する答え方は 2 つあって，1 つは「色々なパターンを見て模倣する」です。「学ぶ」は「真似ぶ」からきているとも言われたりしますが，どのような表現の可能性があるのかについては，実際の応用パターンをさまざま見るのが早いでしょう。馴染みのない料理や調理法があれば，レシピ本を買って応用パターンをいろいろ仕入れますよね。そうすることで，「こんな工夫ができるのか。じゃあ自分はこうしてみよう，ここをちょっと変えてみよう」と前に進むことができるのですから。こうしたモデルがたくさん載っているものとして，Lee and Wagenmakers (2013) や豊田 (2017, 2018, 2019) などがありますので，パラパラっとみながら応用例を広く眺めてみるといいでしょう。

もう 1 つの答えは，「紙とペンをつかって設計図を書く」というものです。この方法は，この講義の中で再三お伝えしてきたやり方になります。データを見た時に，どういうメカニズムでデータが得られたかを考えるところにこそ，心理学者の本当の興味関心があるはずです。このデータは心のこういう仕組みを反映しているのではないか。データの背後には変数同士のこういう関係が潜んでいるのではないか，ということを考えていくわけです。わからないところは確率で表現しつつ，変数間関係を関数で記述する。そのための設計図です。設

計図と Stan があれば答え (事後分布) は得られるのですから，データ生成の仕組みを作り込むことに我々は全力を傾けることができます。複雑な関数関係をいきなり思いつくのが難しい，ということもあるでしょう。その場合はデータを可視化し，データに合うような関数はどんな形なのかを考える，あるいはいったん，単純な線形モデルを当てはめて，それを徐々に複雑にしていく，というやり方が良いでしょう。Kruschke (2014) は心理学者が書いたベイズ統計の本ですが，その後半は線形モデルです。一般化線形モデルや階層モデルについて，とくに複雑な前置きなく「こういう仕組みになってるんだから，とりあえずそれを反映して，関係は線形でも大体うまくいくでしょ」という感じで話が進んでいきます。簡単なモデルから徐々に複雑にしていく，というのがモデリングの王道でもあります。完成品だけをみるととても複雑に思えますが，設計図片手に紐解いていけば，意外とわかりやすいかもしれません。

モデリングは自由に設計図を書くことができるので，**分散分析**や **t 検定**など平均値差だけにこだわっている方法をみると，そちらがいかにも窮屈そうにしていると思えるかもしれません。講義を通じて得た知識や技術で，自由にデータを調理していただければと思います。

31.2 帰無仮説検定の代案

ベイジアンモデリングは，自由に書いた設計図から**確率的プログラミング言語 (stocastic programming language)** をつかって答えを得る方法です。プログラミング言語が使えるようになるの大変です。これまでの一般的な統計ソフトでは，ボタン 1 つで答えを出してくれたのに，と思う人もいるかもしれません。しかしそれは，従来の分析方法がさまざまな仮定や前提に基づいた，マニュアル的アプローチを許す範囲のものに限定されていたからですし，「簡単，楽ちん」ということが「知らなくていい」ということだとすると，その考え方がさまざまな問題を引き起こしてきたことを知らなければなりません。これは**心理学における再現性問題** (池田・平石, 2016) として取り沙汰されており，心理学そのものに疑問符を投げかけるような事態を招いています。その原因の 1 つが，統計的技術の誤用・悪用にありました。

心理学ではながらく，帰無仮説検定によって理論を積み重ねてきました。帰無仮説検定では p 値が 5% より小さければ良し，という「ここだけ見ておけば良い」というようなマニュアル的基準があり，差があればすごい効果が発見されたぞ！と喧伝するということで進んできた側面があります。すでに述べたように，この**有意差**を出すために要因計画の方にこそ心理学者は注力してきたのですが，$p < 0.05$ になりさえすれば良いという表面的な理解しかしていないマニュアル人間ができあがってしまうことが，問題を大きくしたのかもしれません。

ベイズ統計によるアプローチの場合，データがどのように出てきたのか，事前分布はどうなのか，といったことを考えないわけにはいきません。自由に設計図を描くことができるというのは，裏を返せば，最初は白紙でしかないということであり，きちんと動く統計ツールを作るためには細部まですべて記述しなければならないのです。帰無仮説検定では，極端な場合「データ生成には正規分布が仮定されている」ということを知らなくても，p 値がどうなるかだけをみる事はできてしまうのに，です。このように，ベイズ統計を使う利点の 1 つは，前提や仮定についてすべて自覚的に，明示的に記述しなければならないというところにあり，これが悪用・誤用を生みにくくする安全装置となっている面があります。

最近では JASP(JASP Team, 2021) という，GUI でベイズ分析もできるソフトウェアが出てきました。帰無仮説検定と同じような分析を，ベイズでやったらどうなるか，すぐに試すことができます。この便利なアプリでは Stan や JAGS などとは違って，どのような前提があるかを自覚しなくても，分析結果を得る事はできます。ですから，今挙げた「安全装置」としてのベイズ統計の利点は，簡単に外れてしまう日が来るかもしれませんし，その方がユーザにとってはいいかもしれません。この講義では第 20 講から第 23 講まで，帰無仮説検定の代わりにモデリングアプローチをする例を紹介してきました。さて，ではこうした帰無仮説検定の代わりと

して，ベイズ統計を使う利点はどこにあるでしょうか。

もちろんいくつか考えられますので，箇条書き的に解説してみたいと思います。

■幾重にも組み合わさる仮定や補正からは無縁　たとえば二群の平均値の差を検定する場合，データが正規分布に従っているかどうかを検定し，また二群の分散が等しいかどうかも検定して，問題なければ t 検定に進みます。もし分散が等しくないということになれば，Welch の補正をしなければならない，とマニュアルにはあります。またたとえば，Within の ANOVA をするときも，球面性の検定を行なってデータが分析の前提となる仮定にあっているかどうかを判定しなければなりません。主効果が見られた，交互作用が見られたとなれば，今度はどこに差があるのかをみるために – 検定の繰り返しを避けるために，下位検定を行います。

このように，仮説検定をする場合は条件にあっているかどうかを検定し，あっていなければ補正をするなどの手続きが必要です。同時に，仮説検定を同一のデータに何度も繰り返す事は，5% 水準という**タイプ 1 エラー**を生じる確率が適切に制御できなくなるため，本来なら実験単位ではなく「一連の分析単位」でこれを調整しなければなりません。正直なところ，ここまで厳密な調整が，あらゆる心理学研究できちんと行われているかと言われれば，その答えは No なのですが。

ベイズ統計の場合，こうした問題は非常に軽微なものに変わります。たとえば二群の分散が等しいかどうかについては，「等しくないモデル」を考えれば良いのです。具体的にいうと，二群の分散が等しいモデルは，

$$X_i \sim N(\mu_1, \sigma), Y_i \sim N(\mu_2, \sigma)$$

であり，等しくないモデルは

$$X_i \sim N(\mu_1, \sigma_1), Y_i \sim N(\mu_2, \sigma_2)$$

とするだけです (σ に添字がある，すなわち違う大きさとして推定する)。球面性の検定についても分散共分散行列の要素それぞれを推定するだけであり，その数値に事前に定められた制約はありません。

またたとえば，タイプ 1 エラーの制御についても，そもそもタイプ 1 エラーという発想がありませんから存在しないのです。データがあって，平均構造を線形モデルで表現したのであれば，そこから出てくる事後分布は 1 つしかありません。その単一のパラメータ同時事後分布をどのように切り分けようとも，ある判断のエラーに確率が伴う事はないのです。

帰無仮説検定を厳密に行うには，下位検定を事前に計画してどこにどのような差があると考えるかも準備しておく必要があります。帰無仮説検定は勝敗を決める手法ですから，ゲームのルールは事前に決めなければならないのです。ゲームの設定如何によっては，判定結果が変わってくることがあるからです。これに対してベイズ統計的アプローチであれば，結果 (事後分布) は決まっているので，下位検定を事前に計画する必要はなく，事後分布をどう切り分けようと分析者の自由です。

■停止規則やサンプルサイズの問題が生じない　同様のことは，サンプルサイズの設計についても言えます。帰無仮説検定の場合は，ゲームのルールを事前に決める必要があると言いました。このルールの中には，サンプルサイズも含まれています。サンプルサイズが大きくなればなるほど，有意であると判定される可能性は上がっていきます。帰無仮説検定における，最もやってはいけない研究実践のひとつは，データを取りながら検定を繰り返し，有意になったらデータを取るのをやめる，というものです。検定を繰り返すと，誤った判断をしてしまう可能性が増えますし，サンプルサイズが大きくなると有意になりやすくなります。つまり「勝つまでゲームをやめない」というのは卑怯な方法なのです。

しかし実際は，数人相手にデータをとってみたけど有意差はでなかった，でも惜しいから頑張ってデータを増やした，結果としてある日有意な差になったので，サンプルサイズをちゃんと記載して論文として提出した，ということがあるわけです。努力して真実に辿り着くのはいい話ですが，この例は努力したことが結果の捏造になっていたかもしれないという，より悲しい話になってしまいます。

帰無仮説検定は事前にサンプルサイズを決めなければいけません。このサイズ，この基準で勝負するぞ，というルールがあってからの一発勝負なのです。どこでデータの収集を止めるかというのは，**停止規則 (Stopping Rule)** といって，これまた事前に決めておかねばならないことなのです。事後的に，勝負に勝ったからこのやり方でよかったんだろう，と考えるのは間違っているのです。

これに対して，ベイズモデルの場合はこうした問題が生じません。最後が確率的な判断にならないので，こうした問題が生じないのです。データが蓄積されていくことは，徐々に確信度が高まっていくことでもあります。差があるのかないのか，という判断をするにあたって，よりその違いが明確になっていくだけなのです。

■より柔軟な仮説を立てることができる 帰無仮説検定は，帰無仮説と対立仮説という 2 つの仮説の比較判断になります。ここで帰無仮説は「差がない」「相関がない」といったものになります。たとえば二群間の母平均を比較する場合，帰無仮説は $\mu_1 = \mu_2$ になります。しかし実際には，差がないということを主張したい場合もあるかもしれません。帰無仮説に「差がある」というのをおくことができないのは，「あるというのは，どれぐらいあればいいのか」という問題がすぐに浮上するからです。たとえば母平均の差が $\mu_1 - \mu_2 = 0.1$ なのか，$\mu_1 - \mu_2 = 0.11$ なのか，$\mu_1 - \mu_2 = 0.111$ なのか$\cdots$言い出すとキリがないですね。p 値は帰無仮説のもとで計算されるものですから，差があるという仮説は無限に作れてしまうので，検証すべき帰無仮説も無限に膨れ上がってしまいます。差がないという状態は $\mu_1 - \mu_2 = 0$ と一意に定まるので，帰無仮説を設定することが可能なのです。

では帰無仮説を積極的に支持すればよいではないか，と思われるかもしれませんが，それができません。判定結果の p 値は帰無仮説のもとで計算していますから，「帰無仮説のもとで考えるとおかしな結果になった」という背理法が成り立たないのです。「帰無仮説のもとでおかしな結果にならなかった」というのは，帰無仮説以外の可能性を除外できるものではありません。

このように，帰無仮説検定では「効果がなかった」とか「関係がなかった」とは言えず，せいぜい「効果がないとは言えない」というにとどまるのでした。これに対して，ベイズ統計では**モデル比較**の観点から検証することになります。帰無仮説は $\mu_1 = \mu_2$ というモデル，対立仮説は $\mu_1 \neq \mu_2$ というモデルです。言い換えれば，帰無仮説は

$$X_i \sim N(\mu_1, \sigma_1), Y_i \sim N(\mu_1, \sigma_2)$$

というモデルであり，対立仮説は

$$X_i \sim N(\mu_1, \sigma_1), Y_i \sim N(\mu_2, \sigma_2)$$

というモデルだとしているに過ぎないからです。

これまでみてきたように，ベイズ統計のアプローチで平均値の差を考える場合は，$\mu_1 - \mu_2$ のような差の分布を**生成量**でつくり，その大きさをそのまま吟味できます。どうしても「差があるのか，ないのか」という判断がしたければ，**実質的に等価な範囲 (Region Of Practical. Equivalence; ROPE)**(→ セクション 20.2, Pp.220) を事前に設定し，その区間に差の分布が入ってくるかどうかを判断するのでした。そもそも差も分布しているわけですから，どれぐらい重複しているかという幅で考えるほかないわけです。帰無仮説検定の文脈においても，**効果量 (Effect Size)** とよばれる**標準化された差**の大きさを算出して考えますが，このように「どれぐらい違いがあれば差があると言えるか」という程度問題が前面に出てくることが重要なのです。5% 水準のように機械的に判断するのではなく，領域固有の知識 (ドメイン知識) に基づいて，実質的な差の大きさをケースバイケースで判断することが重要です。もちろん機械的に手続きを進められないことは不便かもしれませんが，不便だからといって真実を曲げるようでは本末転倒です。

また生成量を使ったアプローチのところで解説したように，パラメータについて考えるだけでなく，そのモデルから生成されるであろうデータについて仮説を立てたり検証したりできるのも，ベイズ統計的アプローチの

利点です。実際どの程度の差があると意味があるのかについて，具体的な数値シミュレーションで考えることができるからです。パラメータが δ 以上の差があれば良いとか，二群のデータが D 以上違ってくる確率はどれぐらいか，といった検証の仕方は，仮想空間上の p 値よりも具体的で意味のある情報をもっています。パラメータやデータについて，「同じかどうか」以上の情報を検証できることは，ベイズ統計の長所といえるでしょう。

31.3 モデル比較

先ほど，**モデル比較**ということに言及しました。このことについて，最後に少し触れておきたいと思います。ベイジアンモデリング，すなわちモデルをたててベイズ統計学的に推定するなかで，そのモデルの優劣を考える方法があります。

もう一度ベイズの公式を見てみましょう。ベイズの公式は次のようなものでした。

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

ここで右辺の分母は $p(D)$，つまりデータの確率をあらわすもので**周辺尤度 (marginal likelihood)** と呼ばれます。これはどのように計算されるのでしょうか。尤度は $p(D|\theta)$ で表されますが，$P(D)$ はこのパラメータが無くなったものになります。どのようにしてパラメータをなくすかというと，パラメータのありそうな可能性をすべて足し合わせることで**周辺化消去**すれば良い，ということになります。

たとえば性別と学年のクロス表があったとして，ランダムに選んだ 1 人の学生が女性である確率は，すべての学年の女性の確率を足し合わせたものになるわけです。記号で考えると，条件付き確率 $P(D|\theta)$ において，θ の可能性を全部考えれば $P(D)$ だけになるということです。これはパラメータ θ が離散的な例ですが，連続的な場合は

$$p(D) = \int p(D|\theta)d\theta$$

のように積分で考えればよいでしょう。

これはモデルに含まれるあらゆるパラメータのパターンを網羅して，データが得られる確率を計算したことになりますから，モデルが表現できる世界の中でどれぐらいデータを捕まえることができたか，ということを表しているとも言えます。ここでモデルが複数ある場合を考えましょう。あるモデルを $\mathcal{M}$ と表現することにして，第一のモデルを $\mathcal{M}_1$，第二のモデルを $\mathcal{M}_2$，と表したとします。すると事後分布も「あるモデルを仮定した時の事後分布」ということになりますから，次のようにすべて「モデルのもとでの」という条件付き確率で表現できます。

$$p(\theta|D, \mathcal{M}_i) = \frac{p(D|\theta, \mathcal{M}_i)p(\theta|\mathcal{M}_i)}{p(D|\mathcal{M}_i)}$$

ここで右辺の分母を見ると，$p(D|\mathcal{M}_i)$，つまり「モデル i のもとでデータが得られる確率」を表していることになります。もしこの確率が高ければ，このモデルはこのデータを生む確率が高いわけです。逆にもしこの確率が低ければ，このモデルはこのデータを生む確率が低いということになります。この周辺尤度は，モデルがデータを支持する度合いであり，証拠 (**エビデンス**) の強さとも言えるわけです[*1]。

その上で，次のような数字を考えます。

[*1] これを考えると，どんなデータでも作れる幅広いモデルを考えておけばいいじゃないか，と思うかもしれませんが，モデルを広く考えすぎるとそこから出てくるデータはそのごく一部でしかなく，かえってエビデンスとしては弱くなってしまいます。なんでも予測できるモデルは何も予測していないことと同じなのです。

$$BF_{12} = \frac{p(D|\mathcal{M}_1)}{p(D|\mathcal{M}_2)}$$

この数字は**ベイズファクター (Bayes Factor)** と呼ばれるもので，モデル 2 に比べモデル 1 がどれほどデータに支持されているかを示す数字です。この数字が 1.0 より大きければ，モデル 1 はモデル 2 に比べて相対的に強くデータに支持されているということになります。逆に 1.0 を下回ると，モデル 2 のほうがモデル 1 よりもいいね，ということです。モデル同士の比による表現ですので，BF_{21} としても構いません。ここでモデル 1 が対立仮説，モデル 2 が帰無仮説だとすると，どちらがよりデータをうまく表しているか，どちらがよりデータに支持されているかがわかるわけです。最近では，帰無仮説検定の文脈でも BF を併記して，どちらのモデルが良いかを示すこともありますし，この方法ですと帰無仮説 (というか差がないというモデル) を積極的に支持する，ということもできます。

この手法はベイズ推定するものであれば一般的に通用する考え方ですので，1 つのデータに対していろいろなモデルを考えついたとしても，ベイズファクターを計算して相互に比較できるわけです。すなわち，ベイズのモデル比較はベイズファクターという共通の基盤を手にしたとも言えます。そんな優れた方法があるのであれば，なぜもっと早く教えてくれなかったのか，という声が聞こえてきそうです。実は理屈上ではこの通りなのですが，この方法には実践上の問題が残されているのです。それは，周辺尤度の計算が難しいというものです。すべてのパラメータについて，考えられるすべての可能性を計算して足し上げるわけですから，それはそれは大変な計算になります。有効な近似計算の方法として，全パラメータの全確率空間を計算し尽くすかわりに，確率が高そうなところを効率よくサンプリングして推定値を得る**ブリッジ・サンプリング (Bridge Sampling)** という手法が考えられており，その計算をする R のパッケージ (Gronau, Singmann and Wagenmakers, 2020) が提供されていますから，今後こうしたモデル比較も増えてくるでしょう。

またモデル評価の方法として，**情報量規準**によるアプローチも考えられています。これは「良いモデルとは良い予測をするモデルである」と考えるところから始まっており，平均的に良い予測をするモデル程度を指標化するものです。ベイズの定理を基本としながら，データを真のモデルから生成されるサンプルと考えたり，確率分布同士の近さを**カルバック・ライブラー情報量 (Kullback-Leibler divergence)** で表現するなど，非常に高度な数学的体系が確立されています。詳しくは浜田・石田・清水 (2019) など丁寧な解説書があり，R パッケージなどで簡単に計算できるようになっています (Vehtari, Gelman and Gabry, 2017)。数学的にやや複雑な内容を含んでいますので，本講義の中では扱いきれんませんが，より進んだ研究や利用のために言葉の紹介だけはしておきます。

31.4 おわりに

本講ではここまで，ベイズ統計と従来の統計的分析を比較して議論してきました。ベイズ統計の方がメリットが大きいような解説をしてきましたが，これは筆者の好みもかなり含まれるものです。帰無仮説検定はなにも悪いものではなく，前提や仮定をしっかり守れば，一般的なソフトウェアで誰にでも結論を出すことができるという点で，とても良いものなのです。前提や仮定が多くて面倒だから，マニュアル人間ができてしまうというのは，あくまでもユーザ側の問題であって方法論の問題ではありません。ベイズ統計だって，モデルが間違っていれば結論は間違ったものになりますし，なによりデータに合わせてモデルを逐一設計図から書き起こすのは，大変コストがかかることでもあるからです。本来，これら 2 つの流儀は相反するものではなく，それぞれ依って立つ理論や考え方があって，必要に応じて選べるようになるのが一番です。

またベイズ流の研究アプローチ，とくにモデル比較については，まだまだ議論が活発に行われている段階であり，とくに心理学研究にこうしたアプローチがどれほど有用かについては，いまだに結論が出ていないところ

でもあります。パラメータの推定値で考えるのが良いのか，モデル比較をして議論を進めるのかについても，まだまだ定番の方法というのは見つかっていません。

そもそも心理学の知見を数式に落として**モデリング**できるようにするためには，心理学そのものの理論的発展や，人間についてのより深い理解も必要です。人間の行動と予測が完全にできるようなモデルや一貫した理論体系は，まだまだ遥か先の目標に過ぎないというのが現状でしょう。それでも有用なツールを武器に，少しずつ知見を積み重ねていくことができれば，心理学をより楽しく学べるのではないかとおもいます。Enjoy!

付録 A

よくある質問とミスの例

A.1 Frequently Miss and Comments

Rmd でレポートを提出したのに，なんだか中身の問題じゃないのに突き返された，中身を見てくれよ！と思う人もいるかもしれません。テキストでは R や Rmd での課題を提出するよう求めているところがありますが，その際よく見られる学生さんのミスとその対応についてのコメントをここにまとめておきます。

A.1.1 FMC1；そもそもファイルの書式が違う

Rmd で提出してください，R で提出してください，という指示に対して，違うものが提出されてくることがあります。書式があっていない，というのは些細なことのように思えるかもしれませんが，学術論文は書式に捉われず内容に集中するためにも，書式は整えられたものである必要があります。学会誌に掲載されている論文も，みなさんが書く卒業論文も，レポートに至るまで，書式や指示に沿ったものを準備する必要があります。書式があっていない場合は，門前払いになっても文句が言えないのがアカデミックの世界です。

ということですので，R や Rmd で提出してください，という指示があれば，R や Rmd で提出してください。ファイルの種類は拡張子で分類され[*1]，R ファイルは`.R`，Rmd ファイルは`.Rmd` という拡張子になっています。たまに`.Rproj` というファイルを提出してくる人がいますが，これは R プロジェクトのファイルで，これには R スクリプトも文書も含まれておらず，みなさんの計算環境情報が少し書いてあるだけです。また，`.Rmd` だけ入っていれば良いかというとそうではありません。まれに `Filename.Rmd.R` というようなファイルを送ってくる人がいます。これはファイル名にピリオドが含まれているだけで，ファイルの種類を識別する拡張子は`.R` ですから Rmd ファイルではありません[*2]。そもそもファイル名には 2 バイト文字はもちろん，`!`や`?`などの記号を含めるべきではありません。ピリオドも当然記号の一種ですから，ファイル名にするのは不適切です[*3]。

ではどうやって適切なファイル形式にするか，ということですが，最も素直な方法としては RStudio で新しくファイルを開くときに，R markdown 形式 (略して Rmd) を選ぶようにしましょう。もし間違って，R script 形式 (`.R` 形式) で開いてしまった，というときは，そのファイルを破棄して新しく作り直すのが一番ですが，エディタペインの右下にあるファイル種類表示 (図 A.1) をクリックして修正することもできます。

[*1] 拡張子についてはセクション C.5,Pp.386 参照

[*2] 最近の OS は拡張子を表示しない設定になっているものも少なくないので，このようなミスが生じます。

[*3] 長いファイル名などの場合，空白を入れるのも適切ではありません。できなくはないのですが，やるべきではないのです。どうしても空白を入れたければ，アンダースコアなどで区切ると良いでしょう。

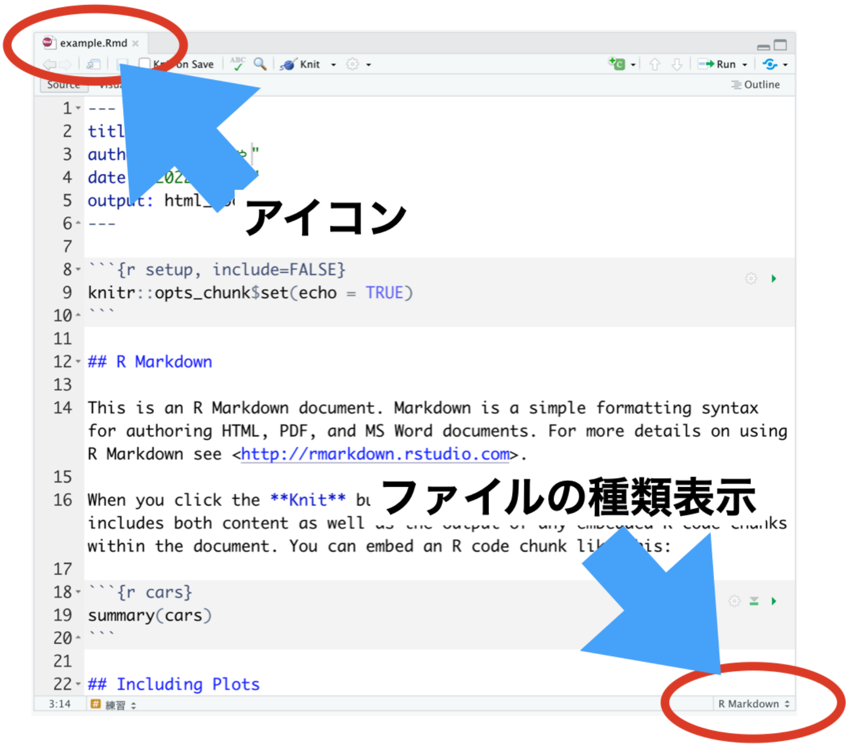


図 A.1　ファイルの種類を判別する方法

A.1.2　FMC2；やっぱりファイルの形式が違う

Rmd 形式のファイルは，拡張子だけで決まるものではありません。図 A.2 にあるように，Rmd ファイルは冒頭の 6 行 (正確には---で囲まれた領域) が YMAL と呼ばれるところで，文書全体の設定をしています。その下に，R のコードを実行する部分 (**チャンク (chunk)**) や文章の領域があります。文章のところは#記号で見出しを作ったりできます。

この YMAL 部分が壊れている，あるいはチャンクが正しく記述されていない場合，拡張子が`.Rmd` であっても適切な Rmd ファイルにはなっていません。YMAL 部分の書き方はよくわからない，という人も多いと思いますので，RStudio で Rmd ファイルを作ったときの状態をなるべく変更しないように注意すると良いでしょう[*4]。

チャンクは R のコードを書くところで，バッククォーテーション 3 つでくくるのが決まりです。チャンク領域が始まるところに{r}とかいて「ここが R で計算するところですよ」というのを指定するわけです[*5]。このバッククォーテーション 3 つ (```) が全角だったり (｀)，ダブルクォーテーションだったり (") すると，機械は正しくチャンクであると認識しません。フォントなどの見せかけ上は，微妙な違いのように見えますが，機械にとって違う文字列は違う意味を持ちますので注意してください[*6]。RStudio で編集する場合，チャンクの領域はや

[*4] Rmd ファイルを新しく開くときに，文書タイトルや著者名，日付，出力ファイル形式などを設定することができるウィンドウが開きますので，そこに必要な情報を書くと自動的にそれを使った YMAL が生成されます。また，本文として幾つかのサンプルコードが最初から含まれていますが，これらに関してはサンプルをそのまま使うことはないので，全て削除してしまって結構です。
[*5] ほかにも Python や Julia などの計算言語を混ぜることもできます。
[*6] 記号の名称や入力については，セクション E,Pp.395 を参照してください。

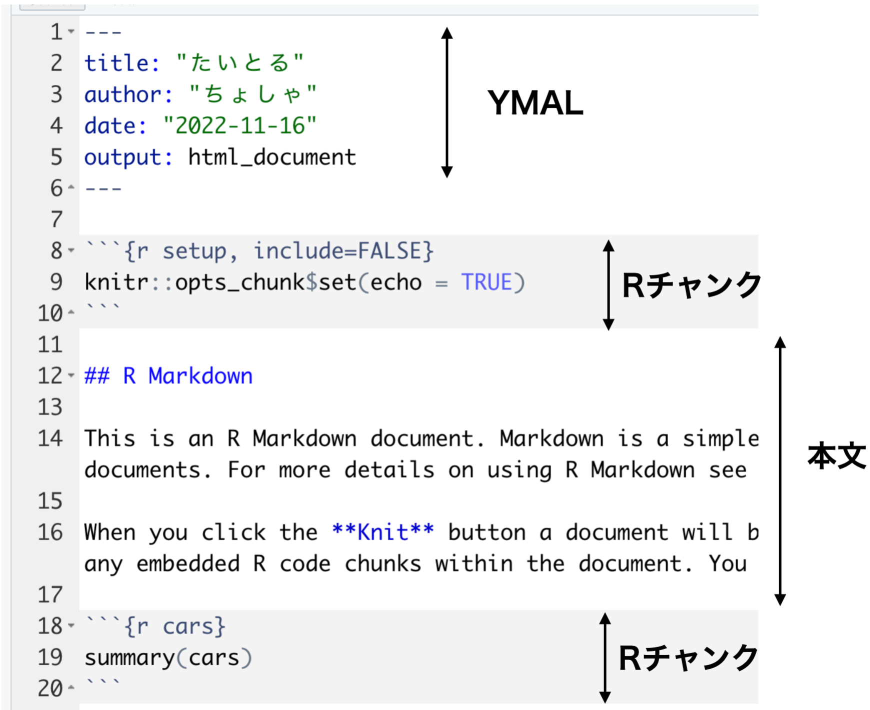


図 A.2 Rmd ファイルの中身における書式

や灰色がかった強調表示がされますので，どこにチャンクがあるかわかると思います。もし強調表示されていないようであれば，チャンクとして認識されていない可能性を疑った方が良いでしょう。

チャンクはバッククォーテーション 3 つで開き，おわったら同じくバッククォーテーション 3 つで領域を閉じます。閉じなければずっと R の計算領域が続くと解釈されますし，最後までチャンクが閉じられないと Rmd ファイルとして正しくない書式ということになります。チャンクが正しく入力できているかについて，常に注意を払っておく必要があります。また，自分でバッククォーテーション 3 つを使ってチャンク領域を開いたり閉じたりするのが面倒だ，という人は RStudio のチャンク挿入ボタンから挿入すると間違いないでしょう。

図 A.3 RStudio のチャンク挿入ボタン

A.1.3 FMC3；Rmd が knit できない

Rmd は文書作成に加えて，R での計算がセットになったファイル形式であり，文中の数字や分析結果は「書き写す」のではなく「その場で生成する」ものです。生成する，というのは Rmd ファイルを変換して，

HTML や DOCX，PDF 形式のファイルにすることを指します[*7]。このファイル変換を**ニット (knit)** といいます。編み物を編むようなイメージですね。このときに，タイトルをつけ，見出しのサイズを変え，R の計算をして結果を埋め込む作業をするわけです (図 A.4)。こうすることで，結果のコピペを避けること，再現性を担保することができるようになるわけです。すでに説明したように，チャンクを使って計算に必要な指示を Rmd

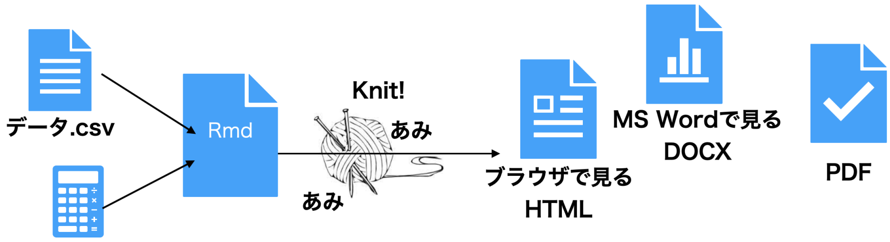


図 A.4　knit してレポートが完成する

ファイルに書きます。この指示はエラーのないコードである必要があります。当然ですが，スペルミスや間違った R コードは「実行できない」というエラーになるわけです。その場合，knit は中断され結果のファイルが出力されません。提出されたレポートは knit して出来上がったもののことを指しますから，knit できない Rmd ファイルを提出されてもレポートが提出されたことにはならないわけです。

knit できない Rmd ファイルになっていないか，というのはご自身の RStudio 環境で knit して確認できることです。提出前に一度，きちんと機能する Rmd ファイルになっているかどうか確認するようにしてください。以下で述べるように，自分の環境で knit できても，提出先の (私の) 環境で knit できないということもあり得ます。しかし，自分の環境で knit できないのに提出先でできる，ということはあり得ませんので，「knit できませんよ」というコメントをつけて返される前に自分で確認するようにしてください。

A.1.4　FMC4；外部環境を参照してしまう

さて，R で行う作業の中には，csv ファイルを読み込むといった「外部のファイルを使う」指示もありえます。例えば次のようなコードです。

code：A.1　Rmd 中の R コードが外部環境を参照してしまう

```
dat <- read.csv("social_effects.csv")
dat <- read.table("clipboard")
```

このコードでは，上の行は `social_effects.csv` というファイルを，下の行はクリップボードの内容を読み込むようになっています。ファイルを読み込む指示は，ファイルがなければ当然エラーになりますから注意が必要です。レポート等で Rmd ファイルを提出するとき，Rmd ファイルの他に必要な読み込むべきファイルがあれば一緒に提出するようにしてください。また，クリップボードの内容は再現できません。クリップボードとは，コピーアンドペーストのコピーを行ったとき，PC の内部で一時的にコピー内容を覚えておく場所のことです。つまり，みなさんがみなさんの環境で，クリップボード上にデータを一旦保持している場合は，このコードでエラーがしょうじることはありません。しかし，レポート提出先の (私の) 環境で，事前にデータのコピー作業を行なっているわけではないのですから，提出先でエラーが発生します。

[*7] HTML は Hyper Text Markup Language の略で，ブラウザで開くファイル形式です。DOCX は Microsoft Word のファイル形式です。PDF は Portable Document Format の略で，データを紙に印刷した状態のようにサイズを固定して出力したファイル形式であり，OS がもつビューワーや Adobe Acrobat Reader などで見ることができます。

そもそも Rmd ファイルは，作業の再現性を担保するためのファイルになっているわけですから，クリップボードの利用のような「自分の環境だけで可能な記録されない作業」をそのファイルに含めるのは適切ではありません。Rmd ファイルは Rmd ファイルだけで分析作業が完結するように，必要な記録は全て記載されている必要があります。同様に，分析作業に関係のない冗長な指示や無駄な指示は Rmd ファイルに含めるべきではありません。

A.1.5 FMC5；提出先の環境を変更してしまう

R はさまざまなパッケージを使って分析環境を拡張することができます。パッケージは **CRAN** を通じてインターネット経由で配布されますから，ネット環境があれば誰でも最新のパッケージをとってくることができます。みなさんが Rmd ファイルの中で使う R の関数の中には，パッケージの関数も含まれているでしょう。パッケージがなければ関数が動きませんから，パッケージをインストールする作業も Rmd に書いておきたいと思うかもしれません。しかし，これは推奨できません。

パッケージのインストールは，実行環境の準備にあたる作業です。Rmd ファイルを使ってコードのやり取りをするとき，提出先の環境で分析することになりますが，提出先の環境にどのようなパッケージをいつ入れるかは，提出先の環境の管理人が判断すべき問題です。提出する Rmd ファイルにパッケージをインストールする関数，すなわち `install.packages()` 関数が含まれているというのは，相手の環境を勝手に操作してしまうことと同じであり，セキュリティ的にも適切な発想ではありません。

環境の準備は提出先の管理人が管理すべき問題であり，またすでにパッケージが入っている場合は無駄な上書き作業をさせることにもなります。また `install.package` 関数は CRAN のサーバを参照したりしますから，適切な設定がなければ R チャンク実行時にエラーが発生します。いずれにせよ，R チャンクの中に `install.package` 関数を含めないようにしましょう。

以上が Rmd ファイルや R ファイルでレポートを提出するときの留意点です。その他にも R や RStudio を使うときによくある質問がありますので，それらも一問一答型で紹介しておきましょう。

A.2 Frequently Asked Questions；よくある質問と答え

Q. A.2.1: テキストを参考にパッケージをインストールしようとしたところ，エラーが発生しました。

A. 質問ではなくて報告ですね。お返事としては，「わかりました。エラーが発生して大変ですね。」としか言いようがありません。どの環境で，何をしようとして，どのようなエラーが出たのか，明示してください。メールのやり取りで指示が明確になるように，テキストを準備しています。何章，何ページの，どの文章を参考にしたのかも教えてください。

Q. A.2.2: 添付のようなエラーが発生しました (図 A.5)。対応策を教えてください。

A. エラーの発生画面を送ってこられていますが，この画面に写っていない上の方でエラーが発生していますから，これではどのエラーなのかわかりません。スクリーンショットを送ってこられることはよくありますが，ほとんどの場合，適切な箇所が写っていません。複数枚添付してこられる人もいますが，画面を拡大しながら読むのも難しいので，**R コードそのものを送ってきてください**。そうすると，何行目のどこにエラーがあるかが明確になり，対応に関係ない無駄なやり取り (ex.「もう少し上を写してください」「違う，もっと上」) が減ります。

8989

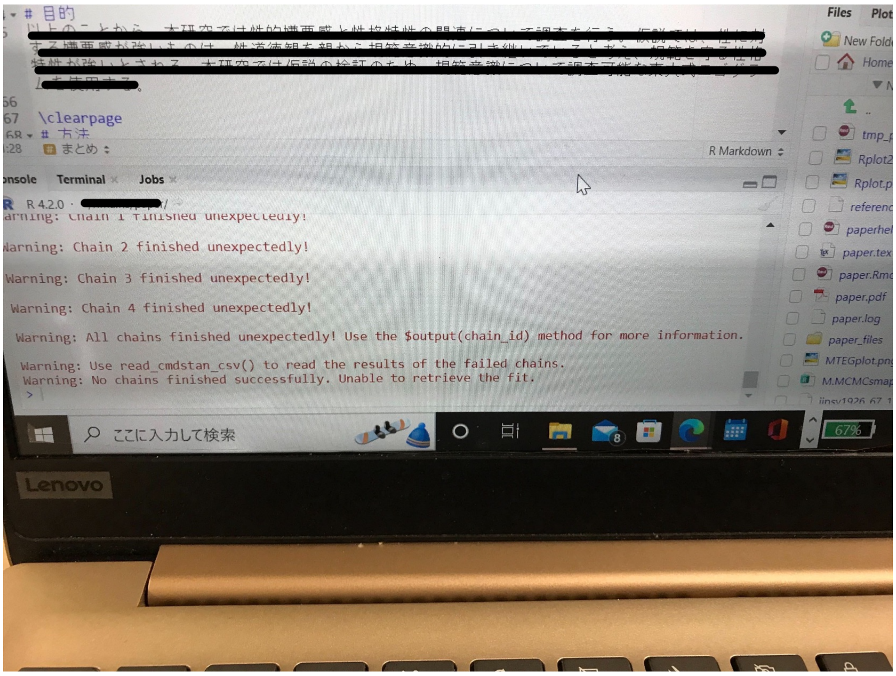


図 A.5　添付されてくるスクリーンショットの一例

Q. A.2.3: ファイルを読み込もうとすると次のようなエラーが出ます。どうすれば良いですか。

`'xxxxx'` に不正なマルチバイト文字があります

A. 文字化けの原因は，たとえば UTF-8 形式で提供されているファイルを，Windows 標準の CP932 形式で読み込もうとする，という文字コードの不一致です。ですからそれをオプションで指定してあげれば問題解決です。`read.csv` 関数を使っている場合，オプションの指定は `read.csv("foo.csv",fileEncoding="UTF-8")` のようにします。`tidyverse` の `read_csv` 関数を使っている場合，locale オプションで，locale 関数の encoding を明示的に UTF-8 とします。`read_csv("foo.csv", locale=locale(encoding = "UTF-8"))` のようにします。

8990

Q. A.2.4: ファイルを読み込もうとすると次のようなエラーが出ます。どうすれば良いですか。

`'xxxxx'` **に不正なマルチバイト文字があります**

A. 文字化けの原因は，たとえば UTF-8 形式で提供されているファイルを，Windows 標準の CP932 形式で読み込もうとする，という文字コードの不一致です。ですからそれをオプションで指定してあげれば問題解決です。`read.csv` 関数を使っている場合，オプションの指定は `read.csv("foo.csv",fileEncoding="UTF-8")` のようにします。`tidyverse` の `read_csv` 関数を使っている場合，locale オプションで，locale 関数の encoding を明示的に UTF-8 とします。`read_csv("foo.csv", locale=locale(encoding = "UTF-8"))` のようにします。

8991

Q. A.2.5: パッケージを読み込もうとすると次のようなエラーが出ます。どうすれば良いですか。

`library(tidyverse)` **でエラー；**`'tidyverse'` **というパッケージはありません。**

A. パッケージがないので，インストールしてください。

8992

Q. A.2.6: パッケージをインストールしようとすると次のようなエラーが出ます。どうすれば良いですか。

```
Warning in install.packages:
    'lib = "C:/Program Files/R/R-4.0.5/library"' is not writable
```

A. ユーザがファイルに書き込みをする権限を持っていないので，(パッケージファイルをドライブに) 書き込みできないというエラーです。RStudio を実行する時に，「管理者として実行」を選びましょう。

8993

Q. A.2.7: パッケージをインストールしようとすると次のようなエラーが出ます。どうすれば良いですか。

```
Warning in install.packages:
    ディレクトリ 'C:～(任意の文字列)～\????' を作成できません。理由は
```'Invalid argument' です。
```

A. ユーザ名が全角文字を含んでいるため，文字化けして R 側から操作ができません。解決する方法は二つあります。

**ユーザを作り直す方法** ユーザ名に全角文字を含まない，新しいユーザを作成します。設定＞アカウントから新しいアカウントを作りましょう。

**インストールフォルダを指定する** R のコンソールで `.libPaths()` と実行すると，パッケージをインストールするフォルダが出てきますが，ここを変更する方法です。次の 2 ステップで対応できます。

- まず C ドライブのすぐ下にインストール先のフォルダを作ります。`myLib` としましょう。
- 次に R のコンソールで `.libPaths("C:/myLib")` 書いて実行します。

これでインストール先が変わりますので，書き込み·インストールができるようになります。老婆心ながら付け加えますと，フォルダ名はなんでも構いませんが，全角文字ではいけません。また場所も C ドライブのすぐ下である必要はありませんが，全角文字や空白を含むフォルダの下に入れてしまってはいけません。OndDrive のようなクラウドを指定するのも良くありません (探しに行った時にオフラインだとまたエラーになります)。

8994
```

Q. A.2.8: ファイルが読み込めません

```
file(filename, "r", encoding = encoding) でエラー:
  コネクションを開くことができません
 追加情報:  警告メッセージ:
 file(filename, "r", encoding = encoding) で:
  ファイル 'foo.csv' を開くことができません: No such file or directory
```

A. 指定された場所にファイルがないので，読み込むことができないエラーです。確認すべきは，「プロジェクトを開いているか」，「プロジェクトフォルダの中に当該ファイルはあるか」，「ファイル名のミススペルはないか」です。プロジェクトってなんだという人は，RStudio の基本に立ち戻り，プロジェクトでファイルやフォルダを管理するようにしてください。プロジェクト管理については，Pp.87 にもその説明があります。

プロジェクトによる管理とは要するに，R が今見ているフォルダの場所を固定する方法です。プロジェクトを開くと，プロジェクトフォルダが「今見ているフォルダ (ワーキングディレクトリ)」になりますので，その中のファイルを参照することになります。プロジェクトフォルダの中に当該ファイルがないと読み込むことができませんので[a]，当該ファイルをプロジェクトフォルダ内に移してきてください。

[a] フォルダの位置を相対的・絶対的に指定してやれば，どこにおいていても読み込むことはできますが，この問題で悩んでいる人は相対・絶対パスの指定というところでさらに疑問が深まることになると思いますので，気にしないでくれて結構です。相対・絶対パスが知りたい人は，付録 C,389 でファイル場所とは何かを再確認してください)。

8995

Q. A.2.9: ANOVA 君が読み込めません

```
file(filename, "r", encoding = encoding) でエラー:
  コネクションを開くことができません
 追加情報:  警告メッセージ:
 file(filename, "r", encoding = encoding) で:
  ファイル 'http:/riseki.php.xdomain.jp/index.php?plugin=attach&refer=
 ANOVA 君&openfile=anovakun_486.txt' を開くことができませ
ん: Invalid argument
```

A. ファイルを読み込みにいく先が URL，すなわちインターネット上になっています。ANOVA 君のファイルを一度手元の PC のフォルダにダウンロードし，そのローカルのファイルの位置を指定して読み込むようにしてください。

8996

Q. A.2.10: 効果量として Hedges の g を算出してください。という指示について実行するとgではなく d が出てしまうようなのですが、どうすればよいでしょうか。コードは次の通りです。

```
cohen.d(value ~ condition, data = dat, hedges.crrection = T)
```

A. Hedges の補正 (correction) のオプションが通っていません。スペルミスです。オプションのスペルが間違っているので無視されたので，関数名通り Cohen の d が算出されます。

8997

Q. A.2.11: 描画の際に次のような注意が出てきます。これはどういう意味ですか。

```
Removed 671 rows containing missing values (geom_point).
```

A.「671 件の行で欠損値が含まれています」ということです。つまりデータセットの中に欠損値(観測されていない，数値が入っていない行)が 671 件あったので，それは表示できませんでしたよ，という意味です。警告が嫌だということであれば，データセットの中で欠けているものを除外する必要があります。R の関数では，`na.omit` で欠損値を除外することができます。

8998

Q. A.2.12: lm 関数を実行するコードでオブジェクトがないと言われます。

```
result <- lm( weight - height, data = dat)
```

A. 従属変数と独立変数とをつなげるのはチルダ (~) という記号です。そこがハイフン (-) になっています。スペルミスの一種です。

8999

Q. A.2.13: lm 関数を実行するコードでオブジェクトがないと言われます。

```
result <- lm( weight - height )
```

A. データを与えていないので，`weight` というオブジェクトを探しに行って，見つからないというエラーです。`data` オプションでデータを与えてあげてください。

9000

Q. A.2.14: 因子分析の Robust 法での RMSEA の p 値って超えてたらまずいですか。Standard(DWLS) の方だけで報告はだいじょぶでしょうか。

A. 非常に専門的な質問をしておられますが，まず，「因子分析」「Robust 法」「RMSEA」など，個々の用語の意味はわかっているでしょうか。キーワードによる検索で，「ロバスト法とは」「Robust の意味」などは答えが出てくると思いますが，これらを分析法の体系的な文脈の中に位置付けないと，答えられない種類の問題です。この例をもとに説明すると，Robust を辞書で引くと「壮健」「たくましいこと」などと出てきますが，もちろんそういう意味ではありません。ロバスト法を検索すると，「統計学の分野でロバスト推定法というやり方がある」「観測値に外れ値が含まれている可能性を考え，その影響を抑えることを目的とした手法」などの説明が出てきます。ロバスト推定法は因子分析に限らず，回帰分析など他の手法でも使われる考え方なので，このような一般的な解説になります。ですがここで知りたいのは「因子分析におけるロバスト推定」ですから，因子分析でロバスト推定するとはどういうことか，因子分析の観測値における外れ値とは何か，因子分析における推定とは何を推定するのか，そもそも因子分析とは何を目的としているのか，なぜ自分は因子分析方を使うのか，さらに何故因子分析の中のロバスト推定を使いたいと思っているのか，といったことがわかっていないと，この質問に正しく回答することができないのです。このように，複合的な要素について一回で質問しても，適切な答えに辿り着けないことがあります。聞くことは恥ずかしいことではありません。知らないことを知っているふりをすることが恥ずかしいことであり，知らないまま「みんなそうやっているから」「テキスト/ネットに書いてあったから」と看過してしまうことこそ恥ずべきことなのです。ちなみに，質問に対する回答を間違えることも恥ずかしいことではありませんから，「正しく答えられなければ恥をかく (から質問しない)」というのも同様に恥ずべきことです。こうした恥ずかしさ (保身) から，「専門用語を使ってそれっぽく質問してわかった感じになろう」というのは，かえって遠回りになります。実は回答よりも，質問の仕方でその人の理解度が明らかになってしまっています。このように書くと，なんでも反射的に「わかりません，教えてください」という人もいますが，それも適切な質問方法ではありません。教員がアドバイスできるのは，「答え」ではなく「理解」が欲しい人に対してだけなのです。質問する場合は，自分は何がわかっていて何がわかっていないのか，何が知りたいのかを明確に言語化して質問するようにしてください。

9001

付録 B

標準正規分布から尺度値を求める計算方法

Likert 法では，態度が標準正規分布すると仮定するのでした。標準正規分布をカテゴリの相対度数で分割し，あるカテゴリ c の上限の確率点 z_c，下限の確率点 z_{c-1} の確率密度の差分を，相対度数 p_c で割ることで，尺度値 Z_c が下の式で得られます。

$$Z_c = \frac{(y_{z_{c-1}} - y_{z_c})}{p_c}$$

ここで y_{z_c} は標準正規分布をカテゴリで区分し，当該カテゴリ c までの累積確率点 z_c における確率密度，p_c はカテゴリ c の相対頻度です。図 B に記号の対応関係を示しましたので，確認してください。

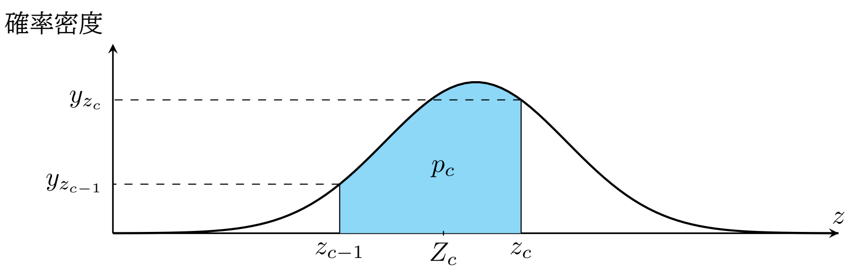


図 B.1　標準正規分布と対応する記号の確認

尺度値 Z_c を求める計算が確率密度 $y_{z_{c-1}}, y_{z_c}$ と，相対度数 p_c で算出されるというのは一見奇妙です。どうしてこのようになるのかを見ていきましょう。

まず標準正規分布の確率密度の式を確認しておきます。確率点 z における確率密度 y は次の式で算出できます[*1]。

$$y_z = f(z) = \frac{1}{\sqrt{2\pi}} e^{(-\frac{z^2}{2})}$$

この関数は確率密度の曲線を表しており，確率はその面積です。z_{c-1} から z_c までの面積 (確率) は，積分を使って

$$p_c = \int_{z_{c-1}}^{z_c} f(z)dz = \int_{z_{c-1}}^{z_c} \frac{1}{\sqrt{2\pi}} e^{(-\frac{z^2}{2})} dz$$

---

*1 e^x を $\exp(x)$ と書くことも少なくありませんし，この e は自然対数の底で $e = 2.718\ldots$ という実数です。この数字は微分しても変わらない，$(e^x)' = e^x$ という便利な特徴を持っています。

で表されます。

さて，ある確率点 z における密度の高さを $f(z)$ としたとき，z の微小な増分 Δz を考えると，区間の面積 $S = f(z)\Delta z$ を考えることができますから，逆にある点 z が知りたい時は

$$z = \frac{\sum Sz}{\sum S}$$

とすれば良いことになります。積分はこの微増分 Δz の極限を

$$\lim_{\Delta z \to 0} \Delta z = dz$$

と考えることですから，ここで考えたいのは

$$Z_c = \frac{\int_{z_{c-1}}^{z_c} f(z) z dz}{\int_{z_{c-1}}^{z_c} f(z) dz}$$

になります。分母は確率密度関数の積分ですから面積すなわち確率で，ここでは p_c であり，その面積を実データの相対度数で代えてもいいでしょう。

分子については少し式の変形が必要です。記号を見やすくするために，$a = z_{c-1}, b = z_c$ と書き換えておきましょう。

$$\begin{aligned}\int_a^b yzdz &= \int_a^b \frac{1}{\sqrt{2\pi}} e^{(-\frac{z^2}{2})} zdz \\ &= \frac{1}{\sqrt{2\pi}} \int_a^b e^{(-\frac{z^2}{2})} zdz\end{aligned}$$

ここで変数変換をして，$u = -\dfrac{z^2}{2}$ とすると

$$\begin{aligned}\frac{du}{dz} &= -z \\ du &= -zdz\end{aligned}$$

となり，積分の下限は $-a^2/2$, 上限は $-b^2/2$ になりますから，以下のように展開できます。

$$(\text{与式}) = -\frac{1}{\sqrt{2\pi}} \int_{-a^2/2}^{-b^2/2} e^u du$$

$$\int e^u du \text{ は } e^u \text{ なので}$$

$$= -\frac{1}{\sqrt{2\pi}} \left[e^u\right]_{-a^2/2}^{-b^2/2}$$

$$\begin{aligned}&= -\frac{1}{\sqrt{2\pi}} e^{(-\frac{b^2}{2})} - \left(-\frac{1}{\sqrt{2\pi}} e^{(-\frac{a^2}{2})}\right) \\ &= \frac{1}{\sqrt{2\pi}} e^{(-\frac{a^2}{2})} - \frac{1}{\sqrt{2\pi}} e^{(-\frac{b^2}{2})} \\ y &= \frac{1}{\sqrt{2\pi}} e^{(-\frac{z^2}{2})} \text{ であることから,} \\ &= y_a - y_b \\ &= y_{z_{c-1}} - y_{z_c}\end{aligned}$$

以上のことから，リッカート法において標準正規分布をもとに尺度得点を決めるには，

$$Z_c = \frac{(y_{z_{c-1}} - y_{z_c})}{\int_{z_{c-1}}^{z_c} f(z)dz} = \frac{(y_{z_{c-1}} - y_{z_c})}{p_c}$$

とすれば良いことになります。

この計算に至る理論的背景は，より専門的には**系列範疇法 (Method of Successive Categories)** と呼ばれ，順序カテゴリに数値を付与する心理測定論，精神物理学理論からきています。詳しくは Guilford (1954) や西村 (1977) も参照してください。

付録 C

電子計算機のイロハ

C.1　前置き

このセクションは心理統計ではなく，コンピュータについての四方山話をダラダラと書いています。そんなの聞かなくてもわかってるよ，という人もいるかもしれませんし，知らなくてもみなさんはきっとスマートフォンやタブレットを使えていることと思います。しかし知らずに使うことと，知ってて使うことには大きな違いがありますし，今後大学でレポートや論文を書いたり，それに必要な統計処理をするためにも，計算機の基本的な特徴を知っておくと，トラブルに出会した時に「あぁこれってひょっとして」というヒントが得られたり，納得できるようになるかもしれません。知らなければ「何だかわからないけどパソコンが壊れた」というか，「パソコン運が悪い」「自分はパソコンが苦手なのだ」と間違えた帰属をしてしまうことになります。

コンピュータ関係の授業で聞いたことがある話，これから聞く話もあると思いますが，もし苦手意識を持っている人がいたらこれを機に再入門するつもりで読んでください。

C.2　コンピュータの基礎

21 世紀に生きる私たちは身の回りをコンピュータに囲まれて生きています[*1]。それは携帯電話の形をしていたり，ノートパソコン，デスクトップパソコンの形をしていたりします。また時々テレビなどで報道されますが，気象予報や飛沫がどのように飛び散るかをシミュレーションする大型計算機「富嶽」などもコンピュータですね。これらは形は違いますが，いずれも電子計算機であり，電子計算機には次の 5 つの装置があります。

入力装置　キーボードやマウス，タッチパネルなどを使って情報を取り込むデバイス (装置)

出力装置　モニタやプリンタ，タッチパネルなどを経由して情報を出力するデバイス

演算装置　プログラムの命令に従って計算処理 (四則演算や論理演算) をする装置。一般に電子計算機の中央で一括して処理するので，Central Processing Unit(CPU) と呼ばれるものです。

制御装置　演算結果に従って他の装置に指示を出す装置のこと。演算装置とまとめて CPU に実装されています。

記憶装置　計算結果などの情報をいったん保持しておく装置のこと。コンピュータの内部にあって一時的な

[*1] 私は 1976 年生まれですが，生まれた頃は周りにコンピュータなんかありませんでした。小学生の頃にマイコン (マイクロコンピュータ，小さなコンピュータという意味でもありますが，My Computer，私のコンピュータという意味でもあります。つまり個人単位でコンピュータが使えるようになった，というだけでも大きな出来事だったのです。) という言葉が出てきて，なんかかっこいいなと思った記憶があります。私が 10 歳になったころ，ビデオゲーム (テレビゲームでやるゲームが家庭でできるようになり，それをこのように呼びました。) が身の回りに出てきました。ファミリーコンピュータ，とくに「スーパーマリオブラザーズ」によって日本中の子供たちが熱狂したのが 11 歳の頃，「ドラゴンクエスト」によって社会問題になったのが 12 歳の頃になります。ともかくこの頃は，コンピュータといってもゲーム機のような扱いでした。

計算に使われる一次記憶装置，ハードディスクドライブ (Hard Disk Drive,HDD) やソリッドステートドライブ (Solid State Drive,SSD) などの二次記憶装置など。

最後の記憶装置については，一次記憶装置，二次記憶装置と種類が分かれていますがこの区別は簡単で，電源を落とした時に記憶が消えてしまうのが一次記憶装置，電源を落としても記憶が消えないのが二次記憶装置です。たとえば $12 + 38 =$ という計算をするとき，頭の中で「えーっと一の位が 2 と 8 だから 10 になって繰り上がるから･･･」と考えてから，ノートに $= 50$ という答えを書くと思います。次の問題に進むと，先ほどの「1 繰り上がるから･･･」という情報は忘れてますよね。でもノートに書いた $12 + 38 = 50$ というのは残っています。このノートがいわば二次記憶装置であり，頭の中で一時的に保持していた情報が一次記憶装置ということになります。一次記憶装置は RAM(Random Access Memory) とも呼ばれます[*2]。

ところで，コンピュータがやっているのは計算だけです。私はこの資料を PC に向かって書いており，キーボード (入力装置) を叩きながら，画面 (出力装置) を見て文字を連ねています。これも「キーが押されたら文字を表示させ，その文字列を記録する」という処理を機械が淡々とこなしているに過ぎません。あるいはマウスやトラックパッドで，アイコンを指し示し，カチリと押す[*3]ことで選択し，押し込んだまま移動させ (ドラッグ)，離すことで別の場所に置いたりします (ドロップ)。トラックパッドの場合は 2 本指で同時に押したりしますし，タッチパネルの場合は二本指を広げたり (ピンチアウト)，逆に二本指を狭めたり (ピンチイン)，3 本以上の指でファサーっと触って場所を広げたりします。たとえば「ファイルを掴んでゴミ箱に捨てる (削除する)」というのが我々にとっての操作ですが，コンピュータの内部では実はこんなことをしていません。ファイルは (二次) 記憶装置に書き込まれた情報です。記憶装置は原稿用紙のように小さなマス目がたくさんあって，ファイルとはそのマス目の XXX 番目から YYY 番目までの情報，ということです。このファイルを削除するというのは，記憶装置のある場所 (アドレス) に「削除されたものなので画面に表示しない」という情報を書き込む，という操作をしているだけです。じゃあなぜ私たちは「ゴミ箱にドラッグ＆ドロップ」なんてするのでしょう？ それはそのほうがわかりやすいからですよね。「ファイルを削除する」というのは，「メモリアドレスの XXX 番地に別の情報を書き込む」という操作だと言われてもピンとこないので，コンピュータが人間にとってわかりやすい表現をして見せてくれているのです。このユーザにとってわかりやすい幻を見せてその気にさせてくれるというデザインのことを，ユーザーイリュージョンと言います。ともかくこういう「画面で見ながら操作する」ことをグラフィカル･ユーザ･インターフェイス (Graphical User Interface,GUI) といいますが，これのおかげでコンピュータの操作は随分楽になっています[*4]。

C.3 コンピュータの歴史

コンピュータの装置，すなわちハードウェアについての解説につづいて，ソフトの側面についても解説を加えようと思うのですが，そのためには少し歴史的な流れを説明したほうがわかりやすいかもしれません。

コンピュータの発展の歴史は，小型化の歴史でもあります。最初にできたコンピュータは ENIAC といいます。1946 年の話です。この ENIAC は 27 トン，広さにして倉庫 1 つ分 ($167m^2$，90 畳以上の広さ) が必要なもので，真空管を使った計算機でした。軍事的な計画のために開発されたオーダーメイドのもので

[*2] 実はこのように人間を 1 つのコンピュータに喩えて，そこではどのように計算がされているのか，人間の記憶装置や演算装置，入出力装置の特徴はどうなっているのか，というのを研究するのも心理学の仕事です。記憶や演算，制御については認知心理学や学習心理学，入出力については知覚心理学や生理心理学が専門的に扱っています。そういう意味でもコンピュータの登場は心理学に大きな影響を与えているのですが，それはまた別の講釈で。

[*3] 押すときの音から，この操作をクリック click と言います。2 回続けて押すことをダブルクリックと言います。

[*4] 実は人間の意識もこのユーザーイリュージョンのようなもので，実際の体の動かし方や感覚情報の受け止め方，処理の仕方は別ですよね。すべての情報を意識に上げるのではなく，「私は XXX をしている」と身体がそれっぽい幻想を投影して見せてくれているのが意識の正体ではないか，という議論があります。興味のある人は Norretranders (2002) を読んでみてください。

すから，一般人が触れるはずがないものです。時代が下がって真空管が半導体，IC チップになった頃，やっとサイズが小さくなって，家庭用・個人用のコンピュータというのができるかもという時代が来ました。Apple コンピュータの創始者，スティーブ・ジョブズとスティーブ・ウォズニアックが最初のパソコン (Personal Computer,PC)，Apple I を発売したのが 1976 年。爆発的に売れた Apple II が発売されたのが 1977 年です。AppleII はブラウン管表示装置とキーボードを持っていましたので，今の PC の原型とも言えるかもしれません[*5]。PC を作っている会社は Apple だけでなく，IBM や DEC などがありましたが，まだこの頃はパーソナルなレベルのものよりも大型計算機の開発が進んでいました。IBM が PC を作ったのは 1980 年で，この頃から小型化が進められていきます。

私事で恐縮ですが，1976 年に生まれた私が初めて PC を手にしたのは 1991 年，高校入学のお祝いで買ってもらった Fujitsu の FM-Towns という機体でした。この頃は「マルチメディア」という名前もなく，「ハイパーメディア」と読んで売り出していました。この機体は他の PC(NEC の PC-9801 や Sharp の X68000 シリーズが有名でした) とは違って，CD-ROM ドライブをつけていたことが画期的だった時代です。その後 1994 年に大学生になりましたが，この頃は連絡を取り合うツールはポケベルが主流であり，携帯電話 (や PHS) のような個人端末は高級品という時代でした。大学に入るとコンピュータを学ぶ授業があり，アカウントをもらったりするのですが，それは大学が持っている大型計算機に端末からアクセスするためのものでした。いろんな部屋にあるのは「端末」で，それほど機能の優れた PC ではなく，複雑な計算 (統計的な計算など) は大型計算機に仕事を依頼しその返信を待つ，というスタイルでした。関西私立のマンモス校でしたので学生数は非常に多かったのですが，多くの学生が一度にアクセスしても，大型計算機はものすごくものすごーく計算が早かったので，瞬時に回答をもたらしてくれるものでした[*6]。つまり，まだ「専門的な計算は大型計算機」という時代であり，パーソナルなコンピュータになるにはもう少し時間が必要でした。

どれぐらいの時間が必要だったかというと，実はその次の年なのです。1995 年，Windows 95 という OS が発売されました。これを機に日本でも PC がどんどん浸透していくことになります。Windows95 は物凄いんだぞ，と発売前からテレビでも散々とりあげられ，発売日には行列ができて真夜中のカウントダウンと同時に大フィーバー，という売れ行きでした。当時のそれは何が凄かったのでしょう？ コンピュータにはそれ動かす基本ソフトが必要です。HDD にデータを書き込み，キーボードからの入力をディスプレイに表示する，といったごく基本的な装置を統括し，メモリ番地をファイルという単位で扱うと言った基本的な操作は OS いうソフトウェアが担当します (図 C.1)[*7]。この OS，大型計算機は Unix と呼ばれるものを使っていましたが，各企業が個人向けにコンピュータを売り始めるときにも当然必要で，各社で開発もしていましたが，PC の共通規格をつくることで OS 部分は共有できるようになりました。そこを提供したのが Microsoft 社のビル・ゲイツです。どんなパーツで作られた PC であっても Windows という OS が共通のフィールドを用意してくれるので，ユーザは Windows で動くアプリケーションを選ぶだけで良い，ということになったのです。

そして Windows95 は，GUI，つまり「ファイルを掴んでポイ」といった直感的な操作で使えることも大きな特徴でした。大型計算機で使われている Linux は基本的に Command User Interface,CUI で，黒い画面にプログラムを書いて実行するといった手法で，初心者には人気がなかったのです。GUI については，その頃 Apple を追放されていたスティーブ・ジョブズが，NeXT という会社で GUI を備えた OS を開発していました。この NeXT はその後 Apple に買収され，ジョブズは Apple に戻って活躍することになります。その頃から巷では，Windows の GUI は Apple の OS を真似したものだと非難されていたのですが，商業的には Windows が大勝利，というわけです。ともかくこれを機に企業などはもちろん一般家庭でも PC を使うよ

[*5] Mac の歴史については Walter (2012) が読み物としておもしろいですよ。
[*6] Time Sharing System,TSS，時分割システムとよばれる機構です。命令を小さな単位に分割し，それを順次捌いていくという方法でした。
[*7] 物理的な機構と OS との間に Basic Input/Output System,BIOS というのが入りますが。

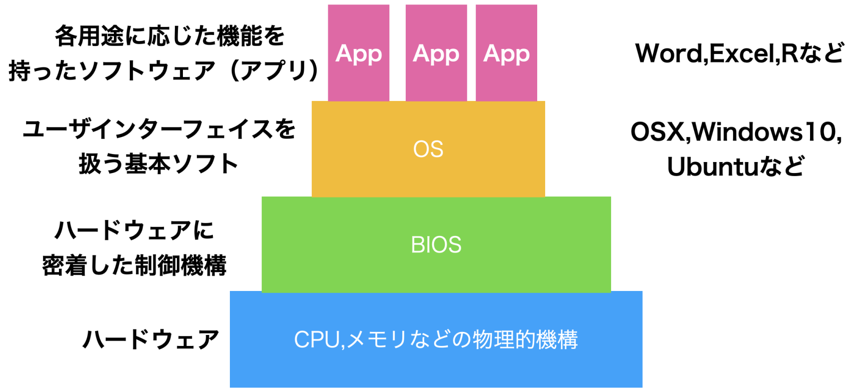


図 C.1 コンピュータのハードウェアとソフトウェアの関係

うな時代になりました。

ちなみにインターネットが広まったのもこの頃です。私は大学 3 年生の時，大学の授業で初めてインターネットを介して世界の情報を得る，という経験をしました[8]。その頃から徐々に，大型 PC のアカウントではなくインターネットで使えるメールアドレスというのを個々人が持つようになりました。携帯電話も廉価な PHS が広まり，メッセージだけでなく徐々に音声，写真，短い動画が遅れるようになっていきます。当初は当然画質・音質も今とは比べものにならないのですが，それでもインターネットを経由してつながるという経験は想像を超えたものでした。

皆さんはすでに，携帯電話やインターネットがある時代に生まれた世代だと思います。こんな話はすでに過去のものであり，不便な頃の話を聞かされても困る，と思うかもしれません。私の世代は，幸いにもこのようにちょうどコンピュータが使われはじめ，広がり，高性能になっていくにつれて育ってきていますので，そのぶん機械の根本的な理解に直結しやすい社会環境にあったのです。みなさんは，便利な時代ではありますが，言い換えると「初心者が苦労しないように補助輪をつけておく」「何もしなくてもできているような感覚が得られるようなイリュージョンを見せておく」という状態におかれているので，補助輪が対応できない道に進んだり，多くの人とは違う使い方をするときに，急にサポートが外れどこで困っているかわからなくなる，ということになりかねません。非常に大雑把な Historical Review ではありますが，理解の一助になればと思います。

ところで，先ほどサポートという言葉を使いましたが，このサポートを商売にしているのが Microsoft や Apple という IT 企業です。これらの会社は，ユーザが使いやすいようにソフトウェアを提供してくれますが，有償ですし想定外の使用をするユーザに対してはサポートをしてくれません。逆にいうと，「決められた路線を走らなければ面倒を見ない」ということでもあり，これはユーザに不自由を強いているとも言えます。また，どのような仕組みで動いているのかを尋ねても，企業秘密と言って答えてくれません。コンピュータは誰でも自由に使えるべきものであり，勝手にユーザの情報を盗んだりしていないか，と言ったチェックをするためにもオープンであるべきではないか。そういう考え方に基づく，フリーソフトウェアというソフトウェアのあり方があります。Unix という OS を PC 用にした Linux がそうであり，オフィスソフトの LibreOffice，統計環境の R などもこの精神に賛同するものです。フリーソフトウェアは自由であり，無償です。お金と秘密を払ってサポートを受けるのではなく，ユーザが相互に助け合ってオープンで自由な世界を広げていこうという活動です。急に何の話なんだ，と思うかもしれませんが，心理学ひいては科学的活動すべてにおいて重要な問題であるこ

[8] 教職関係の科目で，教育技術として今後使われるだろうということで担当教員が実演してくれました。隣の部屋から電話線を延長し，モデムというパソコンの信号を音情報に帰る機器を繋げて，NASA の Web ページを Netscape というブラウザでみたのが初めての体験でした。

とをご理解いただきたいと思います。

C.4 情報の単位

コンピュータを取り巻く世界の話はこれまでにして，ソフトの側面についての解説に入りましょう。

コンピュータは文字，音，絵，動画ファイルいずれについても，すべて 0 か 1 のデータとして管理します。0/1 の 1 つの単位を 1bit(ビット) と言います。1bit であれば Yes か No か，という二択の情報しか提供できませんが，これが 7 つあれば $2^7 = 128$ ですから，これで 128 種類の状態を表現できます。コンピュータは一般に，8bit で 1 つの単位として計算します。8bit のことを 1byte(バイト) と言います。この 1byte が 1024 集まったものを 1kb(キロバイト) といいます[*9]。1kb の次は，1024kb=1Mb(メガバイト) です。さらに 1024MB=1GB(ギガバイト) で，1024GB=1TB(テラバイト)，1024TB=1PB(ペタバイト)，1024PB=1EB(エクサバイト) と続きます。K,M,G,T,P,E といった名称は 1000 倍ごとに変わる大きさの桁をあらわしているのであって，「ギガが減る」というのは本来意味をなさない表現です[*10]。

このビット･バイトは情報に関する基本単位なのであちこちに使われます。記憶装置について使われるとき，一次記憶装置も二次記憶装置も同じ単位なので混同するかもしれませんが，2021 年現在では二次記憶装置の単位は GB から TB が使われます。USB フラッシュメモリーや外付け HDD などは数 TB の容量が一般的でしょう。これに対して，一次記憶装置は 4〜64GB ぐらいが相場かと思います。一次記憶装置は暗算の途中経過のように一時的に記憶する場所に過ぎないので十数 G でも問題ありませんが，二次記憶装置は結果の記録なので大きければ大きいほど余裕が持てますね。たとえば数 TB でもテレビドラマや映画を何本も記憶できるのですから，PB や EB なんて使うのかな，と侮ってはいけません[*11]。すぐにそれぐらいのサイズが必要な時代が来ることでしょう。

実際，それぞれ単位ではどれぐらいの情報が記録できるのでしょうか。1byte は 128 文字表現できるので，英語のアルファベット 26 文字に加え，数字や簡単な記号であれば 1byte で表現できます。たとえば A という文字は 01000001，B という文字は 01000010，小文字の a は 01100001，と言ったように 0/1 の文字列 8 個と一対一対応させて考えるのです。日本語は 1byte では足りませんので，一文字あたり 2byte が割り当てられます。また 1KB(=1024byte) は 500 文字ですから，原稿用紙一枚ぐらいになります。1MB(=1024KB) は文字だけだと新聞一紙 (朝刊の 40 ページ分) ぐらいで，昔の記録媒体であるフロッピーディスク一枚に保存できるのがちょうど 1MB でした[*12]。ちなみに「カメラ映像 + 音声」のオンラインビデオ会議を 1 時間やると 200〜300MB ぐらいの容量をやりとりしていることになります。ビデオ会議では文字 (チャット) だけでなく，画像 (動画) や音声も送り合いますね。実は画像や音声も，0/1 に置き換えています。音声の場合，1 秒間を短い間隔にくぎります。この区切りのことをサンプリング周波数といい，たとえば 44.1 kHz という単位は 1 秒間に $44.1 \times 1000 = 44100$ 点のデータの採取をします。この 6 データ点において，音の振動幅を区切ります。ビットレートと言いますが，たとえば 16bit で区切る場合は $2^16 = 65,536$ 段階で区切り，その高さの音がある (1) かない (0) かで表すわけです。このように時間と音階を細かく区切り，その目

[*9] キロメートルやキログラムのように，キロは 1000 の単位を表す言葉ですが，コンピュータは 2 進数なので 1000 ちょうどではなく 1024 で 1 つ上の位に上がることになります。

[*10] 同様に「USB を紛失する」というのもよく聞くおかしな表現です。USB は Universal Serial Bus の略で，データ転送規格のことを指します。なくすことができるのはそれにつなげるメモリースティックなどです。

[*11] 私事ですが，私が 1991 年に生まれて初めて手した PC，FM-Towns はハードディスクが 40MB，RAM は 2MB で，CD-ROM がついていましたが，それは 360MB の容量でした。購入するときに，「ハードディスクが 40MB もあって何を記録するんです？ CD-ROM なんか情報が詰まり過ぎてますよ！」と店員さんに笑われたのを今でも覚えています。数年後，PC の動きが遅くなったので，2 万円で 2M の RAM を追加したのも良い思い出です。今でこそ，2MB なんて USB フラッシュメモリーでも売ってないほど微小なサイズですが。

[*12] 正確には 1.2MB 入る規格 (2DD) と 1.44MB 入る規格 (2HD) とがあります。

に情報があるかないかを積み重ねてデータとするわけです。テキストよりも圧倒的に情報が多くなるのが分かりますね。画像も同様に，図面を細かい単位 (ピクセルなど) で分けて，そこの色合いを色々な段階で区切ります。色は R(赤)G(緑)B(青) の組み合わせで表現でき，それぞれを 8bit= $2^8 = 256$ 段階で表現したりします。一点一点にその情報がありますから，図の情報も非常に多くなるのがわかると思います。動画はその画像が時系列的に細かく分割されたものと思ってください。このように分解しますので，テキスト，音声，図，動画の順にデータサイズが大きくなります。通信機器が最初ポケベル＝数 byte の情報しか送れなかったものから，徐々に絵文字，ショートメッセージ (音声)，写真[*13]，動画が送れるように発展していきました。今では町中のあちこちで，誰もが手軽に動画をモバイル端末で見られるようになっています。あらためて，すごい進歩ですね[*14][*15]。

ところで，1byte は 256 種の情報が記録できるので，英語のアルファベットや数字は 1byte あれば十分だが，日本語や中国語など，英語以外の言語は文字種が多いので，2byte で一文字を表すという話をしました。この 2 バイト文字も，たとえば「あ」とか「亜」という文字に 111000111000000110000010 とか 111001001011101010011100 という文字列を割り当てるのですが，言語ごとによってどの数字をどの文字に割り当てるかという対応表が変わってきます。これを文字コードと言います。日本語はかつて Shift-JIS というコードで変換していましたが，今は世界のあらゆる言語に対応している共通企画である，UTF-8 という文字コードで変換することが一般的です。ところがなぜか，日本の Windows OS だけ Shift-JIS をいまだに使い続けており，他の PC とファイルをやり取りするときに文字コードの変換エラー問題が起きます。ファイルを開いて文字化けをしたりとか，プログラムが実行される際に「ファイルにアクセスできない」というエラーが生じたりするのは，この文字コードの問題が大きいのです[*16]。受身的な対策法になってしまいますが，PC でつかうユーザ名やファイル名などは半角英数文字を使い，短くした方がこうしたエラーに出くわしにくくなります。逆に，全角文字やスペース (空白) などを含んだファイル名，やたらと長いファイル名を使っていると，こうした問題に出くわしやすくなるということです。

C.5　ファイルの種類と拡張子

ここまで述べてきたように，計算機というのは基本的に物理的実体 (記録装置，記憶装置) の上で 0/1 のデータをやり取りしているだけです。記録 (記憶) 装置上に置かれている情報のセットは「ファイル」という形で記録されています。スマートフォンやタブレットは，ユーザの利便性のためにファイルの存在を意識しなくても良いようになってはいますが，バックエンドでは実行されるアプリケーションもファイルですし，開かれる音声や動画もファイルです。とくにパソコンでは，どの媒体，どのアプリで使うどういうファイルかを識別するために，拡張子 (かくちょうし) と呼ばれる識別記号をファイルの後ろにつけています。拡張子はファイル名の背後にピリオドで区切って追記されています。ついてないように見えても，OS がそれを表示させない設定にしてあるだけであることに注意してください。代表的な拡張子と，それに対応づけられているアプリケーションは次

[*13] できた当時は写真を撮ってメールができることをとくに「写メールする」と言い表したほどです。

[*14] 実は音でも画像でも，分割してそのままデータにしてしまうと膨大になりすぎるので，人間が気付きにくい周波数や色合いなどは削除して作ります。これを非可逆圧縮処理と言います。一度落としてしまった情報は戻らない，という意味です。ライブや生きている人間が処理している情報は，携帯の画面から得られるものの何億倍もの情報量なんですよ!

[*15] デジタル化のすごいところは，こうした文字，音，図版，動画といったものを bit という共通の単位に落とし込んだことです。こうすることですべて一元的 (bit という共通次元) で処理することができるようになったのです。メディアの違いが問題にならなくなり，0/1 の情報であれば複写も簡単にできてしまいます。情報化社会においては情報に特別性はなく，情報があるかないか，それを生み出せるかどうかこそが重要なのです。

[*16] Windows だけ世界標準から外れているので，早く修正して欲しいのですが，歴史的な経緯からユーザ数が多くて切り替えられないでいることと，こうした違いがあることをユーザに説明しない (素人は知らなくていい，と馬鹿にされているようなものです) ので，問題が解決される日はまだ先になりそうです。

のようなものがあります。

- .docx　マイクロソフト社の文書作成アプリケーション，Word で使うファイル
- .xlsx　マイクロソフト社の表計算アプリケーション，Excel で使うファイル
- .pdf　Adobe 社が開発した Portable Document Format 形式。OS が違っても同じレイアウトで文書を表示できるのが利点で，PDF 形式を読むことができるアプリケーションは多数。
- .txt　シンプルな文字だけのテキストファイル。文字の飾りやレイアウトなどの情報がない最もプレーンな形式なので，OS が違っても文字コードさえ合っていれば読むことができる。
- .mp3　音楽，音声のファイル。音声データには他の種類もあります。
- .jpg　画像のファイル形式の一種。
- .png　画像のファイル形式の一種。
- .csv　comma/character separated variables ファイル。変数をカンマ (,)，あるいはタブ，半角スペースなど文字コードで区切ったファイルという意味で，中身は.txt と同じく装飾のない文字/数字だけであり，文字コードさえ間違えなければ OS を問わずに読み書きできる。データのやり取りはこの形式で行われることが多い。
- .zip　圧縮ファイルの一種。1 つまたは複数のファイルをパックして圧縮してあるもの。ファイルの冗長な部分をうまくまとめてコンパクトにまとめ上げるため，ファイルサイズが小さくなるし，複数のファイルもひとまとめにできる。また圧縮の際にパスワードをかけることもできるため，メールなどに添付する場合はこの形式にまとめられることが多い[*17]。可逆圧縮であり，圧縮されたファイルは展開する (解凍するともいう) ことでパッキングを開封できる。zip ファイルの圧縮/展開は各種 OS が標準的に対応している。

`.txt` や `.csv` といった形式は，「装飾のない，文字だけの」ファイルです。こうした種類のことを ASCII ファイルと言います。メモ帳などのエディタと呼ばれるプログラムで読み書きできます。逆に `.docx` など特定の会社が提供するアプリケーションに対応しているファイルは，アプリの中でのさまざまな操作・装飾を暗号化して保存しており，メモ帳などで読んでも意味がわかりません。対応しないアプリでは開くこともできません。こうした形式は ASCII ファイルに対してバイナリファイルと言います。

OS は拡張子を見てファイルの種類を判別し，そのファイルを開くのに適したアプリケーションを自動的に起動し，開いてくれます[*18]。`.csv` ファイルは Excel などのアプリケーションで開くことも当然できますが，その際文字コードのエラーが生じたり，保存するときに文字コードを変えたりして，形式・内容が気づかずに変わっていることがあります。Windows も良かれと思ってやっていることなのですが，処理が徹底してないのか，かえって不便になってしまっています[*19]。

[*17] **PPAP** というとピコ太郎の楽曲を思い出す人もいるかもしれませんが，圧縮ファイルの文脈では "「Password つき zip ファイルを送ります。Password は次のメールで送ります」Angoka(暗号化) Protocol の略です。つまりメールでパスワード付きのファイルを送り，そのファイルを開くためのパスワードをまたメールで送るという，日本でよく見られるおかしな風習です。おかしな，というのは，メールがハッキングされていたらパスワードもどうせバレるわけで，同じメールに書いてあるのはもちろん馬鹿馬鹿しいですが，すぐ次のメールに書いてあるのも同じぐらいに馬鹿馬鹿しいことです。情報セキュリティ対策手法のつもりで行われる慣習が広まっていますが，PPAP の標語のもと，馬鹿馬鹿しいのでやめましょうという風潮になってきました。

[*18] 見たことのない拡張子の場合は，どのアプリケーションで開くべきか尋ねてくるでしょう。

[*19] 実際，この授業での課題データを UTF-8 形式の csv ファイルで提供しても，Excel で開いたばっかりに文字化けして分析できなくなる，という相談がこれまで多く寄せられています。根本的な解決策として，Windows を使うのをやめることをお勧めします。

C.6　クラウドとは

すでに述べたように，計算機は基本的に 0/1 データのやりとりであり，それを保存してあるのがファイルとよばれるものです。ファイルは HDD や USB メモリ，SSD に保存することができます。

ところで，最近はこうした手元の物理的実体にファイルを置くことに加えて，クラウドに保存することも少なくありません。クラウドとは雲という意味で，インターネットの向こう側のどこか，ということを意味します。ですが，基本的にはインターネットで繋いだその先にも電子計算機があるのです。たとえばパソコン A とパソコン B をケーブルで繋ぐと，パソコン A からパソコン B のファイルにアクセスできます[*20]。このケーブルをどんどん伸ばすと，遠く離れていてもこの操作ができます。このケーブル網を世界レベルに広げているのがインターネットです[*21]。

ここで覚えておいて欲しいのは，当たり前のようですが，ネットといっても基本的には電子計算機と電子計算機を繋げている実体がどこかにあって，ファイルのやりとりをしているだけだということです。ブラウザはウェブサイトの情報を書いたファイルを取り込んでホームページを見せてくれていますし[*22]，Youtube は動画のファイル，Instagram は画像のファイルへのアクセスをして見せてくれているのです。最近ではクラウドサー

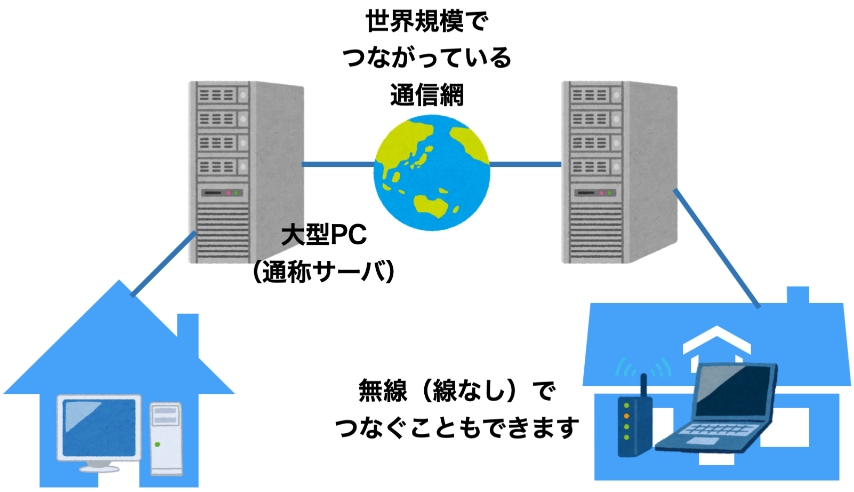


図 C.2　インターネットの世界

ビス，というのがよくあります。中でも Dropbox とか OneDrive，iCloud などが有名です。これらは，ファイルをインターネットを介した別の大型電子計算機に保存 (アップロード) し，インターネットを介して読み込み (ダウンロード) して使う，というものです。手元の電子計算機の中に記録されているファイルのことを「ローカル」，サーバなど遠隔地にある大型機に記録されているファイルのことを「サーバ」「クラウド」と呼んで区別しますが，このローカルとサーバのファイルを常に同じものに同期しておく便利なシステムです。こうしたクラウドサービスを使うと，たとえば大学で作業して保存したファイルを，USB メモリに保存して自宅のパソコンにコ

[*20] もちろんファイルの情報をどのように信号に変えて送受信するか，アクセス権限はどうするかといった細々したことを調整しなければなりませんが，そのあたりの仕事をやってくれるのが OS のありがたいところです。

[*21] インターネットは軍事通信網として始まりましたが，それを電話回線や企業内通信網，大学間通信網などと接続しあって世界中に広げています。網と網が相互に (inter) 繋がっているので inter net であり，大学内･企業内のネットワークのことはイントラ (intra) ネットと言います。ちなみに一般用語としてのインターネットは Internet と大文字で書き始めます。

[*22] ホームページとは本来ブラウザが最初に開くページのことを指し，企業や個人が情報発信しているページのことはウェブサイトというのが適切です。

ピーする，という操作をしなくても，大学でクラウドサービス上のフォルダ内に保存しただけで，自宅のパソコンにそのファイルがコピーされているため[*23]，意識せずにそのまま作業を続けられるということです。自動的にバックアップをとってくれているとも言えるので便利ですね。

もちろん注意すべきこともあります。「他の人に見られたら困るファイル」，「アプリケーションの実行ファイル」などは通信網の向こうではなく，手元 (ローカル) に置いておきましょう。個人情報・機密情報などを，クラウドドライブに保存すると，悪意を持った人が大型計算機に攻撃を仕掛けて情報を盗んでいく可能性があるからです。情報化の怖いところは，取られても気づかない (コピーすればいいだけで元ファイルになんの影響もない) ところにあります。加えて，取られたものをばら撒かれる＝インターネットを介して誰でもアクセスし保存できるようにされると，すべてを回収できなくなるのも問題です。失言が記録されて拡散されると大変なことになるのは，みなさんもこの時代に生きる人間のマナーとして色々見聞するところと思います。また，クラウドサービスで自動的に同期されるといっても，アプリケーションの実行ファイルなどはローカルに保存するべきです。アプリケーションは実行に際してさまざまな関連ファイルにアクセスしますので，1 つでも場所が違うところがあるとエラーになって動かなくなります。同じ OS でも，です。インターネットからとってきたソフトウェアをうっかり OneDrive に保存してしまうと，アクセスできないエラーで起動しない，ということもありますので注意してください[*24]。

C.7 ファイルの位置の指定

ここでファイルとそのパスについての話をしておきたいと思います。

計算機が情報を 0/1 で管理し，それらがファイルとなってどこかに保存されている，ということでした。私たちは Finder や Exploer などファイルブラウザをつかって，実行したいファイル，参照したいファイルを探していきますね。ファイルはまとめてフォルダの中に含まれていますし，フォルダの中にフォルダがあるといった，階層状態になっていることも少なくありません。ちなみに**フォルダ**と同じ意味で**ディレクトリ**という言葉が使われることもあります。

C.7.1 相対パスと絶対パス

普段 PC を使っているときは気にすることがありませんが，R や RStudio などプログラミング言語をつかっているときは，「今どこで作業しているか」という現在地が重要になってきます。たとえば，RStudio で `C:\User\kosugitti\Document\kiso1\` というところでプロジェクトを開いているとします。スラッシュ (`\`) はフォルダ，コロン (`:`) はドライブを表す記号です。プロジェクトフォルダは，C ドライブの User フォルダの下にある，kosugitti フォルダの下にある，Document フォルダの下にある，kiso1 というフォルダということになります (プロジェクト名が `kiso1` だとそうなります。)。この kiso1 フォルダが現在地です。

このフォルダの中で，Rmd ファイルや R スクリプトファイルを使って，他のファイルを参照するようなコードを書くとしましょう。たとえば `script1.R` というファイルに `read_csv` 関数を書いたとします。読み込みたいファイルは，同じフォルダの中にある，`sample.csv` とだとします。このとき，`read_csv` の書き方は次のようになるでしょう。

[*23] これはユーザが特段の指示をしなくても，アプリケーションがユーザの見えないところで (バックグラウンドで) アップロード，ダウンロードの作業を進めているからです。パソコンはシャットダウンしていなければ，裏でこうした作業を淡々とこなし続けてくれています。

[*24] マイクロソフト社は，これまたユーザのためを思ってやっているのかもしれませんが，デフォルトで OneDrive に保存させようとします。それでうまくインストールできなかったという相談も多々寄せられています。抜本的な解決策として，Windows を使わないことをお勧めします。

code : C.1　相対パスで読み込む

```
dat <- read_csv("sample.csv")
```

しかし別の書き方もあります。たとえば code:C.2 のような書き方でも問題ありません。

code : C.2　絶対パスで読み込む

```
dat <- read_csv("C:\User\kosugitti\Document\kiso1\sample.csv")
```

後者 code:C.2 の書き方は，ファイルの場所を全部書いてありますから，確実にその場所が特定できます。それに比べて前者 code:C.1 の書き方は，なぜファイルを書いただけでいいのでしょうか。これは，このコードを実行している現在地と同じフォルダの中に `sample.csv` ファイルがあるからです。プログラムは，命令を受けるとファイルを探しにいきますが，現在地と同じフォルダの中を探すことになっているのです。この現在地，すなわち現在作業しているフォルダのことを**作業フォルダ (working directory)** と言います。

では作業フォルダと別のフォルダの中にファイルがあれば，アクセスできないのでしょうか。そんなことはありません。code:C.2 の書き方を使えば，作業フォルダがどこにあっても位置を特定できますから，作業フォルダを問わずに書くことができます。ちょっと長くて面倒ですが，確実にある場所を指定しているからです。この書き方のことを**絶対パス**による指定といいます。一方，code:C.1 の書き方は，今の作業フォルダから見た場所，という相対的な書き方になっています。この書き方のことを**相対パス**による指定，といいます。相対パス指定で，違うフォルダにアクセスする場合には，次のようにします (code:C.3)。ここでは，`Document` フォルダの中に，kiso1,kiso2 フォルダがあり，kiso1 フォルダの中で作業している時に kiso2 フォルダの `sample2.csv` ファイルを読み込む例を書いています。

code : C.3　絶対パスで読み込む

```
# 絶対パス指定
dat <- read_csv("C:\User\kosugitti\Document\kiso2\sample2.csv")
# 相対パス指定
dat <- read_csv("..\kiso2\sample2.csv")
```

絶対パスはそのままなのですが，相対パスは `..\` という記号になっていますね。このピリオドを 2 つ打つ方法で，「ひとつ上のレベルの」という意味になります。このように，現在地からの相対的な位置関係で，ファイルを指定することもできます。

絶対パスと相対パスのどちらが良いのか，というのは一概には決められません。絶対パスは，PC が変わったりフォルダの構造が変わったりすると役に立ちませんから，使い勝手が悪いと言えなくもないですが，確実に指定できる方法です。相対パスは，PC が変わったりしても「現在地から相対的に見てどこか」という話ですから，たとえばこの例で `kosugitti` フォルダごと別の場所に移しても (たとえば `D:\Univ\Classes\kosugitti\kiso1` のように)，コードはエラーなく動きます。kiso1 フォルダ，kiso2 フォルダの相対的な位置関係が変わらなければいいのですから。バックアップを取ったり，複数の環境で同期しながら作業する場合などは相対パスの方がいいでしょうね。

いずれにせよ，現在どこで作業しているかということ，すなわち作業フォルダの場所を，意外と意識しておかなければならないということには注意が必要です。ファイルをどこに置いたか，どんなファイルを置いたか，自分はどこにいるのか，これが変わってくると「ファイルが見つかりません」というエラーになるのです。言い方を変えると，**ファイルが見つかりませんエラーの原因は，この 3 つのどれかである**ことがほとんどです。

C.8 ファイルのバージョン管理

これからみなさんは大学生活の中で，たくさんのファイルを生み出していくことになるでしょう。たとえば 4 年生の時に卒業論文を書くことになりますが，データファイル，分析ファイル，図を書いたファイル，引用文献リストを書いたファイル，卒論本文などなど，1 つの研究でも複数のファイルが作られることはよくあります。さらに，これらのファイルは日々加工されますから，その度に上書き保存することになります。いわば，ファイルがバージョンアップしていくのです。

卒論などの場合はとくに，「途中で保存しておく」ことが重要です。途中まで書いていた時に，横に置いていたマグカップが倒れて PC にコーヒーがかかり，変な音を立てて PC が壊れてしまった，ということがあるかもしれませんからね。紙に書いていた時代は，その手のハプニングがあってもせいぜい原稿用紙数枚がダメになっただけで，「ちくしょう，やりなおしかぁ」で済んだのですが，電子データの場合は電子の藻屑になると復元させることができません。ですから**バックアップは非常に大事**なのです[*25]。

バックアップの基本は，「別の場所に」「別のファイル名で」というものです。同じ名前で上書きすると元に戻ることができませんから，面倒でもコツコツと違う名前をつけましょう。そうするとよくあるのが，`soturon1.docx,soturon2.docx,soturon3.docx,soturon3(`**修正**`).docx,soturon3(`**最新**`).docx,soturon3(`**最新**`)(`**修正**`).docx,soturon3(`**最新**`)(`**修正**`)(`**提出版**`).docx,soturon3(`**最新**`)(`**修正**`)(`**提出版** 2`).docx`.... というようになっていくやつで，「どれが最新版だっけ...」と書いてる本人でも探すのに苦労することになります。

この問題の解決策として，`soturon1001.docx`，`soturon1005.docx` のように日付を入れるというものがあります。10 月 1 日分，10 月 5 日分，としていけば「いつまで戻れば良いか」もわかるのでいいやり方ですね。日付の数字をファイルの先頭につけておくと，並べ替えも簡単です。この日付をつけて保存するというのを習慣化し，1. 昨日までのファイルを開いて今日の日付で別名保存する，2. 作業を進めて，時々上書き保存，最後にも上書き保存，3.PC に USB メモリをさして，バックアップ保存して作業終了，というルーティンを作っておくと，確実に記録が残って良いでしょう。

ただし，この方法の問題は，ファイルサイズが大きくなりすぎることです。図表などを含めたファイルが数百 MB になることは少なくありません。それを次々複製していくわけですから，大容量の USB メモリでも限界が来るかもしれません。これは「丸ごとコピー」していることが原因で，たとえば昨日は 10 行目まで書いた，今日は 11 行目から 14 行目まで書いた，というのであれば，この増えた 4 行分 (差分) だけを追加保存すればいいのに･･･と思いませんか。

こうしたバックアップやバージョン管理をやってくれる仕組みとして，`Git` というものがあります。`Git` は作業フォルダの中身の差分だけを記録し，必要であれば過去のバージョンに戻すこともできるシステムです。毎回上書き保存 (`commit` する，といいます) のたびに「どこを変更したか」というメモを付けて保存しておけば，そのメモを見ながら「ここの時点まで戻ろう」という使い方をすることができます。ファイル名は変更する必要なく，同じファイル名で進めていけますから，同じようなファイルがたくさんあって訳がわからなくなるということもありません。さらに保存先をクラウドにした `GitHub` というものもあり，これを使うとクラウド上に追記していくことができます。この `GitHub` は IT 企業などでプログラムをチームで進めていく時にも使われている技術で，全体のプログラムに個別の機能を複数人が追加，管理者が必要なものだけ取り入れる，というように使われています。国里ゼミや小杉ゼミでは，卒論を GitHub で管理し，学生が書いた分を commit し，それを

[*25] ちなみに私の経験上，レポートが電子の藻屑になったので助けてください！と言われることがよくあり，4 年間の学生生活の間では平均 10–15 名に 1 人の割合で発生することのようです。

hub 上にアップロード (push といいます) する，というようにします。教員の方は学生の進捗が管理できますし，どこがどう変わったかが分かりやすく，バージョン管理と同時にバックアップもできるという便利な仕組みです。GitHub は無料でアカウントを作ることができますから，興味があれば皆さんもチャレンジしてみてください[*26]。

さてここで，ひとつ注意をしておきます。卒論の原稿やプログラムは日々変化するものですからバージョン管理が必要ですが，データファイルはアップデートする必要がありません。いや，アップデートしてはおかしいのです。たとえば 100 人分のデータを取って分析をしていて，後で「やっぱりこのデータを削ろう」というのは，研究不正が疑われかねません。自分に都合の良いデータだけで議論し，都合の悪いデータは削除して統計的に有意な結果が出るように細工しよう，なんてことがあれば困るのです。データファイルはバージョンアップせず，それを加工，計算するプログラムがバージョンアップしていく。そしてその加工プロセスは誰でも見ることができるように公開されている。少なくとも，科学的な営みをする上では，そうしたやり方が必要なのです。自分だけのデータで自分だけの分析方法で，良い結果だけ示すというのは適切な方法ではありません。

Open Science Framework(https://osf.io/) はこうした「オープンな科学」にむけた取り組みの一種です。このアカウントは誰でも無料で作ることができ，ここにファイルをアップロードしたり，分析計画を事前に記録しておくことができます。何も難しいことではなくて，クラウド上のファイル置き場だ，というぐらいに思っていただければ結構です。ここにおかれたファイルは自動的にバージョン管理され，同じファイル名のものがアップデートされてもその記録 (ログ) が残ります。最近はこの OSF をつかって，論文化されたデータやプログラムを公開するという取り組みも進んでいます。

クラウド，バックアップ，オープンサイエンスといった新しい研究方法は日々生まれてきています。皆さんも便利な機能はどんどんキャッチアップしていきましょう！

C.9　おわりに

古臭い話をしてしまうのは，私が歳をとった証拠でしょうか。皆さんはこんな話を知らなくても，スマホやタブレットを使いこなしていることと思います。細かいことを知らなくても，ユーザとして利用するだけなら知らなくて良い話なのかもしれません。私はここにも書いたように，高校生の頃からコンピュータの発展と一緒に大人になってきましたから，学ぶともなく学んできたところがあります。皆さんは生まれた頃からコンピュータやがあったネイティブ・デジタル・カウボーイですから，苦労なんかする必要なかったわけです。

しかし細かい仕組みを知らないということは，問題が生じた時に「何か・誰かが，どこかでどうにかなって，今私が困っている」という状況になります。問題を特定できないと，解決することもできません。コンピュータは文房具に過ぎませんから，それを使いこなせないほうが格好悪いのです。しかも今後ますますコンピュータに囲まれた世界になっていくのは自明ですから，ここに学習コストをかけない方が勿体無い。幸い，わからないことに対して，自ら調べて学んだ利する時間と環境が用意されているのが大学という世界なのですから，今のうちにしっかり基礎固めをしておきましょう。

ここのくだらない懐古的エッセイのような文章が，何かの足しになれば幸いです。

[*26] このテキストやシラバスも GitHub で管理していますし，公開されているサイトも GitHub 上のものです。これからは教科書も日々成長していくものになるかもしれません。

9407 付録 D

9408 ギリシア文字一覧

ギリシア文字ってかっこいいけど，読み方わからない・・・という人のための一覧。ついでに $\TeX$ 表記も。

表 D.1　ギリシア文字とその読み方

読み方	大文字	小文字	英語表記	$\TeX$ 表記
アルファ	A	α	alpha	`\alpha`
ベータ	B	β	beta	`\beta`
ガンマ	Γ	γ	gamma	`\gamma`
デルタ	Δ	δ	delta	`\delta`
エプシロン	E	ε	epsilon	`\varepsilon`
ゼータ	Z	ζ	zeta	`\zeta`
イータ	H	η	eta	`\eta`
シータ	Θ	θ	theta	`\theta`
イオタ	I	ι	iota	`\iota`
カッパ	K	κ	kappa	`\kappa`
ラムダ	Λ	λ	lambda	`\lambda`
ミュー	M	μ	mu	`\mu`
ニュー	N	ν	nu	`\nu`
クサイ	Ξ	ξ	xi	`\xi`
オミクロン	O	o	omicron	`\mathrm{o}`
パイ	Π	π	pi	`\pi`
ロー	R	ρ	rho	`\rho`
シグマ	Σ	σ	sigma	`\sigma`
タウ	T	τ	tau	`\tau`
ウプシロン	U	υ	upsilon	`\upsilon`
ファイ	Φ	ϕ	phi	`\phi`
カイ	X	χ	chi	`\chi`
プサイ	Ψ	ψ	psi	`\psi`
オメガ	Ω	ω	omega	`\omega`

9409

付録 E

記号の入力とキーボードの場所

プログラミングのミスでよくあるのが打ち間違い，スペルミスです。X と x，S と s など大文字と小文字で形が同じものや，l(エルの小文字) と 1(数字のイチ) の違いなどは，プログラミング用フォントにして違いがわかるようにするとか，文字列の意味から類推する (Normal とあればノーマルであって，ノーマ・イチではないと察する) など工夫が必要かもしれません。

また，理由はよくわからないのですが頻発するスペルミスは，データ (data) をデート (date) と書いてしまうことです。データはラテン語の datum(与えられたもの) の複数形なのですが，最近のデータサイエンスの文脈では data も単数形と考えるようです。ともかく，日付を表す date とは由来も意味もスペルも全て違うので，気をつけましょう。

さて，これらはまだ序の口。プログラミングのコードを読んでも，日本語の五十音に入っていない記号の違いがわからない，どこでそれが入力できるかわからない，質問しようにも読み方がわからないといったものもあります。ここではこれらをまとめて解説します。記号の上では微妙な違いですが，当然形が違うのでプログラミング上は違う文字として扱われますので，形の細部までよくみてください。なお，プログラミングでつ変わる時は言語に依存することもありますので，ご注意ください[*1]。

特に目立つのはハイフンとチルダ，アンダースコアの入力ミスです。それぞれ文字を書く領域における位置が違ったり，形が違ったりするのでよくみてください。

ハイフンは真ん中	A-B
アンダースコアは下	A_B
オーバーバーは上	A‾B
チルダはニョロ	A~B

これを踏まえて，そのほかの記号の名称や意味を表 E.1 で確認しましょう。

一般的な日本語キー配列の場合，1 つのキーに 4 つの文字・記号が割り当てられていますが，英語入力モードの場合はキーの左側，日本語入力モードはキーの右側を見ることになります。キーを押すと下の段の文字が入力されますが，シフトキーを押しながらキーを押すことで上の段の文字が入力されることになります (図 E.1)。

これを踏まえて，日本語キーについては E.2，US キーについては図 E.3 に代表的な記号とキー配列の位置を示しました。入力に困った場合は一度図を見て確認してください[*2]。

[*1] たとえばコメントアウトは C 言語では\\，R では#，TeX では % など，それぞれ異なります。
[*2] US キー配列はキートップに一種類の文字しかなく，美しい配置なのでおすすめです。

表 E.1　記号と読み方

記号	読み方	解説
:	コロン	英文中では「すなわち」などの意味。セミコロンと間違えないように
;	セミコロン	英文中では文章の区切り，接続詞のようにつかう
.	ピリオド	英文の終わりを意味する。日本語で言う句点
,	カンマ	英文の区切りを意味する。日本語で言う読点
@	アットマーク	メールアドレスに用いられることで有名
$	ドルマーク	米国の通貨単位。R では変数名指定のときにもちいる
/	スラッシュ	割り算の記号
*	アスタリスク	掛け算の記号
+	プラス	足し算の記号
-	マイナス	引き算の記号
^	ハット	累乗の計算の記号
=	イコール	等号。プログラミングでは==で一致しているかどうかの判定にも
!	エクスクラメーション	強調。プログラミングでは否定 (NOT) の意味になることも
_	アンダースコア，アンダーバー	位置に注意。ハイフンではなく文字領域の下の線 変数名をつなげる時 (ex. `snake_case`) に使ったりする
~	チルダ	R では独立変数と従属変数をつなぐときに使う ハイフンやオーバーライン，アンダースコアと間違えられる率高め
‾	オーバーライン，オーバーバー	アンダースコアの逆。滅多に使わないが。文字化けを直したときにみられる。
%	パーセント	プログラミングではコメントアウトの時などに使われたりする
&	アンパサンド	プログラミングにおける AND(論理積) の記号など
|	縦棒	プログラミングにおける OR(論理和)，条件付き確率の記号にも
\	バックスラッシュ	プログラミングではコメントアウトの時などに使われたりする
"	ダブルクォーテーション	文字列の開始・終了を表す。同じ記号で閉じる
'	シングルクォーテーション	文字列の開始・終了を表す。同じ記号で閉じる
`	バッククォーテーション	文字列の開始・終了を表す。同じ記号で閉じる
[]	大括弧	プログラミングでは配列を意味することがある
{}	中括弧	プログラミングではブロックの開始と終了を意味することがある
()	小括弧	数式のまとまりを表す

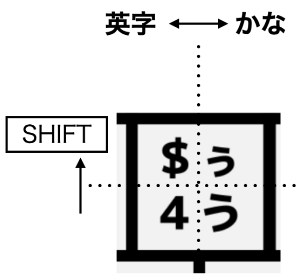


図 E.1　日本語キーで入力する場合

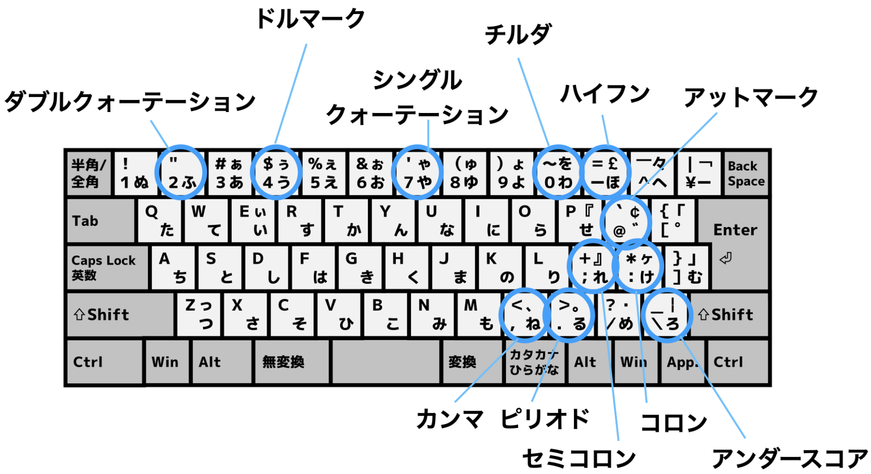


図 E.2 代表的な記号と日本語キー配列

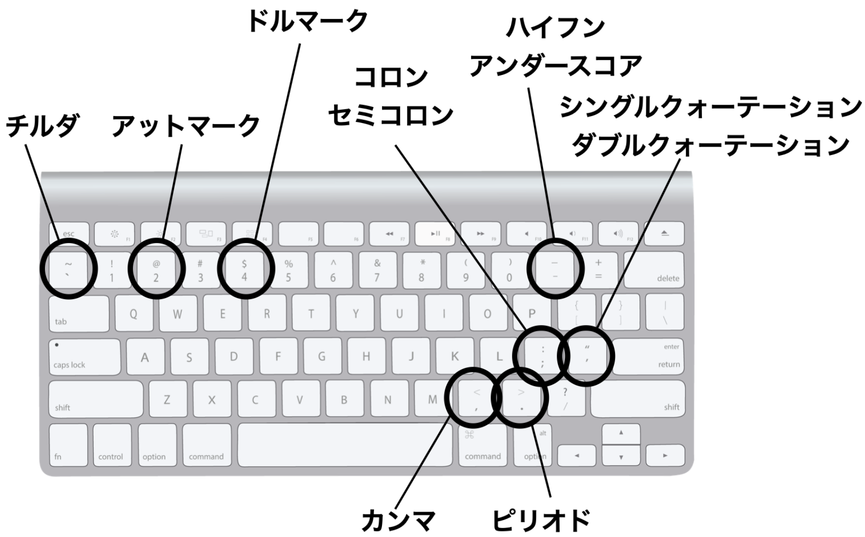


図 E.3 代表的な記号と US キー配列

付録 F

本講義に対応する詳細シラバス

F.1　イントロダクション

F.1.1　授業内容

科目の中でのこのコマの位置づけ

この講義の位置付けは，基礎的な心理統計の学習は終わった後の応用的内容となる。基礎的な内容として，確率の基本的な考え方，線形モデル (回帰分析，群間の平均値差の検討)，さまざまな推定法による母数の推定と検定の考え方を理解しているものとする。これに基づいての応用であるから，扱うデータも単変量ではなく多変量であるし，数学的には行列表現を用いることになる。これらを使って，回帰分析や因子分析の数理的理解を目指す。このコマではこの講義によって扱われる領域を外観するとともに，基礎的な内容で扱ったものがしっかりと定着しているかどうかを確認することを目的とする。

コマ主題細目

正規線形モデルの世界　単変量ではなく多変量を扱う統計の領域に入るので，多変量データとはどのようなものであるかに言及した上で，本講義の扱う領域を概観する。心理統計の応用的分野では，正規分布を仮定した線形モデルがその大半を占めている。正規線形モデルに含まれるさまざまな下位モデルの名称を紹介するとともに，構造方程式モデリングに統合されることや，非線形なモデルとの違いについて理解する。加えて正規分布ではない分布を扱うモデルも増えてきている昨今，これらについてのモデリングアプローチの存在についても講義する。

→ 正規線形モデルの枠組みについては，三中 (2018) 参照。

尺度の四水準　心理統計の基礎で触れたが，データ化として扱う数値はその尺度水準によってどのような計算が可能かということに違いが生じる。このことは，そのまま分析モデルや名称の違いに繋がるため，改めて名義，順序，間隔，比率の 4 水準を確認しておく。

→ Stevens (1946) の論文は短く，ネットで読むこともできる。入門書としては川端・荘島 (2014) の Pp.9–16，あるいは山田・村井 (2004) の Pp.22-25.

平均と分散　間隔尺度水準以上の数字であれば，平均値や分散，標準偏差によってその特徴を要約できる。ここではこれらの代表値の表記について，数学的記号とともに確認する。加えて，分散式を展開して表現したものや，分散がデータから得られる情報の上限であることを確認する。

共分散と相関係数　共分散やそれを標準化した相関係数は，複数の変数間関係を表現する最もシンプルな

もののｌつである。ここではこれらの複数の変数間に関わる代表値について，数学的記号とともに確認する。加えて，この他の関係の表現方法として，距離や共頻度などの共変量について解説し，それらの違いに応じて統計モデルが変わりうることを確認する。

→ 記述統計量については川端・荘島 (2014) の Pp.26-33 など基礎的な心理統計の教科書を参照すると良い。

キーワード

- 正規線形モデル
- 尺度水準
- 平均と分散
- 共分散と標準偏差

F.1.2 授業情報

■コマの展開方法 講義

予習・復習課題

■予習 心理統計の基礎について，今一度基礎的なテキストを参照しながら，自分の理解度を再確認しておくと良い。とくに尺度水準や記述統計量の計算方法などは今後この講義でも頻出するので，確認しておく必要がある。

■復習 数式の展開を踏まえて理解しておくとともに，実際のデータを使ってて計算しながら確認すると良い。とくに分散は二乗のオーダーになるので元の単位に比べて大きな数字になること，標準化のプロセスや相関係数の大きさなど，逐一確認しておくべきである。

F.2 心理尺度を作る

F.2.1 授業内容

科目の中でのこのコマの位置づけ

目に見えないものを測定するために心理学が洗練してきた手法が，心理尺度である。心理尺度の作成方法としてサーストン法，リッカート法，SD 法などがあり，心理学の初頭コースで習うものも少なくないが，その本質は反応カテゴリに数値を割り当てる，というところにある。「そう思わない」「ややそう思わない」などといったカテゴリーに対する反応が，なぜ 5 や 4 といった数字にすることが許されるのか。カテゴリカルな反応が連続的な量として扱うことができる理由などについて，よく知られていない現実がある。この点をしっかり理解しないまま進んだ分析を行うと，結果の解釈はもちろんそもそもの研究が足元から崩壊することにもなりかねない。本講義ではこの点について，作成方法から数値化まで一通り確認し，最後に心理尺度の評価基準である信頼性と妥当性について理解する。

コマ主題細目

サーストンの等現間隔法 サーストン法と呼ばれる尺度構成法は，態度とよばれる心理学的特性を仮定している。この態度は対象，符号，強度をもち，正規分布すると仮定されている。個々人の態度を測定す

るために，事前に評定者集団を用意して項目を採点しておく必要がある。そこで評定値に等間隔性を持たせる工夫をしているため，態度の数値化ができるという原理を理解する。

→ サーストン法による尺度作成については末永 (1987) の Pp.149–152 参照。

リッカートのシグマ法　リッカート法は最もよく使われるスタイルの心理尺度である。カテゴリーに無頓着に数字を割り振る慣例がみられるが，本来は潜在的態度が正規分布することを想定し，確率分布の確率点を得点とする方法であった。このような数値化がされているからこそ，順序尺度ではなく間隔尺度水準と「見なす」ことが許されてきているのである。この原理を理解しておくことは，後のより進んだ尺度作成法を理解する助けになる。

→ リッカート法による尺度作成については，宮谷・坂田・林・坂田・入戸野・森田 (2009) の Pp.150–153 を参照。ただしシグマ法についての言及はなく，田中 (1977) などの古典を当たらねばならない。

尺度を評価する　作られた尺度を評価する方法として，IT 相関を求めるものや内的整合性信頼性を求めるものがある。その背後には信頼性と妥当性の考え方があることを確認する。

→ 心理尺度の信頼性については，末永 (1987) の Pp.156–158，妥当性については Grimm and Yarnold (2001) の第 4 章も参照。

キーワード

- サーストンの等現間隔法
- リッカートのシグマ法
- 信頼性
- 妥当性

F.2.2　授業情報

■コマの展開方法　講義

予習・復習課題

■予習　基礎実習で尺度作成法や尺度の分析をしたことがあれば，その時の資料を再確認しておく。とくに反応カテゴリをどのように採点したか，また尺度の評価どのように行ったかを確認する。とくに尺度作成の経験がない場合は，関連書籍を参考に方法論を予習しておくことが望ましい。

■復習　IT 相関やアルファ係数は統計環境 R で簡単に計算できる。とくに psych パッケージにはこれらの関数がすでに準備されている。サンプルデータを使ってこれらを計算してみよう。

F.3　テスト理論と因子分析

F.3.1　授業内容

科目の中でのこのコマの位置づけ

目に見えないものを測定するという意味で，テスト理論は心理学の測定と関係が深い。前回は社会心理学における態度の測定を前提に議論したが，測定に関してはテスト理論で一般的に議論できる。

真のスコアと誤差とに分解すること，誤差の基本的な仮定を確認した上で，古典的テスト理論を項目と被験者の特性に分割することで因子分析モデルに展開されるところを見る。また，多因子モデルに拡張した上で，その数理的展開から，信頼性と妥当性に言及できることを確認する。数式の展開は代数の基本的な特徴を確認すれば問題なくフォローできるはずである。

コマ主題細目

古典的テスト理論　古典的テスト理論についての復習である。その基本モデルについて触れ，その平均値と分散が意味するところから測定モデルの意味するところ（誤差が相殺しあうこと）と信頼性の定義が導出できることを改めて確認しておく。

因子分析モデル　因子分析モデルは，古典的テスト理論のモデルを拡張したものである。まずは単因子モデルを例に，項目特性と被験者特性が分離されたことを確認する。その上で，性格検査や知能検査などの歴史に触れながら，多因子モデルについて解説する。多因子モデルを例に記号や添字を確認しておく。

→ 小杉 (2018) の Pp.173–177

因子分析の第 2 定理　因子分析モデルを展開することで，相関係数が因子負荷量の積和で表現できること，因子得点がその仮定から計算上消えることを確認する。得られた指揮は因子分析の第二定理と呼ばれ，妥当性に関する議論がここから導かれることをみる。

因子分析の第 1 定理　ある項目自身の相関係数を考えることで，因子分析の第一定理にたどり着く。ここで共通因子の二乗和を共通性と呼ぶことにすると，信頼性の考え方が項目レベルで行われるように発展したことが確認できる。

→ 小杉 (2018) の Pp.173–177

キーワード

- 古典的テスト理論
- 因子分析法
- 因子分析の定理

F.3.2　授業情報

■コマの展開方法　講義

予習・復習課題

■予習　一年時に信頼性・妥当性について，あるいは古典的テスト理論について学んだことを復習し，どのような概念であったかを再確認しておくことが望ましい。

■復習　テスト理論と因子分析モデルの関係について，因子分析モデルはどこが新しく何を改定しようとしたのかについて，自分なりの言葉で説明できるようになろう。

F.4　現代テスト理論

F.4.1　授業内容

科目の中でのこのコマの位置づけ

テストの理論も目に見えないものを測定するという意味では，心理学と同じモデルを実践する領域である。心理学的尺度作成法の発展には，テスト業界における理論的展開の位置付けを知ることが役に立つ。

因子分析によって項目と被験者の特徴を分離して考えることができるようになった。ここで学力テストに目を向けると，単因子でよいことと従属変数がバイナリになっていることがわかる。

この特殊な測定法についてのモデルを考えるために，まずは通過率の概念を導入したうえで，累積正規分布とその近似としてのロジスティック曲線，および 1,2,3PL モデルを紹介する。これらのテストは新しいテスト理論とよばれるが，それはこれまでのテスト理論に含まれていた集団に依存した測定であったこと，完全データに限定されていたことなどを乗り越えられるからである。もちろんテストの等価がしやすいという側面もある。

テスト理論の展開としての項目反応理論と，因子分析モデルとの相同性を強調することで，見えないものを測定しようとするアプローチという意味では同じであったことを確認する。

コマ主題細目

因子分析とテスト理論　単因子モデルの特殊事例として，学力テストの例を考える。学力テストの性質から，因子構造よりも因子得点に注目するという強調点の違いはあるが，因子分析モデルの一環として捉えることを強調する。

→ 高橋 (2002) は最も平易なテスト及び現代テスト理論への入門書である。最初の数ページだけでも参考になる。

通過率と累積正規分布　学力テストの分析例として，通過率の計算から累積正規分布へとつなげる。累積正規分布をそのまま確率モデルに繋げてもよいが，ロジスティック曲線を使う方が関数の形が簡単であり，こちらの方が実際には使い勝手が良い。ベルヌーイ分布を用いた線形回帰モデルの文脈で考えれば，ロジスティック回帰分析をしていることでもあることに言及する。

→ 通過率については豊田 (2012) の Pp.1–8 を，ロジスティック関数と累積正規分布の関係については加藤他 (2014) の Pp.81–83 を参照

項目母数の特徴　ロジスティック曲線を導入することで，関数の変形がたやすくなった。ここでは 1PL,2PL ロジスティックモデルを導入し，どの項目母数が関数の位置や形をどのように変えるか，そしてそれが意味するところを理解する。モデル的には 5 母数モデルまで考えられるが，実際にはせいぜい 3PL モデルである。この講義では後の因子分析との対応関係も考えるため，2PL モデルまでの紹介に留める。

→ 項目母数については豊田 (2012) の Pp.31–34, 加藤他 (2014) の Pp.71–80 が参考になる。

被験者母数の特徴 項目母数が明らかになった状況に置いて，どのように被験者母数を推定するかを考える。ここで ICC から逆算的に被験者母数の位置がどこにあるか，該当領域を絞り込んでいく尤度関数を視覚的に確認する。この方法を使うと，すべての項目についての回答が得られていないと推定できないといった不便がなく，また被験者母数の位置によっては ICC がそれほど有用な情報を与えてくれないこともある。これらの点は，完全情報最尤推定や情報関数にもつながるため，しっかりと理解しておくことが必要である。

→ 被験者母数の絞り込みについては，小杉・清水 (2014) の Pp.171–172. が参考になる。

キーワード

- 通過率
- ロジスティックモデル
- 被験者母数の推定について

F.4.2 授業情報

■コマの展開方法 講義

予習・復習課題

■予習 テストの前提となる標準正規分布について復習しておく。とくに R を使って出力できる確率密度，確率点，累積確率など手を動かして予習しておくと良い。

■復習 適当なグラフ描画ツール (R でよい) をつかって，ロジスティックモデルを描写し，項目母数をどのように変えるとどのように曲線の形が変わるかを確認してみよう。

F.5 現代テスト理論その 2

F.5.1 授業内容

科目の中でのこのコマの位置づけ

項目反応理論の数学的特徴を踏まえることと，現代的尺度構成法の理論的基礎を学ぶ。

項目反応理論の導入によって，被験者母数と項目母数が完全に分離され，項目の特徴を細かく記述できるようになった。また，項目の特徴がわかっていればテストの実践方法も変わってくる。ひとつは CAT に代表されるように，ダイナミックに出題を変化させることができるようになること，そうしたうえでもテストの平均点が事前にコントロールしうることなどが示される。項目から得られる情報という観点から項目情報曲線が，項目情報曲線の累積からテスト情報曲線が導出される。

つづいてこのテスト理論の発展形として，多段階モデルに拡張可能なことをみる。とくに段階反応モデルは，リッカートのシグマ法のように段階反応をモデルかできるという意味で，現代的リッカート法であるともいえる。段階反応モデルを用いることで，適切な反応段階のチェックをできるなど，応用的側面が高いことを確認する。

またテスト理論は因子分析の特殊系であるという扱いだったが，多段階，多因子へと展開することで再び

因子分析モデルに統合されていくことを確認する。

コマ主題細目

現代テスト理論の特徴　現代テスト理論の特徴は，項目母数と被験者母数の分離，完全情報最尤推定，項目情報曲線による信頼性の表現，項目プールがあれば事前にテストの平均点を設計できることがあげあられる。また Computer Adopted Test の形式を用いることでテストのあり方そのものも変わってしまう。ただし実際には，膨大な項目プールが必要であること，事前に項目母数を準備しておく必要があること，その他「公平性のために新しいテストでなければならない」という信念などが弊害となって実践的には敷居が高いことなどを解説する。

→ 古典的テスト理論との比較については，加藤他 (2014) の Pp.67–69，あるいは豊田 (2012) の前書きが十分に詳しい。

段階反応モデル　テスト理論はバイナリデータに対する分析だが，多段階の反応に拡張する方法がいくつか考えられている。1 つは段階反応モデルとよばれるもので，これを使うと適当な反応段階数がデザインできるなど利点は大きい。またその考え方はリッカートのシグマ法を洗練したものであるとも言え，せめてこうした方法を使わないと多段階反応を適当に分析できていない。統計パッケージなどの実装も進んでいるので，計算コストはほとんど障壁にならない。また，ポリコリック相関係数を用いた因子分析を実行すると，段階反応モデルのパラメータに変換できることから，因子分析とテスト理論が同じものであったことを再確認できる。

→ 豊田 (2012) の Pp.155–172 が詳しい。

因子分析の歴史と展開　因子分析モデルもテスト理論も潜在変数モデルとしては同じであり，一方が単因子・二段階，他方が多因子・多段階であることが道を分つ。またその性質から，一方が因子得点に，他方が因子構造に着目するため，テストの構成についての考え方が異なることにも注意する。繰り返しになるが，統計パッケージ上の実装は進んでいるので，どちらを使うにしてもとくに苦労することなく，積極的にカテゴリ軽モデルを推進していくべきである。

キーワード

- 項目情報曲線，テスト情報曲線
- 段階反応モデル
- 因子分析モデルとテスト理論

F.5.2　授業情報

■コマの展開方法　講義

予習・復習課題

■予習　項目反応理論，とくに 2PL モデルによる因子得点の算出方法を確認しておくと同時に，心理尺度ではどのように尺度値を定めていたかについて復習しておく。

■復習　信頼性についての考え方が，古典的テスト理論，因子分析論，現代テスト理論を通じてどのように変わってきたかを確認しておこう。

F.6　行列計算の基礎

F.6.1　授業内容

科目の中でのこのコマの位置づけ

テスト理論や因子分析モデルの展開を理解した上で，さらに次のステップに進むためには，より数学的な構造の理解が必要である。ここまで因子分析モデルでは，因子得点をどのように算出するかが論じられていない。また相関行列を分解して因子負荷量を算出するにあたっても，どのように計算するかについては言及されてこなかった。これらの点を理解するための道具となるのが線形代数である。具体的には，行列の固有値分解を通じた解釈をすることで，因子分析，回帰分析など多変量データの方程式モデルを統一的に表現・理解できるようになる。そのための道具立てとして，線形代数の基礎知識を習得する必要がある。本講はこのより進んだ理解に向かうための，新しい数学ツールの導入を行う。

線形代数は方程式を簡便的に表現するための表現法であり，行列の観点から新たに四則演算を定義し直すことで一般的な表現が可能になることを示す。

コマ主題細目

行列とベクトル　多変量データを行列とベクトルで表現することをみる。学ぶべき用語として，スカラー，縦ベクトル，横ベクトル，行列，正方行列，対称行列，対角行列，単位行列をあげる。

行列の四則演算　ベクトルとベクトルの和，行列と行列の和，スカラーとベクトルの積，スカラーと行列の積，縦ベクトルと横ベクトルの積，横ベクトルと縦ベクトルの積，行列と行列の積をみる。とくにサイズが変わることに注意が必要である。

行列による便利な表現　連立方程式が行列で表現できることを見る。

逆行列と連立方程式　行列の割り算に当たるのが逆行列である。逆行列は存在しないこともあるが，もし適当なものが見つかればそれは連立方程式の解を一気に計算ができることになる。

キーワード

- ベクトル，スカラー，行列
- 行列の四則演算
- 連立方程式

F.6.2　授業情報

■コマの展開方法　講義

予習・復習課題

■予習　とくに予習の必要は感じないが，授業に参加するにあたってはノートの準備が必要である。

■復習　計算方法に慣れておく必要があるので，練習問題を繰り返して行うことで，とくに行列の積の計算ができるようになっておく。線形代数の入門書としては，数学のテキストとして読みづらさを感じるかもしれないが，村上他 (2016) がよく，一冊手元に置いて演習をしながら進めると良い。

F.7 行列による関係の表現

F.7.1 授業内容

科目の中でのこのコマの位置づけ

線形代数についての基礎的なルールを習得する段階である。今回はより実践的·具体的に，データ行列をどのように線形代数で表現できるかを考える。データ行列から分散共分散行列，相関行列へと形を変えることを学ぶ。つづいて線形モデル，とくに従属変数が明確な回帰分析モデルを行列で表現することを見，線形モデルとデザイン行列について考える。さらに因子分析モデルを行列で表現することを考える。行列で表現することで，1 つの式の中に第一，第二定理の両方を含んだ形で表現できることを理解する。

コマ主題細目

データの行列表現　実際に手にするデータセットは，表計算ソフトウェアの画面で見る行列形式の数列であるが，これを記号で表現することで一般的に扱うことができるようになる。添字に気をつけながら要素ごとの表示をすることに加え，行列の計算をこのデータ行列に与えることによって，変数の平均や変数ごとの平均偏差を持った行列が表現できる。平均偏差行列を用いると，行列の積の特徴から分散と共分散を含んだ正方行列が作られることがわかる。また，データを標準化することで，標準化された行列の積が相関行列を表すことになる。このように一般的に表現するために，これまでの行列計算の方法が作られたのだと逆算的に理解すること，加えて行列のサイズに注目しながら，扱うデータの大きさがイメージできるようになることが肝要である。

→ 岡太 (2008) の Pp.77-110

線形モデル　行列表現の利便性は，データの変換だけにあるのではなく，統計モデルを表現する際にも生きてくる。基礎で学ぶ線形モデルは，基本的にエレメントワイズな表記法であったが，行列を使うことで単回帰も重回帰も同じ式で表現できることがわかる。このように表記の統一性があることが，線形代数の利点である。また統一的な表記にするために，切片項にかかる列を追加するなどの工夫をすることにも注意する。これらの点は，R など統計ソフトウェアを扱う上でもヒントになることが多い。

デザイン行列　基礎の段階で行った帰無仮説検定は，説明変数が離散変数であったことから，線形モデルの特殊形に過ぎなかったことを再確認する。その上で，先の回帰分析を行列表記にしたように，離散変数で説明する時の係数にかかる行列の形を確認する。この行列はとくにデザイン行列と呼ばれること，また自由度の関係から制約を加えた表現になるが，それがデザイン行列の中でどのように書き表されるかを確認する。

→ J.Dobson (2008) の Pp.41–45 にごく簡単な紹介が，豊田 (2000) の Pp.47–62 には計画行列として構造方程式の枠組みで説明されている。

因子分析モデルの行列表現　因子分析モデルはここまでエレメントワイズで表現されていたが，同様に行列表現にするとどのようになるかを確認する。とくに行列のサイズに注目することが重要である。というのも，統計ソフトウェアを使っていると因子得点が表示されないことが少なくないが，行列の形で見ると因子負荷量は項目数 × 因子数，因子得点は回答者数 × 因子数になることがより意識されやすいからである。他にも因子分析に関する特徴量が行列のどの要素にはいっているか，また因子分析の定理が行列のどこで表現されているかを確認することが重要である。

因子分析の行列的表現については → 芝 (1979) が良書だが，現在は絶版。同様の内容は小杉 (2018) にもある。

キーワード

- データの行列表現
- 分散共分散行列，相関行列
- デザイン行列

F.7.2　授業情報

■コマの展開方法　講義

予習・復習課題

■予習　行列の掛け算がメインになってくるので，計算方法並びに計算結果のサイズを確認する方法を見ておこう。

■復習　行列表現によって重回帰方程式が 1 つの形になることを確認する。平均値の差を見るために線形モデルが用いられることを確認する。また因子分析モデルを行列表現すると，一気に 2 つの定理が 1 つの式で表現できることを確認する。

F.8　固有値と固有ベクトルと因子分析モデルの関係

F.8.1　授業内容

科目の中でのこのコマの位置づけ

因子分析モデルを行列表現することで，いよいよ因子をどのように算出しているのかについての答えが明らかになる。

因子負荷量を算出するためには，線形代数でいうところの固有値についての理解が必要である。まずは固有値と固有ベクトルを導入し，どのように計算するかを見る。とくに固有ベクトルはノルムが定まらないことを確認する。そこから，固有値と固有ベクトルがどのような性質を持っているかを幾何学的観点から確認する。正方行列が座標変換を行うためのものであると考えれば，固有ベクトルは変換行列の基底となることがわかるだろう。データ解析にあたって，相関行列の基底を求めるとはどういうことかをイメージするだけでも，因子分析の理解がまた一歩深まるだろう。

コマ主題細目

固有値と固有ベクトル　行列の固有値と固有ベクトルの性質を理解する。直感的には，正方行列がスカラーに変わることが，情報圧縮になっていると言えるだろう。また，行列のサイズと同じ数だけ固有値が見つかること，固有値の総和が元の行列のトレース trace になることを確認する。とくにデータ解析の領域では，分散共分散行列か相関行列が分析対象になることが基本であり，こうした対称行列の固有値は実数になること，相関行列のトレースは項目数と合致することを改めて確認することで，データの情報圧縮になることについての直感的理解をめざす。

固有ベクトルを求める　2×2 行列を例に，固有値と固有ベクトルを求める計算を行う。固有方程式を導入

し固有値の計算を行うことは比較的簡単であるが，固有ベクトルの求め方が直感的にはわかりにくい。というのも，固有ベクトルはその大きさが定まっておらず，要素同士の相対的な大きさを示すだけだからである。ここでベクトルのノルムを導入して標準化解を算出することを確認する。また行列のサイズが大きくなると方程式が高次になるため，一般解が得られないこと，結果的に近似解を求める計算方法が開発されていることをみる。

固有値と固有ベクトルの幾何学的意味　正方行列は一次変換行列であり，固有ベクトルはその基底であることを単純な行列から理解する。固有ベクトルはノルムが定まっていないこと，すなわち方向性だけを持ったものであることを理解する。また固有値はその総和が元の行列のトレースと一致することから，分散あるいは項目数 (相関行列の対角) を組み替えたものであり，固有値の大きさの順に考えることはすなわち，より明確な次元を抽出したことになることを確認する。

→ これについては平岡・堀 (2004) にアニメーション付きで説明されているのがわかりやすい。また長沼 (2011) は固有値の章だけでなく，付録を読むとまた固有値と固有ベクトルの多角的な理解が進む。

因子分析モデルの意味　因子分析モデルは相関行列を固有値分解することであり，それはすなわち相関行列の中にある基本的な次元・座標を求めることにある。すなわち複数人の反応パターンの共通要素を取り出すということであり，これは心理学的アプローチをほぼ直接的に数学表現したものであることを理解する。座標の回転についても触れ，仮定を緩めた場合の表現も理解する。

キーワード

- 因子分析モデルの行列表現
- 固有値
- 固有ベクトル
- 固有ベクトルの幾何学的理解

F.8.2　授業情報

■コマの展開方法　講義

予習・復習課題

■予習　因子分析の基本モデル，第一・第二定理の導出を復習しておこう。

■復習　因子分析モデルが何をやっているかを考えた上で，心理学における尺度の利用やその解釈においてどのような注意をしなければならないかを言語化してみよう。

F.9　R をつかっての行列計算

F.9.1　授業内容

科目の中でのこのコマの位置づけ

行列の計算は単純な計算ではあるが，要素の数が多くなるので反復回数が増え，また計算の法則も慣れるまでは難しい。人間にとってはミスが多くなりがちなこの計算が，計算機 (コンピュータ) は最も得意とする

ところである。計算機は疲れることなく，単純な反復計算を瞬時にこなす。多変量データ解析は計算機の発展の歴史ともにあり，昨今の計算機パワーは非常に複雑な統計解析も瞬時に答えを出すようになった。

この行列計算は表計算ソフトにはできないことであり，統計環境 R のような，統計パッケージを利用することになる。本項では，統計環境 R を用いて行列の基本的な計算を演習によって習得することを目的としている。また R で行列の計算ができることは重要ではあるが，実際に統計分析をする時にはより便利なパッケージを利用することになる。心理学関係の数値計算については，`psych` パッケージが便利である。これを導入し，記述統計量や信頼性係数など基本的な分析が便利になることを確認する。

コマ主題細目

R による行列計算　R についての基本的な使い方 (環境の準備，RStudio によるプロジェクト管理，パッケージの導入，基本的な四則演算等) については習得済みであることを前提とする。行列計算にあたっては，データをマトリックス型で保持している必要があり，また行列の計算は四則演算と異なること，ベクトルの長さが時には再利用されることなど注意が必要な点がある。それらを踏まえて，データの方を考えながら行列の四則演算を確認する。

R によるデータの変換　R の行列計算を使って，前時までに行った raw data の変換計算，すなわち平均，平均偏差行列，分散共分散行列，相関行列などの計算プロセスを確認する。また，`cov` や `cor` 関数を使うとこれらが一気に計算されるが，分散の関数には不偏分散が用いられていることに注意する必要がある。

R による固有値計算　R の `eigen` 関数を使って，固有値と固有ベクトルが計算されるところを確認する。固有ベクトルは標準化されていることに注意する。

キーワード

- 行列型
- 行列関数

F.9.2　授業情報

■コマの展開方法　R を使った演習

予習・復習課題

■予習　R/RStudio を使った分析環境を再確認しておこう。またデータの読み込みや記述統計量などの算出関数を確認しておこう。

■復習　授業時間内に収まらなかったところがあれば，必ずキャッチアップしておくこと。いくつかの練習問題を実践し，エラーや警告がでても対応できるようになろう。

F.10 R をつかった因子分析と尺度作成法

F.10.1 授業内容

科目の中でのこのコマの位置づけ

ここでは心理尺度を開発するような心理学研究を想定し，より実践的な順序に則って演習を進めていく。この講義の目標は，自らが質問紙調査を使った研究をした場合にどのような手順で行うかを理解し，実践できるようになることである。具体的には前回導入した `psych` パッケージを用いて，さまざまな推定オプションを追加していくことで出力が変わっていくことを確認しながら進める。

コマ主題細目

psych パッケージ概説 心理学研究に用いられる便利な関数群である `psych` パッケージのマニュアルを見ながら，`describe.by` などの記述統計量関数，`alpha` や `omega` といった信頼性係数の関数を使ってロウデータの分析を行う。

調査研究の手順 心理尺度の作成研究の手続きを外観する。まず構成概念の設定，定義，妥当性を考えた上で，具体的な項目を選出し，テストデータを取る。探索的な因子分析によってその因子的妥当性を確認し，標準化のための本調査を行う。あるいは 1 次元性を確認した上で，IRT によって反応段階の確認，項目母数の確認，テスト情報関数の確認などが必要である。尺度の翻訳や検証的妥当性のチェックなどについては，構造方程式モデリングによる分析を行うのでここでは扱わず，参照するにとどめる。

共通性推定の問題 分析にあたって，改めて因子分析モデルの行列表現を提示し，行列の固有値分解によって因子負荷量が求められることを確認する。しかしその際，共通性をどのように推定するかの問題が残されていたことを確認し，そのためにいくつかの方法が提案されていることを理解する。これらは因子分析を行う上で，推定方法のオプション指定に関わってくる点であり，ソフトウェアが変わっても同様の指標が必要であることをみる。

→ 小杉 (2018) の Pp.91–94.

fa 関数と探索的因子分析 探索的因子分析の手続きを `fa` 関数を使いながら考える。探索的因子分析の場合は因子構造，因子負荷量について何ら前提を置かないため，因子数の推定から始めなければならない。まずは `fa.parallel` 関数でスクリープロットを描画する。スクリープロットを読むときの形状について確認する。続いて因子数と共通性推定方法を定めた上で `fa` 関数を実行し，因子負荷量や共通性などアウトプットを確認する。続いて解釈を簡単にするために因子軸の回転を行うことを解説し，実行のために `rotate` オプションを追加することをみる。回転前の結果との比較，また直交回転と斜交回転の違いを確認する。

→ 小杉 (2018) の Pp.81–91.

因子得点の算出 因子数と因子負荷量が明らかになると，そこから逆算的に因子得点を計算できる。`fa` 関数には `scores` オプションをつけることで，出力されたオブジェクトから因子得点を取り出すことができるのをみる。こうした方法とは別に，項目同士の素点の平均から因子得点を計算することもある。これは推定値を実体とすることの懸念が出発点であり，その長所と短所を把握しておくことが必要であ

る。この簡便法は平均値情報を含んでいるため，尺度カテゴリに依拠した解釈が可能である。また取り出された因子得点と簡便的因子得点の相関を見ることを確認する。

因子分析の注意点 因子分析を行う上で注意しなければならないのは，因子が実体としてあるのではなく，あくまでも準備された項目群の相関関係から得られる基底に過ぎないことを理解する点である。因子分析の流れの中では因子に命名することが 1 つの手順としてあるが，言葉として確定するとあたかもそれがあるかのように考えられてしまうこと，それしかないように考えられてしまうことの危険性を理解する。心理尺度の呪いやてっちゃんの手品になってしまわないように注意し，常に元の項目群に戻って考える必要があることをしっかりと理解する。

キーワード

- 信頼性係数
- 共通性
- 因子負荷量
- 因子得点
- `psych` パッケージ
- `fa,fa.poly,fa.parallel` 関数

F.10.2 授業情報

■コマの展開方法 R を使った演習

予習・復習課題

■予習 パッケージの読み込みや関数の結果を見る方法を確認しておこう。一年時のことを思い出して，`lm` 関数を例に R の操作方を思い出しておく。

■復習 授業時間内に収まらなかったところがあれば，必ずキャッチアップしておくこと。いくつかの練習問題を実践し，エラーや警告に対応できるようになろう。心理学研究など心理学の専門雑誌を参考に，どのような分析結果がどのように報告されているかを確認しておくことも，理解を進める。

F.11 R をつかった項目反応理論

F.11.1 授業内容

科目の中でのこのコマの位置づけ

項目反応理論を実践的に理解する演習パートである。

カテゴリカルな因子分析と数学的に同等ではあるが，より項目の特徴を広く表現できる項目反応理論の利用が，今後より重要なものになってくるだろう。

ここではまずテスト理論の根本に立ち返り，二値単因子のデータを使って 1PL,2PL モデルの分析を行う。分析結果は数値で見ることも重要であるが，ICC や IIC，TIC などを使って可視化するとより理解が深まるだろう。多段階の反応についても，同様に GRM を実行し，閾値や識別力，IRCCC や IIC，TIC が描画できることを確認する。とくに IRCCC による反応段階の読み取り方には注意する。最後に多段階で多因子の場合，項目反応理論の文脈から言えば多次元 IRT になり専用のパッケージが必要になることを紹介しつ

つ，カテゴリカル因子分析でも同様のことができることを確認する。

コマ主題細目

項目反応理論の実際　項目反応理論はテスト理論がその出自に当たるので，まずは二値データで単因子が想定できるような例を元に分析を行う。分析には `irtoys` パッケージや `ltm` パッケージを用いて，1PL モデル，2PL モデルの演習を行う。項目母数の値と意味が，具体的な設問に照らし合わせて考えることで，より実感をもって理解できるようになると思われる。とくに，ICC や IIC,TIC など可視化することでその意味が理解しやすくなるだろう。3PL モデルなどさらに拡張したモデルも利用可能である。

段階反応モデルの実際　続いて単因子，段階反応モデルの実践を行う。因子構造として，前回の授業で扱った多因子の内，ある因子に限定して分析を行うこととする。段階反応モデル (GRM) の出力結果を数値だけでなく可視化することで，項目の特徴がどのように表現されているかを考える。とくに反応段階の山が潰れているようなケースは，適切な反応段階でなかったことを意味するので，数値の置き換えなど元データを修正しつつ分析し直すことを考える。これらを通じて，適切な反応段階による調査法が必要であることを理解する。

カテゴリカル因子分析との対応　多段階，多因子の場合は `psych` パッケージの `fa` 関数にオプションを追加することでできる，カテゴリカル因子分析と同じである。出力結果について，これまでの相関係数を用いているものとの違いを確認する。また数値をどのように変換すれば対応するのかを見ることで，数学的に等価であることを確認しておく。IRT の側面から多因子に拡張した，多次元 IRT についても，`mirt` パッケージを利用すれば実行できる。この解析には計算時間がかかるが，完全情報最尤推定の結果が得られることは利点である。

キーワード

- 1PL モデル，2PL モデル，3PL モデル
- 段階反応モデル
- カテゴリカル因子分析
- `irtoys,ltm,psych,mirt` パッケージ

F.11.2　授業情報

■コマの展開方法　R を使った演習

予習・復習課題

■予習　`irtoys,ltm` パッケージを事前にインストールして環境を整え，データファイルの読み込みなど R/RStudio の基本的な使い方を確認しておこう。とくに R の `data.frame` 型に含まれる変数が，`numeric` なのか `factor` なのかによって挙動がかわることがある。変数の型についても再確認しておこう。

■復習　本講で習ったパッケージを使って，具体的なデータを因子分析，IRT，カテゴリカル IRT などいくつかのモデルで分析し，それぞれの違いを確認しておこう。

F.12　構造方程式モデリング

F.12.1　授業内容

科目の中でのこのコマの位置づけ

構造方程式モデリングは，因子分析と回帰分析を統合して扱う，総合的分析モデルである。言い換えれば，これまでの多くの多変量解析モデルのほとんどは，構造方程式モデルの下位モデルとして表現できる。ここではこれまでのモデルを統合した，より現代的でより上位のモデルである構造方程式モデリングを理解することで，すべての多変量解析を網羅的かつ俯瞰的に捉えることが狙いである。

構造方程式モデリングを理解するには，変数の種類と関係性の区分に注意したパスダイアグラムの描き方を知ることが早い。パスダイアグラムを用いると，回帰分析と因子分析は説明変数が観測変数なのか潜在変数なのかといった違いであることが明らかである。また因子分析と似た主成分分析がどのように表されるかも，パスダイアグラムを見れば一目瞭然である。

パスダイアグラムには変数の尺度水準までかきこまれることはないが，ここに注意していろいろなモデルを描画すると，それがかつて多変量解析においてさまざまな名称で呼ばれた分析方法であったことがわかる。あるいは，今後どのようなモデルが開発される可能性があるか，どのようなモデルをどのように希釈すれば良いかもイメージできる。

加えてこの統合的なモデルがなぜそうした複雑なモデルを表現できるのかについても，モデルを方程式で描画し，行列で考えることで，モデル行列とデータ行列を近づけることと理解できる。この観点から，データにモデルを当てはめる適合度の考え方が改めて理解されるだろう。

コマ主題細目

パスダイアグラムの書き方　これまで学んできたモデルを図で表現することを学ぶ。そのためには，変数を観測変数と潜在変数に区別することと，変数間関係を因果関係と相関関係に区別する必要がある。観測変数を矩形，潜在変数を楕円形，因果関係を一方向矢印，相関関係を双方向矢印で表現することで，因子分析や回帰分析が図で表現できることを学ぶ。

パスダイアグラムによるさまざまなモデル　因子分析と回帰分析をパスダイアグラムで表現したことで，この両者を統合するような表現ができること，また潜在変数同士の関係を記述する，構造方程式を描画できるようになったと言える。因子分析と似た手法とされる主成分分析や，尺度水準の違いによるさまざまな統計モデルを表現する方法を手に入れたことになる。この手法を総称して，構造方程式モデリングと呼ぶ。

→ 小杉・清水 (2014) の Pp.7–10

構造方程式モデルによる未知数の推定　構造方程式モデリングでは，パスダイアグラムでも表現されるが，変数間関係を方程式で書くこともできる。方程式で描画することで，構造方程式も潜在変数の方程式と観測変数の方程式，それらが入れ子になった方程式で描画できることがわかる。またこれらのモデルを行列のイメージで捉えると，最終的には分散共分散行列という実態を持った数字に対して，未知数で描画された方程式を接続したことが直感的にわかるだろう。未知数の増え方と分散共分散行列の要素の増え方を比較すると後者が圧倒的に早く，未知数よりも既知数が多い方程式は解くことができるという原理から，未知数が推定しうることを理解する。

→ 小杉 (2018) の Pp.191–193.

適合度によるモデルの評価　データ行列とモデル行列をイコールで結んだ方程式を解くことが，未知数を求める根本的な原理であるが，このことからモデルがデータとどの程度合致しているかという適合度が，モデル評価の統合的観点として浮かび上がってくる。回帰分析では R^2 であったが，因子分析をはじめとしたさまざまな多変量解析モデルも，この評価次元で考えることができる。ただしその指標にはいくつかの特徴があり，これらを総合的にみて評価するという実践的ノウハウも確認する。

→ 小杉・清水 (2014) の Pp.10–12

キーワード

- パスダイアグラム
- 観測変数と潜在変数
- 因果関係と相関関係
- 構造方程式モデリング
- 適合度

F.12.2　授業情報

■コマの展開方法　講義

予習・復習課題

■予習　回帰分析と因子分析という 2 つの分析方法についてはすでに学んでいるが，この両者の共通点と相違点がどこにあるかを事前に考えてみよう。外的な基準の有無，説明変数の種類の観点から，自分の言葉で表現できるようになっていると良い。

■復習　これまで学んださまざまな統計モデルを，構造方程式モデリングの表記法に則ってパス図を書いてみよう。またさまざまな尺度水準の組み合わせからなるモデルを考え，それらがどのような意味を持つのかと推論するのも理解の助けになる。

F.13　R による構造方程式モデリング

F.13.1　授業内容

科目の中でのこのコマの位置づけ

これまでの流れと同じで，統計技術の理論を知っただけではなく，自分で実際に計算できる演習を経てこそ理解が深まるということから，本講では R をつかって実際に構造方程式モデリングを解くことを演習的に学ぶ。構造方程式モデリングを実装するパッケージは複数あるが，最も応用範囲がひろい `lavaan` パッケージを用いることにする。

まずは観測変数だけからなる簡単なパス解析を行う。データの入力の仕方，方程式の設定，関数の使い方などを一通り習得する。続いて潜在変数を含んだモデルによる解析を行う。モデルの適合度や修正指数を参考に，徐々にモデルを書き換えていく手順を学ぶ。注意すべきは，適合度を上げることが目的になって，不自然な仮定やパスをおいてしまうことである。あくまでも具体的かつ妥当なモデリングを心がけるべきである。

オプショナルな設定になるが，観測変数がカテゴリカルである場合や，推定方法の選択なども確認する。最後に，R 以外の統計パッケージによる構造方程式モデリングの実践例がいくつか紹介される。

コマ主題細目

方程式の入力　まずは観測変数同士の関係をパスでつなぐモデルで練習する。パス解析は回帰分析の繰り返しで実行することもできるが，構造方程式モデリングによってパスの繋がりを 1 つのモデルで表現し，適合度も統一できるなどの利点がある。観測変数だけからなるモデルの結果と，実際に `lm` 関数で実行した結果と比較すると良い。またパッケージにもよるが，自動的にパスダイアグラムを描画してくれるものもある。方程式とそのパスダイアグラムによる表現の対応を確認する。

→ 小杉･清水 (2014) の Pp.55–60

測定モデルの実践　因子分析モデルを SEM 上で実行してみる。探索的因子分析と違い，どの項目にどの因子が影響しているかを固定したモデリングが可能であり，この検証的因子分析による結果と，いわゆる因子分析関数との結果を比較することで，パスが引かれていないところはその係数が 0 であるという強い仮定をおいていることを確認する。また尺度作成の観点からは，検証的因子分析をすることで因子的妥当性や弁別的妥当性，収束的妥当性を検討することもできる。さらに同じモデルを別のデータに適用することで多母集団同時分析を行うことになる。このように，モデルの暗黙の過程や，モデルとデータの適合という側面にとくに注意する。さらに測定モデルと測定モデルをつなぐ，構造方程式を扱ったモデルへと拡張する。

→ 小杉･清水 (2014) の Pp.87–90

実践上の注意点　ここまでを通じて，一通りモデルを作成できるようになった。とくに構造方程式を踏まえると，心理的実体同士の関係を描画したと解釈できるため，心理学的概念間の関係を記述できることは魅力的に映るかもしれない。しかしデータを越えての解釈はご法度であり，潜在変数が心理的実在であるかどうかの議論は，理論的背景や測定の適切さ，標準化されないスコアが実際にどのように変化すれば何が言えるのか，といったところに一足飛びに行かぬよう注意する必要がある。またモデル改良のステップにおいて，適合度や修正指数を過度に参照していないか，注意する必要がある。

そのほかの統計パッケージ　構造方程式モデリングの利点は，モデルを可視化したことにもある。たとえば AMOS は GUI でモデルを作成できる。他に Mplus はカテゴリカルな変数にも対応しているし，高度に複雑なモデルであっても表現が可能である。

キーワード

- 測定方程式
- 構造方程式
- 潜在変数を含んだモデル
- 多母集団同時分析
- 適合度

F.13.2　授業情報

■コマの展開方法　講義

予習・復習課題

■予習　構造方程式モデルは，数式レベルでの理解は難しいが実際は統合的なものであり，回帰分析や因子分析をその下位モデルとして含んでいる。改めて，回帰分析や因子分析を単体で行った場合にどのような出力がなされるのか確認しておくと，同じものを構造方程式で実践したときの違いが明確に意識できるようになる。

■復習　さまざまなモデルを試すなかでは，エラーや警告が出ることもある。そうしたエラーや警告の意味を理解し，またそれに対応するためにはどのような方法が取れるかを考える必要がある。まずは手元のデータを用いて，これらの練習を行うと良い。

F.14　双対尺度法

F.14.1　授業内容

科目の中でのこのコマの位置づけ

ここまでは尺度化によって与えられた数値を元に，因子構造を検証したり (潜在) 変数間関係を記述したりすることをみてきた。ここでは尺度化によって与えられる数値に改めて注目し，得られた情報を最も有効に活用するように数字を割り振る，数量化の考え方へと考え方を進める。

因子分析や構造方程式モデリングで用いられる相関関係や共分散関係は，いずれも与えられた数字がどの程度の直線的関係にあるかという観点からモデルを組み立てていた。そのモデルの表現力は非常に豊富なので，スタートとなる変数同士の直線性 (相関係数を使うこと) を改めて問うことがなかった，あるいは多少の違和感を抑えて進んでしまうことがある。改めてその線形性に注意を払い，逆に線形性を最大にするように数値をデータに付与するという発想を転換させるのが，数量化の手法である。

数量化はいくつかの種類があるが，ここでは III 類を取り上げる。III 類は双対尺度法や対応分析とも呼ばれ，行と列の両方に数字を付与する。この方法によってさまざまな表現ができることを確認し，これが応用的側面ではテキストマイニングなどにも利用されているところを見る。こうした応用方法は臨床場面でも活きるものであり，どのようなデータにどのように応用できるかを考えることで，データに対して積極的に関わる姿勢を身につけることが期待される。

コマ主題細目

直線的でない関係　心理学的には中庸が良いことも少なくないが，であればカテゴリに付与される尺度値からは U 字あるいは逆 U 字の関係がえらえる。この関係は相関係数としては 0 に近くとも，無関係，無意味を表すものではない。そこから意味が取り出せないのであれば，数値化のルールを修正するべきであって，どのように関係性の高い数値を与えるかについては，行または列の平均からカテゴリの値を付け直すことである。ここでの目的は，直線性を取り出すためにカテゴリに数値を与える数量化という別の観点であることに注意する。

→ 西里 (2010) の Pp.6–25.

林の数量化理論　分析者にとって最も有用な情報が得られるようにカテゴリに数値を与える，という観点を数量化と呼ぶ。数量化の対象は離散変数や名義尺度水準の変数であってもよく，これらを用いた分析方法については，行動計量学の祖である林知己夫の多彩な手法をまとめた呼称である数量化 I 類，II 類，III 類などがある。I 類，II 類は重回帰分析や判別分析とも関係するが，数量化という観点から議論されていることに注意が必要である。数量化 I 類，II 類の応用例を外観することで分析方の イメージを掴む。

→ 小杉 (2019b) の Pp.189-195.

双対尺度法による分析　数量化 III 類は開発者によって双対尺度法や対応分析など異なる名称を持つが，数学的には等価でいずれも目指すところは名義尺度水準の主成分分析である。リッカート法での尺度に値をつけた時のように，行及び列に含まれるカテゴリカルな区分に対してもっとも線型性が高くなるように数値を割り当てる。ここで行列計算にその目を向ければ，矩形行列に対する特異値分解によって，行の空間と列の空間を用意し，カテゴリに座標を与えることを意味していることになる。数量化 III 類，双対尺度法，対応分析の細かな違いにも注意しつつ，行カテゴリと列カテゴリを共通空間に表した図から何が読み取れるかを考える。

→ 小杉 (2019b) の Pp.195-199.

テキストマイニングへの応用　数量化理論の対象が名義尺度水準であることを考えれば，およそ言語化できたものはすべて多変量解析として分析できることになる。逐語録や自然言語の解析にはテキストマイニングと呼ばれる手法が用いられるが，この技術は形態素解析と多変量解析の組み合わせであり，多変量解析の元になる共変動が何で表されているかに着目すれば，統合的に解釈することが可能である。テキストマイニングには専門的なソフトウェアがあるが，軽く言及するにとどめる。

テキストマイニングについては → 樋口 (2020) を参照せよ。

キーワード

- 数量化
- 双対尺度法
- 特異値分解
- テキストマイニング

F.14.2　授業情報

■コマの展開方法　講義

予習・復習課題

■予習　構造方程式モデリングでさまざまなモデルを考えた際，変数が観測･潜在の別だけでなく尺度水準の組み合わせも変えて考えた場合，どのようなモデルがありえるかが想定できると思います。今回はとくに名義尺度水準のモデルになるので，名義尺度水準のモデルはどのようなものがあるかを想定してみてください。

■復習　名義尺度水準のモデルを手に入れたことによって，さまざまな分析の可能性が広がったのではないでしょうか。今までおよそ統計的･数値的アプローチの対象にないと思われていたものに対しても，どのように，どこまでであればアプローチ可能で，どこからが限界になるのかを考えることが実践的には役に立つ視点です。統計モデルを使いこなすために，ぜひさまざまな応用例を考えてみてください。

F.15 多次元尺度構成法

F.15.1 授業内容

科目の中でのこのコマの位置づけ

数量化まで学ぶことによって，カテゴリに適切な数値を割り振るという尺度化の原点に立ち返ることができた。ここではさらに，心理尺度のような回答法ではない変数間同士の関係から，次元を取り出して分析する多次元尺度構成法について考える。この方法は実験刺激や知覚的反応，直感的判断などを対象にできるため，応用可能性が非常に高いだろう。

多次元尺度構成法を理解するためには，まず実際の距離行列を分析して地図を再構成できるかどうかを見るところから始めるのが良いだろう。データとして与えられるのが距離行列であり，行動計量学ではこれを心理的な距離や意味的な距離が数値化されたものだと捉えることで，心理学的な地図を作っていると解釈してきた。この仮定には最大限の注意を払いつつ，必ずしも計量的でない場合の数値化をする非計量的多次元尺度法に拡張することで，更なる心理学的用途が広がることを見る。なお，多次元尺度方は数量化 IV 類と同じである。

多次元尺度法は分析の元が距離行列であり，その基底を固有値分解によって得るといういみで，多変量解析としてはお馴染みの考え方であるともいえる。しかしデータが距離 (を意味するもの) であれば良く，数学的にも簡単な拡張をすることで，個人差を表現するモデルに拡張することもできる。また心理尺度に対する考え方として，個人の内的な次元からの近さに応じて反応すると考える，展開型のモデルを使うことは，心理尺度の利用に新な視点をもたらす。

コマ主題細目

多次元尺度構成法　多次元尺度構成法 (MDS) は，距離行列を元にした多変量解析の一種であり，距離関係から次元 (基底) を選び出し，対象に座標を与える方法である。まずは座標の復元例から考え，抽出する次元数をどのようにして求めるかといった基礎的な知識を得る。また，行列の考え方からみると正方対称行列の固有値分解であるから，これを確認するだけでも他の多変量解析と合わせた統合的な理解ができるだろう。因子分析モデルも多次元尺度法の一種であるということもできる。

距離と心理学のデータ　距離行列があれば次元が抽出できることが分かったが，さて何を距離とみなすかを考えれば，非常に多くの可能性が広がることがわかる。距離の定義は非負で対称性と三角不等式が成り立つことであり，さまざまな距離の定め方があるし，共分散や相関もその一種と考えることができる。元になるデータも尺度評定を用いる方法，刺激の混同率，代替価/連想価，刺激の汎化勾配，反応潜時，ソシオメトリックなデータなど，心理学のさまざまな領域で得られるデータが，距離とみなすことができる。応用可能な領域が広いことを知ることで，統計モデルをハンドリングできることになる。

→ データの例に関しては高根 (1980) の Pp.14–27.

非計量多次元尺度法　計量 MDS によって算出される座標は元のデータをうまく復元するが，心理学的なデータの場合はデータの大小関係の表現 (順序尺度水準) がせいぜいであり，これに対応した非計量多次元尺度法が考案されている。この手法を用いることで，一対比較や順序比較などのより制限の少ないデータからであっても数量関係を導き出すことができる。

→ 計量・非計量多次元尺度構成方については，すでに絶版になったが岡太・今泉 (1994) がもっとも簡潔でわかりやすく説明している。手に入るところでは足立 (2006) の P.135–143，あるいは小杉

(2019b) の P.199-203.

多次元尺度構成法の展開　多次元尺度構成法で作られたものは地図である。地図には点を書き込んだり，複数の地図を重ねたりできるように，多次元尺度法で得られた地図にも情報を追加したり，モデルを展開するなどしてさまざまな応用的モデルを作ることができる。ここでは Prefmap や楕円モデル (非対称 MDS)，INDSCAL など応用例をいくつか示し，この技術の応用可能性を考える。

展開法　Coombs が考えた心理尺度の展開法は，サーストン法やリッカート法とはまた別の尺度化の考え方を表している。この方法は被験者とカテゴリーの両方を地図上にプロットできる。具体的な分析例をみながら，尺度やそれに数字を与える方法についての考え方を見る。

→ 展開型モデルについては，多少複雑な工夫が組み込まれているが，清水 (2018) が参考になる。

キーワード

- 多次元尺度構成法
- 非計量多次元尺度構成法
- 個人差多次元尺度構成法
- 展開型多次元尺度法

F.15.2　授業情報

■コマの展開方法　講義

予習・復習課題

■予習　数量化の関係を再確認しよう。すなわち数値に値を与えるというものであり，尺度のカテゴリだけでなくより一般的な心理的刺激を考え，どういったモデルで表現できるかを考えると本講だけでなく後期の授業にもつながる気づきを得るだろう。

■復習　多次元尺度法によって，どのような分析ができるかを考えてみよう。とくに尺度法にかかわらず，実験刺激からの反応を距離と見做せる関係にすれば分析でき，その結果をどのように解釈するかについても自由度はかなり多い。紹介された発展的なモデルなどについても自分の研究関心にどのように応用できるか考えてみよう。

F.16　プログラミングの基礎

F.16.1　授業内容

科目の中でのこのコマの位置づけ

後期の授業の主眼は「データから意味のある情報を取り出す」ことにある。すなわち，これまでのデータ駆動型 (Data Driven) な発想を逆転し，データがどのように生成されたと考えるかという観点から，データ生成モデル (Data generating Model) による理解を試みる。

モデルは数学的表現がなされ，モデルの形成やデータとの照合 (fitting) には計算機の利用が必須である。後期の初回となる今回の目的は，プログラミングの基本的な考え方を身につけ，次回以降の本格的な運用に備えることである。プログラミングの基礎はコマンドによる命令であり，コマンドの書き間違えはエラーと

なって帰ってくる。一見不親切に思えるが，即時反応により学習の効率は良く，コード補完機能などを用いることで簡単なミススペルは回避することができる。小さなものから大きくしていくこと，1 行ずつ実行することなど，基本的な姿勢についても理解する。

またプログラミング言語 (高級言語) はいくつかあるが，文法的な違いを除けばその本質は代入，反復，条件分岐である。この三点について理解し，基本的な書き方を学ぶ。実際に R でコードを書きながら，その挙動について確認する。最終的な到達段階として，Fizz-Buzz 問題や行列計算ができるコードを書くこととする。

コマ主題細目

プログラミングの基礎　プログラミングにあたって重要なことは「思った通りに」動くのではなく，「書いた通りに」動くことである。ミススペルや大文字小文字の違いにも注意が必要である。コードのスペルチェックや補完機能を活用し，また 1 行ずつ確認しながら進めるといったプログラミングに関わる基本的な心構えについて理解する。

いくつかのプログラミング言語　R はプログラミング言語の一種であると言っても良い。プログラミング言語には他にも Python や Basic，C 言語などがある。これらの基本的な関係について理解するとともに，コンパイラとインタプリタという実行形式の違いについても理解する。この違いは今後確率的プログラミング言語を利用する際の知識として生きてくる。

高級言語の基本的な働き　高級言語と呼ばれるプログラミング言語の基本的な働きは，代入，反復，条件分岐である。R では`<-`や`=`で代入を，`for` や `while` で反復を，`if` や `if_else` で条件分岐を行う。これらの表現は言語間を通じて共通なものが多いため，その基本的な振る舞いを確認することは技術の一般化に役立つ。

キーワード

- プログラミング言語
- コンパイル
- 代入
- 反復
- 条件分岐

F.16.2　授業情報

■コマの展開方法　講義/遠隔/演習

予習・復習課題

■予習　事前に環境の準備をしておく必要がある。環境の準備についてはいくつかの方策があり，これについては導入資料を参照しながら準備しておくこと。なお，環境準備中に問題が生じた場合はいち早く教員か TA に相談し，実行できるようにしておくこと。

■復習　反復計算の練習課題，条件分岐の練習課題など，複数の課題にしっかり取り組むこと。

F.17　データ生成メカニズムとモデリング

F.17.1　授業内容

科目の中でのこのコマの位置づけ

モデリングアプローチには実質的にベイズ推定が必須であり，そのためにはまず推定法としてのベイズ法の位置付け，ベイズの定理の基礎，実践的方法としての MCMC 法について理解する。とくに，従来の心理統計ではモーメント法と最尤法に言及されるにとどまるものが多い。確率モデルとしての表現は最尤法から導入されており，尤度による推定を事後分布という確率分布で表現するようになったものとして，ベイズの定理を位置付けると良い。MCMC 法は新しい方法であるが，その前に確率分布の特徴を記述するために乱数を利用することができる，という事実を確認すれば，事後分布の記述との相性の良さも明らかである。このように，ベイズ法を過度に新規でまったく異なるものであるという印象を与えることなく，従来の方法の延長線上にあるものとして考えられるようにする。

コマ主題細目

データ生成モデリング　推測統計学が母集団分布における仮定から母数を推定することを目的とし，モーメント法，最尤法，ベイズ法といったアプローチで推定を行うものであったことを再確認する。その上で，帰無仮説検定など心理学一般で使われているモデルは得られた結果のみに基づいてモデル比較をする形である。これは客観性を重視しデータにいかなる前提も置かないことを考えてのアプローチであり，いわばデータ駆動型の分析方法である。これに対して，データがどのように生まれてきたかを考え，その仕組みに基づいて分析する方法がデータ生成モデリングである。データ駆動型分析法は実験方法にそのメカニズムを埋め込んでおり，データ生成モデル駆動型分析法はメカニズムそのものを検証する形になっている。

→ 松浦 (2016) の Pp.18–25.

ベイズ推定の基礎　データ生成モデル駆動型分析にあたっては，未知なるパラメータが多くなるため，実質的にベイズ推定を使うことになる。ベイズ推定の原理となる確率やベイズの公式を再確認することが必要である。

→Kruschke (2014) の Pp104–123 ほか枚挙に遑がない

MCMC　ベイズ推定は事後分布を用いて分析結果を考えることになり，これは結果がパラメータの確率分布として得られることを意味する。そこで確率分布を分析する方法として，乱数を用いてアプローチすることを考える。乱数を用いるアプローチの利点は，積分計算が記述統計量で済むこと，周辺化分布についても当該変数について考えるだけで済むこと，精度を上げる時にサンプル数を増やすだけで良いことなどが挙げられる。また乱数を用いるアプローチに対応したソフトウェアとして JAGS や Stan が挙げられる。これらが確率的プログラミング言語が乱数発生機であることを踏まえ，簡単な実践を行ってみる。

→Kruschke (2014) の Pp.147–194.

乱数によるアプローチの例　簡単な確率分布を使って，乱数によるアプローチを実践する。サンプルサイズを増やすことで精度が上がること，記述統計量が確率分布の特徴を記述することを再確認する。とく

に平均値，中央値，パーセンタイル，分散や標準偏差など，分布の要約統計量の計算について，R スクリプトで算出できるよう練習する。

キーワード

- データ生成モデリング
- ベイズの定理
- マルコフ連鎖モンテカルロ法
- 乱数による近似

F.17.2 授業情報

■コマの展開方法 講義/遠隔/演習

予習・復習課題

■予習 環境の準備が整っていないものは，急ぎ準備を行うこと。また 1 年次の心理学データ解析基礎において，確率分布や乱数を用いるアプローチについても言及はしているので，振り返って「確率という数字の公理」を確認しておくことが望ましい。

■復習 R を用いて乱数を発生させ，理論上の値の近似値になることを確認する。正規分布，ベルヌーイ分布，二項分布，ポアソン分布など複数の既存関数を確かめることで，一般化されない事後分布についての乱数発生機が手に入ったことの重要性に気づくことができるだろう。

F.18 ベイジアンアプローチと確率的プログラミング 1

F.18.1 授業内容

科目の中でのこのコマの位置づけ

ベイズ推定によるモデリングアプローチを始めるにあたって，比較的親近性の高い正規分布を用いた例を導入する。とくに分散/標準偏差に個人差を入れるモデルを用いることで，平均値以外にも推定モデルが考えうることから，モデリングの対象とする領域の広さを意識させる。また確率的プログラミング言語を用いた初の演習でもあるから，コードの書き方，コンパイルなどの手順，出力結果の診断と解釈など，今後の分析に必要な技術的要素についても確認する。

コマ主題細目

7 人の科学者 Lee and Wagenmakers (2013) より「7 人の科学者」の例を紹介する。この例の利点として次の 3 点がある。1. 正規分布を用いていること，2. 平均値以外のパラメータを用いていること，3. 小サンプルからの推論であり，ごく簡単なコードで実演できること。カバーストーリからモデルを想像し，コードに落としていくプロセスをたどりながらモデリングの実際を学ぶ。

→Lee and Wagenmakers (2013) の Pp.48–50.

Stan コードの書き方 カバーストーリーに沿ったモデル図 (設計図) ができれば，後は Stan の言語仕様にそって記述していくだけである。ブロックによる分割，セミコロンによる一行の終わり，変数の宣言と利

用など言語仕様を外観したあとで，モデルブロックからパラメータ，データと逆順に書いていく書き方を試す。尤度と事前分布の違いにも注意し，コメントをつけながらコードを書いていく。

Stan を使った MCMC の実践　Stan コードは Stan ファイルに記載し，コンパイルは Stan そのものを用いるものである。これらファイル，インターフェイス (パッケージ)，実行エンジンなどの関係を明らかにしつつ，MCMC について学ぶ。

→Kruschke (2014) の Pp.407–425.

MCMC の結果の診断　MCMC の結果は Stanfit オブジェクトとして得られるが，それがどう言った情報を持っているか，また MCMC の代表性，正確性，代表性など確認すべき点について理解する。

→Kruschke (2014)Pp.180-194

MCMC の結果の解釈　事後分布からの代表値として適切な MCMC 結果が得られたら，それを用いて結果の解釈を行う。結果はすべて分布として得られているので，確率分布をどのように代表するかについて，前時の復習をしながら進める。また複数のパラメータの同時分布で結果が得られていること，すなわちあるパラメータの点推定値の組み合わせが，同時分布の適切な代表になっていない可能性にも注意が必要である。

同時分布については →Lee and Wagenmakers (2013) の Pp.42–46.

キーワード

- 精度のモデリング
- Stan
- data ブロック，parameters ブロック，model ブロック
- Rhat，有効サンプルサイズ
- EAP，MAP，MED
- 同時分布

F.18.2　授業情報

■コマの展開方法　講義/遠隔/演習

予習・復習課題

■予習　Stan を使う初めての実習になるので，プログラミングの心得 (思った通りに動くのではなく，書いた通りに動く) を再確認しておこう。

■復習　データサイズや MCMC のサンプルサイズを変更するなどして，同じ乱数生成機の何をどう変えればどう変化するのかを確認しておこう。

F.19 モデリングの目から見た検定 1；二群の平均値の差

F.19.1 授業内容

科目の中でのこのコマの位置づけ

データ生成モデリングの観点を踏まえた上で，検定的アプローチとベイズ的アプローチの違いを学ぶ。

二群の平均値の差の検定，いわゆる対応のない t 検定の場合は，同一の正規分布から得られたデータに対して平均値の差があると判断して良いかどうかを判断するという枠組みであった。これらの前提と判断基準を確認し，それがデータ生成モデルの観点ではどのように表されるかを検証する。ここで結果が分布として推定されること，差があるかないかというのは二群の推定された平均値の差であることから，生成量を使って平均値の差を出力することを考える。ここで効果量に改めて目を向けるとその理解が進む。

コマ主題細目

t 検定の仮定 二群の平均値の差を検定するときは t 検定が利用されるが，データが正規分布から得られているという仮定，分散が同じであるという仮定などを踏まえて設計図を書き，これを Stan で表現することを考える。あらためて t 検定のやり方や結果と比べてみることで，モデリングがデータ生成メカニズムに注目していること，パラメータの推定を行なっていることなどが確認できるだろう。また等分散性の仮定を外す方法についてもすぐに応用ができる。

帰無仮説検定と二群の差の検定について，→ 山田・村井 (2004) や一年次の資料をもとに確認しておく。

差の分布 検定は推測に加えて判断を行なっていた，ということを改めて確認するとともに，帰無仮説検定では母平均の差をターゲットにしていたことを確認する。MCMC は母集団からの代表値であるので，推定された結果を使って差を表現することができる。これは R 側で得られたサンプルで行っても良いし，Stan の生成量を使っても良い。ここで `generated quantities` ブロックの考え方を導入し，平均値の差の分布を確認すること，帰無仮説検定が差の分布の一点についての仮説であったことを確認する。また一方が他方よりも大きくなる確率はどれぐらいかとか，一方と他方が c 以上に違っている確率はどれぐらいか，と言ったことが生成量を使って計算することができるようともいえる。

帰無仮説検定を省みる ここまでくると，帰無仮説検定のロジックや考え方について別の視点から見ることができるようになるであろう。まずは帰無仮説と対立仮説という対立のさせ方の不平等さである。帰無仮説は一点についての仮説であり，対立仮説はそれ以外であればなんでも良い，という非対称な関係になっていた。それを省みると，差があるかないかといった二値判断に陥ることがいかに危険であるかがわかるだろう。また量的な判断ができないことから，効果量を合わせて報告することが望ましいとされている。効果量とは，標準化された差の大きさのことであり，生成量を使って簡単に算出することができる。また方向性を持った検定について，片側・両側検定などで考えられてきたが，生成料を使えば自然にそれが検証できることがわかる。ただしこれらの検証の仕方は，今回のデータと仮定されたモデルという前提の上で成立する程度であって，過度な一般化にはならないように注意する必要がある。

キーワード

- t 検定
- 生成量

- 効果量
- 片側検定，両側検定

F.19.2　授業情報

■コマの展開方法　講義/遠隔/演習

予習・復習課題

■予習　Stan の基本的なブロック構成，設計図からコードに落とすやり方を確認しておこう。

■復習　データのサイズが変わるだけでなく，平均値の差，効果量が変化したときの t 検定の結果とベイズ推定の結果がどのように変わるのか，さまざまなケースを想定して「遊んで」みるとよい。加えて，そのほかの仮説検定がどのようなデータ生成メカニズムで表現できるかを考えることは，次回以降の準備にもつながる。

F.20　モデリングの目から見た検定 2；パタメータの世界とデータの世界

F.20.1　授業内容

科目の中でのこのコマの位置づけ

データ生成メカニズムの観点から帰無仮説検定を省みた場合，拙速な結論に飛びつかずに慎重な議論ができることを確認した。また事後予測分布を作ることで，柔軟な仮説を考えられることなども示された。

本時は，同じく事後予測分布を使いながら，パラメータでなくデータのレベルでの比較ができることに言及し，実質的に差があるとはどういうことであるかを考える。パラメータの世界，データの世界を分けて考えられるように注意を促す。まずは事後予測分布をみることで，モデルが現在のデータを正しく再現しているかを見ることで，視覚的にモデルの正しさが検証できることを確認する。その上で，新しく作られた分布の特徴から，閾上率や優越率などを計算することができる。これらはデータに基づく予測であるから，より具体的で実感を得やすい予測として使えるだろう。翻って，仮想データを生成して検証する，パラメータリカバリの手法を学ぶ。この方法では真値やサンプルサイズを自由に設定し検証できることから，例数設計に応用することが可能である。ベイズ推定をしない場合であっても，シミュレーションによる例数設計が有用であることを理解する。

コマ主題細目

事後予測分布　推定値をつかって，新たにデータを生成した場合どのようなことが言えるか。事後予測分布を描くことでモデルの正しさが確認できる。事前分布の特徴を反映した，事前予測分布についても触れる。

事前と事後の予測については →Lee and Wagenmakers (2013) の Pp.38–42 が詳しい。

データレベルの仮説　これまで考えてこられた仮説は，パラメータについての仮説であった。一方，事後予測分布が新しいデータを作っているのであれば，そこからデータレベルの仮説を考えることもできる。閾上率や優越率といった，データのレベルでの仮設を検証したり検討したりすることを考える。これらの視点は帰無仮説検定およびモーメント法による算出よりもわかりやすいかもしれないし，統計的に差があるということがどの程度意味のあることなのかを実感するのにも役立つだろう。

優越率，閾上率については，→ 豊田 (2016)，Pp.69–70.

パラメータ・リカバリ 事後予測分布は乱数発生による新しいデータの生成である。であれば乱数発生のアプローチは，理論に従う仮のデータを生成することができる，ということでもある。シミュレーションとして仮にデータを作ってみて，サンプルサイズがどの程度であればどの程度正確な推定ができるのか，と言った理論的な検証をすることができる。これはパラメータ・リカバリという試みでもあり，モデルが複雑になって行ったときに正しく機能するかどうかをチェックする方法でもある。また，サンプルサイズも自由に変えることができるのだから，どの程度のサンプルがあればどのような結果が得られるのか，と言ったシミュレーション，あるいは実験前のサンプルサイズ設計にもつながる。

モデリングの基礎的手順について，→ 松浦 (2016) の Pp.12–81.

キーワード

- 生成量
- パラメータ・リカバリ
- 事後予測分布
- 例数設計

F.20.2 授業情報

■コマの展開方法 講義/遠隔/演習

予習・復習課題

■予習 `generated quantities` ブロックの書き方について，数値を色々変えて確認しておくとよい。

■復習 さまざまなサンプルサイズ，効果量の仮想データを生成し，帰無仮説検定やベイズ法による推定を繰り返すことで，各手法の長所や短所を考えることができる。遊び心を持って，さまざまな状況生成して，実際に試してみること。

F.21 モデリングの目から見た検定 3；多群の平均値差を求めるモデル

F.21.1 授業内容

科目の中でのこのコマの位置づけ

ここまで `generated quantities` ブロックの活用で，事後分布，事後予測分布を生成できること，パラメータの世界，データの世界それぞれでの仮説が検証できることを見てきた。またパラメータリカバリの方法を見ることで，データ生成モデルを分析前に活用する方法についても見た。

続いて多群の平均値差を求めるモデルを考える。まずは一要因 3 水準の Between モデルから，三つの平均値をバラバラに求めること，生成量から差分を計算することを考える。データやパラメータレベルでの仮説的な検証方法を再確認し，帰無仮説検定の枠組みで考えなければならなかったアルファ水準のインフレ問題が生じないことを，確率の考え方の違いに即して理解する。続いて差をモデルに組み込む方法を考える。ここでパラメータの数に制約をかける方法として `transformed parameters` ブロックの使い方を導入する。ま

たパラメータの数の制約は，自由度の概念と深く関係していることへの洞察を得る。またパラメータリカバリのコードがリバースエンジニアリングのコードと同じもので，鏡合わせの関係にあることを確認する。続いて交互作用が含まれるモデルを考える。ここで制約からどれだけのパラメータが必要か，どのように組み上げるかを学ぶことができる。

コマ主題細目

要因計画モデル　一要因 3 水準 Between モデルを考える。対応のない二群の時のように，これは素直に 3 群のモデルとして表現できるし，群間の差を生成量として計算できることを再確認する。また検定と違って差の大きさを直接検証すること，確率的判断を含まないことから，アルファ水準のインフレ問題に悩む必要がないことがわかる。この考え方は，確率の捉え方の違いにもつながることに留意する。

パラメータの変形と制約　三群のうち，ある群を基準にした差分を直接パラメータとして推定するモデルを考えると，二つの差分を計算することができる。またある群を基準におかなくとも，全体平均を基準に置くことができるが，その場合は差分のパラメータに制約をかける必要がある。これらの制約を含んだモデルを，`transformed parameters` ブロックを使って作ることを確認する。ここでパラメータの自由度について理解する。

モデルの洗練　技術的な問題であるが，多群モデルの場合は一般的に描画するためにも，整然データの形式に整えておくことが望ましい。書いたモデルの一般化という観点から，変数にできるところは変数にするなど，コードの洗練を試みる。ここで群を識別する変数を導入することは，今後の個人内反復測定モデルにも応用できる点であるので，しっかり理解する。

パラメータリカバリ　これらのモデルのパラメータリカバリから，仮想データはデータ生成モデルを逆転させるだけで出来上がことがわかり，要因計画を裏側から眺めるような，新しい観点からの理解が進むと考えられる。

キーワード

- 要因計画
- 整然データ
- 生成量
- パラメータリカバリ

F.21.2　授業情報

■コマの展開方法　講義/遠隔/演習

予習・復習課題

■予習　パラメータリカバリの必要性など，データ生成モデルのアプローチにおける標準的な手順を再確認する。また対応のない二群の平均値差の検定，要因計画の検定についてこれまでの復習をしておくことで，今回の内容の理解が深まるだろう。

■復習　要因数が増えた場合どのようになるか，またその都度 Stan モデルを書き換えなくても良くなるような一般的な書き方について，自分なりに試行錯誤することが望ましい。

F.22 モデリングの目から見た検定 4;対応のある群の比較

F.22.1 授業内容

科目の中でのこのコマの位置づけ

ここまで Between 計画についてのモデル化を勧めてきたが，ここからは Within モデルについて考えることにする。Within モデルは相関係数の考え方と，階層モデルへの入り口として位置付けることができる。

Within モデルは対応がある場合であり，変数間に相関関係を想定することになる。相関関係のあるデータ生成分布は，多変量分布であり，正規分布を多次元正規分布に拡張する必要がある。

多変量正規分布が必要とするパラメータはベクトルと行列であることから，Stan におけるベクトル型，行列型など特殊な型についての理解を進める。

コマ主題細目

対応のある群 対応のあるデータというのは，反復測定あるいは測定間に相関関係が想定されるデータでとして考えることができる。データの間に相関関係がある場合，多次元正規分布から出てくるモデルになることから，これを使ったデータの書き方を習得する。まずは多次元正規分布の考え方とその表現方法を理解する。

ID を持ったデータ構造 既に識別変数をもった整然データのモデル化については習得している。コード化するときは群の識別 ID ではなく個人の識別 ID を持ったコードを作成することになる。代入された変数がさらに代入されるという入子構造のプログラミングに慣れる。

個人差と変化量を想定した書き方 モデルを 3 群以上の比較にすることを考えると，多次元モデルをさらに広げることになるが，別の表現の仕方としてベースラインとそこからの変化量として表現できることになる。この時，ベースラインの散らばりすなわち個人差は，別の分布から出てきていることになる。この表記方法は今後の階層モデルにつながる観点でありことに言及する。

キーワード

- Within 計画
- 分散共分散行列
- ベクトル型と行列型
- 多次元正規分布

F.22.2 授業情報

■コマの展開方法 講義/遠隔/演習

予習・復習課題

■予習 対応のある t 検定や Within モデルなど，伝統的な方法による分析方法について復習しておく。

■復習 パラメータリカバリや身の回りのデータを使って，今回のモデルが具体的にどのように使うことができるかを考えておく。

F.23 モデリングの目から見た検定 5；カテゴリカル分布をつかって

F.23.1 授業内容

科目の中でのこのコマの位置づけ

コマ主題細目

これまでは連続変数についてのモデルばかり扱ってきたが，度数など心理学で扱うデータの中にはカテゴリカルなものも少なくない。これらを帰無仮説検定の文脈で扱う時は，χ^2 検定が有用であるが，カテゴリカルな分布を使ったベイジアンアプローチももちろん，結果の解釈には有用かつ直感的である。

ここでは連続分布と離散分布の違いを確認し，いくつかの代表的な離散分布を演習とともに学び，カテゴリカルな出力変数の分析にも確率モデルが有用であることを確認する。

離散的な分布　変数には離散･連続の違いがあり，確率分布でも確率質量と確率密度の違いがある。ここでは代表的な離散確率変数であるベルヌーイ分布，二項分布，多項分布を導入する。

→ 松浦 (2016) の Pp.82，83，85–89.

χ^2 検定　カテゴリカルな変数の検定についてはこれまで扱って来なかったため，ここで改めて χ^2 分布を使った検定の例を導入する。χ^2 検定は比率 (割合)，独立性，関連の強さ，適合度の検定などに用いられることを確認する。帰無仮説のおき方に注意することと，この検定がモデル適合度などの文脈でも望一られることに言及する。

→ 山内 (2010) の Pp.189-197.

カテゴリカル分布のモデリング　確率分布を用いたアプローチをすることで，母比率を直接検証したり，連言命題が成立する確率など生成量を使ってさまざまな検証ができることを確認する。

→ 豊田 (2016) の Pp.136–163.

κ 係数の算出　変数間の関連の強さを見る指標として κ 係数がある。一致率の係数ともして知られており，記述統計的アプローチでも算出できるものではあるが，確率モデルで表現する場合は工夫が必要である。

→Lee and Wagenmakers (2013) の Pp.56–59.

キーワード

- 離散分布と連続分布
- ベルヌーイ分布
- 二項分布
- 多項分布
- クロス集計表
- κ 係数

F.23.2 授業情報

■コマの展開方法 講義/遠隔/演習

予習・復習課題

■予習 確率変数が連続的か，離散的かということがどういう違いであるのかについて，データ解析基礎の確率に関する資料などを参考に確認しておく。また尺度水準による数値データの分類についても見直しておくと良い。

■復習 カテゴリカルな分類，集計に関しては身の回りに多くのデータ例がある。たとえば官公庁の統計資料などをもとに，母比率の推定や連言命題が成立する確率など，さまざまな仮説を自ら立てて検証すると良い。

F.24 一般化線形モデル

F.24.1 授業内容

科目の中でのこのコマの位置づけ

コマ主題細目

ここまで要因計画が線形モデルと同一であること，すなわち一般線形モデルについて議論されてきた。あらためて回帰分析の確率モデルを考えると，平均に構造を入れたモデルという意味で同じであることが確認できる。ここで確率分布を違う形に変えることでより一般的な線形モデル，一般化線形モデルに拡張することができる。確率モデルによっては結果変数の型の違いによって確率分布が変わり，確率分布によってはパラメータの取りうる範囲が定まるのでリンク関数によって変換する必要があること，結果を解釈するときはリンク関数を経由して分析されていることなどに注意が必要である。まずはパラメータの数が少ないロジスティック回帰について学ぶ。

一般線形モデル 正規分布の平均構造を導入するという意味で，回帰分析のベイズ推定はこれまでと同様に実施，解釈することができる。事後分布や事後予測分布などを使って，最尤推定のモデルと異なる点を確認しておく。

データに合わせた確率分布 データの形によっては確率分布の形を変える必要がある。まずはベルヌーイ分布によるロジスティック回帰分析を通じて，確率分布関数を選択できること，そのためにパラメータの形をリンク関数を経由し得て変換することを学ぶ。

リンク関数とパラメータの解釈 リンク関数を介して線形モデルを考えるので，独立変数が一単位増えることがそのまま従属変数が一単位増えることにはつながらない。これらの関係を知るために，リンク関数，逆リンク関数の関係を辿って考える。ロジスティック回帰の場合は，オッズ，オッズ比などの用語にも触れることになる。またリンク関数による一般化が理解できれば，同様の考え方で他のさまざまな離散確率分布に応用できることが用意に想像できるだろう。

キーワード

- 一般化線形モデル
- ベルヌーイ分布
- ロジスティック回帰分析
- リンク関数
- オッズ比

F.24.2　授業情報

■コマの展開方法　講義/遠隔/演習

予習・復習課題

■予習　確率変数が連続的か，離散的かということがどういう違いであるのかについて，データ解析基礎の確率に関する資料などを参考に確認しておく。また尺度水準による数値データの分類についても見直しておくと良い。

■復習　離散的なデータについて，身近な例を考えてみると良い。また今回導入した分布関数以外にも応用を考えることができるから，確率分布とリンク関数の一覧を参考にさまざまなモデルに思いを馳せてみると良い。

F.25　階層線形モデル

F.25.1　授業内容

科目の中でのこのコマの位置づけ

ここまで線形モデル，一般化線形モデルの例を見てきたが，久保 (2012) の例にならって一般化線形混合モデル，階層ベイズモデルへとモデルを展開させていく。

一般化線形混合モデルについては，Within デザインですでに対応しており，正規分布以外の確率分布を使うことで一般化可能である。ここに切片や傾きなど，係数の方に分布が混ぜ合わせられることで階層化されたモデルとなる。ネストされたデータの具体例として，反復測定と大規模調査の二種類を取り上げ，また階層モデルの設計図を書いてから分析コードを書く手順を確認する。

コマ主題細目

一般化線形混合モデル　Within デザインの分析モデルを再確認するところから考える。このモデルは個体差のような個人ごとに変わる要因が含まれており，ここで変量効果と固定効果の違いを考える。また従属変数が正規分布でないモデルにすることで，一般化線形モデルと考えることができる。

ネストされたデータ　すでに反復測定データの場合が該当するが，データが階層性を持っているネストされたデータの例として，プロ野球データや大規模調査の例を考える。これらに対して，階層化しない分析とする分析とで解釈が異なる例を挙げ，階層的なデータに対して適切な分析が必要であることを理解する。

階層線形モデル　階層化されたモデルを数式的に理解する。名称として，レベル 1/2 の効果，個人/集団レベルの効果と呼ばれることもあることを確認する。またモデルの設計図をかき，それをコードに起こすことで分析ができることを確認する。個別の回帰直線を引く場合に比べて，縮小が起こっていることをモデル比較を通じて確認する。

キーワード

- 混合モデル
- ネストされたデータ
- 固定効果

- 変量効果
- 階層モデル

F.25.2 授業情報

■コマの展開方法 講義/遠隔可/演習

予習・復習課題

■予習 反復測定デザインの分析例を確認しておく。そこで分布が混合されていることを確認しておく。

■復習 階層線形モデルが応用できるようなデータ例を身の回りから考えてみるとよい。その上で，データがどのような背景から生成されているかの設計図を書き，設計図からコードに起こすという手順を一歩ずつ確認しておこう。

F.26 混合分布モデル

F.26.1 授業内容

科目の中でのこのコマの位置づけ

ここまで GLMM，HLM とさまざまな分布を組み合わせて利用するモデルについてみてきた。

今回は混合正規分布モデルと 0 過剰ポアソンモデルを考える。これまではデータが全体的に均質的であることを想定していたが，そもそも異なる種類のデータが混じり合っていると考えられる場合，異なる分布をあてがう方が良い。ここでどちらの分布に属するかがある種の確率で代わり，それに従って続くモデルが異なるという，離散確率分布による条件分岐をデータ生成メカニズムに導入することになる。またこれを Stan で実行する場合には，離散変数が直接扱えないことから，すべての可能な場合を数え上げて足し合わせるという周辺化して消去するというテクニックを利用することになる。技術的に高度な側面もあるが，条件分岐をデータ生成メカニズムに組み合わせることができれば表現力は一気に広がることになる。

コマ主題細目

混合分布モデル 混合分布モデルは確率的なクラスター分析 (Model Based Clustering) でもある。階層線形モデルと違って，クラスター分析はデータの背後にあるクラスが明示的に示されておらず，データの適合から考えることになる。まずデータの可視化によってまずその可能性に気づき，これをどのようにモデル化できるか，そのアイデアを理解する。具体的には，ベルヌーイ分布など離散変数のパラメータによって条件分岐が発生し，各条件のもとで確率モデルが描かれることになるが，この確率モデルをどのように統合するかということを設計図の段階で理解する。設計図での理解を踏まえて，Stan での実装レベルでの理解に進むことができるのだから。

→ 松浦 (2016) の Pp.209–213.

周辺化消去 Stan は離散確率分布を直接モデルの中に組み込むことはできない。そこで `log_sum_exp` という特殊な関数を使って，すべての場合わけを行った離散モデルの統合を考える。まずターゲット記法について理解し，続いて周辺化消去の書き方を理解する。また混合正規分布モデルの場合，ラベルスイッチングの問題が生じることが考えられるから，`ordered vector` の型を利用することが多い。こうした特殊な関数やベクトルについても理解を深める

→ 松浦 (2016) の Pp.203–208.

0 過剰ポアソン データの性質を考えると，必ずしも正規分布モデルばかりではなく，離散変数など一般的なモデルまで拡張することが可能である。そこで野球データなど具体的な分布情報とともに，ゼロ過剰ポアソン分布を使ったモデルを考える。データ生成のメカニズムがより具体的，実践的に考えることができるので，モデリングによる分析の自由度が高まることを実感できるだろう。

→ 松浦 (2016) の Pp.82，83，85–89.

キーワード

- クラスター分析
- 混合分布モデル
- 周辺化消去
- ゼロ過剰ポアソン分布

F.26.2 授業情報

■コマの展開方法 講義/遠隔可/演習

予習・復習課題

■予習 データの描画から気づかされることは無数にある。探索的にデータプロットができるように，`ggplot` やデータハンドリング技術について復習しておくことが望ましい。

■復習 混合分布モデルが応用できるようなデータ例を身の回りから考えてみるとよい。その上で，データがどのような背景から生成されているかの設計図を書き，設計図からコードに起こすという手順を一歩ずつ確認しておこう。

F.27 確率的プログラミングの応用 1; 項目反応理論

F.27.1 授業内容

科目の中でのこのコマの位置づけ

ここまでで線一般形モデル，一般化線形モデル，階層線形モデル，混合分布モデルと定型的な分析モデルについて一通り学んできた。以後はアラカルト的に，確率的プログラミングの応用による柔軟なモデリング例のトピックスを取り上げる。最初に扱うのは，既に前期に学んだ項目反応理論のモデリングである。尺度作成の文脈で，理論的概要は一通り説明が終わっているところであるが，改めて確認するとともに確率的モデリングとして実装する。確率モデルとして考えると，0/1 の反応に対するロジスティック回帰の応用であり，実装自体は既有知識の応用で可能であろう。コーディングのポイントとして，long 型データ (tidy data) にしておくことで欠損値が含まれる場合も対応できるようになることが挙げられる。

コマ主題細目

ロジスティックモデルの復習 本講では前期のうちに，ロジスティックモデルについての理論的説明と，R の `ltm` パッケージによる実践例を解説済みである。とはいえ，以前学んでから時間が空いているの

で，あるいは後期のみ履修する学生もいることが考えられるんので，授業の冒頭で 15 分程度の時間をかけて，大まかな理論・モデルの復習をしておく必要があるだろう。

ロジスティック回帰モデルでの実装　ロジスティック回帰分析を思い出しつつ，1PL,2PL,3PL モデルそれぞれを `transformed parameters` ブロックで記述することを演習で学ぶ。

整然データでの分析　データを整然データの形にして分析することで，欠損値が含まれないデータセットを作って分析に応用することができる。ここでは個人と項目それぞれを識別する変数が必要になるが，これまで学んできた技術で十分対応可能であると考えられる。

キーワード

- 項目反応理論
- 1PL ロジスティックモデル
- 2PL ロジスティックモデル
- 3PL ロジスティックモデル
- 整然データ

F.27.2　授業情報

■コマの展開方法　講義/遠隔可/演習

予習・復習課題

■予習　これまでの知識や技術を組み合わせて問題に対応することになる。項目反応理論とロジスティックモデル，GLM におけるロジスティック回帰分析，データハンドリングにおける整然データの考え方など，これまでの資料に戻って復習しておくと良い。

■復習　自分で描いたモデルが R のパッケージが出す答えとどの程度一致するのかを確認しておこう。また欠損値がある被験者の被験者母数は，その確信区間が広くなると考えられる。なぜそうなるかを改めて考え，実際のデータ適用例で確認しておこう。

F.28　確率的プログラミングの応用 2; 変化点と折線回帰

F.28.1　授業内容

科目の中でのこのコマの位置づけ

コマ主題細目

変化点検出は，時系列的なデータの中に異なる二つの平均値を持つ群があることをモデリングする手法である。とくにある時点から異なる群に属する，という系列的な意味があることと，変化点があるとすればどのあたりになるかという「変化点の位置的不明確さ」を確率分布で表現し，データから検出するという観点は，確率モデルの表現の自由さとデータとの接合を許す確率的プログラミング言語の面白さを味わうには最良の材料である。

まずは混合分布モデルのように，二つの群を分類するモデルを再確認し，その上で時系列的なデータという既有知識から「変化点」という考え方の導入，モデリングへと繋げる。またデータによっては，一定の点を期に線形モデルの傾きが変わるような表現が可能なものがある。この変化点と回帰分析を融合させた，折線回

帰モデルを考えることで，固定的なモデルを超えた柔軟なモデリングが可能であることを理解する。

ただしここで使うデータは時系列的なものであるから，一般的な回帰分析の前提であるサンプルの独立性がない。その意味で不適切なモデルであることに注意し，次回の時系列分析へと繋げる。

混合分布モデル　データは可視化することが重要であり，見れば明らかに異なる状態の混合であることがわかる場合がある。具体例とともに可視化を行い，またこれまで学んだ混合分布モデルで表現できることを再確認する。ここで用いるデータは，小杉の体重記録データを用いる。

変化点検出　データの横軸が時系列的な意味を持つのであれば，時空を超えて二つの群が混合しているというのは不自然な前提である。そこで横軸に時系列的な意味を置くと，ある時点から状態が変化したものとして考えることができる。ここでその時点が「いつ」であるのかは不明であるが，わからないことを確率で表現するのが確率モデルのおもしろい点である。変化点を確率的パラメータとし，その前後で群が異なるというモデルは，変化点検出のモデリングと言われる。このモデリングはポリグラフ検査など，実践的な場面での利用価値も高い。

→Lee and Wagenmakers (2013) の Pp.59–61, 松浦 (2016) の Pp.238-245

折線回帰　平均点の位置が変わるだけでなく，変化の傾向が明らかな場合は線形モデルを当てはめることができる。変化点の前後で傾きが変わるような線形モデルは，折線回帰とも呼ばれる。折線回帰モデルの実装については，変化点と回帰モデルを組み合わせたあとで，折れる点を繋げる数学的補正を加えたモデルへと修正する。最後に，説明変数が時点であることから回帰分析の前提として標本の独立性が担保されていない問題を指摘する。

キーワード

- 混合分布モデル
- 変化点
- 折線回帰
- 時系列分析

F.28.2　授業情報

■コマの展開方法　講義/遠隔可/演習

予習・復習課題

■予習　混合分布モデルの応用になるので，混合分布モデルの基本的な書き方や解析方法について，第 F.26 講を復習しておくことが望ましい。

■復習　折線モデルが応用できそうなデータを見つけて，自分なりに実践してみると理解が深まるだろう。とくに折れる点が複数あるモデルや，折れる点の数を検出するモデルへと拡張するなど，モデル展開の可能性をかんがえることもできる。

F.29 確率的プログラミングの応用 3; 状態空間モデル

F.29.1 授業内容

科目の中でのこのコマの位置づけ

前時に時系列的なデータを導入したが，時系列的な性質を無視したモデリングになっていた。時系列的な分析方法は，心理学においてもウェアラブル端末の利用や SNS など公的なデータを分析することなどにも利用できるため，非常に有用なものになりうる。しかしデータの特徴として非独立性の問題，周期性やトレンドの存在などがあり，周波数解析をおこなったり多次元の行列分解などが必要である。中でも状態空間モデルは比較的シンプルであり，とくにベイジアンアプローチで実装が容易になったと言えるだろう。

ここでは状態空間モデルの基本的な考え方を導入し，モデリングについて解説する。ここで状態と観測の分離を行い，とくに観測が行われていない点があっても分析できること，観測が行われていない点をパラメータとして保管することに言及する。観測が行われていない点が保管できるのであれば，未来の時点についても予測が可能になるということである。ホワイトノイズモデルでそれを行うと，確信区間が広がっていくことが観測される。そこでトレンドを入れたモデルにすることで，さらに予測の形を変えられることを学ぶ。

そのほかにも季節項など，時系列特有の情報を組み込めることや，二次元に展開することで空間データの分析にも応用できることに言及する。

コマ主題細目

時系列データの特徴 時系列的なデータがどのように得られ，どのようなシーンで利用可能であるかを概観する。ここで時系列データはサンプルの独立性が満たされていないという問題があるため単純な線形回帰は不適切であること，また周期性やトレンド，介入効果が出てくるまでの期間など独自に考えなければならないことがいろいろ含まれている。これまで研究されてきた領域や研究方法について概観する。

状態空間モデル さまざまな分析方法がこれまで考えてこられているが，状態空間モデルはその中でも比較的簡単な数理的構造を持ち，またベイジアンモデリングを利用することでかつての分析モデルが必要としたスムージングなどを，特段意識することなく分析できる。状態と観測というモデルの基本構造を提示し，これらがどのように実装可能かをみる。

→ 松浦 (2016) の Pp.229-235, 馬場 (2019) の第 5 部

欠損値の補間 観測時点には欠損が含まれることもあり，これをパラメータとして推定・補間することができる。またこれが可能であるということは，未来の時点を欠損値として考えれば予測ができることにもなる。プログラミングの工夫により，欠損を補間するようなコードの書き方を学ぶ。また単純なホワイトノイズモデルであればあまり予測として意味がないが，トレンドを考えることで時系列的な影響について考えることができる。

状態空間モデルの展開 状態空間モデルは，説明変数を加えた回帰モデルに応用したり，周期性をモデリングすることなども可能である。さらに時系列は一次元的であるが，二次元にも広げると空間分析にも利用が可能であることに言及する。これからの心理学は，時系列や空間など状況変数をより積極的に取り組んだモデルも利用するようになるだろう。

キーワード

- 状態空間モデル
- トレンド
- 補間

F.29.2 授業情報

■コマの展開方法 講義/遠隔可/演習

予習・復習課題

■予習 時系列データを，時間を独立変数とした回帰分析にすることでどういった問題があるのかについて，回帰分析や確率モデルの前提などを考えて振り返っておくことが良い予習になるだろう。

■復習 身の回りの身近ところからでもデータを取ることができるのが，時系列データのおもしろいところでもあるので，応用可能なデータを探して分析してみると良い。可能であれば今日からでも，時系列的なデータを取り始めると，長期的に見て非常に興味深い分析ができるようになるだろう。

F.30 モデル比較

F.30.1 授業内容

科目の中でのこのコマの位置づけ

最終回となるこの回では，これまで後期の授業で扱ってきたさまざまなモデルについて総括し，ベイジアンモデリングの心理学的位置付けについて解説する。加えて最後の話題提供として，モデル比較について言及する。帰無仮説検定の代わりとして考えるのであれば，区間推定を使った比較が必要であるし，ベイジアンモデリングの観点からはモデル比較になる。ベイズファクターによるモデル比較とそれを実行するためのブリッジサンプリング法，また WAIC など予測的観点から評価する方法があることなどに言及する。

コマ主題細目

ベイジアンモデリング 一般的に「モデリング」という観点から，これまでの授業内容だけでなく心理学における研究法としてのその意味や意義を考える。帰無仮説検定の Alternative として利用するだけでなく，心理学的メカニズムをより具体的に，緻密に記載するために数学的方法を用いることで，心のメカニズムの理解を深めることができるかもしれない。そこに含まれる仮定や前提について，自覚的に記述する必要があることがモデリングの利点であり，MCMC をはじめとするベイズ統計の技術は，それを可能にしてくれる方法論的補助にすぎない。

帰無仮説検定の代案 帰無仮説検定の代わりにモデリングアプローチを取ることの利点はさまざま挙げられるが，p 値のように「ここだけ見ておけば良い」というような機械的判断ができる基準がない。むしろそうした機械的判断の弊害が指摘されてきているのであるが，代案としてはどのような基準があるのかはドメイン知識に基づく必要がある。むしろパラメータだけでなくデータの観点から考察できるようになったことや，実質的な値に基づいて考えられることを利点と捉えつつ，判断基準としての ROPE などについて一瞥する。

パラメータ推定かモデル比較かについては，→Kruschke (2014) の Pp.341–361.

モデル比較　モデルとその有用性を考えるにあたって，パラメータ推定かモデル比較かという二つのアプローチがあり得る。後者については，階層モデルによるもの，ベイズファクター，WAIC が考えられる。ただし WAIC については，渡辺ベイズ理論ともいうべき，より包括的なベイズ理論の枠組みで捉え直す必要があり，心理学研究にこの枠組みがどれほど有用かについては，いまだに結論が出ていないところでもある。ここでは概略的にその特徴に触れるにとどまり，受講生諸君の今後の活躍に期待したい。

WAIC については → 浜田他 (2019) が丁寧である。

キーワード

- ROPE
- ベイズファクター
- サヴェージ・ディッキー法
- ブリッジサンプリング

F.30.2　授業情報

■コマの展開方法　講義/遠隔可/演習

参考文献

Abelson, R. P. (1954). A technique and a model for multi-dimensional attitude scaling. *Public Opinion Quarterly*, *18*(4), 405–418.

足立　浩平 (2006). 多変量データ解析法ー心理・教育・社会系のための入門　ナカニシヤ出版，第単行本版，171

岡太　彬訓 (2008). データ分析のための線形代数　共立出版，第単行本版

Allport, G. W. (1967). Attitudes. In Fishbein, M.(Ed.) *Readings in Attitude Theory and Measurement*(pp. 3–13). New York: John Wiley & Sons Inc

Amrhein, V., Greenland, S., & McShane, B. (2019). Scientists rise up against statistical significance.

馬場　真哉 (2019). 実践 Data Science シリーズ R と Stan ではじめる ベイズ統計モデリングによるデータ分析入門 (KS 情報科学専門書)　講談社

Chalmers, R. P. (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software*, *48*(6), 1–29, DOI: http://dx.doi.org/10.18637/jss.v048.i06.

Epskamp, S. (2021). semPlot. https://github.com/SachaEpskamp/semPlot.

藤原　武弘 (2001). 社会的態度の理論・測定・応用　関西学院大学出版会

Gelman, A. et al. (2006). Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian analysis*, *1*(3), 515–534.

Grimm, L. G., & Yarnold, P. R. (1994). *Reading and Undestanding Multivariate Statistics*: American Psychological Association.（グリム，L.G. ・ヤーノルド，P.R. 小杉　考司・高田　菜美・山根　嵩史 (訳)(2016). 研究論文を読み解くための多変量解析入門 基礎篇: 重回帰分析からメタ分析まで　北大路書房）, URL: http://amazon.co.jp/o/ASIN/4762829404/

Grimm, L. G., & Yarnold, P. R. (2001). *Reading and Undestanding More Multivariate Statistics*: American Psychological Association.（グリム，L.G. ・ヤーノルド，P.R. 小杉　考司・高田　菜美・山根　嵩史 (訳)(2016). 研究論文を読み解くための多変量解析入門 応用篇: SEM から生存分析まで　北大路書房）, URL: http://amazon.co.jp/o/ASIN/4762829439/

Gronau, Q. F., Singmann, H., & Wagenmakers, E.-J. (2020). bridgesampling: An R Package for Estimating Normalizing Constants. *Journal of Statistical Software*, *92*(10), 1–29, DOI: http://dx.doi.org/10.18637/jss.v092.i10.

Guilford, J. (1954). *Psychometric Methods*. New York: McGraw-Hill Book Company.（ギルフォード，J.P 秋重　善治 (訳)(1959). 精神測定法　培風館）

南風原　朝和・芝　祐順 (1987). 相関係数および平均値差の解釈のための確率的 な指標　教育心理学研究，*35*(3)，259-265，DOI：http://dx.doi.org/10.5926/jjep1953.35.3_259.

浜田　宏・石田　淳・清水　裕士 (2019). 社会科学のためのベイズ統計モデリング　朝倉書店 , URL: http://amazon.co.jp/o/ASIN/4254128428/

豊田　秀樹 (2012). 項目反応理論 [入門編](第 2 版) (統計ライブラリー)　朝倉書店 , 第単行本版

樋口　耕一 (2020). 社会調査のための計量テキスト分析ー内容分析の継承と発展を目指して【第 2 版】KH Coder オフィシャルブック　ナカニシヤ出版 , 第単行本版，264，URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=4779514746

平岡　和幸・堀　玄 (2004). プログラミングのための線形代数　オーム社 , 第単行本版，355，URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=4274065782

清水　裕士 (2018). 阪神ファンー巨人ファンの 2 大精力構造は本当か　豊田秀樹　(編) たのしいベイズモデリング (pp. 21–32)　北大路書房

Holzinger, K. J., & Swineford, F. (1939). A study in factor analysis: The stability of a bi-factor solution.. *Supplementary educational monographs*.

池田　功毅・平石　界 (2016). 心理学における再現可能性危機:問題の構造と解決策　心理学評論, *59*(1), 3-14, DOI: http://dx.doi.org/10.24602/sjpr.59.1_3.

JASP Team (2021). JASP (Version 0.16)[Computer software]. URL: https://jasp-stats.org/.

J.Dobson, A. (2008). 一般化線形モデル入門 原著第 2 版, 田中　豊・森川　敏彦・山中　竹春・冨田　誠 (訳)　共立出版 , 第単行本版，280，URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=4320018672

川端　一光・荘島　宏二郎 (2014). 心理学のための統計学入門 [心理学のための統計学 1]: ココロのデータ分析　誠信書房 , URL: http://amazon.co.jp/o/ASIN/4414301874/

加藤　健太郎・山田　剛史・川端　一光 (2014). R による項目反応理論　オーム社 , 第 Kindle 版版

小杉　考司 (2018). 言葉と数式で理解する多変量解析入門　北大路書房 , URL: http://ci.nii.ac.jp/ncid/BB27527420

小杉　考司 (2019a). R でらくらく心理統計：RStudio 徹底活用　講談社 , URL: http://ci.nii.ac.jp/ncid/BB27718917

小杉　考司 (2019b). その他の他変量解析　楠見　孝・日本心理学会　(編) 公認心理師の基礎と実践 5 心理学統計法 (pp. 189–206)　遠見書房

小杉　考司・清水　裕士 (編) (2014). M-plus と R による構造方程式モデリング入門　北大路書房 , 第単行本版，332，URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=4762828254

小杉　考司 (2014). 学校適応感尺度 FIT の開発　研究論叢. 第 3 部, 芸術・体育・教育・心理, *64*, 69-82, URL: https://ci.nii.ac.jp/naid/120005596041/.

Kruschke, J. (2014). *Doing Bayesian data analysis 2nd Ed.* NewYork: Elsevier2nd ed.（クルシュケ, J.K 前田　和寛・小杉　考司 (監訳)(2017). ベイズ統計モデリング: R,JAGS, Stan によるチュートリアル 原著第 2 版　共立出版）

Kruskal, B., Joseph (1964a). Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis. *Psychometrika*, *29*(1), 1–27.

Kruskal, B., Joseph (1964b). Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, *29*(2), 115–129.

久保　拓弥 (2012). データ解析のための統計モデリング入門 – 一般化線形モデル・階層ベイズモデル・MCMC (確率と情報の科学)　岩波書店 , 第単行本版，272

Lee, M. D., & Wagenmakers, E.-J. (2013). *Bayesian Cognitive Modeling:A Practical Course*: Cambridge University Press.（マイケル・ D.　リー・エリック-ジャン　ワーゲンメイカーズ井関

龍太 (訳)(2017). ベイズ統計で実践モデリング: 認知モデルのトレーニング　北大路書房), URL: http://amazon.co.jp/o/ASIN/4762829978/
松浦　健太郎 (2016). Stan と R でベイズ統計モデリング (Wonderful R)　共立出版，URL: http://amazon.co.jp/o/ASIN/4320112423/
三中　信宏 (2018). 統計思考の世界 ～曼荼羅で読み解くデータ解析の基礎　技術評論社
宮川　雅巳 (1997). グラフィカルモデリング (統計ライブラリー)　朝倉書店
宮谷　真人・坂田　省吾・林　光緒・坂田　桐子・入戸野　宏・森田　愛子 (編) (2009). 心理学基礎実習マニュアル　北大路書房，第単行本(ソフトカバー)版
村上　正康・佐藤　恒雄・野澤　宗平・稲葉　尚志 (2016). 教養の線形代数　培風館，第単行本版
Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *ETS Research Report Series*, *1992*(1), i–30.
長沼　伸一郎 (2011). 物理数学の直観的方法〈普及版〉 講談社，第 Kindle 版版, 301, URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=B00JQYYCPA
西村　武 (1977). 主観評価の理論と実際　テレビジョン, *31*(5), 369-377, URL: https://cir.nii.ac.jp/crid/1390001205397311616, DOI: http://dx.doi.org/10.3169/itej1954.31.5_369.
西里　静彦 (2010). 行動科学のためのデータ解析—情報把握に適した方法の利用　培風館
Norretranders, T. (2002). ユーザーイリュージョン – 意識という幻想, 柴田　裕之 (訳)　紀伊國屋書店，(トール　ノーレットランダーシュ)
岡太　彬訓 (2008). データ分析のための線形代数　共立出版，URL: http://amazon.co.jp/o/ASIN/4320018591/
岡太　彬訓・今泉　忠 (1994). パソコン多次元尺度構成法　共立出版，第単行本版, 174, URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=4320014723
小野島　昂洋 (2021). lav2tikz.R　, https://github.com/onoshima/myfunction.
小塩　真司 (2020). 性格とは何か　より良く生きるための心理学 (中公新書)　中央公論新社，第 Kindle 版版
Partchev, I., Partchev, M. I., & Suggests, M. (2017). Package ‘irtoys’. *A collection of functions related to item response theory (IRT)*.
Revelle, W. (2021). *psych: Procedures for Psychological, Psychometric, and Personality Research*, Northwestern University. Evanston, Illinois, URL: https://CRAN.R-project.org/package=psych, R package version 2.1.3.
Rizopoulos, D. (2006). ltm: An R package for Latent Variable Modelling and Item Response Theory Analyses. *Journal of Statistical Software*, *17*(5), 1–25, URL: http://www.jstatsoft.org/v17/i05/.
Samejima, F. (1997). Graded response model. In *Handbook of modern item response theory*(pp. 85–100): Springer
芝　祐順 (1979). 因子分析法　東京大学出版会，第単行本版
清水　裕士 (2016). フリーの統計分析ソフト HAD：機能の紹介と統計学習・教育, 研究実践における利用方法の提案　メディア・情報・コミュニケーション研究 (1), 59-73, URL: https://ci.nii.ac.jp/naid/120005744983/.
清水　裕士・荘島　宏二郎 (2017). 社会心理学のための統計学 [心理学のための統計学 3]: 心理尺度の構成と分析　誠信書房，URL: http://amazon.co.jp/o/ASIN/4414301890/
清水　裕士 (2021). 心理学統計法 (放送大学教材 1638)　放送大学教育振興会

Stevens, S. S. (1946). On the theory of scales of measurement. *Science*, *103*(2684), 677-680.
末永　俊郎 (編) (1987). 社会心理学研究入門　東京大学出版会 , 第ハードカバー版
高橋　正視 (2002). 項目反応理論入門―新しい絶対評価　イデア イデア出版局 , 第単行本版, 255, URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=4900561002
高根　芳雄 (1980). 多次元尺度法　東京大学出版会 , 第一版, 332, URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=B000J8ABS0
田中　良久 (1977). 心理学的測定法　東京大学出版会 , 第単行本版
豊田　秀樹 (2000). 共分散構造分析 応用編―構造方程式モデリング (統計ライブラリー)　朝倉書店 , 第単行本版, 303, URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=4254126611
豊田　秀樹 (2007). 共分散構造分析―構造方程式モデリング 理論編 (統計ライブラリー)　朝倉書店 , 第単行本版, 287, URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=4254126964
豊田　秀樹 (2008). データマイニング入門　東京図書
豊田　秀樹 (2016). はじめての 統計データ分析 - ベイズ的〈ポスト p 値時代〉の統計学 - 朝倉書店 , URL: http://amazon.co.jp/o/ASIN/4254122144/
豊田　秀樹 (2017). 実践 ベイズモデリング -解析技法と認知モデル- 朝倉書店
豊田　秀樹 (2018). たのしいベイズモデリング: 事例で拓く研究のフロンティア　北大路書房
豊田　秀樹 (2019). たのしいベイズモデリング 2: 事例で拓く研究のフロンティア　北大路書房
豊田　秀樹 (2020). 瀕死の統計学を救え！　朝倉書店
Van Lissa, C. J. (2019). tidySEM: A tidy workflow for running, reporting, and plotting structural equation models in lavaan or Mplus.. https://github.com/cjvanlissa/tidySEM/.
Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, *27*, 1413–1432, DOI: http://dx.doi.org/10.1007/s11222-016-9696-4.
Walter, I. (2012). スティーブ・ジョブズ 1,2, 井口　耕二 (訳)　講談社 , 第新書版
山田　剛史・村井　潤一郎 (2004). よくわかる心理統計, やわらかアカデミズム・「わかる」シリーズ　ミネルヴァ書房 , URL: http://ci.nii.ac.jp/ncid/BA68747748
山内　光哉 (2010). 心理・教育のための統計法　サイエンス社 , 第第 3 版, URL: http://amazon.co.jp/o/ASIN/4781912354/
永田　靖 (2005). 統計学のための数学入門 30 講　朝倉書店 , 第単行本版
シ　(2016). 計算機言語のまとめノート　暗黒通信団 , 第単行本版, 32, URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=4873100518
千野　直仁・岡田　謙介・佐部利　真吾 (2012). 非対称 MDS の理論と応用　現代数学社 , 第単行本版, 331, URL: https://lead.to/amazon/jp/?op=bt&la=ja&key=4768704050