RとJASP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 基礎 v3

「見えないもの」をみよう・測ろう・考えよう

小杉考司

心理学データ解析基礎

小杉 考司

Last Compiled on 2025.4.17

はじめに

授業のテーマ

「見えないもの」をみよう・測ろう・考えようとするのが心理統計の世界である。

一年を通じて伝えたいポイント

母数を知るための推測　大学での統計，心理統計は標本の記述統計量を検証したいのではなく，母集団を表す数字を求めるための推測統計である。

確率を用いた推論　母数を知るための方法は 3 つある。代表値を用いたモーメント法，確率分布をつかった最尤法，ベイズ法である。

要因計画と線形モデル　心理学では，要因計画による研究条件をコントロールし，平均値の差をもちいた考察 (平均因果効果) をおこなうことが多い。そしてそれらは総じて線形モデルとして表現できる。

モデル比較と意思決定　モデルによる母数の推定だけではなく，そこから一定の「結論」あるいは「意思決定」を行うための方法として，モデル比較や NHST といった方法がある。

統計環境 R による実践　統計環境 R を利用して，理論的な理解だけでなく実際に計算ができることも身につけるべき技術である。

なお，各コマにおける「標準シラバスにおける位置づけ」とは，公認心理師大学カリキュラム　標準シラバス (2018 年 8 月 22 日版)[*1] との対応を表しています。

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/psychometrics_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。

[*1] https://psych.or.jp/wp-content/uploads/2018/04/standard_syllabus_2018-8-22.pdf

目次

第 I 部

心理学データ解析基礎 1A(心理学統計法)

第 1 章

心理統計を始める前に

これから「心理統計学」を学んでいくわけですが，皆さんはそもそも心理学と統計学が深く関わっていることをイメージしていましたでしょうか。もし心理学なのに数学の話があるの？ 心理学と統計になんの関係があるの？ と思った人は，そもそも心理学というのはどういうものをイメージしてましたでしょうか。

第 1 回ではまず，心理学を学ぶにあたってなぜ統計学が必要なのかを考えます。そのためには，心理学が科学 (Science) であること，科学であるとはどういうことかを知る必要があり，そのために数量的表現が有用であるから，ということをしっかりと理解してもらわなければなりません。

1.1 はじめに

みなさん，心理統計の世界，いいえ心理学の世界へようこそいらっしゃいました。

みなさんは心理学という言葉を，はじめて聞いたのはいつだったのか覚えてないのではないでしょうか。それぐらい一般的に使われていますが，さてそこではどのようなことを研究しているのか，皆さんはどんなイメージをお持ちでしょうか。皆さんが何をきっかけで心理学を志したのかによって，それは随分と異なってくると思います。

ある人はテレビドラマ「逃げるは恥だが役に立つ」をみて，主人公の女性が心理学の大学院をでたことから興味を持ったのかもしれません。あるいはスクールカウンセラーが学校にいたから，なんなら少し相談に乗ってもらったから，心理職というのがどのようなものなのか興味を持ったのかもしれません。

あるいはまた，テレビドラマ「科捜研の女」をみて，主人公が働く科学捜査研究所における心理職を知ったのかもしれません[*1]。実際の科捜研における心理職は虚偽検出を行っており，これには (生理) 心理学の知見が生かされています。

他にも高校までの科目として，倫理や現代社会でフロイド，エビングハウス，マズローなどの名前を聞いた人もいるかもしれません。知らなくても今から学ぶので問題ないのですが，それでもフロイドとポリグラフ検査はどう繋がるんだろう，と首を傾げるひともいるでしょう。心理学の知見は他にも，進路相談・職業指導や，街中に溢れるいろいろな工業製品のデザイン，トリックアートから神経生理学，果ては AI(人工知能) などにも関わってきます。

たとえば法学部では法律の勉強をするだろうということはわかりますが，「心理学を学んでいるよ」と誰かに言ってみても「それは何をするの？ 」と聞かれ，「心が読めるのか」とからかわれることもあるかもしれません。道又 (2009) には，次のような一節があります。

[*1] 科捜研には物理，化学，法医，文書，心理の 5 職種があり，文書鑑定 (筆跡鑑定) と心理 (ポリグラフ検査) を一人の技師が兼ねているところもあります。

心理学を「盛岡名物わんこそば」だとしよう。わんこそばはうまいし，おもしろいし，実に素晴らしいものだ。しかし，十分な知識のない人が，わんこそばについて勝手に空想的なイメージ (かわいいちいさなお椀にはいったおいしいおそばなど) を形成して，何の心構えもなく実際のわんこそばを注文したらどうなるだろうか。それはきっとかなり恐ろしい体験に違いない。

皆さんはまるでこの人のように，こんなものが心理学だったのかと驚かれるのではないかと思います。心理学は**名前は知られているが中身を知られていない学問**なのです。心理学が自分のイメージと違うなと思った人，とくにイメージはなかった人，色々いると思いますが，皆さんが今持っているイメージはこの 4 年間ですっかり変わってしまうと思います。少なくとも最初のメッセージとして，ともかく色々な領域に触れて自分のイメージが変わることを恐れないでください，ということを伝えておきたいと思います。(心理) 統計なんて嫌だよ，と思ったひとも，いったん真っ白な気持ちで受講してくれれば嬉しく思います。

1.2　心理学とはどういう学問か

ところで「しんりがく」という言葉はどこで区切るかご存知ですか？「し/んりがく」?「しん/りがく」?「しんり/がく」?

正解は「しん/りがく」です。心の理学，つまり「こころの理(ことわり)」についての学が心理学なのです。さらにいうと，「学」ですから，エンターテインメントや怪しげな術ではなく，知識の体系であるというのが大前提です。そして知識の体系とは，「積み重ねることができるもの」であり「誰にでも共有できるもの」でなければなりません。だからこそ大学で教育できるのです。逆に考えてみるとよくわかります。積み重ねられず，天啓を受けて，あるいは生まれ持った特殊な技能で達成できるものや，誰にも言えない特別な秘伝により限られた人や集団にしか教えられない情報は，学問ではなく宗教や奇跡，マジックや詐欺の領域に属するからです。心理学に神秘的で言葉にできない特殊な何かを求めている人には，残念ですがそれは心理学ではない，とお伝えしなければなりません[*2][*3]。

すでに述べたように，心理学は幅広い領域をカバーしています。それでも大きく分けると 2 つの分類ができるでしょう。下山 (2001) は次の表 1.1 のように分類しています。

表 1.1　下山 (2001) による心理学の分類

アプローチ	実証主義的アプローチ	了解的アプローチ
主な分野	基礎心理学	臨床心理学
手法	自然科学的手法	臨床的手法
対象となる側面	客観的事実	主観的事実
データ収集	実験・調査	臨床実践
データの特徴	量的データ (客観的データ)	質的データ (物語性)
モデルの有効性	理論的な検証可能性	現実状況を理解する際の有効性
対象者との関係	介入しない	積極的な関与 (関与観察)

[*2] 「なんだよー，さっきいろいろな心理学があるって言ったじゃないかよー。本当はこっそり秘伝の術があるんだろう，カウンセリングとか催眠術とかさあ。」と思っている人もいるかもしれませんが，残念ながらそれは心理学どころかそもそも学問じゃないんです。世の中には超心理学だとか，サイコなんとかだとかメンタルなんとかだといって，商売ネタに “心理” というワードを使う輩が後を絶たないから，心理学の誤解は無くならないんですよねえ。

[*3] 余談ですが，大学での学問はすべてこうした「秘伝」を否定するものです。もし言語的に伝えられない合理的でないものがあるならば，それは大学で取り扱うものではありません。もし皆さんが学びたいと思っているのに，合理的な理由なく大学教員が教えてあげないというならば，アカデミックハラスメントに該当します。

ここでは心理学のアプローチを，実証主義的なアプローチと了解的なアプローチに二分しています。一般的には基礎と応用といった名称で呼ばれることもありますが，内容的にはこちらの区分の方が良いでしょう。主な分野としてそれぞれ基礎心理学，臨床心理学が挙げられていますが，基礎心理学のなかには知覚，生理，学習，認知といったさらに細かい学術領域があります。

2 つのアプローチが大きく異なるのは，続く研究の方法論的箇所です。実証主義的アプローチは自然科学的手法を模範とし，客観的事実に基づく実験や調査を行い，理論的な検証・予測を目的とします。一方で了解的アプローチの目的は第一に，クライアントの状態が良くなるところにあり，対象に積極的な関与をしますし，理論よりも現実状況を理解する有効性を目的とするのです。しかしいずれにも共通するのは，これが科学 (Science) であり，データに基づく知識の積み重ねであるという点です。

いずれにせよ，心理学者は心を読んだり，操ったりしているわけではありません。心というものがあるとしたら，どういう機序 (＝メカニズム，仕組み，因果関係) があるかを明らかにしようとしているのです。実は，心というものがあるのか，ないのか，それもわかりません。目にも見えず触ることもできないのですから。あなたの隣にいる人だってよくできたロボットかもしれないですよ？ [*4]そしてこの，対象が客観的・物理的に把握できない，社会的実在や主観的経験であることが心理学の難しさなのです。

心理学は「目に見えない対象を研究する」という難しい課題にチャレンジし続けている学問です。それでいて物理学 (物質の理の学) のように，規則性，法則性を探そうとしている学問です。気を付けないと，超魔術やスピリチュアル，宗教のようなものになってしまいます。心理学は，「知識の体系」＝「学」＝科学 (Science) であることを忘れないで欲しいのです。

1.3 近代科学の特徴

さて心理学が科学であるというならば，そもそもその科学とはなんでしょうか。西洋の歴史を考えてみると，そもそもギリシア哲学から学問が発生し，その後は中世に至るまでキリスト教が「世界を説明する教義」として支配的な考え方を提供していました。これが近代科学になるためには，ニュートンやガリレオなどによるルネサンスを待たねばなりません。そこでは一体どういう考え方の転換があったのでしょうか？

近代科学の特徴は，次の 4 つにまとめることができるでしょう。また，それぞれの考え方に関係した人の名前を挙げています。

- 実験主義：ティコ・ブラーエやケプラーによる天文記録，ガリレオのさまざまな実験
- 数量主義 (**数学的現象主義**)：ガリレオやニュートンによる記述
- 機械論的発想：デカルトや，マクスウェルのデモンに代表される考え方
- 要素還元主義的：デカルトの「第二の格率」，「分割と統合」といった考え方

まず実験主義である，というのは「教科書に書いてあるから確かめなくてもいいでしょ」という考え方への反発です。「本当かどうか，確かめてみようじゃないか」というスタイルです。西洋では長らく聖書に書いてあることが世界の答えでした。聖書には重たいものが軽いものよりも先に落ちる，と書いてありました。だって神様がそうしてくれたのだから，というわけです。このことを疑うのは，神を疑うことであり，そんな人は地獄に落ちるぞ，と言われたりします。しかしこれを確かめてみよう，と実験したのがガリレオです。ピサの斜塔から重たいものと軽いものを同時に落とし，同時に着地することを確認する。実際にやって確かめなければならない，思弁的に論じているだけではダメだ，というわけです。他にも，ティコ・ブラーエやケプラーといった天文学者は天体の動きを何年にもわたって記録し，どういう法則性があるのかを考えていました。これも実際にデータ

[*4] 「スワンプマン」と呼ばれる思考実験の例があります。興味があれば「スワンプマン」で調べてみてください。

をとって考えてみるという意味では実験主義的な態度であり，これが科学の基本的なスタイルなのです。

数量主義あるいは数学的現象主義は，数学で現象を記述しようという考え方です。ニュートンは重力を発見した，と言われますが，正確には力がどのように働くのかを数式で表現したのであり，重力を見たわけではありませんよね。データをとってその関係性を記述することで，世界の理解を深めようと考えているのです[*5]。大事なポイントは，重力が何か (What) ではなく，どのように働くか (How)，という問いの大転換をしたことであり，これが近代科学の基礎を築いたのです。

機械論的発想というのは，「世界は機械＝メカだ。メカニズム (機序) があるんだ」という考え方です。西洋では長らく，世界は神様が作ったものだから，その仕組みを探ったりしたらバチが当たる，と考えられていました。デカルトは「神様が作ったのは「心」で，肉体とかの「体」はいくらでも分解して調べて，神様の考えを勉強させてもらったらええんやで」と考えました[*6]。このメカニズムを探ろう！というのが科学の目的であるということです。

最後に要素還元主義的であるというのは，「要素」に戻していこう，という考え方です。大きな問題を小さな要素に分解していき，確実にわかることを積み重ねていけば，いつかは大きな問題全体がわかる，という考え方です。これもデカルトの考え方で，「方法序説」という本に書いてあるので興味があれば読んでみてください (ルネ・デカルト, 1637 谷川訳 1997)。大問題を小問題に分割していこう，というのも近代科学的アプローチの基本です。

さて，心理学は科学ですから，ここにあげた 4 つの基本的な姿勢はそのまま継承しています。また，この 4 つの特徴の根本にあるのは**私秘性の排除**にあるということにも注意してください。私秘性とは聞き慣れない言葉かもしれませんが，要するに「私だけが知っている秘密」という性質であり，これを徹底的に排除しようとするのが科学的な姿勢なのです。私，というのは一個人はもちろん，教会といった特定の組織，ひいては全知全能の神様すら含まれるでしょう。説明できないところなくし，徹底的に言語化すること，誰にでもわかるように世界を説明することが，科学の究極的な目的なのです[*7]。

ですから心理学も当然，「実験的」で「数量的」で「機械論的」で「要素還元主義的」なアプローチをとります。問題を細分化しつつ，実際に実験や介入をへて，その機序を明らかにしていくわけです。でもここで，なんで数字を使うんだ！と思う人もいるかもしれませんね。誤解しないでいてほしいのですが，数字は「言葉」です。いわゆる日常的な言葉 (日常言語) ではありませんが，科学的な営みを進めていく上では，次の点で日常言語より優れているのです。

- 正確であること；5 点は誰が見ても 5 点。客観的で正確な記述に向いており，相対的に確実な事実の伝達ができる。日常言語で「すごい」「すごくすごい」と言いながら伝えることと比べてみるとすぐにわかる。
- 比較可能性；5 点は 3 点より大きい。確率的判断により予測が可能。日常言語で「やばい」「えぐい」といった副詞の程度を比較することを考えてみると，これも明らか。
- 要約可能性；代表値等で全体を表現できる。簡潔な表現でデータの特徴や傾向がわかる。日常言語で「要するに・・・」といった表現をすることがあるが，どのように要約しているのかも記述できる。
- モデルの検証が可能；モデルにデータを与えて仮説を検証したり，客観的で合理的な結論を導くことができる。

[*5] 神様の存在を疑ったり反抗したりするのではなく，逆に「こんなにも完璧に自然を設計している神様まじパネエ」という信仰心から研究が重ねられていたようです。詳しくは池内 (2014) や蛇蔵 (2014) などがおすすめです。

[*6] 関西弁は著者の意訳によるものです。

[*7] 科学というのはそういう意味で，本質的にコミュニケーションなのです。

1.4 心理学のいとなみ

上で述べたように，数字 (数学) の言葉を使って現象を表現することは，正確で比較検証可能な表現であるだけでなく，膨大な情報を要約することもできるなど，利点は明らかです。ここまで来てもなお，「いや，数字で表現できないことがある。それこそ心だ」と言いたい人がいるかもしれません。

いいでしょう，確かに心理学の研究対象は「心」ですが，それが目に見えず触ることもできないものだというのは認めます。さらに言えば，あるかどうか，その存在すら疑わしいですね。ばかな，心はあるに決まってるじゃないか，という人もいるかもしれませんが，近代科学の作法に則ってそのことを表現してみてください。すなわち，心があるかどうかは私にはわかっている，というのは不適切な論法である (私秘性の排除が必要) ということを踏まえて，考えてみて欲しいのです。

心理学者はこの問題について，次のように考えてきました。つまり，「あるかないかはわからないけど，仮にあるとしよう」，と仮定を置きます。その上で，もし心というものがあるのなら，外部から何らかの刺激を与えた場合，その心を経由して，何らかの反応があるだろう。たとえば人を拳で殴ります (刺激)。するとその人は泣き出すわけです (反応)。なぜそうなるのか，それはきっと「殴られて悲しい」からだ，それこそが心の働きだ，というわけです。これを図にしたものが図 1.1 です。

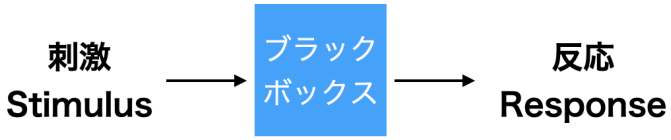

図 1.1 刺激と反応からブラックボックスを探る

心理学では科学的に考えるために，主体 (人間に限らず心があると思われるのであれば何でも) に対してさまざまな刺激を与え，その反応パターンでもって「心があると言えるかも」と考えるのです。これはニュートンが「りんご →?→ 落ちる」の関係から，そこに「重力」があるのかも？ と思ったのと同じ理屈です。彼はこの関係を数式で表現したのでしたね (数学的現象主義)。「こころ」も「こうすれば → 心のせいで → こうなる」という因果を説明する表現方法だといえます。

さきほどの図 1.1 もそうですが具体的な現象を抽象化して表現することで，理解が進んだり新しい解釈ができたりします。この抽象化された，理想的な状態の表現方法を**モデル (model)** と言います。心理学では主体に心というモデルを想定し，もしそのモデルが現実と対応しているのであれば，(理論から導かれた特定の) 刺激に対して，(理論から導かれる結果としての) 反応が現れるはずである，と考えることができます。そこでさまざまな刺激を考え，さまざまな反応を受け止め，心というモデルの正しさを検証しようとしているのです。

もちろん刺激にはいろいろなパターンが考えられます。それこそ無数に考えられてきました。そしてそれに対応する反応もさまざまなものがあるでしょう。1 つの刺激にだって，個人差がありますから，さまざまなバリエーションのある反応が出てくるはずです。こうしたたくさんの刺激 – 反応を正確に記録するために数値化が必要であり，数値化することで要約が可能になり，数式でその関係を表現することで一般化できますし，その数式に実際の数値を代入することで事実に合致しているかどうかを検証できます。このように考えると，心理学で数字を使わないはずがないのです。もちろんそのほかの学問でも，数値化や数字は必須です。物理学のように純粋に数学を扱う領域もあれば，工学のように測定されたデータを扱って物理理論との対応を考える，

いわゆる理系の学問では当然ですが，人文社会科学においても使わないわけはないのです。

統計は，単なる集計に関する学問ではありません。とくに「心理」統計は，心という目に見えないものを見るために，どのように測定するか，どのようにデータを取るか，と言ったところが決定的に重要です。これらは**心理測定法 (psychometrics)** といわれ，心理学のさまざまな領域で考えられています。しかし一度数値化されたデータが手に入れば，それからどのようにメカニズムを解釈するのか，どのようにモデルを立てていくのかについては領域を超えてさまざまな手法があります。それがこの授業，心理学データ解析でみなさんが学ぶべきことなのです。みなさんはこの後，さまざまな専門領域に進んでいきますが，得られたデータを分析するという点については共通した知識・知恵の蓄積がありますので，心理学のごく基礎的な段階から学ぶ必要があります。

数字で心がわからない，と思う人がいるかもしれませんが，これについてはどのように数字にしたのか，という心理測定の問題が理由となっているのではないでしょうか。すなわち，嬉しい気持ち，悲しい気持ちを 4 とか 23 とか言われても，そういうんじゃないんだよな，といいたくなるのはわかりますが，それは嬉しい $=4$ だと誰がどうやって決めたんだ，という問題なのです。仮に数字が正しく心の状態を表現していたとすると，その後その大量の数値から要約したり個人差を除外して，真なる状態を求めることができるかもしれません。近年はあらゆるところに人の行動が記録される機会があります。こうした記録は膨大な量に登るので，ビッグデータと呼ばれています。ビッグデータの中から，人の行動パターンを分析するのも心理統計の範疇です。犯罪プロファイリングやマーケティングなども心理統計の応用的実践例です。もちろん臨床的介入の効果を検証するとか，実験的操作の影響力を評価することも心理統計の守備範囲です。数値化することによって，目に見えなくて触れなくてあるのかないのかもわからないものが，どの程度わからないのか，どこまでならわかるのかといったことも表現できるようになるのです。

心理学と統計法の関係，心理統計の必要性についてご理解いただけましたでしょうか。納得をいただいた上で，心理統計の具体的な話に進んでいきたいと思います。

1.5　課題

■心理学とは何だろうか　みなさんの中で心理学とはどういうイメージを持っていますか。この講義を受けるまえはどのように考えていましたか。この講義を受けた後で何か変わりましたか。できれば今考えている，あなたなりの心理学像を，どこかに書き起こしておくと良いでしょう。前期の終わり，あるいは一年の終わり，そして四年後にそれがどう変わるかを楽しみに。

■心理学にどうして統計が必要なのか　第 1 回は心理統計の中身に入る前に，どうして心理学なのに数字を扱って統計を学ぶのかについてを解説しました。みなさんの身近な人に，みなさんなりの言葉でこのことを説明してみてください。そのときに，心理統計と心理測定の違い，科学としての心理学，というキーワードを入れながら説明できれば完璧です[*8]。

[*8] また後ほど触れることになりますが，ある人が何かを理解している，わかっているということは，そのことについてさまざまな表現を延々と生み出すことができる，ということでもあります。もし言葉に詰まったり，うまく説明できないことがあれば，そこの理解ができていないと考えられます。このことは第??講で触れる古典的テスト理論とも関係しますし，心理学の基本モデルである図 1.1 とも直結する観点です。

第 2 章

電子計算機の基礎

2.1 授業の目的

心理統計の授業では，多くの数値を扱うために計算機を用います。計算機は人間と違って，疲弊したりそのことによって能力に翳りが見えることもなく，指示された通りの計算を粛々とこなしてくれますから，心理統計を実践する上では大きな味方であり，人間の外部にある第二の脳とも言える存在です。ただし気をつけなければいけないのは，計算機は「指示された通りに動く」のであって，「こちらの思った通りに動く」わけではないということです。正しく指示しなければ，正しい結果は得られませんし，間違った指示をすれば間違った結果が出てきます。計算機は人間のように「考える」ことはできませんので，こちらが考えた通りに動くように指示を出す必要があります。

また，皆さんは生まれた頃からインターネットが周りにあり，スマートフォンやタブレット，パソコンが身近にあったと思います[*1]。最近は計算機が含まれていなかったり，インターネットに接続していない電子的デバイスを探す方が難しいほどでしょう。デジタルネイティブな皆さんは，昭和 51 年生まれの私よりもずっと上手に使いこなしていると思います。

ただ，最近の新しい計算機界隈に生まれながらにして親しんできた皆さんは，便利な機械の使い方はご存知でも，仕組みまでは十分理解していないかもしれません。というのも，最近の計算機 (や OS) は親切設計で，ユーザに苦労をさせないような工夫をしているからです。そのこと自体は責められるべきではありませんし，皆さんにはなんの落ち度もないのですが，その工夫が逆に仕組みをわかりにくくしたり，メーカの意図せざる使い方をしてユーザが苦労させられることが少なくありません。というか，計算機がうまく動かない！というトラブルの多くはそれが原因です。

対策として，やはり計算機の仕組みを知っておくことが大切です。計算機の仕組みを知っていると，トラブルに出会ったときに「あぁこれってひょっとして」と思い当たることができたり，納得できるようになります。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。

これまで，情報系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。

[*1] ここまで私は「計算機」と言ってきましたが，これらを全て包括的に表現するつもりでこの言葉を用いています。

2.2　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[*2]。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，気象予報や花粉などの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[*3]。

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[*4]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ッ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

[*2] 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は私が 11 歳の頃です。「ドラゴンクエスト」によって社会問題になったのは私が 12 歳の頃です。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。

[*3] 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。

[*4] 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。

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[*5]。

2.2.1 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, 以下 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[*6]。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[*7]。つまり，まだ「専門的な計算は大型計算

[*5] 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 (1999 柴田訳 2002) を読んでみてください。

[*6] Mac の歴史については Isaacson (1995 井口訳 2011) が読み物としておもしろいですよ。

[*7] 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

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。

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 2.1)[*8]。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。

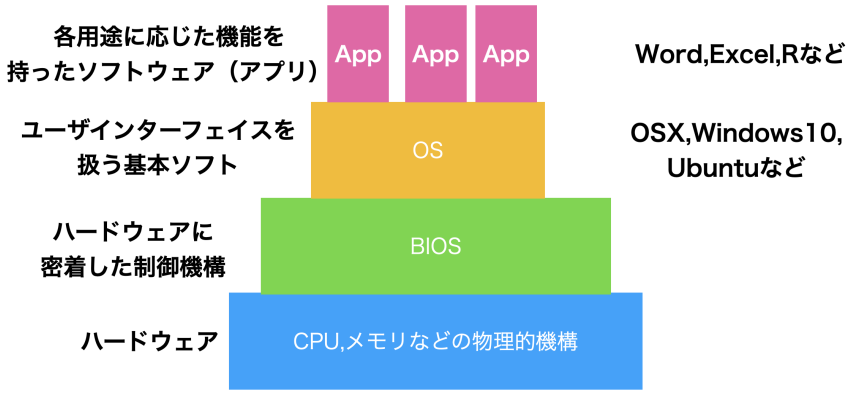

図 2.1 コンピュータのハードウェアとソフトウェアの関係

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[*9]。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

方法でした。

[*8] 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。

[*9] 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 PC 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。

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とをご理解いただきたいと思います。

2.3 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 1 bit(ビット) と言います。1 bit 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 1 byte(バイト) と言います。この 1 byte が 1024 集まったものを 1kb(キロバイト) といいます[*10]。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[*11]。

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，2025 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

[*10] 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。

[*11] 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。

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[*12]。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。1 byte 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 1 byte 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 1 byte 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[*13]。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[*14]，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[*15][*16]。

ところで，1 byte 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 1 byte 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

[*12] 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。

[*13] 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。

[*14] 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。

[*15] 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!

[*16] 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。

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[*17]。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。

2.4 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 PC 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[*18]。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

[*17] 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。

[*18] **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

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[*19]。`.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[*20]。

2.5　ファイルの場所

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 PC A と PC B をケーブルで繋ぐと, PC A から PC B のファイルにアクセスできます[*21]。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[*22]。

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[*23], 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

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。

[*19] 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。

[*20] 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 (半分冗談です)。

[*21] 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。

[*22] 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。

[*23] 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。

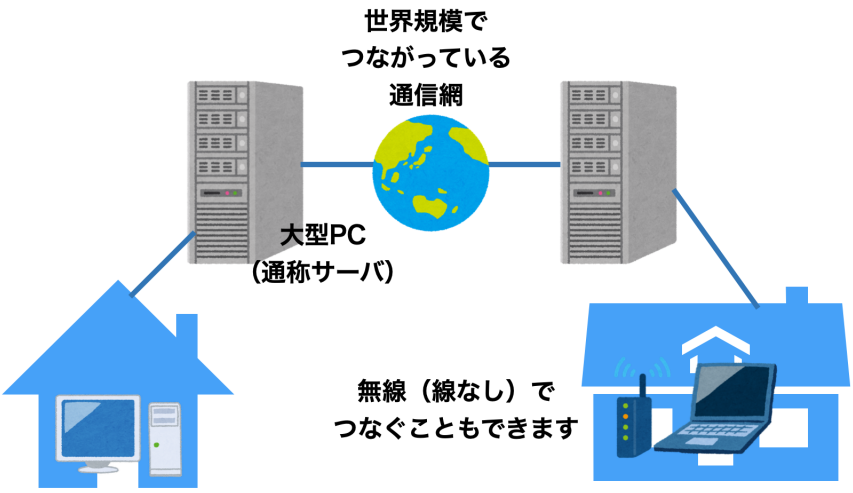

図 2.2 インターネットとクラウド

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 PC にコピーする，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 PC にそのファイルがコピーされているため[*24]，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」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，アプリケーションの実行ファイルは，たとえば Word や Excel などのアプリケーションを起動するためのファイルです。これらは OS がインストールされているローカルに置いておく必要があります。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[*25]。

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

[*24] 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。PC 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。

[*25] 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 (半分冗談です)。

もありますので注意してください[*26]。

余談ですが，R などプログラミングを推している時に最も頻発する質問が「ファイルがありません/読み込めません」です。その理由は大きく 2 つあって，1 つはスペルミスです。ファイル名の大文字・小文字の違いということもありますし，拡張子のまえに半角スペースが入っている，というよくみてみないと気づかないようなものもあります[*27]。2 つめの問題は，パスが通らないことです。デスクトップに置いたのに見つからない，ということもよくあります。教員や TA を悩ませるタイプの問題として，同じファイル名・フォルダ名が別の場所にあることがあります。フォルダ A の中にファイルを置いたのに見つかりません，という質問に対応していて，確かにフォルダ A を開いているのにな，と思ったら別の場所のフォルダ A だった，ということがあるのです。パスは隅々まで同じでなければ意味がないので，自分がどこに何を置いたのかをしっかり把握しておきましょう[*28]

2.6　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説明します。

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 Explorer 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。ファイルをまとめてフォルダにまとめられ，さらに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。

2.6.1　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。

code：2.1　相対パスで読み込む

```
1  dat <- read_csv("sample.csv")
```

[*26] 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。

[*27] k これはたとえば，`sample.csv` というファイルをダウンロードしてね，と指示した時に，複数回同じ場所に保存してしまって，機械が，`sample (1).csv`，`sample (2).csv` などと名前をつける時に起こります。この `(1)` を削除しても，その前に半角スペースが入ってたりするのです。まったく Windows は！

[*28] Windows は OneDrive にもデスクトップがあります。同じ環境をローカルと同時にクラウドにもあって常に同期していればいいのですが，必ずしもそうではないので (OneDrive の設定によります)，`C:\Users\Username\Desktop` と `C:\Users\Username\OneDrive\Desktop` のどちらに保存したのかが分かりにくくなっています。デスクトップに保存したのに，と言われてもどっちのデスクトップなのか，という問題が生じるわけです。さらに専修大学では 2023 年から仮想デスクトップ環境 (VDI) を使うようになりました。これで 3 つ目のデスクトップができたことになるわけです。なんてこった！まったく Windows は！

しかし別の書き方もあります。たとえば code:2.2 のような書き方でも問題ありません。

code : 2.2 絶対パスで読み込む

```
1 dat <- read_csv("C:\User\kosugitti\Document\kiso1\sample.csv")
```

後者 code:2.2 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:2.1 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:2.2 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:2.1 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:2.3)。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 : 2.3 絶対パスで読み込む

```
1 # 絶対パス指定
2 dat <- read_csv("C:\User\kosugitti\Document\kiso2\sample2.csv")
3 # 相対パス指定
4 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。

2.7 キーボードとショートカット

最後に，キーボードの話をしておきましょう。PC を使う上で，マウスを使うことが多いと思いますが，キーボードを使うことも多いと思います。特にプログラミングをする場合は，キーボードを使うことが多くなります。

キーボードは，PC の中にある情報を入力するための装置です。キーボードには，アルファベットや数字，記号などが書かれたボタンが並んでいます。これらのボタンを押すことで，PC に情報を入力することができます。

これから皆さんはキーボードを通じて計算機に指示を与えていくことが増えますが，このとき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。

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，プログラミングでは言語に依存することもありますので，ご注意ください[*29]。

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 2.1 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 2.3)。

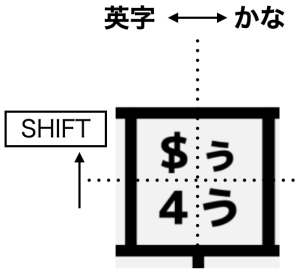

図 2.3　日本語キーで入力する場合

これを踏まえて，日本語キーについては 2.4，US キーについては図 2.5 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[*30]。

これらの配置を覚えた上で，素早く正しく入力できるように普段から慣れておくことが上達への鍵になります。特にこれから先の人生でもキーボード入力をする機会は増えていくはずですから，早いうちに修練した方が良いでしょう。特にプログラミングをする場合は，キーボードを使うことが多くなりますので，ショートカット

[*29] 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。

[*30] 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 2.1 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

キーを覚えておくと便利です。

2.7.1 ショートカットキー

ショートカットキーとは，特定の操作を行うためのキーの組み合わせのことを指します。たとえば，Ctrl キーと C キーを同時に押すことで，選択したテキストをコピーすることができます。このように，ショートカットキーを使うことで，マウスを使わずに素早く操作を行うことができます。

これらのショートカットキーを使うことで，マウスを使わずに素早く操作を行うことができます。特にプログラミングをする場合は，キーボードを使うことが多くなりますので，ショートカットキーを覚えておくと便利です。

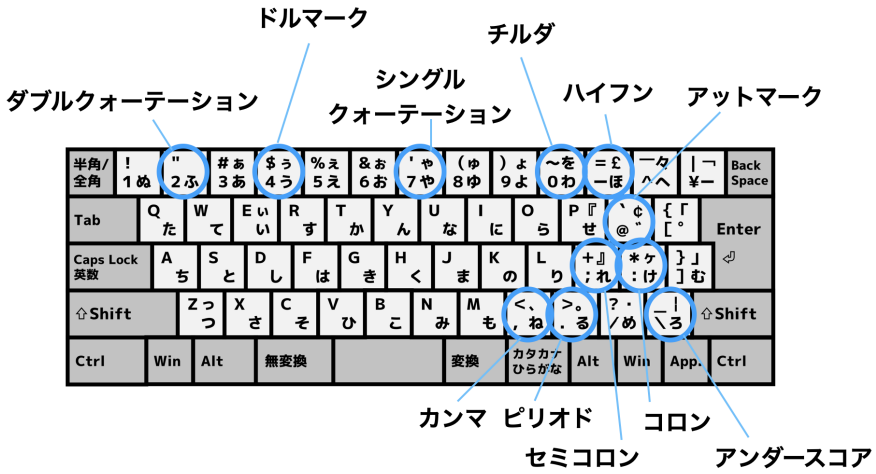

図 2.4　代表的な記号と日本語キー配列

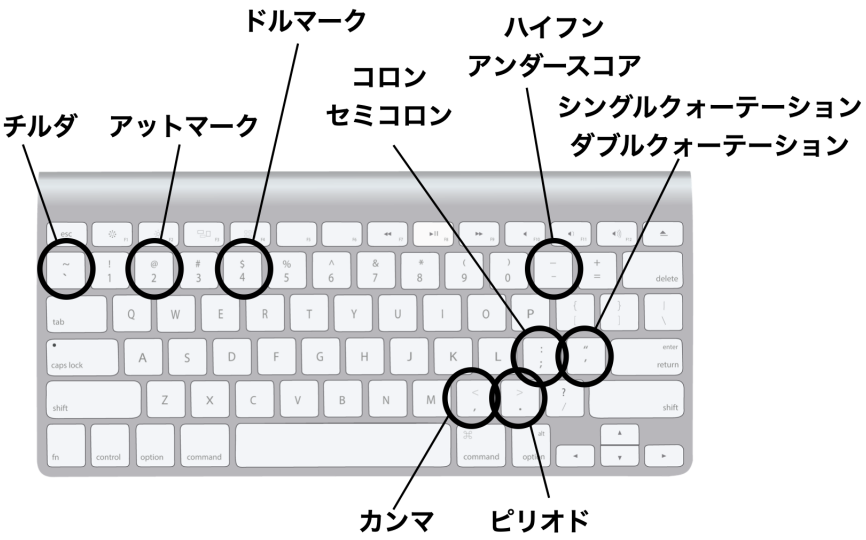

図 2.5　代表的な記号と US キー配列

2.8　まとめ

計算機は，味方にするととても頼りになりますし，人間の計算能力，記憶能力を拡張してくれるデバイスです。現代においては紙とペンと同じような文房具ですので，いかにうまく使うかが重要です。何もしていないのに計算機が壊れた，というようなことはあり得なくて，自分が何をしたか理解していないというのが実態です。もちろん大事な時に意図せぬ動きをされると腹が立ち，実際に困るので嫌いになってしまうことも理解できますが，家族や友人のように好意を持って接し，丁寧に扱うことで素敵な経験を一緒に重ねていくことができます。まずは計算機に名前をつけ，飾ったり (デコったり)，便利な拡張機器を追加するなどして，自分好みの可愛い相棒にしましょう。

表 2.2　よく使うショートカットキー

操作	**Windows**	**Mac**
コピー	Ctrl + C	Command + C
貼り付け	Ctrl + V	Command + V
切り取り	Ctrl + X	Command + X
取り消し	Ctrl + Z	Command + Z

第 II 部

心理学データ解析基礎 1B

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 2.4,Pp.19 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。

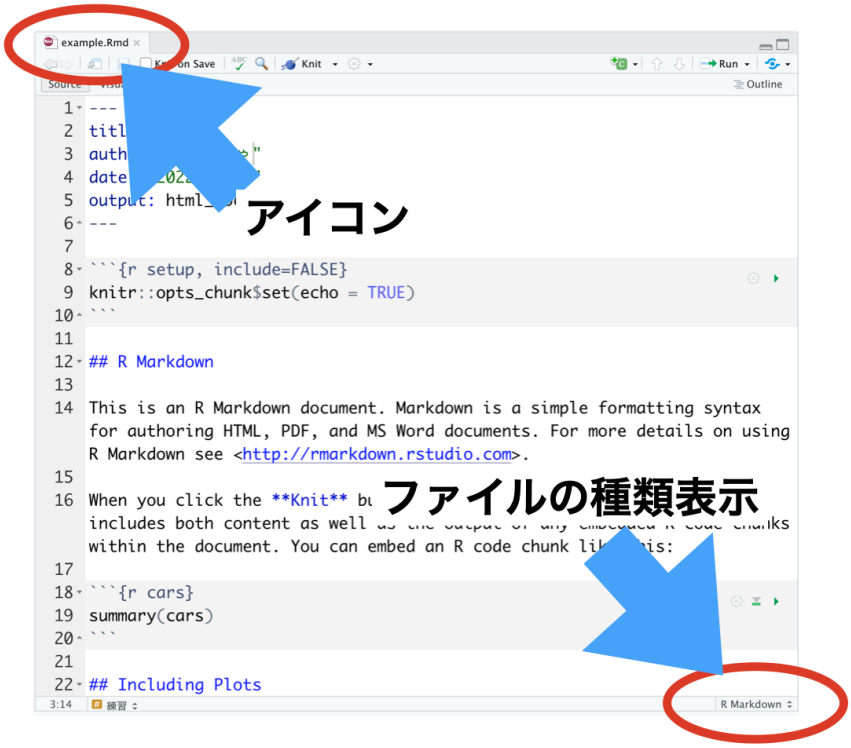

図 A.1 ファイルの種類を判別する方法

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 など他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??,Pp.??を参照してください。

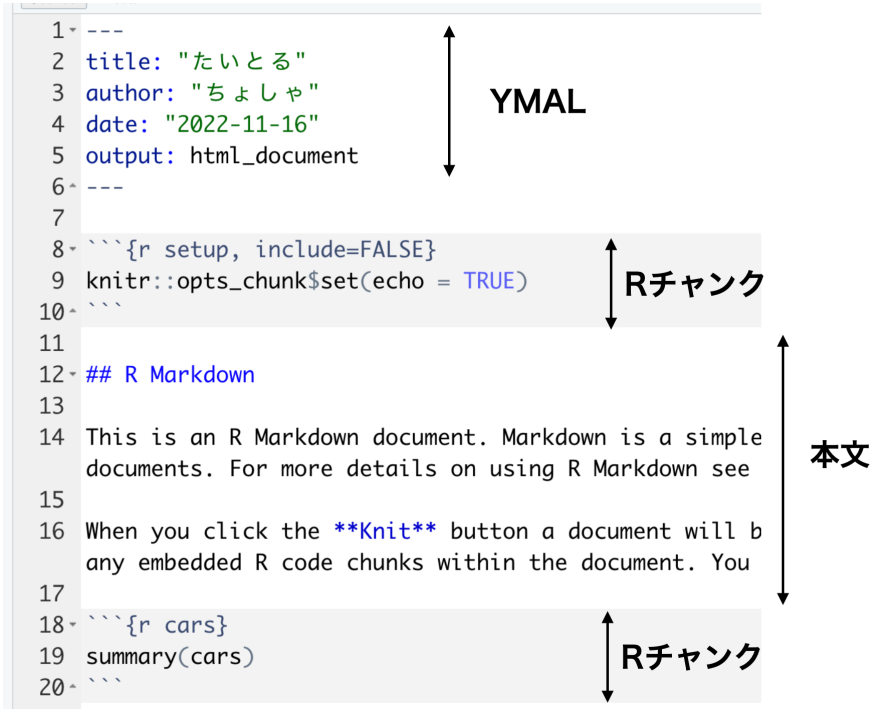

図 A.2　Rmd ファイルの中身における書式

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。

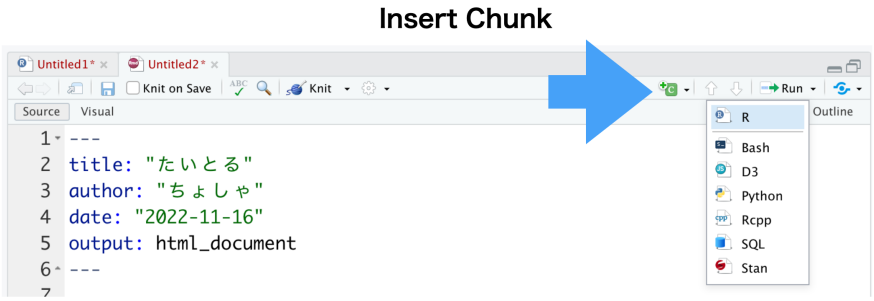

図 A.3　RStudio のチャンク挿入ボタン

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

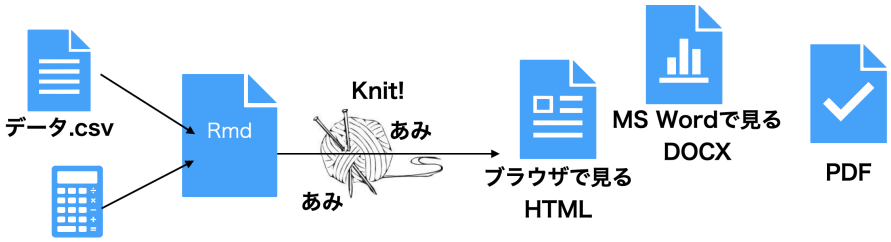

図 A.4 knit してレポートが完成する

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

```
1  dat <- read.csv("social_effects.csv")
2  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生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。

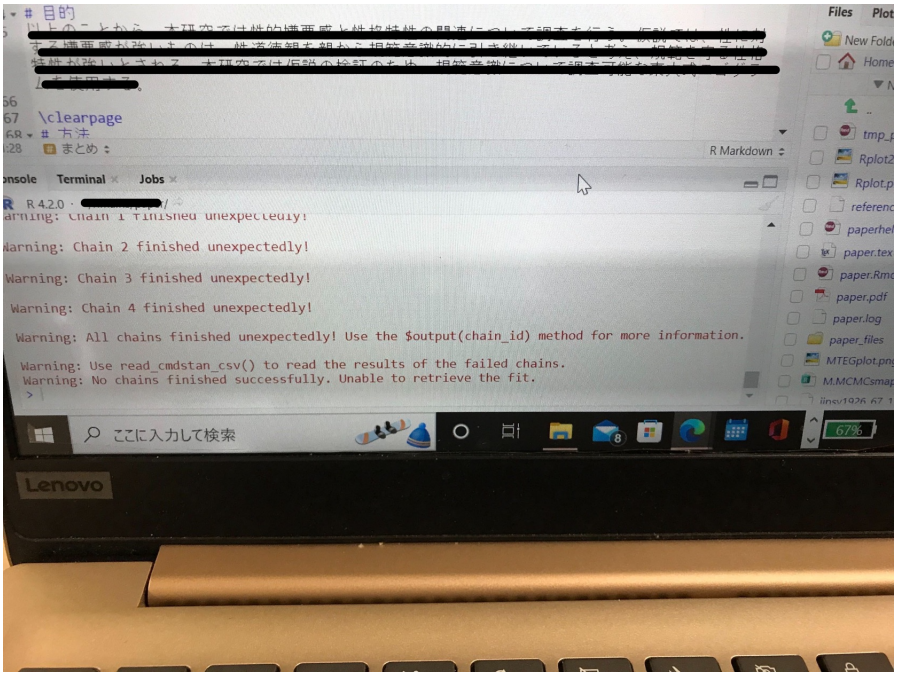

図 A.5 添付されてくるスクリーンショットの一例

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。

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。

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。

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。

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)。
```

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.??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。

Rmarkdown や Quarto ファイルの場合も，必ずプロジェクトフォルダの中に置くようにしてください。これらは Knit するときはファイルの置かれた場所を WorkingDirectory にしますので，プロジェクトフォルダの中にないとパスが通らない問題が頻発します。

[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 2,22 でファイル場所とは何かを再確認してください)。

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。

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。

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 件の行で欠測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測値 (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` で欠測値を除外することができます。

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。

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。

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。

付録 B

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 B.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`

引用文献

蛇蔵　(2014). 決してマネしないでください。(1)　講談社コミックス

池内 了 (2014). 宇宙論と神 (集英社新書)　集英社

Isaacson, Walter. (1995). Steve Jobs. JSTOR.

(アイザクソン, W.　井口 耕二 (訳) (2011). スティーブ・ジョブズ　講談社)

道又 爾 (2009). 心理学入門一歩手前――「心の科学」のパラドックス――　勁草書房

Norretranders, Tor. (1999). The user illusiton. Penguin.

(ノーレットランダーシュ, T.　柴田 裕之 (訳) (2002). ユーザーイリュージョン――意識という幻想――　紀伊國屋書店)

下山 晴彦 (2001). 臨床心理学研究の多様性と可能性　下山 晴彦・丹野 義彦 (編) 講座臨床心理学 (pp. 3–24)　東京大学出版会

Descartes, René. (1637). Discours de la méthode.

(ルネ・デカルト　谷川 多佳子 (訳) (1997). 方法序説　岩波書店)

索引