電子計算機えとせとら

小杉□考司

電子計算機えとせとら

小杉 考司

Last Compiled on 2024.4.13

こちらは心理統計教育教材のサイト (https://kosugitti.github.io/psychometrics_syllabus/) で提供されているテキストに共通してつけられる「付録」です。

パソコンなどの電子計算機は，統計を学ぶ時の強力なパートナーであり，基本的な文房具の一種です。筆者は 1976 年の生まれですから，まだパソコンのことを「マイコン」と呼んでいた時代からの付き合いです。1980 年代の電子計算機は，今から考えると本当に冗談のような演算力，記憶力でした。その発展のスピードは目覚ましく，最近パソコンを触り始めた人にとっては，スマートフォンやタブレットに比べてパソコンは不便で難しいものに映るかもしれません。これは少し悲しい話で，最近はどんどん初心者にわかりやすくしようとするあまり，その内部で何が行われているかを隠すことになってしまっているからです。しっかり理解して使うためには，歴史的な経緯やその内部で何が行われているかについて，概略でもいいので知っておく必要があります。そして得てしてその手の知識は，知っている人からしたら「何を今更」というような内容なので，教えるモチベーションもノウハウも育たないのです。

この資料は，筆者のくだらない思い出話も多々含まれていますが，キーボード配列やギリシア文字の読み方といった，周辺知識からなっています。補助教材としてご利用いただければと思います。

2024 年 4 月 13 日

小杉考司

目次

第 1 章

電子計算機のイロハ

1.1 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。

1.2 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]。

1.3　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

[*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 (1999 柴田訳 2002) を読んでみてください。

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,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 1.1)[*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

[*5] 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。

[*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。

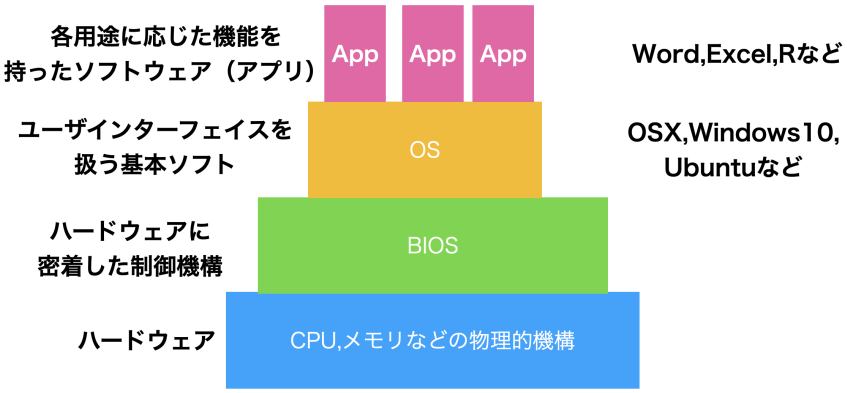

図 1.1　コンピュータのハードウェアとソフトウェアの関係

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[*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

[*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。

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。

1.4　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

[*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。

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[*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。

1.5 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

[*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。

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[*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。

1.6　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

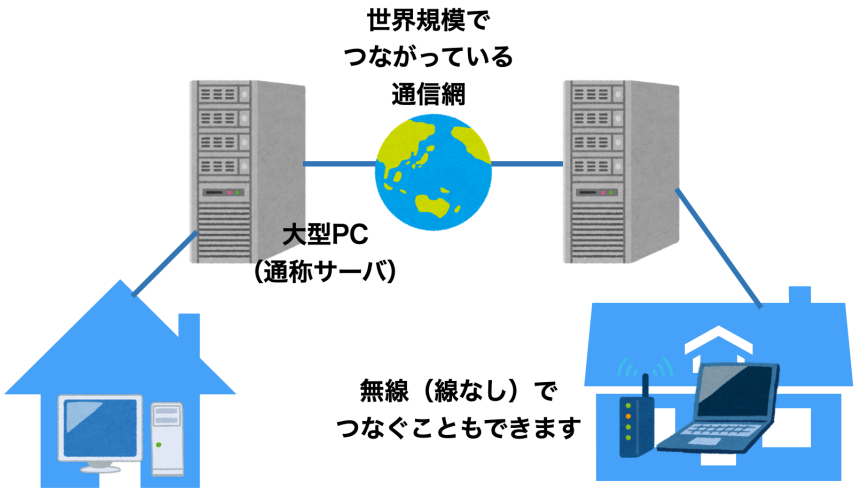

図 1.2　インターネットの世界

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]。

1.7　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。

1.7.1　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 : 1.1 相対パスで読み込む

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

しかし別の書き方もあります。たとえば code:1.2 のような書き方でも問題ありません。

code : 1.2 絶対パスで読み込む

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

後者 code:1.2 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:1.1 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:1.2 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:1.1 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:1.3)。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 : 1.3 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。

1.8　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！

1.9 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。

第 2 章

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 2.1　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`

第 3 章

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 3.1 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 3.1)。

これを踏まえて，日本語キーについては 3.2，US キーについては図 3.3 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 3.1　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

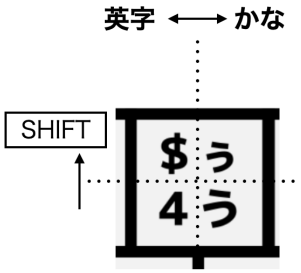

図 3.1　日本語キーで入力する場合

図 3.2　代表的な記号と日本語キー配列

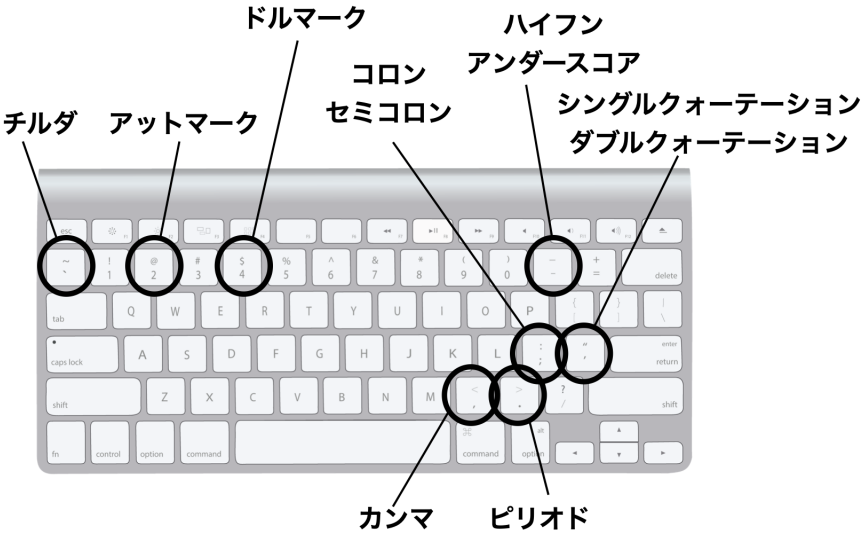

図 3.3　代表的な記号と US キー配列

引用文献

Isaacson, Walter. (2011). 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)
Norretranders, Tor. (2002). 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)

索引

さ