はじめに

このテキストは、慶應義塾湘南藤沢高等部の 5 年情報の授業で用いるものとして書かれたものである。

「情報」という言葉の響きからは、インターネットの情報検索や、プレゼンテーション技術など、情報の受信・発信の技能や、そのためのパソコンの基本的なソフトウェアの修得というイメージがつきまとう。確かに情報通信技術の発展は、情報社会を発展させてきた原動力であり、またそれらを使う能力を修得することを無視するわけにはいかない。しかし、情報を受信し、そこから発信する間で、その情報を人間が処理し理解するという作業が必要となる。そのための技術は、単にコンピュータを使用した受信・発信の方法を理解しただけでは到底取得することはできない。

この授業では、情報を理解するという点に注目し、その処理方法について考えていくものとする。特に、扱う情報としては数値情報に重きをおき、その道具として数学を積極的に利用することにする。これは、数学の応用例の一つであり、「数学が何に役に立つのか」というよくある疑問の回答の一つであるとも考えている。

さらに、プログラミング言語の経験という観点から、GUI システムではない、対話環境を利用して実習を進めていくことにする。この実習で用いるソフトウェアはフリーソフトウェアであるため、自宅での実習も可能である。プログラミング言語は通常の言語と同じく、使わなければ絶対に上達はしない。プログラミングに興味のある者は、入手してみるとよいだろう。

なお、このテキストは 1998 年度の授業プリントを基礎とし、以後 4 年間使用した教材を大幅に改訂したものである。昨年の改訂の際に発生した誤りは、かなりの部分訂正してあるが、それでもなお多数の誤りが残っている可能性がある。疑問な点は遠慮なく指摘していただきたい。

2003 年 4 月　慶應義塾湘南藤沢高等部 数学科

目 次

第1章　R の基礎と一組のデータの処理

1.1 イントロダクション

1.1.1 データ

「データ」という言葉を辞書で調べてみよう。

> データ
>
> 1. 判断や立論のもとになる資料。「—を集める」
> 2. コンピューターの処理の対象となる事実。状態・条件などを表す数値・文字記号。
>
> （松村明編、大辞林、三省堂（1988））

つまり「データ」は、資料、そして時には計算機上の資料を表している。この授業における「データ」は、いずれのものも指すが、実際に扱うものは計算機上で用いることのできる資料とする。

データの例としては、以下のものが考えられる。

- テストの点数
- 身体測定の身長・体重
- アンケートの集計結果のうち、性別や趣味など

上の例をみると、「数値にできるか、できないか」という点で区別することができる。すなわち、テストの点数、身長・体重などの値は数値化できる（またはされている）。実際に扱うのはこれらの数値であろう。このように、数値化できるデータを**量的データ**という。これに対して、アンケートの項目は簡単には数値化できないものがある。このようなものを**質的データ**という。ある種のものは、アンケートの項目のように 1、2、··· と番号をふって数値化することも可能であるが、このようなものは四則計算などをすることには意味がないことが多い。

このように、「データ」といっても、必ずしも数値を扱うわけではない。また、数値となったにしても、それらを画一的に扱うことは好ましくない。

— N O T E —

1.1.2 データ解析と R

現在は情報社会といわれるように、世の中には情報が氾濫しているといってもよい。このようなものの場合、個々のデータについて考えるより、データ全体を眺め、いろいろなことを読み取る、つまり、そのデータが何を語っているのかを探るほうが重要である。そのような手法の体系を、**データ解析**という。

この授業では、データ解析を行うにあたっての道具として、R というものを用いることにする。

R は、データ解析のための言語と環境をそなえたシステムであり、米国 AT&T の Bell 研究所で開発された、データ解析のための言語 S に似せて作ったものである。実際、S の開発に携わった者の一部が、現在でも R の開発に携わっている。

その特徴として、

- 対話的な作業が可能—trial and error というものの繰り返しができる。
- プログラミングが可能—決まりきった動作があるときに、関数定義をして作業の効率化を図ることができる。
- グラフィックスが扱える—視覚に訴える解析が可能。ただし、あくまで解析するために便利であって、見ためが美しいわけではない。

というものがある。

R には、データ解析の環境とともにプログラミング言語という側面を持っている。実際にプログラムを作成することまではしないが、この授業ではデータ解析以外に、いろいろな式の組み立てなど、プログラミングを意識した実習も一部行うこととする。

なお、R には実習用パソコンのように、Microsoft Windows 上で動作するシステムの他に、Apple MacOS や UNIX 系 OS（FreeBSD や Linux などのいわゆる「PC Unix」を含む）上で動作するシステムもある。これらはインターネットから自由に取得できる（むしろ配布を阻害してはならない）ので、興味のある者は取り寄せてみるのもよい。詳しくは巻末を参照のこと。

また、R は現在も進化しているシステムであるため、そのバージョンは次々に上がっていく。追いかけてもよいが、実習で使うレベルの内容は変化がないと思われるので、そんなに神経質にならなくともよい。

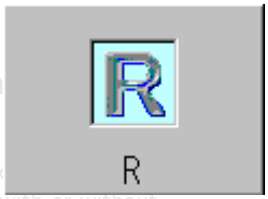
—NOTE—

1.2 R の初期設定

この実習では、テキストを見ながら実際に入力し（またはそれを観察し）、その出力結果を見て考えることが重要である。そのための準備作業を行う。

1.2.1 R の起動と終了

多目的教室 A の計算機にあるメニューのうち、右のボタンが R を起動するためのものである。これを押すと、R が起動する。以下、計算機への入力は以下のような書式を用いることにする。斜体の部分は入力であり、↵は Enter キーを押すことを意味する。



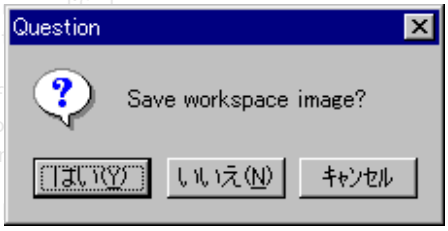
COMPUTER OPERATION ▷
```
> 
```

「>」は、R が入力を催促している状態であることを示す。これを**プロンプト**という。

R を終了するには、> が出ている状態で q() と入力し、リターンキーを押す。

COMPUTER OPERATION ▷
```
> q()↵
```

すると、右のようなウィンドウが開く。これは、現在の作業状態を保存するか、ということを問うている。R は、作業した結果を残し、次に起動したときにそれを復元する機能がある。このため、作業を中断してもあまり問題がない。



ここでは基本的に「はい」を選択する。「いいえ」では作業が全く残らないので、時には大きなダメージになる。誤ってこの状態にしてしまった場合は、「キャンセル」を選択することによって R に戻ることができる。

1.2.2 初期設定

今後の実習のために、R を起動させて、次の作業をすること。

COMPUTER OPERATION ▷
```
> local.setup(2003)↵
```


— N O T E —

◎なぜ R か

諸君（特にコンピュータの知識がある者）の中には「なぜ R？」という疑問がおきるかもしれない。また、この授業で使うシステムは、「対話型」という形式で、文字を入力してその結果を得るという、現在主流の GUI（Graphical User Interface、グラフィックを多用した操作環境）ではないため、最初のうちは戸惑いが多いかもしれない。R を選択した理由を以下にあげておこう。

1. 統計関係を扱うソフトウェアとしては、「本物」である

   Microsoft Excel などの表計算ソフトウェアでも、似たようなことは確かにできる。しかし、その実装には無理が多い。本格的に使うシステムとして、表計算ソフトウェアはなんとも中途半端なものなのである。もともと「表」を扱うのが得意なものであるが、データ解析には「表」以外のデータも扱うことがあるため、そのようなことを実現させるには無理があるのである。

2. 統計関係を扱うソフトウェアとしては、価格が極端に安い

   世間一般に販売されている、統計関係のソフトウェアの値段は、少なくとも 1 本あたりで 5 桁の金額がする。6 桁の金額のものもざらにある。そのようなものをおいそれとは導入できないし、パソコンを所有する生徒が自宅で実習するのも難しいだろう。なんといっても、R は本体価格は 0 である。以前は本校でも、販売されているものを使っていたが、現在 R の出現とともにその環境に移行させている最中である。

3. 言語環境としては、意外と初心者にやさしい

   「なぜ C や Java ではないのか？」との意見もよく聞かれるが、このような言語は「プログラムをエディタで書き」「コンパイルという作業を行い」「必要に応じて間違いを直す」ということを繰り返す。「プログラム」を書かなければ絶対になにもできないのである。この授業は、「プログラム」の作成は**メインの内容ではない**。しかし、**その雰囲気を味わう**ことも目標の一つであり、そのためには、「入力したものに対してすぐ反応が返ってくる」対話型の言語のほうが都合がよい。

もちろん、R よりよい環境も存在するかもしれないので、以上の条件を満たすような R 以外の環境が存在するのならば、知らせていただきたい。

— N O T E —

1.3 R における式

基本的に、

```
> ■
```

で待っているときは「式」を入力する。式を入力すると何かしらの反応（それがエラーを示すものであっても）が返ってくる。

1.3.1 計算機としての R

「計算機」と名乗っている以上、計算ができるのは当然と考えてもよいであろう。R では、式を入力すれば、通常の四則計算の順序にほぼしたがって計算をしてくれる。

たとえば、$2+3$ を計算したければ、

COMPUTER OPERATION ▷

```
> 2+3↵
[1] 5
```

と入力する。先頭の [1] は次節以降で利用するため、**今は**無視してよい。5 が結果である。空白は自由にいれてよいので、先の例では

```
> 2␣+␣3↵
```

としてもよい。ただし、数の 20 を $2␣0$ というように、ひとかたまりで 1 つの意味をなす文字の並びの途中に空白をいれることはできない。以下、見やすいように適宜空白をいれることにする。

計算は、普通の式と同じように入力すればよい。記号の優先順位は数学のものと同じである。

記号	例	その意味	記号	例	その意味
加法 +	3 + 5	$3+5$	減法 -	7 - 2	$7-2$
乗法 *	7 * 8	7×8	除法 /	7 / 8	$7\div 8$
負の符号 -	-4	-4	べき ^	-4^3	-4^3

記号	例	その意味
かっこ ()	(3-(7-(4+8)/3))*2	$[3-\{7-(4+8)\div 3\}]\times 2$

編集は、マウスは使えない。キーボードの操作になる。

→	カーソルを右に 1 つ移動	←	カーソルを左に 1 つ移動
↑	1 つ前に入力したものを表示	↓	1 つ後に入力したものを表示
HOME	カーソルを行頭に移動	END	カーソルを行末に移動
BackSpace	カーソル前にある文字を消去	Delete	カーソル上にある文字を消去

— N O T E —

開きかっこが閉じかっこより多い状態で改行すると、式が完成されていないとみなされ、プロンプトが+となって入力待ちとなる。たとえば $3+4\times(2-5)$ を計算しようとして

COMPUTER OPERATION ▷
```
> 3+4*(2-5↵
+ 
```

となった場合は、閉じかっこが 1 つ足りないので、

```
+ )↵
```

とすればよい。式の修復をあきらめて新たに入力しなおすときは ESC を押す。

問 1 次の式を計算する R の式を入力せよ。

(1) $(-3)^3$

(2) $3+20\div4\times9$

(3) $(-2)^3\div(2+7\times2-4\times5)$

(4) $\{(3+2)\times(6-8)-2\times(-1)\}\div(-4)$

(5) $7\div\{5\times7-3\times2+4\times(-5)-3^2\}$

COMPUTER OPERATION ▷
```
>
```

最後の式は、$5\times7-3\times2+4\times(-5)-3^2$ の結果が何であるか考えてみるとよい。R で計算してみよ。

COMPUTER OPERATION ▷
```
>
```

0 による除法や、$\log_{10}(-1)$ など、成り立たない計算が発生した場合、S は Inf（無限大）や NaN（非数値）といった値を返す。これらは、たいてい希望しない値だろう。

なお、電子計算機の計算は、微妙な誤差を含むことがある。たとえば、$0.5-0.4-0.1$ は数学では当然 0 だが、これを R で計算させると

COMPUTER OPERATION ▷
```
> 0.5-0.4-0.1↵
[1] -2.775558e-17
```

となる。`-2.775558e-17` というのは -2.775558×10^{-17} という意味で、0 に非常に近い値だが、0 ではない。このような誤差をいつでも完全になくすことはできない。

非常に大きな数、たとえば 26343200000000000 は 2.63432×10^{16} だが、これは `2.63432e+16` と表現する。

— N O T E —

1.4 オブジェクト

R では**オブジェクト**というものが作業中重要な役割を果たす。いろいろな計算をしたものはすべてオブジェクトになるため、その扱いや作成方法について解説する。

1.4.1 オブジェクトと付値

R において、計算にかかわるものをすべて**オブジェクト（object）**とよぶ。実際に注目すべきオブジェクトの一つとして、値を保存するオブジェクトがある。式を入力しても、そのままでは計算結果を表示して、計算機内部からは消えてしまう。再利用する値は保存するとよい。値をオブジェクトに与えることを**付値**という。

オブジェクトの名前には、英数字（大文字・小文字は区別する）とピリオドを使うことができる。

たとえば、オブジェクト x に 2 という値を付値するためには、つぎのどれかを行えばよい。

COMPUTER OPERATION ▷
```
> x <- 2↵
> 2 -> x↵
```

<- と -> は 2 文字で 1 つの記号である。<␣- などと空白をいれてはならない。以後、付値は ← と → で書くことにする。この記号を使えば、次のように書き直される。

```
> x ← 2↵
> 2 → x↵
```

付値の記号は最後に評価される（式の値を求めることを**評価する**という）。したがって、計算結果を付値するには、たとえば次のようにする。付値の場合は計算結果は表示されない。

COMPUTER OPERATION ▷
```
> x ← 4+(9-3)/6↵
> 5-(8-(2/6-1)*3) → y↵
```

問 1 オブジェクト x に −3 を付値せよ。どうすれば見やすくできるか。

COMPUTER OPERATION ▷
```
>
```

オブジェクトの値を見るには、オブジェクトの名前だけを入力する。また、オブジェクトの値は式の中で利用することもできる。

COMPUTER OPERATION ▷
```
> y↵

> 3*(x+3)/4-1↵
```

— N O T E —

値が付値されていないオブジェクトは用いることはできない。一方、一旦付値されたオブジェクトは、消さない限り残っている。次の例でそれを確認せよ。

COMPUTER OPERATION ▷
```
> zzz↵
Error: Object "zzz" not found
> q()↵
（R を終了させ、改めて起動する）
> y↵
```

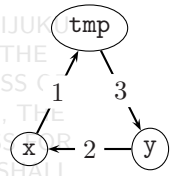

1.4.2 オブジェクトのコピーと中身の交換

COMPUTER OPERATION ▷
```
> y ← x
```
というのは、x の中身を y に付値するので、コピーになる。

すでに付値されているオブジェクトに対して改めて付値を行うと、内容が上書きされて以前の内容は消える。したがって、オブジェクト x と オブジェクト y の内容を入れ替えるために、

COMPUTER OPERATION ▷
```
> y ← x↵
> x ← y↵
```
としてもうまくいかない。これでは x と y が等しくなってしまう。

問 2 オブジェクト x と オブジェクト y の内容を入れ替えるためにはどうすればよいか。右の図を見て考えよ。



COMPUTER OPERATION ▷
```
>
>
>
```

◎オブジェクトの消し方・作成したオブジェクトの一覧

作成したオブジェクトの一覧を見るには次のようにする。

COMPUTER OPERATION ▷
```
> ls()↵
```
誤って作成したオブジェクトを消すには次のようにすればよい。

COMPUTER OPERATION ▷
```
> rm(消したいオブジェクトの名前)↵
```
消したいオブジェクトが複数ある場合、, で区切って並べればよい。

なお、消すことができるのは、自分で作成したオブジェクトだけである。また、**一旦消してしまったものを復活させることはできない**と考えおくとよい。（厳密には、R の終了時の質問に「いいえ」で答えておけば消えないが、こうするとこんどは R 起動後に作成したオブジェクトが消えてしまう）。

— N O T E —

1.5 ベクトルとは

データ解析で用いるデータは、たいていはある程度たくさんの数の集まりであることが多い。このようなものを扱うときに、ベクトルは便利である。

1.5.1 ベクトルと関数

データの並びを**ベクトル（vector）**という。ここでは、60 人のあるテストの点数のデータを、オブジェクト s.point1 として用意しておいた。

COMPUTER OPERATION ▷

```
> s.point1↵
 [1] 63 53 65 57 56 57 53 58 58 52 68 55 55 64 64 68 60 55 58 57
[21] 56 68 43 66 57 55 57 58 58 50 46 56 57 67 56 64 63 51 75 51
[41] 44 70 45 66 56 48 55 60 55 66 49 64 60 58 67 54 60 66 45 68
```

ベクトルを表示させると、改行ごとに先頭に［n］という形の表示がある。n は自然数である。これは、先頭の要素がベクトルの n 番めの要素であることを表している。通常のデータは、要素の数が 1 のベクトルとみなすことができる。なお、ベクトルの要素の数を、ベクトルの**長さ**という。

ベクトルの長さを知ろうとするとき、表示されたものをいちいち数えるのではなく、計算機に数えさせたい。ここで、「ベクトルの長さを数えろ」と R に働きかけをしなければならないが、このようなものを R では**関数（function）**という。関数は必ずその名前の後ろに（ ）（かっこ）をつけなければならない。かっこの中のものをその関数の**引数（argument）**といい、R に働きかけをするための材料となる。

ベクトルの長さは、関数 length() でわかる。

COMPUTER OPERATION ▷

```
> length(s.point1)↵
```

式の中に関数があると、R は引数に与えられた材料を用いて処理を行い、結果をオブジェクトとして返す。これは、ちょうど数学の関数と同じ働きである。関数の結果は式の中で再利用できるので、たとえば

COMPUTER OPERATION ▷

```
> length(s.point1) - 1↵
```

とすれば、s.point1 の長さから 1 を引いたものが返ってくる。

数学らしい関数も紹介しておこう。sqrt() という関数はかっこの中の平方根を求めてそれを返す。

COMPUTER OPERATION ▷

```
> sqrt(5)↵
```

— N O T E —

関数は本来何らかのオブジェクトを返すものなのだが、関数の中には R の動作そのものに働きかけるものもある。このように、計算以外の効果をその関数の**副作用**という。q() というのも関数で、これは「R を終了させる」という働きを持つものである。

ベクトルを作るときは、関数 c() を使う。並べたいものを () の中に並べればよい。すべて並べたものをベクトルオブジェクトとして返してくる。

COMPUTER OPERATION ▷
```
> x ← c(7, 5, sqrt(5))↵
> x↵
```

c() はベクトルどうしをつなげるときにも用いる。

COMPUTER OPERATION ▷
```
> y ← c(3, 1, 9)↵
> x ← c(1, x, y, 4)↵
> x↵
```

ここまでで出てきた関数をまとめておこう。

`q()`	R を終了させる。
`length(x)`	ベクトルオブジェクト x の長さを求める。
`sqrt(x)`	オブジェクト x の平方根を求める。
`c(...)`	かっこの中の要素をすべて使ってベクトルを作る。
`ls()`	作成したベクトルの一覧を表示する。
`rm(...)`	かっこの中に並べたオブジェクトを消す。

◎ 関数もオブジェクト !?

実は、関数もオブジェクトである。だから、たとえば

COMPUTER OPERATION ▷
```
> ls↵
```

などとするとわけのわからないものが表示される。これは、この関数の挙動について定義しているものである（ただし、実体が見えやすいものとは限らない）。さらに、R では自分で関数を作ることもでき、これがプログラミングにつながるのである。ただし、ここではプログラミングについては触れない。

R では、扱うものは実は何でもオブジェクトと呼ぶ。「オブジェクト」が処理の主体であり、このような考え方を「オブジェクト指向」という。

1.6 データの図示とデータを表す数値

データが与えられたとき、まず考えるべきことは、「どのような感じのデータなのか」をつかむことである。そのためには、「データの全体の様子を図からつかむ」ことと、「データをその様子がわかるいくつかの数値で代表させて表現する」ことを行う必要がある。R ではこの両方を行うことができる。

1.6.1 ヒストグラム

データが与えられたとき、「ある範囲のデータがどのくらいあるだろうか」ということを確認するのは重要である。このとき、それを図示できるとよいだろう。R ではグラフィックスを使ってそれを視覚的に確認することができる。

一番確認しやすい図として**ヒストグラム（histgram）**がある。これは、データをいくつかの範囲（これを**階級**という）に分け、その範囲にどのくらいのデータがあるか（これを**度数**という）を柱状のグラフにしたものである。`hist()` 関数でこれを作ることができる。引数にはヒストグラムにするベクトルオブジェクトを与える。

COMPUTER OPERATION ▷
```
> hist(s.point1)↵
```

このようにすると、グラフィックスウィンドウが新たに開き、ヒストグラムが表示される。範囲は勝手に決められるが、横の目盛りの値には注意すること。

比較のために、`s.point2` というデータも用意してあるが、そのまま表示するだけでは比較がしにくい。それぞれの画面で横の目盛りが自動的に変化してしまうためで、「目盛りを固定する」「これらを並べる」ことができれば比較がしやすい。

ヒストグラムで、横軸の範囲を 0 から 100 にしたいのならば、

COMPUTER OPERATION ▷
```
> hist(s.point1, xlim=c(0,100))↵
```

とすればよい。「`xlim=c(0,100)`」という引数が横の目盛りを指定した部分である。もちろん、0 が下限、100 が上限である。なお、引数が複数あるときは、それらを `,`（カンマ）で区切ること。

次に、図を並べる方法だが、R では図を 1 画面に「縦に n 個」「横に m 個」格子状に並べることができる。たとえば、「縦に 2 個」「横に 1 個」の図を並べるためには、次のどちらかを行う。前者が図を横に並べていき、後者は図を縦に並べていく。

```
> par(mfrow=c(2,1))↵
> par(mfcol=c(2,1))↵
```

問 1 もとに戻したければ「縦 1 個」「横 1 個」にすればよい。その方法を 2 通り考えよ。

COMPUTER OPERATION ▷
```
>
>
```

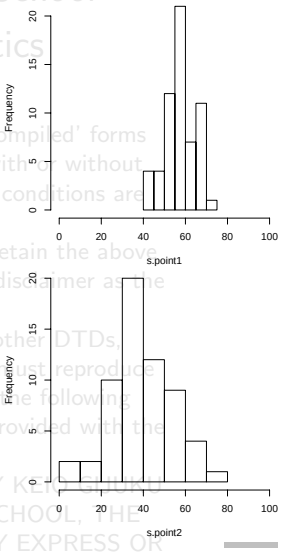
— N O T E —

問 2 画面を「縦に 2 個」「横に 1 個」図を並べる状態にしてから、s.point1 と s.point2 のヒストグラムを並べて表示せよ。横軸の範囲は 0 から 100 とする。

COMPUTER OPERATION ▷

```
>
>
>
```


1.6.2 データの比較の基準となる値



このようにしてデータを眺めるときは、少なくとも

- どこらへんのデータが集中しているか
- データが広範囲にあるのか、狭い範囲にあるのか

ということに注目するとよい。s.point1 と s.point2 という 2 つのデータでも、差があるはずである。

問 3 以上の点について、この 2 つのデータから読み取れることは何か。

データを眺める際に、ヒストグラムで与えられる上記の事柄を、客観的な数値として与えることができれば、2 つのデータの比較を、数値の比較という判定のしやすい方法で行うことができる。そこで、データの特性を

- データを代表する値 · · · **代表値**
- データの広がりを表す値 · · · **散布度**

という観点から、少数の数値で表現することを考えていくことにする。

`hist(x, xlim=c(a,b))` ベクトルオブジェクト x のヒストグラムを描く。xlim=c(m,n) を同時に与えると、横軸の最小値を m, 最大値を n と仮定して描いてくれる（指定しなければ勝手に決められる）。

`par(mfrow=c(m,n))` `par(mfcol=c(m,n))` 一つの画面で図を縦 m 個、横 n 個並べるようにする。前者は図を横に並べ、後者は図を縦に並べる。

— N O T E —

1.7 データの代表値

データを代表させる値として、よく知られているのは「平均値」だろう。ただ、その性質については考えたことがないかもしれない。ここでは、2 種類の代表値の定義を考える。

1.7.1 平均値の定義

データの代表値として一番有名なものに、**平均値**がある。N 個のデータ x_1、x_2、…、x_N があったとき、

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{x_1 + x_2 + \cdots + x_N}{N}$$

が平均値として一番使われるものである。「データを全体に平らにならしたときの値」ということで、非常にわかりやすいものである。

ベクトルオブジェクト `s.point1` の平均値を求めてみよう。定義から、このデータをすべて加えることができなければならないが、そのためにベクトルの要素のすべての値の和を求める関数 `sum()` という関数がある。

COMPUTER OPERATION ▷
```
> sum(s.point1)↵
```

また、ベクトルの要素の数は `length()` で求まる。これを組み合わせれば平均値を求めることができる。

```
> sum(s.point1) / length(s.point1)↵
```

問 1 この方法で、オブジェクト `s.point2` の平均値を求めよ。

COMPUTER OPERATION ▷
```
>
```

しかし、実際には平均値を求める関数 `mean()` が用意されているため、こちらを用いること。専用の関数は注意深く作られているため、目的の用途を持つ関数がある限りそちらを使ったほうがよい。関数 `mean()` を用いてベクトル `s.point1`, `s.point2` の平均値を求めてみよ。

COMPUTER OPERATION ▷
```
> mean(s.point1)↵

>
```

— N O T E —

1.7.2 中央値の定義

平均値は「**異常な値の影響を受けやすい**」という特徴がある。よく「誰が平均点をつりあげてるんだ !?」という話が出ることがあるが、そのような例をあげよう。

問 2 s.point2 に、100 点を取ってしまった者を加えた状態での平均値を求めよ。どのくらい変化があるだろうか。

COMPUTER OPERATION ▷

```
> s2 ← c(s.point2, 100)↵
>
```

このような現象があるため、平均を「代表」とするのは場合によっては抵抗がある。対策として「異常な値を排除する」という方法があるが、たとえばテストの結果で、高得点を取った者に「お前の点数は異常だから除いて考えるぞ」などというわけにもいかない。また、平均値はデータのすべての値を用いて計算するので、その値はデータに存在しない値を示すことがある。実際には存在しない値を「代表」とするのも、場合によっては疑問であろう。

そこで、代表値の選び方として、別の方法を考えてみる。「データの中心」を取り出すというのは、代表値の選び方としては自然である。そこで、「データを小さい順に並び換え、その中央の値を取り出す」というのを「中心を取り出す」方法の一つと考え、このようにして定めた代表値を**中央値（メディアン、median）**という。偶数個のデータの場合、中央値は中央の 2 つの値の平均で定義する。

中央値は median() という関数で計算できる。

COMPUTER OPERATION ▷

```
> median(s.point1)↵
```

問 3 s.point2 と、s.point2 に 100 点の者を加えたものの、それぞれの中央値を求めよ。違いはあるだろうか。

COMPUTER OPERATION ▷

```
>


>
```

sum(x)	ベクトルオブジェクト x のすべての要素の和を求める。
mean(x)	ベクトルオブジェクト x の平均値を求める。
median(x)	ベクトルオブジェクト x の中央値を求める。

— N O T E —

1.8 平均値・中央値の性質

平均値と中央値は求め方が違うため、その性質は違うと考えるのは自然である。ここでは、それらの特徴についてさらに考察する。

1.8.1 平均値・中央値の挙動の違い

平均値は「**データの重心である**」という特徴がある。つまり、全体で見れば「データは平均値のあたりに集中している」のである。データをおもりをつり下げる位置と考えると、平均値はその支点である。このことを視覚的に表現するために、`balance()`という関数を用意した。この関数で、平均値とデータの関係をながめてみよう。

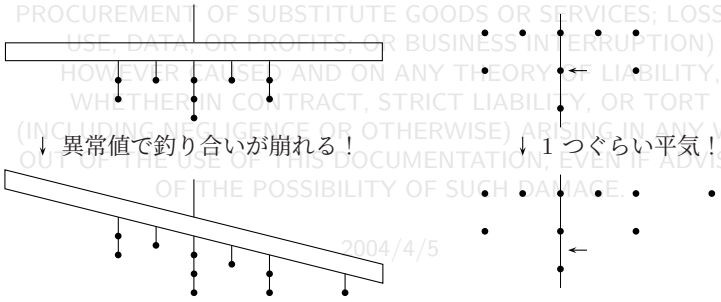
COMPUTER OPERATION ▷
```
> balance(s.point1)↵
```

表示されている値が平均値であるが、このような値を取るデータは `s.point1` には存在しない。このように、データの重心（ある意味では中心に近い値）であるが、現実の値ではないことに注意しなければならない。

「重心」ということは、支点から遠いデータはそれだけ影響が大きい。つまり、釣り合いがとれている状態に異常値が加わると、それだけで釣り合いが崩れてしまうのである。これを防ぐには、支点を動かさなければならない。もちろん、もともとの支点から遠い異常値ほどその影響が大きい。平均値が異常値に敏感に反応するのは、このような理由もある。

一方、中央値はデータの個数での中央だから、中央値の上と下のデータの個数がほぼ等しい。この図の縦線の部分が中央値だが、この線を境に左右の点の数はほぼ等しいはずである。異常値が 1 つ加わっても、変化しないことが多く、変化するにしても、順に並べたものの中で 1 つずれるだけだから、どんな異常値でも値の変動は一定である。

こうみると中央値はいいことづくめのように見えるが、平均値のほうが数学的な扱いは簡単であり、また中央値は求めるときに並び換えの手間が発生する。



— N O T E —

問 1 s.point2 と、s.point2 に 100 点を取った者を加えたものについて、その様子を観察せよ。balance() 関数の引数として xlim=c(0,100) を加えると、幅を固定するので比較しやすい。

COMPUTER OPERATION ▷

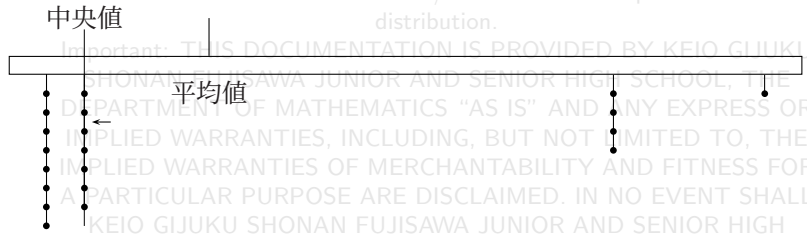
```
> par(mfrow=c(2,1))↵
>
>
```


1.8.2 ヒストグラムの対称性と平均値・中央値

ヒストグラムの形と平均値・中央値の差には関係がある。もし、ヒストグラムの形が左右に対称ならば、中央値は当然対称軸のあたりにある。平均は、データの重心であり、釣り合いがとれる部分はやはり対称軸のあたりである。

ところが、たとえば小さい値が多いようなデータの場合、次のようなことがおこる。

- 中央値は、データの個数が多い小さい値に近づく
- 平均値は、大きい値が釣り合いの上で大きく働くので、中央値よりも大きな値になる



問 2 大きい値が多いデータの場合、中央値と平均値の関係はどうなると考えられるか。

大きい値が多いデータの場合、中央値と平均値の関係はどうなると考えられるか。

このように、代表値はそれぞれ特性があるため、どれがよいかは一概には言えない。

balance(x, xlim=c(m,n))† ベクトルオブジェクト x のデータを用いて、天秤状に図示する。xlim=c(m,n) を同時に与えると、横幅の最小値を m、最大値を n として描く。

※ † がついている関数は、実習のために用意したもので、標準の R にはない。

— N O T E —

1.9 データの散らばりと四分位数

データを 1 つの値で代表させることは、データの集中している部分の情報を与えることになるが、全体的な様子はわからない。そのため、全体的な様子—データの散らばり—を与えるものを用意すると便利であろう。

1.9.1 データの散らばりとは

もう一度、s.point1 と s.point2 のヒストグラムをながめてみよう。このとき、次のことに注目して、比較した結果を以下の表に書き込んでみよ。

COMPUTER OPERATION ▷

```
> par(mfrow=c(2,1))↵
> hist(s.point1, xlim=c(0,100))↵
> hist(s.point2, xlim=c(0,100))↵
```

	s.point1	s.point2
山の頂上の位置		
山の存在する幅		
山の形		

ヒストグラムを眺めるとわかるが、一つの山ではあるが、その形は異なる。この違いは「データの散らばりかた」であり、そのような状態を表現する値があれば、その様子を知ることができる。

散らばり具合を示すいちばん簡単な方法は、データの最大値と最小値（およびその差）を使うことである。最小値・最大値はそれぞれ関数 min(), max() で求めることができる。

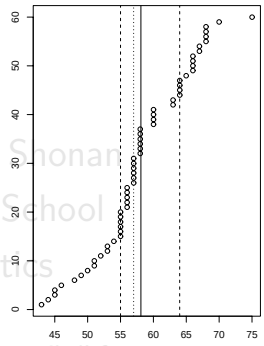
COMPUTER OPERATION ▷

```
> min(s.point1)↵

> max(s.point1)↵
```

しかし、これは異常値に影響されやすい。異常値はデータの端にあることが多い。そのような場合にも対応できる方法はないだろうか。

— N O T E —

1.9.2 累積度数グラフと四分位数



データの存在の状態を図示する方法として、右の図のようなグラフがある。この図は横軸がデータの値、縦軸がデータの相対的な位置を表している。一番上が最後のデータを表し、縦方向の値が、データを小さい順に並べたときにどのあたりにあるかを表す。このような図を**累積度数グラフ**という。

この図は cumcurve() という関数で描くことができる。

COMPUTER OPERATION ▷

```
> cumcurve(s.point1)↵
```

特徴的な値に点線を加えておいたが、この図の点線は、左から順に、データを小さい順に並べたときの下から $\frac{1}{4}$ の値、真ん中の値、上から $\frac{1}{4}$ の値を示している。

真ん中の値は中央値であり、残り 2 つは順に第 1（下側）四分位数、第 3（上側）四分位数という（「第 2 四分位数」とは何だろうか？）。なお、実線は平均値である。

問 1 第 1 四分位数と第 3 四分位数の間には、データ全体に対してどのくらいの割合のデータが含まれているか？

問 2 この累積度数グラフにおいて、点を結んだ線分の傾きの大・小は、何を表すかを考えてみよ。

傾きが大きいと、それだけ四分位数の間の幅が狭くなる。つまり、それだけデータが集中しているということになる。逆に、傾きが小さいと、それだけ四分位数の間の幅が広くなるから、その分集中する度合いは緩やかになる。

`min(x)` `max(x)` ベクトルオブジェクト x の最小値・最大値を求める。

`cumcurve(x, xlim=c(m,n))`† ベクトルオブジェクト x の累積度数グラフを描く。`xlim=c(m,n)` を同時に与えると、最小が m、最大が n となるように横軸を定める。

※ † がついている関数は、実習のために用意したもので、標準の R にはない。

— N O T E —

1.10 箱ひげ図とデータの散らばり方

累積度数グラフと四分位数の値から、データの様子をある程度知ることができた。すると、もう少し簡単な図でデータの様子を表現することはできないだろうか。

1.10.1 箱ひげ図

四分位数・中央値の間の幅が小さいということは、データが集中している。また、四分位数・中央値の間の幅が大きいいうことは、それだけデータが広がって存在しているということになる。

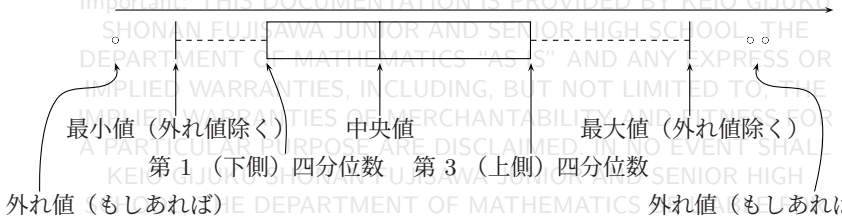
ということは、四分位数の位置だけを図示することにより、ある程度の判断ができるのではないだろうか。その位置だけを示した図を描けば、データの様子がわかることになる。そのための図式として**箱ひげ図（box-and-whisker plots）**がある。関数 boxplot() によって描くことができる。

COMPUTER OPERATION ▷
```
> boxplot(s.point1)↵
```

このままでは縦置きだが、次のように horizontal=TRUE という引数を与えれば横置きにできる（TRUE の意味は後述）。また、データの幅を示したいときは、ylim という引数をつければよい。

```
> boxplot(s.point1, ylim=c(40,80), horizontal=TRUE)↵
```

ここで描かれる図は、次のような構造をしている。



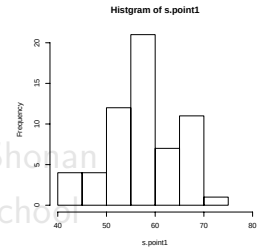
累積度数グラフ・箱ひげ図・ヒストグラムを並べて描いてみよう。これらの間の関係を、ながめてもらいたい。

COMPUTER OPERATION ▷
```
> par(mfrow=c(3,1))↵
> hist(s.point1, xlim=c(40,80))↵
> boxplot(s.point1, ylim=c(40,80), horizontal=TRUE)↵
> cumcurve(s.point1, xlim=c(40,80))↵
```

このように並べてみると、累積度数グラフの中央値・四分位数の値がそのまま箱ひげ図の箱に対応していること、そして、箱の大きさとデータの集中のしかた、そしてそれがヒストグラムに対応していることがわかる。

— N O T E —

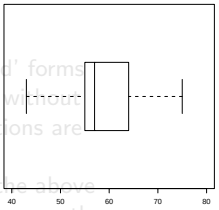
問 1 s.point2 で、同じような図を描いてみよ。表示範囲として、c(0,80) を与えるとよい。xlim と ylim の違いに気をつけること。



COMPUTER OPERATION ▷

```
>
>
>
```

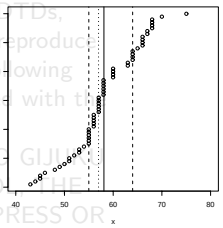
さて、s.point1 と s.point2 の間には、データの散らばり方の違いがあるが、それを比較するにはどうすればよいだろうか。単純に累積度数グラフを並べても違いはわかる。



COMPUTER OPERATION ▷

```
> par(mfrow=c(2,1))↵
> cumcurve(s.point1, xlim=c(0,80))↵
> cumcurve(s.point2, xlim=c(0,80))↵
```

しかし、これを見れば、散らばり方は四分位数の値である程度見当がつくので、箱ひげ図だけを並べれば比較ができるということがわかるだろう。



箱ひげ図で比較する場合には、boxplot() 関数の引数に、並べたいオブジェクトを書けばよい。引数を複数与えると、それに応じて箱ひげ図を並べてくれる。

```
> par(mfrow=c(1,1))↵
> boxplot(s.point1, s.point2)↵
```

前回の例にならって、s.point1 について、そのままの場合と、100 点と 0 点の生徒がいた場合の状態も観察してみよ。四分位数が異常な値に強いということが、この図でも観察できる。

```
> boxplot(s.point1, c(s.point1, 100, 0))↵
```

boxplot(..., ylim=c(m,n))	引数として与えたベクトルオブジェクトの箱ひげ図を描く。複数並べれば、その分だけ描く。ylim=c(m,n) を同時に与えると、最小が m、最大が n となるように軸を定める。

※ † がついている関数は、実習のために用意したもので、標準の R にはない。

— N O T E —

1.11 分散

四分位数は、それなりに散らばりの様子を示してくれるが、すべてのデータを用いて計算した値ではないため、まったく違う様子を持つデータで同じ値を示してしまうことがある。そのため、「すべてのデータの値を用いた散らばりを表す値」を定義することも重要である。

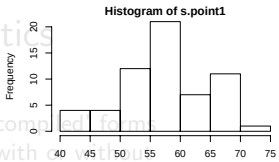
1.11.1 四分位数の欠点

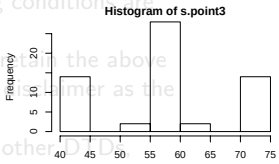
s.point3 というオブジェクトの平均・中央値・四分位数は、s.point1 とまったく等しい。箱ひげ図を描いても全く同じものが得られる。

```
> boxplot(s.point1, s.point3)↵
```

ところが、ヒストグラムを描くと全く様相が異なる。

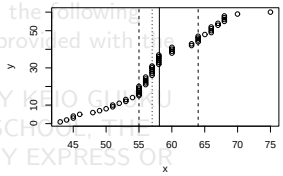
```
> par(mfrow=c(2,1))↵
> hist(s.point1)↵
> hist(s.point3)↵
```

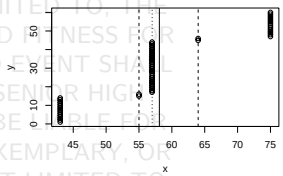




cumcurve() 関数を用いればその理由がわかる。

```
> cumcurve(s.point1)↵
> cumcurve(s.point3)↵
```

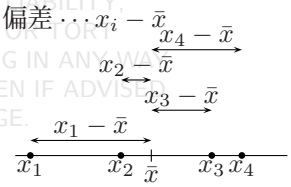




散らばりかたは全然違うが、四分位点の位置がたまたま一致していたので、それが検出できなかったのである。一部の値だけで散らばりの様子を示したため起こったことである。四分位数は、ちょうど平均に対する中央値のようなものであり、散らばりの具合を平均のようにすべてのデータの情報を用い計算したものもあるほうがよい。

1.11.2 分散

データ全体の値を使って散らばりの具合を示すために、データのそれぞれの「平均からの距離」を考えることにする。各データと平均との距離（正負を考える）を、そのデータの**偏差（deviation）**という。そして、それらの和を散らばり具合を示すものにできないかと考えるのはごく自然である。しかし、N 個の各データを x_i、平均を $\bar{x}$ とすると、偏差は $x_i - \bar{x}$ であるが、この和を求めても意味がない。



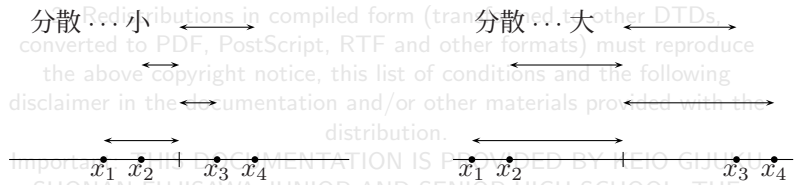
— N O T E —

問 1 この理由を、$(x_1 - \bar{x}) + \cdots + (x_N - \bar{x})$ を求めることにより示せ。

符号を打ち消せばよいわけなので、偏差の絶対値の和でもよいのだが、絶対値は数学的な扱いがやっかいなので、絶対値のかわりに 2 乗した

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{(x_1 - \bar{x})^2 + \cdots + (x_N - \bar{x})^2}{N-1}$$

を用いることが多い。これを、**（標本）分散（(sample) variance）**[*1]という。



問 2 なぜ和を求めたあとで N に関係する値（実際には $N-1$）で割っているか、理由を考えよ。実例「1, 3, 5」と「2, 2, 2, 2, 3, 4, 4, 4, 4」で考えてみるとよい。

また、2 乗したままだと単位がかわってしまう。たとえば、元のデータの単位が cm であったとき、分散の計算の中に長さを 2 乗する作業が含まれているため、このときの分散の単位は cm^2 である。単位をデータのものとそろえるため、分散 s^2 の平方根 s も用いる。これを、**（標本）標準偏差（(sample) standard deviation）**という。

[*1] $N-1$ のかわりに N で割るものも存在します。これを母分散といいますが、標本分散と母分散の違いは難しいので、ここでは分散は標本分散に統一することにします。

— N O T E —

1.12 分散の計算方法とベクトルの計算

前節の結果を用いて、分散を計算してみる。その際、ベクトルに対する演算が便利なことがあるため、その方法についてもふれることにしよう。

1.12.1 ベクトルの計算

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{(x_1 - \bar{x})^2 + \cdots + (x_N - \bar{x})^2}{N-1}$$

が標本分散の計算方法である。この計算を行うためには、

1. データの平均を求める
2. 各データから平均を引く
3. その結果のそれぞれを二乗する。
4. その結果の和を求める。
5. その和を「データの個数 − 1」で割る

という作業が必要となる。`s.point1` の平均は `mean(s.point1)` だし、和を求めるのは `sum()` 関数を用い、データの個数は `length()` 関数を用いればよい。

残りは「各データから平均を引く」「それぞれの値を二乗する」という作業である。R においては、ベクトルのそれぞれの要素から、ある値を引くという作業に対応する。そのようなことをするためには、単純に

COMPUTER OPERATION ▷
```
> A1 ← s.point1 - mean(s.point1)↵
```

とすればよい。ベクトルと通常の数との演算は、**ベクトルの各要素とその数との演算**になる。通常のプログラミング言語では、繰り返しをする機能を用いて各要素の計算が必要となるが、R ではその必要はない。

問 1 次の作業を行う R の式を完成させよ。

(1) 上の `A1` を 2 乗したものを `A2` とする。

```
>
> A2↵
```

(2) その和を求め、ベクトルの長さから 1 を引いたもので割る。これが分散となる。

```
>
> VA↵
```

— N O T E —

1.12.2 式の組み立てと分散を求める関数

以上の操作は、計算のそれぞれの部分を分けて計算しているが、ほとんどの値は分散が求まった後では不要になる。「途中経過」を残さないような式を作ってみよう。A1 と A2 は

```
A1 ← [s.point1 - mean(s.point1)]
A2 ← [A1]^2
```

というように作ったものだから、「A2 の部分に A1 の作成した式をそのまま書けばよい。つまり、次のようになる。

COMPUTER OPERATION ▷

```
> A2 ← (s.point1 - mean(s.point1))^2↵
```

問 2 VA を求める式を同じように考え、s.point1 と関数 mean()、length() のみを使った式で答えよ。

```
A1 ← [s.point1 - mean(s.point1)]
A2 ← ([          ])^2
VA ← sum([          ])
        /(length([     ])-1)
```

COMPUTER OPERATION ▷

```
>
```

実は、分散を求める関数 var() があるので、通常はこちらを利用すること。sqrt() 関数と組み合わせれば、標準偏差を求めることもできる。

問 3 var()、sqrt() を用いて、s.point1 の分散と、標準偏差をそれぞれ求めよ。

COMPUTER OPERATION ▷

```
>

>
```

var(x)	ベクトルオブジェクト x の標本分散を求める。

— N O T E —

1.13 データの分布

いままでの知識を用いて、データのおおまかな形をつかむことをしてみよう。

1.13.1 データの分布の形をつかむ

データがどのように存在しているか、ということを表す言葉として「**分布**」という言葉がある。「このデータの分布は広い」などというように、状況を表す言葉とともに用いる。分布の状態を図示したもので、いちばんわかりやすいのはヒストグラムである。分布の状態を、このヒストグラムの形で表現することも多い。

そこで、データの分布を、いままでの知識で調べることにしよう。特に、箱ひげ図・ヒストグラムと、平均・標準偏差の関係を見ることにする。

データを調べるときに、平均値や標準偏差といった代表値とともに、ヒストグラムや箱ひげ図などで、データ全体を眺め、その分布の概略をつかむことが大切である。確実なことを知るためには、ヒストグラム等を描くとよいのだが、代表値だけで見当をつけるときは、次のようなことに注目するとよいことがある。

- 山の形について
  以下に述べることは、すべて山が 1 つのものについてである。このような形を単峰形という。山が 2 つ以上あるものについては適用できないことが多い。
- すその広さについて
  分散の大・小や、四分位数の幅で判断する。分散が大きいほど、四分位数の幅が大きいほどすそ野が広い。
- 山の対称性について
  左右対称かどうかは、箱ひげ図である程度判断できる。中央値からの上下の幅に、著しく差がある場合、非対称なことが多い。
- 山の頂上と平均について
  原則として、平均値の近くに山がある。左右非対称だと位置が多少ずれる。

問 1 次に示した平均と標準偏差、および箱ひげ図は、`s.point1`、`s.point2`、`s.point4`、`s.point5` の 4 つのデータに対するものである。対応をつけよ。

なお、ヒストグラムは次のようにすれば描くことができる。

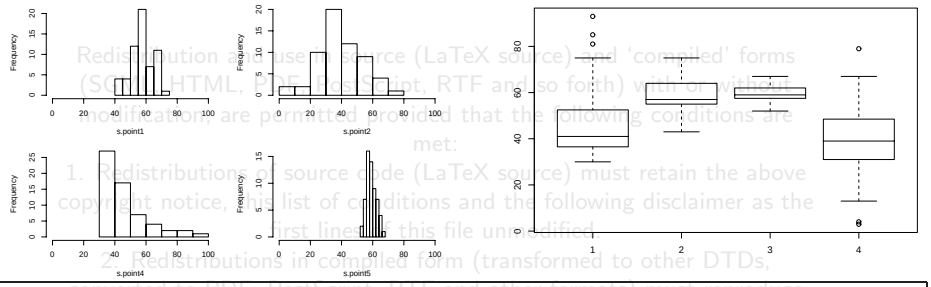
COMPUTER OPERATION ▷

```
> par(mfrow=c(2,2))↵
```

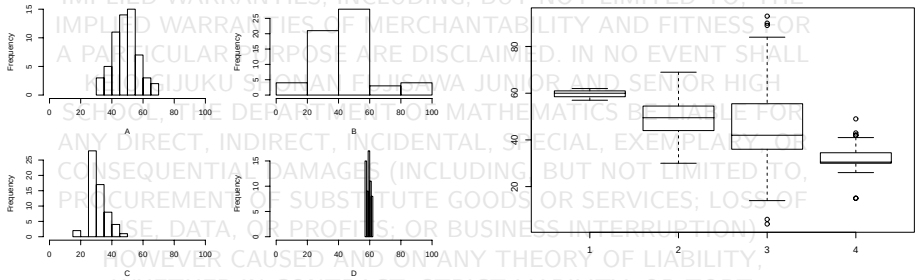
— NOTE —

```
> hist(s.point1, xlim=c(0,100))↵
> hist(s.point2, xlim=c(0,100))↵
> hist(s.point4, xlim=c(0,100))↵
> hist(s.point5, xlim=c(0,100))↵
```


	(A)	(B)	(C)	(D)
平均	46.0333	39.5667	59.51667	58.1
標準偏差	14.230508	14.863636	3.191576	7.055951



問 2 以下の図のヒストグラムと箱ひげ図の対応をつけよ。



— N O T E —

1.14 論理数

計算機で扱えるものの中で、一番簡単なものは「2 つの値のみを持つ値」である。本来電気で動いている電子計算機であり、その状態は本質的には電気の ON・OFF だけである。したがって、そのような状態を表すデータというのも、本質的である。

1.14.1 論理数とは

height は 120 人の身長のデータ（単位 cm）なのだが、このデータには男子と女子のものが混ざっている。男子と女子で、身長の分布が異なることは明らかだから、このデータを男女のものに分離したい。実際、ヒストグラムを描けば、あまりよい分布をしていないことがわかる。

COMPUTER OPERATION ▷
```
> par(mfrow=c(1,1))↵
> hist(height)↵
```

height.male というベクトルには、height のベクトルの各要素（身長）が男子のものかそうでないか（ここでは女子である）が格納されている。この中身は次のようになっている。

COMPUTER OPERATION ▷
```
> height.male↵
[1] TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE ···
```

この TRUE と FALSE は、数値とは異なるもので、「はい・いいえ」や「ほんと・うそ」「有・無」というような 2 つの状態を示す場合に使われる。これを**論理数**という。このデータでは、男子である場合に TRUE、女子である場合に FALSE である。

なお、入力するときは、便宜をはかるため、TRUE は T、FALSE は F というように入力できる。ただし、そのように入力しても、勝手に長い文字列に置き換えられる。

1.14.2 論理数の演算

論理数は通常の数ではないが、計算をするときには TRUE が 1、FALSE が 0 となっている。したがって、

COMPUTER OPERATION ▷
```
> x ← c(TRUE, FALSE, TRUE, TRUE ,FALSE)↵
> sum(x)↵
```

とすれば、これは次と同じである。

```
> y ← c(1,0,1,1,0)↵
> sum(y)↵
```

— N O T E —

これを見ればわかるとおり、x の TRUE の数と y の 1 の数は一致する。1 を n 個加えれば n になるから、論理数ベクトルの中の TRUE の数を数えたければ、関数 sum() を用いればよいことがわかる。

問 1 このデータに含まれる男子のデータの数を数えよ。

COMPUTER OPERATION ▷

```
>
```

女子のデータの数を数えるのは、総数がわかれば計算はできるが、それ以外の方法として、「TRUE と FALSE をひっくりかえす」という作業ができればよい。このときには ! を使う。! は + や - などの演算をする記号の 1 つであり、論理数または論理数ベクトルの頭につけると、その T と F をひっくりかえす。

COMPUTER OPERATION ▷

```
> x ← c(TRUE, FALSE, TRUE, TRUE, FALSE)↵
> !x↵

> sum(!height.male)↵
```

論理数は、比較などの作業をすると発生する。通常の数だけでなく、ベクトルに対しても比較はでき、各要素との比較になる。正しければ TRUE、正しくなければ FALSE である。

COMPUTER OPERATION ▷

```
> 3 < 4↵

> s.point1 > 65↵
```

比較のときに用いる記号は次の通り。

a < b	a は b より小さい	a > b	a は b より大きい
a <= b	a は b 以下	a >= b	a は b 以上
a != b	a は b と等しくない	a == b	a は b と等しい

Note: 3 と -4 の比較をするときは、3 <␣-4 と必ず ␣ の部分に空白を入れること。3<-4 ではうまくいかない（なぜか？）。

問 2 height の中で、「身長 175 cm 以上の者」「身長 160 cm 未満の者」の人数を数えるにはどうしたらよいか。

COMPUTER OPERATION ▷

```
>
>
>
```

— N O T E —

1.15 ベクトルの要素抽出（その 1）

あるベクトルから必要な要素を取り出すというのは、データを処理する上では基本的なことである。ここでは、まず基本的な操作について触れよう。

1.15.1 ベクトルの要素抽出の方法

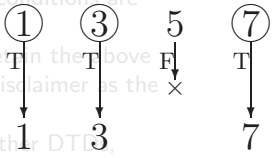
前節の height から、男子だけのデータ、女子だけのデータを取り出してみればきれいな形のヒストグラムが得られるかもしれない。このような場合、条件を満たすベクトルの要素を取り出す（抽出する）ことが必要となる。

◎論理数を用いる方法

ベクトルの要素抽出の 1 つめの方法は、論理数を用いるものである。

COMPUTER OPERATION ▷

```
> A ← c(1, 3, 5, 7)↵
> x ← c(TRUE, TRUE, FALSE, TRUE)↵
> A[x]↵
```



とした場合、[] の中の論理数ベクトルは、1・2・4 番目が TRUE である。このようにした場合、A の TRUE に対応する要素のみが取り出される。これから、男子のデータのみを height.M に付値するには次のようにする。

```
> height.M ← height[height.male]↵
```

問 1 女子のデータを height.F に付値せよ。

COMPUTER OPERATION ▷

```
>
```

問 2 height のデータの中から、身長 180 cm 以上の者のデータを取り出し、height.tall に付値せよ。

COMPUTER OPERATION ▷

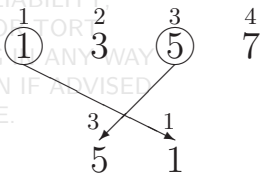
```
>
```

◎数値で場所を指定する方法

ベクトルの要素抽出の 2 つめの方法は、[] の中に数値のベクトルを与えるものである。

COMPUTER OPERATION ▷

```
> y ← c(3,1)↵
> A[y]↵
```



— N O T E —

とすると、A の要素のうち 3 番目と 1 番目の順に取り出される。取り出す要素が 1 つの場合はベクトルにしなくてよい。これから、height から 6, 3, 11 番目のデータを取り出すには、次のようにする。

COMPUTER OPERATION ▷
```
> x ← c(6, 3, 11)↵
> height[x]↵
```

または、直接次のように書いてもよい。慣れたら次のように書いた方がよいだろう。

```
> height[c(6, 3, 11)]↵
```

問 3 height から次に示す場所の要素を取り出す R の式を答えよ。出力結果は書かなくてよい。

(1) 4 番目、9 番目、2 番目（この順に）

COMPUTER OPERATION ▷
```
>
```

(2) 25 番目

```
>
```

(3) 最後

```
>
```

1.15.2 要素の変更

[ ] は要素を取り出すだけでなく、変更することもできる。たとえば、

COMPUTER OPERATION ▷
```
> x ← c(1, 2, 3, 4, 5)↵
```

として作ったベクトルの、2 番目の要素を 0 にするには次のようにする。

```
> x[2] ← 0↵
> x↵
```

問 4 次のようにして作ったベクトル y において、「y の要素のうち 5 以上のものをすべて 0 にする」方法を考えよ。

COMPUTER OPERATION ▷
```
> y ← c(4, 7, 3, 2, -1, 8, 9)↵
>
```


— N O T E —

1.16 ベクトルの要素抽出（その 2）

ここでは、多少高度な要素抽出の方法について考える。特に、規則的に取り出したりする場合、「数列」を作って取り出すことも必要となる。

1.16.1 等差数列を作る演算子

1 番目から 20 番目という場合、c(1,2,...,20) とすべて並べていくのはたいへんだろう。R では、公差が 1 あるいは −1（初項と終項の大小関係で自動的に決まる）である等差数列を発生する、: という記号が用意されている。「初項:終項」の形で与え、結果はベクトルになる。ベクトルの演算機能をうまく使えば、いろいろな数列を作ることができる。

たとえば、初項が 3 で、公差 1 の等差数列の項を 10 まで作るときは、次のようにする。

COMPUTER OPERATION ▷

```
> 3:10↵
```

問 1 初項が 3、公差 2 の等差数列の項を 25 まで作るときはどうすればよいか。次の出力を見て考えよ。

COMPUTER OPERATION ▷

```
> 1:12↵
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
>
```

1.16.2 等差数列を用いた要素抽出

規則的な場所を指定するときは、: をうまくつかうとよい。また、[] の中に負の数を指定すると、その要素を**除いた**要素を返すので、「一部だけ取り除く」というのもできる。

COMPUTER OPERATION ▷

```
> 1:5↵

> -(1:5)↵

> 5:1↵
```

COMPUTER OPERATION ▷

```
> height[1:5]↵

> height[-(1:5)]↵

> height[5:1]↵
```

— NOTE —

問 2 height から次に示す場所の要素を取り出す R の式を答えよ。出力結果は書かなくてよい。

(1) 23 番目から 35 番目

COMPUTER OPERATION ▷
```
>
```

(2) 「13 番目から 21 番目」を除いた残りすべて

```
>
```

(3) 「8 番目から 11 番目」と「19 番目から 31 番目」

```
>
```

(4) 偶数番目の要素をすべて（ヒント：要素の数は全部で 120 個）

```
>
```

(5) 29 番目から 33 番目を番号が減っていく順番で

```
>
```

問 3 height の要素を逆順に並べるにはどうしたらよいか。なるべく一般的に通用する方法を考えよ（実際には、このような操作をする rev() という関数が用意されている）。

COMPUTER OPERATION ▷
```
>
```

1.16.3 層別

これらの操作で、データの分別をすることができるようになる。そこで、男女別に分けた先のデータで、もう一度ヒストグラムを描いてみよう。

COMPUTER OPERATION ▷
```
> par(mfrow=c(1,2))↵
> hist(height.M)↵
> hist(height.F)↵
```

このように、特性が異なると考えられるものをいくつかのグループに分けることにより、それらの特性を正しく反映したものが得られることがある。このような操作を**層別**という。層別により構造が見えやすくなるため、データを調べるときは数値以外のことも注目するとよい。

— N O T E —

第2章　2 組以上のデータの処理と比較

2.1 標準偏差とデータの範囲

分散・標準偏差は、散らばりの度合の比較に用いているが、その値は具体的に何を指しているのだろうか。データの範囲と標準偏差について考える。

2.1.1 チェビシェフの定理

データの分散・標準偏差は、その式を見ればわかるとおり、平均からの散らばりが大きければ大きい値を示す。平均からの散らばりが小さければ小さい値を示す。すると、次のようなことが考えられるだろう。

> 分散・標準偏差が小さいデータは、平均の近くにデータが集中しているはずだ。ならば、分散・標準偏差が小さければ、平均値は大多数のデータに近い値になるから、その意味でデータの代表となっているのではないか。

この考察は正しいのだが、具体的にどのくらいの割合のデータの代表になっているかはこれだけではわからない。しかし、標準偏差の値が得られれば、ある程度の評価ができる。その 1 例が**チェビシェフ（Chebyshev）の定理**である。

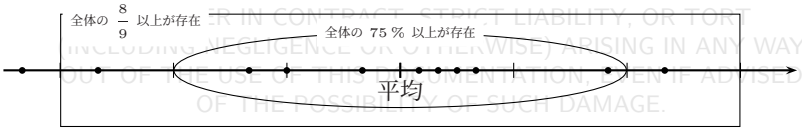
◎チェビシェフの定理

平均が $\bar{x}$、標準偏差が s のデータにおいて、$|x_i - \bar{x}| \leqq ks$ $(k > 1)$ の範囲に含まれるデータの数の全体に対する割合は、$1 - \dfrac{1}{k^2}$ 以上である。

Ex. 標準偏差が 5 のデータでは、平均の上下 10 の範囲にあるデータの、全体に対する割合は、上で $k = 2$ の場合にあたるから、$1 - \dfrac{1}{2^2} = \dfrac{3}{4} = 0.75$ となる。したがって、**少なくとも** 0.75、つまり 75 % のデータが平均の上下 10 の範囲にある。

同じ条件で、平均の上下 15 の範囲にあるデータの、全体に対する割合は、$k = 3$ の場合であるから $1 - \dfrac{1}{3^2} = \dfrac{8}{9}$ となる。

以下の図では、1 目盛りは標準偏差の大きさである。この定理の意味を、以下の図でイメージとしてつかんでほしい。以下、だ円は $k = 2$ の範囲（全体の 75 % 以上のデータがある範囲）、長方形は $k = 3$ の範囲（全体の $\dfrac{8}{9}$ 以上のデータがある範囲）を表すことにする。



チェビシェフの定理は、どんなデータにも当てはまる。そのため、精度としては非常に悪い。たいていは、もう少し多くのデータが平均近くにあることが多い。

— N O T E —

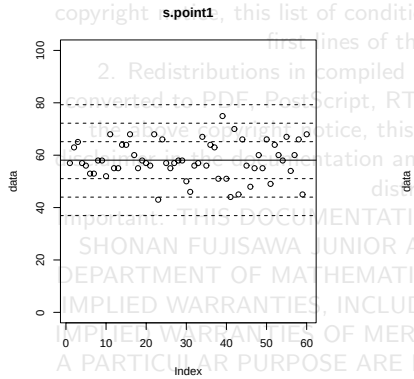
2.1.2 管理図

このような状況をグラフィックスを使って視覚的に確認してみよう。そのためには、横軸に「データの番号」、縦軸に「データの値」を持つような図を描くとよい。これを**管理図**という。管理図は、特に生産現場で品質の管理状況を見るときによく使われる。管理図を描く関数として、ctlchart() を用意しておいた。

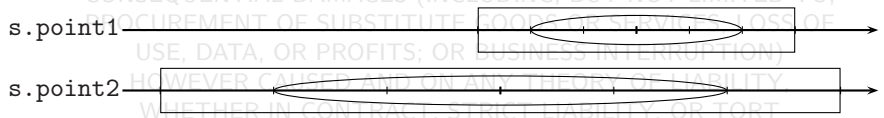
COMPUTER OPERATION ▷
```
> ctlchart(s.point1)↵
```
これで、縦方向を点数とした点をうってくれる。実線は平均、破線は平均から標準偏差の 1・2・3 倍の値だけ離れた位置を指す。

s.point1 と s.point2 を並べてみよう。ylim= を指定すると、縦軸の表示すべき範囲を指定することができる。ここでは ylim=c(0,100) としてみる。

COMPUTER OPERATION ▷
```
> par(mfrow=c(1,2))↵
> ctlchart(s.point1, ylim=c(0,100))↵
> ctlchart(s.point2, ylim=c(0,100))↵
```



これら 2 つの状態を、前ページの図で表すと、次のようになる。管理図は、この図を縦にしたものだと思えばよい。



ctlchart(x, ylim=c(m,n))† ベクトルオブジェクト x の管理図を描く。引数として ylim=c(m,n) を同時に与えると、最小が m、最大が n となるように軸を定める。

※ † がついている関数は、実習のために用意したもので、標準の R にはない。

— N O T E —

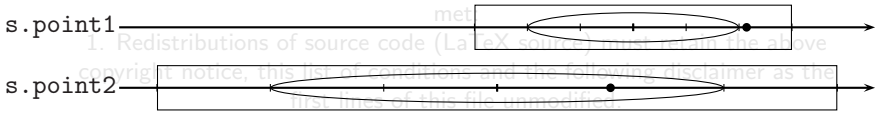
2.2 データの標準化

平均と分散が異なるデータどうしをそのまま比較することはできない。散らばり具合が異なるため、平均との差の価値が異なるからである。前節の管理図の、点線間の幅をそろえるようなことが必要である。

2.2.1 データを比較する際の問題

それぞれのテストにおいて、「平均点より 15 点高い点数を取った」ことにしよう。とりあえず、平均点を超えたことを喜ぶかもしれないが、一体この「15 点」の価値はどのくらいだろうか。

`s.point1` と `s.point2` のそれぞれの場合において、平均点より 15 点高い場所を、前ページの図に黒い点で記入すると、だいたい次のあたりになる。



黒点は、`s.point1` においてはだ円の外側に。一方、`s.point2` においてはだ円の内側にある。

問 1 だ円の外側にあるデータは、全体の何 % 以下か？　また、そのような状態のときには、最悪でも上から何 % の位置にいることがわかるか？

問 2 どちらのほうが評価が高いといえるか？

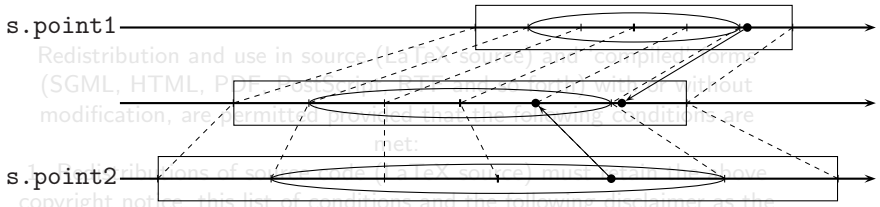
この違いは「標準偏差の大きさ」によるところが大きい。だ円の幅は標準偏差で決まるわけで、「標準偏差が大きい」ほどちらばりが大きいのだから、1 点の価値は薄くなってしまうのである。

— N O T E —

2.2.2 データの標準化

このような状況だと、単純なデータの比較というのはできない。たとえば、成績評価を行いたいとき、テストは科目の違いやそのテストの難易度、さらには受験者によって結果が毎回異なるわけで、比較をする際には工夫が必要なのである。

問題は、この図のだ円や長方形の大きさが異なることなのだから、これらを比較するときには、この大きさをそろえてしまえばよい。つまり、点数の関係をくずさずに「平均」や「標準偏差」をそろえてしまえばよいわけである。イメージとしては、次のようなことを行えばよい。



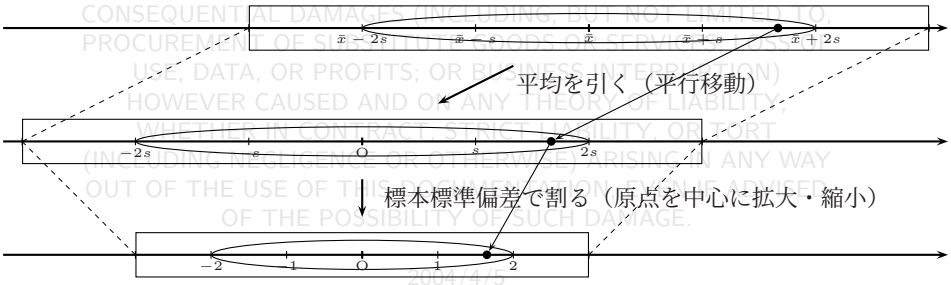
ここでは、データの

- 平均を 0 に、
- 標準偏差を 1 に

することを考えてみよう。そのためには、次のような変換を施せばよい（$\bar{x}$ はデータの平均、s はデータの標本標準偏差）。このような作業をデータの**標準化**という。

$$z_i = \frac{x_i - \bar{x}}{s}.$$

この式は 1 次式の変換であるから、平均（ある種の中心）を 0 にした後に x 軸方向に伸縮させて基準をそろえている。ここでは、数学的な証明はせずに、図による直感的な説明にとどめる。この感覚を身につけてもらいたい。



— N O T E —

2.3 偏差値

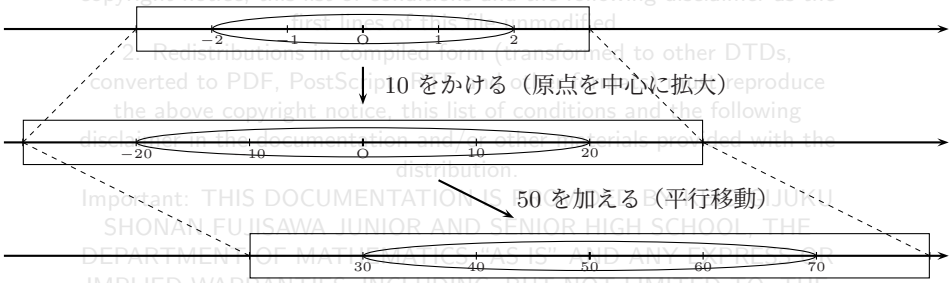
標準化すれば比較ができるとはいっても、小数点以下の値までをしっかり見なければならないため、実用上問題がある。そこで、テストの点に似せて表現したものが偏差値である。

2.3.1 偏差値

データの標準化をすれば、理論的にはデータの比較は可能であるが、実用上は問題があることが多い。テストの結果が -1 だとか、点数の違いが小数点以下で現れると、何かと使いにくいことが多い。

そこで、標準化したデータを 10 倍し、さらに 50 を加える。つまり、$T_i = z_i \times 10 + 50$ という T_i について注目する。

この作業により、平均は 50 に、標準偏差は 10 となるデータとなる。この操作も、図で説明すると次のようになる。



このように変換した値を**偏差値**と呼ぶ。世間で呼ばれる偏差値も、このようにして計算した値である。この値が試験の際に使われる理由は、いままでの説明でわかるだろう。本質的に偏差値というのはデータを標準化したもので、毎回異なる状況で行われる試験を比較するには便利だからである。世間で問題視されているのは、その値の使われ方（この値で受験校を輪切りにして決める、など）であり、計算方法とはまた違う次元の話である。

問 1 先の式を組み合わせて、平均 $\bar{x}$、標本標準偏差 s のテストで、自分の点数が x_i だったときの、自分の偏差値を求める式を作れ。

— N O T E —

2.3.2 標準化の作業・偏差値の求め方

s.point1 を標準化してみよう。まず、s.point1.c に、s.point1 の各データから平均を引いたものを付値する。

COMPUTER OPERATION ▷
```
> s.point1.c ← s.point1 - mean(s.point1)↵
```

s.point1.sd に、s.spoint の標準偏差を付値する。

```
> s.point1.sd ← sqrt(var(s.point1))↵
```

s.point1.z に、s.point1.c の各データを、もとのデータの標準偏差で割った値を付値する。

```
> s.point1.z ← s.point1.c / s.point1.sd ↵
```

問 2 以上の作業を 1 行で書いてみよ。

COMPUTER OPERATION ▷
```
>
```

問 3 s.point1.t に、それぞれ s.point1 に対応する偏差値を付値せよ。s.point1.z を使えばよい。

COMPUTER OPERATION ▷
```
>
>
```

問 4 s.point1 の n 番目の要素は出席番号 n の生徒の点数であるとする。s.point1 における出席番号 1 の生徒の偏差値を求めよ。

COMPUTER OPERATION ▷
```
>
```

◎標準偏差と異常な値について

先にも述べたが、チェビシェフの定理は、データの数の評価としては非常に甘い。実際には、対称な分布を持つデータの場合、次のようなことが言えることが多いのである。

> 安定している状況において、平均から標準偏差の 3 倍以上離れた値のデータが発生することはめったにない。

つまり、ctlchart() 関数で描いた図の、平均値から 3 つ離れた点線外にあるデータは、ほとんど存在しないことが多い。なお、「めったにない」を具体的な数値で示すとなれば、「1％ 未満」であることが多い。標準偏差の 2 倍以上離れた値の割合も「5％ 未満」であり、これも「めったにない」と考えられることがある。逆に考えれば、平均から標準偏差の 3 倍（または 2 倍）はなれたデータは、異常な値として捨ててしまうなり、特別な現象として注目したりと、何らかの処置が必要な場合も多いのである。

— N O T E —

2.4 文字列・データの属性

ベクトルを扱うとき、ある要素がどのような内容であるかを示すために、「名前」をつけることができる。そのとき、名前を表すデータは「数値」でも「論理数」でもない。新しいデータの形が必要となる。また、それを扱う式の表現も必要となる。

2.4.1 文字列

名前などの文字情報を扱うために、S では「文字型」というデータの型が用意されている。文字列は必ず「"」でくくって与える必要がある。

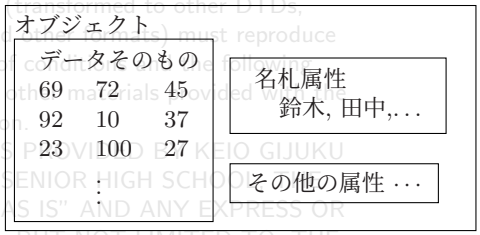
文字列もベクトルにすることができる。

COMPUTER OPERATION ▷

```
> s.subject ← c("kokugo","seibutsu","rekishi")↵
> s.subject↵
```

2.4.2 名札属性

ベクトルを表示させたとき、各行の左端に [1] という表示があって、何番目の要素かを示してくれるが、具体的に何を表すかわかりにくい。



ところで、s.name には、30 人分の（架空の）生徒の氏名が用意されている。

COMPUTER OPERATION ▷

```
> s.name↵
```

生徒の氏名を今まで用いてきたデータに付け加えて扱いやすいものにしてみよう。このためには、「オブジェクトに補助的な情報を加える」という考え方が必要になる。R では、このような情報を**属性（attribute）**という。ここでは「それぞれの要素の名前（名札）を付ける」属性（**名札属性**）を用いる。

— N O T E —

2.4.3 名札属性の用い方と R の関数

ベクトルを作ったばかりのとき、名札属性は「空」である。R では「空」の状態を「NULL」で表現する。

あるオブジェクトの名札属性は、関数 names() で表現できる。s.koku の名札属性は names(s.koku) で表現できるが、空の状況を確認すると、次のようになる。

COMPUTER OPERATION ▷

```
> names(s.koku)↵
NULL
```

このオブジェクトに名札属性を与える。名札として用いるデータは s.name である。ただし、 s.koku は実習用に用意した読みだし専用のオブジェクトであり、書き換えることができない。一旦 s.koku2 に複写して、s.koku2 に対して作業をする。

```
> s.koku2 ← s.koku↵
```

ここで、R の式の構成についてもう少し触れておこう。関数とは、R に働きかけをするものであるが、式中で記述する際には、普通の言葉に置き換えられるように理解していくことが重要である。たとえば、names() 関数の場合は

```
> names ( s.koku2 )↵
  名札属性  何の？  s.koku2 の      は？
```

だから、「names(s.koku2)」で「s.koku2 の名札属性」を表現するのである。名札属性を与える場合にも関数 names() を用いる。「s.koku2 の名札属性に s.name を付値する」と考えるとよい。

```
> names ( s.koku2 ) ← s.name ↵
  名札属性  何の？  s.koku2 の  に付値せよ 何を？  s.name  を
> s.koku2↵

> names(s.koku2)↵
```

名札属性を消したければ、NULL を名札属性に付値すればよい。

```
> names ( s.koku2 ) ← NULL ↵
  名札属性  何の？  s.koku2 の  に付値せよ 何を？  空  を
> s.koku2↵
```

なお、今後のことを考えてもう一度名札属性を付値しておく。

COMPUTER OPERATION ▷

```
> names(s.koku2) ← s.name↵
```

— N O T E —

2.5 名札属性と要素の指定

名札属性を用いると、要素指定が「わかりやすく」なる場合がある。

2.5.1 名札属性を用いた要素抽出

名札属性がついたベクトルでは、名札を用いて要素指定を行うことができる。

s.koku2 の「aikawa さん」のデータを取り出したければ、

COMPUTER OPERATION ▷
```
> s.koku2["aikawa"]↵
```

「sakai さんと hayashi さん」のデータを取り出したければ、次のようにする。

```
> s.koku2[c("sakai","hayashi")]↵
```

問 1 次の手順を行う R の式を答えよ。

(1) s.bio2 に s.bio をそのまま付値し、s.bio2 の名札属性に s.name を付値する。

COMPUTER OPERATION ▷
```
>
>
```

(2) s.bio2 を用いて「narita さんと baba さんの生物の点数」を取り出す。

```
>
```

名札属性がついたベクトルオブジェクトでも、これまでの要素の指定方法は使える。入力量は番号での指定のときより増えてしまうが、番号を捜し出す手間はなくなる。

names(x)	ベクトルオブジェクト x の名札属性を表す。これに付値をすれば、名札属性を変更できる。

◎ R の文法

今後作業が複雑になるので、式を作る規則を理解するのは重要なことになる。一般に、コンピュータに与える指示の集まりを**言語**、その言語の規則を**文法**という。会話で使う「言語」「文法」と意味は似ているが、コンピュータは融通が効かないため、文法から少しでも外れたものを解釈はしない。

— N O T E —

コンピュータ言語を扱うときは**文法をきちんと身につけなければならないのである**。この授業で使う R のことがらで重要なことは

- 基本概念・文法はきちんとおさえる
- R に何かを作業させたいときには関数を使うことを知る
- 関数の具体的な使い方は必要なときに調べられるようにする

ことである。これは多くのコンピュータ言語にいえることなので、この習慣をつけておけばほかの言語を扱うことになってもあまり抵抗はないだろう。たとえば「R でのベクトルとは何か？」ということは知っていなければならないことである。また、関数の名前は調べればよいが、調べる際にはかっこの中に何を並べるかも同時に調べなければならない。たとえば、

- `sqrt()` は平方根を求める関数で、かっこの中には平方根を求めたい数を書く。
- `length()` はベクトルの長さを求める関数で、かっこの中には長さを求めたいベクトルオブジェクトを書く
- `q()` は R を終了させる関数で、R に与える引数は他に必要ない（終わらせるというのに一体他になにが必要なんだ！）ので、かっこの中には何も書かない

ということを調べる（またはまとめておく）必要がある。もちろん、`length()` 関数を使うときに「ベクトルオブジェクトの長さとは何か?」ということは理解している必要がある。さらに、その解釈の仕方も知っておくとよいだろう。たとえば

COMPUTER OPERATION ▷

```
> ctlchart ( s.point1 ) ↵
  管理図描画   何の？  s.point1 の   は？
> a ← 3 + sqrt ( 5 ) ↵
  オブジェクト a に付値せよ 何を？ 3 に加えたもの 何を？ 平方根 何の？ 5 の を
```

ということから、「ベクトルオブジェクト `s.point1` の管理図を描く作業だな」「`a` に $3+\sqrt{5}$ を計算したものを付値しているな」と読むわけである。付値とそれ以外の違いも感じてもらいたい。
このあたりの作業は、数学を理解していく過程と似ているところがあるので、着実に作業をしていかないと、おいていかれてしまう。これはどんな言語を選んでも同じであるから、言語のせいにはしないこと。

— N O T E —

2.6 行列

いままでは、一列に並んだデータのみを扱ってきた。実際には、データはいくつかの数値の組であることが多い。そのような場合、どのように処理をすればよいだろうか。ここでは、このようなデータが与えられたときの操作を解説する。

2.6.1 データの次元と行列

いろいろなデータの観測において、得られる値は 1 つであるとは限らない。たとえば、テストの点数でも、複数回のテストをまとめて扱えば、それだけで数値はたくさん出てくることになる。

一番簡単なのは表にすることであろう。右のようなデータがあったものとする。この場合、生徒 1 人あたりに数値が 3 つ出てくることになる。それぞれの測定対象で得られる値の個数を、そのデータの**次元**という。この例の場合、このデータの「次元は 3 である」「これは 3 次元のデータである」といういいかたをする。いままで扱ってきたベクトルのデータは、1 次元のデータである。

氏名	英語	数学	国語
やぎ	66	85	86
ばば	85	65	68
やまだ	77	58	87
⋮		⋮	
いまはし	77	67	59

多次元（1 次元より大きい次元）のデータを R で扱うためには、**行列（matrix）**というものが必要となる。これは、簡単にいえば表そのものである。横長方向を**行（row）**、縦長方向を**列（column）**という。「行」と「列」の決め方にはだいたいの決まりがあり、通常、測定対象を行単位で並べ、各列はそれぞれの得られた値を表す。たとえば、「生徒の英・数・国テストのデータ」ならば、測定対象は生徒であり、各測定対象は 3 教科の科目の点数を持つため、各行は「各生徒のデータ」、列は「各教科の点数」を示すことになる。

問 1 「日本各所にある 10 か所の気象観測点（地名はすべてわかっているものとする）から得られる気温・気圧のデータ」を行列にしたい。通常行と列にはそれぞれ何を対応させればよいか、適当な例を作って答えよ。また、行と列の数はそれぞれいくつか。

— N O T E —

2.6.2 行列の作成方法

行列の作成方法はいろいろあるが、ここでは 1 種類、行列の成分となるベクトルを用意し、それを行列のマスに流し込んでいくという方法だけを紹介する。

COMPUTER OPERATION ▷

```
> a ← 1:20↵
```

と用意したベクトル a を用いて、5 行 4 列の行列 m を作ってみる。このとき、「流し込む方向を列単位にするか行単位にするか」は問題である。

用いる関数は matrix() だが、列単位で流し込むときには次のようにする。

COMPUTER OPERATION ▷

```
> m ← matrix ( a , 5 , 4 )↵
```

m に付値せよ　何を？ 行列にしたもの　何の？ もとのデータは a　5 行　4 列　を

```
> m↵
```

通常

	1	2	3	4
1				
2				
3				
4				
5				

行単位で流し込むときには次のようにする。もちろん TRUE は T としてもよい。

```
> m.byrow ← matrix ( a , 5 , 4 , byrow = TRUE )↵
```

m.byrow　に付値せよ　何を？ 行列にしたもの　何の？ もとのデータは a　5 行　4 列　行単位で流し込む？　はい　を

```
> m.byrow↵
```

byrow=TRUE

	1	2	3	4
1				
2				
3				
4				
5				

問 2 上の例で byrow=FALSE とするとどうなると考えられるか？

問 3 s.koku, s.bio, s.hist はいずれも長さが 30 のベクトルである。これから、各行がそれぞれの生徒、各列がそれぞれ国語・生物・歴史の点数になっている行列 s.matrix3 を作成せよ。まず 3 つのベクトルを結合する必要がある。

COMPUTER OPERATION ▷

```
> a ←
> s.matrix3 ← matrix(a, 30, 3)↵
```

matrix(x, m, n, byrow=TRUE)　ベクトル x から m 行 n 列行列を作成する。
byrow=TRUE をつけなければ列単位、つければ行単位に要素を流し込む。

— N O T E —

2.7 行列の操作

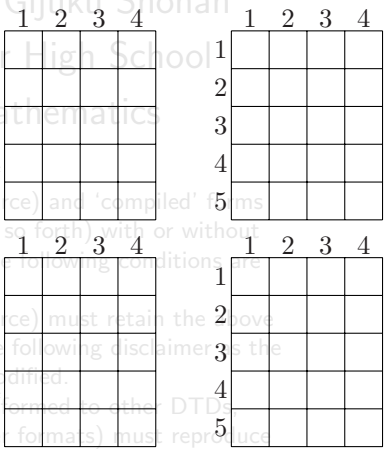
ベクトルと同じように、行列も特定の位置を指定したり、名札をつけたりすることができる。文法は似ているので、以前の知識を応用させるとよいだろう。

2.7.1 行列の要素指定

ベクトルと同じように、行列の一部の要素を取り出すことができる。前節の m の 3 行 2 列の位置の値を取り出すには、

COMPUTER OPERATION ▷

```
> m↵
   （値を右のますに書くとよい）
> m[3,2]↵
```

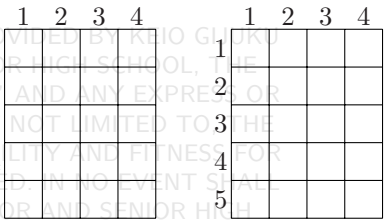


複数の要素を指定したいときは、それぞれの要素を示す部分をベクトルにすればよい。取り出した結果が 1 行または 1 列になるときは自動的にベクトルになる。

問 1 次のそれぞれの式は、どの位置を指すか。

COMPUTER OPERATION ▷

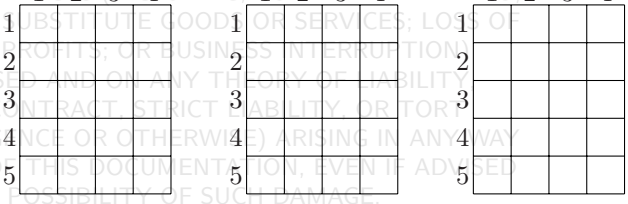
```
> m[c(2,4),1]↵
> m[c(2,4),c(1,3)]↵
> m[1:4,2:3]↵
```



「特定の行（または列）**すべて**」という場合、行番号・列番号指定を省略すればよい。

COMPUTER OPERATION ▷

```
> m[1,]↵
> m[,2]↵
> m[1:3,]↵
```



Note: `m[1,]` と `m[,1]` と `m[1]` は**全く異なる**ものである。混同して用いると、思いもよらない結果が返されることがある。

— N O T E —

2.7.2 軸名札属性を用いた要素の抽出

行列にも名札属性のような属性を与えることができるが、行列は「行」と「列」を指定するため、ベクトルのような「個々の要素に名前を与える」のではない。そのため、名札属性とは別の属性、**軸名札属性**が用意されている。rownames(), colnames() で、行列のそれぞれの行・列についている名札が、表現できる。

s.name に 30 人の名前があるので、s.matrix3 の行に名札をつけよう。

COMPUTER OPERATION ▷

```
> rownames ( s.matrix3 ) ← s.name ↵
```

（行の名札　何の？　s.matrix3 の　に付値せよ　何を？　s.name　を）

問 2 同様にして、s.matrix3 の列に科目名の名札をつけよ。s.subject というオブジェクトがすでにあるはずである。

COMPUTER OPERATION ▷

```
>
```

s.matrix3 を表示させて、表示の変化を見てみよう。

COMPUTER OPERATION ▷

```
> s.matrix3↵
```

使い方は名札属性と似ている。軸名札属性がある行列では、要素を抽出するのに行番号・列番号を使うかわりに、名札そのものを指定してよい。

s.matrix3 において、これから、「sakai」さんのデータ（行）を指定するときは、

COMPUTER OPERATION ▷

```
> s.matrix3["sakai",]↵
```

としてよい。複数の場合は、やはりベクトルにする。

```
> s.matrix3[c("sakai", "tsukahara"),]↵
```

```
> s.matrix3[,"rekishi"]↵
```

rownames(x)　colnames(x)	行列 x の行・列の名札を表す。付値も可能。

— N O T E —

2.8 相関

多次元のデータを扱うとき、それぞれの値どうしが何らかの関係を持つ場合がある。このような関係の調べ方を、一番簡単な 2 次元のデータをもとに、いろいろな場合で考えてみる。

2.8.1 散布図

2 次元のデータは、それぞれが 2 つの値の組であるから、最初の値を x 座標、後の値を y 座標とすれば、座標平面上の点をみることができる。そのようにして実際にデータを点として描いた図を、**散布図（scattergram）**という。散布図は、データを眺める方法としては非常によく使われるので、とりあえずデータがあったら散布図を描いてみるとよいだろう。

R では、`plot()` 関数で散布図を描くことができる。引数として、ベクトルを 2 つ、あるいは 2 列の行列を 1 つ与える。なお、x 軸・y 軸の名札は、オブジェクト名や軸名札属性などから、適当につけられる。

`s.matrix3` は前節の行列ベクトルとする。次のような散布図を描いてみよう。

COMPUTER OPERATION ▷

```
> plot ( s.point1 , s.point2 )↵
```
散布図描画 何の？ s.point1 と s.point2 の は？

```
> plot ( s.matrix3[,2:3] )↵
```
散布図描画 何の？ s.matrix3 の第 2 列（生物）と第 3 列（歴史）を使って は？

問 1 `s.matrix3` を用いて、国語と歴史の点数の散布図を描け。x 軸・y 軸の順番は問わない。2 通り考えてみよ。

COMPUTER OPERATION ▷

```
>
>
```

2.8.2 相関関係とは

これらの散布図はそれぞれ特徴があり、

A. 一方が増えれば他方も直線的に増える傾向がある関係

B. 一方が増えれば他方は直線的に減る傾向がある関係

C. 全く傾向に脈絡がない

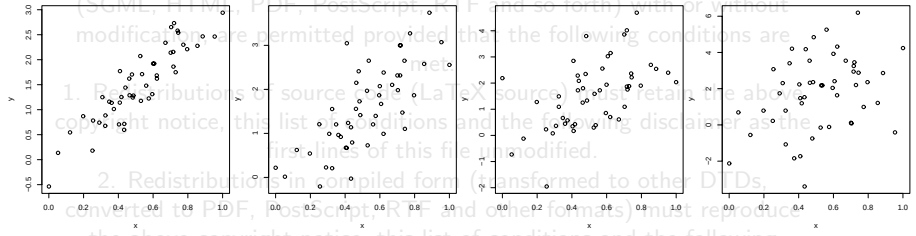
という 3 種類がある。

— N O T E —

問 2 A, B, C のどれがどの散布図に対応するか。

A を「**正の相関がある**」、B を「**負の相関がある**」、C を「**相関がない（無相関）**」という。また、相関関係は「傾向」を表すものなので、「強い」「弱い」でその状況を表すことがある。ただし、**統計・データ解析において、「相関する」というサ行変格活用の動詞は通常用いない。**

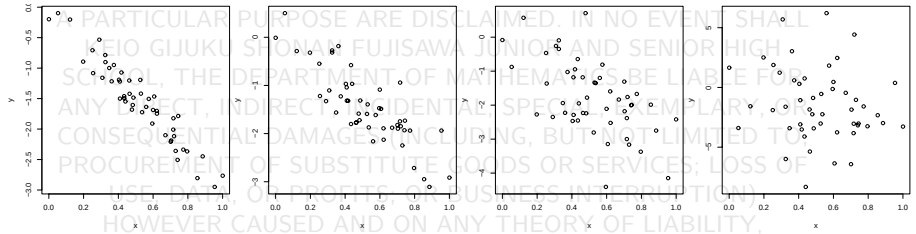
いろいろな状況の散布図の例を以下にあげる。その雰囲気をつかんでもらいたい。



強い ← 正の相関が → 弱い

無相関

強い ← 負の相関が → 弱い



`plot(...)` 散布図を描く。引数として、「2 つのベクトルオブジェクト」または「2 列の行列オブジェクト」を与える。それぞれ最初のデータが x 軸、次のデータが y 軸になる。

— N O T E —

2.9 相関係数

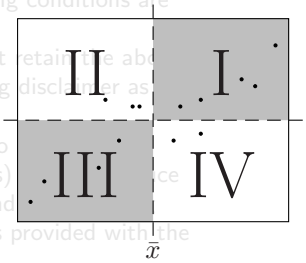
散布図で相関関係が確認できても、客観的な相関の強さを表す数値がないと、比較のとき困る場合がある。ここでは、相関関係の強さを表す数値を定義する。

2.9.1 ピアソンの積率相関係数

相関関係の強さを表す数値を**相関係数**という。一番よく使われる**ピアソンの積率相関係数**をここでは定義として用いよう。

2 次元のデータ $(x_i, y_i)(i = 1, 2, \ldots, n)$ の、x_i と y_i の平均をそれぞれ $\bar{x}, \bar{y}$ とする。これを、データのある種の原点と考えることにする。右の図の $*$ は、それぞれの点と考えるとよい。ここで、x_i と y_i の偏差積 $(x_i - \bar{x})(y_i - \bar{y})$ を考えると、次のようになる。

	網掛け	$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})(y_i - \bar{y})$
I	あり	$+$	$+$	$+$
II	なし	$-$	$+$	$-$
III	あり	$-$	$-$	$+$
IV	なし	$+$	$-$	$-$



この表と左の図から、偏差積の符号と網掛けの有無は一致する。したがって、偏差積をすべて加えれば、その点が網掛けありの側にあるか、網掛けなしの側にあるかがだいたいわかるであろう。網掛けありは右上がり、網掛けなしは右下がりの傾向ということがいえるので、この和が正ならば右上がりの傾向、負ならば右下がりの傾向ということができる。また、和が 0 に近いときは、網掛けありと網かけなしの影響が打ち消しあって、直線傾向が見えにくいということになる。

実際には、このままではデータの個数 n の影響もあるため、それを打ち消すために、

$$s_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

と定義し、さらに、比較のためには、それぞれのデータの単位をそろえる必要があるため、s_x と s_y をそれぞれの標本標準偏差とするとき、次のものを 2 つのデータの**相関係数**と定義する。

$$r = \frac{s_{xy}}{s_x s_y}$$

なお、s_{xy} を**共分散（covariance）**という。r には次の性質がある。

— N O T E —

- $-1 \leqq r \leqq 1$。
- r の絶対値が 1 に近いほど、相関が強い。0 に近いほど無相関。特に、絶対値が 1 ならば、完全な直線関係にある。
- r の符号が正ならば、正の相関、符号が負ならば、負の相関。
- （経験則ではあるが）r の絶対値が 0.7 以上のとき、相関関係があるといえる。

2.9.2 相関係数の求めかた

R で、s.koku と s.hist の相関係数を定義通りに求めてみよう。このとき、「それぞれのベクトルの各要素ごとの計算」が必要になるが、ベクトル同士の演算は、「各要素ごとの計算」の計算になるため、この機能を用いれば相関係数は計算できる。

問 1 以下の手順で、相関係数を求めよ。

（1） s.dev1 に、「s.koku のそれぞれの要素から s.koku の平均を引いたもの」を付値する。

COMPUTER OPERATION ▷
```
>
```

（2） s.dev2 に、「s.hist のそれぞれの要素から s.hist の平均を引いたもの」を付値する。

```
>
```

（3） s.dev1 と s.dev2 の積を求め（つまり各データの偏差積を計算し）、それを s.dev.prod に付値する。

```
>
```

（4） s.dev.prod の和を求めて、「要素の数 −1」で割ったものを s.cov に付値する。

```
>
```

（5） s.cov を、s.koku の標準偏差と s.hist の標準偏差の積で割る。

```
>
```


しかし、これを計算するために、cor() 関数が既に用意されている。引数として、2 つのベクトルを与える。

COMPUTER OPERATION ▷
```
> cor(s.koku, s.hist)↵

> cor(s.koku, s.bio)↵
```

— N O T E —

2.10 相関係数の応用

3 つ以上のデータに対して、相関を調べる方法は、完全なものはない。しかし、その代替になるものはある。その点についても少し考えておこう。また、相関係数の使い方の注意にも触れる。

2.10.1 3 次元以上のデータの相関係数

一般に、3 次元以上のデータにおいて、それぞれがどのような相関を持っているかを調べるのは難しい。簡単に考えるには、まずは、「それぞれの組ごとの相関係数を調べる」方法をとることができる。例えば、ここに準備した `s.matrix7` データには 7 教科（国語・数学・英語・社会・理科・体育・音楽）の点数があるが、このデータの場合は、「国語と数学」「国語と英語」「国語と社会」などのすべての組合せを調べてみるのである。

問 1 この例では、何通りの組み合わせがあるか。

散布図を眺めてみよう。それぞれのデータを取り出して `plot()` 関数で散布図を描いてもよいが、1 つ 1 つを表示していると大変なので、すべてを一気に表示させることができる。これは**対散布図**というもので、`pairs()` 関数を用いて作成する。

COMPUTER OPERATION ▷

```
> pairs(s.matrix7)↵
```

格子状に散布図が並ぶが、左上から右下までの対角線上には科目名が並んでいる。これから、どの散布図がどの組み合わせになるかを読むことができる。

相関係数も計算してみよう。行列という形になっている場合は、`cor()` 関数ですべての組み合わせを計算してくれる。

COMPUTER OPERATION ▷

```
> cor(s.matrix7)↵
```

出力は行列の形になる。それぞれの相関係数は、縦と横がぶつかるところの値を読めばよい。この行列は、**相関係数行列**と呼ばれる。

— N O T E —

問 2 次の問いに答えよ。

(1) この例で、国語と社会の相関係数はいくつか。

(2) この行列は行と列の数が等しい。このような形の行列を、**正方行列**という。そして、相関係数行列では、行と列が等しい部分（これを**対角成分**という）が必ず 1 になる（1 でないものも、1 にかなり近い値になっている）。この理由を考えよ。

(3) i 行 j 列と、j 行 i 列の部分の値は必ず等しい。この理由を考えよ。

2.10.2 相関についての注意

相関は、基本的に「お互いが直線の関係にあるか」を調べることにある。したがって、直線の対応がつけられないものに関しては、直接利用することはできない。

COMPUTER OPERATION ▷

```
> x ← 1:15↵
> y ← (x-8)^2↵
> cor(x, y)↵

> plot(x, y)↵
```

この例では、x と y の間には $y = (x-8)^2$ という関係があるが、この式は放物線（曲線）を表すため、直線関係という見方では判定はできない。しかも、偶然にも相関係数の値が 0 になってしまうため、散布図を見なければその様子が観察できないのである。計算だけでなくグラフィックスによる確認というのも、重要である。

`cor(...)`	相関係数（行列）を求める。2 つのベクトルを与えれば、その間の相関係数を求め、行列を与えれば、各列の間の相関係数を計算した相関係数行列を求める。
`pairs(x)`	行列オブジェクト x に対する対散布図を描く。

— N O T E —

第3章　簡単な回帰分析

3.1 回帰分析の初歩

「相関」では、2 つのデータの関係の度合いを測ることをした。では、2 つのデータの間に因果関係があり、「一方のデータ」から「他方のデータを推測する」ためには、どのようなことを考えればよいだろうか。

3.1.1 データの因果関係と変数

2 つのデータを扱うとき、具体的な値 $x_1, x_2, \ldots,$ や $y_1, y_2, \ldots$ を扱うかわりに、それを代表する変数 X, Y などで表現することがある。特に、2 つのデータの間に因果関係があることが想像できるときは、「一方の変数 X を用いて他方の変数 Y を**説明（explain）**する」作業はよく行われる。数式でいえば、$Y = f(X)$ のような形で表し、具体的な値を代入すれば、この式が（それなりに）成り立つように $f(X)$ を定めるわけである。このとき、説明する変数 X を**説明変数**、**独立変数**といい、説明される変数 Y を**被説明変数**、**従属変数**という。

そして、そのような分析方法を**回帰分析（regression analysis）**といい、定まった方程式 $Y = f(X)$ を**回帰方程式（regression equation）**という。以下、何も仮定しないとき X は説明変数を、Y は被説明変数を表すものとする。

ただし、何が「原因」で何が「結果か」を判定するのは難しい。明らかでないものも多く、さまざまな知識・事実を元に、解析者が判断することも必要である。

問 1 「製品の宣伝費用」と「製品の売上高」では、どちらが「原因」でどちらが「結果」か?

3.1.2 線型回帰

`tokyo.atm` と `fukuoka.atm` は、ある日の東京の気圧と、その前日の福岡の気圧（単位ミリバール mb、ヘクトパスカル hPa $= 10^2\,\mathrm{N/m^2}$ と同じ）のデータである。なお、海面気圧であり、高さの影響はないと考えてよい。

実際に散布図を描いてみる。

COMPUTER OPERATION ▷
```
> plot(fukuoka.atm, tokyo.atm)↵
```
直線関係があるように見える。相関係数も計算する。

COMPUTER OPERATION ▷
```
> cor(fukuoka.atm, tokyo.atm)↵
```

—NOTE—

ところで、天気は西から東へ変化するから、東京の気圧の値は前日の福岡の気圧の値を少なからず反映しているはずである。つまり、この 2 つのデータの間には、何らかの因果関係があると考えることができる。そこで、福岡の気圧を説明変数、東京の気圧を被説明変数として、回帰方程式を求めてみることを考える。

ここでは、直線関係が見えるので、一次式 $Y = A + BX$ の形で表されるものを求めることにする。このように、回帰方程式が一次式で表される回帰を**線型回帰**という。回帰方程式が表す直線を回帰直線という。

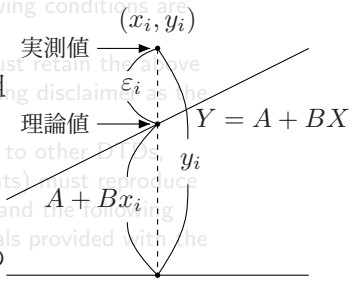
当面の目標は、係数 A と B を求めることである。散布図の上にそれらしき直線を適当に描こうと思えばできるが、それでは定める根拠としては弱い。この直線はどのように定めるとよいのであろうか。

3.1.3 最小二乗法

直線 $Y = A + BX$ が決まったとすれば、データの組 (x_i, y_i) に対して、それぞれ

$$\varepsilon_i = y_i - (A + Bx_i)$$

というものが決まる。これは、各 x_i に対して直線上の点（理論的な値、**理論値**）と、実際のデータ（**実測値・観測値**）との差であり、**残差（residual）**という。



A、B を定めるにおいて、残差 ε_i ができるだけ小さくなるようのにするは自然なことだが、特にデータ全体の残差をなるべく小さくするのが方法の 1 つである。

そこで、

$$L = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \{y_i - (A + Bx_i)\}^2$$

を最小にすることを考える。このような方法を**最小二乗法（method of least squares）**という。その具体的な方法は省略するが、結果は次のような A、B となる。なお、$\bar{x}$、$\bar{y}$ をそれぞれ x_i、y_i の平均とする。

$$B = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \quad A = \bar{y} - B\bar{x}.$$

— N O T E —

3.2 回帰方程式の計算とリスト構造

前節では、線型回帰の理論を考えた。実際に R で計算をさせる場合は、これらの計算はすべて R がやってくれるため、その結果についての処理を知っていればよい。ここでは、その処理のしかたと、そのときに必要なデータの構造についてふれる。

3.2.1 回帰方程式の求めかた・リスト

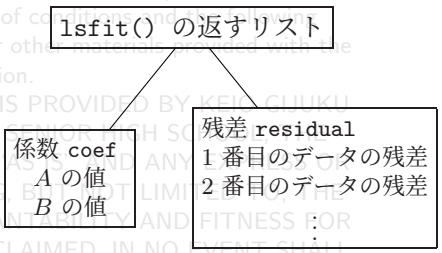
R で回帰方程式を求めるには、`lsfit()` 関数を用いる。これは、

```
lsfit(x, y)
```

という書式を持ち、`x`、`y` はそれぞれ説明変数・被説明変数に対応するデータを表すベクトルである。回帰方程式 $Y = A + BX$ でいえば、`x`、`y` はそれぞれ X、Y にあたるデータを表す。

ところで、計算した結果は単なる A や B の値だけではない。少なくとも関数 `lsfit()` は、これ以外に「各観測対象における残差」を返してくる。これは係数の値とは形も質も異なるデータであるが、これらをまとめて返すためには今までのデータ構造だけでは不十分である。

R では、このような場合**リスト（list）**というデータ構造を用いる。リストとは、簡単にいえば「いくつかのデータをとにかくまとめて 1 つにしたもの」である。図示するときは、枝分かれした図で表すことが多い。S の関数が返してくるリストには、枝にそれぞれ名前がついているため、参照するためにはその枝の名前を指定する必要がある。



関数 `lsfit()` の返す値はリストであるが、具体的には少なくとも次のものを含む。

- 係数（`coef`）··· 回帰方程式の係数 A と B を表す、長さ 2 のベクトル、
- 残差（`residual`）··· 各観測対象における残差のベクトル、長さは観測対象の数（つまり Y を表すデータのベクトルの長さ）と等しい

実際に回帰方程式とその直線を求めてみよう。

COMPUTER OPERATION ▷

```
> ls.result.atm ← lsfit ( fukuoka.atm , tokyo.atm ) ↵
```

`ls.result.atm` に付値せよ　何を？ 回帰分析の結果　何の？ 説明変数は `fukuoka.atm`　被説明変数は `tokyo.atm`　を

で、回帰方程式を求めたことになる。`ls.result.atm` の結果はリストであるが、ここから係数の値を取り出したい。リストの要素を取り出すためには「オブジェクトの

— N O T E —

名前$枝の名前」というように指定する。係数は coef という名前の枝にあるから、

COMPUTER OPERATION ▷
```
> ls.result.atm$coef↵
```

で得られる。このベクトルには名札属性がついていて、どの要素が何を表すかわかるようになっている。

問 1 上の例では、定数 A と定数 B の値はそれぞれ何か。名札属性についている単語から判断せよ（Intercept ··· 切片）。また、回帰方程式を答えよ。

問 2 ls.result.atm 内にある、それぞれのデータの残差を表すにはどうすればよいか？

COMPUTER OPERATION ▷
```
>
```

散布図が描かれているとき、引数に lsfit() 関数の返す値をそのまま与えると、散布図の上に回帰方程式の表す直線を描くことができる。

COMPUTER OPERATION ▷
```
> abline ( ls.result.atm ) ↵
```
（直線描画／何の？ ls.result.atm（回帰分析の結果、つまり回帰直線を／は？）

Note: この例でわかるとおり、R では「回帰方程式の計算」と「その直線の描画」は、関数の仕事としては分離している。通常の統計ソフトウェアでは、回帰分析はこれら両方を同時に行うのが普通であるが、R は途中結果である「回帰方程式の計算結果」をわざと残すようになっている。慣れないうちは面倒だが、その分細かな作業ができるようになっているというのが特徴である。

`lsfit(x, y)` 回帰分析をした結果を返す。x は説明変数を表すオブジェクト（今回はベクトルだが、行列もあり得る）、y は被説明変数を表すベクトルオブジェクト。結果としてはリストとなっており、coef の枝には回帰方程式の係数が、residual の枝にはそれぞれのデータとの残差がある。

`abline(...)` グラフィックス画面上に直線を引く。... にはいろいろなものが入るが、lsfit() 関数の返した値を入れれば、その係数（coef の枝）に従った直線を引く。

— N O T E —

3.3 外れ値とは

最小二乗法を使うことにより、どのようなデータでも回帰方程式を作ることができるが、直線とかけはなれているデータについても注目することは重要である。まず、そのようなデータを眺めてみよう。

3.3.1 外れ値があるデータについて

全体的に直線関係に見えるデータの中に、ごく一部がそれに従っていない場合がある。そのような「外れ値」が含まれているデータに関して回帰分析を行ってみる。

問 1 30 名の年齢 `p.age` と血圧 `p.blood` のデータがある。

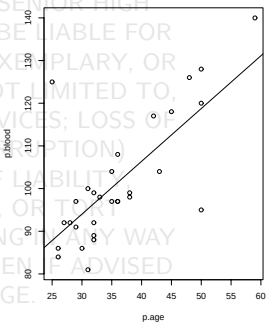
(1) この 2 つのデータの相関係数を求め、x 軸を年齢とする散布図を描け。

COMPUTER OPERATION ▷

```
>
>
```

(2) 「加齢によって血管壁も老化が進み（動脈硬化が起こり）、その結果血圧が上昇する」と考えれば、年齢が説明変数、血圧が被説明変数とすることができる。このとき、X を年齢、Y を被説明変数として回帰方程式を求め、回帰直線を散布図に重ね描きせよ。`lsfit()` 関数の返値は、`ls.result1` に付値すること。

```
>
>
>
```



散布図を見ると、左上と右下に、右上がりになっている集団とは明らかに離れている「外れ値」が存在することがわかる。このようなものが発生してしまうのは、入力ミスや測定ミス、何らかの異常な状態がたまたま発生したなどの「例外」であることが多い。

しかし、外れ値が何であるかを調べることは重要である。その例外が新たな発見につながる場合があるからであり、外れ値をそのまま捨ててしまうのは、このような現象を逃してしまうことになる。たとえ外れ値をはずして再度回帰分析を行うとしても、何をはずすかがわからなければ作業ができない。

— N O T E —

3.3.2 個体の特定

外れ値の処理をするためには、まず「特定の点がどのデータに対応するか」が見つけられなければならない。R で散布図上の点について調べるためには、identify() 関数を用いる。これは、x 軸・y 軸の他に、それぞれに対応する名前のデータをベクトルで与えればよい。例えば、次のように。

COMPUTER OPERATION ▷

```
> sel ← identify ( p.age, p.blood, labels = p.name )↵
```

sel に付値せよ　何を？　個体を特定した結果　何の？　x 座標は p.age　y 座標は p.blood　それぞれの個体の名前は p.name である　を

こうすると、マウスカーソルが変形する。データを表す点の上で左または右ボタンを押せば、対応するデータがあればその名前を表示する。止めるには左右ボタンを同時に押す。

Note: グラフィックスウィンドウ内で、最後の「左右ボタン同時押し」をしないと、R に制御が戻らず入力しても何も起こらない。これに気付かずにいつまでたっても作業ができない状態にならないように。

identify() 関数は、実際に選んだデータが何番目の値であるかを、ベクトルの形で返してくれるので、上のように適当なオブジェクトに付値しておけば、後で再利用できる。

```
> sel↵

> p.name[sel]↵
```

問 2 ここで選んだ者のそれぞれの年齢を表示せよ。

COMPUTER OPERATION ▷

```
>
```

identify(x, y, labels=n)　グラフィックス画面上で、特定の個体を選択するために操作できるようにする。x と y は x 座標と y 座標を与えるベクトルオブジェクト、n はそれぞれに対応する文字列ベクトルオブジェクト。x, y, n の長さは一致していなければならない。マウスボタンを「左右同時押し」すると選択終了を R に伝え、選択したデータの番号を返す。選択終了が伝わるまで、R は操作を待つ。

— N O T E —

3.4 外れ値の除去

前節の方法を用いて、外れ値と思われるものを取り除く処理を行ってみよう。その際、どのように外れ値を決定するかについても考えた方がよい。

3.4.1 外れ値の選択

前節のデータ p.blood と p.age について、外れ値と思われるものを選んでみよう。このとき、直感で選んでみるのも悪くはないのだが、回帰直線は一般に斜めであるから、目視では微妙な判定がしづらいことがある。基本的に、「理想的な値からあまりにもかけ離れている値」が外れ値だから、理論値と実測値の差、つまり「残差」の大きさの散布図が描けるとより精密な判定ができる。縦軸が残差の大きさになるような散布図（管理図）と、その残差の標準偏差を同時に描く関数 ctlchart() を以前用意してあったので、それを用いてみよう。残差は関数 lsfit() の返してくるリストの residual という名前の枝にある。

COMPUTER OPERATION ▷
```
> ctlchart(ls.result1$residual, ylim=c(-50,50))↵
```
ctlchart() 関数の ylim= 引数は、データに応じて適当に指定すること。

これなら、y 方向に 0 から大きく離れているものを選べばよいのでわかりやすい。微妙な判定が必要な場合、このような方法も有効である。平均値から標準偏差の 3 倍以上離れたものを外す、というのが通常だが、状況に応じて判断は変化させる。

管理図上の点は、y 座標はデータの値である。x 座標はそのデータの番号である。管理図は見方をかえれば散布図である。本来、散布図は x 座標・y 座標がそろって描くことができるのだが、一方しか与えなかったときは、y 座標と解釈され、x 座標はデータの番号 1、2、3、... というようにみなされる。identify() 関数でも同様のため、この図では次のようにすればよい。

COMPUTER OPERATION ▷
```
> sel ← identify ( ls.result1$residual,
```
sel に付値せよ　何を？ 個体を特定した結果　何の？ y 座標は ls.result1$residual

```
    labels = p.name )↵
```
それぞれの個体の名前は p.name である （x 座標はデータの番号） を

Note: なお、このような図は plot() 関数でも描くことができる（関数 ctlchart() は内部で plot() を用いている）。

identify(y, labels=n)　y と labels=n の部分しか指定しない場合、y 座標と対応する文字列だけを指定したことになり、x 座標はデータの番号となる。それ以外は以前と同じ。

— N O T E —

3.4.2 外れ値の除去と回帰分析のやりなおし

今回は外れ値を単純に削除して回帰分析をやりなおすことにする。このようにすることは多い。

問 1 sel で選ばれた外れ値を除去し、再度回帰分析を行う。

(1) p.age、p.blood の中から sel の示す場所の値を除去したものを、それぞれ p.age.sel、p.blood.sel として付値せよ。

COMPUTER OPERATION ▷

```
>
>
```

(2) 外れ値を除去したデータについての相関係数を求めよ。

```
>
```

(3) 外れ値を除去したデータで、回帰分析を行い、係数 A と B の値を求めよ。さらに、散布図に回帰直線を重ね描きせよ。lsfit() 関数の返値は ls.result2 に付値すること。なお、abline() 関数の引数 lty=2 は、線種を指定するものである。

```
> plot(p.age, p.blood)↵
>
>
> abline(ls.result1, lty=2)↵
> abline(ls.result2)↵
```

一般に、外れ値を除去したデータのほうが、「残差の分散」は小さい。分散を大きくする要素を取り除いたのだから、当然である。これから、相関関数の値以外に、残差の分散の大きさでも、ある程度回帰分析の評価を行うことができると考えてよい。

問 2 ls.result1 と ls.result2 について、残差の分散を求め、上の事実を確認せよ。

COMPUTER OPERATION ▷

```
> var(ls.result1$residual)↵

> var(ls.result2$residual)↵
```

abline(..., lty=n)　lty=n（n は数）を指定すると、実線以外の線を引くことができる。多数の線を引くときに便利である。

— N O T E —

3.5 回帰方程式の解釈

回帰方程式はただの数式であり、計算しただけで満足してはならない。ここから情報を読み取る作業は解析する者の仕事である。数式をどのように解釈すればよいかを考えてみよう。

3.5.1 回帰方程式の意味

線型回帰では、回帰方程式 $Y = A + BX$ は一次関数だから、係数 B は「X が 1 増加したときの Y の増加する量」である。

東京と福岡の気圧のデータの処理結果は `ls.result.atm` に格納されている。もう一度係数を見てみよう。

COMPUTER OPERATION ▷
```
> ls.result.atm$coef↵
```

問 1 この結果からわかる回帰方程式を答えよ。さらに、係数 B にあたる量は、福岡の気圧と東京の気圧の間の関係をどのように表しているか説明せよ。

また、$Y = A + BX$ のような式が求まったということは、「ある現象において、X の値を定めるとそれに応じて Y の値を予想することができる」ということも意味している。つまり、原因に対しての結果の予想が（ある程度）できるということであり、その値が理論値となるわけである。このとき、予測値を計算する場合、`lsfit()` の返すオブジェクトの値を直接使いたい。`coef` の枝は、ベクトルであるから、これを使えば計算させることができる。

たとえば A にあたる値を取り出すための、一つの方法は次の通りである。

COMPUTER OPERATION ▷
```
> ls.result.atm$coef[1]↵
```

問 2 もう一つの方法として、どうすればよいか。このベクトルには名札属性がついていることを用いる。

COMPUTER OPERATION ▷
```
>
```

— N O T E —

「福岡の気圧が 1010 mb であったときの、翌日の東京の気圧を予想したい」というときは、次のようにするのが方法の一つである。

COMPUTER OPERATION ▷

```
> ls.result.atm$coef[1] + ls.result.atm$coef[2] *    1010    ↵
```

（回帰方程式の A の値 ＋ 回帰方程式の B の値 × 福岡の気圧 1010 mb）

問 3 福岡の気圧が「1011 mb、1013 mb、1017 mb」のときの翌日の東京の気圧を予想するために、次のようにオブジェクト x を作成した。このときの、翌日の東京の気圧の理論値を y に付値するためには、どうすればよいか。

COMPUTER OPERATION ▷

```
> x ← c(1011,1013,1017)↵
>
> y↵
```

問 4 年齢と血圧のデータで、外れ値を除いたのち回帰分析を行った結果 ls.result2 を用いて、以下の問いに答えよ。

(1) X を説明変数（年齢）、Y を被説明変数（血圧）とする。回帰方程式を答えよ。

COMPUTER OPERATION ▷

```
>
```

(2) 年齢が 1 つ上がると、血圧はどのくらい上がると考えられるか。

(3) この回帰方程式を用いて、年齢が 55 歳の人の血圧を予測せよ。

COMPUTER OPERATION ▷

```
>
```


— N O T E —

第4章　モデルとデータ

4.1 モデルとは？

今までの内容は、個々の統計の知識の解説であった。これからはこれらをまとめて「データ解析」の世界に入るわけだが、その前にデータ解析の目標を解説する。

4.1.1 実体・モデルと残差

いろいろな現象には、いろいろな性質を持っているが、必要な性質を取り出し、使いやすい形にして模倣したものを、その**モデル（model）**という。モデルのもとになったものを、**実体（real object）**ということにしよう。例えば、現実の地形を「実体」とするならば、その場所の地図は「モデル」である。

このモデルは、実体の本質をつかむ上で重要な役割を示す。モデルを見れば、実体を直接見ることなく、その実体の特徴を知ることができる。地図を見れば、実際に行ったことのない場所の構造がある程度理解できるのも、地図が実体であるその土地の性質をある程度表しているからである。

しかし、モデルは必ずしも実体のすべてを表しているわけではなく、必ずこの間には差がある。地図の上に建物も表示がしてあっても、建物の色が必ずしもわかるわけではないし、材質がわかるわけでもない。このような差を、**残差（residual）**という。この残差は、実体からモデルを作るときに捨てられてしまった情報であるが、この残差に何が含まれているかを理解していないと、モデルを使う意味がなくなってしまう。つまり、モデルを使う上では、必ず残差—実体と何が違うのか—を意識している必要があるのである。

4.1.2 よいモデルとは

ある実体におけるモデルは、一つとは限らない。事実、同じ地形を表すのに複数の地図が存在することはよくあることであろう。詳しければ、たしかに実体である地形を忠実に再現しているわけで、残差が小さいモデルということができるが、これが優れたモデルであるとは言い切れない。情報量が多すぎる地図は、時として「見にくい」という短所を持つことになる。必要な情報だけを載せた簡単な地図は、「誰にでも理解しやすい」という長所を持つのである。

このようなことを考えると、**よいモデル**というのは、以下のような点を満たすようなものといえるであろう。

- 実体の重要な性質をなるべく忠実に再現する（正確に）

— N O T E —

- 使いやすい（便利）
- 美しい（美的）
- 安く作ることができる（安価）

これらは相反する内容を含んでいるため、実際にはこれらを**バランスよく**実現させることが重要である。何を重要視するかによってモデルは変わることもあり、時と場合によって使い分けるということも必要となる。

問 1 「『プラモデル』と『本物の乗り物』」という例において、残差とは何か。また、「よいモデル」とはどういうものをいうだろうか。

問 2 「モデル」と「実体」の例を一つ作れ。「残差」が何であるかも答えること。

4.1.3 回帰モデルの構築について

データ解析の目的の一つとして、「よいモデルの構築」がある。回帰分析で、$Y = A + BX$ のような式を導くということを考えたが、このような式を求めるというのは「ある特定の要素で、その他の要素を説明しよう」というモデルを構築する、という作業を行っていると考えてもよい。それぞれの要素は変数というもので代表され、モデルを表すためには数式が用いられる。

回帰分析の手法で作られるモデルを「回帰モデル」という。「原因」と「結果」を数式で表そうという考え方と思えばよい。

また、実際の作業では、理論的なこと以外の作業（データの浄化など）を考えなければならない。そのような作業についても考えていくことにする。

— N O T E —

4.2 データの読み込み

R でのモデル構築の具体的な作業を、これから進めていくが、第一歩として、データを R で使えるようにしなければならない。つまり、データの読み込みである。いままではあらかじめデータが用意されていたが、このようなデータは本来自らで探して準備するものである。

4.2.1 これから扱うデータの説明

今回扱うデータは、全国 81 か所にある観測点の、位置と 1 月から 12 月までの平均気温のデータ[*1]である。3 つのファイルにデータは分けてあり、内容は次の通りである。いずれもテキストファイルである。

- `J:¥R¥local¥2003¥obstemp.txt`…81 か所の観測点の、1 月から 12 月の平均気温。各行がそれぞれの観測点であり、それぞれ 12 個の数字（1 月〜12 月の平均気温に対応する）がある。
- `J:¥R¥local¥2003¥obsgeo.txt`…81 か所の観測点の経度（東経）・緯度（北緯）・高度（メートル）。
- `J:¥R¥local¥2003¥obsname.txt`…81 か所の観測点の名前。

いずれも、同じ行が同じ観測点に対応しているようになってはいるが、自分が入力していないデータについては中身を確認するように習慣をつけること。中身を確認もせずに、いきなり内容を信用するのは危険である。何らかのミスで、規則性が崩れているかもしれないからである。また、今回はテキストデータであるから関係はないが、スプレッドシートのデータファイルなどを扱う場合はウィルスなどに感染している恐れもあるから、そのようなチェックも怠ってはならない。

まず、スプレッドシートソフトウェアで読み込んでみるとよいだろう。ソフトウェア起動後、[ファイル (F)]-[開く (O)] で、このファイルを読み込めるようにすれば簡単に読めるはずである。

一方、このデータはテキストファイルである。R に読み込むファイルは必ずテキストファイルでなければならない。また、エディタで読むとわかるが、「空白」で数値を区切ってある形式であり、**CSV ファイルではない**。

[*1] 国立天文台編、理科年表 平成 8 年、丸善株式会社

— N O T E —

4.2.2 R への読み込み

R でテキストファイルのデータを読むには、scan() 関数を用いる。この関数はファイル名を指定すると、そのファイルの内容をベクトルとして読み込む。ファイル名の指定の際には、¥ は / に置き換える。もちろん、その結果を付値しないと使い物にはならないだろう。

たとえば、平均気温のデータを読み込みには、

COMPUTER OPERATION ▷
```
> tmp ← scan ( "J:/R/local/2003/obstemp.txt" ) ↵
```
tmp に付値せよ　何を？　ファイルから読み込んだ結果　何の？　ファイル J:¥R¥local¥2003¥obstemp.txt　を

とする。

ただし、scan() 関数で読み込んだ結果はベクトルになるから、このままではまずいだろう。各行が観測点になるように行列にしなければならない。ファイルは行単位で読むため、行列に流し込む方向は「行方向」である。したがって、行列にしたものを obs.temp に付値するには

COMPUTER OPERATION ▷
```
> obs.temp ← matrix ( tmp, 81, 12, byrow= TRUE ) ↵
```
obs.temp　に付値せよ　何を？　行列にしたもの　何を？　もとのデータは tmp　81 行　12 列　行方向で流し込む？　はい　を

問 1　obs.temp を 1 行で作る式を答えよ。さらに、obs.geo も同様に作成せよ。

COMPUTER OPERATION ▷
```
>
>
```

obs.name は文字列ベクトルである。ファイルの中身も文字列である。このときは、「文字列であること」を指定しなければ誤動作する。scan() 関数に引数として "" を指定すればよい。

COMPUTER OPERATION ▷
```
> obs.name ← scan ( "J:/R/local/2003/obsname.txt",
                    "" ) ↵
```
obs.name　に付値せよ　何を？　ファイルから読み込んだ結果　何の？　ファイル J:¥R¥local¥2003¥obsname.txt

ファイルの中身は文字列である　を

scan(filename)　ファイル filename（文字列）から数値データを読み込み、R のベクトルとしたものを返す。

scan(filename, "")　ファイル filename（文字列）から文字列データを読み込み、R のベクトルとしたものを返す。

— N O T E —

4.3 データの整形（その 1）

読み込んだデータは、必ずしも解析に適していない場合もある。このようなデータを整形して、解析の準備をすることも重要である。

4.3.1 軸名札属性の付値

これから扱うデータが具体的に何かを、軸名札属性として表現することは悪くない。たとえば、ある行の観測点がなんであるかを意識しながら作業をするのは大きな意味がある。

obs.temp は月ごとの平均気温がある行列オブジェクトである。観測点は obs.name として名前が用意されている。一方、「月」の名前は、略称があらかじめ R 側で month.abb として用意されているので、これを用いるとよいだろう。

obs.temp の行に「観測点名」、列に月の略称を付値する。

COMPUTER OPERATION ▷

```
> rownames ( obs.temp ) ← obs.name ↵
> colnames ( obs.temp ) ← month.abb ↵
```

（rownames：行の名札　何の？ obs.temp　に付値せよ　何を？ obs.name（観測点名）　を）

（colnames：列の名札　何の？ obs.temp　に付値せよ　何を？ month.abb（月の略称）　を）

問 1 obs.geo の行に観測点名、列に「longitude, latitude, height」、付くように軸名札属性を付値せよ。なお、a を使うと便利である。

COMPUTER OPERATION ▷

```
> a ← c("longitude", "latitude", "height") ↵
>
>
```

4.3.2 データの補正の作業

今回扱うデータのうち、経度・緯度は角度である。しかし、入力の手間を省くために、小数点以下は分の値をそのまま入力してある。たとえば、138°42′ は 138.42 としてある。1° = 60′ であるから、本当は 138.7 としなければ正しいデータにはならない。

このように、得られたデータは必ずしも使用する際に適切な値でない場合もある。その際は、何らかの作業をともなうものであるが、S ではベクトルの計算により、一気に作業をさせることができる。今回は、次のような関数が役立つだろう。

— N O T E —

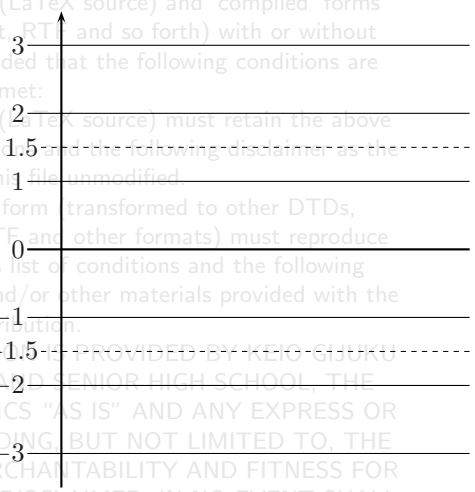
`floor(x)`	ベクトル `x` のそれぞれの要素を、`x` 以下の最大の整数にして返す。
`ceiling(x)`	ベクトル `x` のそれぞれの要素を、`x` 以上の最小の整数にして返す。
`trunc(x)`	ベクトル `x` のそれぞれの要素を、0 に近くなるように切り捨てて返す。

それぞれ、「床関数」「天井関数」「打ちきり関数」と呼ばれるものである。数学では、床関数は $\lfloor x \rfloor$、天井関数は $\lceil x \rceil$ で表される。

問 2 次の式で、`floor()`、`ceiling()`、`trunc()` の違いを理解せよ。

COMPUTER OPERATION ▷

```
> floor(1.5)↵

> floor(-1.5)↵

> ceiling(1.5)↵

> ceiling(-1.5)↵

> trunc(1.5)↵

> trunc(-1.5)↵
```



問 3 以上の関数を用いて、角度の補正を行え。実際の作業は、以下の図をみて考えること。なお、行列の要素抽出をうまく使うと、行列の一部を変更することができる。

COMPUTER OPERATION ▷

```
>
>
>
>
```

— N O T E —

4.4 データの整形（その 2）

もう少しデータを整形しておこう。さらに、地理データであるから、特殊なグラフィックス—地図—を用いて場所を確認する。このようにして、イメージを作っていく作業も重要である。

4.4.1 行・列単位の計算

現在、各観測点には 12 個の気温のデータがある。12 か月を個別に持つのも不便であるから、各観測点の 1 年間の平均気温について考えることにしよう。

本来ならば、各月の長さは微妙に異なるから、その分を考えて「平均」を考えなければならないのだが、ここでは計算を簡略化するために単純に平均をとることにする。この際、「obs.temp の各行のデータの平均を求める」作業が必要となるが、行単位（または列単位）の作業を一気に行う関数も、次のように用意されている。

> `apply(X, MARGIN, FUN, ...)`　行列 X の行・列単位をベクトルとみなして、一括して演算を行う。MARGIN は処理の向きで、1 ならば行単位、2 ならば列単位をベクトルとみなして関数 FUN の第 1 引数に渡す。関数 FUN の第 2 引数以降は、... の部分に書けばよい。

これは、スプレッドシートソフトウェアで、和を求めたり平均値を求めたりする作業に対応する。R では、このような作業も関数を用いるのである。

たとえば、

COMPUTER OPERATION ▷

```
> a ← matrix(1:20, 5, 4)↵
> apply ( a,  2,  sum )↵
```

行列の一括演算　何の？　用いる行列は a　列単位で　それぞれの列に対して関数 sum() を適用　は？

とすれば、行列オブジェクト a の各列をベクトルと見なして和を計算してくれる。

問 1　obs.temp の各行のデータの平均を求め、結果を temp.av に付値せよ。

COMPUTER OPERATION ▷

```
>
```


— N O T E —

4.4.2 地図データ

「日本の観測地点」であるから、場所を確認してみよう。このような付加情報を眺めておくのも悪くはない。本来、グラフィックスはいろいろなグラフを出すことが多いのだが、地図を描画するものも用意しておいた。

問 2 地図を描くとき、x 軸、y 軸はそれぞれ何の値を用いるべきか？

日本地図を描く関数は jpn() である。

COMPUTER OPERATION ▷
```
> jpn()↵
```
日本地図描画　与える情報はない　は？

でそれらしい日本地図が描けるが、このままでは次のような問題がある。

- 観測点が描けない（plot() 関数を用いると一旦画面が消されてしまうため）
- 本来の位置にない沖縄県は、ここでは使わないほうがよい。

沖縄県は描かないことにしよう。jpn() 関数の引数で okinawa=FALSE を与えれば、沖縄県は描かない。

COMPUTER OPERATION ▷
```
> jpn(okinawa=FALSE)↵
```
日本地図描画　何の？　沖縄県は描くか？　いいえ　は？

次に、すでにあるグラフィックスに点を使うには、points() 関数を用いる。文法は plot() 関数と同じ。まず、観測点を先に描くことにする。これは、普通の散布図と同じようにすればよい。実際に、散布図に日本地図を重ねた状態になっている。

COMPUTER OPERATION ▷
```
> points(obs.geo[,1:2])↵
```
点を追加描画　何の？　obs.geo[,1:2] を使って　は？

問 3 identify() 関数を用いて、いくつかの点の地名を調べよ。ただし、あまり地図が大きくないので、観測点の土地名をたくさん出すと見にくくなるかもしれない。

COMPUTER OPERATION ▷
```
>
```

jpn(okinawa=FALSE)†　グラフィックス画面に日本地図を描く。x 座標は経度、y 座標は緯度を表す。okinawa=FALSE をつけなければ沖縄県を別の場所に描く。

points(...)　グラフィックス画面に点を追加する。書式は plot() 関数と同じ。

※ † がついている関数は、実習のために用意したもので、標準の R にはない。

— N O T E —

4.5 回帰分析の実例

前節までで、分析の準備はととのった。まず、「緯度」が「平均気温」に影響を与えている（北海道は沖縄より寒い！）と考えるのは自然なので、その考え方が正しいことを確認してみよう。

4.5.1 回帰分析の実行

まず、緯度と平均気温との関係を調べてみる。緯度を説明変数 X、平均気温を被説明変数 Y として、回帰分析を行ってみる。

問 1 緯度を横軸、平均気温を縦軸となるように散布図を描画せよ。

COMPUTER OPERATION ▷

```
>
```

問 2 目的の回帰分析を行う。`lsfit()` 関数の結果は `ls.result.g1` に付値せよ。さらに、回帰直線を描き、回帰方程式を定めよ。

COMPUTER OPERATION ▷

```
>
>
>
```

回帰方程式：

計算はこれでできたが、本当にこの計算は問題ないだろうか。散布図をよく見ると、直線から明らかに離れている点—外れ値—も少し見受けられる。残差をもとに、外れ値を選んでみよう。

問 3 `ls.result.g1$residual` と `ctlchart()` を用いて、残差の大きさが y 軸になるような散布図を描け。

COMPUTER OPERATION ▷

```
>
>
```

問 4 残差が著しく大きいものを選び出し、それを `outlier` に付値させよ。

COMPUTER OPERATION ▷

```
>
>
>
```

— N O T E —

outlier として選び出したものを外して、回帰分析をやり直してみる。**問 1、問 2 の入力をもう一度行って、散布図を描き** ls.result.g1 **を用いて最初に求めた回帰直線を描く。**このとき、plot() 関数の引数として xlab="latitude" と ylab="temperature" を加えれば、見栄えがよくなるだろう。

さらに、この図に問 4 で選び出した地点の結果を書き加えるのだが、それには次のようにすればよい。

COMPUTER OPERATION ▷

```
> text ( obs.geo[outlier,2],
       temp.av[outlier],
       labels = obs.name[outlier], pos = 4) ↵
```

（text：文字列を追加描画　何の？　x 座標は obs.geo[outlier,2]（outlier で指定した点の緯度）　y 座標は temp.av[outlier]（outlier で指定した点の平均気温）　点の名前は obs.name[outlier]（outlier で指定した点の名前）　名前の配置は 右 は？）

この式がきちんと読めるようであれば、R の文法は心配ないだろう。

問 5 問 4 で選び出した地点のデータをとり除く。outlier で指定した地点を除いた、緯度のデータを lati に、平均気温のデータを temp に付値せよ。

COMPUTER OPERATION ▷

```
>
>
```

問 6 lati と temp を用いて回帰分析をやり直す。lsfit() 関数の結果は ls.result.g2 に付値し、回帰直線を記入せよ。さらに、回帰方程式を求めよ。

今度の直線は破線にする。abline() 関数の引数に lty=2 を加えるとよい。

COMPUTER OPERATION ▷

```
>
>
>
```

回帰方程式：

text(x, y, labels=n, pos=m)　グラフィックス画面上に、文字列を追加する。x, y は x 座標と y 座標を表すベクトルオブジェクト、n はそれぞれに対応する文字列ベクトルオブジェクト。x, y, n の長さは一致していなければならない。

文字列を書く場所は pos=m で指定する。m は数値で、1 〜 4 でそれぞれその点の下・左・上・右に書く。

— N O T E —

4.6 作業結果の記録

解析結果をレポートなどにして提出する場合、R の出力を別の形式で保存できるようにすることが必要となる。印刷・ファイルに保存・他のソフトウェアへの貼り付けなど、R 以外の作業についてふれる。

4.6.1 作業履歴の保存

操作したものは、基本的にウィンドウ内に残っていて、スクロールバーを移動させることによって、前のものを見ることができる。さらに、その内容を保存するときは、[File]-[Save to File] でよい。テキストファイルで保存されるので、他のソフトウェアで読み込むときには気をつけること。

4.6.2 グラフィックスの印刷

前節の結果がグラフィックス画面に得られた。これを単独で印刷することは可能である。ただし、その際には必ず**作業者のログイン名をグラフィックス上に貼り付けること。**後で手書きで名前を付け加えても、正式な物であるとはみなさない。これは、印刷した者が誰か、という責任所在を明らかにするものであり、確かにレポート作成者が印刷したものだという証拠にもなる。

グラフに名前を貼り付けるには、

COMPUTER OPERATION ▷

```
> stamp()↵
```

とする。

次に、グラフィックスが表示されているウィンドウの、[File]-[Print] メニューを選ぶ。Windows の標準的な印刷ダイアログが表示されるので、**必ずプリンタ名を確認して**「OK」を押すこと。これでプリンタに印刷指令が送られる。

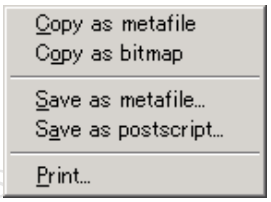
印刷指令が送られても、すぐには印刷されない。プリンタは共有して使っていて、多数の印刷指令が送られてきた場合には「列」になって待っている。プリンタの処理に時間がかかる場合、列がなかなか短くならないので、印刷されなかったの錯覚する場合が多いのだが、**一旦列に並んだ印刷指令は、よほどのことがない限り勝手には消えない。**自分が出した印刷指令は、以下の URL で確認できる。時間がかかるようであれば、web ブラウザでチェックしてみること。なお、この画面から列に並んだ印刷指令を取り消すこともできる。

2004/4/5

http://www2.sfc-js.keio.ac.jp/printer/

— N O T E —

4.6.3 他のソフトウェアへの貼り付け



グラフィックスは、他のソフトウェアへも「貼り付け」られる。グラフィックスが印刷されているウィンドウ内でマウスの右ボタンを押すと、右のようなポップアップメニューがあらわれる。ここで、

- Microsoft Word に貼り付けるなど、通常の使い方ならば、「Copy as metafile」（メタファイル形式でコピー）を
- Web に貼り付けるなど画像として扱いたいならば「Copy as bitmap」（ビットマップ形式でコピー）を

選択する。図がクリップボードに保存される。

例として、Microsoft Word に貼り付けてみよう。「Copy as metafile」で、メタファイル形式でクリップボードに保存してから、Microsoft Word を起動し、[編集(E)]-[貼り付け (P)]（または Ctrl+V）をしてみよう。図として貼り付けられる。そのままだと使いづらいだろうから、大きさを変化させたり、「図の書式設定」の「レイアウト」で調整するなどしてもらいたい。

ビットマップ形式で貼り付けも可能だが、メタファイル形式と比べて図の品質が下がったり、貼り付けた先のファイルが大きくなったりとあまりいいことはない。

`stamp()`†	図にログイン名と作業した日時を加える。

※ † がついている関数は、実習のために用意したもので、標準の R にはない。

— N O T E —

4.7 重回帰分析と空間の散布図

前節では余計なデータを取り除いたが、取り除いたデータが何かを考えれば、何が問題かは予想はつくだろう。この場合、高度のデータも用いた分析を行えば、よりよい結果が得られることが期待できる。このとき、「説明変数」が 2 つ以上になる。

4.7.1 複数の説明変数がある回帰分析

回帰分析において、説明変数は複数あってもよく、説明変数が 1 つの場合を**単回帰（simple regression）**、2 つ以上の場合を**重回帰（multiple regression）**という。重回帰分析では、たとえばデータから

$$Y = A + B_1 X_1 + B_2 X_2$$

などという形の式の係数 A, B_1, B_2 を求めることになる。

ここでは、説明変数は緯度を X_1、高度を X_2、被説明変数 Y を平均気温 Y として、上のような式を作ってみることにする。

ところで、説明変数が 2 つのとき、一つ一つのデータは 3 つの値の組になるため、データの表す点は空間内の点と認識できる。空間では、x 軸・y 軸のほかに z 軸が現れる。さらに、上の式で表された式は空間内の平面を表す。これらを表す図を描いてみよう。

4.7.2 空間の散布図を描くために

コンピュータの画面は平面であるから、空間の様子を描くときは、平面に投影してそれらしく見せることになる。R でも擬似的にそのように見せる関数が用意されているが、そのままでは使うことができない。

コンピュータ言語では、ある一定の機能を集めたもの（関数）の集団を**ライブラリ（library）**ということがある。これらの中には、標準で（つまり何もしなくても）使えるものと、ある手続きをして初めて使えるようになるものがある。いちいち手続きしなければ使えないというのは不便のように思えるかもしれないが、そのような関数はたいてい用途が限られている。そういうものをいつでも使えるようにするとコンピュータの資源（メモリなど）が無駄になってしまう。そのため、必要なときだけ使えるようにしているのである。もちろん、（気の利いたソフトウェアならば）用途によって、あらかじめいつでも使えるような手続きを踏むようにすることもできる。

— N O T E —

ここでは、「空間の散布図を描く」ライブラリを「追加する」という作業が必要になる。

COMPUTER OPERATION ▷
```
> library ( scatterplot3d ) ↵
```
ライブラリの追加　何の？　空間の散布図を描くライブラリ　は？

この作業で、scatterplot3d() という関数が追加される。しかし、この関数の使い方は多少難しいため、Microsoft Windows のようなグラフィカルユーザインタフェース（GUI）で操作するライブラリも追加しておこう。ここでは、Tcl/Tk という（Microsoft Windows に限らない）さまざまなウィンドウシステムで使うことのできるツールキット（GUI を提供する機能群）を使うことにする。

COMPUTER OPERATION ▷
```
> library ( tcltk ) ↵
```
ライブラリの追加　何の？ Tcl/Tk GUI ツールキットを使うライブラリ　は？

ここでは、次のような関数を用意しておいた。以上の関数を内部で使っている。

```
eazyplot3d(x, y, z, ang=40, main="")
```

x, y, z はそれぞれ等しい長さのベクトルで、座標のデータである。ang は見る角度を決める値である。main は、この図にタイトルをつけるときに与える。

COMPUTER OPERATION ▷
```
> eazyplot3d ( obs.geo[,2], obs.geo[,3], temp.av ) ↵
```
空間の散布図描画　何の？　x 座標は緯度　y 座標は高度　z 座標は平均気温を　は？

この関数を用いると、グラフィックスウィンドウの他に図を操作するウィンドウが現れる。左右に動くスライダを動かせば、見る角度が変わる。また、Draw plane のチェックをはずすと、平面を描かない。この平面は回帰方程式で表せる平面で、内部で計算している。

この関数は、空間の散布図を「疑似的に」表示する。「擬似的に」というのは、軸の一つは必ず横方向で、もう一つの軸が必ず縦方向になるからである（きちんと平面に投影すると、絶対にこうはならない！）。残りの軸が x 軸をどれだけの角度にするか指定することにより、見え方が変わっているのである。

library(x)　ライブラリ x を追加する。
eazyplot3d(x, y, z, ang, main)†　x, y, z 座標が x, y, z で与えられる空間の散布図を描画する。ang は見る角度を決める値である。main は、この図にタイトルをつけるときに与える。

※ † がついている関数は、実習のために用意したもので、標準の R にはない。

Note: ライブラリ scatterplot3d や、Tcl/Tk GUI ツールキットは、R の標準装備ではない（ライブラリ tcltk は標準装備）ため、別に導入する必要がある。自宅で R の操作をしている者はその点について気をつけること。不明な点は授業担当者に確認するとよい。

— N O T E —

4.8 重回帰分析の実行と回帰の評価

前節の散布図で平面の様子を観察することはできるが、具体的な式を求めるときは、改めて計算しなおさなければならない。また、いままで行ってきた計算を振り返り、どれがよいモデルかを結論付ける必要もある。

4.8.1 重回帰分析の計算方法

平面の方程式を実際に求めるときは、やはり関数 `lsfit()` を用いる。第 1 引数に行列（各列がそれぞれの説明変数に対応する）を与え、あとは同じである。

今回は、`lsfit()` 関数の結果は `ls.result.g3` に付値する。

COMPUTER OPERATION ▷

```
> ls.result.g3 ← lsfit ( obs.geo[,2:3], temp.av ) ↵
```

（`ls.result.g3` に付値せよ／何を？／回帰分析の結果／何の？／説明変数は緯度・高度の順／被説明変数は平均気温／を）

`lsfit()` 関数が返す結果は単回帰の場合と同様である。`ls.result.g3$coef` や `ls.result.g3$residual` は同じように使うことができる。

問 1 `ls.result.g3` を調べ、求めた回帰方程式を書け。X_1 を緯度、X_2 を高度として用いるとよい。

COMPUTER OPERATION ▷

```
>
```

回帰方程式：

問 2 残差がどのようになったか、`ls.result.g3$residual`、`ctlchart()` や `identify()` を用いていくつか調べてみよ。

COMPUTER OPERATION ▷

```
>
>
```

なお、$Y = A + B_1X_1 + B_2X_2$ で与えられた式は、次のことを示している。

- X_1 だけが 1 増加したとき（X_2 は固定）には Y は B_1 増加する。
- X_2 だけが 1 増加したとき（X_1 は固定）には Y は B_2 増加する。

4.8.2 回帰の評価と考察

回帰の評価は、残差の分散の大小（もちろん小さいほうがよい）の他に、単回帰ならば相関係数（この場合は気温と緯度の相関係数）でも行うことができる。一つの方法だけでなく、さまざまな方法を試みて、総合的に判断するとよい。

— N O T E —

問 3 以下の表をうめよ。小数点以下第 3 位を四捨五入する。

	相関係数の絶対値	残差の分散
ls.result.g1		
ls.result.g2		
ls.result.g3	×	

以上の作業で、一通りの解析が終わったわけだが、単純に計算して終わりにしてはならない。ここから何が読み取れるか、現実との比較などをする。

まず、以上の操作で得られた回帰方程式をもう一度書いてみる。X_1 を緯度、X_2 を高度、Y を平均気温としてみるとよい。この式を作成したことで、「緯度・高度から平均気温を導く回帰モデル」を作成したことになる。つまり、「緯度・高度からその場所の平均気温はこうなるだろう」という理想の型を作り出せたことになる。

ls.result.g1 （緯度と平均気温）
ls.result.g2 （緯度と平均気温、外れ値除去後）
ls.result.g3 （緯度・高度と平均気温）

もちろん、これらの式は、そのままでは意味をなさず、「回帰の評価」を行って適切なものを採用することになるだろう。さらに、式のそれぞれの意味を解釈する必要もある。「緯度が 1 度上がると気温はどうなるか」「高度が 100 m 上がると気温はどうなるか」ということは読み取れるはずであり、場合によっては既存の知識と照らし合わせる作業も必要になるだろう。

また、回帰直線・平面と、実際の点の位置関係にも注目するとよい。回帰方程式はあくまで「モデル」であり、実際には何らかの偏りがグラフィックス上から読み取れ、そこから新たな発見があるかもしれないのである。または、新たに考慮すべき要素を発見し、それを説明変数に加えることも有効かもしれないのである。ただし、説明変数の数をやみくもに増やすのは得策ではない。「説明すべき材料」が多くなれば、事柄を正確に表せるのは当たり前のことなのである（しかも、重複する情報を与えすぎるのは有害な場合もある！）。

lsfit(x,y)　　x が行列の場合、重回帰分析を行う。

— N O T E —

付 録A　関数電卓の使いかた

A.1 電卓の使いかた（その 1）

関数電卓（いわゆる「慶応電卓」）での 1 変数統計処理のやりかたについて扱う。

A.1.1 キーの表記について

関数電卓のキーは、1 つのキーに複数の機能が割り当てられている。ただ単にキーを押した場合は、たいていキーの表面に印刷されている機能が働く。SHIFT などのキーを押してから別のキーを押すことにより、別の機能を働かせることができる。ここでは、このようなときSHIFT$1/x[x!]$のように記述することにする。

A.1.2 通常の計算

電源 ON はON、電源 OFF はSHIFTAC[OFF]である。計算を始めるときは、ACを押して現在入力されているものを消しておくとよい。

この電卓では、R と同じように式をそのまま入力すれば計算ができる。負の符号は(-)、引き算は−と区別している。最終結果を得るためには =を押す。5.2×10^{12} や 5.6×10^{-20} という数は5.2EXP12や5.6EXP(-)20でよい。直前の計算の値を使いたいときは、ANSを用いる。

◀▶▲▼で R と同じような編集ができる。また、DELは行末にあるときは最後の 1 文字を、そうでないときはカーソル上の文字を消す。

A.1.3 1 変数統計処理

MODEキーで電卓の状態をいろいろ変更できる。ここでは、統計処理をする状態にしてみよう。MODEを 2 回押し、「SD REG BASE」と表示されている状態にしたら1を押す。もとに戻すときはMODE1でよい。

統計処理を始めるときはSHIFTMODE[CLR]1=で以前のデータを消去する。

データを入力するには、数値を入力してからM+[DT]を押す。必要な数だけデータを入力すればよい。同じ値を複数まとめて入力するならば、M+[DT]を複数回押すか、SHIFT,[;]で区切って度数を入力する。後者の場合、度数が記録される。

▲▼で入力したものが見れる。x ○=というのは、○番目のデータの値、Freq ○=というのは、○番目のデータの値の度数（通常は 1、SHIFT,[;]で重複回数を指定

— N O T E —

したときだけその値）である。以後の説明では、SHIFT[;]は使わない。

データの修正は、その値が表示されているときに、正しい値を入力して=を押す。度数も修正できるため、表示されているのが x ○=か Freq ○=を気をつけること。誤ってデータを書き換えてしまわないよう、普通の計算をするときはACを押してから式を入力することを忘れないように。

データの消去は、x ○=か Freq ○=が表示されているときにSHIFTM+[CL]である。消去したデータ以降の番号が一つずつ繰り上がる。

Ex. 55、54、51、55、53、53、54、52 というデータを入力するには、MODEMODE1で統計処理の状態にし、SHIFTMODE[CLR]1=で以前のデータを消去してから、5 5 M+[DT] 5 4 M+[DT] 5 1 M+[DT] 5 5 M+[DT] 5 3 M+[DT] M+[DT] 5 4 M+[DT] 5 2 M+[DT] と押す。

Ex. 51 のデータを削除するには、これが 3 番目のデータだから、x3=または Freq3=が表示されている状態で SHIFTM+[CL]。

平均などの値を使いたいときは、次の通り。

SHIFT2[S-VAR]1[$\bar{x}$]	平均
SHIFT2[S-VAR]3[$x\sigma_{n-1}$]	標本標準偏差
SHIFT1[S-SUM]3[n]	データの数

これらはすべて計算の途中の値として使うことができる。値を求めるときは=を押すこと。

次のデータの平均・標本標準偏差を求めてみよう。平均は 57.725、標本標準偏差は 8.2803$\cdots$ になる。

45	76	69	60	69	58	60	44	43	51
69	58	59	56	55	57	57	56	58	57
50	65	45	72	65	61	61	56	52	62
61	49	50	65	53	62	55	70	40	58

問 1 次のデータの平均・標本標準偏差を求めよ。

33	39	35	44	61	51	32	51	48	33
50	49	50	46	41	57	32	39	44	41
41	30	41	35	43	52	43	42	29	52
57	38	42	37	39	37	47	38	36	28
46	49	44	36	44	37	44	44	54	32

— N O T E —

A.2 電卓の使いかた（その 2）

関数電卓（いわゆる「慶応電卓」）で相関・回帰の計算をする方法を述べる。

A.2.1 2 変数統計処理の基本

2 つのデータを同時に処理したり、相関係数を求めたりすることができる。ここでは、それぞれのデータを X と Y としておく。

[MODE][MODE][2][1]で、基本的な 2 変数統計処理の状態にできる。

1 変数統計処理の時と同様、[SHIFT][MODE[CLR]][1][=]で以前のデータを消してから処理をはじめること。

今回は、2 組のデータであるから、2 つの値を入力しなければならない。先に X に対応するデータを入力し、[,]で区切って、それから続けて Y に対応するデータを入力し、[M+[DT]]で入力する。データの修正方法は 1 変数のときと同様である。y ○=という形のものもあることに注意せよ。

Ex. データの組 $(4, 1003)$、$(8, 1005)$、$(12, 1010)$、$(16, 1011)$、$(20, 1014)$、$(24, 1017)$ を入力することを考える。まず、[SHIFT][MODE[CLR]][1][=]で以前のデータを消去してから、次の操作をする。

[4][,][1][0][0][3][M+[DT]] [8][,][1][0][0][5][M+[DT]] [1][2][,][1][0][1][0][M+[DT]]
[1][6][,][1][0][1][1][M+[DT]] [2][0][,][1][0][1][4][M+[DT]] [2][4][,][1][0][1][7][M+[DT]]

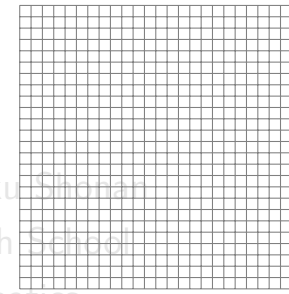
Ex. 上の作業の後、$(24, 1017)$ の値の組を消す場合、これは 6 番目のデータだから、x6=, y6=, Freq6=のいずれかを表示させた状態で[SHIFT][M+[CL]]を押す。

それぞれのデータの組の平均などを求めたり、相関係数・回帰式の係数を使いたいときは次の通り。なお、回帰方程式は $Y = A + BX$ の形である。

[SHIFT][2[S-VAR]][1[$\bar{x}$]]	X に対応するデータの平均
[SHIFT][2[S-VAR]][▶][1[$\bar{y}$]]	Y に対応するデータの平均
[SHIFT][2[S-VAR]][3[$x\sigma_{n-1}$]]	X に対応するデータの標本標準偏差
[SHIFT][2[S-VAR]][▶][3[$y\sigma_{n-1}$]]	Y に対応するデータの標本標準偏差
[SHIFT][2[S-VAR]][▶][▶][1[A]]	入力した値の回帰方程式の係数 A
[SHIFT][2[S-VAR]][▶][▶][2[B]]	入力した値の回帰方程式の係数 B
[SHIFT][2[S-VAR]][▶][▶][3[r]]	入力した値の相関係数 r
[SHIFT][2[S-VAR]][▶][▶][▶][2[$\hat{y}$]]	直前の値を X に代入し、そのときの Y の値
[SHIFT][2[S-VAR]][▶][▶][▶][1[$\hat{x}$]]	直前の値を Y に代入し、そのときの X の値

— N O T E —

問 1 上のデータ（(24, 1017) を消したもの）の相関係数、回帰方程式を求めよ。



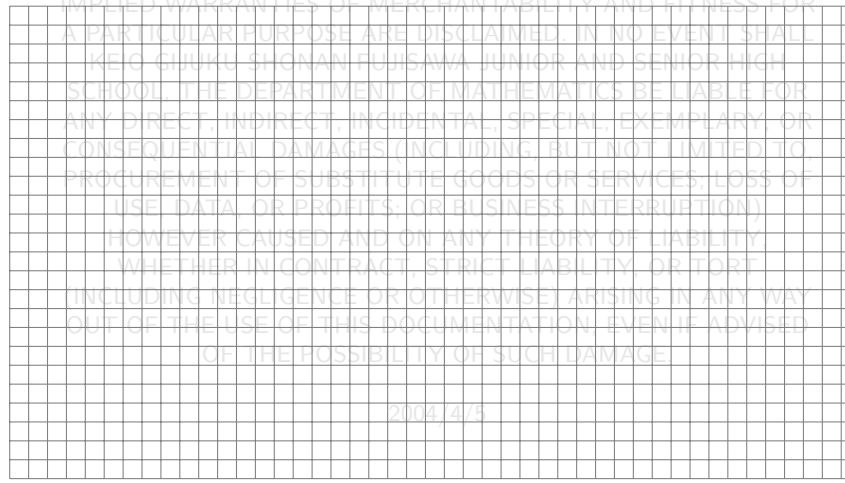
上の例で求めた回帰方程式で、値の予測をしてみよう。$X = 11$ のときの Y の値を予想するには、1 1 SHIFT 2[S-VAR] ▶ ▶ ▶ 2[$\hat{y}$] と押す。答えは 1007.9。

$Y = 1010$ のときの X の値を予想するには 1 0 1 0 SHIFT 2[S-VAR] ▶ ▶ ▶ 1[$\hat{x}$] と押す。答えは 14。

問 2 上の例で求めた回帰方程式で、$X = 18$ のときの Y の値、$Y = 1000$ のときの X の値を求めよ。

問 3 以下のデータについて、散布図を描き、相関係数・回帰方程式を求めよ。また、$X = 25$ のときの Y の値、$Y = 13$ のときの X の値を求めよ。

x	y	x	y	x	y	x	y	x	y	x	y	x	y
32	9	28	11	42	4	34	8	12	20	17	17	14	19
25	13	21	14	19	16	14	16	21	15	19	13	26	25
27	12	13	18	35	5	40	6	12	18	27	11	44	3
28	10	17	19	40	4	23	13	11	20	24	12	11	20
37	7	19	17										



— N O T E —

付 録B　R について

B.1 R の入手方法と作業環境の移動

R は GPL のフリーソフトウェアである。自宅で実習をしたい者は手にいれるとよい。また、（予定だが）CD-ROM の貸与も行う。なお、World Wide Web（web）の情報は刻々と変化している。以下の記述は 2003 年 4 月 6 日現在のものであり、印刷物となったときには変化しているかもしれない。

B.1.1 インターネットからのソフトウェアの入手

この実習で必要なものは、次の 3 つである。

- R 本体
- Tcl/Tk
- 実習用データ

このうち、実習用データ以外はインターネットから入手できる。World Wide Web が一番わかりやすいだろう。R は `http://www.r-project.org/` からたどれるリンク先から入手できる。2003 年 4 月 6 日現在の最新版は 1.6.2 であり、`rw1062.exe` というファイルにすべてが入っている。かなり巨大なファイルなので、電話回線で入手するつもりの者は覚悟すること。

`scatterplot3d()` は標準のライブラリではないので、追加する必要がある。上であげたところから、`scatterplot3d.zip` を入手してほしい。

Tcl/Tk は `http://www.activestate.com/` から ActiveTcl という名前で提供されている。最新版は 8.4.2.0 である。`ActiveTcl8.4.2.0-win32-ix86.exe` というファイルにすべてが入っている。これも巨大なファイルなので、使用の際には注意すること。

B.1.2 インストール方法

以上のソフトウェアは、実習用データとともにあらかじめ入手して CD-R に焼いてあるので、必要がある者は CD-R の貸与をうけるとよい。簡単なインストール方法を以下にあげる。ただし、Windows 版のみである。

まず、作業の前に適当なフォルダを開き、[ツール (T)]-[フォルダ オプション (O)] の中の「表示」タブを選択し、「登録されているファイルの拡張子は表示しない」のチェックをはずす。これにより「`.doc`」だの「`.xls`」だのファイル名のよけいな部

— N O T E —

分も出てきてしまうが、この部分を隠していることによりウィルスにだまされやすくなるのも事実なので、この際慣れておくとよい。

1. rw1062.exe を実行し、R をインストールする。インストール先などを勝手に変更はしないこと。
2. ActiveTcl8.4.2.0-win32-ix86.exe を実行し、Tcl/Tk をインストールする。これも、インストール先などを勝手に変更しないこと。
3. rwork03.exe が実習用データのインストーラである。実行して、出力されるメッセージに従えばデータがインストールされる。
4. デスクトップ上に R のショートカットができるので、それをダブルクリックして実行し、[Packages]-[Install package from local zip file...] を選択する。ファイル選択の状態になるので、scatterplot3d.zip を選択する。

なお、実習用データのインストーラは、My Documents フォルダのなかに rwork というフォルダを作成する。そこに R 関係のデータがすべて格納されている。

学校での個人のデータは I ドライブの .Rdata というファイルであり、これを持ち帰り、rwork フォルダにコピーをすれば作業の続きができる。逆の動作で自宅の作業を学校に持っていくこともできる。ただし、.Rdata ファイルが破損するとデータが消えてしまうため、取り扱いには注意すること。

— N O T E —

付 録C　練習問題

C.1 練習問題その 1

問 1 次の式を計算する R の式を答えよ。また、計算した結果が普通の数ならばその値を、普通の数でない場合はその理由を答えよ。

(1) $6 - 35 \div 7 \times 9$

(2) $(-2)^2 - 3 \times 5$

(3) $6 \div \{9 - (3 + 2) \times 5 + (-4)^2\}$

(4) $\sqrt{2 \times \{3 \div 2 + (6 - 9) \times (-1)^3\}}$

問 2 次の図 1・図 2 について、あとの問いに答えよ。

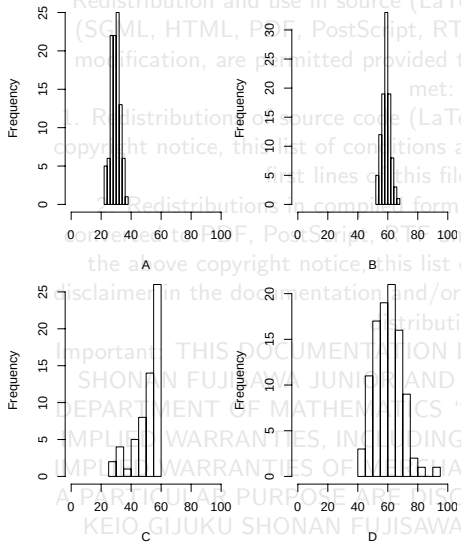


図 1

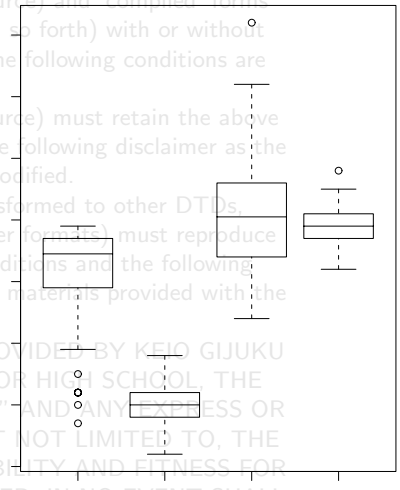


図 2

(1) 図 1・図 2 はそれぞれ何といったか。

(2) 図 1・図 2 は 4 つのオブジェクト a1、a2、a3、a4 に関してのものである。a1〜a4 の平均値・標本標準偏差の値は次の表の通りである。

	a1	a2	a3	a4
平均値	60.61	59.29	29.97	51.41667
標本標準偏差	9.01166	2.886034	3.131640	8.25092

図 1・図 2 を作成する操作（ただし図のタイトルなどは意図的に消去してある部分がある）は、次の通りである。空欄に入るものを答えよ。

— N O T E —

【図 1 の作成方法 なお横軸の範囲は 0 ～ 100 である】

COMPUTER OPERATION ▷

```
> par(mfrow=c(2,2))↵
> hist( ア , オ )↵
> hist( イ , オ )↵
> hist( ウ , オ )↵
> hist( エ , オ )↵
```

【図 2 の作成方法】

COMPUTER OPERATION ▷

```
> par(mfrow=c(1,1))↵
> boxplot( カ , キ , ク , ケ )↵
```

問 3 次の各問に答えよ。

(1) 次の手順を表す R の式を作成せよ。

a. ベクトル A の各要素から、A の中央値を引いたものを A1 に付値する。

b. ベクトル A1 の各要素を 0.7 倍したものを A2 に付値する。

c. A2 の各要素に 30 を加えたものを A3 に付値する。

d. 以上の操作を A3 と A だけを用いて 1 行で表記する。

(2) 前問で、A の要素が 29、49、85、63、53 のとき、A3 の各要素の値を答えよ。

問 4 温度表記で「摂氏 x 度」を「華氏 y 度」に直す式は $y = \frac{9}{5}x + 32$ である。

(1) 「摂氏」で表記しても「華氏」で表記しても同じ値になる温度を求めよ。

(2) ベクトル B の各要素は「華氏」の気温のデータである。これを「摂氏」に直したものを B1 に付値したい。どのようにすればよいか、1 行で答えよ。

(3) ベクトル B の各要素が 0.5, 95, 41, −22 のとき、前問の操作で得られるベクトル B1 の各要素の値を求めよ。

— 解答 —

問 1 (1) `6-35/7*9`, −39 (2) `(-2)^2-3*5`, −11 (3) `6/(9-(3+2)*5+(-4)^2)`, 0 で割っている (4) `sqrt(2*(3/2+(6-9)*(-1)^3))`, 3

問 2 (1) 図 1 · · · ヒストグラム, 図 2 · · · 箱ひげ図 (2) **ア** a3, **イ** a2, **ウ** a4, **エ** a1, **オ** `xlim=c(0,100)`, **カ** a4, **キ** a3, **ク** a1, **ケ** a2

問 3 (1) a. `A1 ← A-median(A)` b. `A2 ← A1*0.7` c. `A3 ← A2+30` d. `A3 ← (A-median(A))*0.7+30` (2) 13.2, 27.2, 52.4, 37, 30

問 4 (1) −40 度 ($x = \frac{9}{5}x + 32$ を解く) (2) `B1 ← (B-32)*5/9` ($x = \frac{5}{9}(y - 32)$ である) (3) −17.5, 35, 5, −30

— N O T E —

C.2 練習問題その 2

問 1 次のようなオブジェクト d がある。

COMPUTER OPERATION ▷

```
> d↵
 [1] 16 23 14 29 29 23 31 23
 [9] 29 33 28 29 20 45 39 37
[17] 33 22 29 28
```

次の操作において、**ア**、**イ**、**ウ** に出力される値を、小数第 3 位を四捨五入して求めよ。計算は R を用いず、電卓などを使うこと。計算の際は、右の表を用いるとよい。

COMPUTER OPERATION ▷

```
> length(d)↵
[1] ア
> mean(d)↵
[1] イ
> sqrt(var(d))↵
[1] ウ
```

平均値 $\bar{x} =$

標本分散 $S^2 =$

標本標準偏差 $S =$

i	x_i	$x_i - \bar{x}$	$(x_i - \bar{x})^2$
1	16		
2	23		
3	14		
4	29		
5	29		
6	23		
7	31		
8	23		
9	29		
10	33		
11	28		
12	29		
13	20		
14	45		
15	39		
16	37		
17	33		
18	22		
19	29		
20	28		
	合計		

問 2 以下の問いに**簡潔に**答えよ。

(1) 量的データと質的データの違いを述べよ。

(2) 代表値とはなにか。平均値と中央値の求め方・性質の違いは何か。

(3) 散布度とは何か。分散と四分位数の求め方・性質の違いは何か。

(4) R を用いていると、行頭に [] でくくられた数値が表示として現れることがよくある。この表示の意味を述べよ。

(5) R の計算結果の表示において、`1.2425e+35`、`5.2315e-12` とはそれぞれどういう意味か、述べよ。

— N O T E —

(6)　R におけるベクトルの長さとは何か、述べよ。

(7)　論理数のベクトルで、TRUE の数を数えるための方法を説明せよ。作業の方法だけでなく、そのようにしてできる理由も説明すること。

(8)　2 組のデータ「0, 4, 8」と「2, 2, 2, 2, 4, 6, 6, 6, 6」において、それぞれ $(x_1-\bar{x})^2+\cdots+(x_N-\bar{x})^2$（分散の定義の分子の式）の値を計算し、これから分散の定義の式で「データの個数」に関係する値で割っている理由を説明せよ。

— 解答 —

問 1

データの個数は 20 個だから、**ア**は 20。平均値は $(16+23+\cdots+28)\div 20=28$ だから、**イ**は 28。

　右の表の結果から、標本分散は $1070\div(20-1)=56.31579\cdots$、標本標準偏差は $\sqrt{56.31579\cdots}=7.504\cdots$ だから、**ウ**は 7.50。

（問 2 は略）

i	x_i	$x_i-\bar{x}$	$(x_i-\bar{x})^2$
1	16	−12	144
2	23	−5	25
3	14	−14	196
4	29	1	1
5	29	1	1
6	23	−5	25
7	31	3	9
8	23	−5	25
9	29	1	1
10	33	5	25
11	28	0	0
12	29	1	1
13	20	−8	64
14	45	17	289
15	39	11	121
16	37	9	81
17	33	5	25
18	22	−6	36
19	29	1	1
20	28	0	0
	合計		1070

※注意

ここにあげた問題はあくまで一つの例であり、たとえ記述問題であってもいろいろ問うべき題材はあります。また、問い方もいろいろ変化させることができます。自分の実力のチェックのために用いてください。**この練習問題にないような問題がテストに出ても不思議ではありません。**そのあたりは勘違いしないように。

C.3 練習問題その 3

数値を求める問題は、すべて小数点以下第 3 位を四捨五入すること。

問 1 s.phys1 はあるクラス（40 人）の物理の 1 学期試験の結果である。

COMPUTER OPERATION ▷

```
> s.phys1↵
 [1] 81 70 50 57 76 57 73 52 66 74 72 57 66 58 68 58 99 64 61 69
[21] 60 53 57 60 62 63 71 50 48 78 65 58 70 63 46 73 80 56 64 79
```

(1) 次の操作において、［ア］、［イ］、［ウ］に出力される値を、小数第 3 位を四捨五入して求めよ。計算は R を用いず、電卓などを使うこと。

COMPUTER OPERATION ▷

```
> length(s.phys1)↵
[1] ［ア］
> mean(s.phys1)↵
[1] ［イ］
> sqrt(var(s.phys1))↵
[1] ［ウ］
```

(2) 「z.phys1 に、物理の 1 学期の点数を標準化したものを付値する」「t.phys1 に、z.phys1 を用いて物理の 1 学期の個々の偏差値を付値する」以上の作業を表す R の式を答えよ。

(3) s.phys2 には、物理の 2 学期の試験結果が、s.phys3 には、物理の 3 学期の試験結果が、すでに付値されている。次の出力を見て、さらに偏差値を計算して、出席番号 1、2、3 の生徒の点数が、1 年間でどのように変化したかを述べよ。

COMPUTER OPERATION ▷

```
> mean(s.phys2)↵
[1] 59.4
> sqrt(var(s.phys2))↵
[1] 7.13
> s.phys2[1:3]↵
[1] 55 63 52
> mean(s.phys3)↵
[1] 71.9
> sqrt(var(s.phys3))↵
[1] 14.5
> s.phys3[1:3]↵
[1] 75 80 60
```

— N O T E —

問 2 5 行 5 列の行列オブジェクト mat がある。○の位置を表す式を答えよ。

(1)

	1	2	3	4	5
1					
2		○	○		
3		○	○		
4		○	○		
5					

(2)

	1	2	3	4	5
1	○	○	○	○	○
2					
3					
4					
5	○	○	○	○	○

(3)

	1	2	3	4	5
1	○		○	○	
2	○		○	○	
3	○		○	○	
4	○		○	○	
5	○		○	○	

(4)

	1	2	3	4	5
1		○	○		○
2					
3		○	○		○
4		○	○		○
5					

問 3 5 行 5 列の行列オブジェクト mat において、以下の式で表される場所を示せ。

(1) `mat[2,]` (2) `mat[,3:5]` (3) `mat[c(2,5),c(1,3)]`

— 解答 —

文法を問うものは、以下に挙げるもの以外の解も存在する可能性がある。

問 1. (1) ア 40、イ 64.6、ウ 10.71

(2) `z.phys1 ← (s.phys1 - mean(s.phys1)) / sqrt(var(s.phys1))`、
`t.phys1 ← z.phys1 * 10 + 50`

(3) 偏差値は、1 学期は 65.32, 55.04, 36.36、2 学期は 43.83、55.05、39.62、3 学期は 52.14、55.59、41.79 となるはずである。「出席番号 1 の生徒は、大きく変化しながら成績降下」「出席番号 2 の生徒は、良く言えば現状保持、悪く言えば伸びていない」「出席番号 3 の生徒は、良くはないが着実な進歩が見られる」みたいなことが言えればよいであろう。

問 2 (1) `mat[2:4,2:3]` (2) `mat[c(1,5),]`
(3) `mat[,c(1,3,4)]`、`mat[,-c(2,5)]` (4) `mat[c(1,3,4),c(2,3,5)]`

問 3

(1)

	1	2	3	4	5
1					
2	○	○	○	○	○
3					
4					
5					

(2)

	1	2	3	4	5
1			○	○	○
2			○	○	○
3			○	○	○
4			○	○	○
5			○	○	○

(3)

	1	2	3	4	5
1					
2	○		○		
3					
4					
5	○		○		

— N O T E —

C.4 練習問題その 4

問 1 exe.kokugo、exe.sugaku、exe.rika、exe.shakai というベクトルオブジェクトは、生徒 16 人のテストの（架空の）点数データである。

(1) このデータを、「生徒の点数は行単位で格納され、1 列目が国語、2 列目が数学、3 列目が理科、4 列目が社会となる」行列オブジェクト exe.score に付値したい。どのようにすればよいか。

(2) 文字列ベクトルオブジェクト exe.subject に、「kokugo、sugaku、rika、shakai」の順で文字列が並ぶように付値せよ。

(3) 生徒の名前のデータが次のようにベクトルオブジェクト exe.name にある。

COMPUTER OPERATION ▷

```
> exe.name↵
 [1] "Iida"      "Abe"       "Yasuda"    "Yaguchi"   "Ishikawa"
   ⋮
```

exe.score の軸名札属性を、exe.name、exe.subject を用いてつけよ。

(4) 生徒の中に「Goto」「Fujimoto」「Matsuura」という者がいることはわかっているが、何番目かはわからない。exe.name や exe.score の全体を見ることなく、彼らの理科のテストの点数のみを取り出すには、どのように入力すればよいか。

(5) 次のような出力結果が得られた。

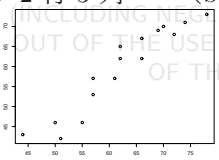
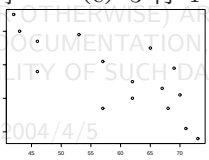
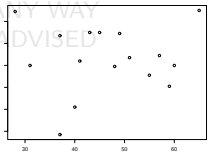
COMPUTER OPERATION ▷

```
> cor(exe.score)↵
          kokugo    sugaku      rika     shakai
kokugo  1.0000000 0.9496047 -0.85440590 0.16818495
sugaku  0.9496047 1.0000000 -0.77039305 0.13612781
rika   -0.8544059 -0.7703931  1.00000000 0.06278093
shakai  0.1681850 0.1361278  0.06278093 1.00000000
```

この行列は、(A) n 行 m 列の値と m 行 n 列の値が等しく、(B) n 行 n 列の値が 1 である。(A)、(B) の理由を簡潔に述べよ。

(6) 次の 3 つのグラフと、以下の (a) ～ (c) の cor(exe.score) の要素との対応をつけよ。

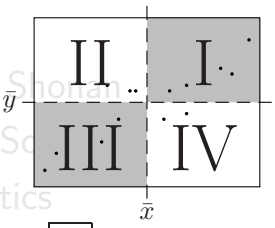
(a) 2 行 3 列　(b) 1 行 2 列　(c) 3 行 4 列

(A) 
(B) 
(C) 

— N O T E —

問 2 以下の文章は、相関係数についての説明の文章である。空らんにはいる言葉を答えよ。

相関係数は、2 つのデータの間にどのくらい直線関係があるかを判断するための一つの材料となる値である。
その求め方の基本的なアイデアは、次の通りである。1 つの値の組 (x_i, y_i) を点とみると、データは右の図のようなグラフになる。ここで、2 つのデータの平均 $(\bar{x}, \bar{y})$ を境に I から IV までの 4 つの領域にわけ、ある 1 点の配置を考えると、



- この点が I の領域にあれば、$(x_i - \bar{x})(y_i - \bar{y})$ の符号は［あ］、
- この点が II の領域にあれば、$(x_i - \bar{x})(y_i - \bar{y})$ の符号は［い］、
- この点が III の領域にあれば、$(x_i - \bar{x})(y_i - \bar{y})$ の符号は［う］、
- この点が IV の領域にあれば、$(x_i - \bar{x})(y_i - \bar{y})$ の符号は［え］、

となる。全体の影響を見たい時は、これらの値をすべて加えればよいが、その和が［お］ならば点は I と III に多く存在することになり、右上がりの傾向、すなわち［か］の相関が見られることになる。和が［き］ならば点は II と IV に多く存在することになり、右下がりの傾向、すなわち［く］の相関が見られることになる。和が 0 に近ければ、それぞれの領域にバランスよく散らばってることになり、相関は［　け　］ということになる。

— 解答 —

文法を問うものは、以下に挙げるもの以外の解も存在する可能性がある。

問 1（1）`exe.score ← matrix(c(exe.kokugo, exe.sugaku, exe.rika, exe.shakai),16,4)`
（2）`exe.subject ← c("kokugo", "sugaku", "rika", "shakai")`
（3）`rownames(exe.score) ← exe.name`、`colnames(exe.score) ← exe.subject`、
（4）`exe.score[c("Goto", "Fujimoto", "Matsuura"),"rika"]`
（5）（A）データが入れ替わっただけだから（B）同じデータの相関係数だから
（6）a–B、b–A、c–C

問 2
あ：正、い：負、う：正、え：負、お：正、か：正、き：負、く：負、け：ない

— N O T E —

C.5 練習問題その 5

数値を求める問題は、すべて小数点以下第 3 位を四捨五入すること。

問 1 ad.cost と sales は、それぞれある企業のある年度における宣伝広告費（単位 億円）と売上高（単位 十億円）の（架空の）データである。また、company.name は、それぞれの企業の名前である。データは次の通りであった。

COMPUTER OPERATION ▷

```
> ad.cost↵
 [1] 33 16 33 23 32 33 28 27 31 42 21 29 37 15 22 44 32 35 32 17
> sales↵
 [1]  85  40  78  62  70  40  63  63  80  99  54  72  87  36  50
[16] 101  67  73  83  38
```

(1) 以下の（A）～（H）の操作の解説を読み、あてはまる R への入力を答えよ。

（A） x 軸（横軸）が宣伝広告費、y 軸（縦軸）が売上高となり、点がそれぞれの企業を表す図を表示する。

（B） マウスで、それぞれの点をクリックしたときにそれぞれの企業名がでるようにする。

（C） （C）で指定したものの、実際の宣伝広告費の値を出力する。

（D） 宣伝広告費と売上高の相関係数を求める。

（E） 「宣伝を打てばそれだけ売上は伸びることが期待できる」という考えのもと、回帰分析（回帰方程式は $Y = A + BX$ の形）を行う。

（F） 回帰分析の結果の定数項 A と 1 次の係数 B の値を読みとる。

（G） 回帰方程式のあらわす直線をグラフィックス上に加える。

COMPUTER OPERATION ▷

```
> [    (A)    ]↵
> sel ← [    (B)    ]↵
> [    (C)    ]↵
[1] 33 33
> [    (D)    ]↵
[1] 0.8695709
> ls.result1 ← [    (E)    ]↵
> [    (F)    ]↵
  Intercept         X
```

— N O T E —

```
  5.569833 2.112721
> [ (G) ]↵
```

（2）（A）で描いた図を何というか。

（3）（B）で選んだ点の個数はいくつと考えられるか。

（4）（E）で、説明変数と被説明変数は何であると考えられるか。

（5） 宣伝広告費の割に売り上げがよくない企業が一つありそうである。残差の管理図を描き、どの企業かを調べてみた。以下の操作にあてはまるものを答えよ。

COMPUTER OPERATION ▷

```
> sel ← [ (H) ](ls.result1$[ (I) ])↵
> sel↵
 [1] 6
```

（6） よく調べてみると、この年この企業は不祥事を起こし、売り上げが伸び悩んだようである。そこで、この企業のデータを外して再度回帰分析を試みてみる。次の操作で、このデータを外したオブジェクト ad.cost.sel、sales.sel を作成せよ。

COMPUTER OPERATION ▷

```
> ad.cost.sel ← [ (J) ]
> sales.sel ← [ (K) ]
```

（7） ad.cost.sel, sales.sel を用いて回帰分析をやりなおした。このとき回帰方程式 $Y = A + BX$ の定数項 A と 1 次の係数 B はそれぞれ何になるか。また、相関係数 r を求めよ。

以下、この回帰方程式について答えよ。単位に気をつけること。

（8） 宣伝広告費が 30 億円の企業の売上高はいくらになると考えられるか。

（9） 480 億円の売上の企業がかけた宣伝広告費はいくらになると考えられるか。

— 解答 —

文法を問うものは、以下に挙げるもの以外の解も存在する可能性がある。

問 1（1）（A）plot(ad.cost, sales)
（B）identify(ad.cost, sales, labels = company.name)（C）ad.cost[sel]
（D）cor(ad.cost, sales)（E）lsfit(ad.cost, sales)（F）ls.result1$coef
（G）abline(ls.result1)（2）散布図（3）2 つ（4）説明変数は「宣伝広告費」、被説明変数は「売上高」（5）（H）identify（I）residual（6）（J）ad.cost[-sel]（K）sales[-sel]（7）$A = 4.00$、$B = 2.23$、$r = 0.96$（8）（A）709 億 4000 万円（B）19 億 7200 万円

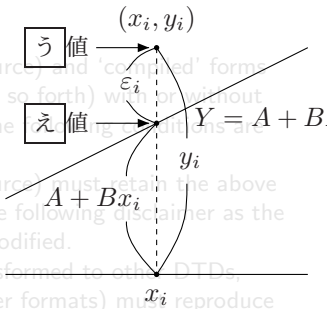
— N O T E —

C.6 練習問題その 6

問 1 以下の文章は、回帰分析についてと、その R 上での方法についての説明の文章である。空らんにはいる言葉を答えよ。

二つのデータの間に何らかの因果関係がある場合、$Y = f(X)$ のような式であらわすことができる。このとき、X にあたるものを［　あ　］変数、Y にあたるものを［　い　］変数という。特に $f(X)$ が一次式の場合（つまり $Y = A + BX$ の形になっているとき）、線型回帰という。

さて、回帰方程式を求めるには、適当に線を引いて定めるのではなく、実際のデータ（［　う　］値）y_i と直線上の値（［　え　］値）$A + Bx_i$ との差 $\varepsilon_i = y_i - (A + Bx_i)$ ができるだけ小さくなるように A と B を定めるようにする。この ε_i を［　お　］という。



1 点だけ考えるのならば、この直線が (x_i, y_i) を通るように A と B を定めればよいのだが、それでは他の点における［　お　］が大きくなってしまう可能性があるので意味がない。そこで、データ全体の［　お　］が小さくなることを考える。全体を考えるときはすべて足してしまうのが一番簡単であるが、特に［　お　］の平方和

$$\sum_{i=1}^{N} \varepsilon_i^2 = \sum_{i=1}^{N} \{y_i - (A + Bx_i)\}^2$$

が一番小さくなるように A と B を定める、［　か　］を用いることが多い。

R で［　か　］によって A と B の値を求める関数の名前は［　き　］である。この関数は A と B の値を計算してくれるが、A と B の値が確定するとそれに応じてすべてのデータに関しての［　お　］を計算することができる。［　き　］関数はこれも計算して返してくれるが、この値は「A と B の値」とは質が異なるため、行列やベクトルとしてまとめられない。R ではこのような質の異なるデータをまとめる形として、［　く　］というデータ形式がある。

— N O T E —

— 解答 —

問 1 あ：説明、い：被説明、う：実測値（観測値）、え：理論値、お：残差、か：最小二乗法、き：`lsfit`、く：リスト

※注意

ここにあげた問題はあくまで一つの例であり、たとえ記述問題であってもいろいろ問うべき題材はあります。他にも問うべき内容があるかもしれませんし、たとえここにあげた内容でも、問い方もいろいろ変化させることができます。自分の実力のチェックのために用いてください。

— N O T E —

C.7 練習問題その 7

問 1 ある会社の 20 か所の営業店舗とセールスマンの人数・広告費・売り上げの関係について調べたい。以下のようなファイルが用意されている。

- `salesdata.txt`··· 各行に「セールスマンの人数（人）・広告費（十万円）・売り上げ（値の解説は後述）」があるデータ
- `salespos.txt`··· 各行に営業店舗の地名がある文字列データ

(1) `salesdata` に、`salesdata.txt` から読み込んだデータを 20 行 3 列のデータとなるように付値せよ。

(2) `salesdata` の 3 列目は「売り上げ」のデータであるが、これは次のような構造をしている。

この営業店舗では A・B という 2 つの製品を売っていて、A は 1 つ 150 万円、B は 1 つ 60 万円である（消費税は考えない）。ここの値がたとえば 2.07 となっていたら、A が 2 つ、B が 7 つ売れたということを表している。

以上のことを用いて、それぞれの店舗の実際の売り上げ金額（百万円）が `salesdata` の 3 列目になるようにするにはどうすればよいか。

(3) `salespos` に、`salespos.txt` から読み込んだ文字列をベクトルとして付値せよ。

(4) `salesdata` の軸名札属性として、行の名札に「地名」、列の名札に「`salesman`」「`ad`」「`sales`」を付値せよ。

(5) これらの営業店舗の「セールスマンの人数」「広告費」「売り上げ」の平均を一気に求めるには、どうすればよいか。

(6) 「広告費をかけることにより売り上げが伸びる」と考えて、単回帰モデルの構築を行う。

- 広告費と売り上げの相関係数を求め、
- 説明変数・被説明変数が何かを考えて散布図を描き、
- 回帰分析を行い（結果は `ls.result1` に付値する）、
- 回帰直線を散布図に加え、
- 回帰方程式を求める

方法をあらわせ。

(7) `ctlchart()` 関数を用いて、「残差の管理図」を描き、外れ値と思われる番号を `outlier` に付値する作業は、どのようにすればよいか。

— N O T E —

(8) 前問の outlier で示されるデータを削除して（削除したあとの「広告費」「売り上げ」データを、それぞれ ad.sel と sales.sel に付値する）もう一度上記の回帰分析を行うにはどうすればよいか。回帰分析の結果は ls.result2 に付値するものとする。

(9) 「セールスの人材投資も売り上げに影響する」と考え、「セールスマンの人数」を X_1、「広告費」を X_2 としたときの回帰方程式を求めるにはどうすればよいか。回帰分析の結果は ls.result3 に付値するものとする。

(10) この結果を用いて、「セールスマンを 12 人」「広告費を 82 万円」という投資を行う営業店舗を新たに作ると、そこの売り上げ（万円）を予測する計算をする式を答えよ。

— 解答 —

問 1 (1) salesdata ← matrix(scan("salesdata.txt"), 20, 3, byrow=TRUE)
(2) a ← floor(salesdata[,3]), b ← salesdata[,3] - a,
salesdata[,3] ← (a*150 + b*100*60)/100
(3) salespos ← scan("salespos.txt","")
(4) rownames(salesdata) ← salespos, a ← c("salesman", "ad", "sales")
colnames(salesdata) ← a　　(5) apply(salesdata, 2, mean)
(6) cor(salesdata[,2], salesdata[,3]), plot(salesdata[,2], salesdata[,3]),
ls.result1 ← lsfit(salesdata[,2], salesdata[,3]), abline(ls.result1),
ls.result1$coef を調べる
(7) ctlchart(ls.result1$residual),
outlier ← identify(ls.result1$residual, labels=salespos)
(8) ad.sel ← salesdata[-outlier,2], sales.sel ← salesdata[-outlier,3],
cor(ad.sel, sales.sel), plot(ad.sel, sales.sel), ls.result2 ← lsfit(ad.sel, sales.sel), abline(ls.result2), ls.result2$coef を調べる
(9) ls.result3 ← lsfit(salesdata[,1:2], salesdata[,3])
(10) (ls.result3$coef[1] + ls.result3$coef[2]*12 + ls.result3$coef[3]*8.2)*100

※なお、このデータは

J:¥R¥local¥2003¥salesdata.txt

J:¥R¥local¥2003¥salespos.txt

として実際に用意してあります。ためしてみる場合は、ファイル名を上記のものにしてやってみてください。なお、R 上では、ファイル名の ¥は/にかえます。

また、自宅で再現してみる場合は、実習用データにこのデータがありますので、特に変更なくできるはずです。

— N O T E —

データ解析ワークブック（2003 年度版）

執筆・編集 慶應義塾湘南藤沢中等部・高等部 数学科 ／ 今橋 貞二