R入門

基礎生物学研究所

ゲノムインフォマティクストレーニングコース

内山　郁夫　**(uchiyama@nibb.ac.jp)**

Rとは

- ベル研究所で開発された統計処理言語「S」を基に、フリーソフトとして開発された**統計解析環境・プログラミング言語**
- **コマンドラインインターフェイス**が基本。対話的なコマンド実行による解析のほか、プログラム（スクリプト）を書いて一括処理を行うことも可能
- **ベクトル・行列演算**が簡便かつ効率的に行える
- 独自の**関数**を作成することによって機能拡張が可能
- 作成した関数等を**パッケージ**単位でまとめることにより、機能拡張が容易に行える。様々なパッケージが公開されており、これらを導入することによって、最先端の統計手法を用いることができる。

作業ディレクトリの設定

- **作業ディレクトリ：　読み込むデータファイルや結果を書き出すファイルを保存するディレクトリ。**
- **メニューから、その他→作業ディレクトリの変更をえらんでディレクトリを選択する。**

作業ディレクトリ　~/data/3_R　に移動

- 作業ディレクトリを常に固定したい場合は、環境設定から起動タブを開いて初期ディレクトリを設定する。
- 作業ディレクトリをコマンドで設定することも可能

  `setwd("ディレクトリ名")`　作業ディレクトリの設定

  `getwd()`　作業ディレクトリの確認

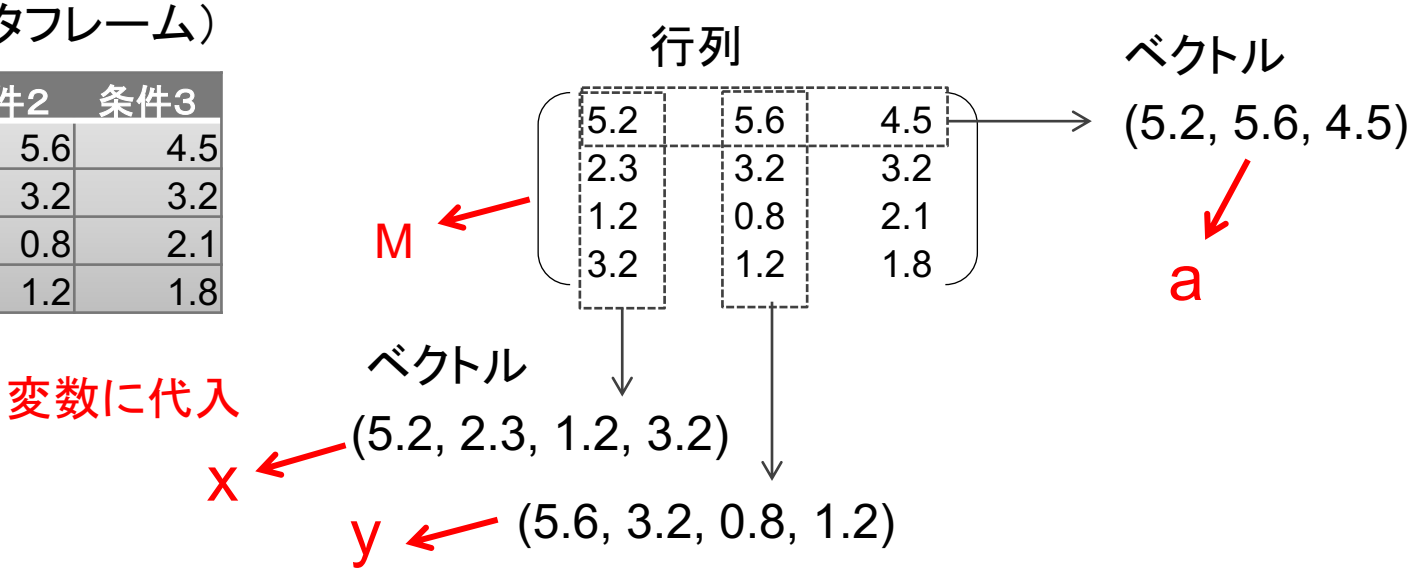
R入門

1. 基本演算

Rにおけるデータ処理の概要（ベクトルと行列）

表データ （データフレーム）

	条件1	条件2	条件3
遺伝子1	5.2	5.6	4.5
遺伝子2	2.3	3.2	3.2
遺伝子3	1.2	0.8	2.1
遺伝子4	3.2	1.2	1.8



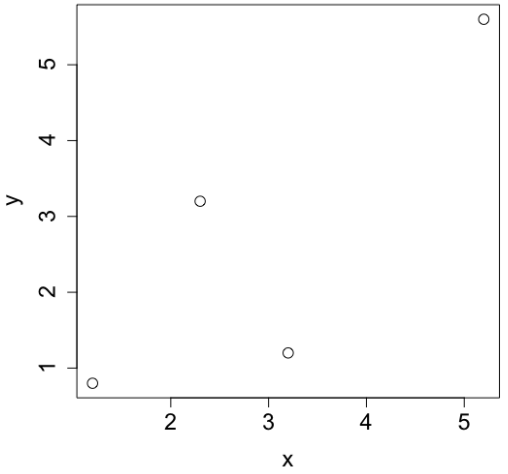
ベクトル演算

x / y

$$\left(\frac{5.2}{5.6}\ \frac{2.3}{3.2}\ \frac{1.2}{0.8}\ \frac{3.2}{1.2}\right)$$

プロットの作成

plot(x, y)



スカラー演算

```
> 1+3
[1] 4
> 1+3*5
[1] 16
> a <- 1+3*5
> a
[1] 16
> a > 10
[1] TRUE
```

通常の通り、*（かけ算）は+（足し算）より優先する

結果を変数 a に代入する

a とだけタイプすると、変数aの内容が出力される

論理演算は、論理値TRUEまたはFALSEを返す

変数と代入

- **計算結果を変数に代入することにより、再利用できる**
  - 変数名にはアルファベット、数字、'.'（dot）、'_' (underscore)が利用できる。ただし，先頭はアルファベット。
  - 大文字と小文字は区別される。
- **代入を表す演算子には、<- ＝ -> の3通りがある。**

以下の3つはいずれもａに４が代入される。

```
> a <- 1 + 3
> a = 1 + 3
> 1 + 3 -> a
```

ヒストリー（履歴）

- **コンソール上で、上下の矢印キー（↑↓）によて、コマンドの履歴を前後にたどることができる。**
- **左右の矢印キー（←→）によって、カーソルを左右に動かせる。これによってコマンドの編集ができる。**
- **以下のコントロールキーを使った操作も可能**
  - Control+P 履歴を前に移動（↑と同じ）
  - Control+N 履歴を後ろに移動（↓と同じ）
  - Control+B カーソルを左に移動（←と同じ）
  - Control+F カーソルを右に移動（→と同じ）
  - Control+A カーソルを行の先頭に移動
  - Control+E カーソルを行の最後に移動
- **GUIからヒストリーパネルを使って履歴をたどることも可能**

ヒストリーパネルの表示・非表示

ベクトル

```
> a <- c(1, 3, 7, 4, 6)
> a
 [1] 1 3 7 4 6
> b <- 3:7
> b
 [1] 3  4  5  6  7
> length(a)
 [1] 5
> b[3]
[1] 5
```

ベクトルは関数 c を用いて作成する

3から7までの連続した整数

ベクトルaの長さ(要素数)

bの3番目の要素

ベクトル演算

a=c(1,3,7,4 6); b=c(3,4,5,6,7) と設定されている

```
> 1 + a
[1] 2 4 8 5 7
```
aの各要素に1を加える

```
> 2 * a
[1]  2  6 14 8 12
```
aの各要素を2倍する

```
> a + b
[1]  4  7 12 10 13
```
aとbの要素ごとの和をとる

```
> a * b
[1]  3  12 35 24 42
```
aとbの要素ごとの積をとる

```
> a > 3
[1] FALSE FALSE TRUE  TRUE  TRUE
```
aの要素ごとに3より大きいか比較する

```
> a < b
[1] TRUE TRUE FALSE TRUE TRUE
```
aとbを要素ごとに大小を比較する

基本統計量の計算

a=c(1,3,7,4,6)と設定されている

```
> length(a)
> sum(a)
> mean(a)
> median(a)
> var(a)
> sd(a)
```

aの長さ(要素数)
aの要素の合計値
aの要素の平均値
aの要素の中央値
aの要素の不偏分散
aの要素の標準偏差

問題） mean(a)をsum(a)とlength(a)を使って計算してみよう。

ベクトル要素の抽出

a=c(1,3,7,4,6)と設定されている

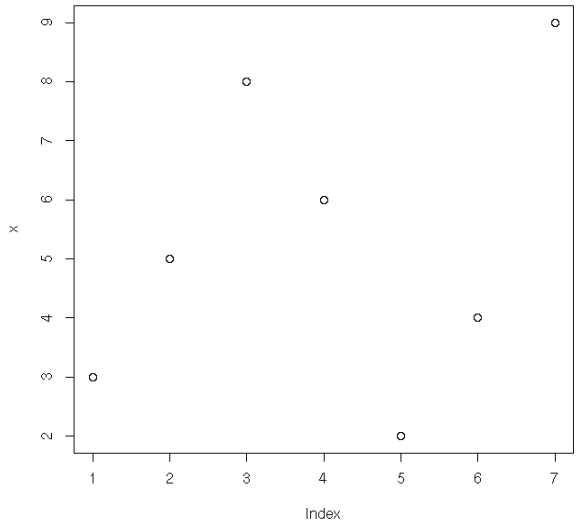
```
> a[2:4]                    # 2番目から4番目までの要素
[1] 3 7 4
> a[ c(3,5,2) ]# 3, 5, 2番目の要素
[1] 7 6 3
> a[ c(T,T,F,F,T) ]  # T(TRUE)である要素だけ出力
[1] 1 3 6
> a[ a > 3 ]                # a>3がTRUEである要素だけ出力
[1] 7 4 6
```

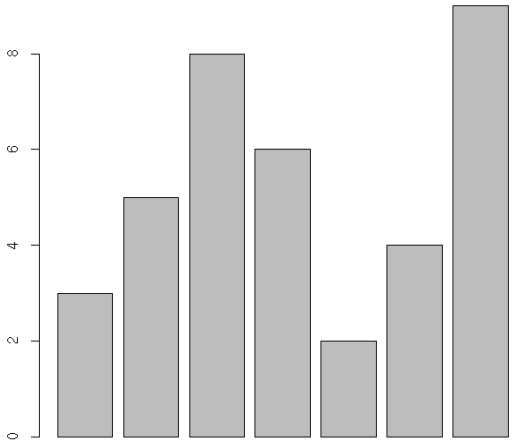
並べかえ（ソート）

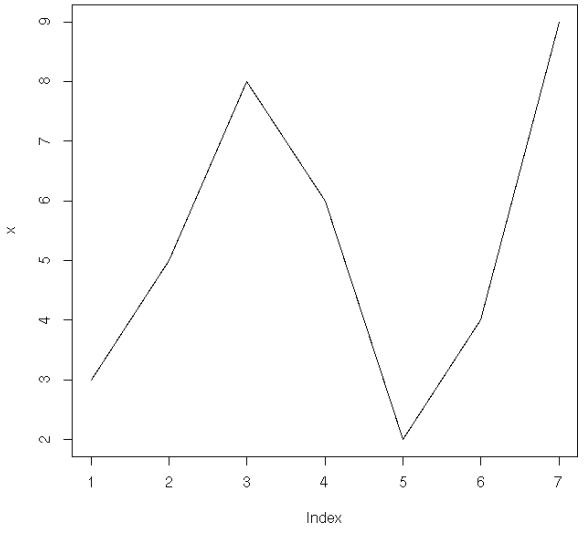
```
> x <- c(3,5,8,6,2,4,9)
> sort(x)                          # 小さい順に並べかえ
[1] 2 3 4 5 6 8 9
> sort(x, decreasing=TRUE)  # 大きい順
[1] 9 8 6 5 4 3 2
```

プロットの作成(1)

```
> x
[1] 3 5 8 6 2 4 9
> plot(x)
> plot(x,type="l")
> barplot(x)
```







プロットの作成(2)

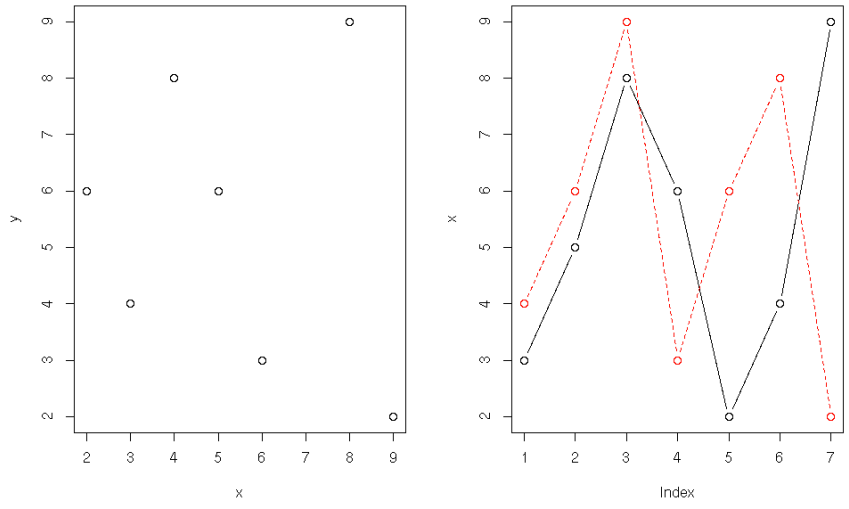
```
# x <- c(3,5,8,6,2,4,9)と設定されている
> y <- c(4,6,9,3,6,8,2)

> par(mfrow=c(1,2))   # 2つのプロットを表示
# x,y をx軸、y軸にとって散布図としてプロット

> plot(x,y)
# x,yを別々にプロット。まずxをプロット。点と線を両方描く。

> plot(x,type="b")
# yを重ねてプロット。linesは枠を描き直さずに線だけを引く。
# lty はline type, col は colorを指定。

> lines(y, type="b", lty=2, col=2)
```



Rにおける基本データ型

- **Rの「原子データ型」として以下のものがある**
  - 数値型 numeric（実数または整数）
  - 論理型 logical（TRUE/FALSE)
  - 文字列型 character("abc"、"123" のように、二重引用符で囲まれた文字列として表す）
- **Rの原子データはベクトルである。スカラー値も長さ1のベクトルである。すなわち、ベクトルは型を持ち、すべての要素は同じ型のデータからなる。**
- **異なる型のデータを集めた構造として「リスト」がある。**
- **mode(x) によって、変数 x の型を調べられる**

演算子

- **算術演算子**

  +（加算）–（減算）*（乗算）/（除算），

  %/%（整数除算） %%（剰余） ^（累乗）

  例） 5/2 (=2.5) 5%/%2 (=2) 5%%2 (=1)

- **論理演算子**

  &（論理積「かつ」） |（論理和「または」） !a（aの否定）

- **比較演算子**

  >, >=, <, <=（不等号）==（等しい）!=（等しくない）

これらの演算子はベクトルの要素ごとにはたらく

（例）a=c(1,3,7,4,6)と設定されている

```
> a >= 2 & a < 5
[1] FALSE  TRUE  FALSE  TRUE FALSE
```

ワークスペースとオブジェクト

- コンソール上で ls() または objects()とすると、変数に保存されたデータ(オブジェクト)のリストを参照できる。

```
>ls()

[1] "a"  "b"  "x"
```

- ワークスペースブラウザ(メニューバー「ワークスペース」から起動)でより詳細な情報を閲覧可能
- rm(変数名)で、オブジェクトを消去できる。

関数

関数名(引数1, 引数2, ...)

- 関数は、一般に複数の引数を入力としてとり、何らかの計算を行って一つのオブジェクトを返す(戻り値)。
- 引数には、必須のものと省略可能なものがある。後者は省略すると「デフォルト値」が使用される。
- 引数は、順番によって指定する方法と、「名前＝値」の形式で指定する方法がある。通例、必須の引数は前者、選択可能な引数は後者で指定する。

例1) ベクトル x を小さい順(increasing order)でソートする

```
sort(x)
```

例2) ベクトル x を大きい順 (decreasing order)でソートする

```
sort(x, decreasing=TRUE)
```

マニュアルの表示

- **help(関数名)または ?関数名　でマニュアルを表示する**

```
> help(median)
```

median　　package:stats　　R Documentation

Median Value

Description:

Compute the sample median.

Usage:

median(x, na.rm = FALSE)

Arguments:

x: an object for which a method has been defined, or a numeric vector containing the values whose median is to be computed.

na.rm: a logical value indicating whether 'NA' values should be stripped before the computation proceeds.

Details:

This is a generic function for which methods can be written. However, the default method makes use of 'sort' and 'mean' from package 'base' both of which are generic, and so the default method will work for most classes (e.g. '"Date"') for which a median is a reasonable concept.

Value:

The default method returns a length-one object of the same type as 'x', except when 'x' is integer of even length, when the result will be double.

If there are no values or if 'na.rm = FALSE' and there are 'NA' values the result is 'NA' of the same type as 'x' (or more generally the result of 'x[FALSE][NA]').

References:

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) _The New S Language_. Wadsworth & Brooks/Cole.

See Also:

'quantile' for general quantiles.

Examples:

```
median(1:4)# = 2.5 [even number]
median(c(1:3,100,1000))# = 3 [odd, robust]
```

Description: 関数の簡単な説明

Usage: 関数の呼び出し方

median(x, na.rm = FALSE)

x: 必須の引数

na.rm: 省略可能な引数

デフォルト値はFALSE

Arguments: 各引数の詳しい説明

Details: 関数の動作の詳しい説明

Values: 戻り値の説明

Reference:方法に関する文献

See Also: 関連するコマンド

Examples: 実行例

（参考） plotのオプション

- `main="title"`　グラフのタイトル
- `xlab(ylab)="label"`　x軸(y軸)のラベル
- `log="xy"`　対数軸の指定
- `xlim(ylim)=c(0,100)` x軸(y軸)の値の範囲の設定
- `type="l"`　`"p"`(点), `"l"`(線), `"b"`(両方), `"n"`(枠だけ) など
- `lty=1`　プロットする線の種類
- `pch=1`　プロットする点の種類(文字)
- `col=2`　プロットする点および線の色
- `cex=0.8` プロットする文字の大きさ

ベクトルとして指定することにより、点ごとに色や種類を変更することもできる。例)`pch=c(1,2,1,3,2)`

										
pch	1	2	3	4	5	6	7	8	9	10
col	1	2	3	4	5	6	7	8	9	10
cex	0.25	0.5	0.75	1	1.25	1.5	1.75	2	2.25	2.5

- ✧ plotのオプションは,、`help(plot)`だけでは調べられない。一部のオプションは`help(plot.default)`, `plot(par)`を参照する必要がある。
- ✧ 既存のグラフ上に線を重ねる場合は`lines`, 点を重ねる場合は`points`を使う。
- ✧ 凡例をつけたいときは`legend`関数を使う。

宿題の確認

```
> 0:20
```
0から20までの整数を並べたベクトル(0, 1, 2, ..., 20)

```
> 0:20/20
```
上のベクトルを20で割ったもの(0, 1/20, 2/20, ..., 1)

```
> 0:20/20*pi
```
上のベクトルに円周率πをかけたもの(0, π/20, 2π/20, ..., π)

```
> sin(0:20/20*pi)
```
上のベクトルの各要素のサインをとったもの(0, sin(π/20), ... sin(π))

```
> plot(sin(0:20/20*pi))
```
上のベクトルの各要素の値を順にY軸にとってプロットしたもの

```
> plot(sin(0:20/20*pi), type="l")
```
上のプロットを線でつないだもの (0≤x≤π の範囲のサインカーブ)

エディタからのコマンドの入力と実行

- メニューから　ファイル→新規文書　でエディタが開く
- Rのコマンド群を入力（複数行にまたがってよい）

- マウスで実行するコマンド群を選択して　Command+Return　を押すと、そのコマンド群がコンソール上にコピーされて実行される

行列の作成

一つのベクトルを行列の形に並べる

```
> v <- 1:9
> m1 <- matrix(v, 3, 3)
> m1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

9つの要素を持つベクトル

これを3行3列の行列として定義

複数のベクトルを行または列として束ねる

```
> a <- c(1,3,5)
> b <- c(2,4,9)
> c <- c(3,5,7)
> m2 <- cbind(a,b,c)
> m2
     a b c
[1,] 1 2 3
[2,] 3 4 5
[3,] 5 9 7
```

3つの要素を持つベクトル3つ

これらを列（縦）に並べて行列とする

行列の要素の取り出し

```
> m1
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> m1[2,3]
[1] 8
> m1[2,]
[1] 2 5 8
> m1[,3]
[1] 7 8 9
> m1[1:2,2:3]
     [,1] [,2]
[1,]    4    7
[2,]    5    8
> dim(m1)
[1] 3 3
```

2行3列目の要素

2行目の要素すべてをベクトルとして取り出す

3列目の要素すべてをベクトルとして取り出す

1,2行目と2,3列目の要素からなる
部分行列を取り出す

行列の大きさ(3行3列)

行列の演算

```
m1 =
 1 4 7
 2 5 8
 3 6 9

m2 =
 1 2 3
 3 4 5
 5 9 7
```

```
> m1+m2
     a  b  c
[1,] 2  6 10
[2,] 5  9 13
[3,] 8 15 16
> m1*m2
      a  b  c
[1,]  1  8 21
[2,]  6 20 40
[3,] 15 54 63
> m1 %*% m2
      a   b   c
[1,] 48  81  72
[2,] 57  96  87
[3,] 66 111 102
> t(m1)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

要素ごとの足し算

要素ごとのかけ算

行列積

転置行列（行と列の入れ換え）

データフレーム

- Rの統計解析でもっとも基本的となるデータ。
- 行列のような表形式のデータだが、各列が一般に異なる型のデータを持ちうる（行列はすべてが同じ型のデータ）。
- 行や列に名前がつけられることが多い。
- 通常、タブ区切りテキストやエクセルファイル等から読み込んで処理する。
- 標準で利用可能なサンプルデータがある。

```
> data()

... 利用可能なサンプルデータの表示

> women
   height weight
1      58    115
2      59    117
3      60    120
4      61    123
5      62    126
...
> plot(women)
height と weight 間のプロット
```

データフレームの要素の取り出し

- 行の取り出し。3行目を取り出す。

```
> women[3,]
```

- 身長の列（1列目）の取り出し　以下の3つは同じ結果を返す

```
> women[,1]
> women[,"height"]
> women$height
```

- 身長が65インチ以上かの判定

```
> women[,1] >= 65
```

（8行目から15行目がTRUE）

- 身長が65インチ以上のデータを取り出す

```
> women[ women[,1]>=65, ]
```

（8行目から15行目のデータを取り出す）

R入門
2. 統計解析

Rにおけるデータ処理の概要（統計解析）

表データ
(エクセル・タブ区切りテキスト)

	条件1.1	条件1.2	条件2.2	条件2.2
遺伝子1	5.2	5.6	4.5	3.0
遺伝子2	2.3	3.2	3.2	3.5
遺伝子3	1.2	0.8	2.1	2.5
遺伝子4	3.2	3.1	1.8	2.0

読込み

read.table()

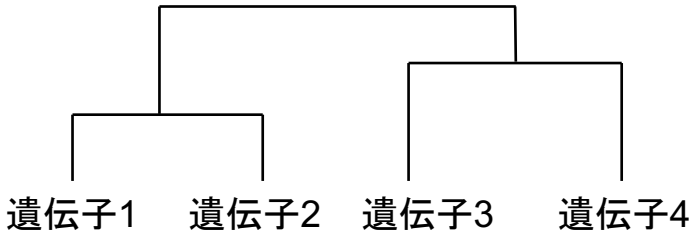
データフレーム

	条件1.1	条件1.2	条件2.2	条件2.2
遺伝子1	5.2	5.6	4.5	3.0
遺伝子2	2.3	3.2	3.2	3.5
遺伝子3	1.2	0.8	2.1	2.5
遺伝子4	3.2	3.1	1.8	2.0

統計的検定

	条件1		条件2	
遺伝子4	3.2	3.1	1.8	2.0

多変量解析
（クラスタリング、主成分分析等）



・有意差検定　(t検定、ノンパラメトリック検定等)
・線形モデル　（回帰分析、分散分析）
　　発現量　～　性別　＋　病態

データフレーム lung_cancer

データフレーム
Rの統計解析で基本となるデータ。
行列のような表形式のデータだが、各列が一般に異なる型のデータを持ちうる

lung_cancer.txt
肺がん患者の遺伝子発現プロファイルデータ（GEO: GDS3257の抜粋）

tissue: 腫瘍or正常
smoking: 喫煙歴
stage: 癌のステージ
gender: 性別
gene1 – gene6: 遺伝子発現データ

ID_REF	tissue	smoking	stage	gender	gene1	gene2	gene3
GSM254629	tumor	never	stage I	female	7.4191	5.9318	5.67496
GSM254648	tumor	never	stage I	female	7.5627	6.93398	5.76701
GSM254694	tumor	never	stage I	female	7.54599	7.53287	5.84134
GSM254701	tumor	never	stage I	female	8.31452	7.88291	5.44759
GSM254728	tumor	never	stage I	female	7.19835	6.58398	4.79089
GSM254726	tumor	never	stage I	male	11.9811	8.45595	5.7083
GSM254639	tumor	never	stage II	female	7.41762	7.75681	4.74084
GSM254652	tumor	never	stage II	female	7.62703	7.7446	4.69937
GSM254700	tumor	never	stage II	female	7.40064	10.1866	4.92612
GSM254625	tumor	never	stage II	male	11.9	8.89865	6.69416
GSM254636	tumor	never	stage III	female	7.09852	6.39122	4.54743
GSM254659	tumor	never	stage III	female	7.39159	7.06924	5.12113
GSM254680	tumor	never	stage III	female	7.32462	7.24284	4.90953
GSM254686	tumor	never	stage III	female	7.65883	7.93856	5.09535
GSM254718	tumor	never	stage III	female	7.67937	7.03277	5.64789
GSM254674	tumor	never	stage IV	male	11.0711	6.3368	5.48673
GSM254668	tumor	former	stage I	male	10.9011	6.52462	5.19564

データフレームの読込み(1)

```
read.table(file, options... )
read.delim(file, options... )
```

Options: sep=列の区切り文字（タブ、空白文字など）
header=ヘッダの有無(TRUE/FALSE)
row.names=各行の名前を何カラム目からとるか　など

タブ区切りテキストファイル lung_cancer.txt からデータの読み込み
区切り文字はタブ（￥t）、ヘッダ行あり、1列目をヘッダ列として指定

```
> cancer <- read.table("lung_cancer.txt", sep="¥t", header=T,
          row.names=1)

> head(cancer)
```

データの先頭数行を表示して内容を確認

```
          tissue smoking   stage gender    gene1   gene2 ...
GSM254629  tumor  never  stage I female  7.41910 5.93180 ...
GSM254648  tumor  never  stage I female  7.56270 6.93398 ...
GSM254694  tumor  never  stage I female  7.54599 7.53287 ...
GSM254701  tumor  never  stage I female  8.31452 7.88291 ...
...
```

データフレームの読込み(2)

ファイルの形式

ヘッダなし		ヘッダ行あり		ヘッダ行、ヘッダ列あり			ヘッダ行、ヘッダ列あり
198	119	Height	Weight	ID	Height	Weight	
192	83	198	119	1	198	119	1 198 119
191	110	192	83	2	192	83	2 192 83
191	91	191	110	3	191	110	3 191 110
		191	91	4	191	91	4 191 91

先頭行のみ要素数が一つ少ない

read.table オプション

`header=FALSE`

`header=TRUE`

`header=TRUE, row.names=1`

（オプションなしで自動認識）

`read.table` が読み込みのための基本的な関数だが、デフォルト値の異なる読込み関数がいくつか用意されている

`read.table(file)`	デフォルトでセパレータまたは空白文字、header=FALSE
`read.delim(file)`	デフォルトでセパレータがタブ、header=TRUE
`read.csv(file)`	デフォルトでセパレータがコンマ、header=TRUE

データフレームの情報表示

str(data)	data のデータ構造の表示
summary(data)	data の内容サマリの表示

```
> str(cancer)
```

各列のデータ型と内容の一部を表示

```
'data.frame':    107 obs. of  10 variables:
 $ tissue : Factor w/ 2 levels "normal","tumor": 2 2 2 2 2 2 2 2 2 2 ...
 $ smoking: Factor w/ 3 levels "current","former",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ stage  : Factor w/ 4 levels "stage I","stage II",..: 1 1 1 1 1 1 2 2 2 2 ...
 $ gender : Factor w/ 2 levels "female","male": 1 1 1 1 1 2 1 1 1 2 ...
 $ gene1  : num  7.42 7.56 7.55 8.31 7.2 ...
 $ gene2  : num  5.93 6.93 7.53 7.88 6.58 ...
 .....
```

```
> summary(cancer)
```

各列のデータの分布の要約を表示

```
 tissue      smoking         stage       gender       gene1             gene2
normal:49   current:40   stage I  :45   female:38   Min.   : 6.729   Min.   : 5.728
tumor :58   former :36   stage II :35   male  :69   1st Qu.: 7.543   1st Qu.: 6.761
            never  :31   stage III:21               Median :11.224   Median : 8.452
                         stage IV : 6               Mean   :10.074   Mean   : 8.894
                                                    3rd Qu.:11.875   3rd Qu.:11.130
                                                    Max.   :12.659   Max.   :12.293
```

因子 factor

- **gender (male, female)のように、限られた数のカテゴリーで表現されるもの。カテゴリー変数、要因などともいう。**
- **因子の取り得る値を水準levelという。**

  例）genderは2つ(female, male)、smokingは3つ(current, former, never)の水準を持つ
- **文字列として表示されるが、内部的には整数値で保持されている。**

データフレームの要素の取り出し

```
> cancer[5,]
          tissue smoking   stage gender   gene1   gene2   gene3   gene4   gene5   gene6
GSM254728  tumor   never stage I female 7.19835 6.58398 4.79089 7.26575 7.46492 5.02376
```

5行目の人のデータの取り出し

```
> cancer[,5]
  [1]  7.41910  7.56270  7.54599  8.31452  7.19835 11.98110  7.41762  7.62703
  [9]  7.40064 11.90000  7.09852  7.39159  7.32462  7.65883  7.67937 11.07110
 [17] 10.90110  9.30280 11.42620 12.10840 11.00710 11.43570  9.58576 11.37820
 [25] 11.46580 11.98080  7.17584 11.62510 11.22410  7.45557  7.70254 11.65150
 [33] 11.01420 12.03040  7.00001  7.53943 10.81220 11.47220 12.05180 11.37000
 ....
```

5列目（gene1）の取り出し

```
> cancer$gene1
```

gene1 データの取り出し（名前によるアクセス。cancer[,5]と同じ）

```
> cancer["GSM254728",]
```

行の取り出し（名前によるアクセス。cancer[5,]と同じ）

```
> cancer[7:10,2:7]
          smoking     stage gender    gene1    gene2   gene3
GSM254639   never stage II female  7.41762  7.75681 4.74084
GSM254652   never stage II female  7.62703  7.74460 4.69937
GSM254700   never stage II female  7.40064 10.18660 4.92612
GSM254625   never stage II   male 11.90000  8.89865 6.69416
```

7-10行、2-7列の要素からなる部分データの取り出し

データフレームの操作
条件式による部分データの抽出

subset(データフレーム, 条件式)

```
# 性別が女性でgene1の発現が8以上
> subset(cancer, gender=="female" & gene1 > 8)
          tissue smoking     stage gender    gene1   gene2   gene3   gene4   gene5   gene6
GSM254701  tumor   never   stage I female  8.31452 7.88291 5.44759 5.99769 7.66485 5.06010
GSM254687  tumor current stage III female 10.15150 7.64551 6.61338 7.24318 7.91872 7.36862
# 喫煙歴がある人（過去または現在に） 結果を変数に保存
> cancer.smoking <- subset(cancer, smoking %in% c('former','current'))


# その数（行数をカウント）
> nrow(cancer.smoking)
[1] 76
```

A %in% B　ベクトルAの各要素について、ベクトルBの要素のいずれかと一致すればTRUE,そうでなければFALSEを返す。

```
> c("A","B","C","B","A") %in% c("B","C")
[1] FALSE  TRUE  TRUE  TRUE FALSE
```

データフレームのファイルへの書き出し

```
write.table(data, file="", options... )
```

```
# タブ区切りテキストとしてファイルに保存
> write.table(cancer.smoking, sep="¥t",
    file="cancer.smoking.txt")

# そのまま読み込んで表示してみる
> read.table("cancer.smoking.txt")
```

Rによるデータ解析
plotによる散布図の作成

各カラム対総当たりの散布図を作成する

```
> plot(cancer)
```

pairs(cancer)としても同じ

gene3とgene4の散布図を作成する

```
> plot(cancer$gene3, cancer$gene4)
```

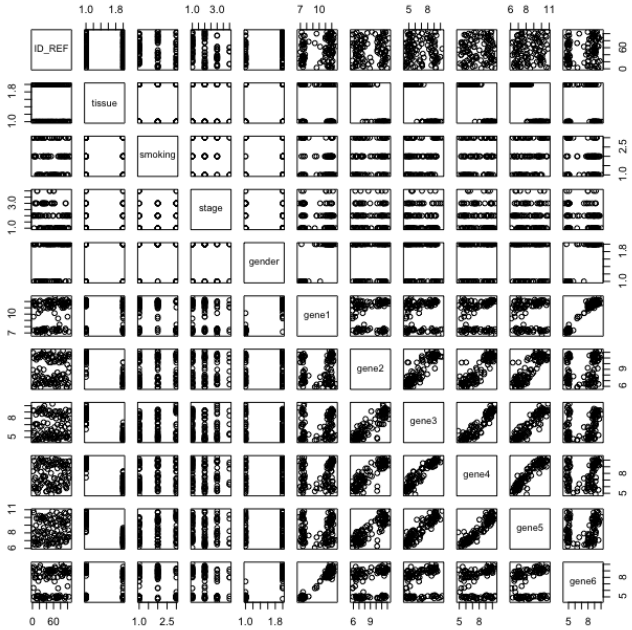
以下もほぼ同じプロットを生成する

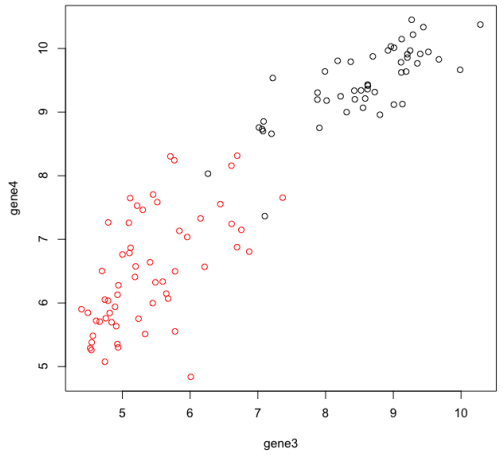
```
> plot(gene4 ~ gene3, cancer)
```

tissue (cancer/normal) によって点の色づけする

```
> plot(gene4 ~ gene3, cancer,
    col=tissue)
```

cancer$tissue は normal=1, tumor=2 として定義されており、各点が 1=black, 2=redで色づけされる。





Rによるデータ解析
回帰直線の追加

gene3とgene4の散布図（前掲）

```
> plot(gene4 ~ gene3, cancer)
```

線型モデルを用いた直線への当てはめ（回帰直線の決定）

```
> lm(gene4 ~ gene3, cancer)
```

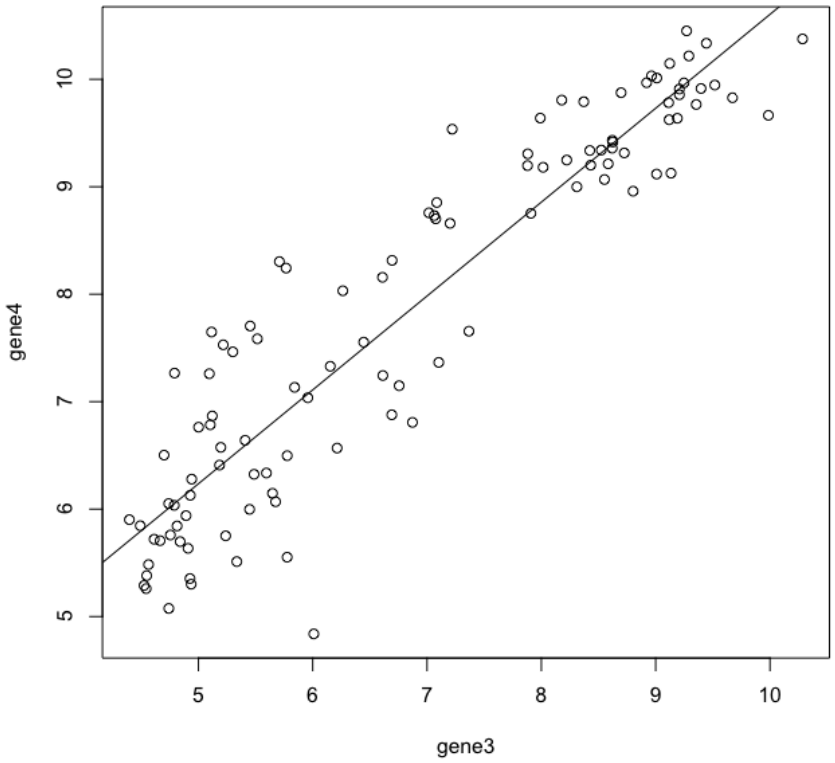
...（結果は画面に出力される）...

結果を変数 result.lm に代入

```
> result.lm <- lm(gene4 ~ gene3,
        cancer)
```

この結果を使って回帰直線をプロットに追加

```
> abline(result.lm)
```



モデル式と線型モデル

- **lm の第1引数で指定される式は、データを当てはめるモデル式を簡潔に表現したものとなっている**

  例） X1,X2を説明変数、Yを目的変数として、

  Y = a * X1 + b * X2 + c + ε（残差）

  という式に当てはめる（残差が小さくなるよう係数a,b,cの最適な値を決める）ことを

  Y ~ X1 + X2

  と記述する。

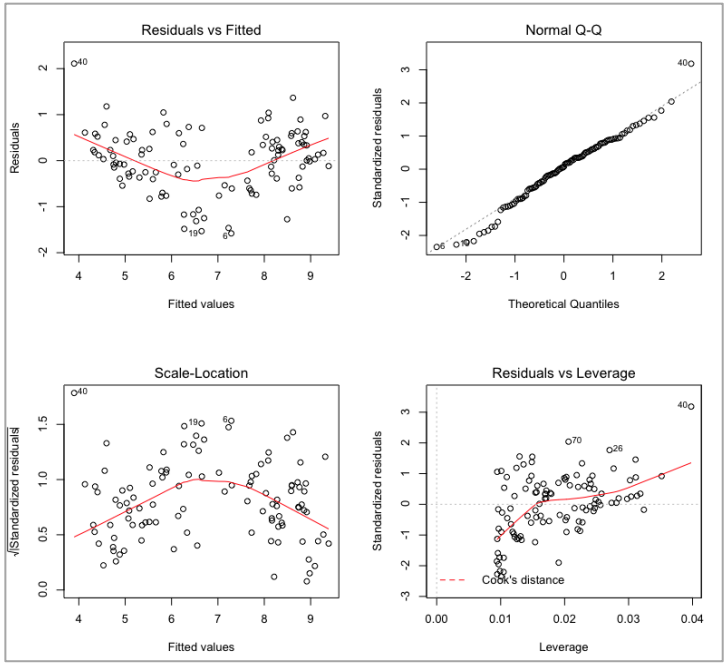
- **モデル式が係数の一次式で表されるモデルを線型モデルといい、最小二乗法で当てはめが行われる。lm関数では、係数の決定と同時に、係数が有意に0でないといえるかの検定も行われる。**

- **線型モデルへの当てはめは、回帰分析だけでなく、カテゴリ変数を説明変数とした分散分析においても用いられる。**

Rによるデータ解析
回帰分析結果の詳細表示

```
> par(mfrow=c(2,2))                    # 4つのプロットを 2x2 の領域に同時に表示
> plot(result.lm)   # 解析結果からプロットの作成
```

プロットから誤差(残差)の分布が適切か(正規分布に従っているか)を確認




```
> summary(result.lm) # 解析結果の詳細表示
Call:
lm(formula = gene4 ~ gene3, data = cancer)

Residuals:
     Min       1Q   Median       3Q      Max
-2.27983 -0.41889 -0.01643  0.45094  1.44834

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.86604    0.24801   7.524  1.9e-11 ***
gene3        0.87403    0.03514  24.870  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6395 on 105 degrees of freedom
Multiple R-squared:  0.8549,    Adjusted R-squared:  0.8535
F-statistic: 618.5 on 1 and 105 DF,  p-value: < 2.2e-16
```

各回帰係数が0でない値を持つといえるかを統計的に検定

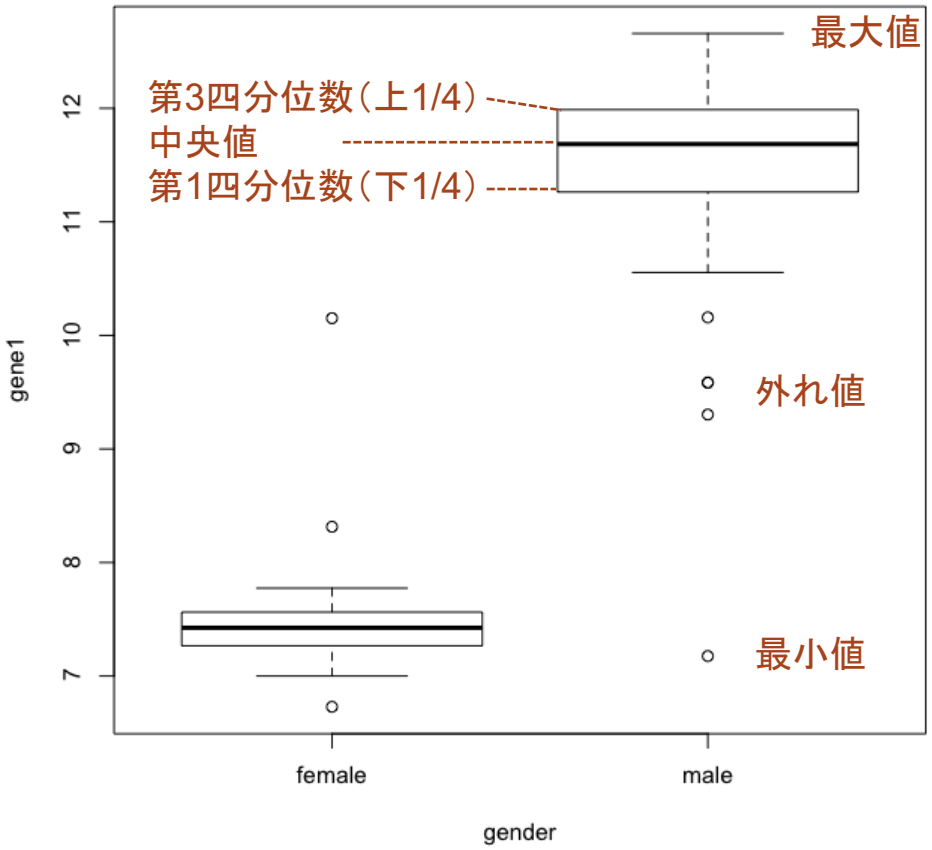
回帰分析全体の有意性

解析が終わったらプロットウィンドウを閉じるか par(mfrow=c(1,1))して、プロットの設定を元に戻しておく

Rによるデータ解析
boxplot（箱ひげ図）の作成

性別ごとのgene1の発現量をboxplotで比較する

```
> plot(gene1 ~ gender, cancer)
```



または
boxplot(gene1 ~ gender, cancer)

Rによるデータ解析
t検定

2組の標本の平均値に差があるかどうかを検定する

男女でgene1の発現量に違いがあるかどうかをt検定で検定する

男性のgene1の発現量を抽出

```
> gene1.m <- subset(cancer, gender=="male")$gene1
```

女性のgene1の発現量を抽出

```
> gene1.f <- subset(cancer, gender=="female")$gene1
```

t検定の実行（等分散性を仮定しないWelchの検定）

```
> t.test(gene1.m,gene1.f)
        Welch Two Sample t-test

data:  gene1.m and gene1.f

t = 30.734, df = 103.48, p-value < 2.2e-16
```

別法（データフレームから直接データを取り出して検定を実行する）

```
> t.test(gene1 ~ gender, cancer)
```

Rによるデータ解析
分散分析

- 線形モデルを用いて、因子の水準によって目的変数の平均値に違いがあるかを検定する
- 因子が3つ以上の水準をもつ場合も、また考慮する因子が複数ある場合でも使える

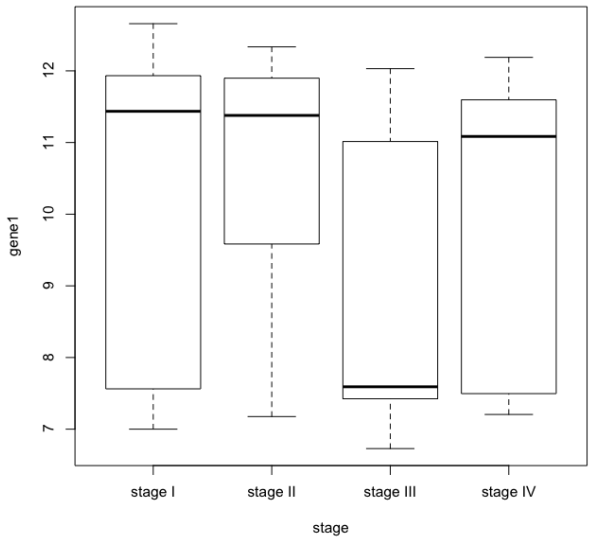
癌のステージ(I～IV の4つの水準がある)によって、gene1の発現に違いがあるかどうかを分散分析で検定する

```
> aov(gene1 ~ stage, cancer)
Call:
   aov(formula = gene1 ~ stage, data = cancer)

Terms:
                    stage Residuals
Sum of Squares    47.0675  399.6562
Deg. of Freedom         3       103

Residual standard error: 1.969812
Estimated effects may be unbalanced
```

plot(gene1 ~ stage, cancer)



結果を変数 result.aov に格納してsummaryで出力

```
> result.aov <- aov(gene1 ~ stage, cancer)
> summary(result.aov)
             Df Sum Sq Mean Sq F value  Pr(>F)
stage         3   47.1   15.69   4.043 0.00921 **
Residuals   103  399.7    3.88
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

分散分析表

stageがgene1の変動に及ぼす効果の検定

解析結果の保存

```
# 指定したオブジェクトを "cancer.Robj" という
# ファイルに保存
> save(cancer, result.lm, result.aov,
     file="cancer.Robj")

# 保存したオブジェクトをファイルからロードする
> load("cancer.Robj")

# 現在のワークスペース中のオブジェクトをすべて
# デフォルトの保存ファイル(".RData")に保存。
> save.image()
```

- 明示的にsave.image()コマンドを実行しなくても、終了時にワークスペースを保存するか聞かれるので、そこで保存を選択することにより保存できる。
- ここで保存したデータは、現在の作業ディレクトリが起動時のディレクトリと同じであれば、次回起動時に自動的にロードされるが、ディレクトリを変更した場合は`load(".RData")`で明示的にロードする必要がある。

補足：データ構造、プログラミング、パッケージの利用

リスト

- ベクトルは同一の型のデータを要素とするが、リストは任意の一般に異なる型のデータを要素として含むことができる。

```
> list(1, "a")
```

- リストは、ベクトルや行列、リストをも要素として含むことができるため、複雑なデータを表現できる。

```
> list(c(1,2), c("a","b"), list(1, "a"))
```

- リストの各要素は[[ ]] で参照できるほか、名前をつけて参照することもできる。

```
> x <- list(first=1, second="b")
> x[[1]]      # 1番目の要素
 [1] 1
> x$second    # second という名前の要素
 [1] "b"
```

- データフレームのほか、各種の統計処理の結果などもリストとして表現されている。リストの内部構造はstr関数で確認できる。

```
> str(x)
> str(cancer)
```

オブジェクトの属性とクラス

- 統計解析の結果などは、一般に決まった型と名前の要素を持ち、一定のクラス名をアサインされたリスト（オブジェクト）として返される。

例）aov 関数の結果を格納した変数 result.aov

```
> class(result.aov)      クラス名の表示
[1] "aov" "lm"
> names(result.aov)      要素名の一覧の表示
[1] "coefficients"  "residuals"  "effects"   "rank"  ...
> str(result.aov)        データ構造の階層的な表示
List of 13
 $ coefficients : Named num [1:4] 10.334 0.195 -1.584 -0.224
  ..- attr(*, "names")= chr [1:4] "(Intercept)" "stagestage II"..
 $ residuals    : Named num [1:107] -2.91 -2.77 -2.79 -2.02 ...
  ..- attr(*, "names")= chr [1:107] "GSM254629" "GSM254648"...
```

このオブジェクトの詳細は、help(lm)のValueセクションで確認できる

総称関数 generic function

- **plot, print, summaryなど多くの関数は、引数となるオブジェクトのクラスによって、異なる関数が呼び出されるようになっている。これらを総称関数という**
  - 例）plot は、引数のクラスによって plot.data.frame, plot.formula, plot.lm などが呼び出される。デフォルトでは、plot.default が呼び出される。
- **これにより、Rでは典型的には以下のような定型的な処理の流れによって解析が進められる。**

```
> result <- some.function(data, opt=value, ...)
> summary(result)      解析結果の要約の表示
> plot(result)         解析結果に基づくプロットの作成
```

参考）コンソールで単に x と入力したときは、print(x)が実行されている。printは総称関数であり、xのクラスによって表示される内容は違っている。

関数の作成(1)

- function 関数によって、新たな関数を作成できる

function (引数リスト) { 関数本体 }

例） 引数の2乗を計算して返す関数を square として定義

```
square <- function(x) {
          sq <- x^2
          return(sq)
}
```

引数：呼び出し時に指定した値がxに代入され、関数本体が実行される

戻り値：関数が返す値（ここではsq=x^2）

```
> square(1:5)
[1]  1  4  9 16 25
```

引数1:5を与えて関数の呼び出し

関数の作成はエディタを使って行うとよい。実行するにはマウスでコマンド全体を選択して、Command+Returnを押す。

関数の作成(2)

- 関数は、通常はファイル上に作成し、読み込んで使う。ファイル上に作成したRコマンドを読み込むにはsource("ファイル名")とする。

ファイル plotAll.R（plotAll: 指定した2つの部分データフレーム間での総当たりプロットを作成する関数）

```
plotAll <- function(x, y=x) {
   par(mfrow=c(ncol(x), ncol(y)))
   for (i in 1:ncol(x)) {
      x.name=names(x)[i]
      for (j in 1:ncol(y)) {
         y.name=names(y)[j]
         if (x.name == y.name) {
            plot(x[,i], main=x.name)
         } else {
            plot(x[,i], y[,j],
               xlab=x.name,ylab=y.name)
         }
      }
   }
}
```

繰り返し

for (変数 in ベクトル) { 式 }

「変数」に「ベクトル」の要素を順次代入して、「式」を繰り返し実行する。

条件分岐

if (条件式) { 式1 } else{式2}

「条件式」がTRUEなら、「式1」を実行し、FALSEなら「式2」を実行する。

プログラムの読み込みと実行

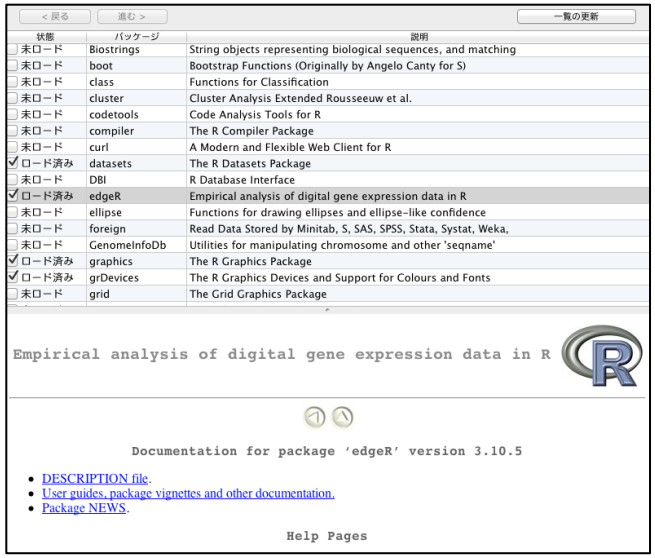
```
> source("plotAll.R")
> plotAll(cancer[,1:4], cancer[,5:10])
```

パッケージの利用

- 様々なパッケージをインストールし、ロードすることによりRの機能を拡張できる。
- バイオインフォマティクス関連では、Bioconductorプロジェクトによって、膨大な種類のパッケージが提供されている。

パッケージインストーラ
パッケージのインストール/更新

パッケージマネージャ
パッケージのロード



コマンドラインからの実行

パッケージのインストール
```
install.packages("パッケージ名")
```

パッケージの更新
```
update.packages("パッケージ名")
```

Bioconductor パッケージの場合
```
source("https://bioconductor.org/
  biocLite.R")　（改行なしで続ける）
biocLite("パッケージ名")
```

パッケージのロード
```
library("パッケージ名")
```