

統計解析ソフト R の活用

～ R 入門 & R でデータハンドリング ～



本日のメニュー

1. R の機能の概要

▶ 起動方法 電卓機能 終了方法


- ▶ 行列計算
- ▶ 簡単なシミュレーション
- ▶ グラフ作成

2. データのハンドリング

- ▶ パッケージと作業ディレクトリについて
- ▶ データの読み込み
- ▶ 重回帰や Cox 回帰用のデータの作成例



R の起動 [Windows 版]

- ▶ R のアイコン  をクリック or スタートメニューから起動
- ▶ Vista / 7 をお使いの方は、アイコンを右クリックして「管理者権限として実行」を選択して起動してください



← クリック！



R の起動 [Mac OS X, Linux, Unix 版]

▶ Mac OS X 版 R

- ▶ Finder 中のアプリケーションフォルダにある「R」というアイコンをダブルクリック

▶ Linux 版 R, Unix 版 R

- ▶ 端末アプリケーションのコマンドライン上で「R」と入力するか OS によってはデスクトップのメニューに「Gnu R」があるのでこれを選択



電卓機能

```
R Console
ファイル 編集 その他 パッケージ ウィンドウ ヘルプ コピー

R version 2.13.1 (2011-07-08)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-pc-mingw32/i386 (32-bit)

Rは、自由なソフトウェアであり、「完全に無保証」です。
一定の条件に従えば、自由にこれを再配布することができます。
配布条件の詳細に関しては、'license()'あるいは'licence()'と入力してください。

Rは多くの貢献者による共同プロジェクトです。
詳しくは'contributors()'と入力してください。
また、RやRのパッケージを出版物で引用する際の形式については
'citation()'と入力してください。

'demo()'と入力すればデモをみることができます。
'help()'とすればオンラインヘルプが出ます。
'help.start()'でHTMLブラウザによるヘルプがみられます。
'q()'と入力すればRを終了します。

> 1+2
[1] 3
> |
```

① 計算式を入力して [Enter] キーを押す
② 計算結果が表示される

演算子

演算子	+	-	*	/	^
意味	加算	減算	乗算	除算	累乗



電卓機能

```
> sqrt(2) ↵
```




```
[1] 1.414214
```

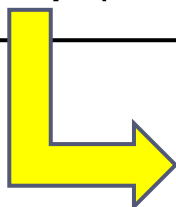
数学関数

関数	$\sin(x)$	$\cos(x)$	$\tan(x)$	$\log(x)$	$\log_{10}(x)$
意味	$\sin x$	$\cos x$	$\tan x$	$\log_e x$	$\log_{10} x$
関数	$\sinh(x)$	$\cosh(x)$	$\tanh(x)$	$\exp(x)$	$\text{sqrt}(x)$
意味	$\sinh x$	$\cosh x$	$\tanh x$	e^x	ルート x
関数	$\text{abs}(x)$	$\text{trunc}(x)$	$\text{round}(x)$	$\text{floor}(x)$	$\text{ceiling}(x)$
意味	絶対値	整数部分	丸め	切り下げ	切り上げ



電卓機能

```
> x <- log(2)  # log(2)を変数 x に代入  
> x  # 変数 x の中身を表示  
[1] 0.6931472  
> help(log)  # 関数 log() に関するヘルプを表示
```



```
log {base} R Documentation  
  
Logarithms and Exponentials  
  
Description  
  
log computes logarithms, by default natural logarithms, log10 computes common (i.e., base 10) logarithms,  
and log2 computes binary (i.e., base 2) logarithms. The general form log(x, base) computes logarithms  
with base base.  
  
log1p(x) computes log(1+x) accurately also for |x| << 1 (and less accurately when x is approximately -1).  
  
exp computes the exponential function.  
  
expm1(x) computes exp(x) - 1 accurately also for |x| << 1.  
  
Usage  
  
log(x, base = exp(1))  
logb(x, base = exp(1))  
log10(x)  
log2(x)  
  
log1p(x)  
  
exp(x)  
expm1(x)
```

← たいてい関数の使用例
が載っているので、
とりあえず例を実行する



電卓機能

- ▶ 5人の「イベント発生までの期間」をベクトル化して変数 x に代入し、
期間の和を算出して変数 y に代入し、イベント発生率を算出

```
> x <- c(1, 2, 4, 6, 7) Ⓜ # 全員がイベント発生例
> ( y <- sum(x) ) Ⓜ # 総観察期間(年)
[1] 20
> 5/y Ⓜ # 人年法によるイベント発生率
[1] 0.25
```

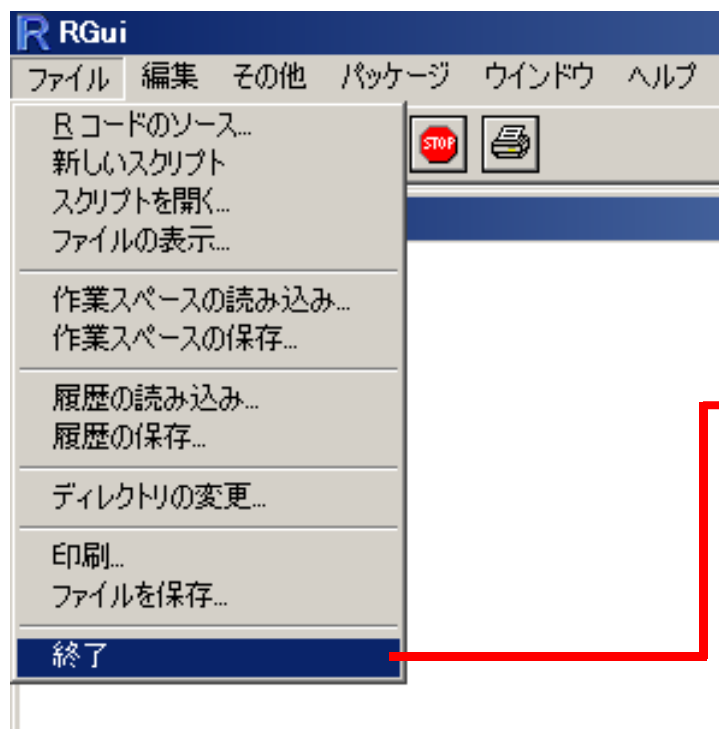
ベクトル用関数

関数	cor(x1, x2)	max(x)	mean(x)	median(x)	min(x)
意味	相関係数	最大値	平均値	中央値	最小値
関数	prod(x)	summary(x)	sd(x)	sum(x)	var(x)
意味	総積	要約統計量	標準偏差	総和	不偏分散



R の終了

- ▶ 関数 `q()` または `quit()` を実行する
- ▶ R ウィンドウの右上の [×] 印をクリックする
- ▶ メニューから選択する



は い : これまでの作業内容を保存する
いいえ : 次に使うときはまっさらな状態



本日のメニュー

1. R の機能の概要

- ▶ 起動方法 電卓機能 終了方法

▶ 行列計算

- ▶ 簡単なシミュレーション
- ▶ グラフ作成

2. データのハンドリング

- ▶ パッケージと作業ディレクトリについて
- ▶ データの読み込み
- ▶ 重回帰や Cox 回帰用のデータの作成例



行列計算

- ▶ 行列 $\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$ を作成する手順は以下のとおり
 1. 行列の要素をベクトルで用意する
 2. 関数 `matrix(ベクトル, 行数, 列数)` でベクトルから行列に変換する

```
> x <- c(1, 2, 3, 4, 5, 6) Ⓜ  
> A <- matrix(x, 2, 3) Ⓜ  
> A Ⓜ  
      [,1] [,2] [,3]  
[1,]    1    3    5  
[2,]    2    4    6
```



行列計算

演算子

演算子	+	-	%*%	*
意味	加算	減算	行列の積	対応する要素同士の積

数学関数

関数	機能
chol(A)	行列 A のコレスキー分解を行う
crossprod(A)	行列 A のクロス積を求める
eigen(A)	行列 A の固有値と固有ベクトルを求める
ginv(A)	行列 A の一般化逆行列を求める (MASS パッケージの呼び出しが必要)
qr(A)	行列 A の QR 分解を行う
svd(A)	行列 A の特異値分解を行う
solve(A)	行列 A の逆行列を求める
t(A)	行列 A の転置を行う



行列計算

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

```
> x <- c(1,2,3,4) Ⓜ
> A <- matrix(x, 2, 2) Ⓜ
> A * A Ⓜ # 対応する要素同士の積 ( 行列の積 )
      [,1] [,2]
[1,]    1    9
[2,]    4   16
> A %*% A Ⓜ # 行列の積
      [,1] [,2]
[1,]    7   15
[2,]   10   22
> t(A) Ⓜ # 行列 A の転置行列
      [,1] [,2]
[1,]    1    2
[2,]    3    4
```



行列計算

コマンド	機能
$A[1,]$	行列 A の 1 行目を取り出す
$A[, 1]$	行列 A の 1 列目を取り出す
$A[1, 2]$	行列 A の 1 行 2 列目の成分を取り出す
$A[c(1,2), 3]$	行列 A の 1, 2 行 3 列目の成分を取り出す
$A[c(1,2), c(2,3)]$	行列 A の 1, 2 行目と 2, 3 列を取り出す

```
> A[1, ] 
```

```
[1] 1 3 5
```

```
> A[1, 2] 
```

```
[1] 3
```



本日のメニュー

1. R の機能の概要

- ▶ 起動方法 電卓機能 終了方法
- ▶ 行列計算

▶ 簡単なシミュレーション

- ▶ グラフ作成

2. データのハンドリング

- ▶ パッケージと作業ディレクトリについて
- ▶ データの読み込み
- ▶ 重回帰や Cox 回帰用のデータの作成例



関数とシミュレーション

- ▶ R では「乱数を利用」して「シミュレーションのための関数を作成」しシミュレーションを行う

乱数生成関数（一部）

関数	機能
<code>rbeta(10, shape1=2, shape2=3)</code>	パラメータ (2, 3) であるベータ分布に従う乱数を 10 個
<code>rbinom(10, size=5, prob=0.3)</code>	成功確率 0.3, 試行数 5 回の二項分布に従う乱数を 10 個
<code>rchisq(10, df=5, ncp=2)</code>	自由度 5, 非心度 2 の χ^2 分布に従う乱数を 10 個
<code>rexp(10, rate=1)</code>	パラメータ 1 の指数分布に従う乱数を 10 個
<code>rnorm(10, mean=0, sd=1)</code>	平均 0, 標準偏差 1 の正規分布に従う乱数を 10 個
<code>rpois(10, lambda=2)</code>	パラメータ 2 のポアソン分布に従う乱数を 10 個
<code>rf(10, df1=2, df2=3)</code>	自由度 (2, 3) の F 分布に従う乱数を 10 個
<code>rt(10, df=8)</code>	自由度 8 の t 分布に従う乱数を 10 個
<code>runif(10, min=0, max=1)</code>	(0, 1) 区間の一様乱数を 10 個
<code>sample(ベクトル, replace=TRUE)</code>	ベクトルからランダムサンプリング (<code>replace=TRUE</code> で復元抽出)

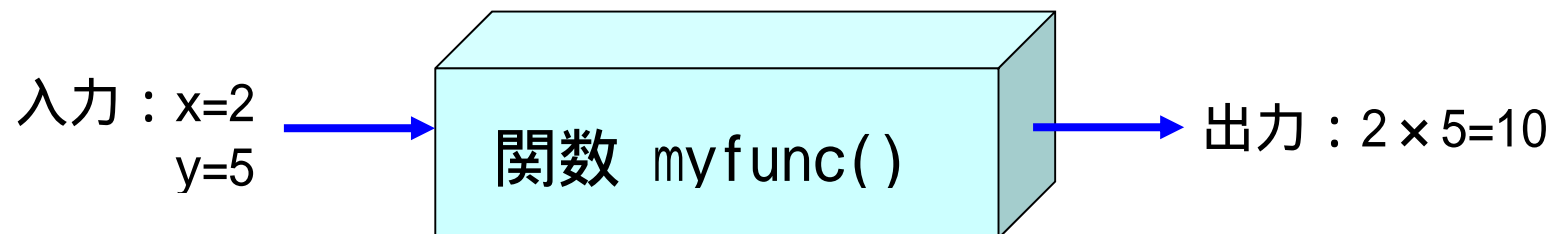


関数の作成例

1. 関数名を決める
2. 入力する変数の個数と種類を指定する
3. 計算手順を 1 行ずつ記述し最後に関数 `return()` で計算結果を出力

【例】 入力した 2 つの数値の積が出力される関数 `myfunc(x,y)`

```
> myfunc <- function(x, y) {           #  
+   return(x*y)                       # 関数定義部分  
+ }                                     #  
> myfunc(2,5)                         # 関数を実行する  
[1] 10
```





関数の作成例

【例】関数 $f(x) = 2x$

```
> f <- function(x) {  
+   return(2*x)  
+ }  
> f(3)  
[1] 6
```

【例】二次元標準正規分布の密度 $z(x, y) = \frac{1}{2\pi} \exp\left\{\frac{-(x^2 + y^2)}{2}\right\}$

```
> z <- function(x,y) {  
+   return( 1/(2*pi)*exp(-(x^2+y^2)/2) )  
+ }  
> z(0,0)  
[1] 0.1591549
```



シミュレーションの例：例数設計

- ▶ 大うつ病を患っている患者さんに薬剤 1 または薬剤 2 を投与した後、薬剤間の QOL の平均値を比較する（QOL は数値が大きい方が良い）
 - ▶ 薬剤 1 の QOL の平均値は 6.5，薬剤 2 の QOL の平均値は 4.0
 - ▶ 標準偏差は両薬剤とも同じ 3.0，例数は 1 群 20 例， $\alpha = 5\%$ （両側）
- ▶ 帰無仮説 H_0 ：平均値の差が 0 である
という帰無仮説に関する 2 標本 t 検定を行ったときに，薬剤 1 が薬剤 2 に勝ることを検出する検出力を算出する

1. 各薬剤の患者さんのデータを正規乱数から 20 例分 $\times 2 = 40$ 個生成
2. 2 標本 t 検定を行い，薬剤 1 が薬剤 2 に勝っているかを確認・記録
3. 1 ~ 2 を多数回（例えば 1 万回）繰り返す
4. 1 万回中「薬剤 1 が薬剤 2 に勝った回数」の割合が検出力



シミュレーションの例：例数設計

```
> mypower <- function(n) {
+   count <- 0
+   for (i in 1:n) {
+     MYDATA <- data.frame(
+       GROUP = c( rep(1,20), rep(2,20) ),
+       QOL    = c( rnorm(20, mean=6.5, sd=3.0),
+                  rnorm(20, mean=4.0, sd=3.0))
+     )
+     result <- t.test(QOL ~ GROUP, var=T, data=MYDATA)
+     if ((result$estimate[1]-result$estimate[2] > 0) &&
+         (result$p.value < 0.05) ) count <- count+1    # 有意差あり
+   }
+   return(count/n)
+ }
> mypower(10000)
[1] 0.7281
```



【参考】検出力の計算

```
> power.t.test(delta=2.5, sd=3.0, sig.level=0.05, n=20,  
+             type="two.sample")
```

Two-sample t test power calculation

```
      n = 20  
  delta = 2.5  
     sd = 3  
sig.level = 0.05  
  power = 0.7284655  
alternative = two.sided
```

NOTE: n is number in *each* group



本日のメニュー

1. R の機能の概要

- ▶ 起動方法 電卓機能 終了方法
- ▶ 行列計算
- ▶ 簡単なシミュレーション

▶ グラフ作成

2. データのハンドリング

- ▶ パッケージと作業ディレクトリについて
- ▶ データの読み込み
- ▶ 重回帰や Cox 回帰用のデータの作成例



グラフの作成手順

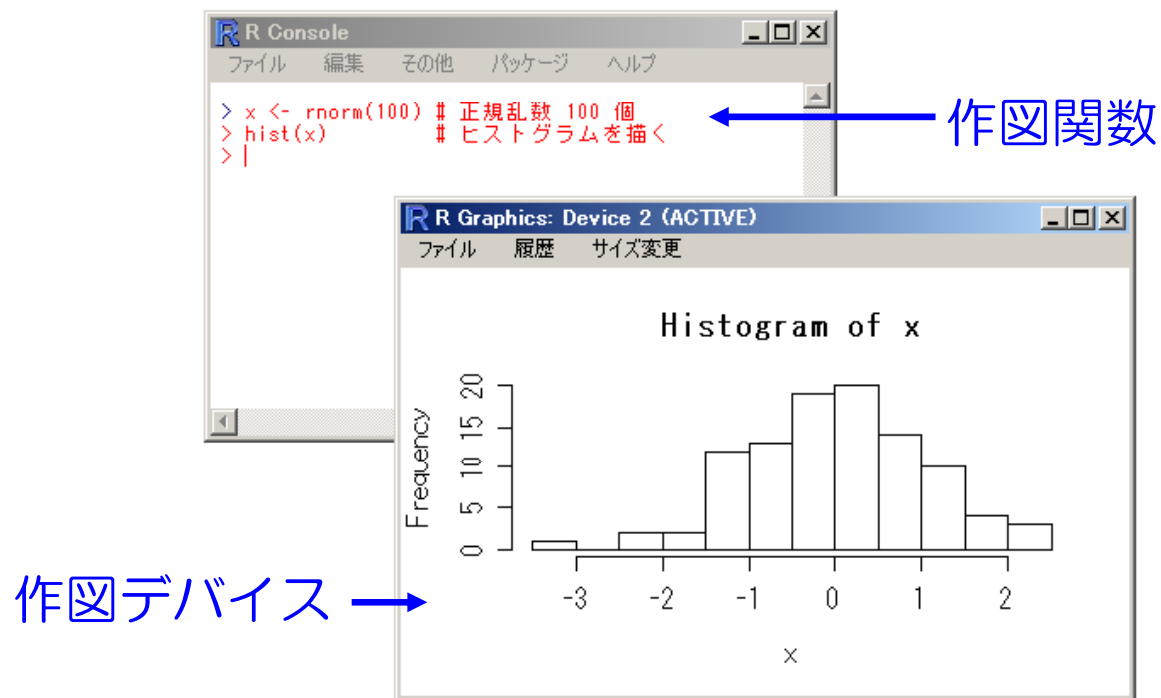
1. プロットするデータや数式を準備する
2. 高水準作図関数で作図デバイスにグラフを描く
3. 低水準作図関数でグラフに装飾を施す
4. 描いたグラフを保存する

- ▶ 作図デバイスって何？
- ▶ 作図関数って何？ 高水準？ 低水準？
- ▶ グラフを保存？ 保存される形式は何？



グラフの作成

- ▶ Rでは、以下の2つを用いて作図を行う
 - ▶ 作図関数：グラフを出力する関数
 - ▶ 作図デバイス：図を出力する装置（ウィンドウ）



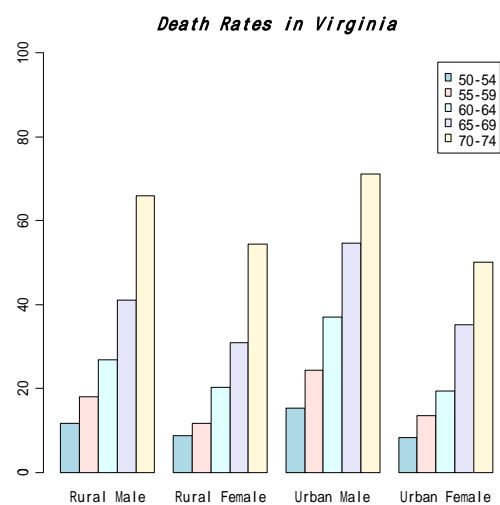


グラフの作成

- ▶ 作図関数は主に以下の 2 つを用いる
 - ▶ **高水準作図関数**：1枚の完成された図を描く
 - (例) 関数 `plot()` を使って散布図を描く
 - (例) 関数 `hist()` を使ってヒストグラムを描く関数 `plot()` や関数 `hist()` が高水準作図関数
 - ▶ **低水準作図関数**：完成された図に図形や文字などを追記する
 - (例) 関数 `legend()` を使って棒グラフに凡例を追記する
 - (例) 関数 `abline()` を使って散布図に回帰直線を追記する関数 `legend()` や関数 `abline()` が低水準作図関数

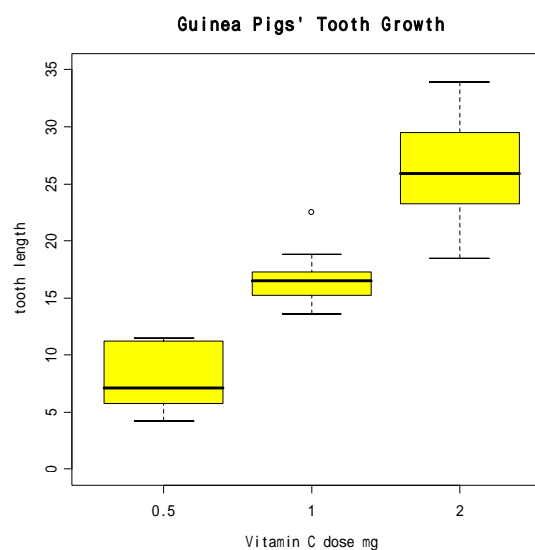


高水準作図関数で描けるグラフのカタログ



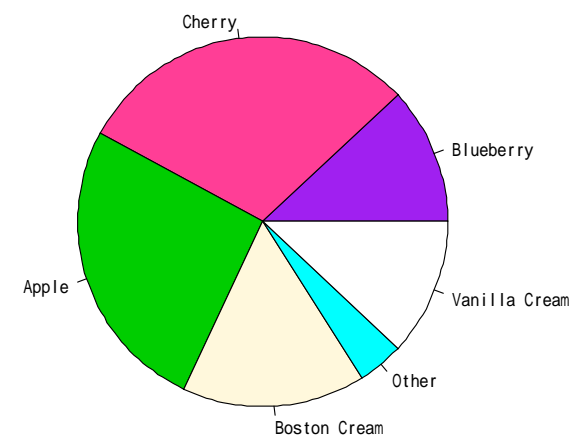
棒グラフ

`barplot(...)`



箱ひげ図

`boxplot(...)`

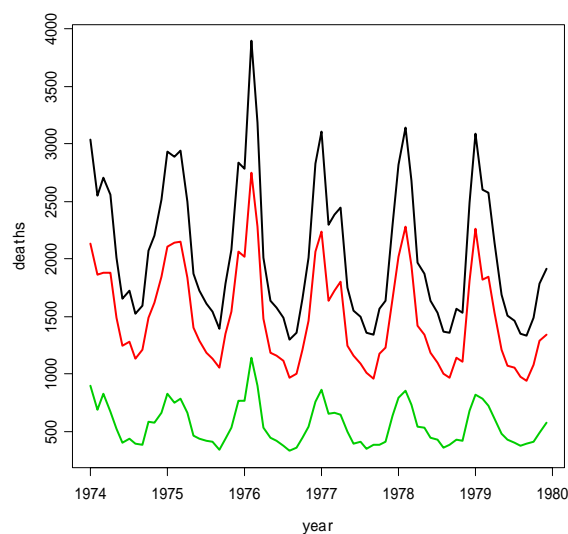


円グラフ

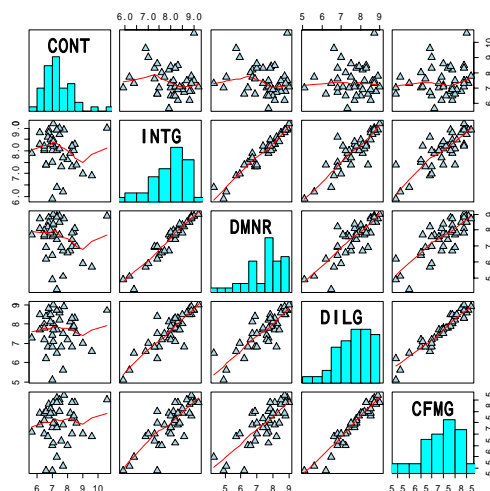
`pie(...)`



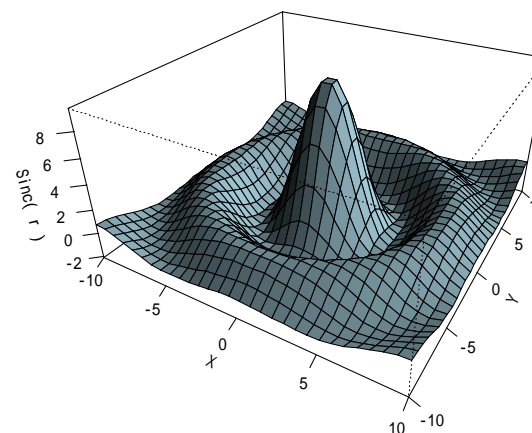
高水準作図関数で描けるグラフのカタログ



時系列データのプロット
`plot(...)`



5変数データの散布図
`pairs(...)`



2変数関数のグラフ
`persp(...)`



低水準作図関数の一覧

追記する図形	関数
点	points()
直線(1)	lines(), segments()
直線(2)	abline(), abline(回帰分析の結果)
格子	grid()
矢印	arrows()
矩形	rect()
文字	text(), mtext(), title()
枠と軸	box(), axis()
凡例	legend()
多角形	polygon()



グラフの作成例

- ▶ pdf 形式の作図デバイスを開く

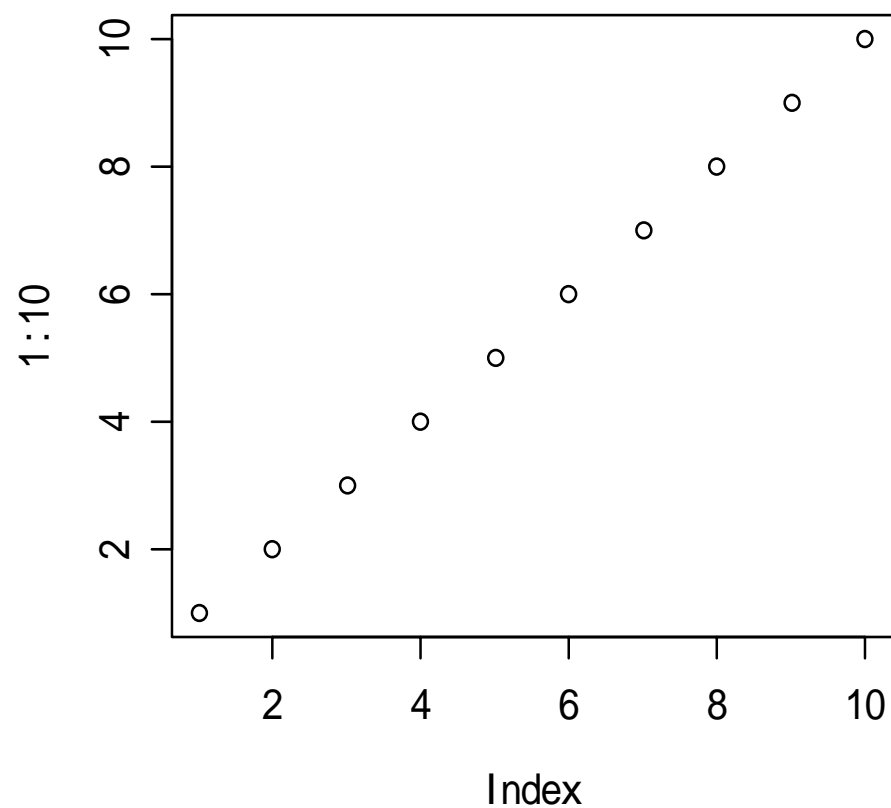
```
> pdf("C:/temp/myplot.pdf")
```



グラフの作成例

▶ 散布図を描く

```
> pdf("C:/temp/myplot.pdf")  
> plot(1:10)
```

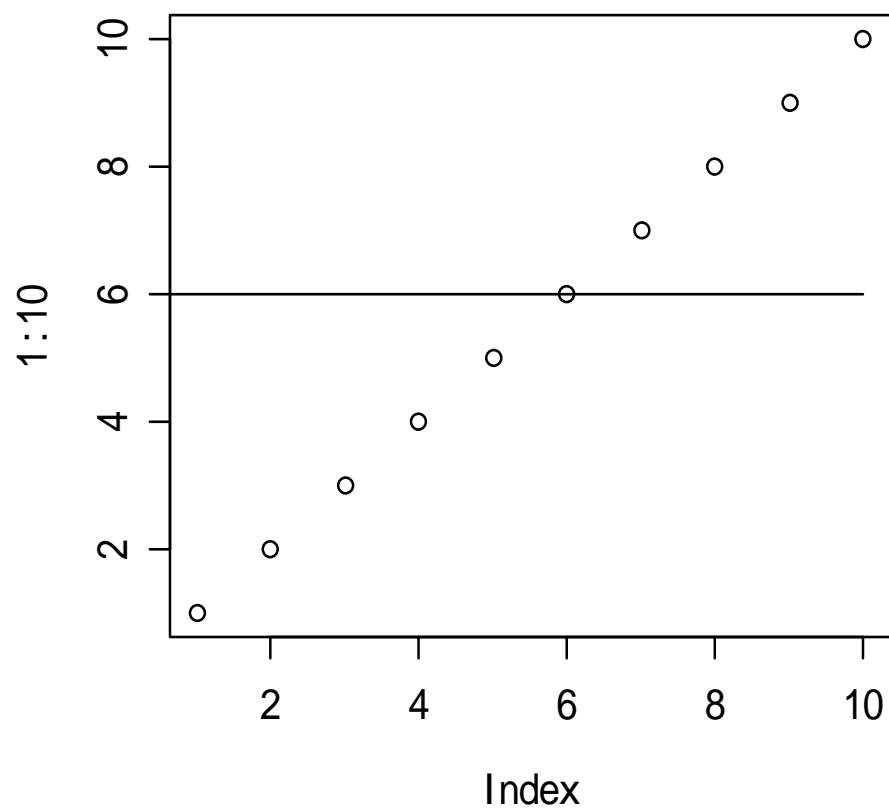




グラフの作成例

▶ 水平線を追記

```
> pdf("C:/temp/myplot.pdf")  
> plot(1:10)  
> lines(c(0,10), c(6,6))
```

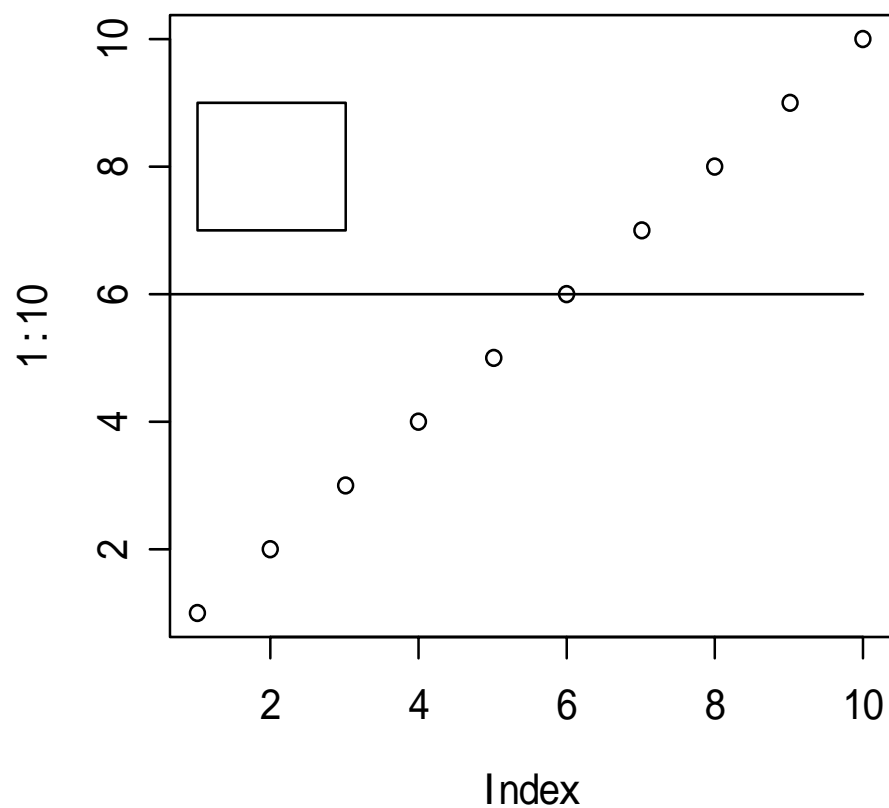




グラフの作成例

▶ 矩形を追記

```
> pdf("C:/temp/myplot.pdf")  
> plot(1:10)  
> lines(c(0,10), c(6,6))  
> rect(1,7, 3,9)
```

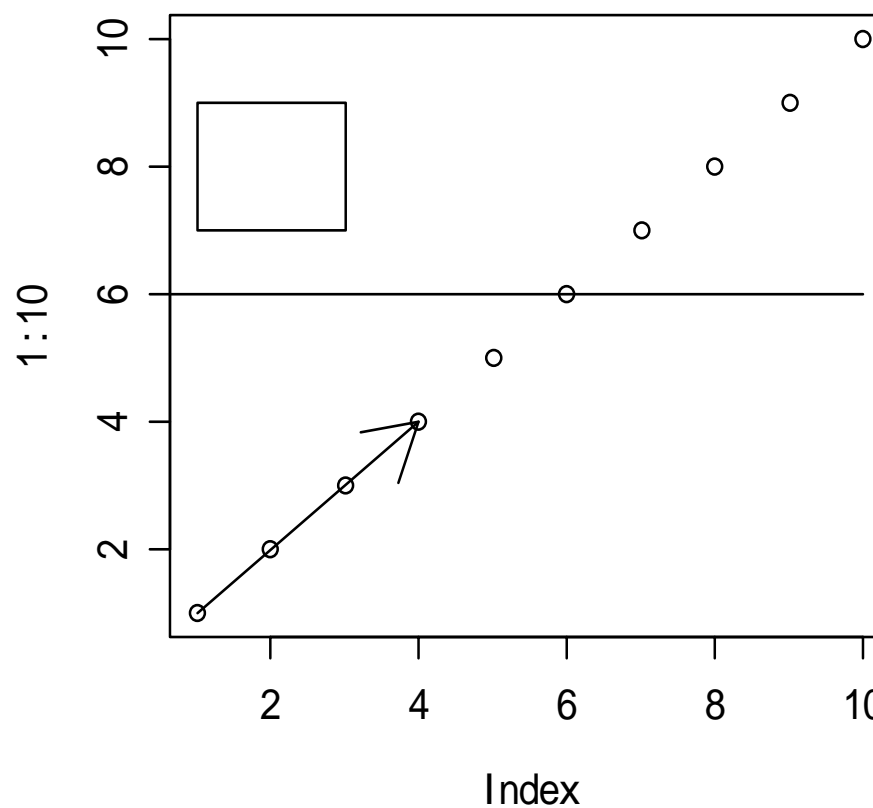




グラフの作成例

▶ 矢印を追記

```
> pdf("C:/temp/myplot.pdf")  
> plot(1:10)  
> lines(c(0,10), c(6,6))  
> rect(1,7, 3,9)  
> arrows(1,1, 4,4)
```

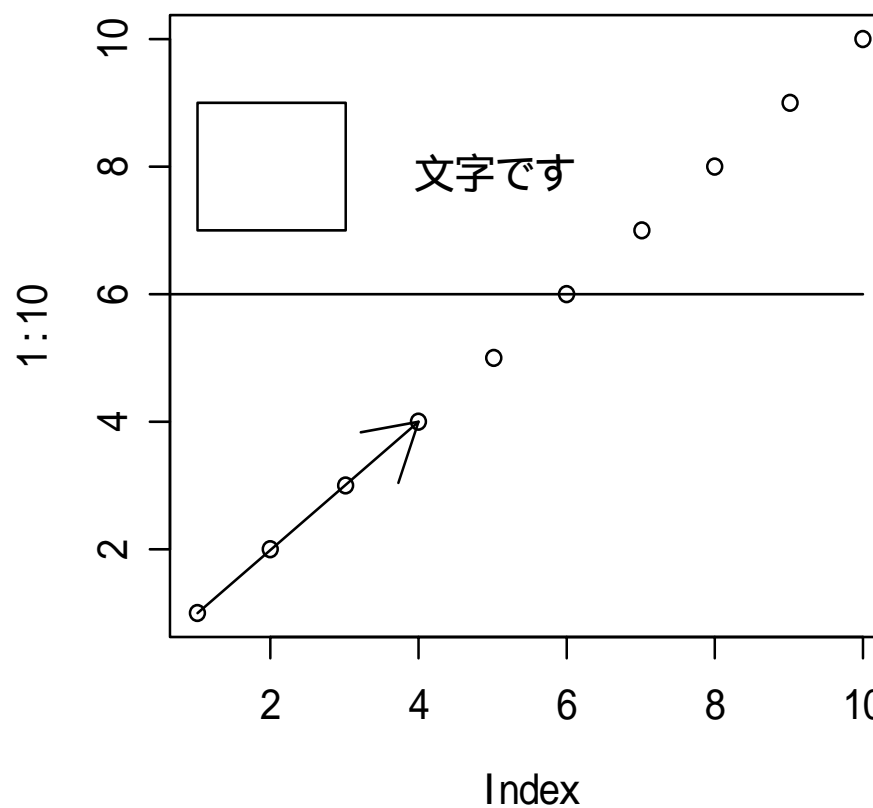




グラフの作成例

▶ 文字を追記

```
> pdf("C:/temp/myplot.pdf")  
> plot(1:10)  
> lines(c(0,10), c(6,6))  
> rect(1,7, 3,9)  
> arrows(1,1, 4,4)  
> text(5,8,"文字です")
```

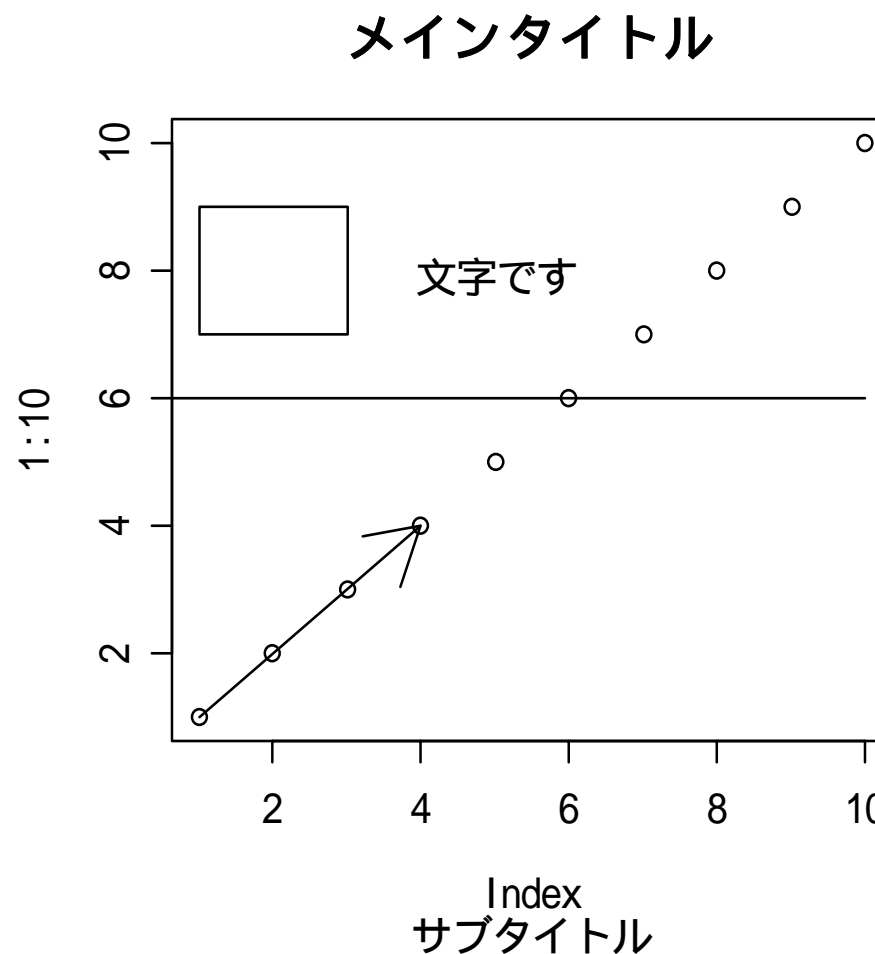




グラフの作成例

▶ タイトルを追記

```
> pdf("C:/temp/myplot.pdf")  
> plot(1:10)  
> lines(c(0,10), c(6,6))  
> rect(1,7, 3,9)  
> arrows(1,1, 4,4)  
> text(5,8,"文字です")  
> title("メインタイトル",  
        "サブタイトル")
```

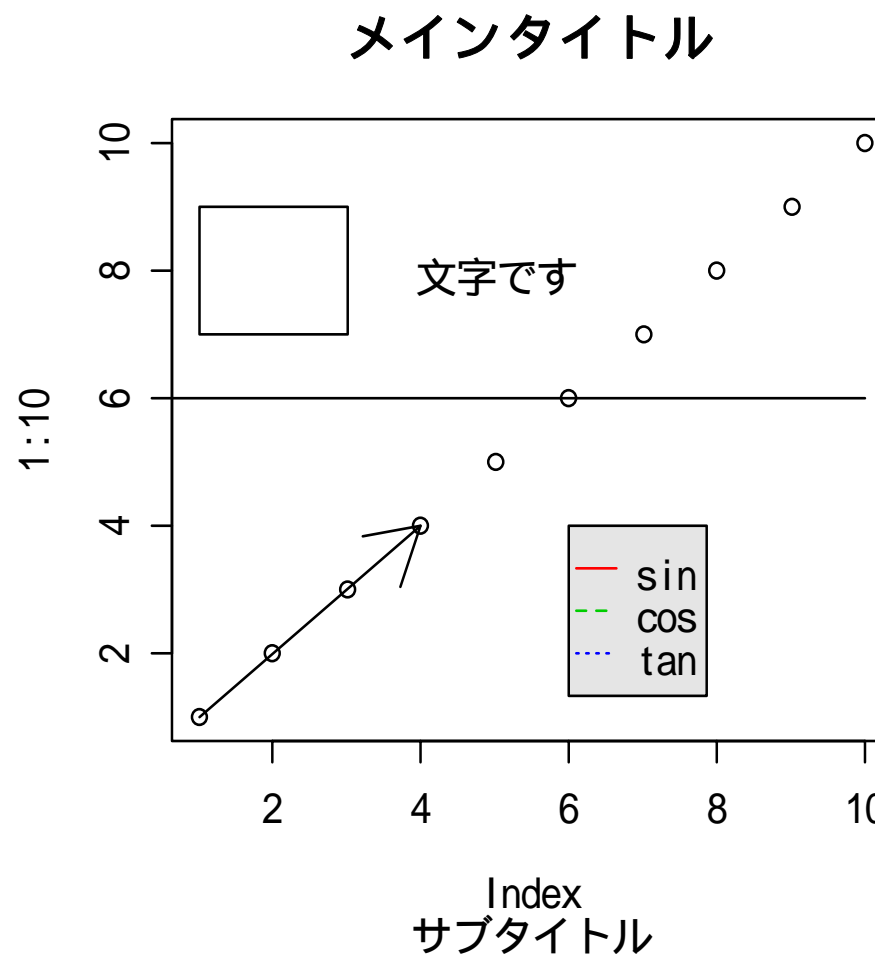




グラフの作成例

▶ 凡例を追記

```
> pdf("C:/temp/myplot.pdf")
> plot(1:10)
> lines(c(0,10), c(6,6))
> rect(1,7, 3,9)
> arrows(1,1, 4,4)
> text(5,8,"文字です")
> title("メインタイトル",
        "サブタイトル")
> legend(6, 4,
        c("sin", "cos", "tan"),
        col=2:4, lty = 1:3,
        bg='gray90')
```

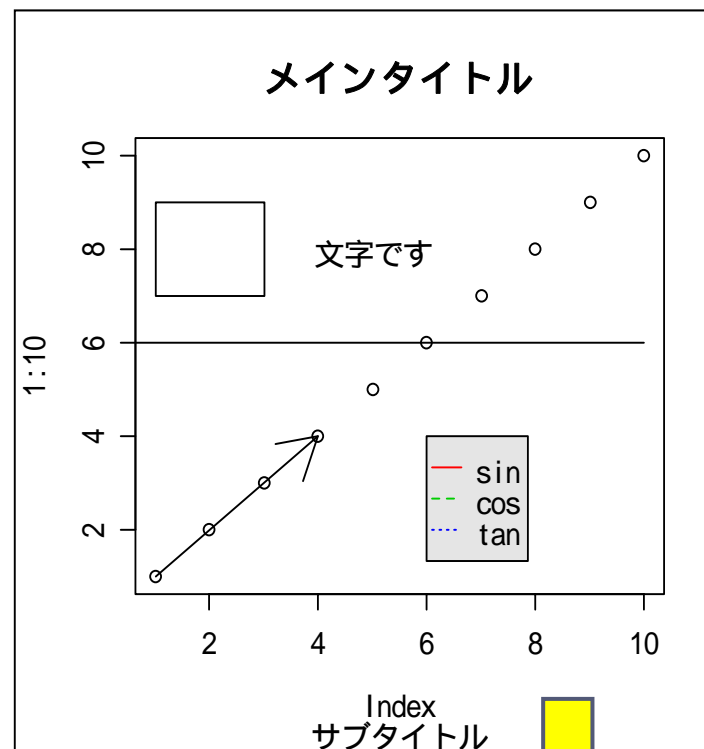




グラフの作成例

- ▶ 作図デバイスを閉じる <C:/temp/myplot.pdf> に出力される

```
> pdf("C:/temp/myplot.pdf")
> plot(1:10)
> lines(c(0,10), c(6,6))
> rect(1,7, 3,9)
> arrows(1,1, 4,4)
> text(5,8,"文字です")
> title("メインタイトル",
+       "サブタイトル")
> legend(6, 4,
+        c("sin", "cos", "tan"),
+        col=2:4, lty = 1:3,
+        bg='gray90')
> dev.off()
```





【参考】作図デバイスの種類

- ▶ 作図デバイスの種類
 - ▶ パソコンの画面に表示するためのデバイス（装置）
 - ▶ 画像ファイルに保存するためのデバイス（装置）
 - ▶ **bmp()**：ビットマップ形式
 - ▶ **jpeg()**：JPEG 形式（3段階で品質が選択出来る）
 - ▶ **pdf()**：ADOBE PDF 形式
 - ▶ **pictex()**：LaTeX の画像形式
 - ▶ **png()**：PNG 形式
 - ▶ **postscript()**：ADOBE PostScript 形式
関数 `dev.copy2eps()` でEPSファイルへの保存も出来る
 - ▶ **win.metafile()**：windows meta file
emf, wmf 形式：パワーポイント上で編集することが出来る形式



【余談】 R を使いこなす近道は？

- ▶ 「メモ」「アンチョコ」「自分用のコマンド集」を作ってください
- ▶ ヘルプの見方, 分からないことが出てきたときの検索方法を身につけてください (ヘルプ, Google, R の書籍など)
- ▶ エラーが出てても気にしないでください
- ▶ 「とりあえず人様が書いたプログラムを実行」
「そのプログラムの一部を修正して実行」
という作業に慣れてください



本日のメニュー

1. R の機能の概要

- ▶ 起動方法 電卓機能 終了方法
- ▶ 行列計算
- ▶ 簡単なシミュレーション
- ▶ グラフ作成

2. データのハンドリング

- ▶ パッケージと作業ディレクトリについて
- ▶ データの読み込み
- ▶ 重回帰や Cox 回帰用のデータの作成例



パッケージとは

- ▶ R は関数とデータを機能別に分類して「パッケージ」という形で用意
- ▶ どのようなパッケージがあるのかは関数 `library()` を実行すると表示

パッケージ名	解説
boot	ブートストラップに関するパッケージ
foreign	R 以外のデータファイルを読み込むためのパッケージ
lattice	ラティス・グラフィックス関数パッケージ
nlme	線形&非線形混合効果モデル用のパッケージ
nnet	ニューラル・ネットワーク用のパッケージ
rpart	CARTに関するパッケージ
splines	スプライン回帰用のパッケージ
survival	生存時間解析用のパッケージ

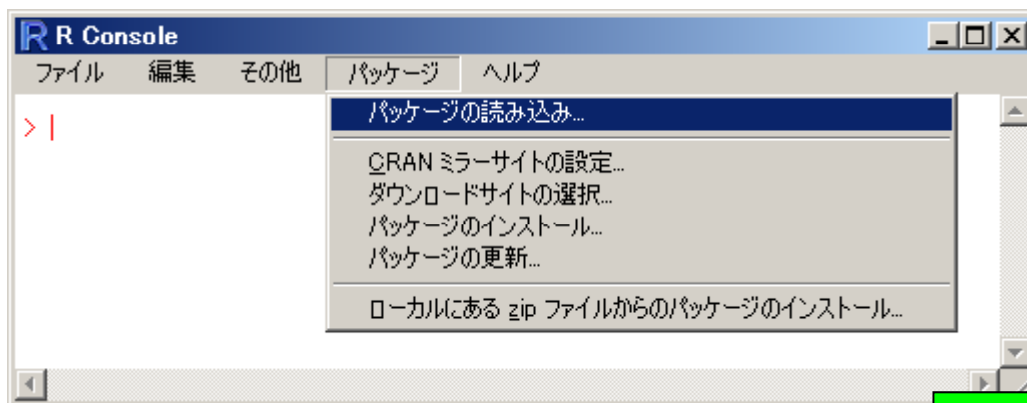


パッケージの呼び出し

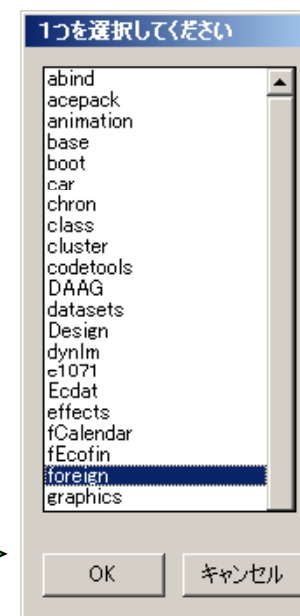
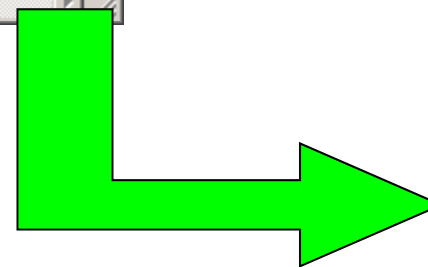
- ▶ コマンドでパッケージ「foreign」を呼び出す場合：

```
> library(foreign)      # パッケージ foreign を呼び出す  
> library(help="foreign") # パッケージ foreign のヘルプ
```

- ▶ メニューからパッケージ「foreign」を呼び出す場合：



- ① メニュー「パッケージ」から「パッケージの読み込み」を選択
- ② 読み込むパッケージ名を選択して [OK] を選択



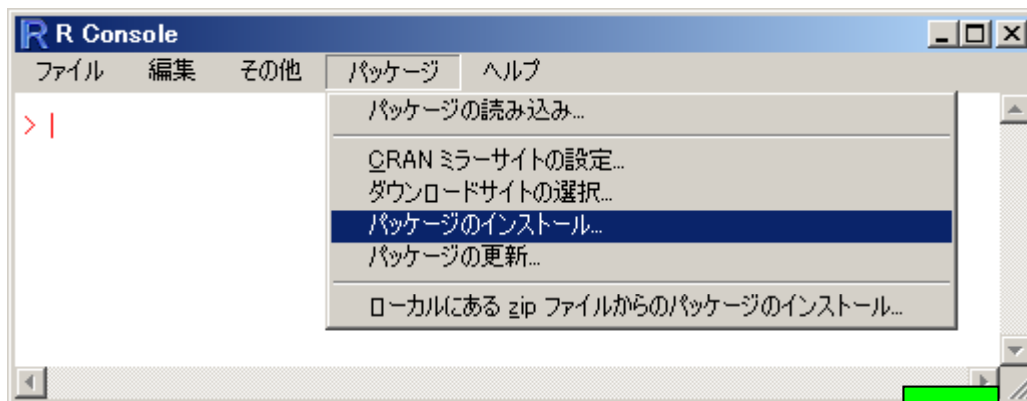


追加パッケージのインストール

- ▶ コマンドでパッケージ「xlsx」をインストールする：

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

- ▶ メニューからパッケージ「xlsx」をインストールする：

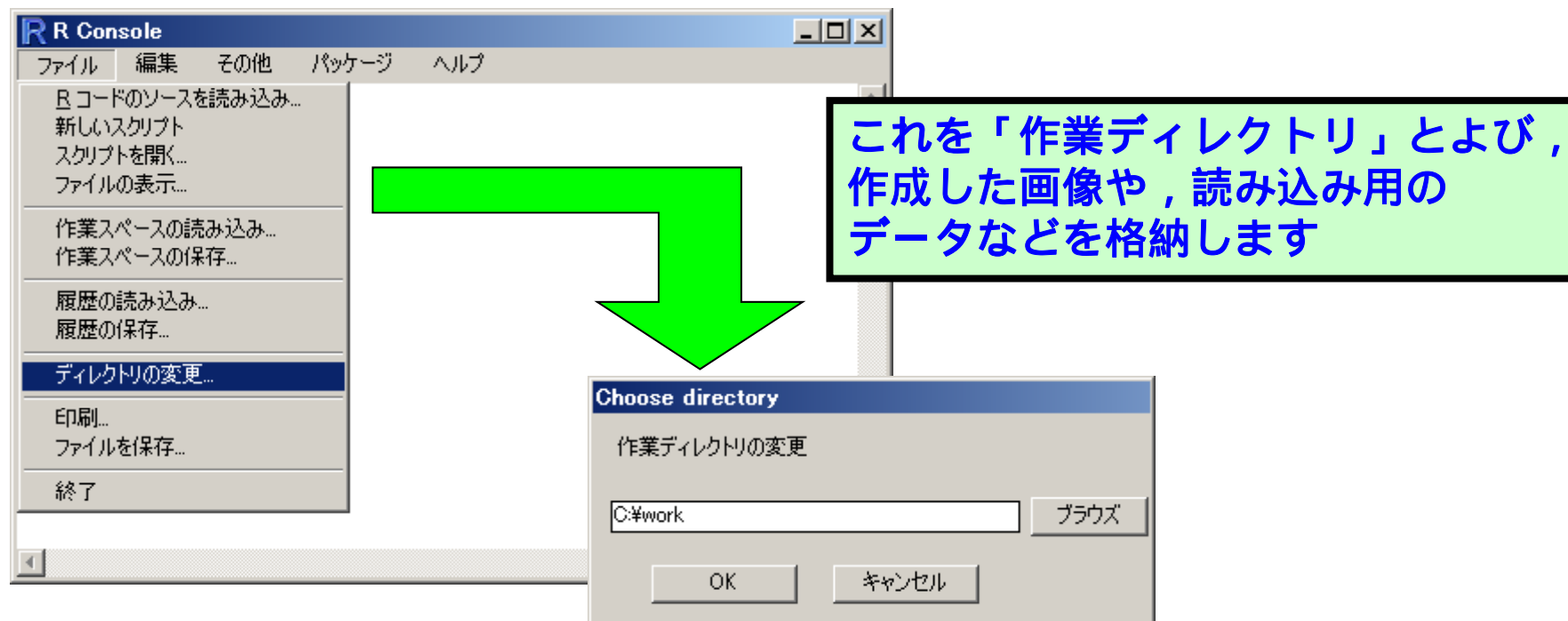


- ① メニュー「パッケージ」から「パッケージのインストール」を選択
- ② 「Japan(Tsukuba)」 [OK] をクリック
- ③ インストールするパッケージを選択して [OK] をクリック



作業ディレクトリの変更と確認

- ▶ Windows 版 R の場合は、「ファイル」 「ディレクトリの変更...」を選択した後、フォルダ「work」を選択してください



```
> setwd("c:/work")      # 作業ディレクトリを変更
> getwd()               # 現在のディレクトリを確認
[1] "c:/work"
```



本日のメニュー

1. R の機能の概要

- ▶ 起動方法 電卓機能 終了方法
- ▶ 行列計算
- ▶ 簡単なシミュレーション
- ▶ グラフ作成

2. データのハンドリング

- ▶ パッケージと作業ディレクトリについて
- ▶ データの読み込み
- ▶ 重回帰や Cox 回帰用のデータの作成例



準備：データの型

- ▶ Rには「データの型」という概念があり、「数値」「文字」「日付」「因子（カテゴリ）」などを区別する [日付の処理例は次頁](#)

```
> height <- c(158,162,177,173,166) # 数値型
> group <- c("A","A","B","C","C") # 文字型
> group <- as.factor(group) # 関数 as.factor で
> # 因子型(カテゴリ)に変換
> groupc <- as.character(group) # 文字型に変換
> date <- as.Date("111111", format="%y%m%d") # 日付型に変換
```

- ▶ 外部ファイルをRに読み込むと「数値」は「数値型」,
「文字」は「因子型（カテゴリ）」に自動変換される
「文字」を「文字型」としたい場合は[要変換!](#)



準備：日付データのハンドリング例

```
> as.Date("2012/01/26", format="%Y/%m/%d") # 文字列を日付に変換
[1] "2012-01-26"
> x <- as.Date("2012/01/26", format="%Y/%m/%d") -
+       as.Date("111111", format="%y%m%d") # 日数の差を計算
> as.numeric(x) # 結果を数値に変換
[1] 76
```

命令	機能
%A, %a	曜日の英語名（小文字は略記）
%B, %b	月の英語名（小文字は略記）
%d	日（01-31）
%m	月（01-12）
%Y, %y	西暦（大文字：4桁表示，小文字：2桁表示）



データフレームとは

- ▶ 統計解析を行うデータの形式は様々
 - ▶ (R上で) データを手で入力して...
 - ▶ テキストファイル, EXCEL, ACCESS, SAS などの形式
- ▶ Rでデータ解析を行う際は, データフレームという形式にデータを変換することが多い(見た目は行列)

	A	B	C
1	sex	height	weight
2	F	160	50
3	F	165	65
4	M	170	60
5	M	175	55
6	M	180	70

EXCEL : シート

sex	height	weight
F	160	50
F	165	65
M	170	60
M	175	55
M	180	70

ACCESS : テーブル

	sex	height	weight
1	F	160	50
2	F	165	65
3	M	170	60
4	M	175	55
5	M	180	70

SAS : データセット



データフレームとは

- ▶ 数値や文字, 因子 (カテゴリ) や日付などの異なる型のデータをまとめてもつ変数
 - 外見は行列と同じ
 - 各列の値の型はバラバラでも良い
- ▶ データフレームの各行・各列はラベルを必ず持ち, ラベルによる操作が可能
- ▶ データは C:/ にあることを仮定する
 - 前もって `setwd("C:/")` を実行

データフレーム「demo」

ID	AGE	GENDER	DATE
2	50	1	2001/01/01
4	55	2	2002/02/02
6	60	2	2003/03/03
3	65	1	2004/04/04
1	70	2	2005/05/05
5	75	1	2006/06/06

GENDER : 1 が男性, 2 が女性



データフレームの作成方法

1. R でベクトルデータを作成した後，データフレームを作成（手入力）
 - ▶ 「ID」「年齢」「性別」「検査日」などのデータをベクトルで用意した後，関数 [data.frame\(\)](#) で1つのデータフレームに変換
2. ファイルからデータを読み込んで，データフレームを作成
 - ▶ テキストファイルから読み込み
 - ▶ 関数 [read.table\(\)](#) や関数 [read.csv\(\)](#)
 - ▶ EXCEL ファイルから読み込み
 - ▶ CSV ファイルに変換した後，関数 [read.csv\(\)](#)
 - ▶ パッケージ `xlsx` の関数 [read.xlsx\(\)](#)
 - ▶ EXCEL のセルを直接コピー

cf. パッケージ RODBC の関数 [odbcConnectXXXXX\(\)](#) でファイルにアクセスした後，関数 [sql.Query\(\)](#) でデータを読み込む



データフレームの作成 (.txt)

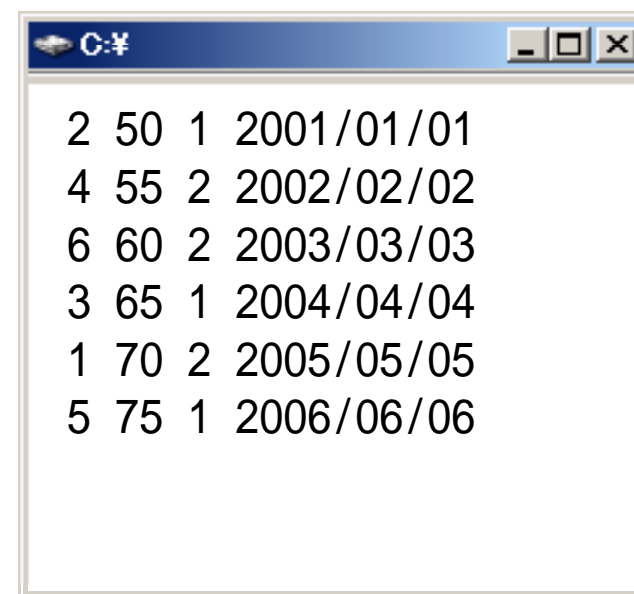
1. 列名がなく，データ間がスペースで区切られている場合

R が勝手に列名を決める

```
> x <- read.table("demo1.txt")
```

```
> x
```

	V1	V2	V3	V4
1	2	50	1	2001/01/01
2	4	55	2	2002/02/02
3	6	60	2	2003/03/03
4	3	65	1	2004/04/04
5	1	70	2	2005/05/05
6	5	75	1	2006/06/06



```
C:\羊
2 50 1 2001/01/01
4 55 2 2002/02/02
6 60 2 2003/03/03
3 65 1 2004/04/04
1 70 2 2005/05/05
5 75 1 2006/06/06
```

demo1.txt

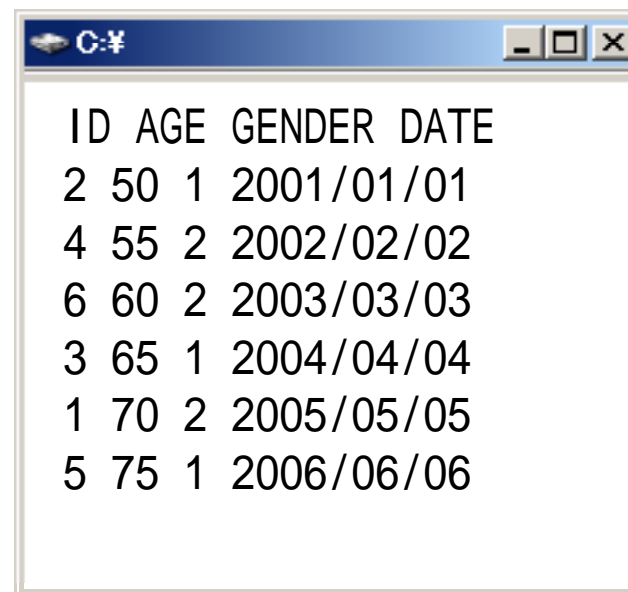


データフレームの作成 (.txt)

2. 列名があり, データ間がスペースで区切られている場合

```
> x <- read.table("demo2.txt",  
+                 header=T)  
> x
```

	ID	AGE	GENDER	DATE
1	2	50	1	2001/01/01
2	4	55	2	2002/02/02
3	6	60	2	2003/03/03
4	3	65	1	2004/04/04
5	1	70	2	2005/05/05
6	5	75	1	2006/06/06



```
C:\>  
ID AGE GENDER DATE  
2 50 1 2001/01/01  
4 55 2 2002/02/02  
6 60 2 2003/03/03  
3 65 1 2004/04/04  
1 70 2 2005/05/05  
5 75 1 2006/06/06
```

demo2.txt



データフレームの作成 (.txt)

3. 1行目にコメント, 2行目に列名があり, データ間がスペースで区切られている場合 1行目を読み飛ばす

```
> x <- read.table("demo3.txt",  
+                 header=T, skip=1)  
> x
```

	ID	AGE	GENDER	DATE
1	2	50	1	2001/01/01
2	4	55	2	2002/02/02
3	6	60	2	2003/03/03
4	3	65	1	2004/04/04
5	1	70	2	2005/05/05
6	5	75	1	2006/06/06

```
### 背景情報 ###  
ID AGE GENDER DATE  
2 50 1 2001/01/01  
4 55 2 2002/02/02  
6 60 2 2003/03/03  
3 65 1 2004/04/04  
1 70 2 2005/05/05  
5 75 1 2006/06/06
```

demo3.txt

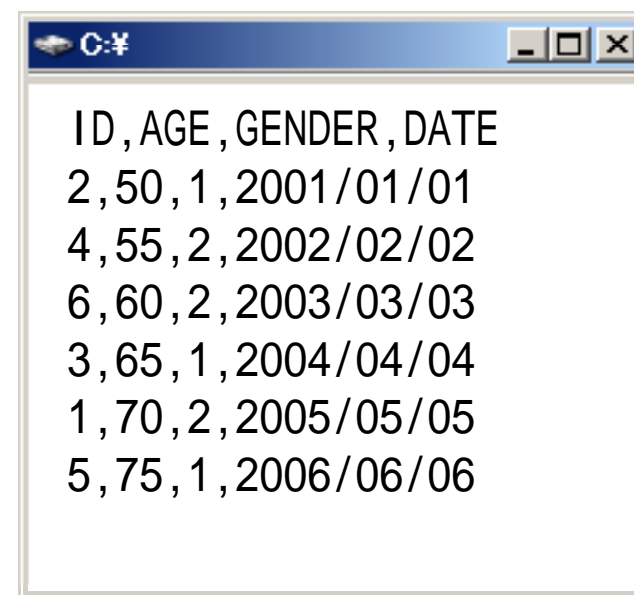


データフレームの作成 (.txt)

4. 列名があり, データ間がコンマで区切られている場合

```
> x <- read.table("demo4.txt",  
+                 header=T, sep=",")  
> x
```

	ID	AGE	GENDER	DATE
1	2	50	1	2001/01/01
2	4	55	2	2002/02/02
3	6	60	2	2003/03/03
4	3	65	1	2004/04/04
5	1	70	2	2005/05/05
6	5	75	1	2006/06/06



```
C:\羊  
ID,AGE,GENDER,DATE  
2,50,1,2001/01/01  
4,55,2,2002/02/02  
6,60,2,2003/03/03  
3,65,1,2004/04/04  
1,70,2,2005/05/05  
5,75,1,2006/06/06
```

demo4.txt



データフレームの作成（ .xls/xlsx ）

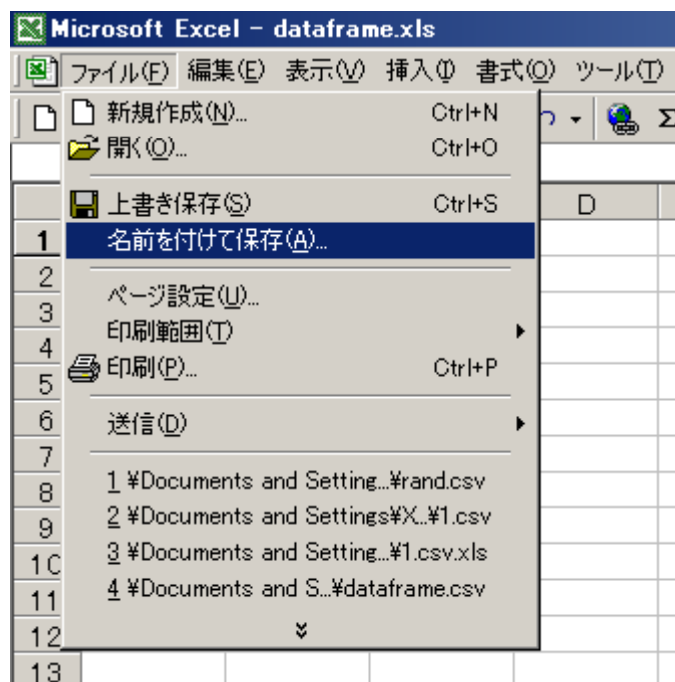
- ▶ EXCEL ファイル（.xls/.xlsx）を R に読み込ませる場合：
 - ▶ .csv ファイルに変換して読み込ませる
 - ▶ EXCEL ファイルをそのまま読み込ませる **パッケージ「xlsx」**

- ▶ .csv ファイルに変換して読み込ませる場合，前もって関数 `read.csv()` で読み込める形式にすること（前節の `demo4.txt` の状態）
 1. まず，EXCEL ファイルを開き，メニューの [ファイル] の [開く] から，[名前をつけて保存] を選択する
 2. 保存する名前をつけた後，次に [ファイルの種類] から [CSV カンマ区切り] を選択して保存する

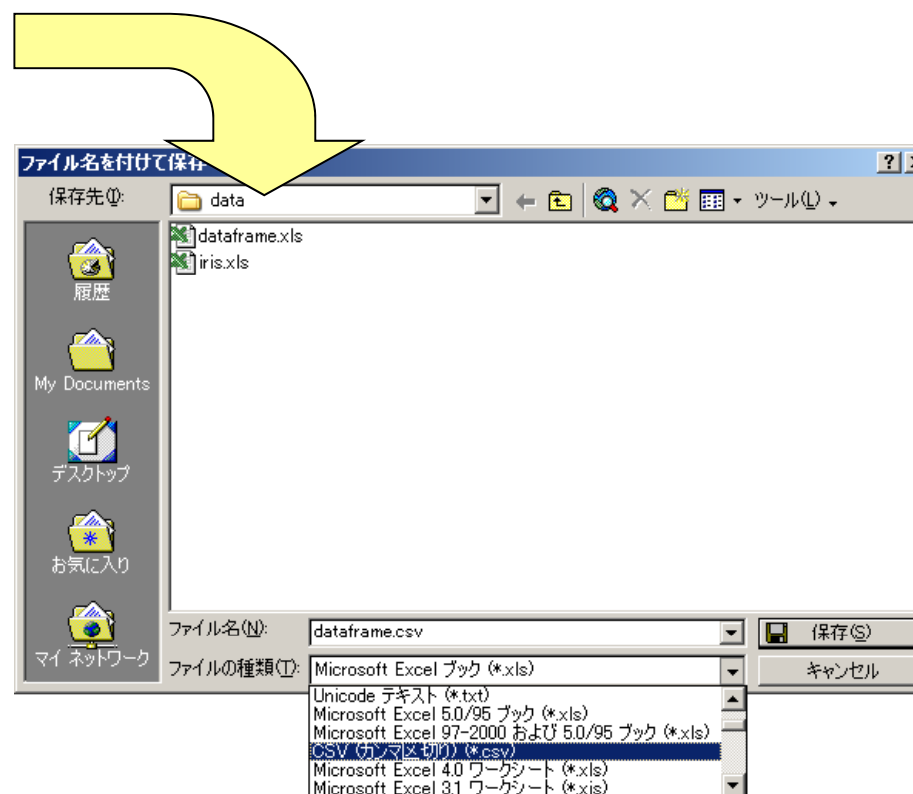


データフレームの作成 (.xls/xlsx)

- ▶ EXCEL を別名で保存 (.csv ファイルとして保存)



別名で保存



CSV (カンマ区切り) で保存

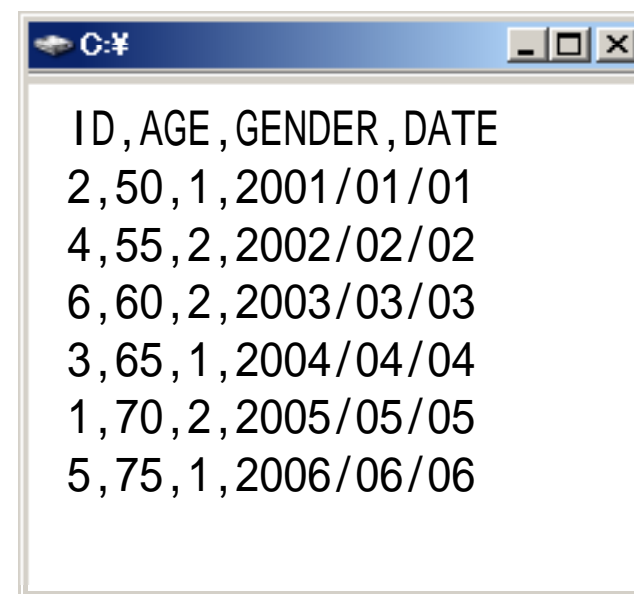


データフレームの作成 (.txt)

4'. 列名があり, データ間がコンマで区切られている場合

```
> x <- read.csv("demo4.csv")  
> x
```

	ID	AGE	GENDER	DATE
1	2	50	1	2001/01/01
2	4	55	2	2002/02/02
3	6	60	2	2003/03/03
4	3	65	1	2004/04/04
5	1	70	2	2005/05/05
6	5	75	1	2006/06/06



```
C:\羊  
ID,AGE,GENDER,DATE  
2,50,1,2001/01/01  
4,55,2,2002/02/02  
6,60,2,2003/03/03  
3,65,1,2004/04/04  
1,70,2,2005/05/05  
5,75,1,2006/06/06
```

demo4.csv



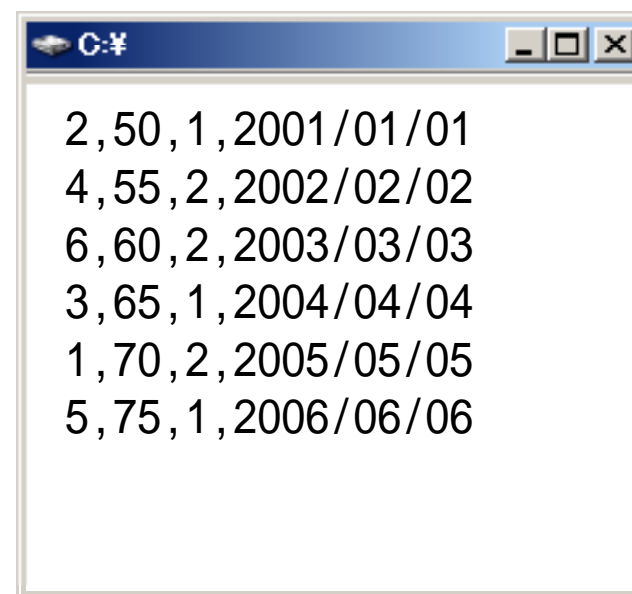
データフレームの作成 (.txt)

5. 列名がなく、データ間がコンマで区切られている場合

先に列名を表す変数を作成した後、関数 `read.csv()` で読み込み

```
> myname <- c("ID", "AGE", "GENDER", "DATE")  
> x <- read.csv("demo5.csv",  
+             header=F, col.names=myname)  
> x
```

	ID	AGE	GENDER	DATE
1	2	50	1	2001/01/01
2	4	55	2	2002/02/02
3	6	60	2	2003/03/03
4	3	65	1	2004/04/04
5	1	70	2	2005/05/05
6	5	75	1	2006/06/06



```
C:\羊  
2,50,1,2001/01/01  
4,55,2,2002/02/02  
6,60,2,2003/03/03  
3,65,1,2004/04/04  
1,70,2,2005/05/05  
5,75,1,2006/06/06
```

demo5.csv



データフレームの作成 (.xlsx)

- ▶ パッケージ `xlsx` の関数 `read.xlsx()` で EXCEL ファイルを読み込む

```
> library(xlsx)
> ( x <- read.xlsx("c:/mydata.xlsx",
+                 sheetName="demo") )
```

```
ID AGE GENDER DATE
1  2  50      1 2001/01/01
2  4  55      2 2002/02/02
3  6  60      2 2003/03/03
4  3  65      1 2004/04/04
5  1  70      2 2005/05/05
6  5  75      1 2006/06/06
```

	A	B	C	D
1	ID	AGE	GENDER	DATE
2	2	50	1	2001/01/01
3	4	55	2	2002/02/02
4	6	60	2	2003/03/03
5	3	65	1	2004/04/04
6	1	70	2	2005/05/05
7	5	75	1	2006/06/06
8				

c:/mydata.xlsx



EXCEL のセルをコピーして作成

- ▶ EXCEL のセルをコピーして、そのまま R に貼り付けることも出来る

	A	B	C	D
1	ID	AGE	GENDER	DATE
2	2	50	1	2001/01/01
3	4	55	2	2002/02/02
4	6	60	2	2003/03/03
5	3	65	1	2004/04/04
6	1	70	2	2005/05/05
7	5	75	1	2006/06/06
8				

コピー

列名をコピーした場合

```
x <- read.delim(pipe("pbpaste"), header=T)
```

列名をコピーしなかった場合

```
x <- read.delim(pipe("pbpaste"), header=F)
```

Mac OS X の場合

列名をコピーした場合

```
x <- read.delim("clipboard", header=T)
```

列名をコピーしなかった場合

```
x <- read.delim("clipboard", header=F)
```

Windows の場合



データフレームの閲覧

- ▶ データフレームの中身を確認したいときは・・・
 - ▶ R のコンソール画面で
 - ▶ R 標準のデータエディタで（ データを見ながらの作業不可）
 - ▶ relimp パッケージのテキストウィンドウで

```
R Console
ファイル 編集 その他 パッケージ ヘルプ
> x
  SEX HEIGHT WEIGHT
1  F    160     50
2  F    165     65
3  M    170     60
4  M    175     55
5  M    180     70
>
```

コンソール上
> x
> head(x)

	SEX	HEIGHT	WEIGHT
1	F	160	50
2	F	165	65
3	M	170	60
4	M	175	55
5	M	180	70

データエディタ
> edit(x)

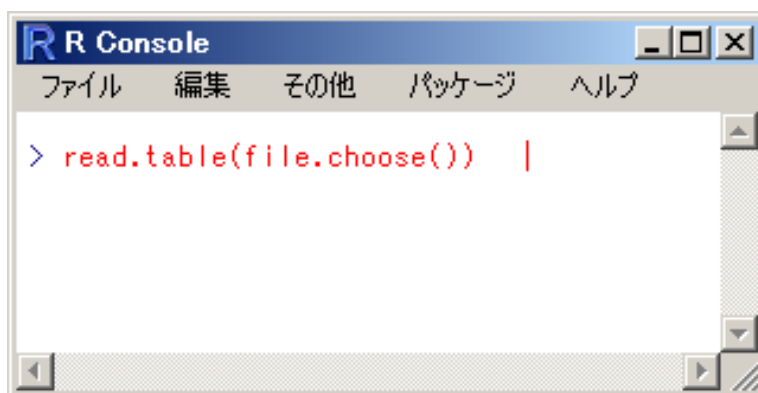
	SEX	HEIGHT	WEIGHT
1	F	160	50
2	F	165	65
3	M	170	60
4	M	175	55
5	M	180	70

テキストウィンドウ
> library(relimp)
> showData(x)



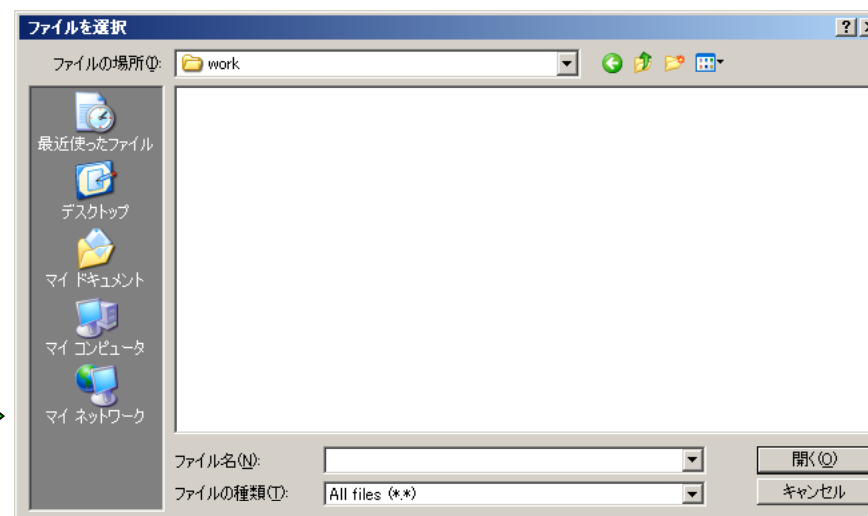
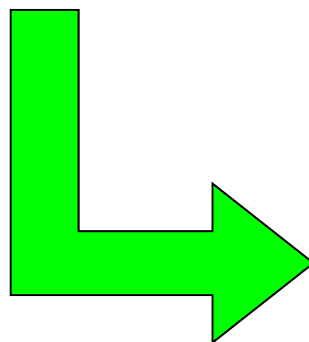
【参考】 データファイルの読み込み

- ▶ Windows 版 R では、関数 `file.choose()` を使用すると、ファイル名を指定するダイアログが表示される



```
R Console  
ファイル 編集 その他 パッケージ ヘルプ  
> read.table(file.choose()) |
```

直接ファイル名を指定せずに
マウスでファイルを指定する
ことが出来るようになる！





【参考】データフレームの作成 (RODBC)

- ▶ パッケージ RODBС 中の関数 [odbcConnectXXXXX\(\)](#) でファイルにアクセスした後、関数 [sql.Query\(\)](#) でデータを読み込むことが出来る

```
> library(RODBC) # パッケージの呼出
> tmp <- odbcConnectExcel("mydata.xls") # データに接続
> tmp <- odbcConnectAccess("mydata.mdb") # (Access の場合)
> sqlTables(tmp) # テーブルを表示
> x <- sqlQuery(tmp, "select * from [demo$]") # 読み込み
> x <- sqlQuery(tmp, "select * from [demo]") # (Access の場合)
> odbcClose(tmp) # 接続を遮断
```

- ▶ 他にも ORACLE のデータベースや、その他のデータ形式ファイル (DBASE, MySQL, PostgreSQL) からデータを読み込むことも可



本日のメニュー

1. R の機能の概要

- ▶ 起動方法 電卓機能 終了方法
- ▶ 行列計算
- ▶ 簡単なシミュレーション
- ▶ グラフ作成

2. データのハンドリング

- ▶ パッケージと作業ディレクトリについて
- ▶ データの読み込み

- ▶ 重回帰や Cox 回帰用のデータの作成例



場面設定と使用するデータの概要

- ▶ 糖尿病予備軍の患者さんに A または B の糖尿病予防薬を投与し、投与終了日における HbA1c (6.5%以上であれば糖尿病) を測定し、治療効果を確認する
- ▶ データフレーム「demo」 <C:/demo.txt> にデータあり
 - ▶ ID：患者さんを表す番号
 - ▶ AGE：年齢（歳），数値
 - ▶ GENDER：性別（1：男性，2：女性），数値
 - ▶ DATE：薬剤の投与開始日，文字列
- ▶ データフレーム「hba1c」 <C:/hba1c.txt> にデータあり
 - ▶ ID：患者さんを表す番号
 - ▶ GROUP：投与される薬剤の種類，文字（A，B）
 - ▶ HBA1C：薬剤の投与終了日に測定したHbA1c（%），数値
 - ▶ DATE：薬剤の投与終了日，文字列



データフレーム「demo」の準備

- ▶ 患者さんの背景データ「demo.txt」を読み込み

列名があり，データ間がコンマで区切られている

```
> demo <- read.table("C:/demo.txt",  
+                    header=T, sep=",")  
> demo
```

	ID	AGE	GENDER	DATE
1	2	50	1	2001/01/01
2	4	55	2	2002/02/02
3	6	60	2	2003/03/03
4	3	65	1	2004/04/04
5	1	70	2	2005/05/05
6	5	75	1	2006/06/06

```
C:\>  
ID,AGE,GENDER,DATE  
2,50,1,2001/01/01  
4,55,2,2002/02/02  
6,60,2,2003/03/03  
3,65,1,2004/04/04  
1,70,2,2005/05/05  
5,75,1,2006/06/06
```

demo.txt



データフレーム「hba1c」の準備

- ▶ 患者さんの臨床検査データ「hba1c.txt」を読み込み
列名があり，データ間がコンマで区切られている

```
> hba1c <- read.table("C:/hba1c.txt",  
+                      header=T, sep=",")  
> hba1c
```

ID	GROUP	HBA1C	DATE
1	1	A	2007/07/07
2	2	B	2008/08/08
3	3	A	2009/09/09
4	4	B	2010/10/10
5	5	B	2011/11/11

```
C:¥  
ID, GROUP, HBA1C, DATE  
1, A, 6.6, 2007/07/07  
2, B, 7.0, 2008/08/08  
3, A, 5.7, 2009/09/09  
4, B, 7.5, 2010/10/10  
5, B, 6.4, 2011/11/11
```

hba1c.txt



作成したいデータのイメージ（目標）

	ID	GROUP	AGE	AGE_CT	GENDER	HBA1C	EVENT	DAY
1	1	A	70	>=65	Female	6.6	1	793
2	2	B	50	<65	Male	7.0	1	2776
3	3	A	65	>=65	Male	5.7	0	1984
4	4	B	55	<65	Female	7.5	1	3172
5	5	B	75	>=65	Male	6.4	0	1984

- ① 「demo」をIDが小さい順に並べ替え（ソート）
- ② 年齢（AGE）が「65歳未満」「65歳以上」を表す変数 AGE_CT を作成
- ③ 性別の変数を因子型（カテゴリ）に変換
- ④ HbA1c が 6.5 未満（0）か 6.5 以上（1）かを表す変数 EVENT を作成
- ⑤ 「demo」と「hba1c」をくっつけ、両方に存在するレコードのみ残す
- ⑥ DAY（投与終了日－投与開始日）を作成
- ⑦ 変数を上記に絞り、変数の順番も上記に従う



【道具】 データへのアクセス方法

データフレーム x に対する命令	機能
<code>x\$列名, x["列名"], x[["列名"]]</code>	指定した列データを表示
<code>x[2], x[[2]]</code>	2 番目の列データを表示
<code>x[3, 2], x[[3, 2]]</code>	3 行 2 列目のデータを表示
<code>x[[3,"列名"]], x[[3,"列名"]]</code>	指定した列の 3 行目のデータを表示
<code>x[c(1, 2)]</code>	1 列目と 2 列目のデータを表示
<code>x[c(3, 4),]</code>	3 行目と 4 行目のデータを表示
<code>x[,c(T,F,T)]</code>	論理ベクトル <code>c(T,F,T)</code> が TRUE となっている列を表示
<code>x[GENDER==2,]</code>	性別が 2 (女性) である行を表示
<code>x[,GENDER==2 & AGE>60]</code>	性別が 2 (女性) かつ年齢が 60 歳より大きい行を表示
<code>subset(x, gender==2 & age>60)</code>	<code>x[,GENDER==2 & AGE>60]</code> と同様の機能



【道具】 データの加工・抽出

データフレーム x に対する命令	機能
<code>head(x, n=a)</code>	先頭から a 行だけ抽出する
<code>tail(x, n=b)</code>	末尾から b 行だけ抽出する
<code>na.omit(x)</code>	NA を含む行を削除する
<code>transform(x, y=ベクトル)</code>	データフレーム x に新たな列 y を追加する
<code>subset(x, 条件式)</code>	条件式に合う行のみを抽出する
<code>subset(x, 条件式, ベクトル)</code>	ベクトルで指定した列に対し, 条件式に合う行のみを抽出する
<code>reshape(x, ...)</code>	データフレーム x を横展開/縦展開する
<code>apply(x[,範囲], 1, 関数)</code>	データフレーム x の指定した範囲について, 各行ごとに関数を適用する (各列ごと: <code>apply(x[,範囲], 2, 関数)</code> とする)



【道具】 データの結合など

データフレーム x に対する命令	機能
<code>ncol(x)</code>	x の列数 (変数の数) を求める
<code>nrow(x)</code>	x の行数 (データ数) を求める
<code>names(x)</code>	x の列名を表示する
<code>rbind(x,y)</code>	x と y を縦に並べて結合する
<code>cbind(x,y)</code>	x と y を横に並べて結合する
<code>data.frame(x,y)</code>	x と y を横に並べて結合する
<code>merge(x,y)</code>	x と y をくっつける (マージする) all=T を指定するとデータを全て残す all=T を指定しなければデータの共通部分が結果として返される

たまに関数 `attach()` や `detach()` を使用している資料が見受けられるが、使わない方が良い (実際にデータ解析を行う場合ではまず使わないです)

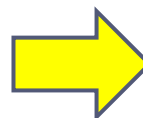


① データフレーム「demo」の処理

- ▶ ID が小さい順に並べ替え（ソート）を行う
- ▶ そのままだと行番号がバラバラ

```
> sortlist <- order(demo$ID)           # 順番を取得  
> demo      <- demo[sortlist,]         # 整列
```

	ID	AGE	GENDER	DATE
1	2	50	1	2001/01/01
2	4	55	2	2002/02/02
3	6	60	2	2003/03/03
4	3	65	1	2004/04/04
5	1	70	2	2005/05/05
6	5	75	1	2006/06/06



	ID	AGE	GENDER	DATE
5	1	70	2	2005/05/05
1	2	50	1	2001/01/01
4	3	65	1	2004/04/04
2	4	55	2	2002/02/02
6	5	75	1	2006/06/06
3	6	60	2	2003/03/03

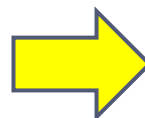


① データフレーム「demo」の処理

- ▶ 行番号がバラバラなので、行番号を整形する

```
> sortlist <- order(demo$ID)           # 順番を取得  
> demo      <- demo[sortlist,]         # 整列  
> rownames(demo) <- c(1:nrow(demo)) # 行番号の整形
```

	ID	AGE	GENDER	DATE
5	1	70	2	2005/05/05
1	2	50	1	2001/01/01
4	3	65	1	2004/04/04
2	4	55	2	2002/02/02
6	5	75	1	2006/06/06
3	6	60	2	2003/03/03



	ID	AGE	GENDER	DATE
1	1	70	2	2005/05/05
2	2	50	1	2001/01/01
3	3	65	1	2004/04/04
4	4	55	2	2002/02/02
5	5	75	1	2006/06/06
6	6	60	2	2003/03/03



② データフレーム「demo」の処理

- ▶ 年齢（AGE）が「65歳未満」「65歳以上」を表す変数 AGE_CT を作成

```
> tmp <- ifelse(demo$AGE<65, "<65", ">=65") # 変数 tmp に退避  
> demo <- transform(demo, AGE_CT=tmp)      # demo に変数追加
```

ID	AGE	GENDER	DATE	ID	AGE	GENDER	DATE	AGE_CT
1	70	2	2005/05/05	1	70	2	2005/05/05	>=65
2	50	1	2001/01/01	2	50	1	2001/01/01	<65
3	65	1	2004/04/04	3	65	1	2004/04/04	>=65
4	55	2	2002/02/02	4	55	2	2002/02/02	<65
5	75	1	2006/06/06	5	75	1	2006/06/06	>=65
6	60	2	2003/03/03	6	60	2	2003/03/03	<65



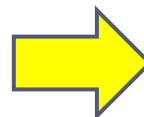
③ データフレーム「demo」の処理

- ▶ 性別の変数を因子型に変換

1 を Male (男性), 2 を Female (女性) なるカテゴリとして扱う

```
> demo$GENDER <- factor(demo$GENDER, levels=c(1,2),  
+                        labels=c("Male", "Female"))  
> demo$GENDER <- ordered(demo$GENDER) # 順序関係をつける場合
```

ID	AGE	GENDER	DATE	AGE_CT
1	70	2	2005/05/05	>=65
2	50	1	2001/01/01	<65
3	65	1	2004/04/04	>=65
4	55	2	2002/02/02	<65
5	75	1	2006/06/06	>=65
6	60	2	2003/03/03	<65



ID	AGE	GENDER	DATE	AGE_CT
1	70	Female	2005/05/05	>=65
2	50	Male	2001/01/01	<65
3	65	Male	2004/04/04	>=65
4	55	Female	2002/02/02	<65
5	75	Male	2006/06/06	>=65
6	60	Female	2003/03/03	<65

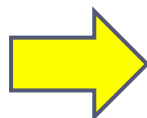


④ データフレーム「hba1c」の処理

- ▶ HbA1c が 6.5 未満 (0) か 6.5 以上 (1) を表す変数 **EVENT** を追加
1: イベント発生 (糖尿病発症), 0: イベントなし

```
> hba1c$EVENT <- ifelse(hba1c$HBA1C<6.5, 0, 1)
```

ID	GROUP	HBA1C	DATE
1	A	6.6	2007/07/07
2	B	7.0	2008/08/08
3	A	5.7	2009/09/09
4	B	7.5	2010/10/10
5	B	6.4	2011/11/11



ID	GROUP	HBA1C	DATE	EVENT
1	A	6.6	2007/07/07	1
2	B	7.0	2008/08/08	1
3	A	5.7	2009/09/09	0
4	B	7.5	2010/10/10	1
5	B	6.4	2011/11/11	0



○ 「demo」と「hba1c」をくっつける前の処理

- ▶ 「hba1c」の中の変数 DATE を DATE_END に変更する
「demo」の中の変数 DATE とゴッチャにならないように

```
> hba1c <- transform(hba1c, DATE_END=DATE) # DATE_END にコピー  
> hba1c$DATE <- NULL # DATE を削除
```

ID	GROUP	HBA1C	DATE	EVENT	ID	GROUP	HBA1C	EVENT	DATE_END
1	A	6.6	2007/07/07	1	1	A	6.6	1	2007/07/07
2	B	7.0	2008/08/08	1	2	B	7.0	1	2008/08/08
3	A	5.7	2009/09/09	0	3	A	5.7	0	2009/09/09
4	B	7.5	2010/10/10	1	4	B	7.5	1	2010/10/10
5	B	6.4	2011/11/11	0	5	B	6.4	0	2011/11/11



⑤ 「demo」と「hba1c」をくっつける

- ▶ 「demo」と「hba1c」をくっつけ（マージ），両方に存在するレコードのみ残す 出来あがったデータセットを「full」とする

```
> full <- merge(demo, hba1c, by="ID", all=F, sort=T)
```

ID	AGE	GENDER	DATE	AGE_CT	GROUP	HBA1C	EVENT	DATE_END
1	70	Female	2005/05/05	>=65	A	6.6	1	2007/07/07
2	50	Male	2001/01/01	<65	B	7.0	1	2008/08/08
3	65	Male	2004/04/04	>=65	A	5.7	0	2009/09/09
4	55	Female	2002/02/02	<65	B	7.5	1	2010/10/10
5	75	Male	2006/06/06	>=65	B	6.4	0	2011/11/11
6	60	Female	2003/03/03	<65				



⑥ データフレーム「full」の処理

- ▶ DAY（投与終了日－投与開始日）を作成する

```
> day <- as.Date(full$DATE_END, format="%Y/%m/%d") -  
+       as.Date(full$DATE, format="%Y/%m/%d") # 日数の差  
> day <- as.numeric(day) # 数値に変換  
> full <- transform(full, DAY=day) # 変数を追加
```

ID	AGE	GENDER	DATE	AGE_CT	GROUP	HBA1C	EVENT	DATE_END	DAY
1	70	Female	2005/05/05	>=65	A	6.6	1	2007/07/07	793
2	50	Male	2001/01/01	<65	B	7.0	1	2008/08/08	2776
3	65	Male	2004/04/04	>=65	A	5.7	0	2009/09/09	1984
4	55	Female	2002/02/02	<65	B	7.5	1	2010/10/10	3172
5	75	Male	2006/06/06	>=65	B	6.4	0	2011/11/11	1984



⑦ データフレーム「full」の処理

- ▶ 変数を「ID, GROUP, AGE, AGE_CT, GENDER, HBA1C, EVENT, DAY」に
絞り，変数の順番も整える（3つの方法あり） **完成っ！**

```
> full <- full[,c(1,6,2,5,3,7,8,10)]  
> full <- full[,c("ID", "GROUP", "AGE", "AGE_CT", "GENDER", "HBA1C", "EVENT", "DAY")]  
> full <- subset(full, select=c(ID, GROUP, AGE, AGE_CT, GENDER, HBA1C, EVENT, DAY))
```

ID	GROUP	AGE	AGE_CT	GENDER	HBA1C	EVENT	DAY
1	A	70	>=65	Female	6.6	1	793
2	B	50	<65	Male	7.0	1	2776
3	A	65	>=65	Male	5.7	0	1984
4	B	55	<65	Female	7.5	1	3172
5	B	75	>=65	Male	6.4	0	1984



解析用データが出来あがった後は・・・

- ▶ 年齢 (*AGE_CT*) と性別 (*GENDER*) のクロス集計

```
> result <- table(full$AGE_CT, full$GENDER)
```

```
> addmargins(result, 1:2)
```

	Male	Female	Sum
<65	1	1	2
>=65	2	1	3
Sum	3	2	5

```
> prop.table(result, 1)
```

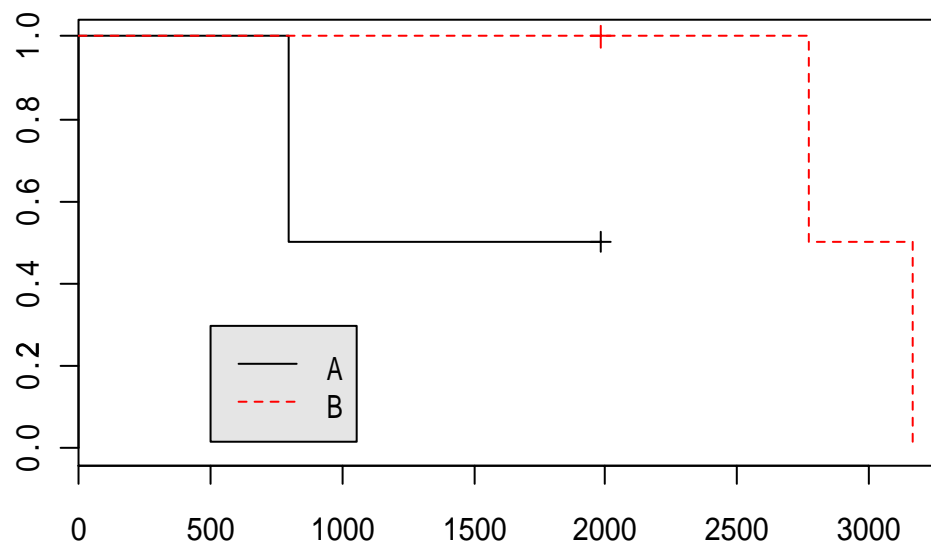
	Male	Female
<65	0.5000000	0.5000000
>=65	0.6666667	0.3333333



解析用データが出来あがった後は・・・

- ▶ 各薬剤のイベント発現率（カプラン・マイヤー法による推定）

```
> library(survival)
> result <- survfit(Surv(DAY,EVENT) ~ GROUP, data=full)
> plot(result, lty=1:2, col=1:2)
> legend(500,0.3,c("A", "B"),col=1:2,lty=1:2,ncol=1,bg='gray90')
```





解析用データが出来あがった後は . . .

- ▶ Cox 回帰：イベント発生までの期間 = 薬剤

```
> coxph(Surv(DAY,EVENT) ~ GROUP, data=full)
```

Call:

```
coxph(formula = Surv(DAY, EVENT) ~ GROUP, data = full)
```

	coef	exp(coef)	se(coef)		z	p
GROUPB	-21.7	3.82e-10	41772	-0.000519	1	

Likelihood ratio test=1.83 on 1 df, p=0.176 n= 5,
number of events= 3



解析用データが出来あがった後は . . .

▶ 他にも

- ▶ 一変数の要約統計量, グラフ
- ▶ 二変数間の関係を数値化, グラフ
- ▶ 重回帰分析
- ▶ ロジスティック回帰分析

等々...



本日のメニュー

1. R の機能の概要

- ▶ 起動方法 電卓機能 終了方法
- ▶ 行列計算
- ▶ 簡単なシミュレーション
- ▶ グラフ作成

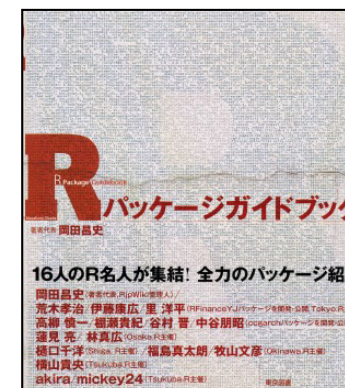
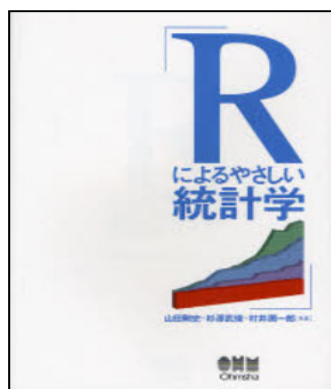
2. データのハンドリング

- ▶ パッケージと作業ディレクトリについて
- ▶ データの読み込み
- ▶ 重回帰や Cox 回帰用のデータの作成例



参考文献

- ▶ 山田 剛史 他著 (2008) 「R によるやさしい統計学 (オーム社)」
- ▶ Phil Spector 著, 石田 基広 他翻訳 (2008)
「R データ自由自在 (シュプリンガー・ジャパン)」
- ▶ 舟尾 暢男 (2009) 「The R Tips 第2版 (オーム社)」
- ▶ 岡田 昌史 他著 (2011) 「R パッケージガイドブック (東京図書)」





【宣伝】 R の統合開発環境「RStudio」

The screenshot displays the RStudio integrated development environment. The top-left pane shows R code for a simulation and data manipulation. The bottom-left pane shows the console output, including the execution of the 'survival' package and the creation of survival plots for two groups, A and B. The right pane shows a survival plot with two curves: a solid black line for group A and a dashed red line for group B. The x-axis represents time (0 to 3000) and the y-axis represents survival probability (0.0 to 1.0).

```
1 nt <- 10000
2 pt <- 0
3 for(kk in 1:nt){
4   ttest1 <- t.test(rnorm(500, mean = 0, sd = 1), mu = 0)
5   pp <- ttest1$p.value
6   if (pp < 0.05){
7     pt <- pt + 1
8   }
9 }
10 pt <- pt/nt
11 print(pt)
12
13
```

```
tmp <- ifelse(demo$AGE<65, "<65", ">=65") # 変数 tmp に退避
demo <- transform(demo, AGE_CT=tmp) # demo に変数追加
demo$GENDER <- factor(demo$GENDER, levels=c(1,2), labels=
hba1c$EVENT <- ifelse(hba1c$HBA1C<6.5, 0, 1)
hba1c <- transform(hba1c, DATE_END=DATE) # DATE_END にコピ
hba1c$DATE <- NULL # DATE を削除
full <- merge(demo, hba1c, by="ID", all=F, sort=T)
day <- as.Date(full$DATE_END, format="%Y/%m/%d") -
as.Date(full$DATE, format="%Y/%m/%d") # 日数の差
day <- as.numeric(day) # 数値に変換
full <- transform(full, DAY=day) # 変数を追加
full <- full[, c(1,6,2,5,3,7,8,10)]
```

```
> library(survival)
要求されたパッケージ splines をロード中です
> result <- survfit(Surv(DAY,EVENT) ~ GROUP, data=full)
> plot(result, lty=1:2, col=1:2)
> legend(500, 0.2, c("A","B"), col=1:2, lty=1:2, ncol=1,
bg='gray90')
>
> library(survival)
> result <- survfit(Surv(DAY,EVENT) ~ GROUP, data=full)
> plot(result, lty=1:2, col=1:2)
> legend(500, 0.3, c("A","B"), col=1:2, lty=1:2, ncol=1,
bg='gray90')
>
>
```

- ▶ フリー
- ▶ 日本語対応
- ▶ 対応 OS :
 - ▶ Windows XP/Vista/7
 - ▶ Mac OS X 10.5+
 - ▶ Debian6+/Ubuntu 10.04+ (32-bit, 64-bit)
 - ▶ Fedora13+/openSUSE 11.4+ (32-bit, 64-bit)
- ▶ 詳しくは・・・
<http://rstudio.org/>

統計解析ソフト R の活用
～ R 入門 & R でデータハンドリング～

終



おまけ

1. この資料で出てきたプログラム・データの在り処
2. R のセットアップ (Ver. 2.14.1) のメモ
 - ▶ Windows 版 R
 - ▶ Mac OS X 版 R
 - ▶ Linux 版 R
 - ▶ ソースからビルドする方法



この資料で出てきたプログラム・データの在り処

□第19回疫学セミナー「統計解析ソフトRの活用」

ホームページ

<http://jeaweb.jp/soukai/no22/kanren/index.html>

■この資料で出てきたプログラム・データ

http://www.occn.zaq.ne.jp/cuhxr802/epi_data.zip



インストール [Windows 版 R の場合]

- ▶ CRAN (筑波大学) からダウンロード

<http://cran.md.tsukuba.ac.jp/bin/windows/base/rpatched.html>

R-2.14.1 Patched build for
Windows (32/64 bit)

[Download R-2.14.1 Patched build for Windows \(46 megabytes, 32/64 bit\)](#)

[Installation and other instructions](#)

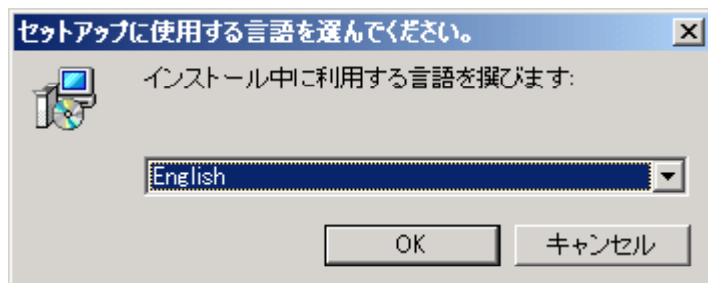
New features in this version: [Windows specific](#), [all platforms](#).

ここ

- ▶ ダウンロードしたファイル R-2.14.1patched-win.exe をダブルクリック
(Vista / 7 の方は「右クリック 管理者権限として実行」)

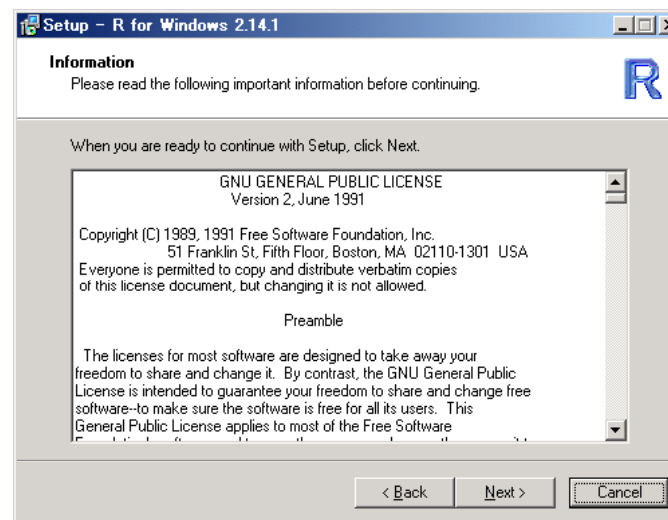
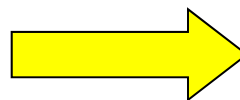
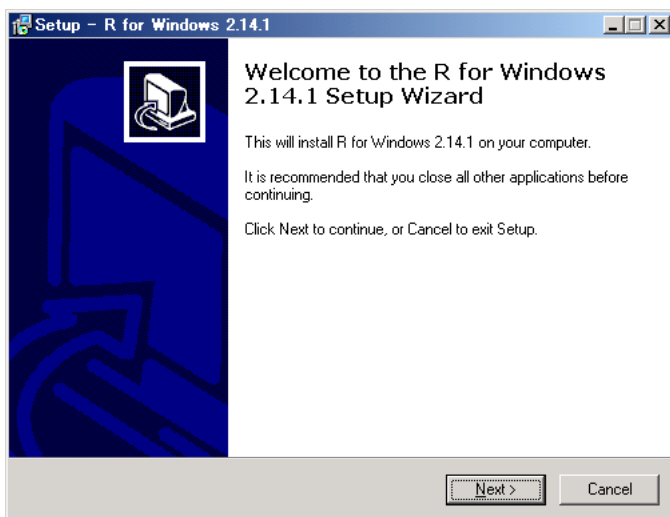
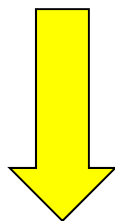


インストール [Windows 版 R の場合]



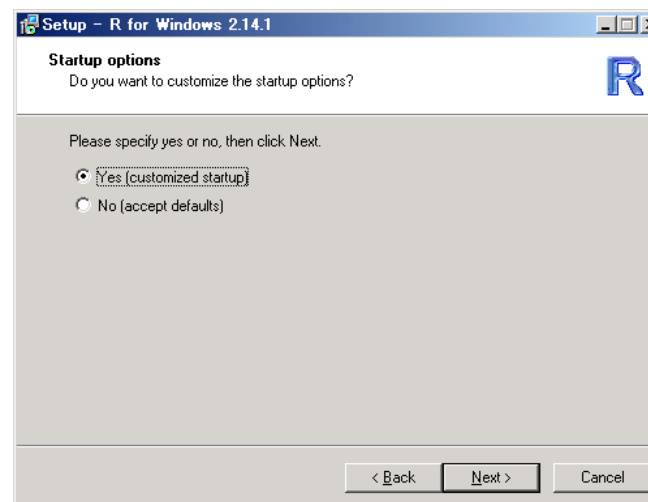
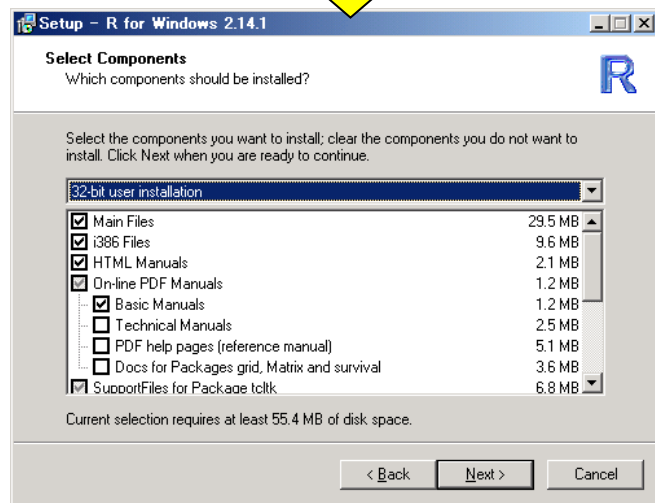
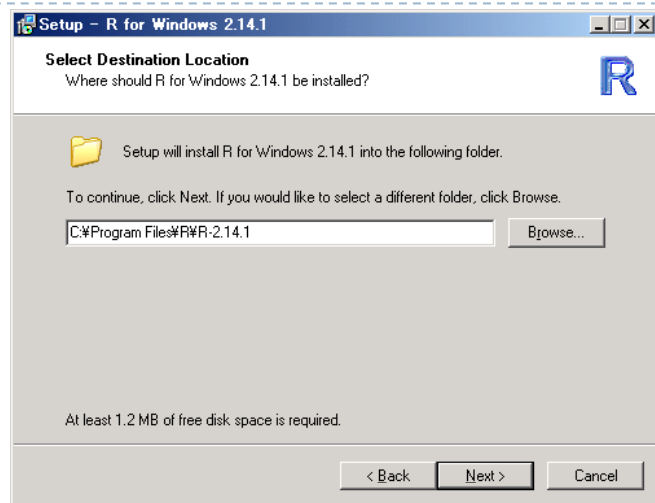
インストール中に使用する言語を
「English」に変更！！
(何もしないと文字化けします)

以降は「Next >」をクリックし続ける





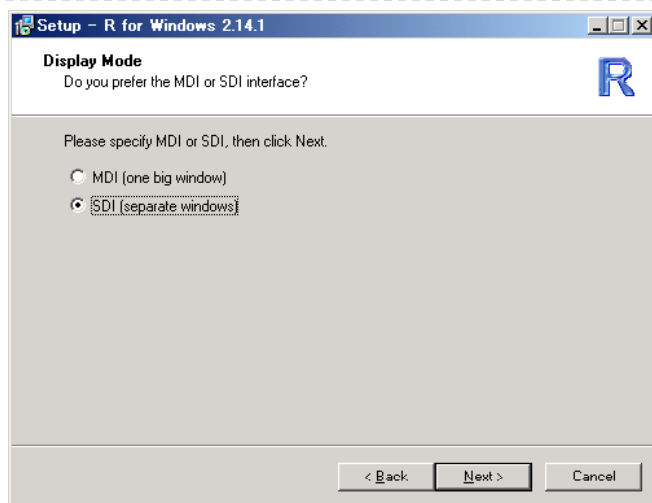
インストール [Windows 版 R の場合]



「Yes」を選択 (カスタマイズする)



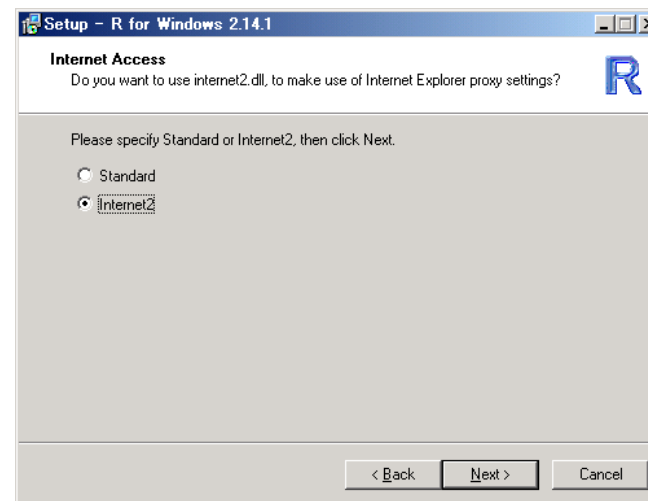
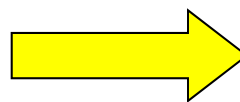
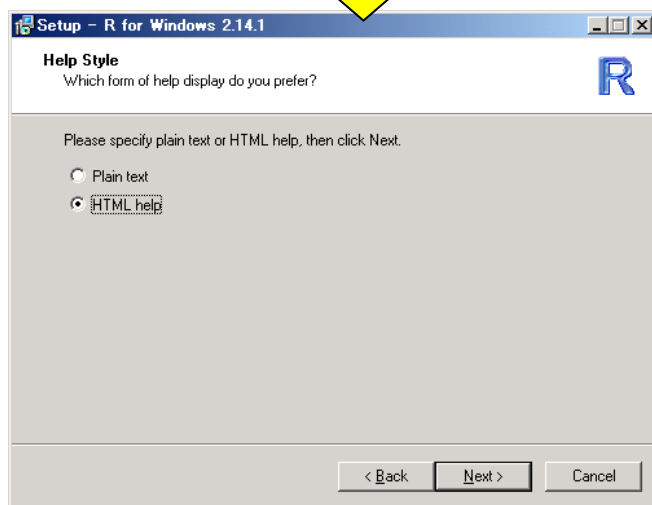
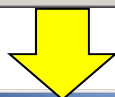
インストール [Windows 版 R の場合]



「SDI（各ウィンドウを分離）」

「ヘルプは HTML 形式」

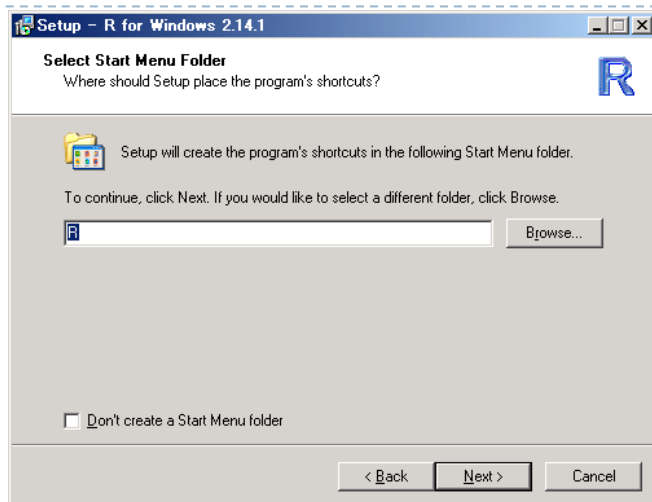
がお勧め



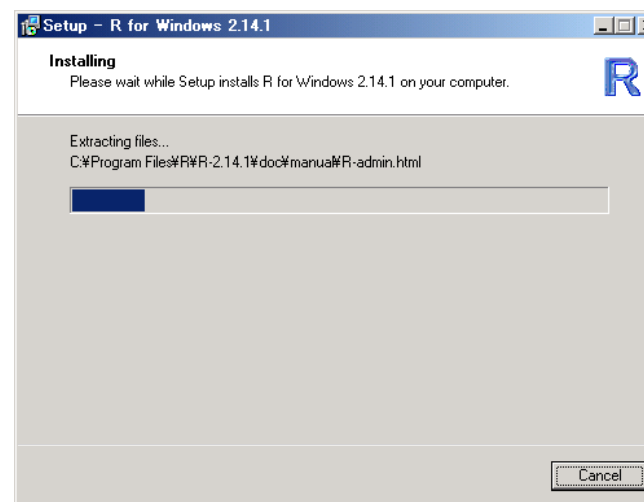
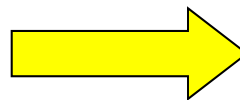
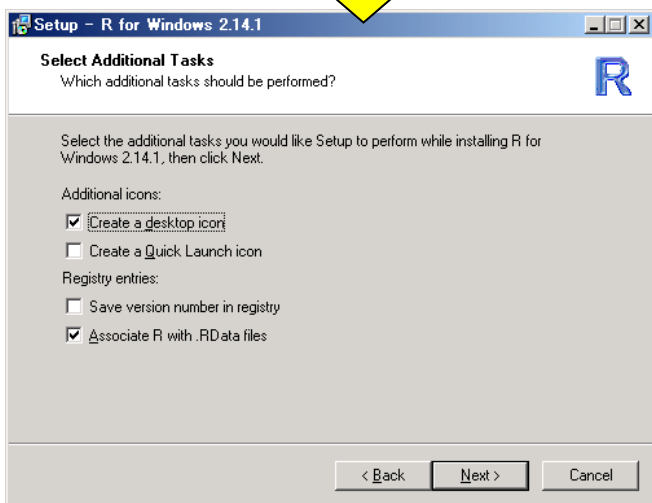
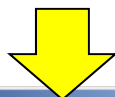
会社で R を使っている場合は
「internet2」を選択しておく



インストール [Windows 版 R の場合]



後は「Next >」をクリックし続けると
インストールが完了する





インストール〔Windows 版 R の場合〕

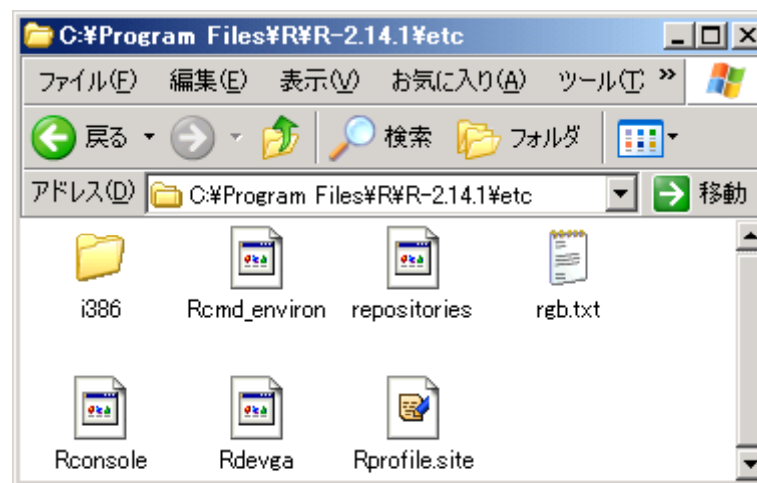
- ▶ 「Rconsole」 「Rdevga」 「Rprofile.site」

<http://cwoweb2.bai.ne.jp/~jgb11101/files/Rconsole>

<http://cwoweb2.bai.ne.jp/~jgb11101/files/Rdevga>

<http://cwoweb2.bai.ne.jp/~jgb11101/files/Rprofile.site>

をダウンロードして、[C:¥Program Files¥R¥R-2.14.1¥etc]にある同名ファイルに上書き 文字化け防止策！



- ★ ダウンロードするときには拡張子が「.txt」になるのでご注意ください
- 1 「Rconsole」 「Rdevga」は拡張子がありません
- 2 「Rprofile.site」の拡張子は「.site」です



インストール [Mac OS X 版 R の場合]

- ▶ CRAN (筑波大学) から R-2.14.1.pkg をダウンロード インストール

<http://cran.md.tsukuba.ac.jp/bin/macosx/>

R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.5 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) [here](#). Releases for old Mac OS X systems (through Mac OS X 10.4) can be found in the [old](#) directory.

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

Universal R 2.14.1 released on 2012/01/04

This binary distribution of R and the GUI supports PowerPC (32-bit) and Intel (32-bit and 64-bit) based Macs on Mac OS X 10.5 (Leopard), 10.6 (Snow Leopard) and 10.7 (Lion). It is possibly the last distribution supporting Mac OS X 10.5 (Leopard) and PowerPC architecture.

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
md5 R-2.14.1.pkg
in the *Terminal* application to print the MD5 checksum for the R-2.14.1.pkg image.

Files:

R-2.14.1.pkg (latest version) MD5-hash: <code>af80add70b331f55136db7bc479d85</code> (ca. 62MB)	Three-way universal binary of R 2.14.1 for Mac OS X 10.5 (Leopard) and higher. Contains R 2.14.1 framework, R.app GUI 1.43 in 32-bit and 64-bit. The above file is an Installer package which can be installed by double-clicking. Depending on your browser, you may need to press the control key and click on this link to download the file.
---	---

CC



インストール〔Mac OS X 版 R の場合〕

- ▶ 「Rprofile.site」をダウンロード

<http://cwoweb2.bai.ne.jp/~jgb11101/files/mac/Rprofile.site>

- ▶ ダウンロードしたファイルを R の「etc」フォルダに保存
文字化け防止策！

場所：[Macintosh HD] [ライブラリ] [Frameworks]
 [R.framework] [Resources] [etc]



インストール〔Linux 版 R の場合〕

- ▶ CRAN（筑波大学）から当該 OS のファイルをダウンロード

<http://cran.md.tsukuba.ac.jp/bin/linux/>

Index of /bin/linux			
<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 debian/	01-Nov-2011 14:20	-	
 redhat/	26-Nov-2009 02:01	-	
 suse/	15-Apr-2011 15:47	-	
 ubuntu/	18-Oct-2011 11:01	-	

- ▶ ルート権限になり apt-get（Fedora は yum）でインストールする

```
$ sudo apt-get update
```

```
$ sudo apt-get install r-base
```

```
$ su
```

```
# yum install r-base
```



インストール〔ソースからビルドする場合〕

- ▶ ルート権限になった後，下記から R-2.14.1.tar.gz をダウンロード
<http://cran.md.tsukuba.ac.jp/src/base/R-2/R-2.14.1.tar.gz>
- ▶ 「R HOME」なるディレクトリを作成し，ここで R-2.14.1.tar.gz を解凍
- ▶ Terminal 上で「R HOME」に移動し，下記コマンドを実行する。

```
$ ./configure
```

```
$ make
```

```
$ make install
```

Linux 環境によっては「f77 (g77で代替化)」「readline」「gcc」「c++」「XOrg-devel」「readline-devel」等がない場合がある。その場合はパッケージを事前にインストールする。「./configure」実行時にエラーが出た場合はログファイル「config.log」の中身を確認して足りないライブラリやヘッダを調査してインストールし，再度「./configure」を実行する