# R講習会

２０１７年６月６日　１０~１２時

資源管理研究センター　資源管理グループ　市野川桃子

# Rとは？

- 統計言語であるSの思想に基づいて開発されたフリーのソフトウェア
- ウェブから誰でも無料で入手できる（http://www.r-project.org/）
- 多様なプラットフォームに対応
  - Unix系OS、Mac OS X, Windows
- 豊富なパッケージ群
  - パッケージ：共通の目的を達成するための関数群

# 一般的な漁業データ解析とそれに必要なソフトウェア

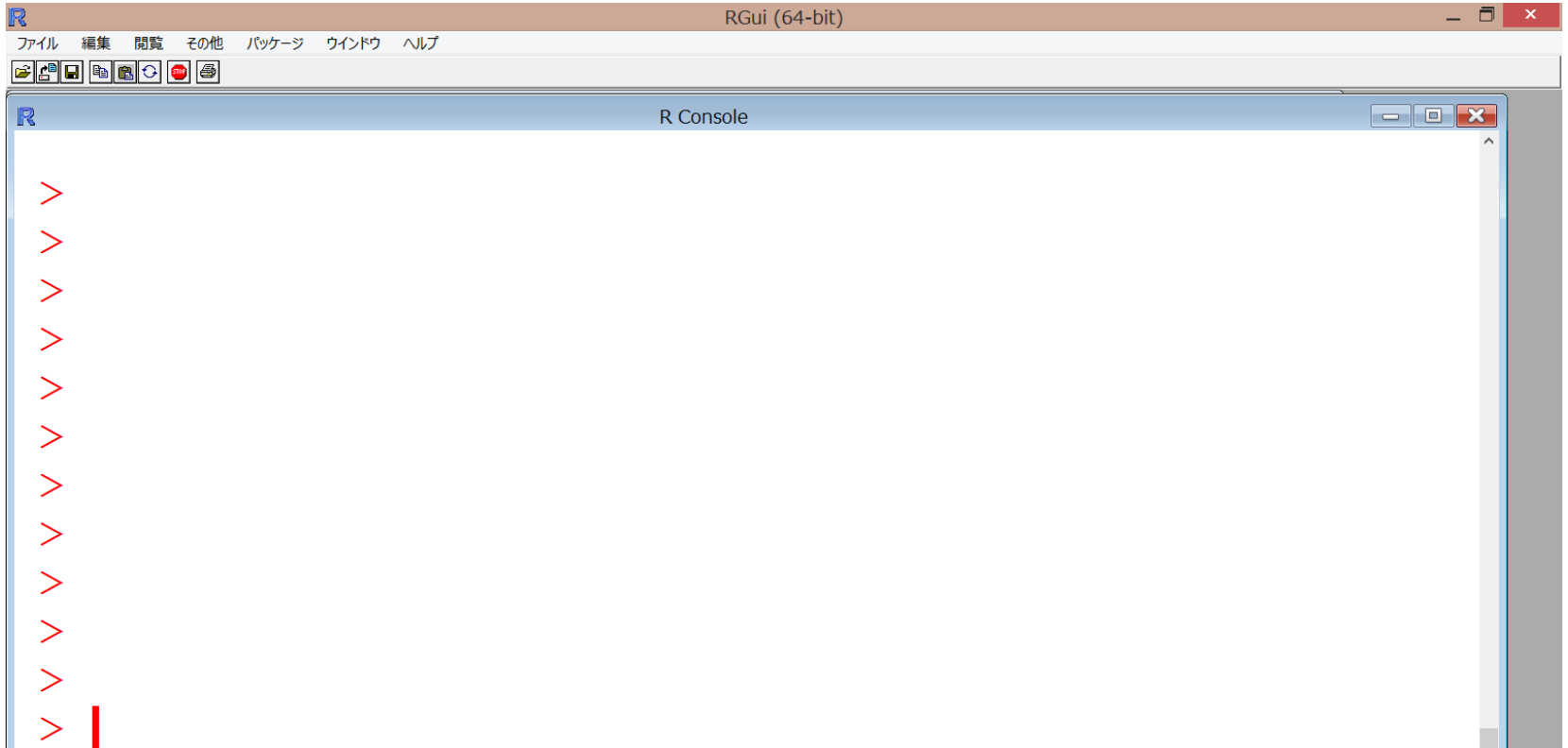
- 漁獲データの可視化,グラフ作成　→　Excelなどの表計算ソフト
- 地図データの解析　→　ArcView, GMTなど
- CPUE標準化等、統計的解析　→　統計ソフト。SAS, S-Plus..
- パラメータ推定　→　Excelのソルバー、AD model builder

→ **多くのソフトは有料，ソフト間でのデータの連携が煩雑**

→ **Rは無料＆同一のソフト上で一気に解析できる**

# ただし

- コマンドラインによるコマンドの入力（プログラミング）
- エクセルのような直感的な操作はできない
- ある程度の「慣れ」が必要



# R実習流れ

1. まず使ってみよう
   - Rでのコマンド入力に慣れる

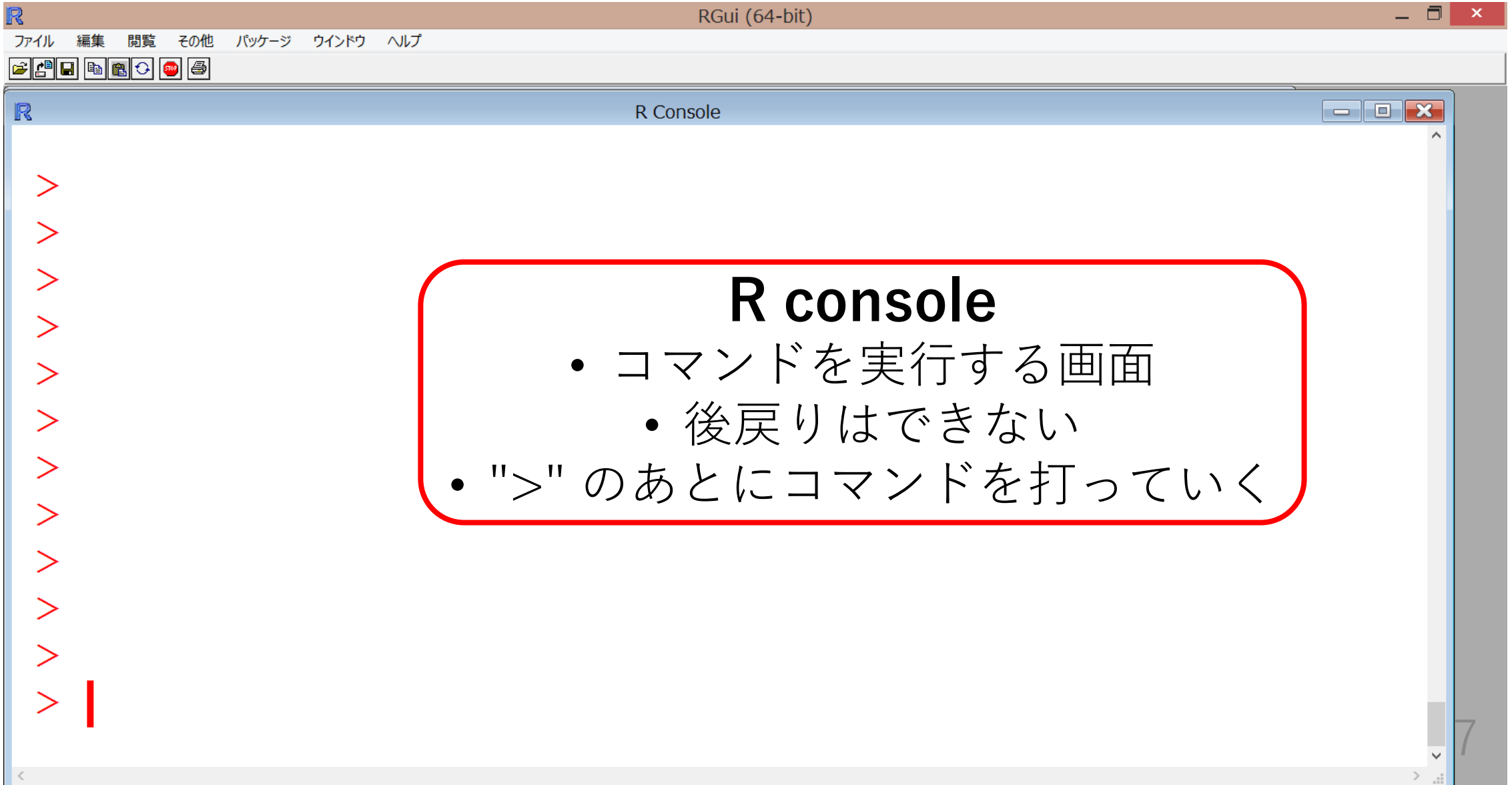
2. 外部との連携
   - データ入出力やパッケージのインストール

3. いろいろやってみよう
   - 配布データを読み込んで，グラフを書いたり，統計解析をしてみたりしよう

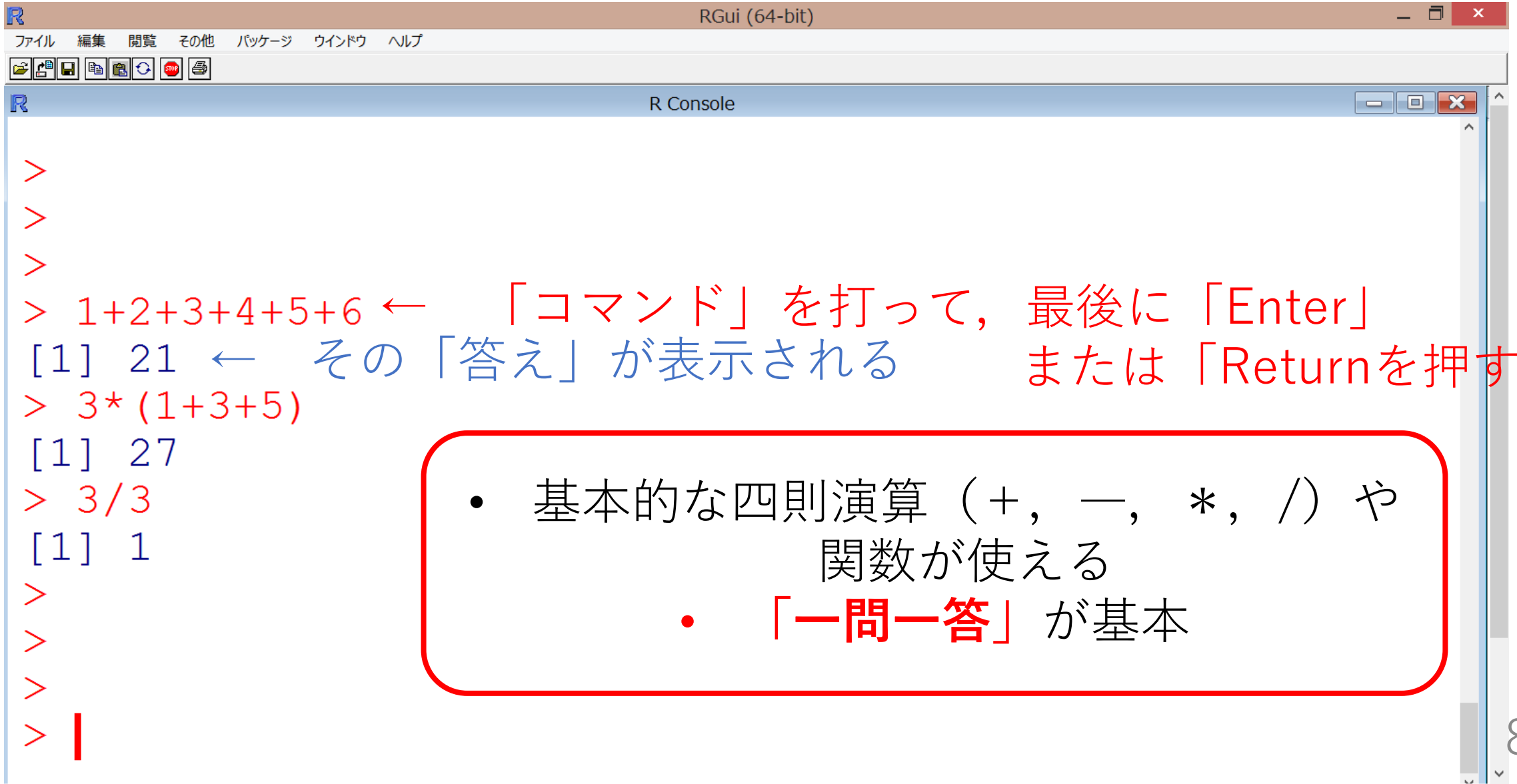
# ～まず使ってみよう～

Rでのコマンド入力に慣れる

- R console上でコマンドを打っていく

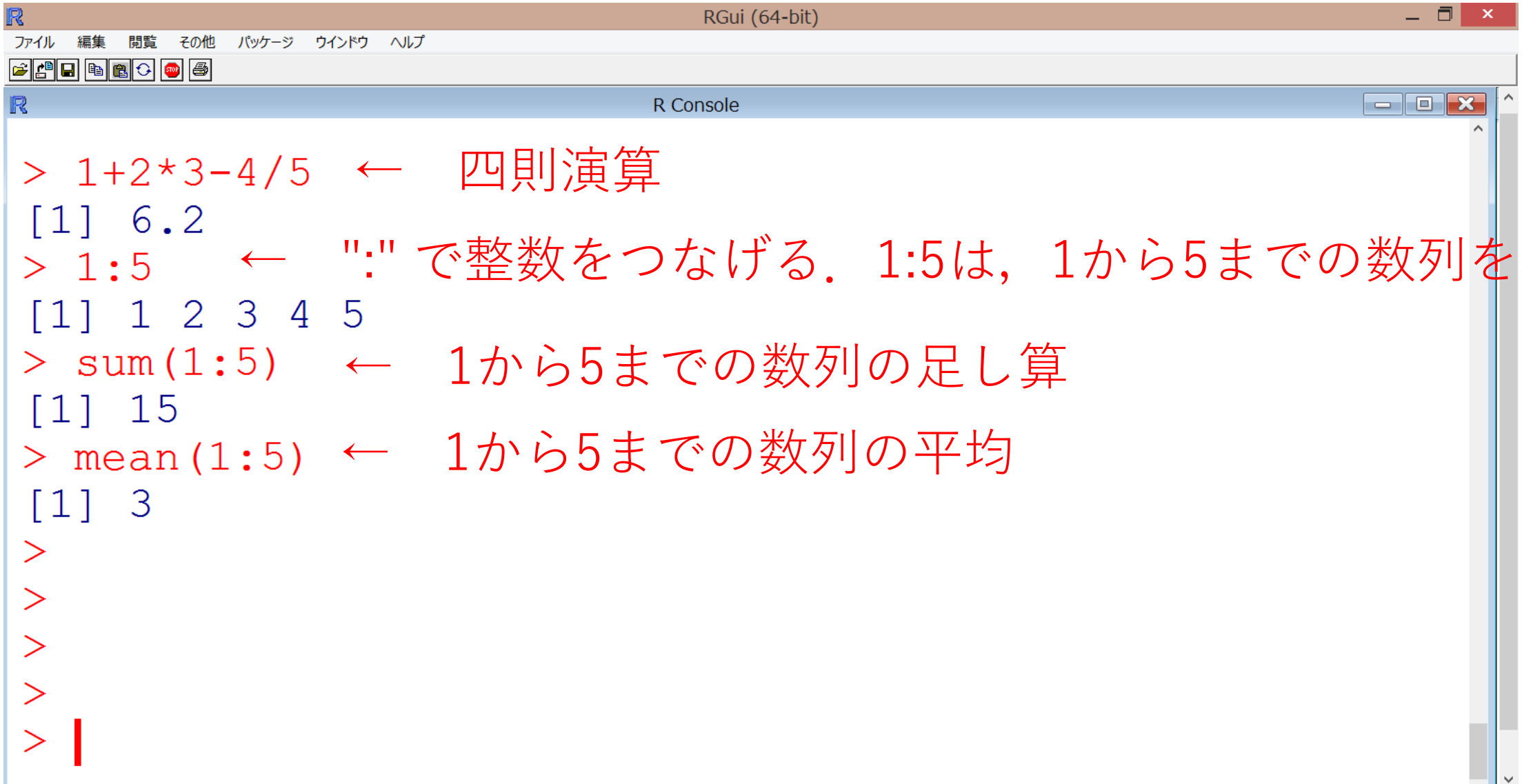


# • 計算機がわりに使う (1)




- 基本的な四則演算（＋，－，＊，／）や関数が使える
  - **「一問一答」**が基本

- 計算機がわりに使う (2)


RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ
R Console
> 1+2*3-4/5 ← 四則演算
[1] 6.2
> 1:5 ← ":" で整数をつなげる．1:5は，1から5までの数列を表す
[1] 1 2 3 4 5
> sum(1:5) ← 1から5までの数列の足し算
[1] 15
> mean(1:5) ← 1から5までの数列の平均
[1] 3
>
>
>
>
>

# よく使う関数

- sum　足し算
- mean　平均
- sqrt　ルート
- log 自然対数
- log10 常用対数
- help ヘルプの表示

関数名(引数)が基本

# 変数に値を代入する


```
>
> a <- 3
> a
[1] 3
> b <- c(1,5,3,6)
> b
[1] 1 5 3 6
> mean(b)
[1] 3.75
>
> x <- c("a","test",3,5)
> x
[1] "a"    "test" "3"    "5"
>
>
```

a　<-　3

変数　代入記号　数

# いろいろな ”型”

## 1. ベクトル

ベクトルの中身
↓
a[3]
↑
変数(ベクトル)

```
> # ベクトル
> b <- c(1,5,3,6)
> b
[1] 1 5 3 6
> mean(b)
[1] 3.75
>
> # ベクトルの要素を取り出す（"[]"を使う）
> b[3]
[1] 3
> b[4]
[1] 6
> b[8]
[1] NA
> # 空のベクトル（"numeric"）
> b0 <- numeric()
> b0
numeric(0)
> b0[3]
[1] NA
> b0[2] <- 100
> b0
[1]  NA 100
> |
```

# いろいろな ”型”

## 2. 行列

```
> # 行列
> x <- matrix(1:6,2,3)
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
>
>
> # 行列
> x <- matrix(1:6,2,3) # matrix(行列の要素,列数, 行数)
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
>
> # 行列の要素を取り出す
> x[2,3]
[1] 6
> x[2,] # 2行目を全て
[1] 2 4 6
>
```

行列の中身
↓
x[2,3]
↑
変数(行列)

```
>
> # 空の行列
> x0 <- matrix(NA,4,5)
> x0
     [,1] [,2] [,3] [,4] [,5]
[1,]   NA   NA   NA   NA   NA
[2,]   NA   NA   NA   NA   NA
[3,]   NA   NA   NA   NA   NA
[4,]   NA   NA   NA   NA   NA
> x0[3,2] <- 1
> x0[4,] <- 1:5
> x0
     [,1] [,2] [,3] [,4] [,5]
[1,]   NA   NA   NA   NA   NA
[2,]   NA   NA   NA   NA   NA
[3,]   NA    1   NA   NA   NA
[4,]    1    2    3    4    5
>
```

# いろいろな”型”
# 3. リスト・データフレーム

```
> test1 <- list(1:5, matrix(0,4,5))
> test1[[1]]
[1] 1 2 3 4 5
> test1
[[1]]
[1] 1 2 3 4 5

[[2]]
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    0    0    0    0    0
[3,]    0    0    0    0    0
[4,]    0    0    0    0    0
```

リストの中身

↓

test[[1]]

↑

変数(リスト)

# いろいろな”型”
# 3. リスト・データフレーム

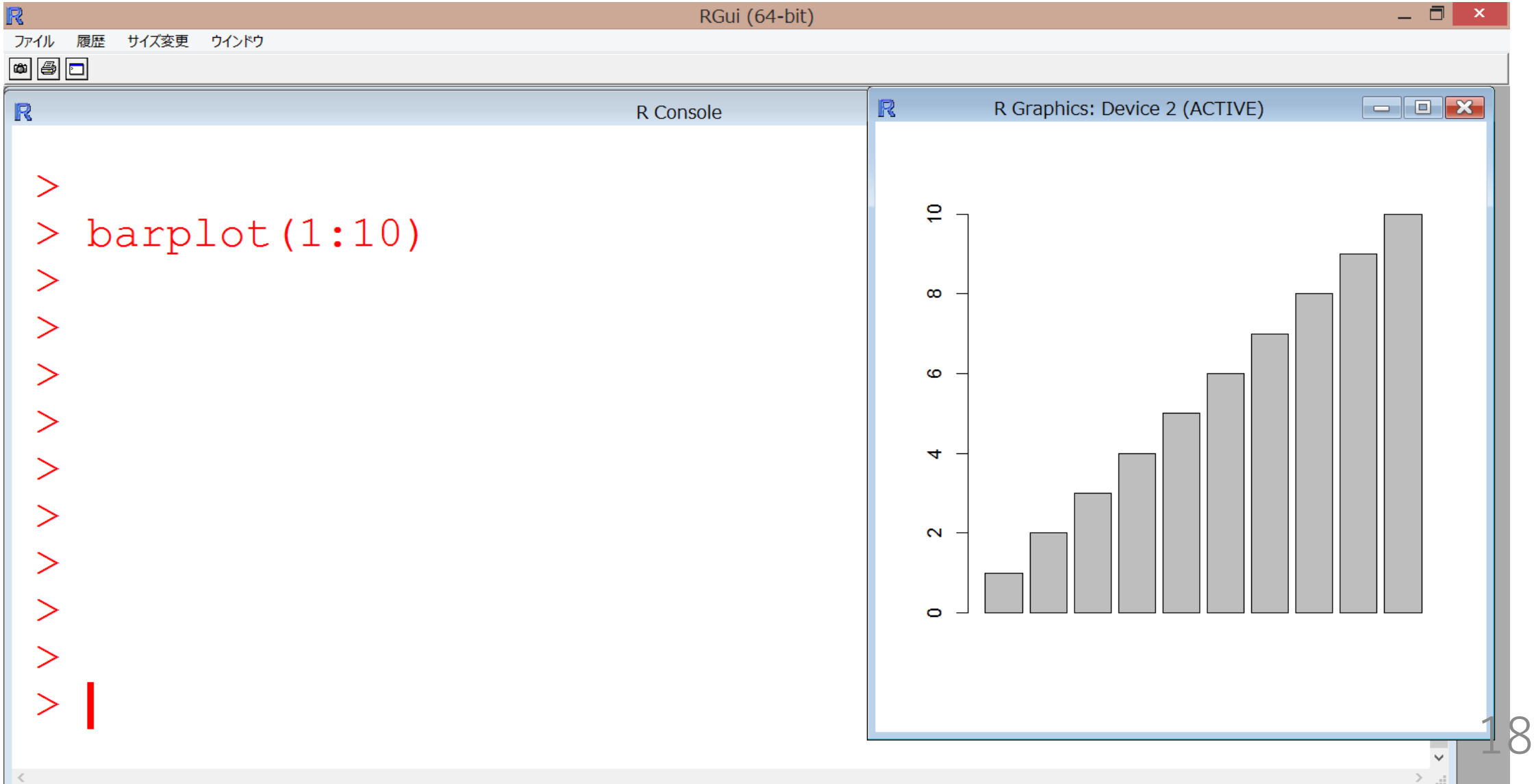
```
> test2 <- list(a1=1:5, a2=list(), a3=matrix(1,3,3))
> test2$a1
[1] 1 2 3 4 5
> 
```

# その他

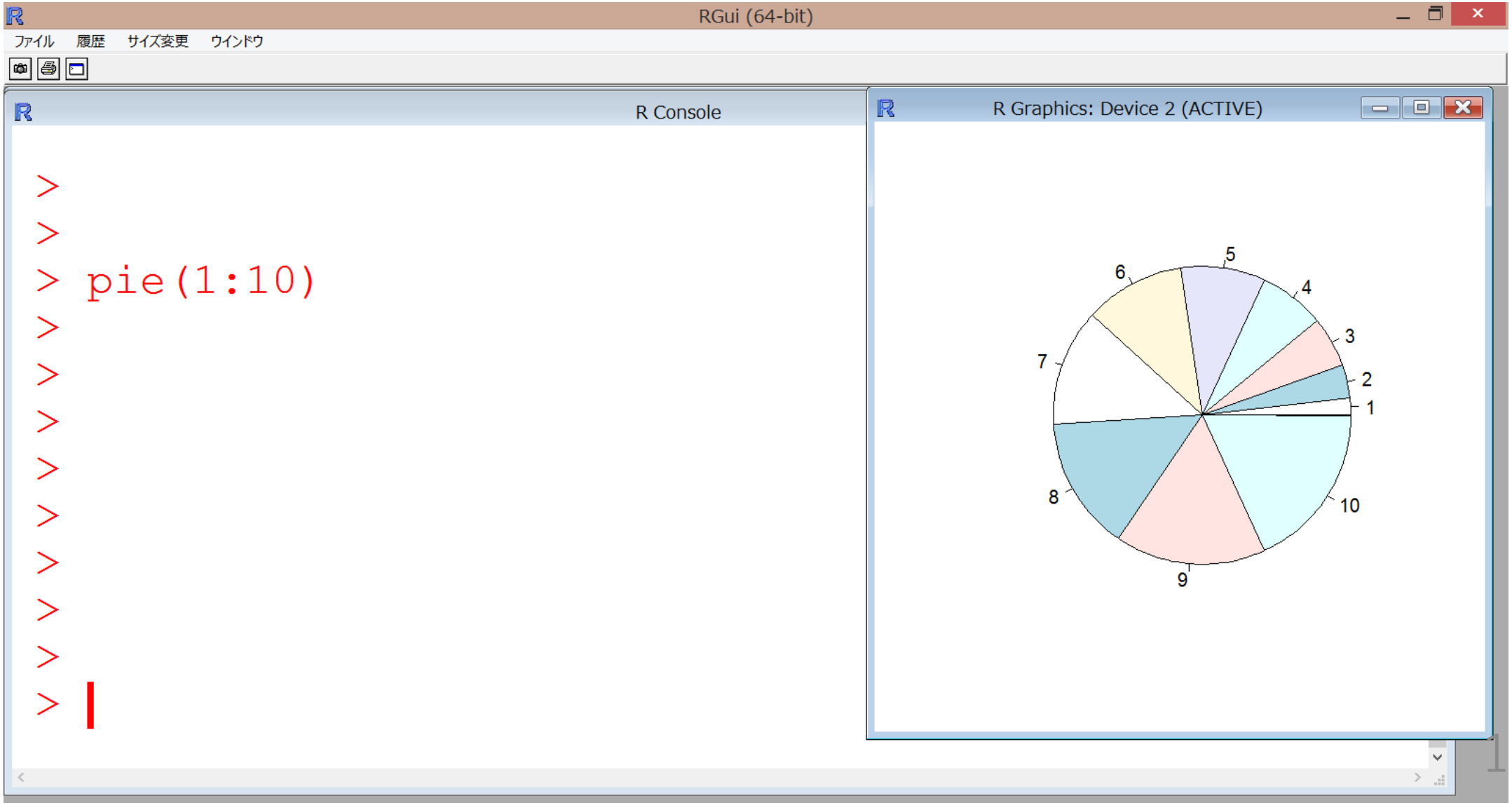
- array: 3次元以上の行列

```
test3 <- array(0, dim=c(3,4,5))
```

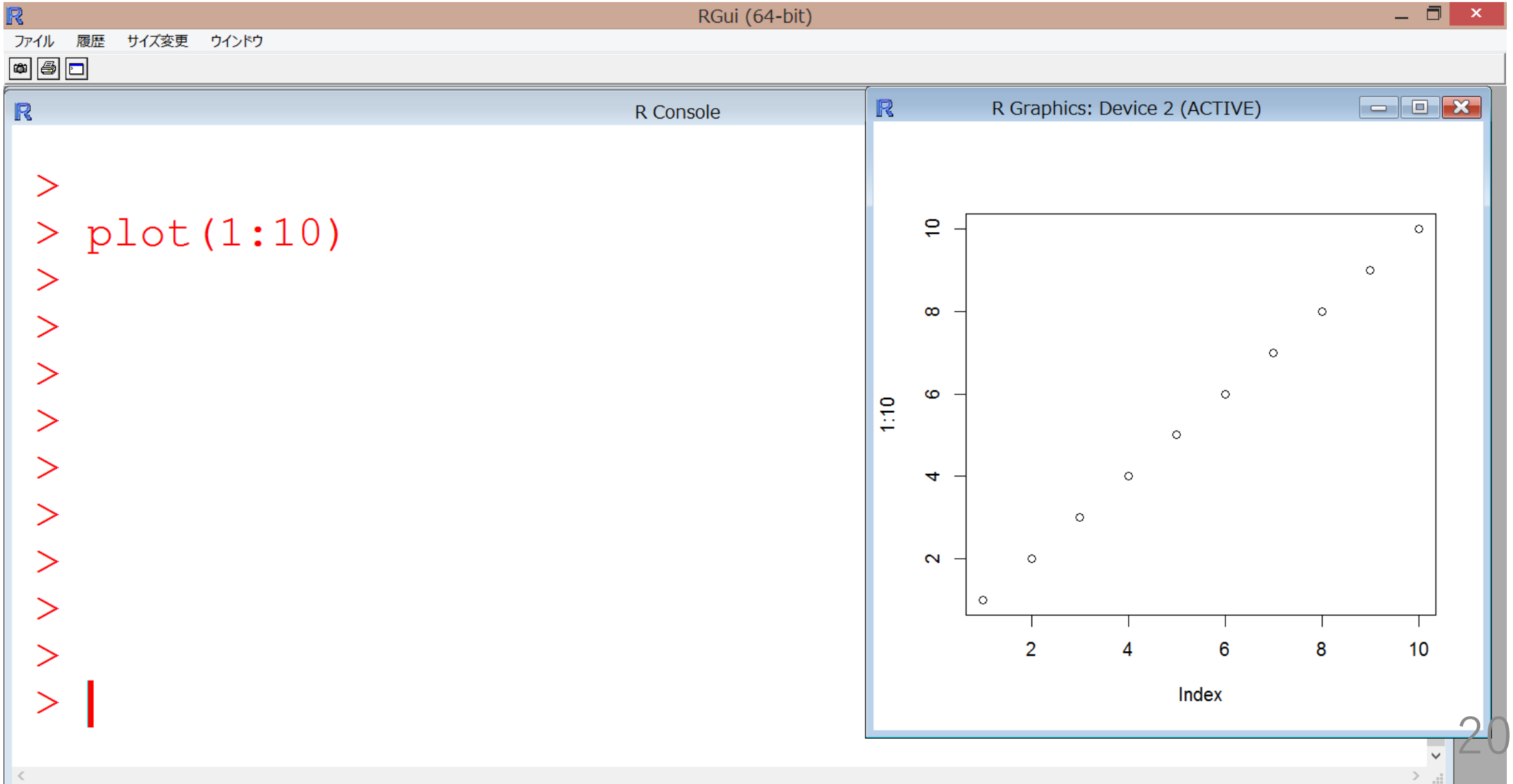
- グラフを書いてみる (1)


RGui (64-bit)
ファイル 履歴 サイズ変更 ウインドウ
R Console
> barplot(1:10)
R Graphics: Device 2 (ACTIVE)
0
2
4
6
8
10

- グラフを書いてみる (2)


RGui (64-bit)
ファイル 履歴 サイズ変更 ウインドウ
R Console
> pie(1:10)
R Graphics: Device 2 (ACTIVE)
1
2
3
4
5
6
7
8
9
10

- グラフを書いてみる (3)


RGui (64-bit)
ファイル 履歴 サイズ変更 ウインドウ
R Console
> plot(1:10)
R Graphics: Device 2 (ACTIVE)
1:10
Index
2
4
6
8
10

# – 関数と引数 –

引数：　関数が計算するさいに必要な情報


> help(plot)　# どのような引数が使えるか等，関数のヘルプ

plot {graphics} R Documentation

# Generic X-Y Plotting

## Description

Generic function for plotting of R objects. For more details about the graphical parameter arguments, see par.

For simple scatter plots, plot.default will be used. However, there are plot methods for many R objects, including functions, data.frames, density objects, etc. Use methods(plot) and the documentation for these.

## Usage

```
plot(x, y, ...)
```

## Arguments

x
: the coordinates of points in the plot. Alternatively, a single plotting structure, function or *any R object with a plot method* can be provided.

plot.default {graphics} R Documentation

# The Default Scatterplot Function

## Description

Draw a scatter plot with decorations such as axes and titles in the active graphics window.

## Usage

```
## Default S3 method:
plot(x, y = NULL, type = "p",  xlim = NULL, ylim = NULL,
     log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
     ann = par("ann"), axes = TRUE, frame.plot = axes,
     panel.first = NULL, panel.last = NULL, asp = NA, ...)
```

## Arguments

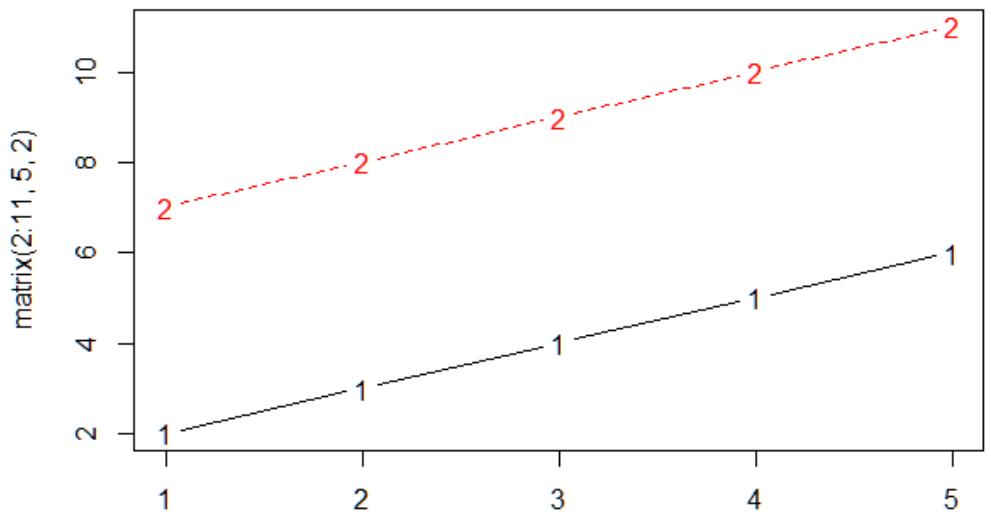
`x, y`

the `x` and `y` arguments provide the x and y coordinates for the plot. Any reasonable way of defining the coordinates is acceptable. See the function `xy.coords` for details. If supplied separately, they must be of the same length.

```
plot(1:10, type="b")

plot(1:10, type="b",
             xlim=c(0,20), ylim=c(0,20), xlab="X label",
             ylab="Y label", col=c("red","blue","darkgreen"))

plot(1:10, type="b", xaxs="i", yaxs="i", xlim=c(0,11),
              ylim=c(0,11)) # 原点をゼロとするグラフ
```

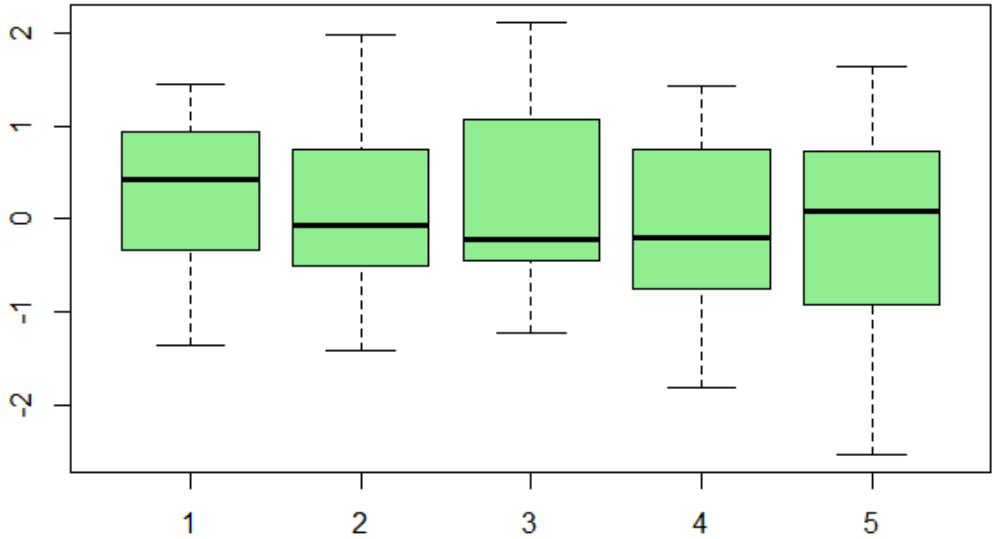
# そのほかのグラフ

matplot; 二次元データのグラフ



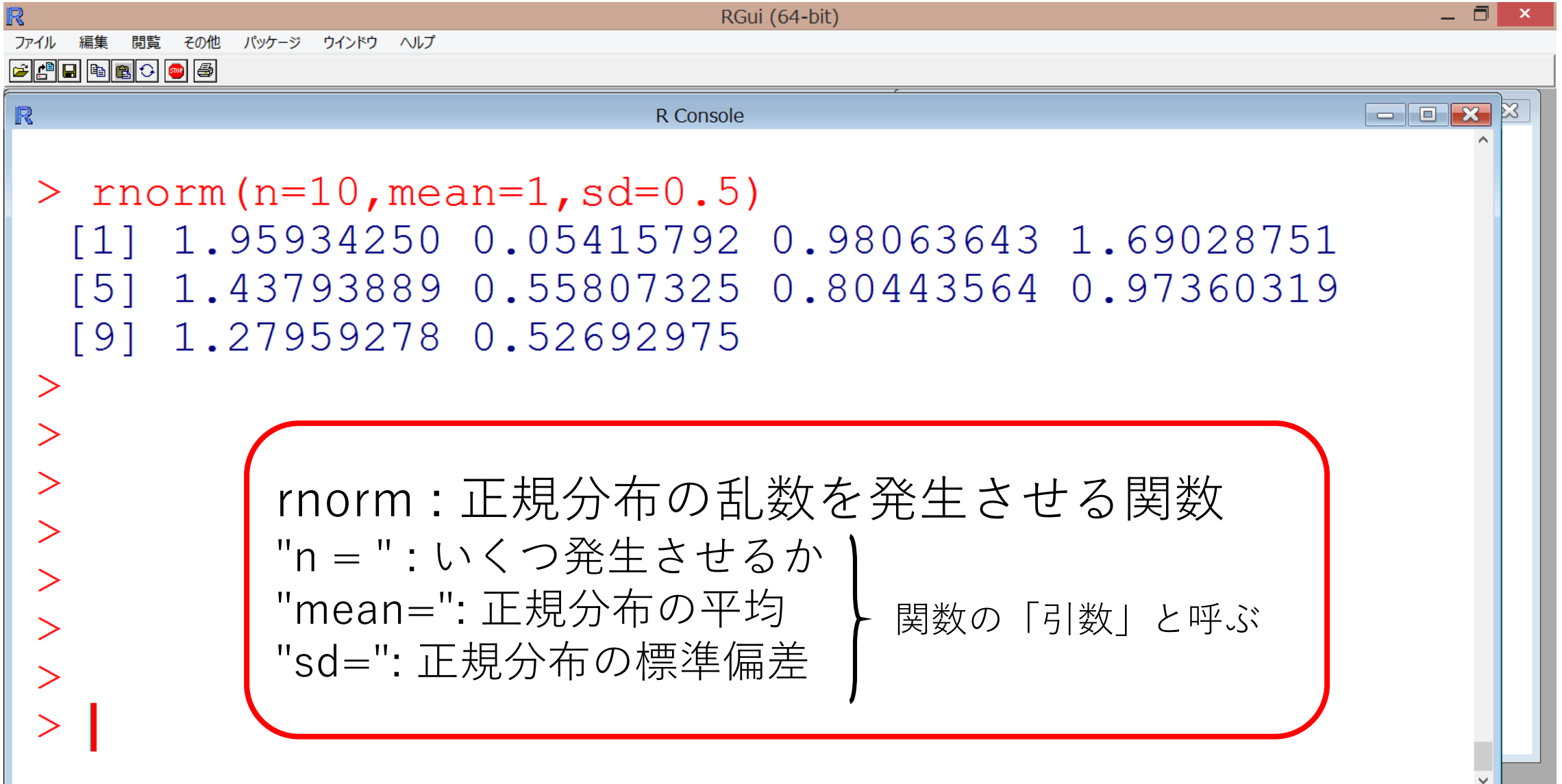

matplot(matrix(2:11,5,2),type="b",xlab="x",ylab="y")

boxplot; 箱型図

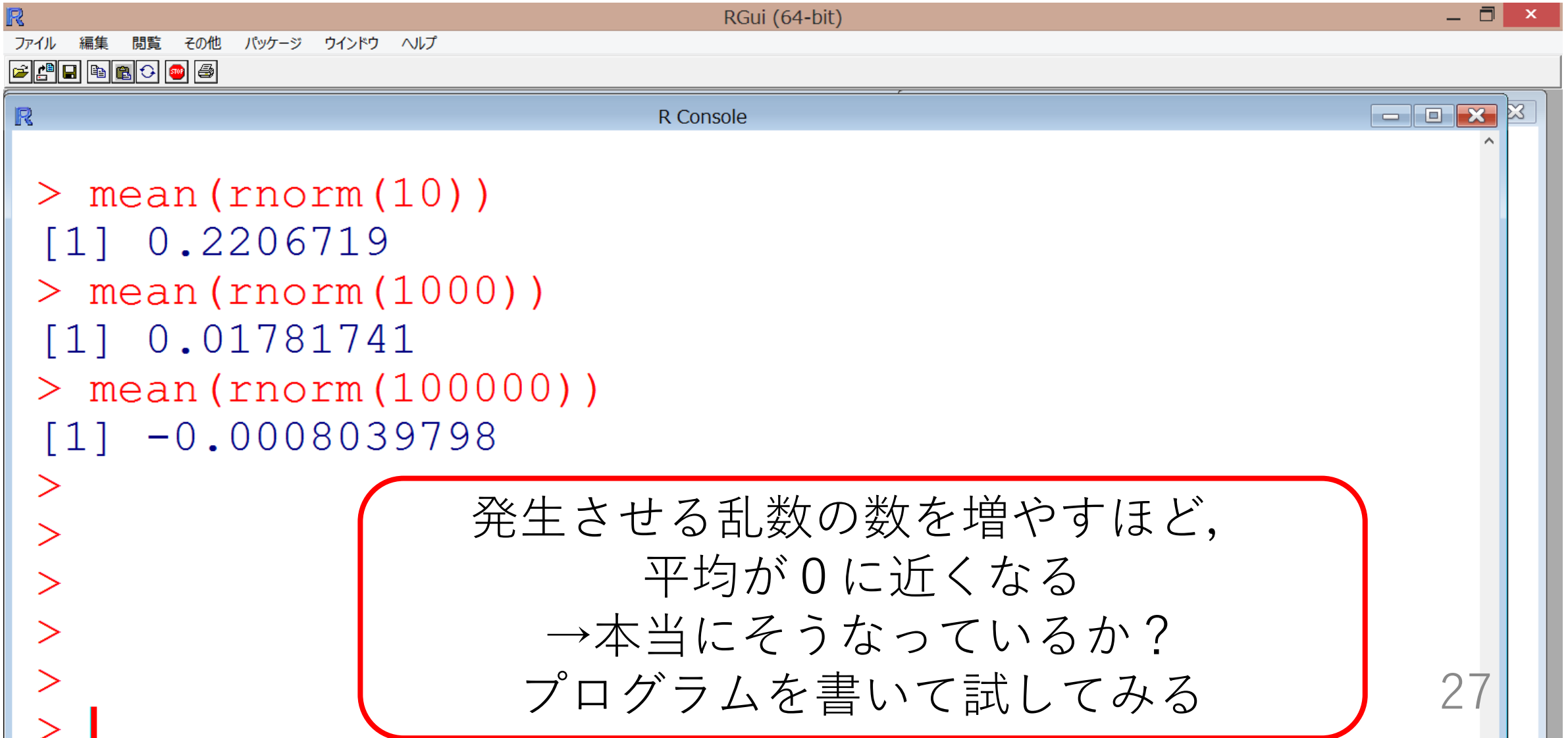



boxplot(matrix(rnorm(100),20,5),col="lightgreen")

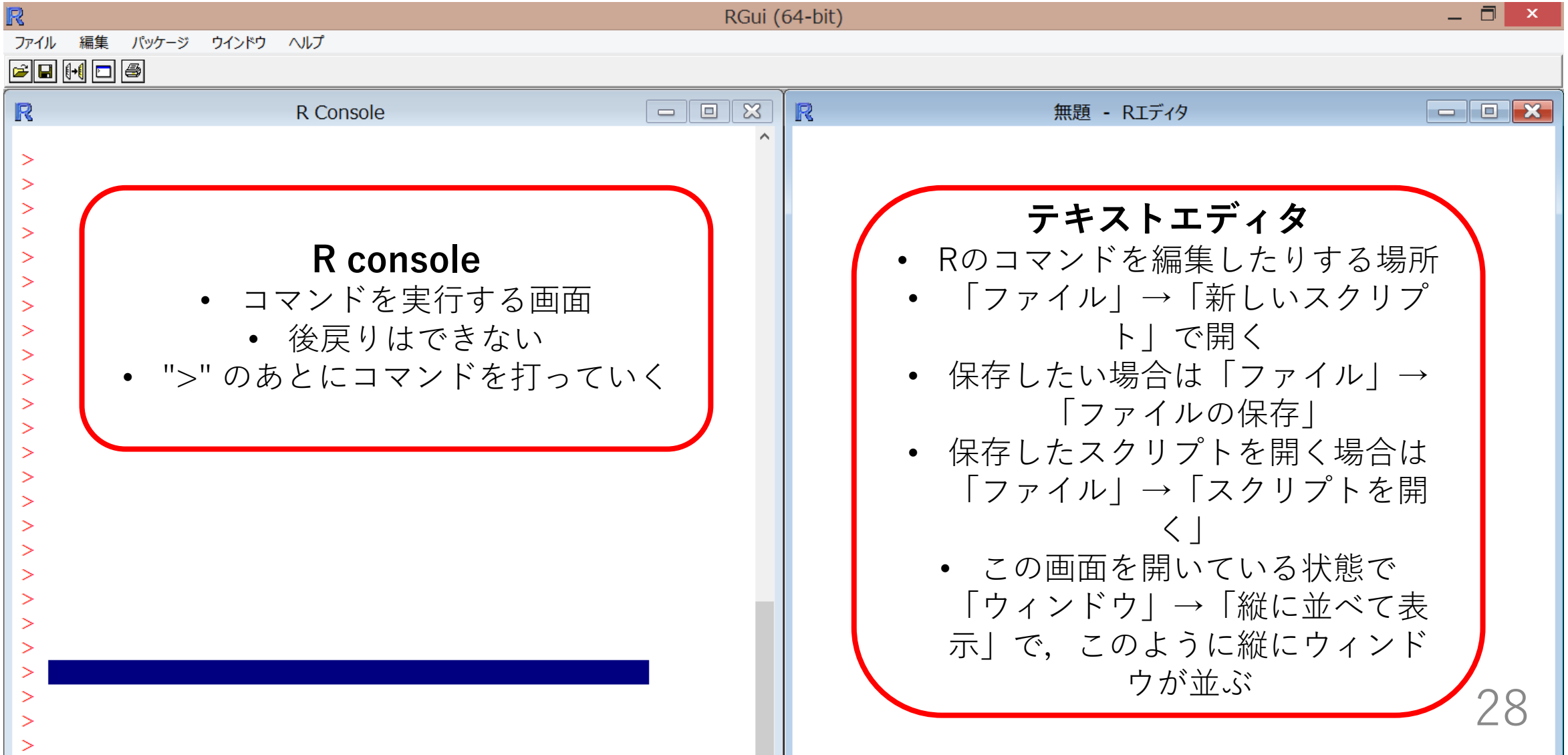
- 乱数の発生 (1)


RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ
R Console
> rnorm(n=10,mean=1,sd=0.5)
[1] 1.95934250 0.05415792 0.98063643 1.69028751
[5] 1.43793889 0.55807325 0.80443564 0.97360319
[9] 1.27959278 0.52692975
>
rnorm：正規分布の乱数を発生させる関数
"n＝"：いくつ発生させるか
"mean=": 正規分布の平均
"sd=": 正規分布の標準偏差
関数の「引数」と呼ぶ

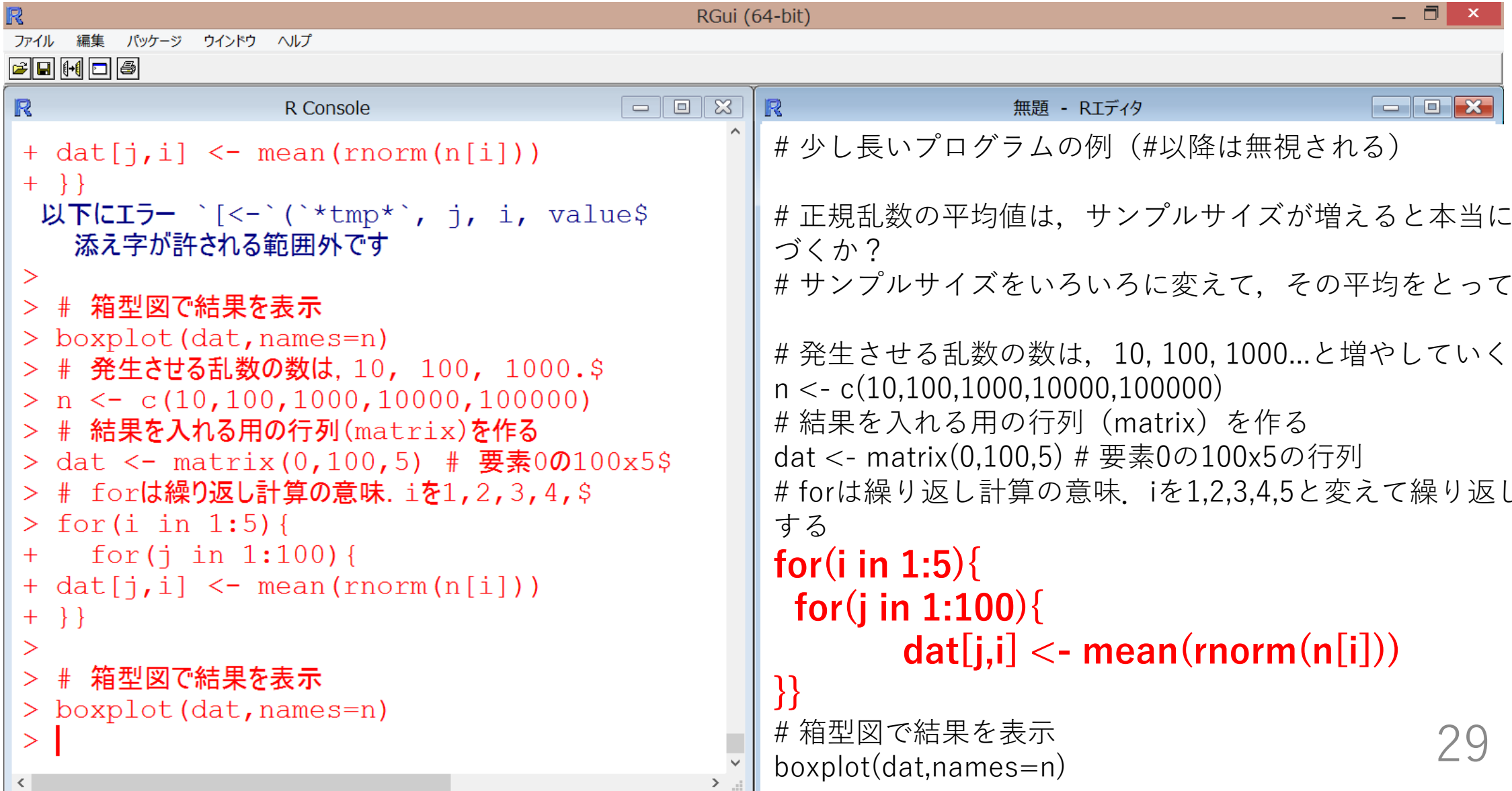
- 乱数の発生 (2)


RGui (64-bit)
ファイル 編集 閲覧 その他 パッケージ ウインドウ ヘルプ
R Console
> mean(rnorm(10))
[1] 0.2206719
> mean(rnorm(1000))
[1] 0.01781741
> mean(rnorm(100000))
[1] -0.0008039798
>
>
>
>
>
>
発生させる乱数の数を増やすほど,
平均が 0 に近くなる
→本当にそうなっているか？
プログラムを書いて試してみる

# 長いプログラムを書く場合は？→例えば，このようにウィンドウを配置する（左；R console, 右; R エディタ）



- 右のテキストエディタでプログラムを編集→実行したい部分を選んで Control + R



RGui (64-bit)

ファイル　履歴　サイズ変更　ウインドウ

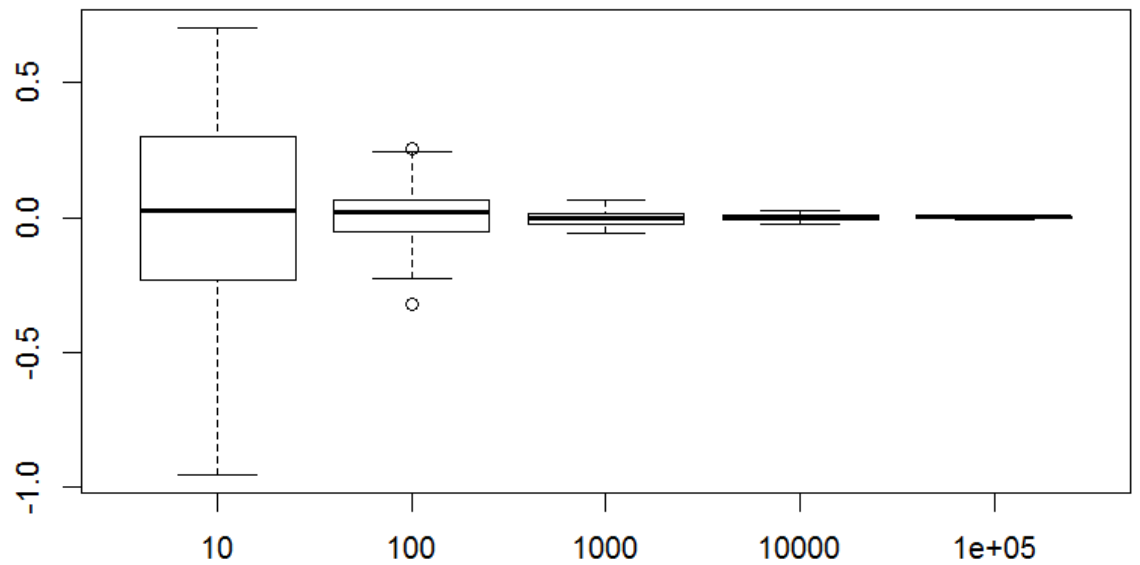
R Console

```
+ dat[j,i] <- mean(rnorm(n[i]))
+ }}
```

以下にエラー `[<-`(`*tmp*
添え字が許される範囲外です

```
>
> # 箱型図で結果を表示
> boxplot(dat,names=n)
> # 発生させる乱数の数は，10
> n <- c(10,100,1000,1
> # 結果を入れる用の行列(ma
> dat <- matrix(0,100,
> # forは繰り返し計算の意味.
> for(i in 1:5){
+   for(j in 1:100){
+ dat[j,i] <- mean(rno
+ }}
>
> # 箱型図で結果を表示
> boxplot(dat,names=n)
>
```

R Graphics: Device 2 (ACTIVE)




無題 - Rエディタ

```
# 少し長いプログラムの例(#以降はコメントアウトされる)
```

当に0に近づくか？
みる

と増やしていく

x5の行列
変えて繰り返し計算する

# ~外部との連携~

データ入出力やパッケージのインストール

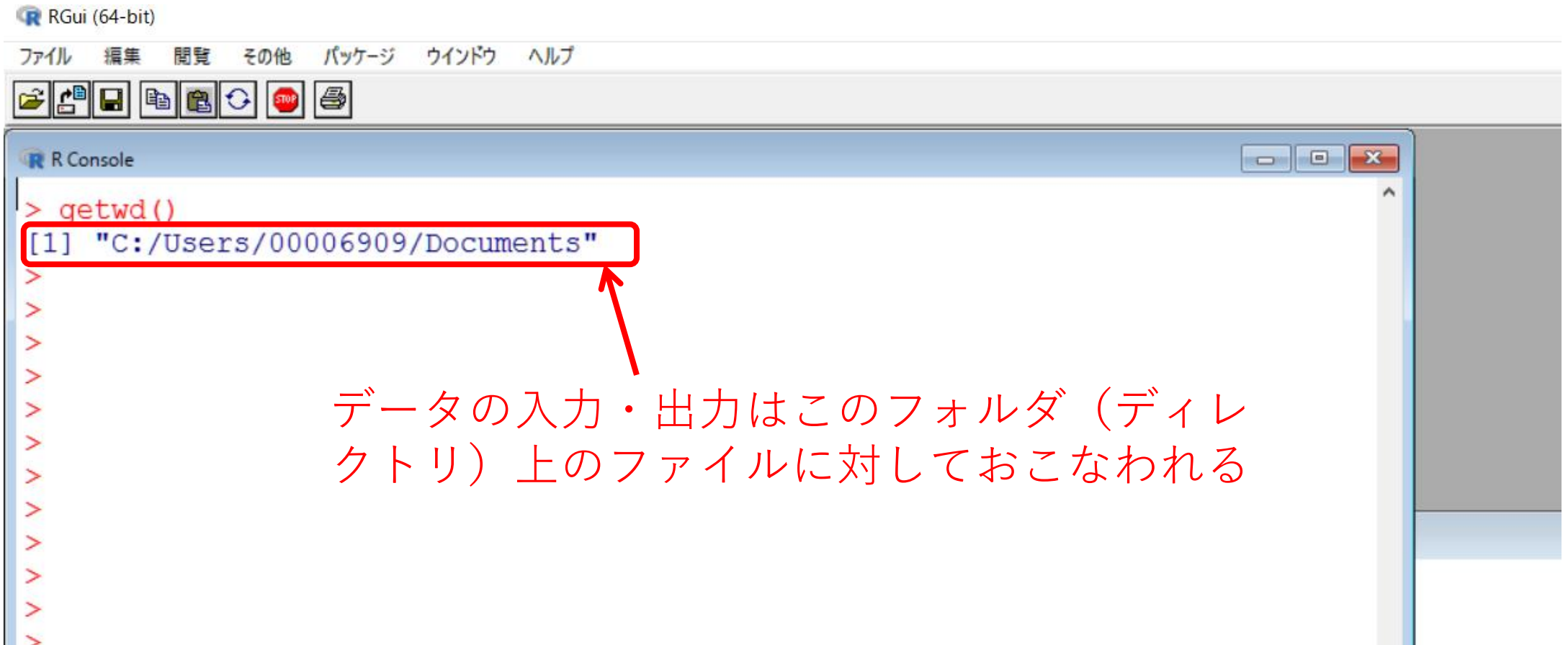
# Rの有利な点：外部との連携

- テキスト形式でのデータの入出力
  - エクセルでデータの整理→Rで解析
  - Rで統計計算→エクセルできれいな表にする

- 他の人が作った関数やパッケージを自分の解析に利用できる
  - 新規的な統計手法をすぐ自分のデータに適用できる

# 外部とのやりとりのさいに重要なこと：自分がどこにいるか把握する

> getwd() # 作業ディレクトリを表示



# 作業用のフォルダを作ってそこに移動する

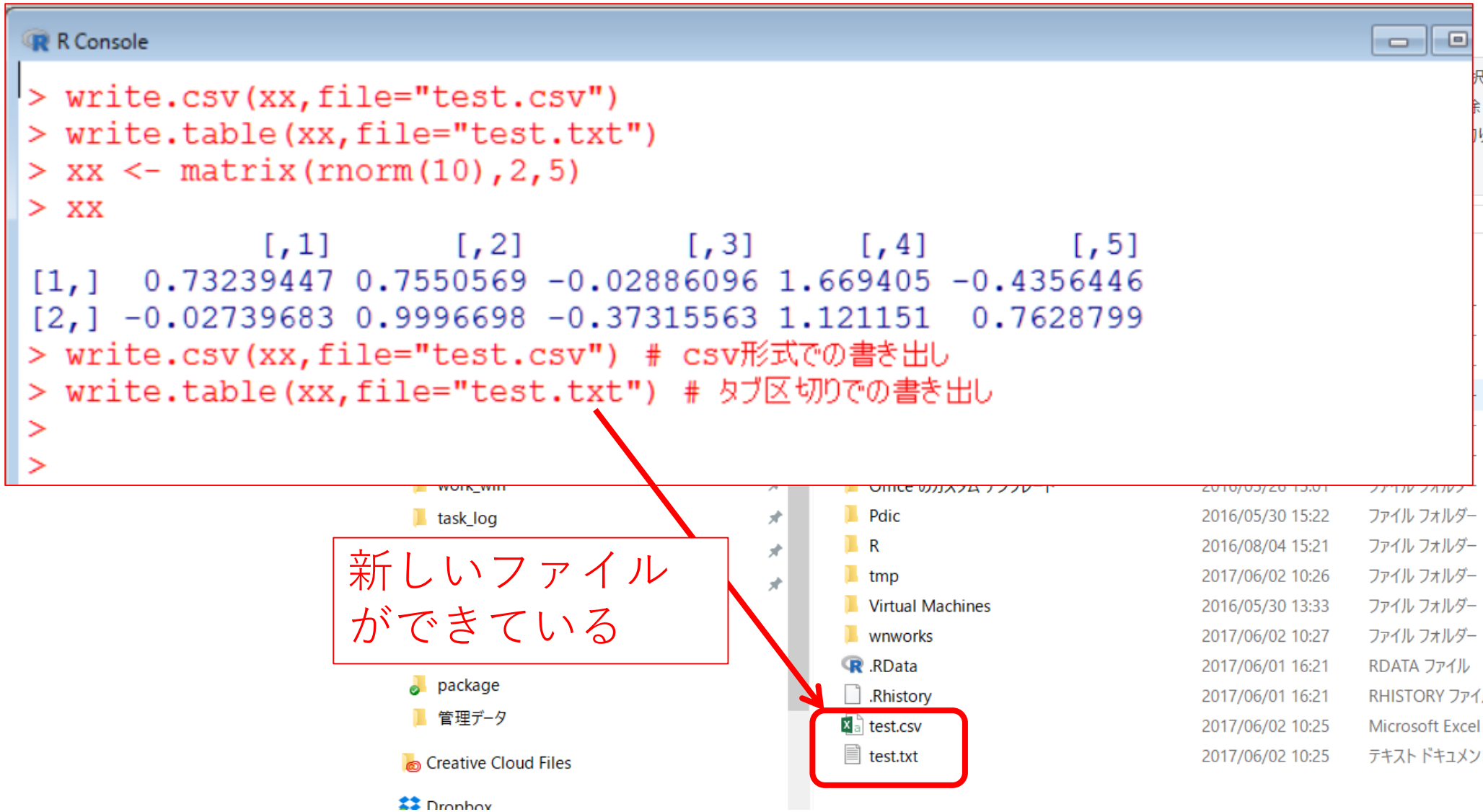
- あたらしいフォルダを作成
- メニュー「ファイル」

→「ディレクトリの変更」

→ 目的のディレクトリを選ぶ

→ getwd() コマンドで目的のディレクトリに
移動できているか確認

# データの書き出し write.csv, write.table



# データの読み込み：read.csv, read.table

R Console

```
> read.csv(file="test.csv")
  X          V1         V2          V3        V4          V5
1 1  0.73239447 0.7550569 -0.02886096 1.669405 -0.4356446
2 2 -0.02739683 0.9996698 -0.37315563 1.121151  0.7628799
> read.table(file="test.txt")
           V1         V2          V3        V4          V5
1  0.73239447 0.7550569 -0.02886096 1.669405 -0.4356446
2 -0.02739683 0.9996698 -0.37315563 1.121151  0.7628799
> x1 <- read.csv(file="test.csv")
> x2 <- read.table(file="test.txt")
>
>
>
```

# Rのオブジェクトそのものの保存，ロード

```
x <- rnorm(10)
save(x, file="x.Rda")

load("x.Rda")
```

# パッケージのインストール：最も簡単な場合

- ネットに繋がっている＆目的のパッケージがCRAN(Rのパッケージが集積されている場所)に登録されている

- 「パッケージ」→「パッケージのインストール」

→ 目的のパッケージをリストから選ぶ

→ ミラーサイトをリストから選ぶ（通常は"Japan"を選ぶ）

```
> utils:::menuInstallPkgs()
 URL 'https://cran.ism.ac.jp/bin/windows/contrib/3.3/acm4r_1.0.zip' を試して$
Content type 'application/zip' length 56571 bytes (55 KB)
downloaded 55 KB

 パッケージ 'acm4r' は無事に展開され、MD5 サムもチェックされました

 ダウンロードされたパッケージは、以下にあります
        C:\Users\00006909\AppData\Local\Temp\RtmpM7JGnO\downloaded_packages
> |
```

# パッケージのインストール：zipファイルまたはtar.gzファイルからのインストール

- インターネットにつながっていない or 目的のパッケージがCRANに登録されていない

- ソースファイルをダウンロード→自分でインストール

- CRANのパッケージ置き場（例　https://cran.ism.ac.jp/）

Available CRAN Packages By Date of Publication

| | |
|---|---|
| Reference manual: | AIG.pdf |
| Package source: | AIG_0.1.6.tar.gz |
| Windows binaries: | r-devel: AIG_0.1.5.zip, r-release: AIG_0.1.5.zip, r-oldrel: AIG_0.1.5.zip |
| OS X El Capitan binaries: | r-release: AIG_0.1.5.tgz |
| OS X Mavericks binaries: | r-oldrel: AIG_0.1.5.tgz |
| Old sources: | AIG archive |

- CRAN以外からのインストール（例 http://cse.fra.affrc.go.jp/ichimomo/fish/rvpa.html）

**RVPA**

VPAを用いた日本国内資源の資源評価を行うためのRのパッケージを配布しています。 パッケージはR3.0.1のもとで作成しましたので、インストールに問題が生じたときは、R3.0.1に近いバージョンのRでお試しください。

**更新情報**

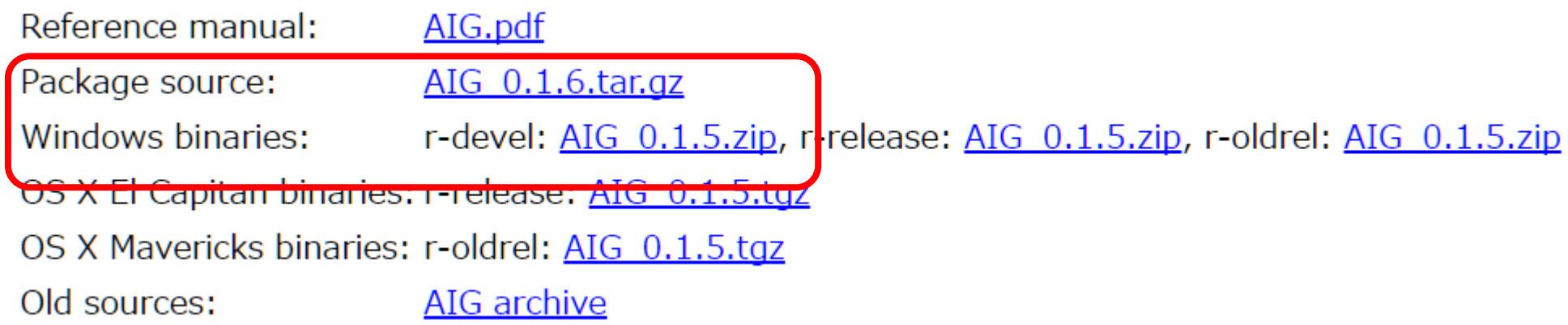
- 2017/05/30 1.8にアップデート

**ダウンロードファイル**

- RVPAソースコード RVPA_1.8.tar.gz
- Windows用バイナリ RVPA_1.8.zip
- 日本語解説つきテキストコード rvpa1.8.r (VPA)
- 日本語解説つきテキストコード future1.8.r (管理基準値計算・将来予測)

# ローカルファイルのインストール

- 自分の使っているパソコンのOSにあったzipファイル，または，****.tar.gzという拡張子のファイルをダウンロード




- 「パッケージ」→「Install package(s) from local files…」→ダウンロードしたファイルを選ぶ

＊Rのバージョンによって上記のメニューが「ローカルにあるzipファイルからのインストール」となっている場合がある．この場合は，拡張子がtar.gzのファイルはこのメニューからインストールできない

```
> install.packages("AIG_0.1.5.tar.gz", source=TRUE)
```

```
> utils:::menuInstallLocal()
ERROR: dependencies 'dplyr', 'rgl', 'magrittr' are not available for package 'AIG'
* removing 'C:/Users/00006909/Documents/R/R-3.3.1/library/AIG'
 警告メッセージ:
1:  命令 '"C:/Users/00006909/Documents/R/R-3.3.1/bin/x64/R" CMD INSTALL -l "C:\Users\00006909\Documents
2:  install.packages(files[tarballs], .libPaths()[1L], repos = NULL,  で:
   パッケージ 'C:/Users/00006909/Downloads/AIG_0.1.6.tar.gz' のインストールは、ゼロでない終了値をもちました
>
```

- エラーが出る場合も

- 他のパッケージも同時に必要だけど，それがインストールされていないというエラー

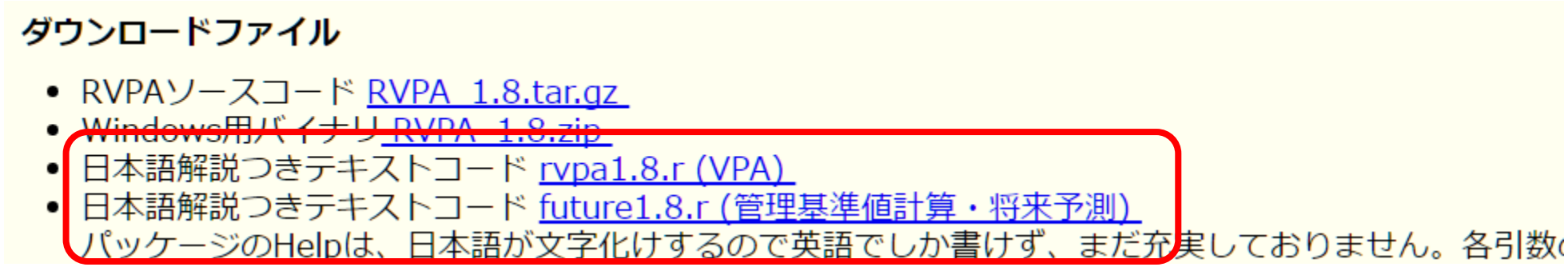
# 配布したパッケージをインストールしてみよう

- 「パッケージ」→「 Install package(s) from local files…」→ mapdata_2.2-6.zip を選ぶ
- 同様に，maps_3.1.1.zipも選ぶ

- Macの人は拡張子がtgzになっているファイルを選ぶ

# さらに原始的な方法：source

- パッケージになっていないRのコードを一括して読む場合




```
> source("rvpa1.8.r")
> source("future1.8.r")
```

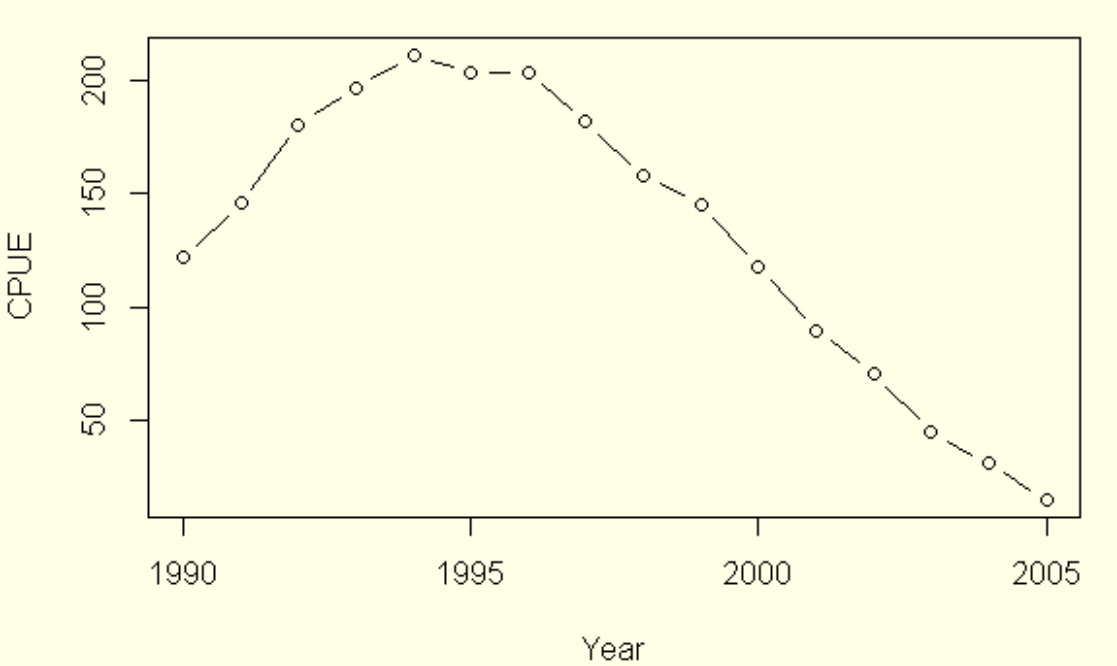
# ~いろいろやってみよう~

配布データを読み込んで，グラフを書いたり，
統計解析をしてみたりしよう

# 配布データを読み込んで，プロットしたり統計解析したりしてみよう

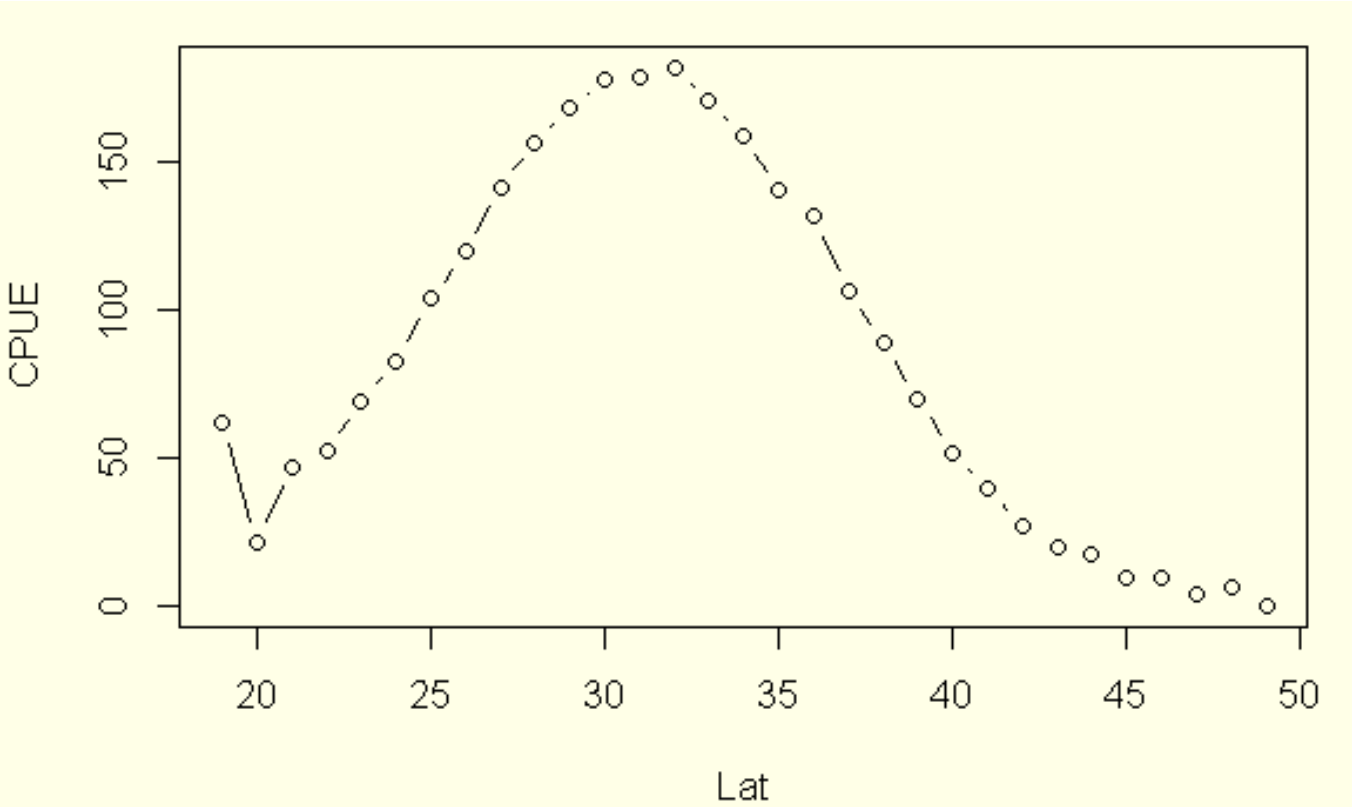
```
# 配布したcsvファイルを作業ディレクトリに置く
# ファイルの読み込み
tdata <- read.csv("testdata.csv")

# tdataがどんなデータか？
head(tdata) # head: 最初の 6 行だけ出力する関数
dim(tdata) # 何行何列のデータなのかを出力する関数
```

```
# 年ごとや緯度経度ごとに平均や合計を算出したい
#                  → tapply 関数を使う(エクセルのピボットテーブル)
#  tapply（集計したいもの, ラベル, 関数）
>x <- tapply(tdat$N, tdat$year, mean)
>plot(names(x), x, type="b",
                xlab="Year",ylab="CPUE")
```



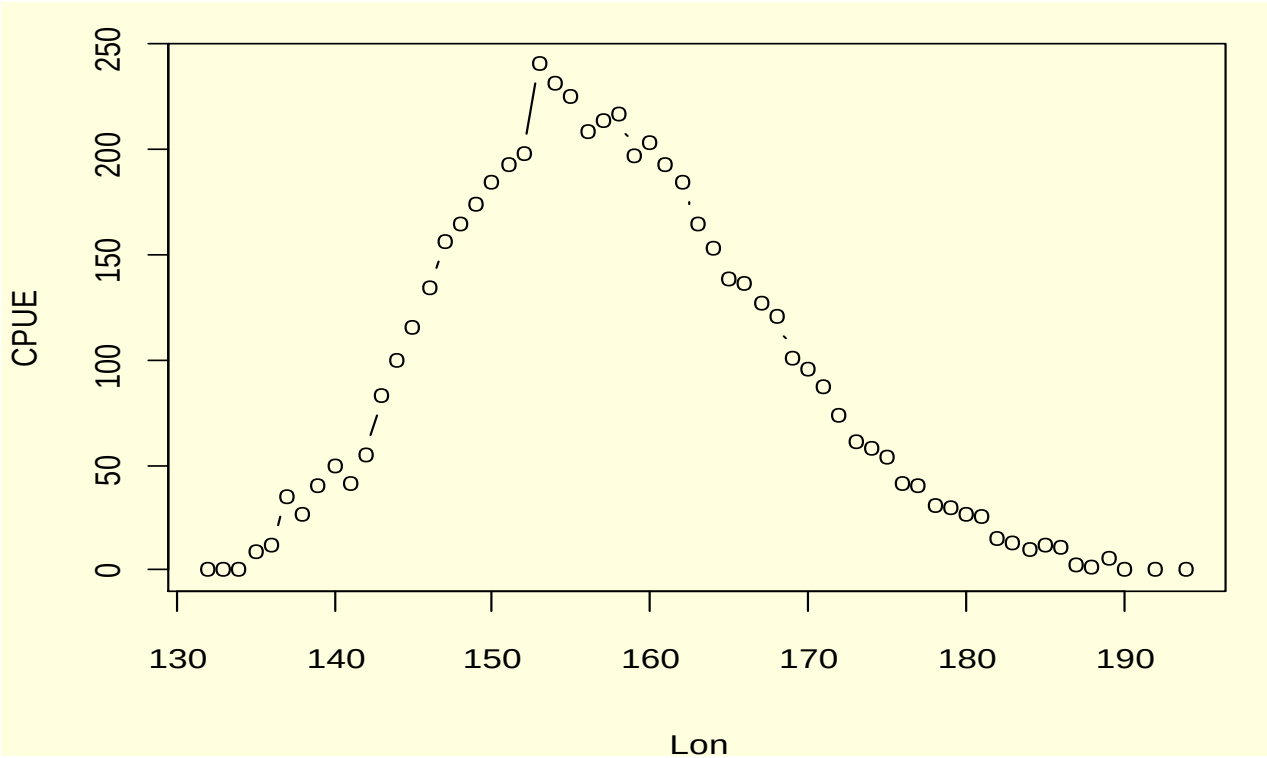
# 緯度ごとの平均

```
>x <- tapply(tdat$N, tdat$lat, mean)
>plot(names(x), x, type="b",
                xlab="Lat",ylab="CPUE")
```

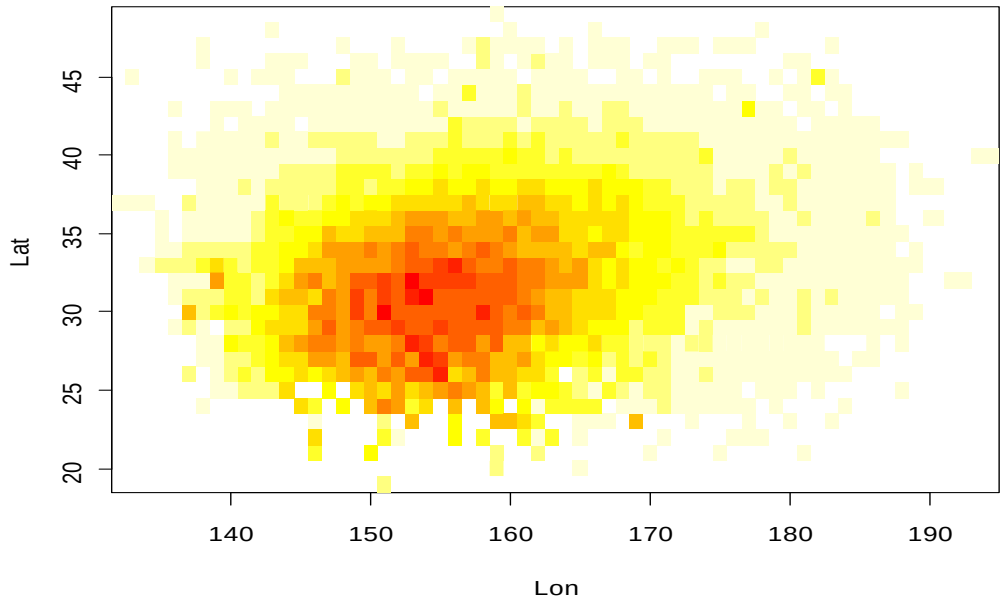


# 経度ごとの平均

```
>x <- tapply(tdat$N, tdat$lon, mean)
>plot(names(x), x, type="b",
          xlab="Lon",ylab="CPUE")
```

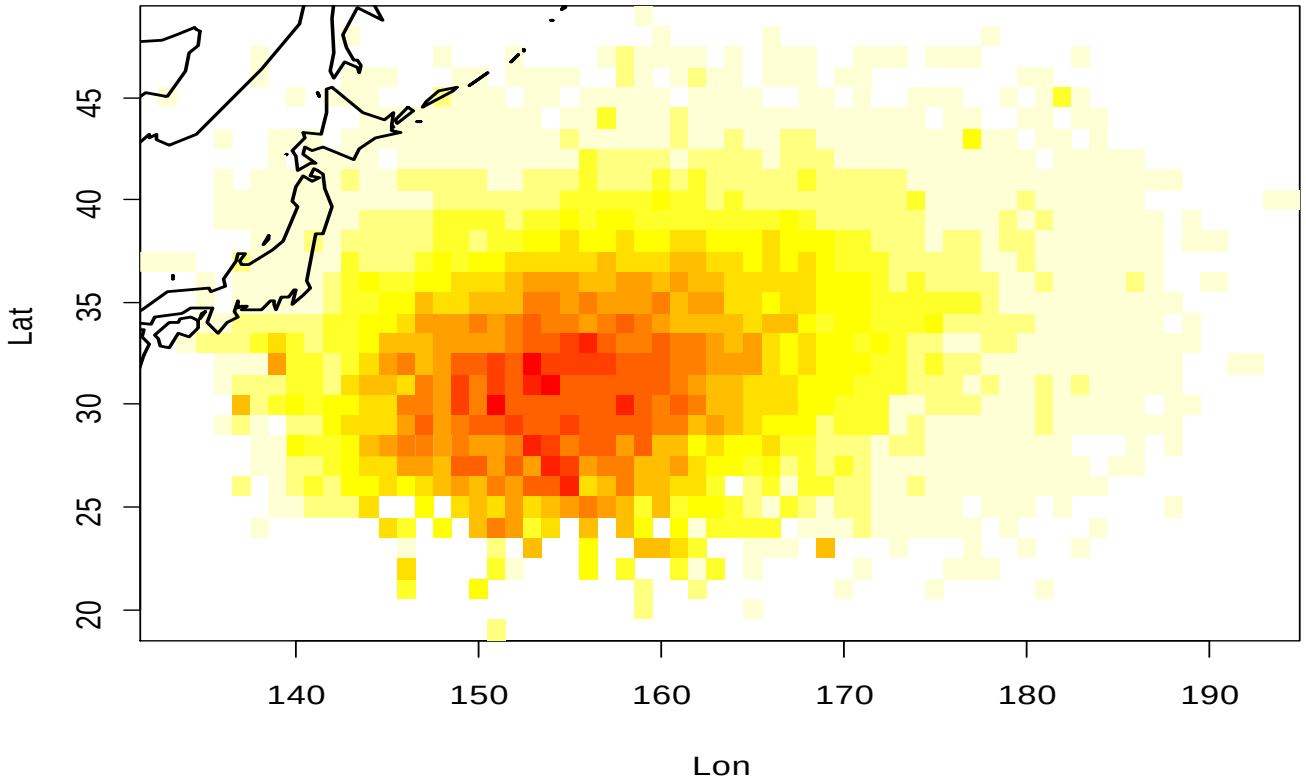


```
# 経度・経度ごとの平均
>x <- tapply(tdat$N,
          list(tdat$lon,tdat$lat),mean)
# これを2次元平面上にプロットしてみる
>x1 <- as.numeric(dimnames(x)[[1]])
>y1 <- as.numeric(dimnames(x)[[2]])
>image(x1,y1, x, xlab="Lon",ylab="Lat", col=rev(heat.colors(12)))
```

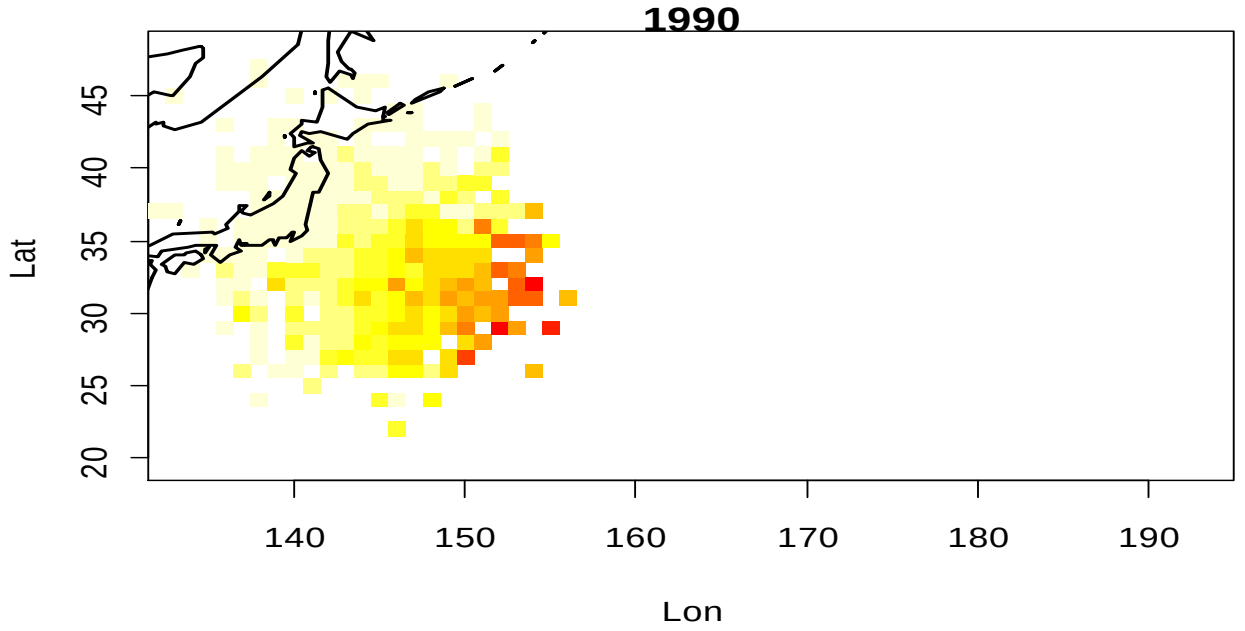


# 地図を上書きする

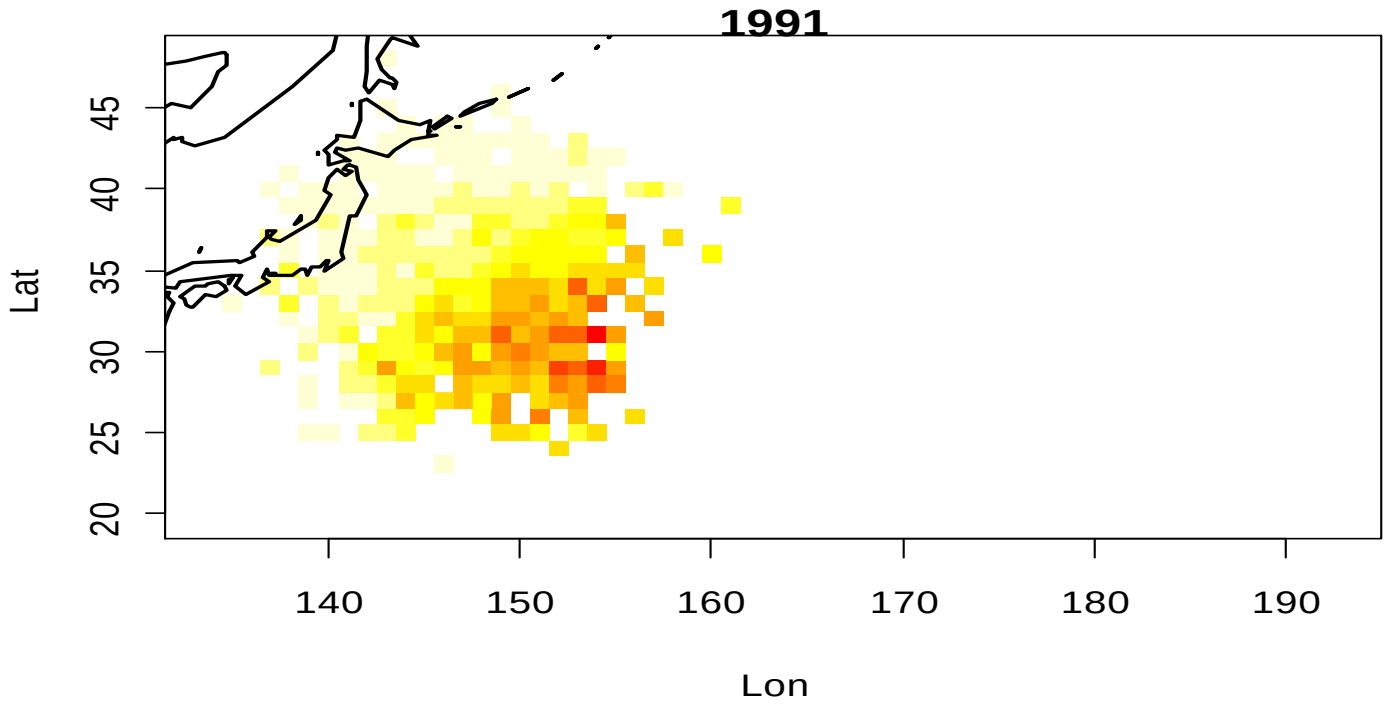
```
> library(maptools) # インストールしておいたライブラリを呼び出す
> library(maps)
> map('world2',add=TRUE,fill=TRUE,col="lightgreen")
```



```
# 年によって分布がどう変わるか？
# x3は3次元の配列となる
> x3 <- tapply(tdat$N,list(tdat$lon,tdat$lat,tdat$year),mean)
> years <- dimnames(x3)[[3]]
> image(x1,y1, x3[,,1], xlab="Lon",ylab="Lat", col=rev(heat.colors(12)))
> map('world2',lwd=2,add=T)
> title(years[1])
```



```
>image(x1,y1, x3[,,2], xlab="Lon",ylab="Lat", col=rev(heat.colors(12)))
>map('world2',lwd=2,add=T)
>title(years[2])
```



# for文による繰り返し

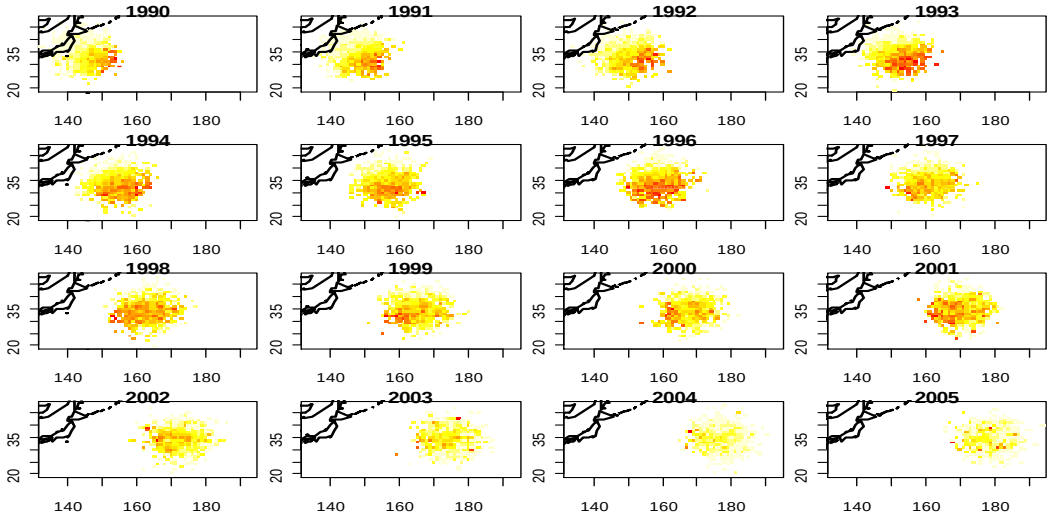
```
>for(i in 1:length(years)){
>       image(x1,y1, x3[,,i], xlab="Lon",ylab="Lat",
                                        col=rev(heat.colors(12)))
>       map('world2',lwd=2,add=T)
>       title(years[i])
>}
```

- 複数の図が一気に描画されるので，最後の図しか見ることはできない
- 「R Graphics」を選択した状態で，メニューの「履歴」→「記録」をクリックすると，過去の図を見ることができるようになる
- 「記録」にチェックを入れた状態で，もう一度コマンドを実施
→PageUp

# for文による繰り返し&複数図を１画面に

```
>par(mfrow=c(4,4),mar=c(2,2,1,0.5))
>for(i in 1:length(years)){
>    image(x1,y1, x3[,,i], xlab="Lon",
                  ylab="Lat", col=rev(heat.colors(12)))
>    map('world2',lwd=2,add=T)
>    title(years[i])
>}
```

漁場の位置が東に移動している

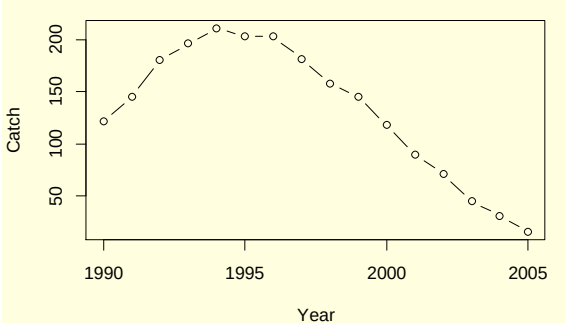



- par(mfrow=c(4,4)) コマンドを使うと，１枚のパネルに16枚の図が書けるようになる
- mar= のオプションは，margin（余白）の大きさ

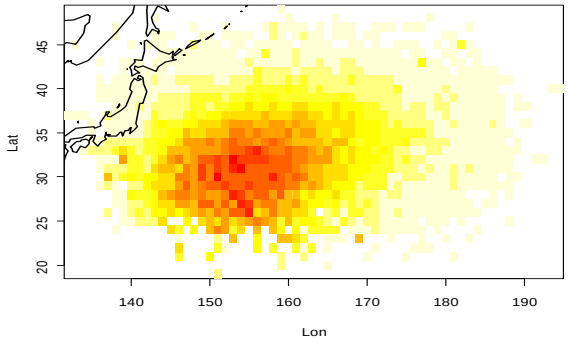
# おまけ：簡単なCPUE解析

# 年によるCPUEの変化は，漁場位置の変化か？資源量の変化か？

- CPUEは増加したのちに減少



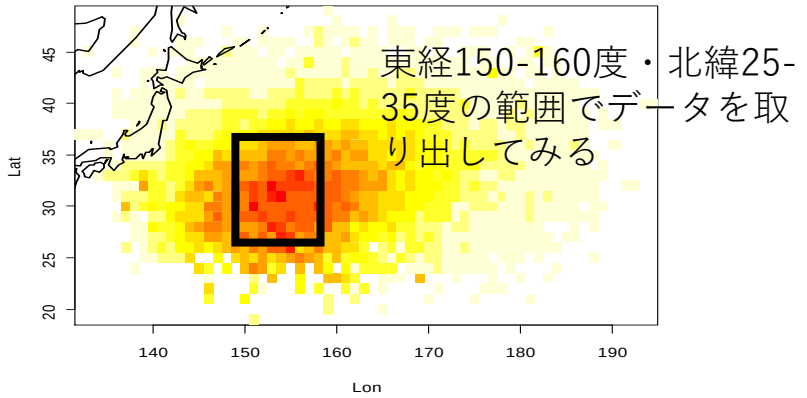

←CPUEの年トレンド

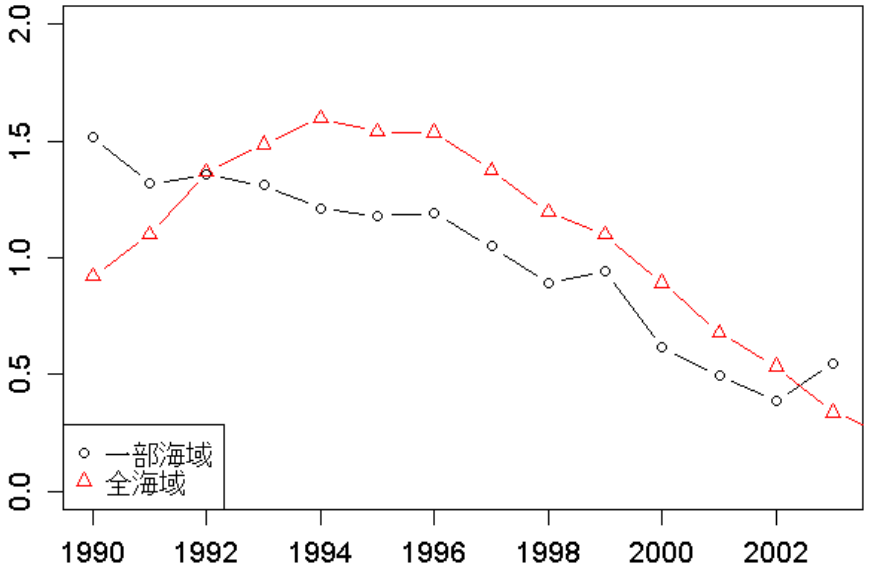



←CPUEの空間分布

- 漁場の位置は，年々東へ

# アプローチA. 同じ海域でCPUEのトレンドを比較してみる

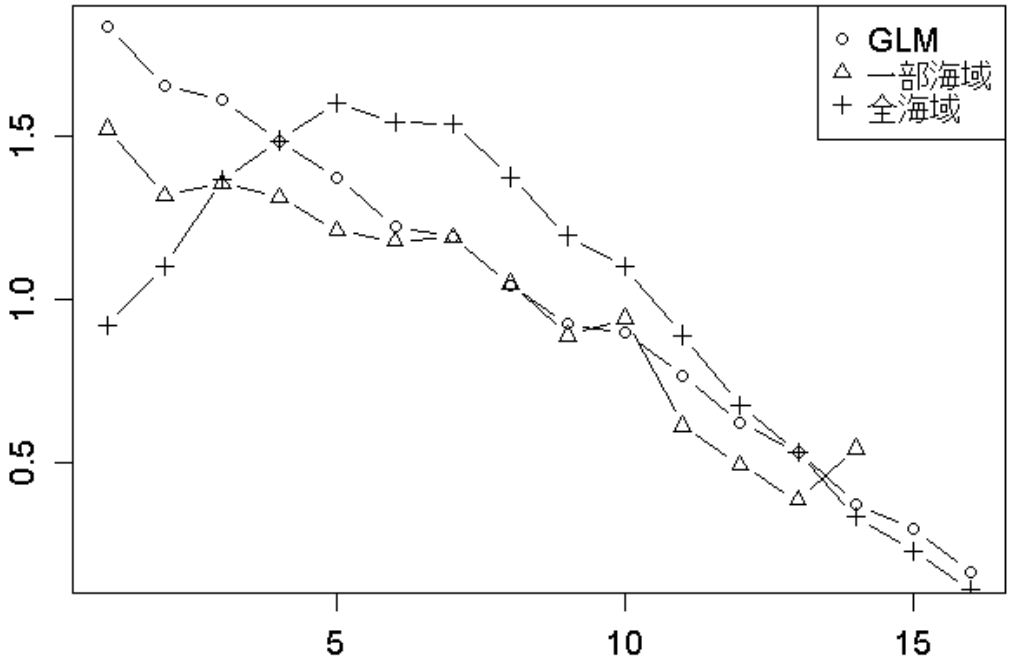



```
> tdat2 <- subset(tdat,
   lon>149 & lon<161 & lat>24 & lat<36)
> cpue2 <- tapply(tdat2$N, tdat2$year, mean)
> plot(names(cpue2), cpue2/mean(cpue2),
        type="b", xlab="Year",ylab="CPUE",
        ylim=c(0,2))
> cpue1 <- tapply(tdat$N, tdat$year, mean)
> points(names(cpue1),cpue1/mean(cpue1),
          type="b",pch=2,col=2)
> legend("bottomleft",pch=1:2,col=1:2,
          legend=c("一部海域","全海域"))
```

- 一部海域だけで見た場合と，全海域で見た場合で，傾向はかなり異なる
- 全海域のデータを使うと，漁場の移動の影響が年のCPUEの変化だと，誤って捉えられてしまう

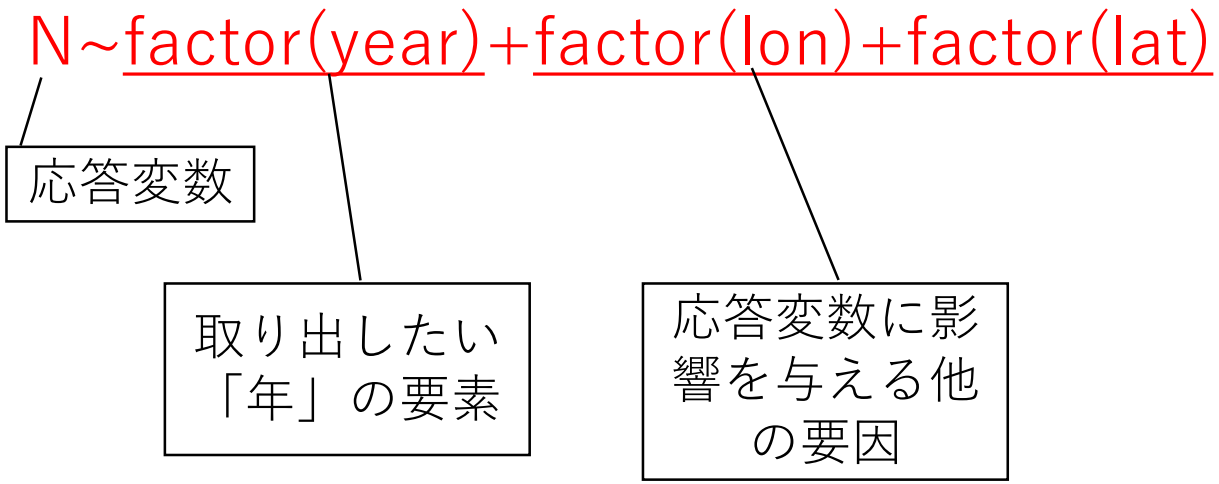
# アプローチB. 一般化線形モデル（GLM）を用いた解析 (CPUE標準化)




```
> gres <- glm(N~factor(year)+factor(lon)+
                factor(lat),
  data=tdat,family="poisson")
> cpue3 <- exp(c(0,gres$coef[2:16]))
> plot(cpue3/mean(cpue3),type="b")
> points(cpue2/mean(cpue2),type="b",pch=2)
> points(cpue1/mean(cpue1),type="b",pch=3)
> legend("topright", pch=1:3,
        legend=c("GLM","一部海域","全海域"))
```

# CPUE標準化とは？

- 統計的な手法（GLM等）を用いて，応答変数に影響を与える様々な要因を特定し，その中から資源量指数を取り出すこと
- 例データでの応答変数＝N
- 資源量指数＝年によるCPUEの変化
- そのほかの要因＝緯度，経度



# CPUE標準化についての参考文献


- Maunder, Punt AE. 2004. Standardizing Catch and Effort Data: A Review of Recent Approaches. Fish. Res. 70, 141-159
- 庄野. 2004. CPUE標準化に用いられる統計学的アプローチに関する総説. 水産海洋研究 68, 106-120
- 大気海洋研究所共同利用シンポジウム発表資料 「日本延縄漁業データを用いた標準化CPUE」 http://cse.fra.affrc.go.jp/ichimomo/Tuna/glm_longline_presentation.pdf

# 補足1

# 等高線図のプロット

```
# 経度・経度ごとの平均
>  x <- tapply(tdat$N,
            list(tdat$lon,tdat$lat),mean)
# これを2次元平面上にプロットしてみる
>  x1 <- as.numeric(dimnames(x)[[1]])
>  y1 <- as.numeric(dimnames(x)[[2]])
>  image(x1,y1, x, xlab="Lon",ylab="Lat",
                col=rev(heat.colors(12)))
>  contour(x1,y1,x,add=TRUE)
```

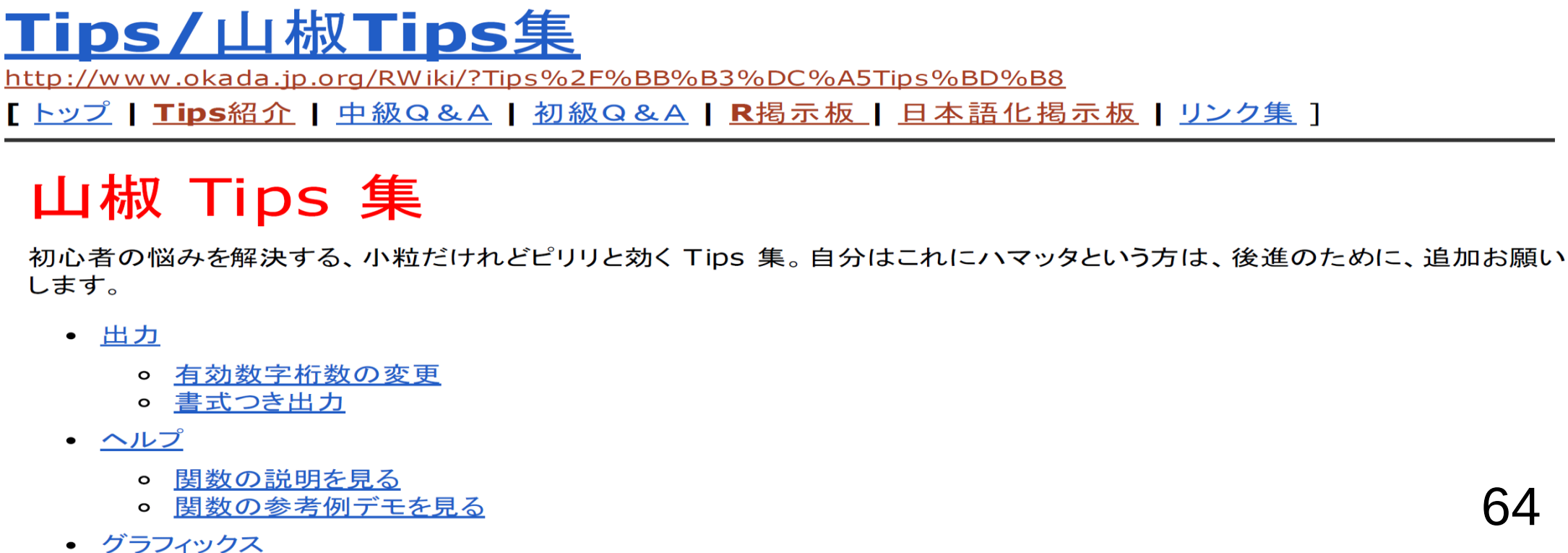
# 補足2

```
# 解像度の高い日本地図
> library(mapdata)
> map("japan")
> map.axes()

> map("japan",xlim=c(135,140),
      ylim=c(32,35),fill=T,col="gray")
> map.axes()
> points(137, 33)
```

# R情報

Rjpwiki　http://www.okada.jp.org/RWiki/

Tips/山椒Tips集
http://www.okada.jp.org/RWiki/?Tips%2F%BB%B3%DC%A5Tips%BD%B8



**Tips/山椒Tips集**

http://www.okada.jp.org/RWiki/?Tips%2F%BB%B3%DC%A5Tips%BD%B8

[ トップ | Tips紹介 | 中級Q&A | 初級Q&A | R掲示板 | 日本語化掲示板 | リンク集 ]

## 山椒 Tips 集

初心者の悩みを解決する、小粒だけれどピリリと効く Tips 集。自分はこれにハマッタという方は、後進のために、追加お願いします。

- 出力
  - 有効数字桁数の変更
  - 書式つき出力
- ヘルプ
  - 関数の説明を見る
  - 関数の参考例デモを見る
- グラフィックス

# R情報

Rjpwiki　http://www.okada.jp.org/RWiki/

知っているといつか役に立つ(?)関数達

http://www.okada.jp.org/RWiki/index.php?%C3%CE%A4%C3%A4%C6%A4%A4%A4%EB%A4%C8%A4%A4%A4%C4%A4%AB%CC%F2%A4%CB%CE%A9%A4%C4%28%3F%29%B4%D8%BF%F4%C3%A3

## 知っているといつか役に立つ(?)関数達

http://www.okada.jp.org/RWiki/?%C3%CE%A4%C3%A4%C6%A4%A4%A4%EB%A4%C8%A4%A4%A4%C4%A4%AB%CC%F2%A4%CB%CE%A9%A4%C4%28%3

[ トップ | Tips紹介 | 中級Q&A | 初級Q&A | R掲示板 | 日本語化掲示板 | リンク集 ]

### 知っているといつか役にたつ(?)関数達

とりあえず使いそうもないが、知っているといつか役にたつかも知れない関数達(およびその使いかた)のメモです。r-help を見ていると「へぇー、こんな
のように見つかりますね。メモしておかないとすぐ忘れそうなので、という理由で設けた極めて個人的なページですが、皆さんの「へぇー」も勝手に付け

注意:(101)以降は 知っているといつか役にたつ(?)関数達(2) へ続く。

- (100)（百回記念、というほどのものではないが）北大久保さんのブログで見かけた問題の解の例（2007.1.11)
- (99) リストとしての環境（2007.1.6)
- (98) sink 関数のオプション split=TRUE (2007.1.4 2010.10.02追記)
- (97) cat 関数の便利なオプション（要望掲示版記事より, 2007.1.3)
- (96) リストも attach, detach できる (2007.1.2)
- (95) コンソール出力の抑制 (2006.12.28)
- (94) gc() は二度繰り返すべし。(2006.12.27)
- (93) 関数！、関数！、関数！ (2006.12.27)

# R情報

Rjpwiki　http://www.okada.jp.org/RWiki/

グラフィックス参考実例集

http://www.okada.jp.org/RWiki/?%A5%B0%A5%E9%A5%D5%A5%A3%A5%C3%A5%AF%A5%B9%BB%B2%B9%CD%BC%C2%CE%E3%BD%B8

## グラフィックス参考実例集

http://www.okada.jp.org/RWiki/?%A5%B0%A5%E9%A5%D5%A5%A3%A5%C3%A5%AF%A5%B9%BB%B2%B9%CD%BC%C2%CE%E3

[ トップ | Tips紹介 | 中級Q&A | 初級Q&A | R掲示板 | 日本語化掲示板 | リンク集 ]

### グラフィックス参考実例集

プログラムを理解する最短の道は、良い実例を見ることです。同様に R のグラフィックスをマスターする王道は、コードと画像の実例を見ることです。この Tips では R の多彩なグラフィックスの例をできるだけ紹介したいとおもいます。主にックス関数の参考例を紹介します。こういうことができると分かれば、後は工夫次第でどうにでもなります。苦労して作っ作（and/or 凡作）を後進のために紹介して下さい。なお、参考画像は直接 png デバイスで処理したものです。

Rのグラフィックスパラメーター一覧 --- Rのグラフィックスパラメータ

パッケージ grid によるグラフィックス --- grid パッケージ事始