R で学ぶデータ解析とシミュレーション

②

～ グラフ作成入門 ～

２時間目のメニュー

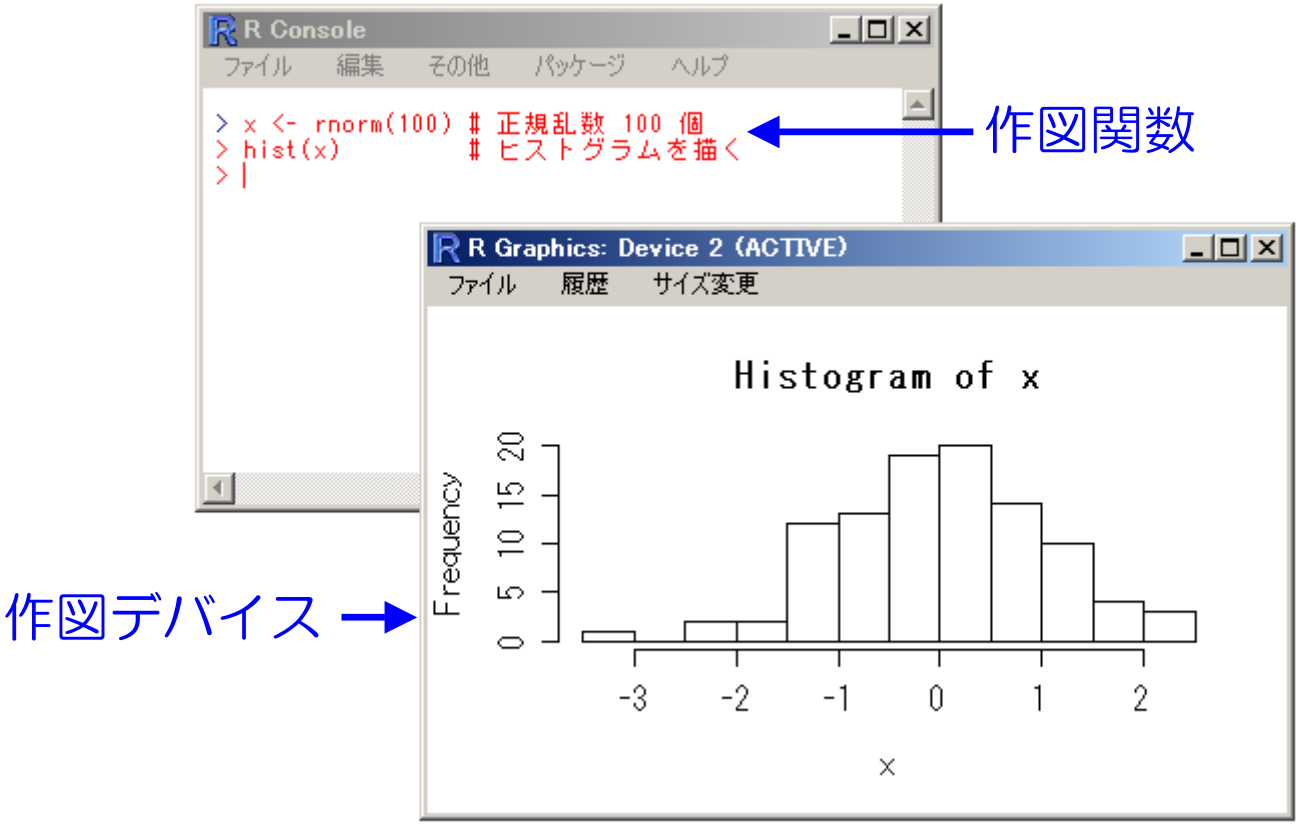
- グラフの作成方法
  - グラフ作成の第一歩 ←
  - 高水準作図関数
    - 高水準作図関数の種類
    - 関数 plot() を用いた作図例
  - 低水準作図関数
    - 低水準作図関数の種類
    - 低水準作図関数を用いた作図例
  - 数学関数のプロット
    - 数学関数の定義方法
    - 数学関数の作図例
  - 参考（重ねた図の描き方，R の画像の編集方法）

グラフィックスは R のセールスポイント！

- 簡単な命令で見栄えの良いグラフを作成することが出来る
- 図のカスタマイズが非常に簡単
  - 低水準作図関数で完成図に追記が可能
  - 専用のパッケージを使ってカスタマイズすることも可能
- 複雑な図や立体的な図も簡単に描くことが出来る
- 様々な種類の画像ファイルに保存することが出来る

グラフ作成の第一歩

- 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）



グラフ作成の第一歩

- 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

グラフ作成の第一歩

- 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

グラフ作成の流れ

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

- ここではまずデータをプロットすることを習得する
- 「1. データを準備する」「2. グラフを描く」の 2 手順だけで，とりあえずグラフが作成できることに注目！

【例】1. データの準備

■ 以下の様な体重に関するデータについてグラフを描く

体重	59	60	61	62	62	63	63	63	63	64
	64	64	65	65	65	66	66	67	68	72

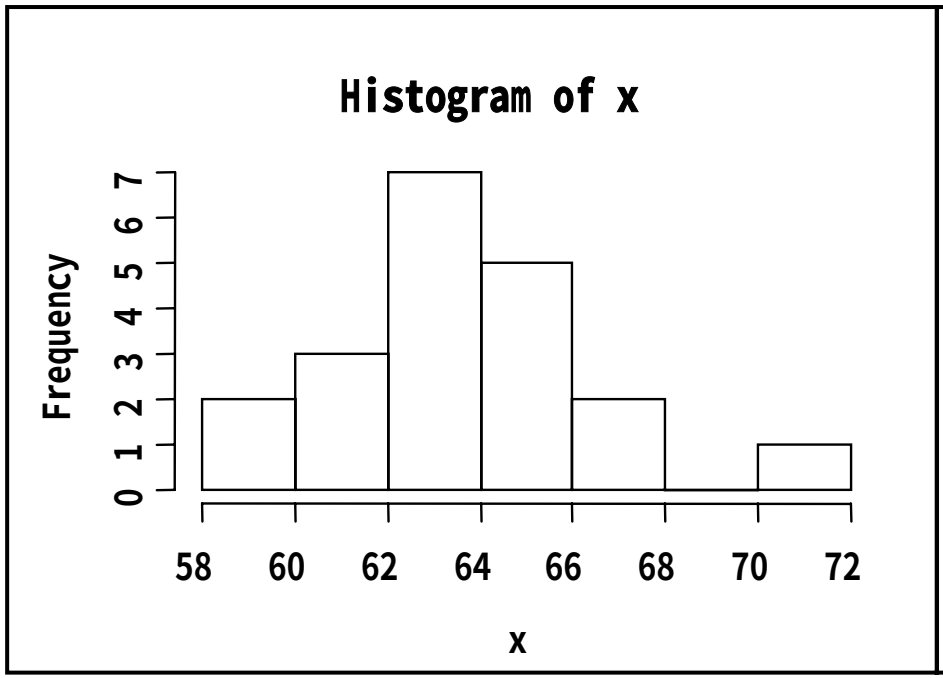
■ まずは上記体重データを変数 x に代入する

```
> x <- c(59, 60, 61, 62, 62, 63, 63, 63, 63, 64,
+        64, 64, 65, 65, 65, 66, 66, 67, 68, 72)
> x
[1] 59 60 61 62 62 63 63 63 63 64 64 64 65 65 65 ...
```

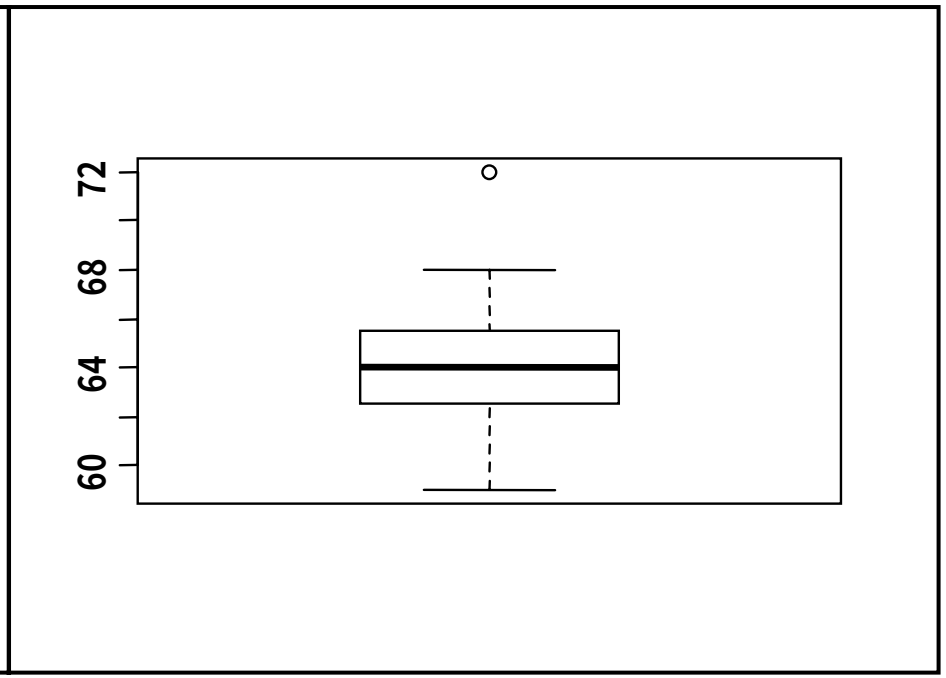
【例】2. グラフの作成

■ 体重データ x に関するヒストグラムと箱ひげ図を作成

```
> hist(x)      # ヒストグラムを描く
> boxplot(x)   # 箱ひげ図を描く
```



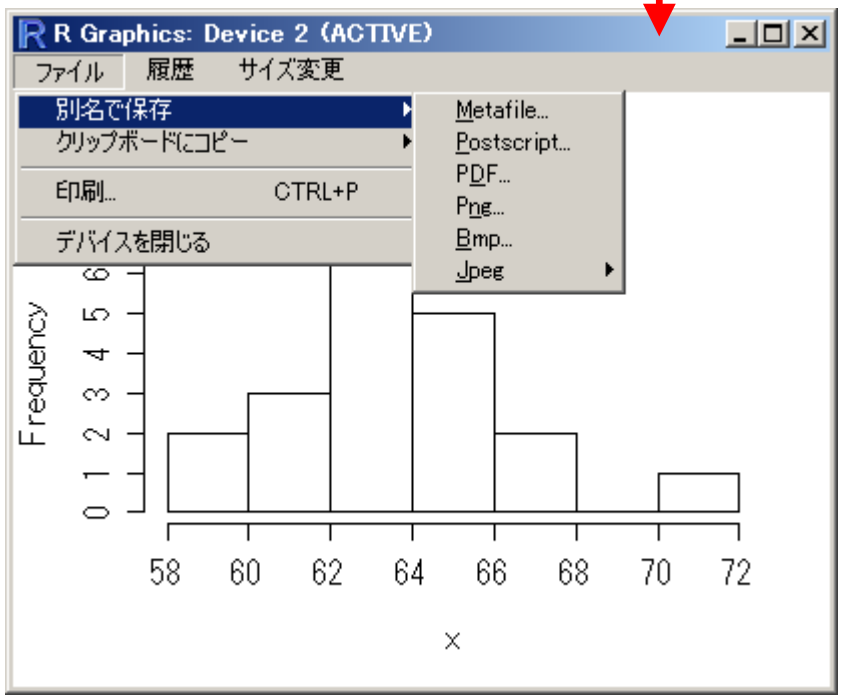
ヒストグラム



箱ひげ図

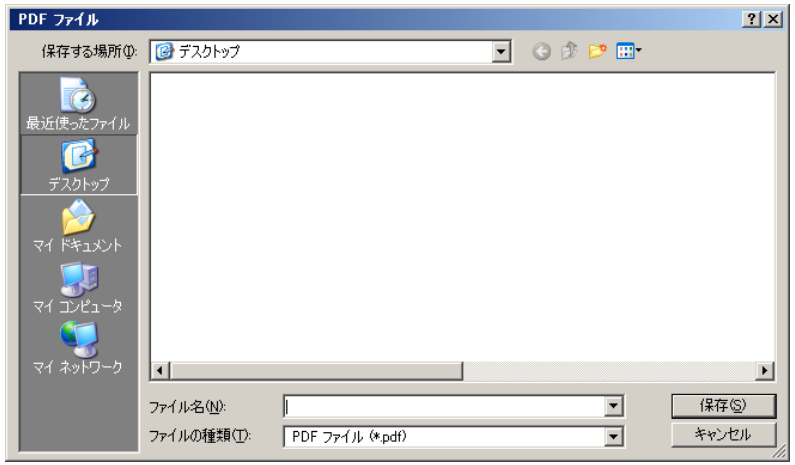
【例】4. 描いたグラフの保存

- [ファイル] → [別名で保存] → [画像形式] を選択した後
- 画像ファイルの名前を入力して [保存] をクリック



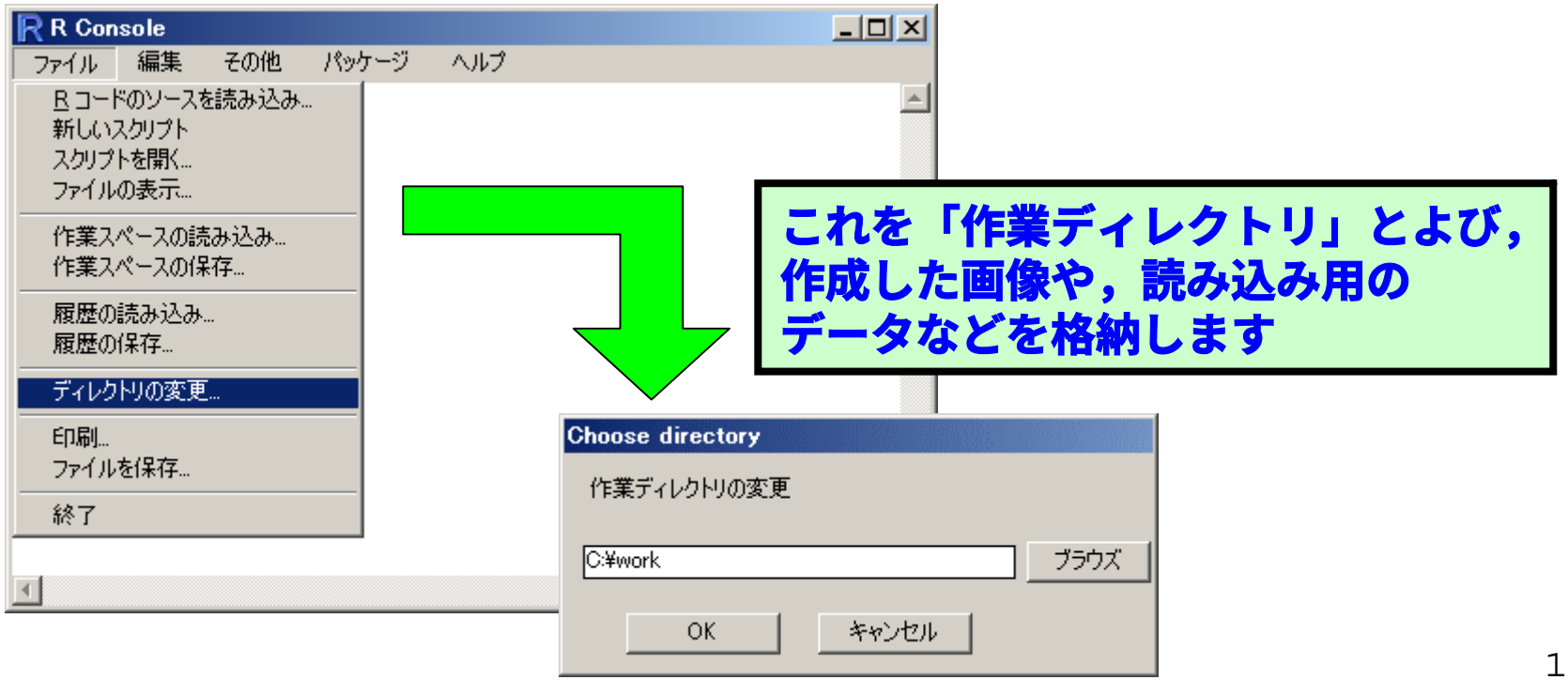
作図デバイスを
アクティブ（青）
にすること！！

画像形式を選択



【演習】作業フォルダの作成＆変更【準備】

1. まず，パソコンのデスクトップに「work」という
   フォルダを作成してください
   次に，R の「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



【演習】

2. 8 枚目に出てきた体重データを変数 x に代入してください
3. 体重データ x のヒストグラムを描いてください
4. 3. で作成したヒストグラムを画像ファイルとして保存してください
5. ヒストグラムを描く関数 hist() のヘルプを見て，ヒストグラムを青色で塗りつぶしてください

5. 関数 hist() のヘルプ

② **Usage** ← ② 関数の雛形

```
hist(x, ...)

## Default S3 method:
hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
     include.lowest = TRUE, right = TRUE,
     density = NULL, angle = 45, col = NULL, border = NULL,
     main = paste("Histogram of" , xname),
     xlim = range(breaks), ylim = NULL,
     xlab = xname, ylab,
     axes = TRUE, plot = TRUE, labels = FALSE,
     nclass = NULL, ...)
```

※ col="・・・" とすれば良い？

5. 関数 hist() のヘルプ

⑩ Examples ← ⑩ 例

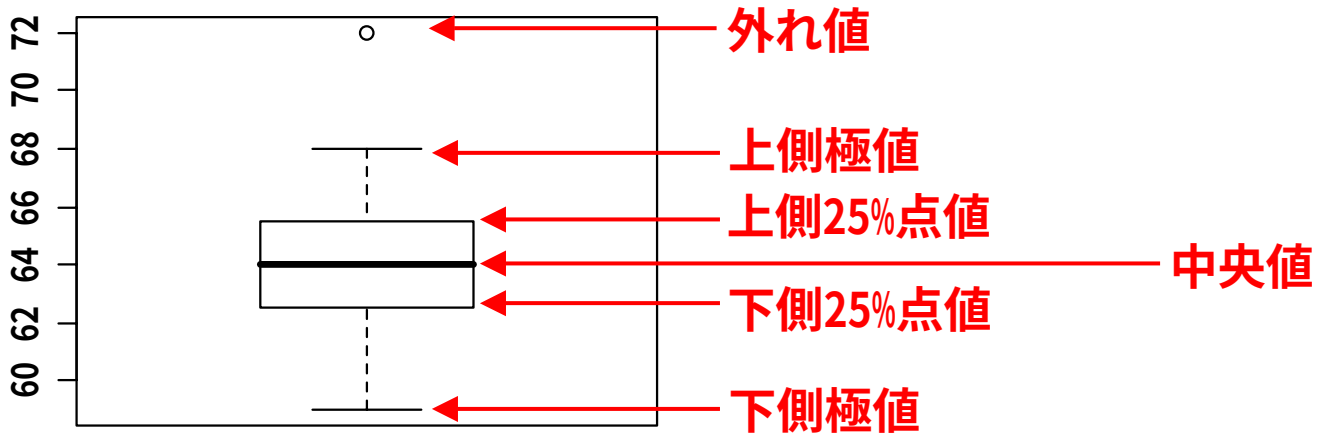
```
op <- par(mfrow=c(2, 2))
hist(islands)
utils::str(hist(islands, col="gray", labels = TRUE))

hist(sqrt(islands), breaks = 12, col="lightblue", border="pink")
##-- For non-equidistant breaks, counts should NOT be graphed ... :
r <- hist(sqrt(islands), breaks = c(4*0:5, 10*3:5, 70, 100, 140),
          col='blue1')
text(r$mids, r$density, r$counts, adj=c(.5, -.5), col='blue3')
sapply(r[2:3], sum)
sum(r$density * diff(r$breaks)) # == 1
lines(r, lty = 3, border = "purple") # -> lines.histogram(*)
par(op)
```

※ col="・・・" とすれば良い？

【参考】箱ひげ図

■ 関数の引数を使って箱ひげ図を装飾することも出来る

関数	機能
boxplot(x, xlab="体重")	箱ひげ図の名前を指定する
boxplot(x, col=c("blue"))	箱を塗りつぶす色を指定する
boxplot(x, horizontal=T)	箱ひげ図を横に寝かせる



※ 極値は箱から四分位範囲の1.5倍である最も端にあるデータ点となる
⇒ 上側極値の場合は「上側25%点＋1.5×(上側25%点－下側25%点)」

【参考】棒グラフと円グラフ

- 体重や身長などのデータ について棒グラフや円グラフを描く場合は，まず度数分布表を作成する

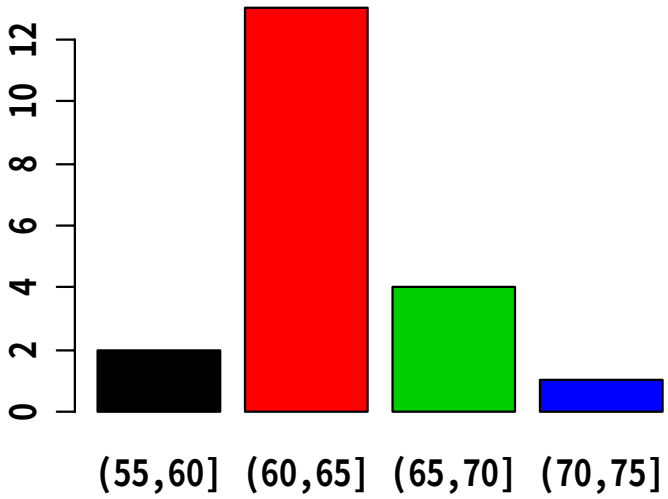
table(cut(元のベクトル，度数分布の区間))

- 10 枚目に出てきた体重データ x について棒グラフや円グラフを描く場合は，まず度数分布表を作成する

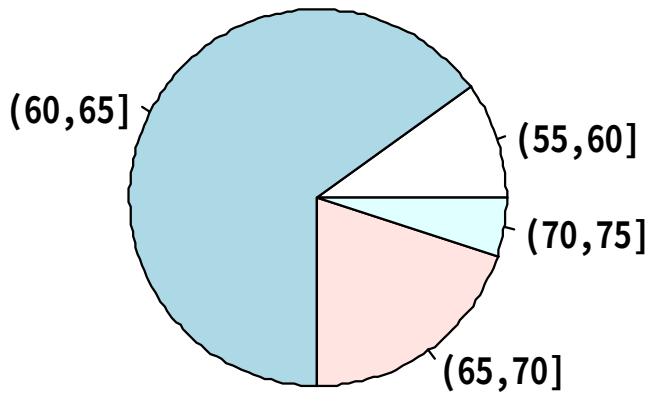
```
> x <- c(59, 60, 61, 62, 62, 63, 63, 63, 63, 64,
+        64, 64, 65, 65, 65, 66, 66, 67, 68, 72)
> y <- table( cut(x, c(55, 60, 65, 70, 75)) ) # 度数分布表を作成
> y                                            # 度数分布表を表示
(55,60] (60,65] (65,70] (70,75]
      2      13       4       1
```

【参考】棒グラフと円グラフ

- 度数分布表 y についてプロット



棒グラフ
barplot(y, col=1:4)



円グラフ
pie(y, r=1.0)

２時間目のメニュー

- グラフの作成方法
  - グラフ作成の第一歩
  - **高水準作図関数 ←**
    - **高水準作図関数の種類**
    - **関数 plot() を用いた作図例**
  - 低水準作図関数
    - 低水準作図関数の種類
    - 低水準作図関数を用いた作図例
  - 数学関数のプロット
    - 数学関数の定義方法
    - 数学関数の作図例
  - 参考（重ねた図の描き方，R の画像の編集方法）

高水準作図関数とグラフの作成

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

- 高水準作図関数：1 枚の完成された図を描く
- 作図できるグラフ：
  散布図，ヒストグラム，棒グラフ，円グラフ，箱ひげ図，
  数学関数のプロット・・・，他多数
- 高水準作図関数を実行すると，自動的に作図デバイスが起動されてウインドウに図が表示される

高水準作図関数の一覧

グラフの種類	関数
散布図	**plot(ベクトル)，plot(ベクトル1，ベクトル2)**
ヒストグラム	**hist(ベクトル)**
箱ひげ図	**boxplot(ベクトル)**
棒グラフ	**barplot(各カテゴリの頻度が入ったベクトル)**
円グラフ	**pie(各カテゴリの頻度が入ったベクトル)**
分割表の図	**mosaicplot(行列データ)**
スターチャート	**stars(行列データ)**
対散布図	**pairs(行列データ)**
1 次元関数のグラフ	**curve(関数，左端，右端)**
2 次元関数のグラフ	**persp(x軸，y軸，z軸)**

関数 plot()

- R で一番良く使われる高水準作図関数が関数 plot()
- 最も基本的で機能も多い関数も plot()
- 散布図や折れ線グラフなどを描くことが出来る
- x 座標のデータだけを指定してプロットする

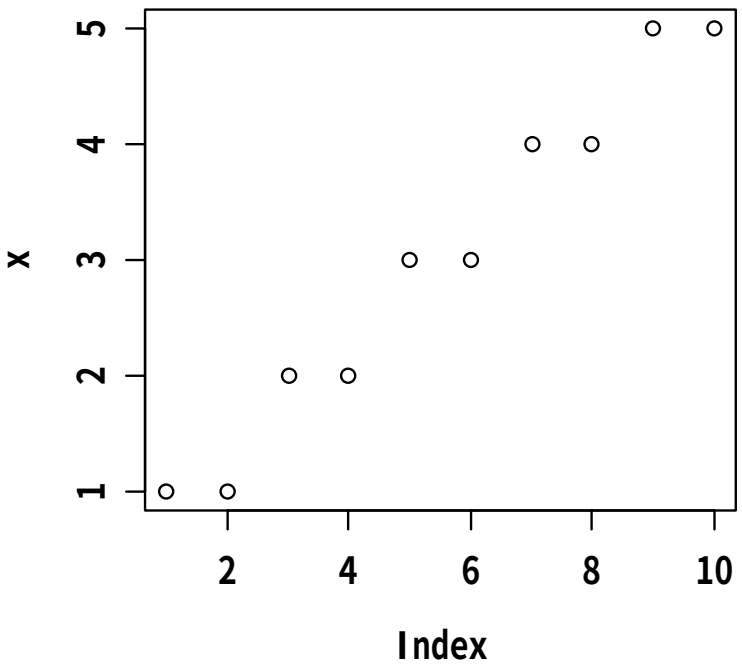
```
> x <- c(1, 1, 2, 2, 3, 3, 4, 4, 5, 5)
> plot(x)    # y 座標は自動的に 1, 2, … となる
```

- x 座標とy 座標のデータを指定してプロットする

```
> x <- 1:10; y <- 1:10    # x <- c(1,2,…,10) と同じ
> plot(x, y)              # 範囲は自動で決まる
```

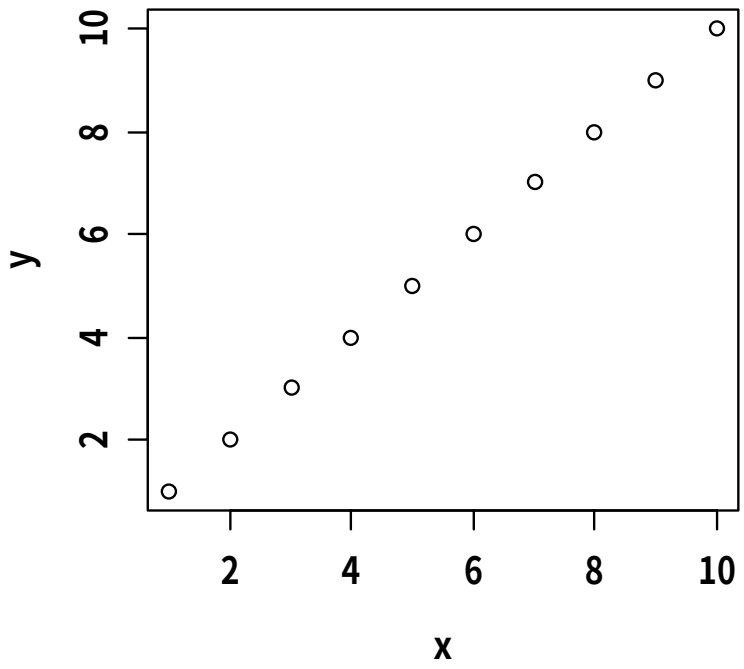
関数 plot()

時系列データ




```
> x <- c(1,1,2,2,3,3,4,4,5,5)
> plot(x)
```

散布図




```
> x <- 1:10;    y <- 1:10
> plot(x, y)
```

関数 plot()：引数の追加

- データの指定以外に「x 軸の範囲」「図のタイトル」「プロットの色」などを決めることが出来る
- 例えば，プロット範囲は引数 xlim, ylim で決める

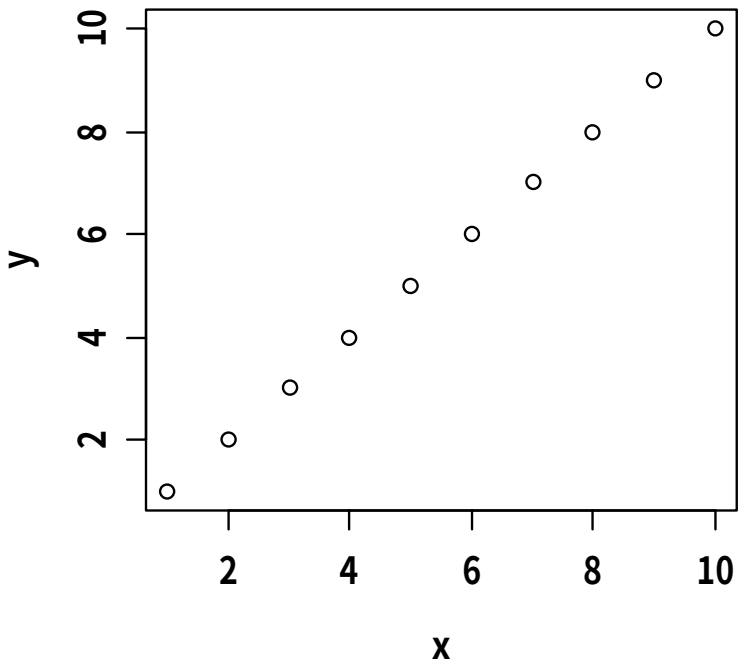
```
> x <- 1:10; y <- 1:10      # x <- c(1,2,…,10) と同じ
> plot(x, y, xlim=c(1,10))
```

- 引数 xlim の値を工夫すると，x 軸の正の向きを左向きにすることができる

```
> plot(x, y, xlim=c(10,1)) # x 軸の正方向を逆にする
```

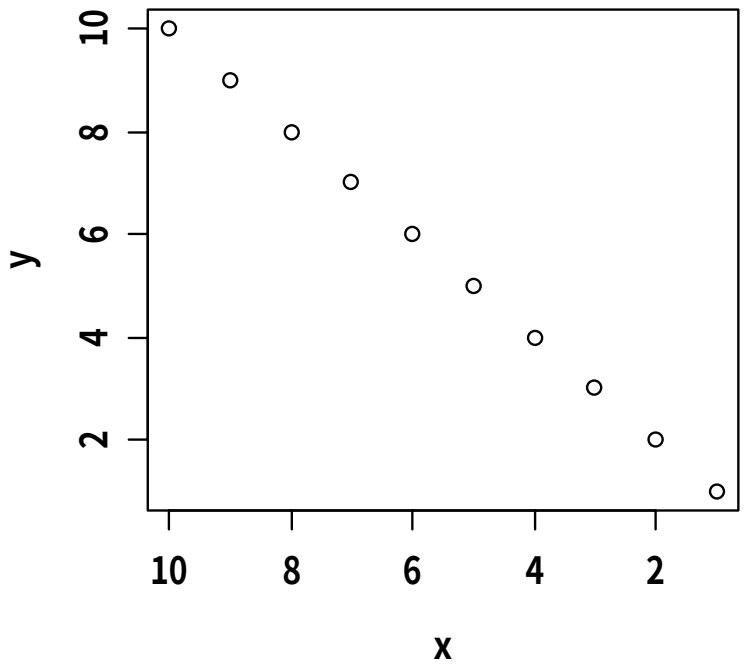
関数 plot()：引数の追加

散布図（Xの範囲：1～10）




```
> x <- 1:10; y <- 1:10
> plot(x, y, xlim=c(1,10))
```

散布図（Xの範囲：10～1）




```
> x <- 1:10; y <- 1:10
> plot(x, y, xlim=c(10,1))
```

関数 plot()：タイトル・ラベルの制御

引数	機能
main="文字"	タイトルを与える文字列を指定する
sub="文字"	サブタイトルを与える文字列を指定する
xlab="文字", ylab="文字"	それぞれ x 座標名， y 座標名を与える文字列を指定する （省略すると，変数名前が座標名となる）
ann = F	軸のラベルを描かないようにする （xlab = “” ， ylab = “” を指定した場合と同じ）
tmag=1.2	タイトルなどに使われるテキストの拡大率を指定する

関数 plot()：軸の種類

引数	機能
log="x"	"x" (x 対数軸) , "y" (y 対数軸) , "xy" (両対数軸) の何れかを指定する （対数は常用対数のみ）
xlim=c(0, 1), ylim=c(-1, 1)	長さ 2 のベクトルで x 座標, y 座標の 最小値と最大値を与える ⇒ xlog, ylog で対数プロットが出来る ⇒ ベクトルを降順に並べる (例：c(2, -2)) とプロットの向きが逆になる
axes=FALSE	軸の生成を抑制する ⇒ xaxs, yaxs で指定することも出来る

関数 plot()：プロットする点の形式

引数	機能
type="p"	点プロット(デフォルト)
type="l"	線プロット(折れ線グラフ)
type="b"	点と線のプロット
type="c"	"b" において点を描かないプロット
type="o"	点プロットと線プロットの重ね書き
type="h"	各点から x 軸までの垂線プロット
type="s"	左側の値にもとづいて階段状に結ぶ
type="S"	右側の値にもとづいて階段状に結ぶ
type="n"	軸だけ描いてプロットしない （続けて低水準関数でプロットする場合など）

関数 plot()：引数 lty について

引数	機能
lty=0 ， lty="blank"	線を透明の線にする
lty=1 ， lty="solid"	線を実線にする
lty=2 ， lty="dashed"	線をダッシュにする
lty=3 ， lty="dotted"	線をドットにする
lty=4 ， lty="dotdash"	線をドットとダッシュにする
lty=5 ， lty="longdash"	線を長いダッシュにする
lty=6 ， lty="twodash"	線を二つのダッシュにする

関数 plot()：色，記号など

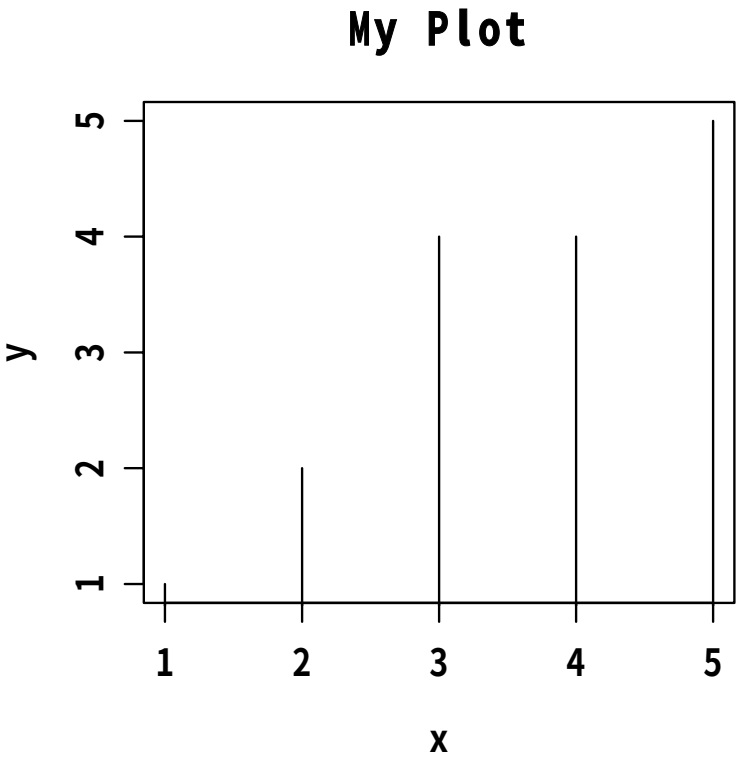
引数	機能
col=1, col="blue"	番号または色名でプロットに使う色を指定する 番号は 1 から順に「黒，赤，緑，青，水色，紫，黄，灰」
pch=0, pch="文字"	プロット点の種類を指定する（詳細は次のスライド）
cex = 1	文字の拡大率を指定する
lwd = 1	線の幅を指定する（値が大きいほど太い線）
bg = "blue"	背景の色を指定する
ann = F	F(FALSE)を指定すると軸と全体のタイトルを描かなくなる
las = 0	0：軸のラベルを各軸に並行して描く 1：軸のラベルをすべて水平に描く 2：軸のラベルを軸に対して垂直に描く 3：軸のラベルをすべて垂直に描く

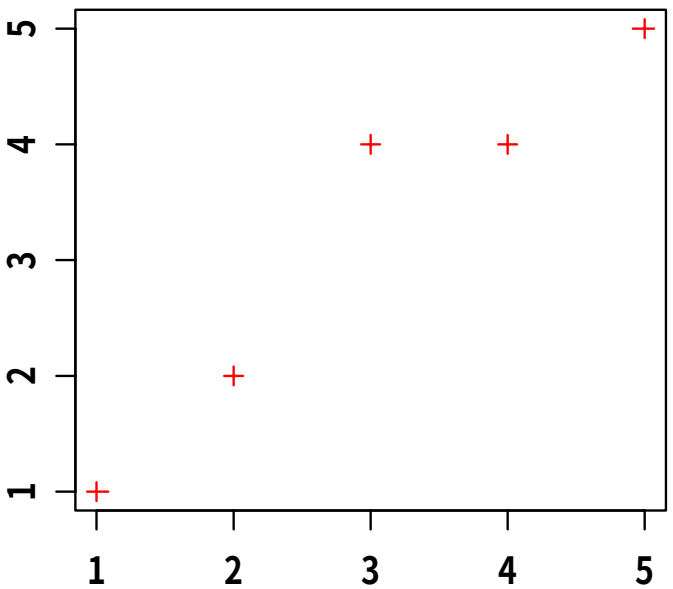
関数 plot()：引数 pch について

0	1	2	3	4	5	6	7	8
□	○	△	＋	×	◇	▽	⊠	✳
9	10	11	12	13	14	15	16	17
◇＋	⊕	✡	⊞	⊗	▽△	■	●	▲
18	19	20	21	22	23	24	25	
◆	●	•	○	□	◇	△	▽	

【演習】

1. [1, 2, 3, 4, 5] というベクトル x を作成してください
2. [1, 2, 4, 4, 5] というベクトル x を作成してください
3. 左下図のような散布図を描いてください
4. 右下図のような散布図を描いてください（点＝赤色）





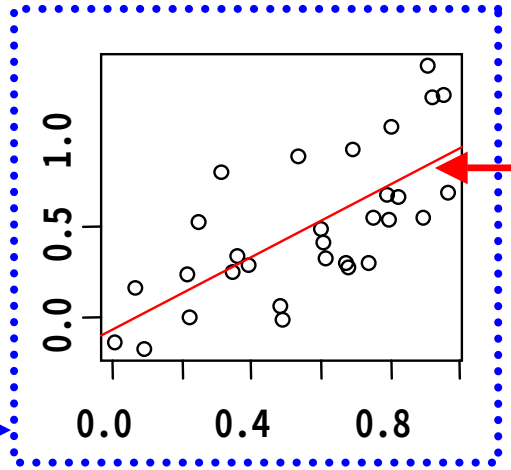
２時間目のメニュー

- グラフの作成方法
  - グラフ作成の第一歩
  - 高水準作図関数
    - 高水準作図関数の種類
    - 関数 plot() を用いた作図例
  - 低水準作図関数 ←
    - 低水準作図関数の種類
    - 低水準作図関数を用いた作図例
  - 数学関数のプロット
    - 数学関数の定義方法
    - 数学関数の作図例
  - 参考（重ねた図の描き方，R の画像の編集方法）

低水準作図関数とグラフの作成

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

- 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
- いきなり低水準作図関数で図を描くことは出来ない
- 追記できるもの：
  点，直線，線分，格子，
  矢印，矩形，文字，枠，
  軸，凡例，多角形　など



低水準作図関数で追記した直線

高水準作図関数で作成した図

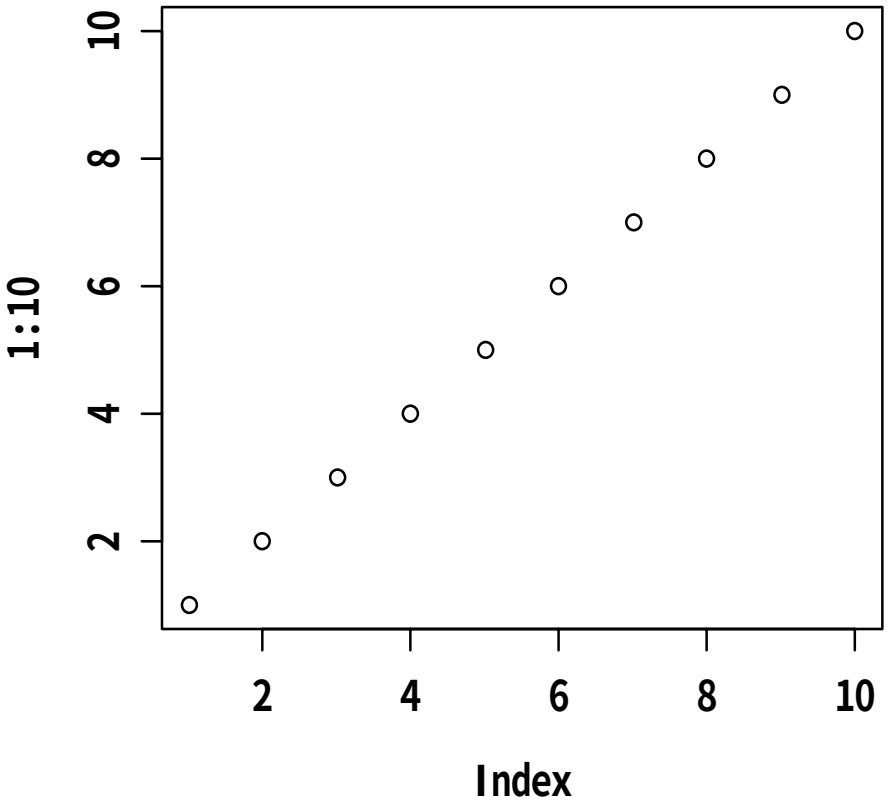
低水準作図関数の一覧

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()**

低水準作図関数を使った図形の追記（1）

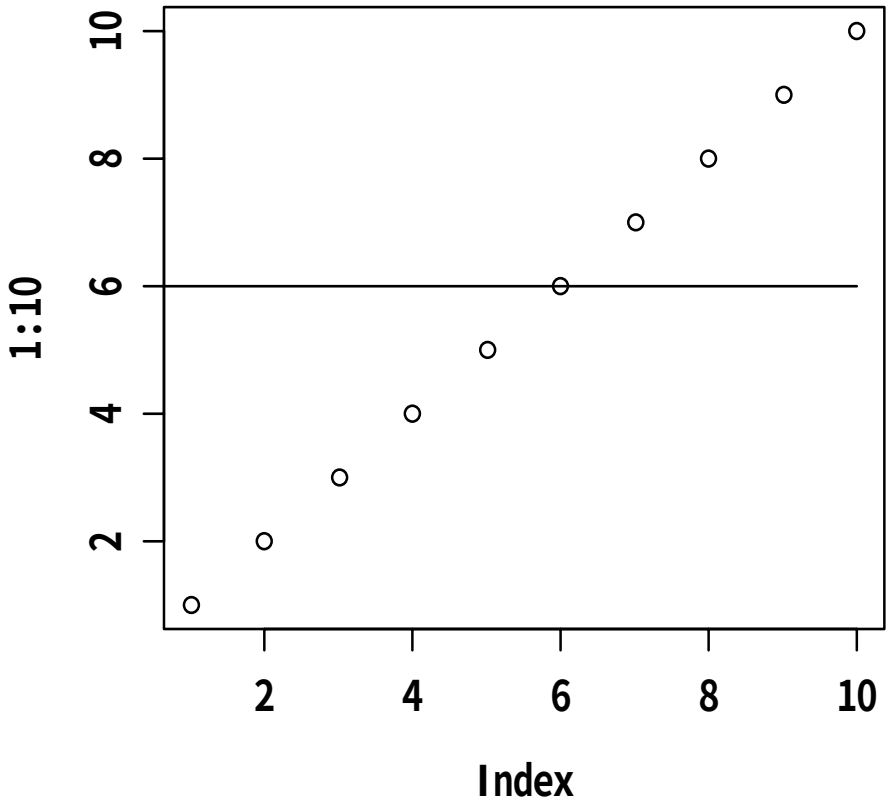
- 関数 plot() を用いて散布図を描いた後・・・

```
> plot(1:10)
```



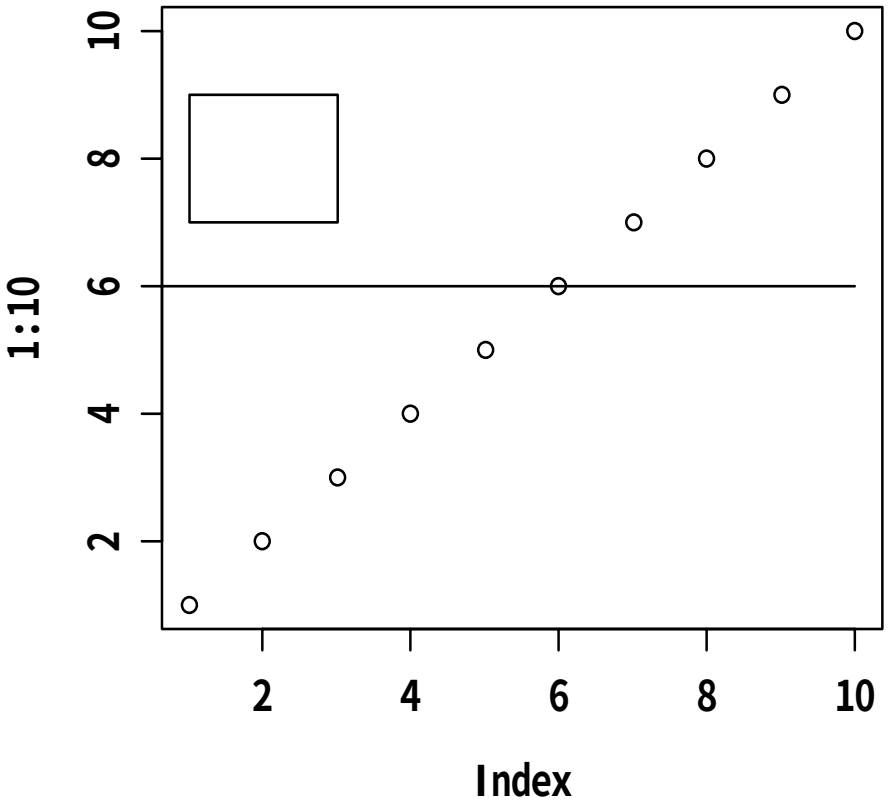
低水準作図関数を使った図形の追記（1）

```
> plot(1:10)
> lines(c(0,10), c(6,6))
```



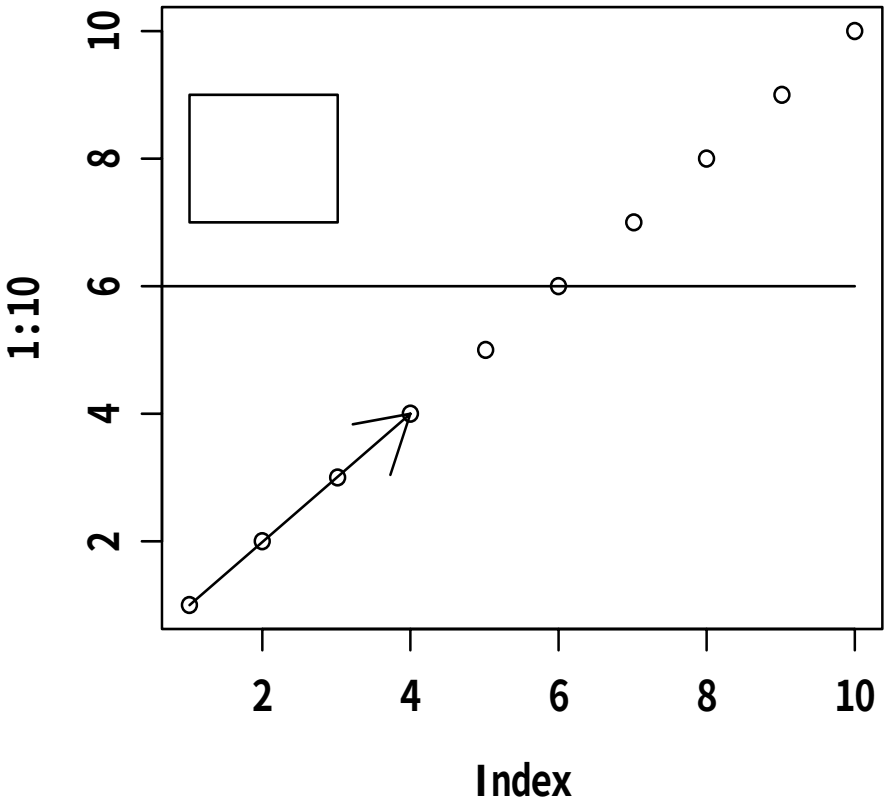
低水準作図関数を使った図形の追記（1）

```
> plot(1:10)
> lines(c(0,10), c(6,6))
> rect(1,7, 3,9)
```



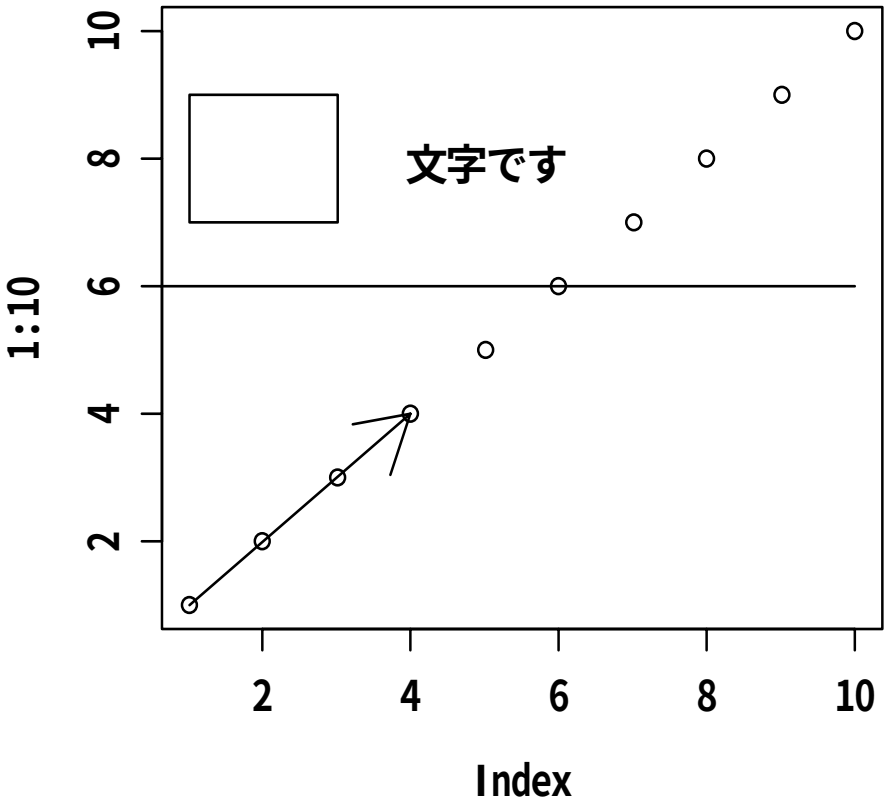
低水準作図関数を使った図形の追記（1）

```
> plot(1:10)
> lines(c(0,10), c(6,6))
> rect(1,7, 3,9)
> arrows(1,1, 4,4)
```



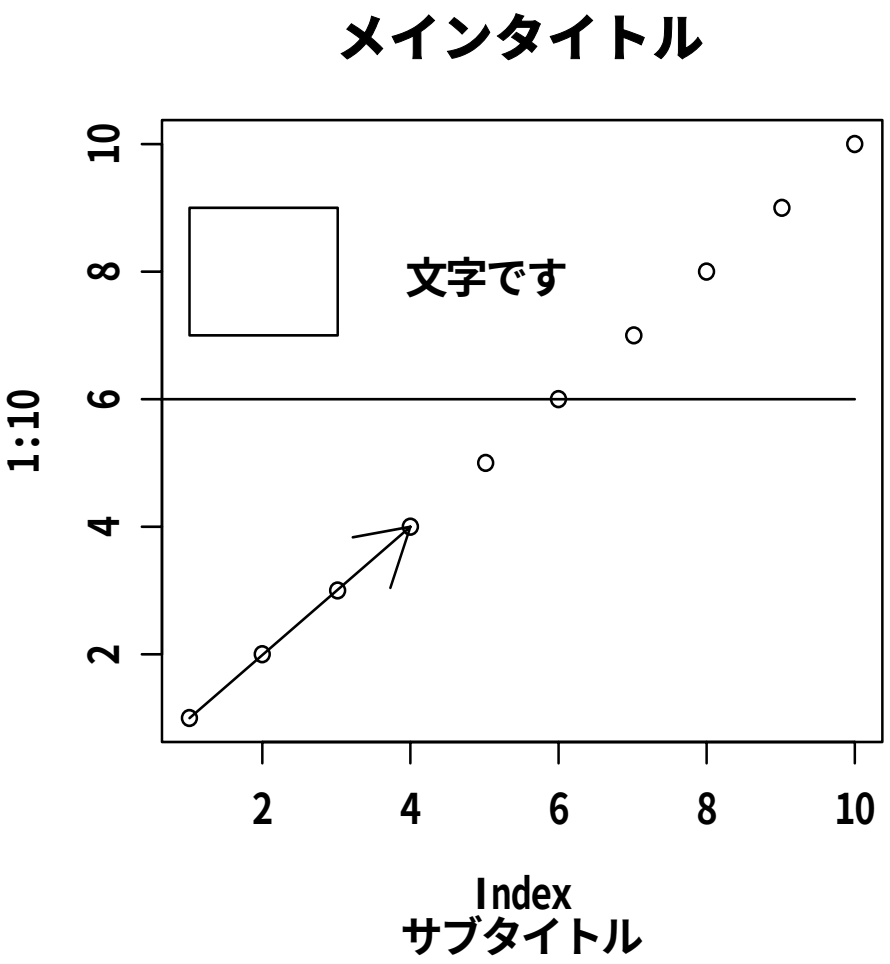
低水準作図関数を使った図形の追記（1）

```
> 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")
```



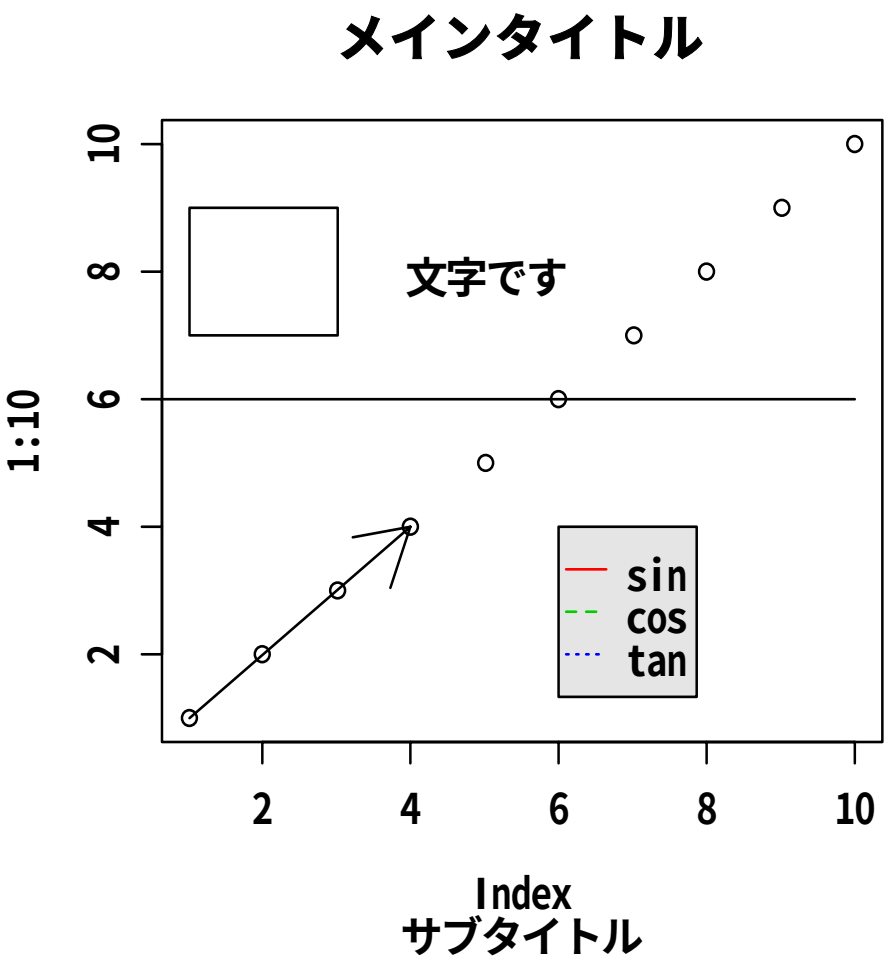
低水準作図関数を使った図形の追記（1）

```
> 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")
```



低水準作図関数を使った図形の追記（1）

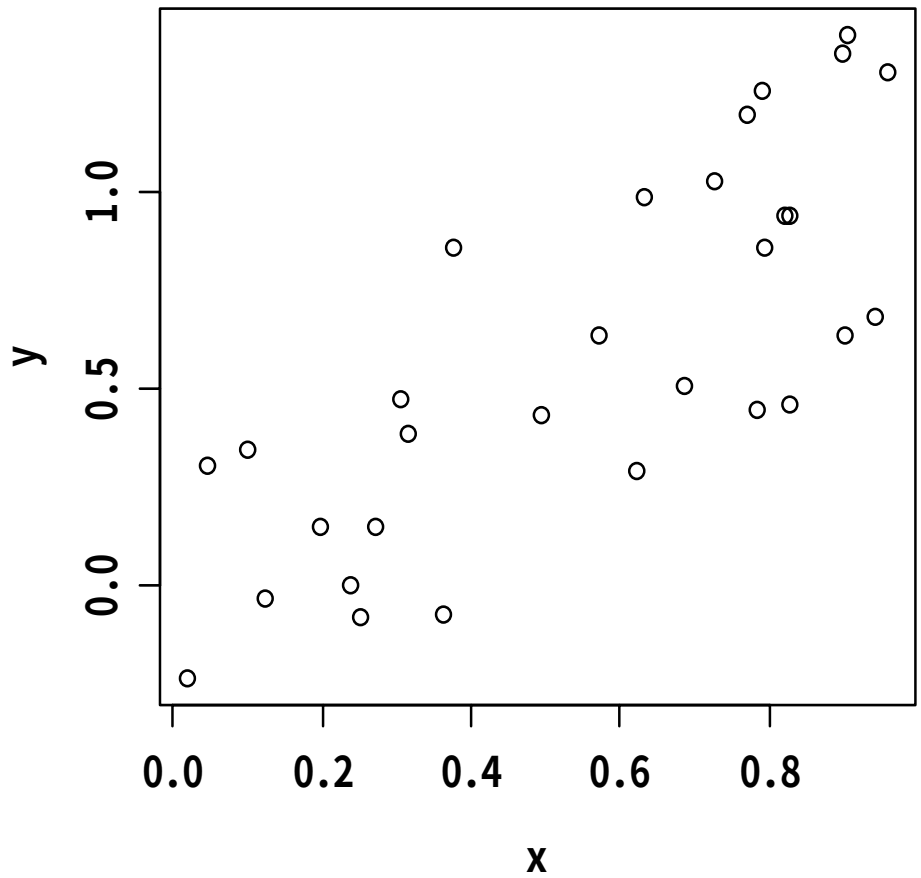
```
> plot(1:10)
> lines(c(0,10), c(6,6))
> rect(1,7, 3,9)
> arrows(1,1, 4,4)
> text(5,8,"文字です")
> title("メインタイトル",
        "サブタイトル")
> legend(6, 4,
        c("sin", "cos", "tan"),
        col=2:4, lty = 1:3,
        bg='gray90')
```



低水準作図関数を使った図形の追記（2）

■ 低水準作図関数を用いて，回帰直線を追記する

```
> x <- runif(30)
> y <- jitter(x, amount=0.5)
> plot(x, y)
```



低水準作図関数を使った図形の追記（2）

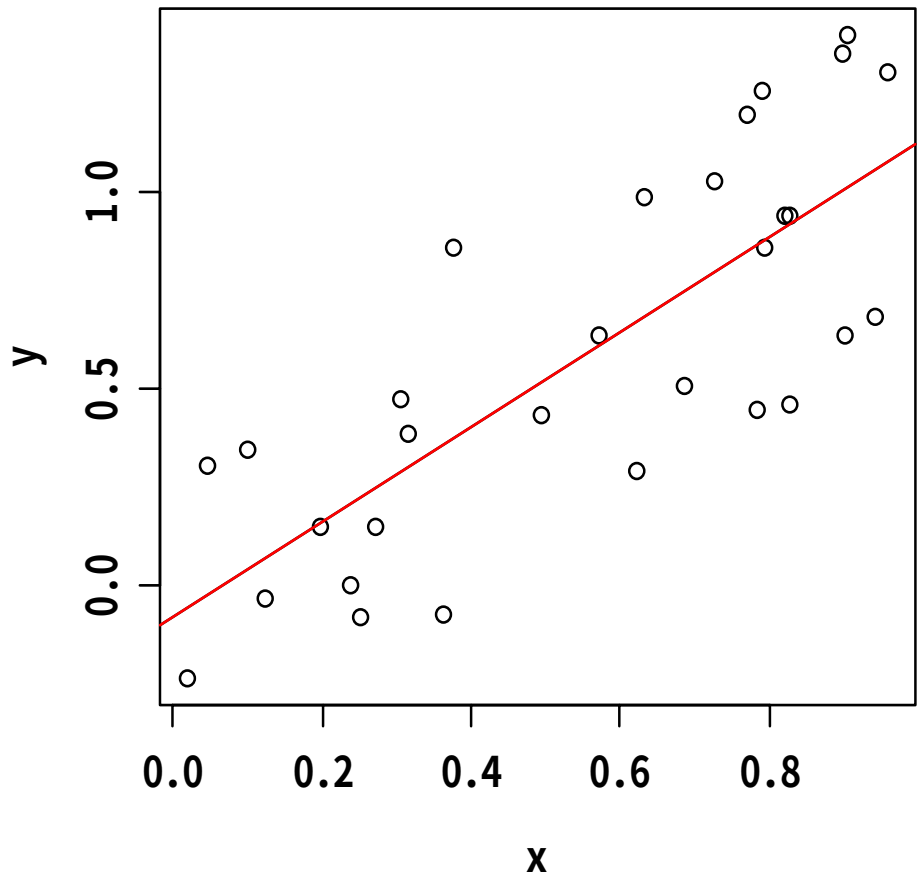
■ 低水準作図関数を用いて，回帰直線を追記する

```
> x <- runif(30)
> y <- jitter(x, amount=0.5)
> plot(x, y)
> result <- lm(y ~ x)
> abline(result, col="red")
```

⇒ 関数 **lm()** で回帰分析を実行

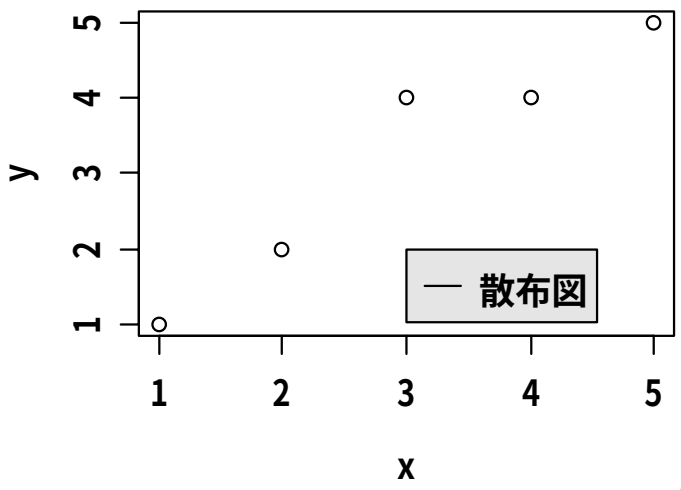
⇒ この後，図を保存しても良い

⇒「 ~ 」は上付きの「～」



【演習】

1. [1, 2, 3, 4, 5] というベクトル x を作成してください
2. [1, 2, 4, 4, 5] というベクトル x を作成してください
3. plot(x, y) として，ベクトル x とベクトル y に関する 2 次元散布図を作成してください
4. 右図のように凡例を追記してください
5. **result <- lm(y ~ x)** を実行して回帰分析結果を変数 result に代入してください
6. 変数 result を使って散布図に回帰直線を追記してください



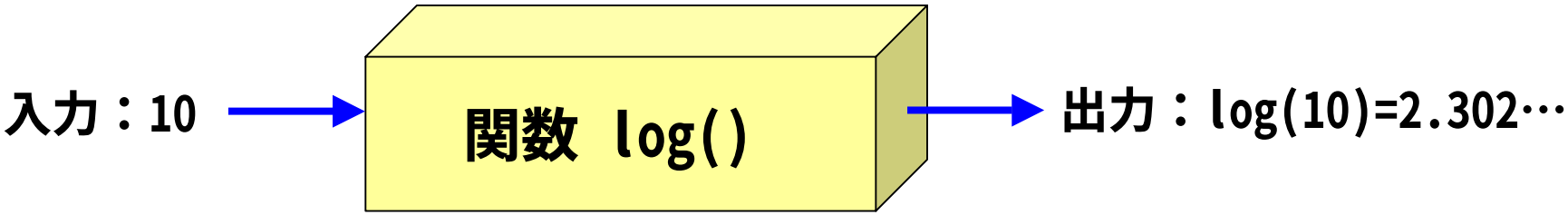
２時間目のメニュー

- グラフの作成方法
  - グラフ作成の第一歩
  - 高水準作図関数
    - 高水準作図関数の種類
    - 関数 plot() を用いた作図例
  - 低水準作図関数
    - 低水準作図関数の種類
    - 低水準作図関数を用いた作図例
  - 数学関数のプロット ←
    - 数学関数の定義方法
    - 数学関数の作図例
  - 参考（重ねた図の描き方，R の画像の編集方法）

関数について

- R では「特別な機能を果たす命令」を「関数」という形で呼び出すことが出来る
- 関数 log(x) は「 x の値の対数を計算する関数」だが，関数に x の値を指定する場合には「引数」という形で関数に与える

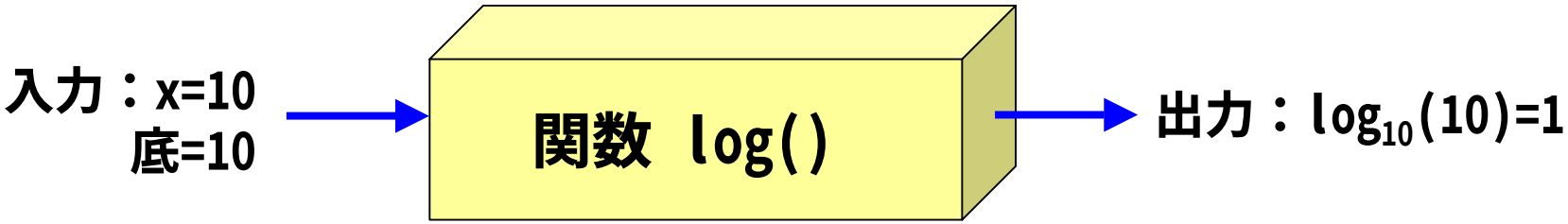
```
> log(10)
[1] 2.302585
```



関数について

- 引数に式や関数を指定すれば，それを評価した値が引数として使われる
- 引数が 2 個以上ある場合はコンマ（ , ）で区切って並べればよい

```
> log(10, base=10)    # 底を指定する引数 base に 10 を指定する
[1] 1
```

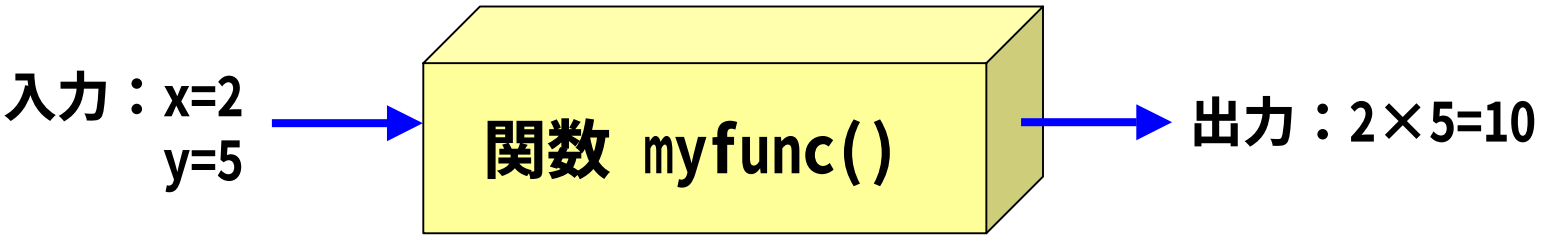


関数を定義する手順

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() で計算結果を出力する

【例】入力した 2 つの数値の積が出力される関数 myprod(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
```



数学関数の定義

【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
```

数学関数のプロット

■ 関数 curve(関数名, x 軸の下限, x 軸の上限) を用いる

【例】関数 $f(x) = x^2$ のプロット

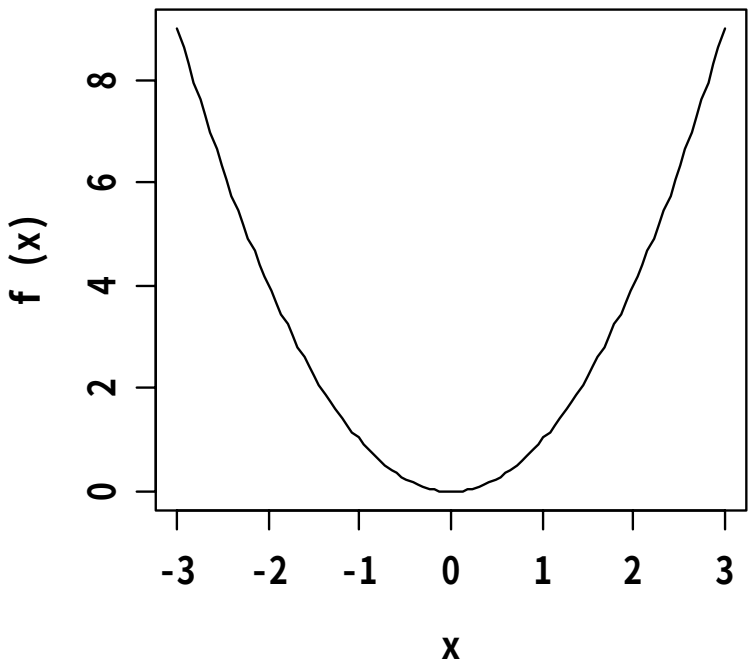
```
> f <- function(x) {
+   return(x^2)
+ }
> curve(f, -3, 3)          # -3 ～ 3 の範囲でプロット
```

【例】関数 $f(x, a) = x^a$ のプロット（a = 3 とする）

```
> f <- function(x, a) {
+   return(x^a)
+ }
> curve(f(x, 3), -3, 3)    # -3 ～ 3 の範囲で a=1 としてプロット
```

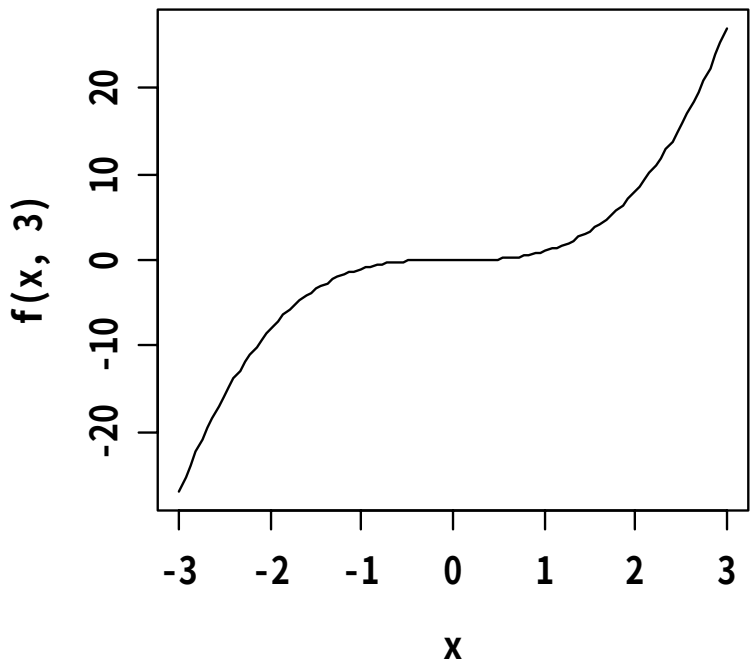
数学関数のプロット

$y = x^2$ のプロット




```
> f <- function(x) {
+   return(x^2)
+ }
> curve(f, -3, 3)
```

$y = x^a$ のプロット（a=3）




```
> f <- function(x, a) {
+   return(x^a)
+ }
> curve(f(x, 3), -3, 3)
```

数学関数のプロット

■ 3 次元のグラフを描くときは関数 persp() を用いる

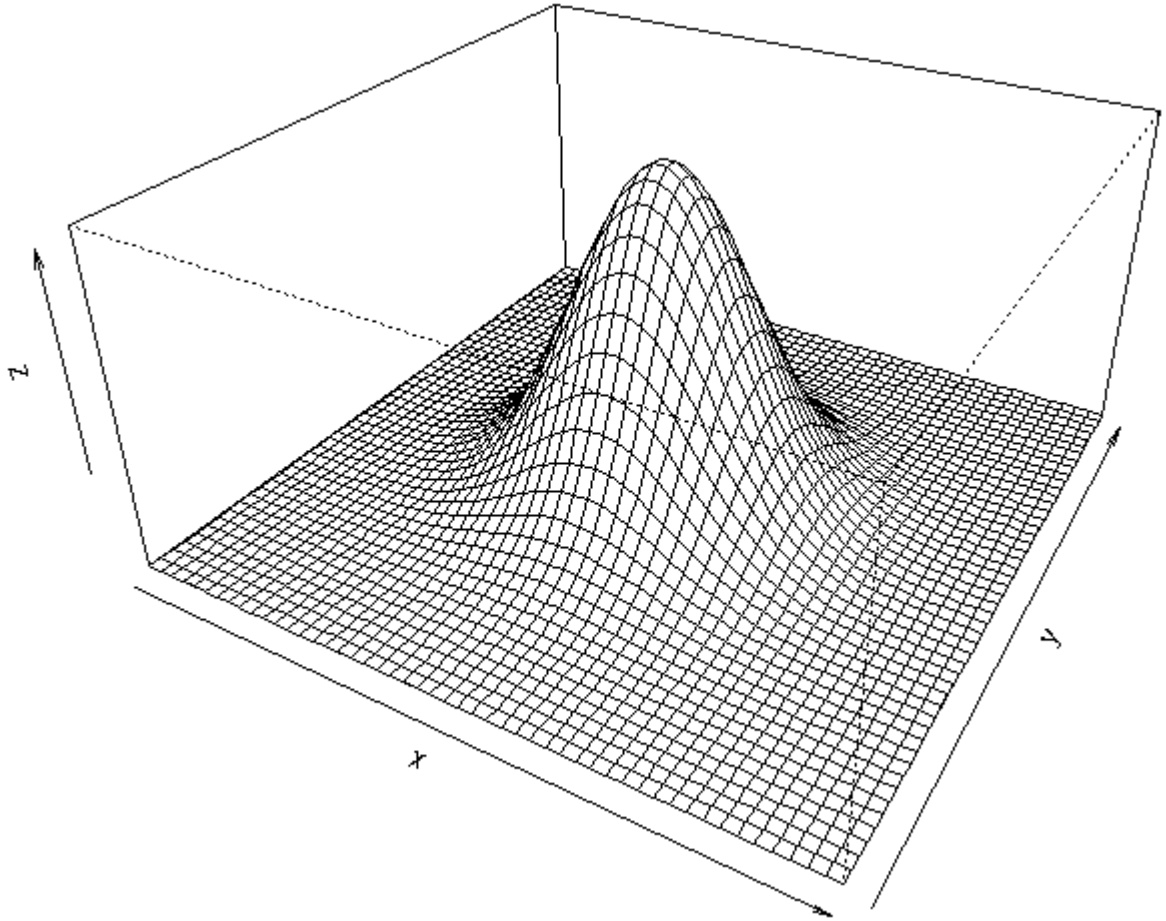
【例】関数 f(x, y) = $\frac{1}{2\pi}\exp\left\{\frac{-(x^2+y^2)}{2}\right\}$ のプロット

1. 関数を定義した後，x 軸と y 軸の点を用意する
2. 関数 outer(x, y, 関数名) で z 軸の点を用意する
3. 関数 persp(x, y, z) でプロットする

```
> f <- function(x, y) return( 1/(2*pi)*exp(-(x^2+y^2)/2) )
> x <- seq(-4, 4, length= 50)
> y <- x
> z <- outer(x, y, f);
> persp(x, y, z, theta = 30, phi = 30, expand = 0.5)
```

数学関数のプロット

関数 f(x, y) = $\frac{1}{2\pi}\exp\left\{\frac{-(x^2+y^2)}{2}\right\}$ のプロット



数学関数のプロット

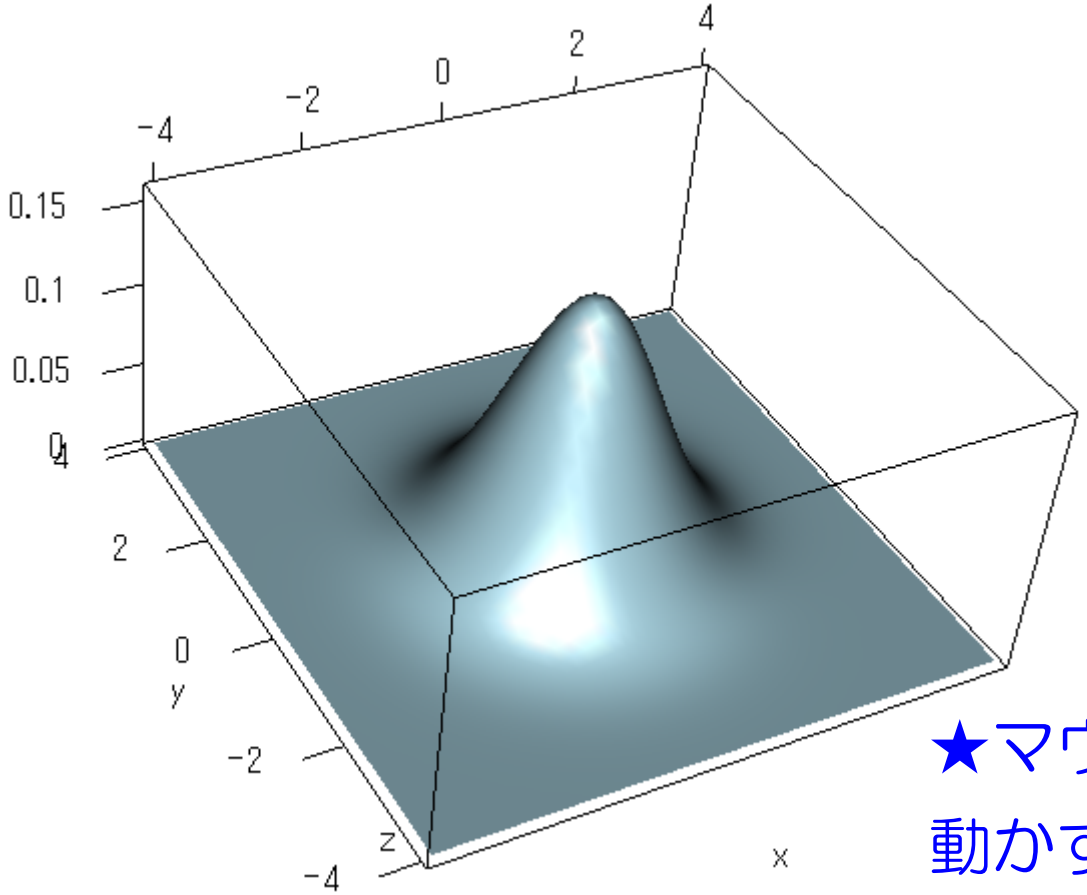
- パッケージ rgl の関数 persp3d() でも OK！

1. 関数を定義した後，x 軸と y 軸の点を用意する
2. 関数 outer(x, y, 関数名) で z 軸の点を用意する
3. 関数 persp3d(x, y, z) でプロットする

```
> f <- function(x, y) return( 1/(2*pi)*exp(-(x^2+y^2)/2) )
> x <- seq(-4, 4, length= 50)
> y <- x
> z <- outer(x, y, f)
> library(rgl)
> open3d()
> persp3d(x, y, z, aspect=c(1,1,0.5), col="lightblue")
```

数学関数のプロット

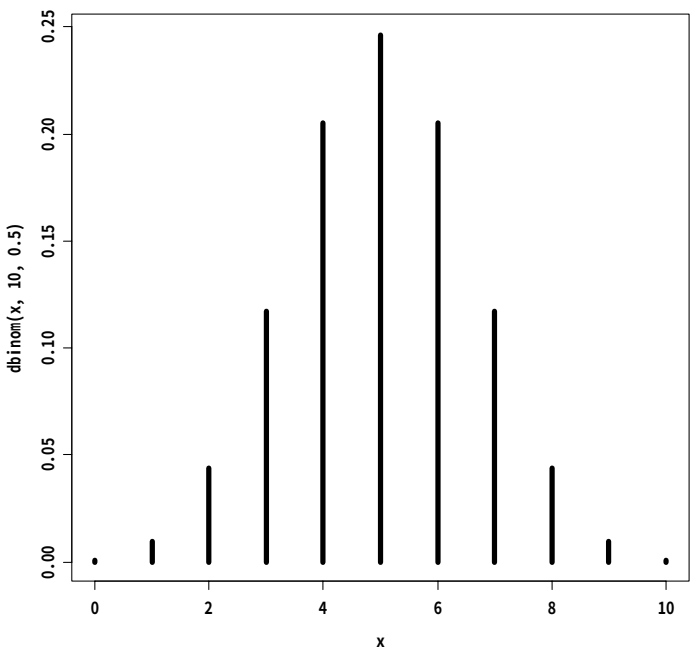
関数 f(x, y) = $\frac{1}{2\pi}\exp\left\{\frac{-(x^2+y^2)}{2}\right\}$ のプロット



★マウスでグリグリ
動かすことが出来る

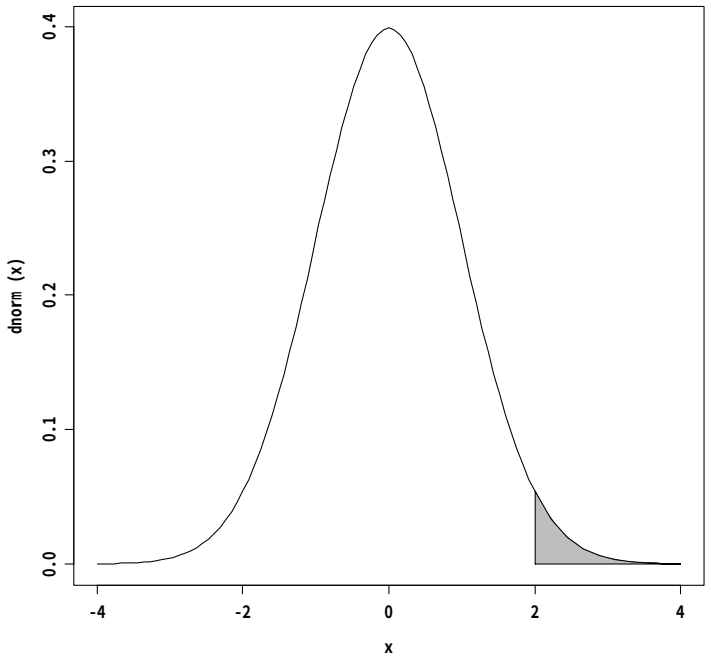
確率分布のプロット

二項分布のプロット




```
> x <- 0:10
> plot(x, dbinom(x,10,0.5),
+      type="h", lwd=5) # プロット
```

正規分布のプロット（影付き）




```
> curve(dnorm, -4, 4, type="l") # プロット
> xvals <- seq(2, 4, length=10) # 影を追加
> dvals <- dnorm(xvals)
> polygon(c(xvals,rev(xvals)),
+      c(rep(0,10),rev(dvals)),col="gray")
```

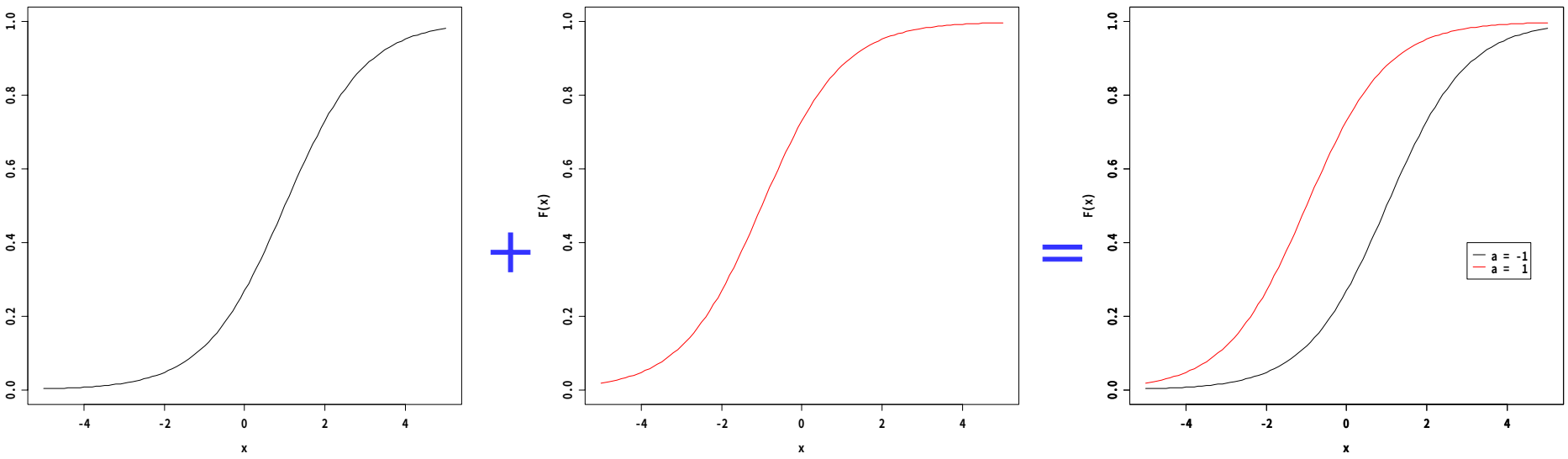
確率分布のプロット

分布名	関数名	分布名	関数名
ベータ分布	dbeta	ロジスティック分布	dlogis
二項分布	dbinom	多項分布	dmultinom
コーシー分布	dcauchy	負の二項分布	dnbinom
カイ二乗分布	dchisq	正規分布	dnorm
指数分布	dexp	ポアソン分布	dpois
F分布	df	Wilcoxon の符号付順位和統計量の分布	dsignrank
ガンマ分布	dgamma	t 分布	dt
幾何分布	dgeom	一様分布	dunif
超幾何分布	dhyper	ワイブル分布	dweibull
対数正規分布	dlnorm	Wilcoxon の順位和統計量の分布	dwilcox
ベータ分布	dbeta	ロジスティック分布	dlogis

２時間目のメニュー

- グラフの作成方法
  - グラフ作成の第一歩
  - 高水準作図関数
    - 高水準作図関数の種類
    - 関数 plot() を用いた作図例
  - 低水準作図関数
    - 低水準作図関数の種類
    - 低水準作図関数を用いた作図例
  - 数学関数のプロット
    - 数学関数の定義方法
    - 数学関数の作図例
  - 参考（重ねた図の描き方，R の画像の編集方法）←

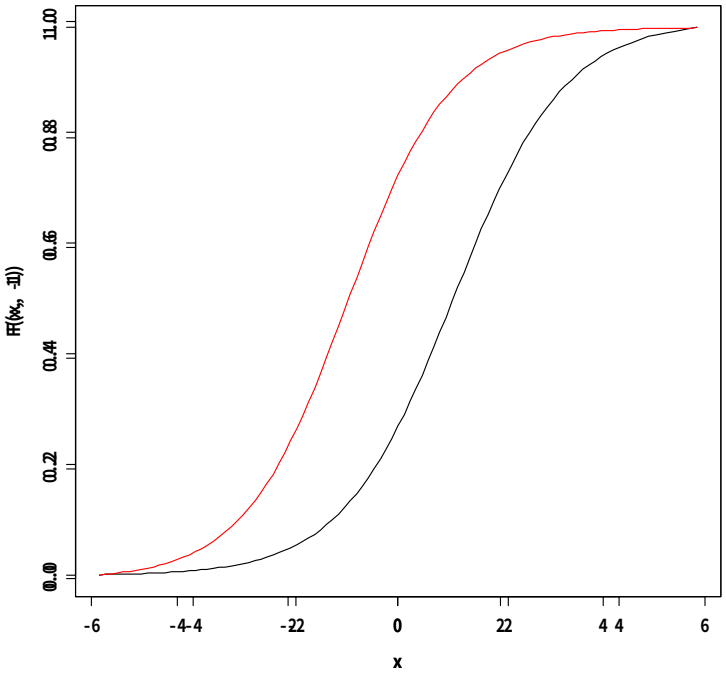
【参考】重ねた図の描き方



■ ２枚の図を１枚に重ねて表示することを考える

```
F <- function(x, a) { 1/(1+exp(-a-x)) }   # 作図する関数
curve(F(x,-1), col=1, xlim=c(-5,5),  ylim=c(-0,1), ylab="")
par(new=T)
curve(F(x, 1), col=2, xlim=c(-5,5),  ylim=c(-0,1), ylab="F(x)")
legend(3, 0.4, c("a = -1","a =  1"), lty=1, col=1:2)
```

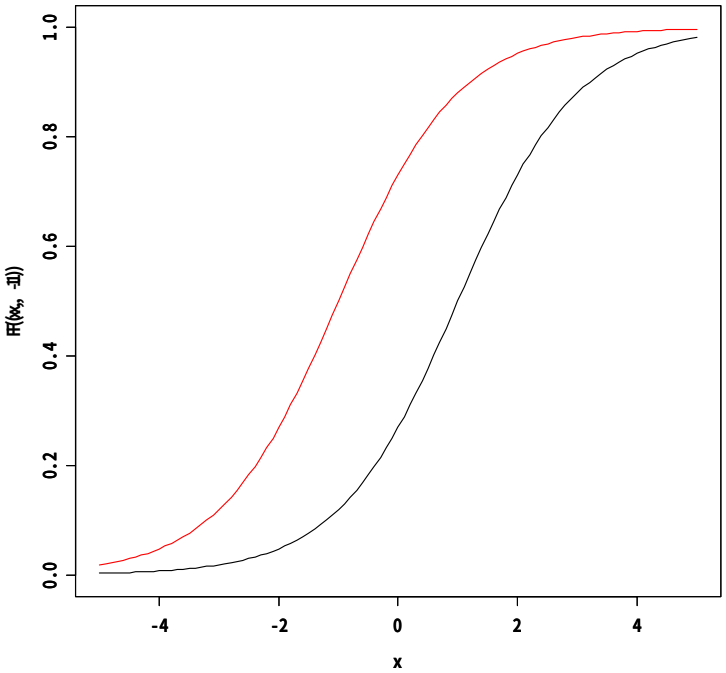
【参考】重ねた図の描き方：失敗例①



- 関数 par()で２枚の図を１枚に重ねて表示出来るが，そのままでは x 軸と y 軸がバラバラ…

```
curve(F(x,-1), col=1)
par(new=T)    # 前の図を残したまま次の図を描く，という命令
curve(F(x, 1), col=2)
```

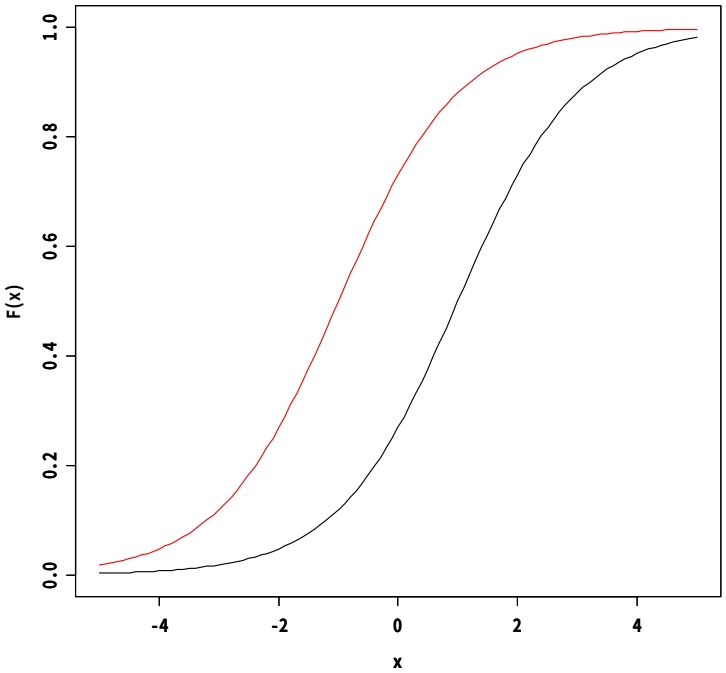
【参考】重ねた図の描き方：失敗例②



- 引数 xlim と ylim で２枚の図の座標を合わせればよいが，今度は y 軸ラベルがおかしい…

```
curve(F(x,-1), col=1, xlim=c(-5,5), ylim=c(-0,1))
par(new=T)
curve(F(x, 1), col=2, xlim=c(-5,5), ylim=c(-0,1))
```

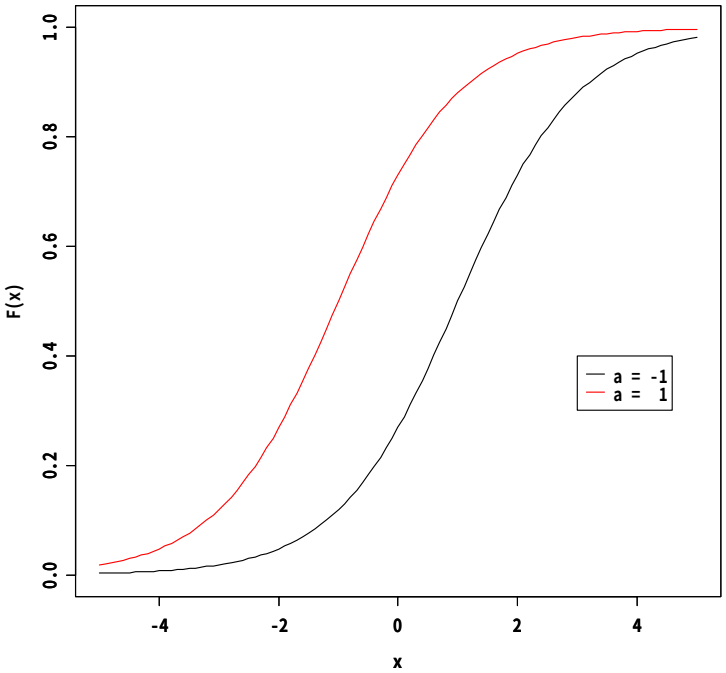
【参考】重ねた図の描き方：成功例①



- １つめの引数 ylab を " "（空白）にして，
  ２つめの引数 ylab で y 軸ラベルを指定する

```
curve(F(x,-1), col=1, xlim=c(-5,5),  ylim=c(-0,1), ylab="")
par(new=T)
curve(F(x, 1), col=2, xlim=c(-5,5),  ylim=c(-0,1), ylab="F(x)")
```

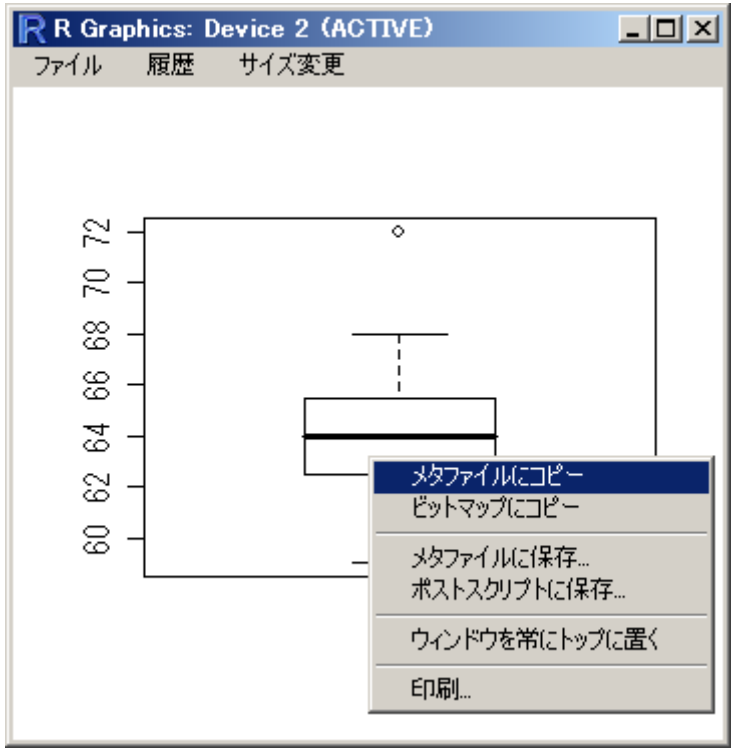
【参考】重ねた図の描き方：成功例②



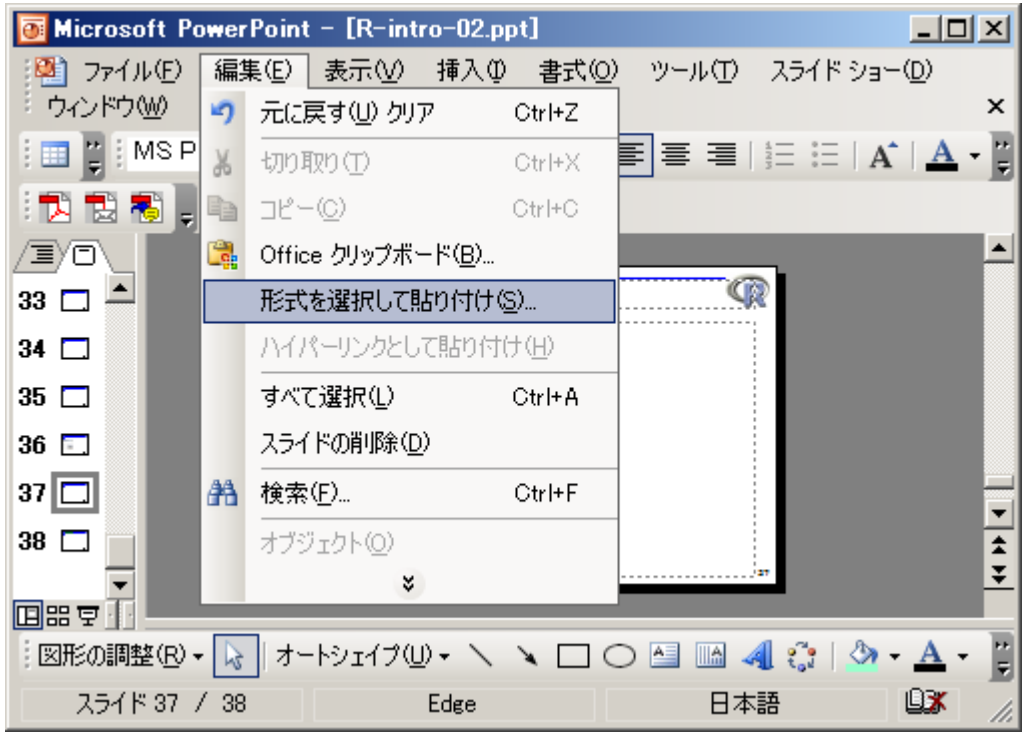
■ 凡例をつけて完成！

```
curve(F(x,-1), col=1, xlim=c(-5,5),  ylim=c(-0,1), ylab="")
par(new=T)
curve(F(x, 1), col=2, xlim=c(-5,5),  ylim=c(-0,1), ylab="F(x)")
legend(3, 0.4, c("a = -1","a =  1"), lty=1, col=1:2)
```

【参考】R の画像をパワーポイントに貼る

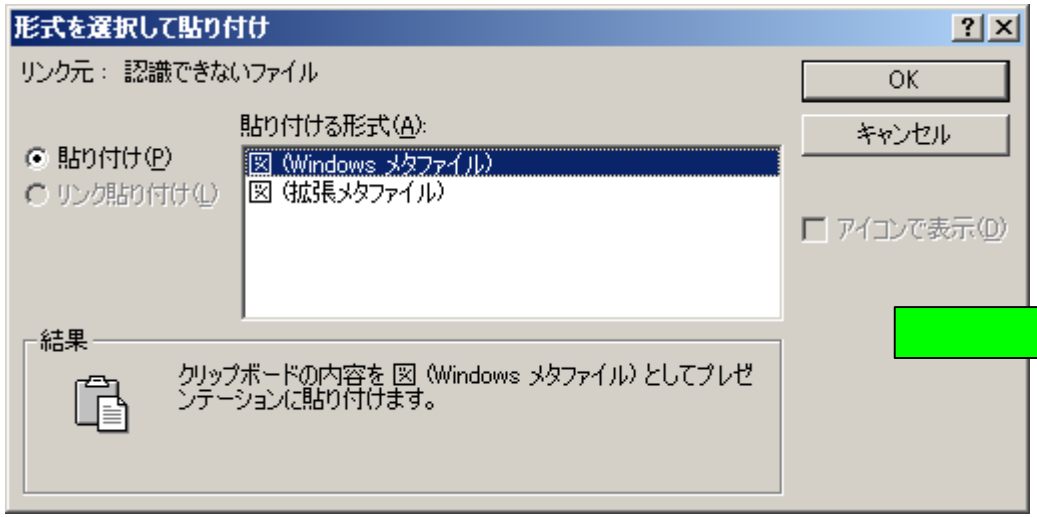


デバイスウインドウを右クリックした後「メタファイルにコピー」を選択

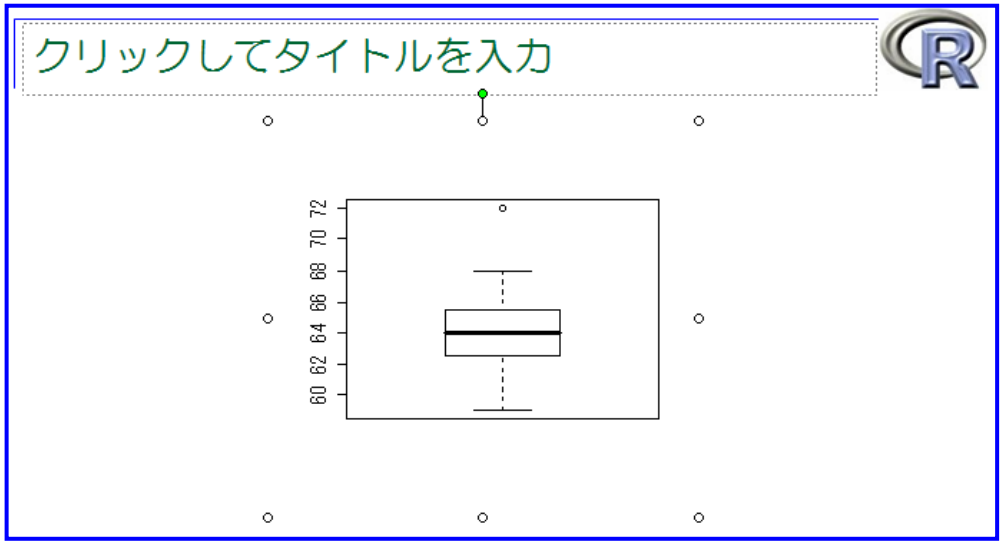


パワーポイント上で［編集］→［形式を選択して貼り付け］

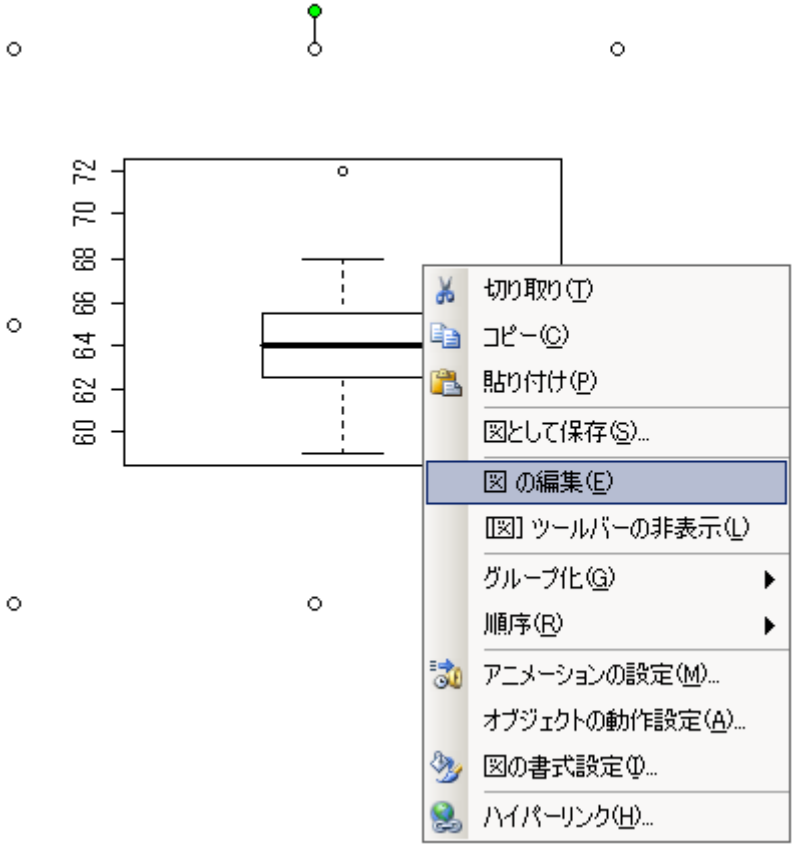
【参考】R の画像をパワーポイントに貼る



[OK] をクリックすると
めでたく図が貼られる

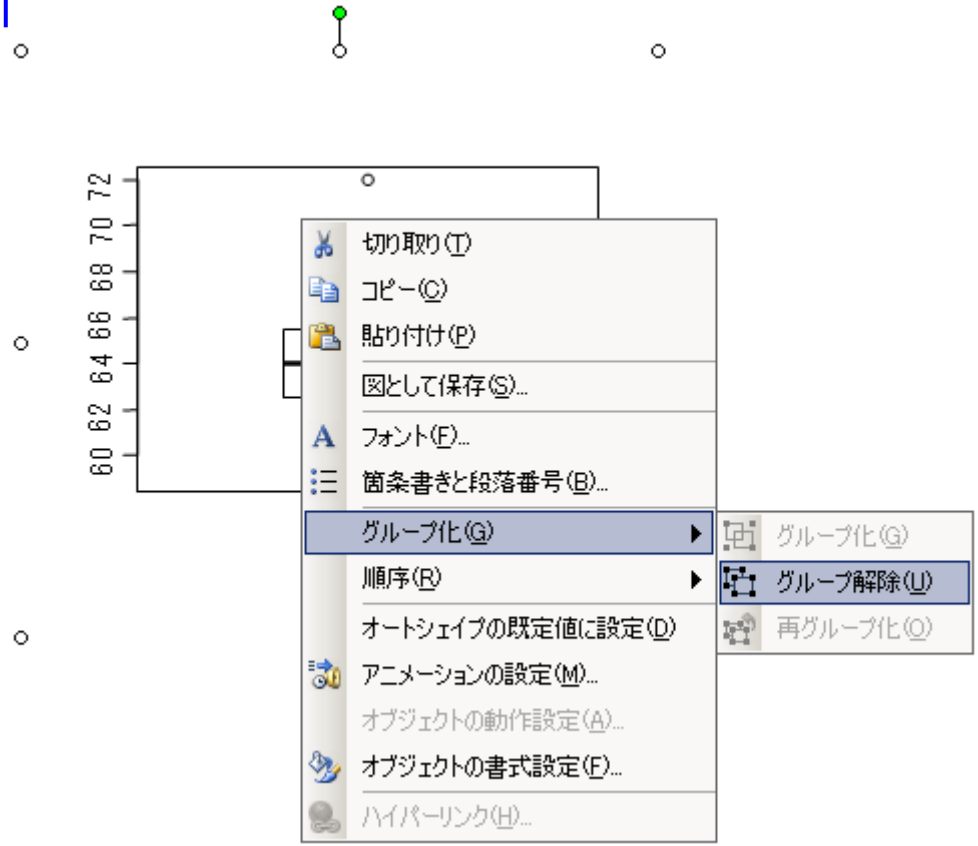


【参考】R の画像をパワーポイントで編集する

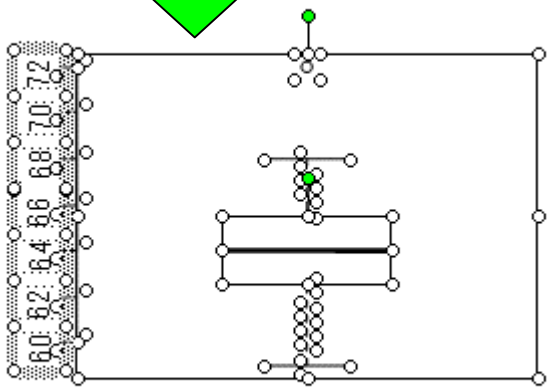


画像を右クリックした後
「図の編集」を選択
（その後 [はい] を選択）

【参考】R の画像をパワーポイントで編集する



再度，画像を右クリックした後
「グループ化」「グループ解除」
を選択（必要ない場合もあり）



図が編集できるようになる →

【参考】関数を用いて図を保存する

- 図を描いた後，関数 dev.copy() で図を保存
  （関数は 6 枚目のスライドで紹介した関数が使用可）

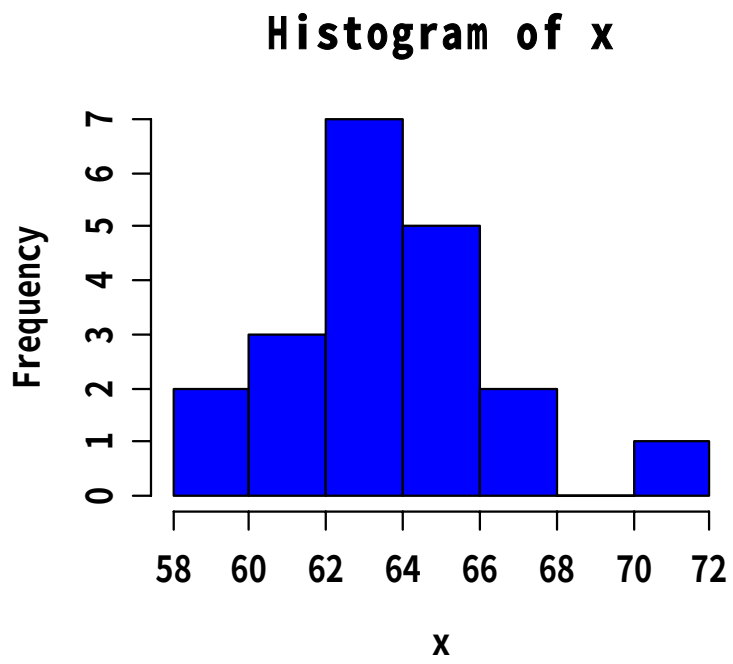
```
> plot(1:10)                          # 図を描いた後
> dev.copy(pdf, "Scatter.pdf")        # pdf に書き出し
> dev.off()                           # ファイルを閉じる
```

- 図を eps ファイルに保存する場合は関数 dev.copy2eps() を使用する

```
> plot(1:10)                          # 図を描いた後
> dev.copy2eps(file="finename.eps")   # pdf に書き出し
> dev.off()                           # ファイルを閉じる
```

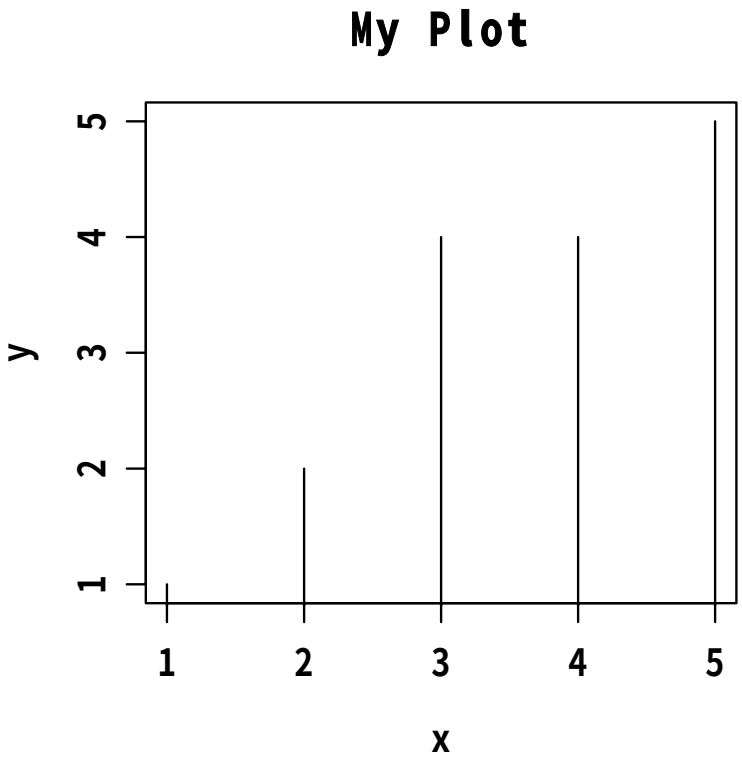
★演習（12枚目のスライド分）の回答例

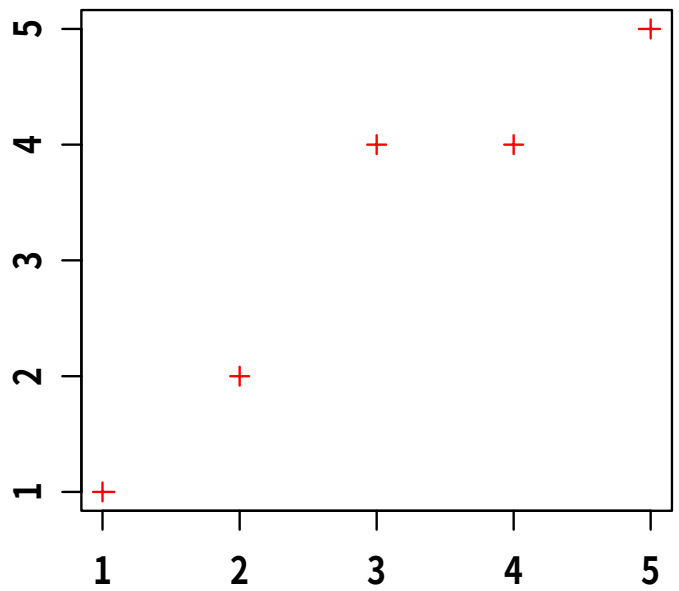
```
> x <- c(59, 60, 61, 62, 62, 63, 63, 63, 63, 64,
+        64, 64, 65, 65, 65, 66, 66, 67, 68, 72)
> hist(x)                    # 設問 3 の回答例
> hist(x, col="blue")        # 設問 5 の回答例
```



★演習（31枚目のスライド分）の回答例

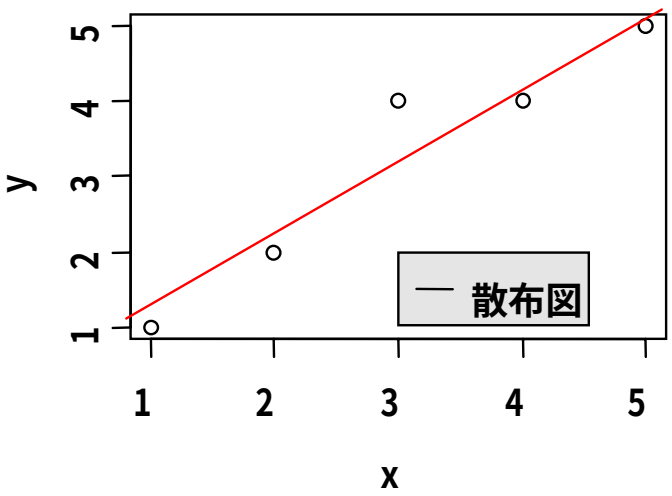
```
> x <- c(1, 2, 3, 4, 5)                    # 設問 1 の回答例
> y <- c(1, 2, 4, 4, 5)                    # 設問 2 の回答例
> plot(x, y, main="My Plot", type="h")     # 設問 3 の回答例
> plot(x, y, col=2, pch=3, ann=F)          # 設問 4 の回答例
```





★演習（44枚目のスライド分）の回答例

```
> x <- c(1, 2, 3, 4, 5)        # 設問 1 の回答例
> y <- c(1, 2, 4, 4, 5)        # 設問 2 の回答例
> plot(x, y)                   # 設問 3 の回答例
                               # 設問 4 の回答例↓
> legend(3, 2, col=1, lty = 1, "散布図", bg='gray90')
> result <- lm(y ~ x)          # 設問 5 の回答例
> abline(result, col="red")    # 設問 6 の回答例
```



２時間目にやったこと

- グラフの作成方法
  - グラフ作成の第一歩
  - 高水準作図関数
    - 高水準作図関数の種類
    - 関数 plot() を用いた作図例
  - 低水準作図関数
    - 低水準作図関数の種類
    - 低水準作図関数を用いた作図例
  - 数学関数のプロット
    - 数学関数の定義方法
    - 数学関数の作図例
  - 参考（重ねた図の描き方，R の画像の編集方法）

終