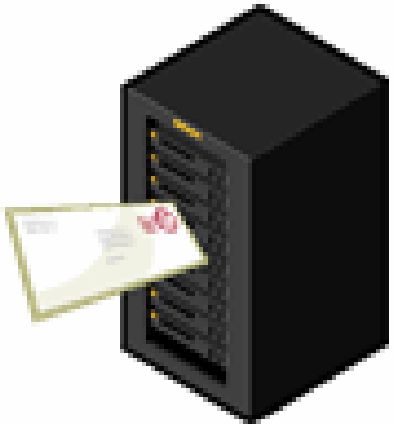
Rで学ぶプログラミングの基礎の基礎

(5) 関数の作成

プログラミングについて

- プログラミングとは
  - 人間がコンピュータに命令をすること
  - R の場合は「ユーザーが R のコマンドをひとつひとつ記述する」作業のこと ⇒ 関数定義！
- R におけるプログラミングのための道具は・・・
  - 変数，ベクトル，関数定義
  - 条件分岐（if）
  - くり返し（for）



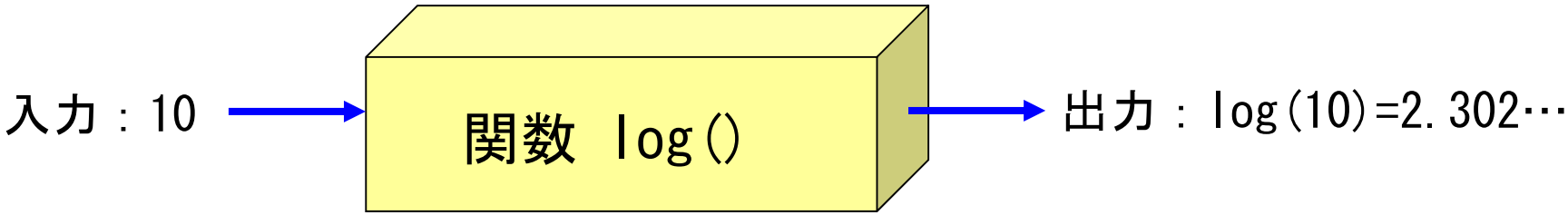
本日のメニュー

1. 関数について
2. 関数の作成
3. 数学関数の作成とプロット

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

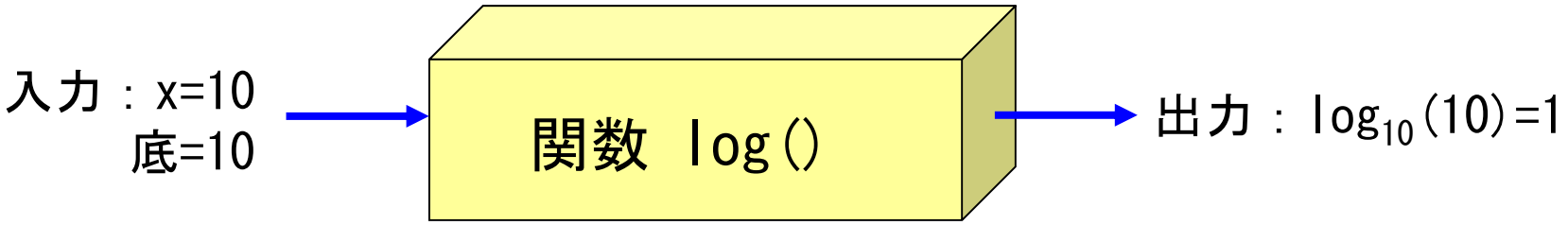
```
> log(10)
[1] 2.302585
```



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
```



本日のメニュー

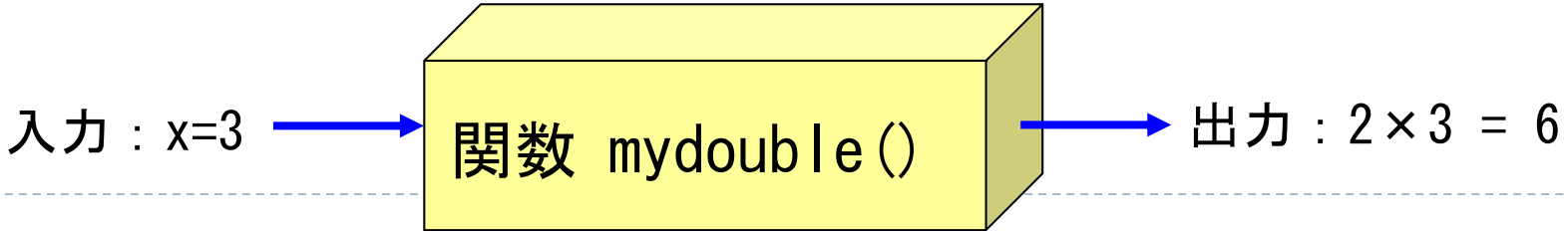
1. 関数について
2. 関数の作成
3. 数学関数の作成とプロット

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】入力した値を 2 倍したものが出力される関数 mydouble(x)

```
> mydouble <- function(x) {          #
+   y <- 2*x                         #
+   return(y)                        # 関数定義部分
+ }                                  #
> mydouble(3)                        # 関数を実行する
[1] 6
```



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】引数が無い関数 myzero()

```
> myzero <- function() {
+   x <- 1
+   return(x)
+ }
> myzero()
[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)

```
> myprod <- function(x, y) {
+   return(x*y)
+ }
> myprod(2,5)
 [1] 10
```

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】返り値が 2 つある関数 my2return(x,y)

→ 返したい結果をベクトルにすればよい

```
> my2return <- function(a, b) {
+   x <- a
+   y <- b
+   z <- c(y, x)
+   return(z)
+ }
> my2return(1,2)
[1] 2 1
```

演習問題

1. 引数が 1 つある関数を定義する練習として，入力した数値を 2 乗したものを結果として返す関数 myfunc1() を定義して下さい
2. 引数が 2 つある関数を定義する練習として，「引数 1」を「引数 2」乗したもの（例えば 4^3 ）を結果として返す関数 myfunc2() を定義して下さい
3. 返り値が 2 つある関数を定義する練習として，引数 1 を引数 2 で割り算したときの商と余りを結果として返す関数 myfunc3() を定義して下さい

演習問題

4. 関数 ceiling(値) は，値の小数部分を切り上げた結果を返す関数で，例えば ceiling(3.45) の結果は 4 となります
   また，関数 runif(1) は実行するたびに返ってくる値が変わるもの（乱数）で，0 ～ 1 の間の実数を等確率で表示します

```
> runif(1)
[1] 0.5255034
> runif(1)
[1] 0.1045277
```

   次の命令を実行した場合，取りうる値の範囲（最小値と最大値）はいくらになるでしょうか

```
> ceiling(6*runif(1))
```

5. 4. を参考にして，実行すると「1，2，3，4，5，6」のどれかを等確率で返す関数 dice() を定義して下さい

本日のメニュー

1. 関数について
2. 関数の作成
3. 数学関数の作成とプロット

数学関数の定義

- 関数 f(x) = 2x

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

- 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

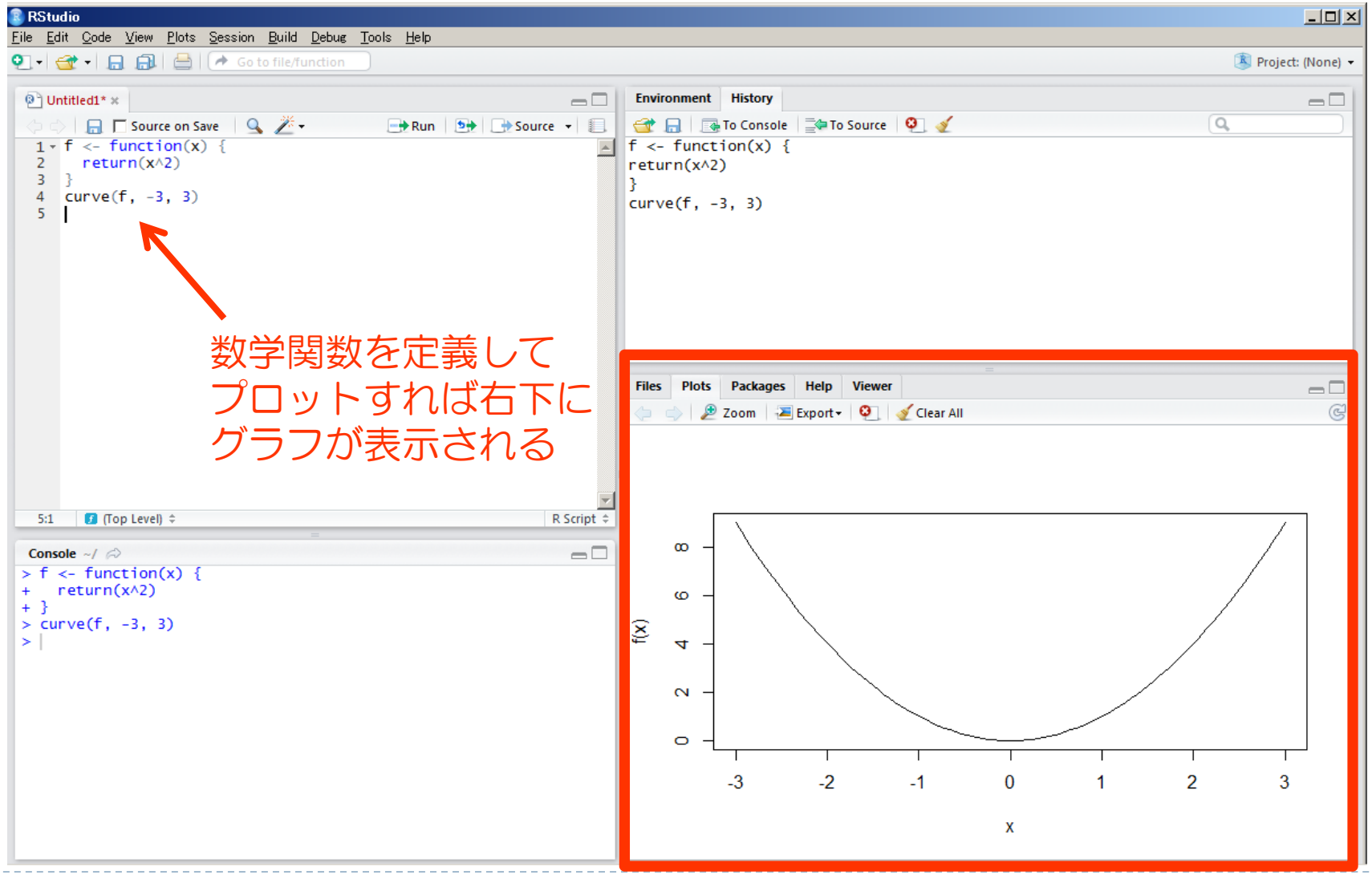
- 関数 curve(関数名, x 軸の下限, x 軸の上限) を用いる
- 関数 f(x) = x2 のプロットを行う場合：

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

- 関数 $f(x, a) = x^a$ のプロットを行う場合（a = 3 とする）：

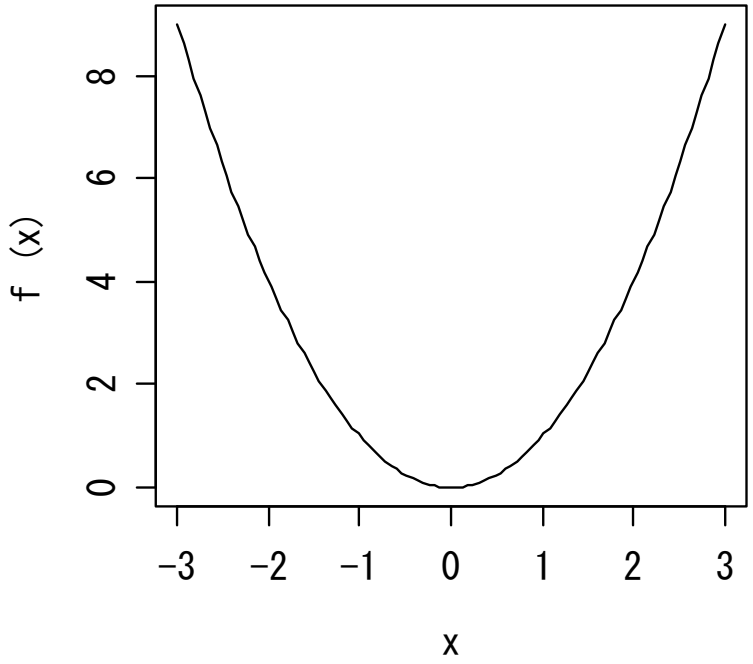
```
> f <- function(x, a) {
+   return(x^a)
+ }
> curve(f(x, 3), -3, 3)    # -3 ～ 3 の範囲で a=1 としてプロット
```

数学関数のプロット


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Untitled1*
Source on Save
Run
Source
f <- function(x) {
return(x^2)
}
curve(f, -3, 3)
数学関数を定義して
プロットすれば右下に
グラフが表示される
Environment History
To Console
To Source
Console
> f <- function(x) {
+ return(x^2)
+ }
> curve(f, -3, 3)
Files Plots Packages Help Viewer
Zoom
Export
Clear All
f(x)
x

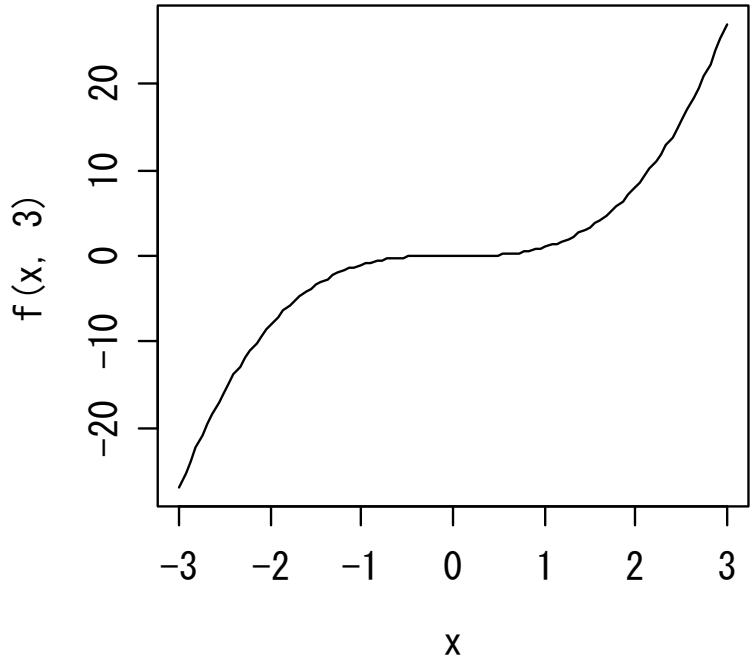
数学関数のプロット

$y = x^2$ 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

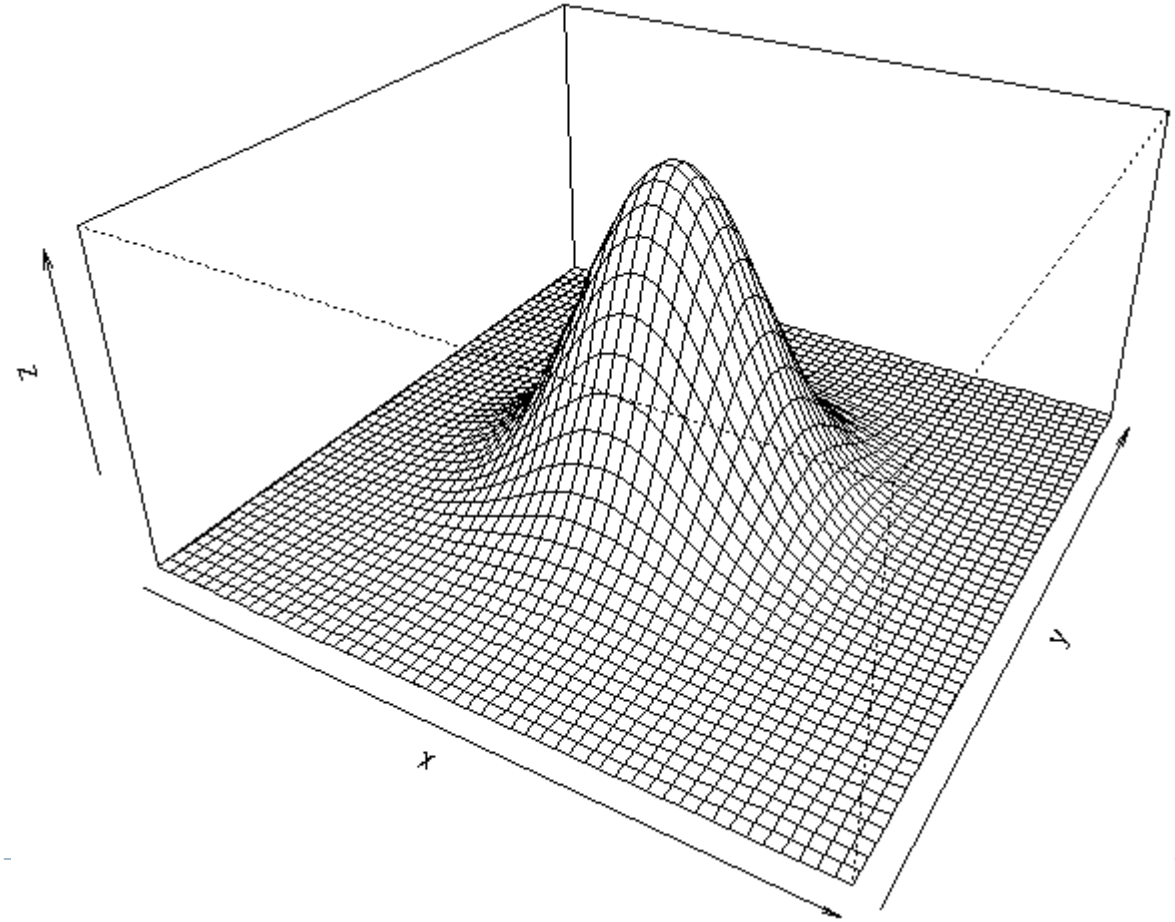
- 3 次元のグラフを描くときは関数 persp() を用いる
- 関数 f(x, y) = $\frac{1}{2\pi}\exp\left\{\frac{-(x^2+y^2)}{2}\right\}$ のプロット

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



本日のメニュー

1. 関数について
2. 関数の作成
3. 数学関数の作成とプロット

このスライドの内容が書籍になっています

- 舟尾 暢男 『 R で学ぶプログラミングの基礎の基礎（カットシステム）』

「R と RStudio のインストール」「R の基礎」「変数とベクトル」
「種々のベクトル」「関数の作成」「条件分岐」「くり返し」
「複雑なくり返し」「グラフ作成の概要」「問題集」「簡単なゲーム作成」

※ 本資料の演習問題の解答も載っています

Rで学ぶプログラミングの基礎の基礎

終