2015年 6月1日(月) 16:30-17:30

統計用言語Rの使い方 ver. 2015

基礎工学研究科 M2　奥野彰文

https://www.library.osaka-u.ac.jp/doc/TA_2014_01.pdf

(ver. 2014 のスライドがアップロードされている)

予定時間　約60分

今日の話のメインターゲット

- Rを(入れてみたが)**使い方が分からない**人
- そもそも**使ったことが無い**人
- Rの**存在を知らなかった**人

今日の目標

R/Rstudioを**インストール**して
簡単な計算を**実行**する.
(Rで何ができるか)把握する.

目次

余った時間:
R-番外編

R言語とは?

[本節の内容]

- ✓ Rって何
- ✓ Rを使って何ができるのか
- ✓ Rstudioの紹介
- ✓ ファイルを読み込む

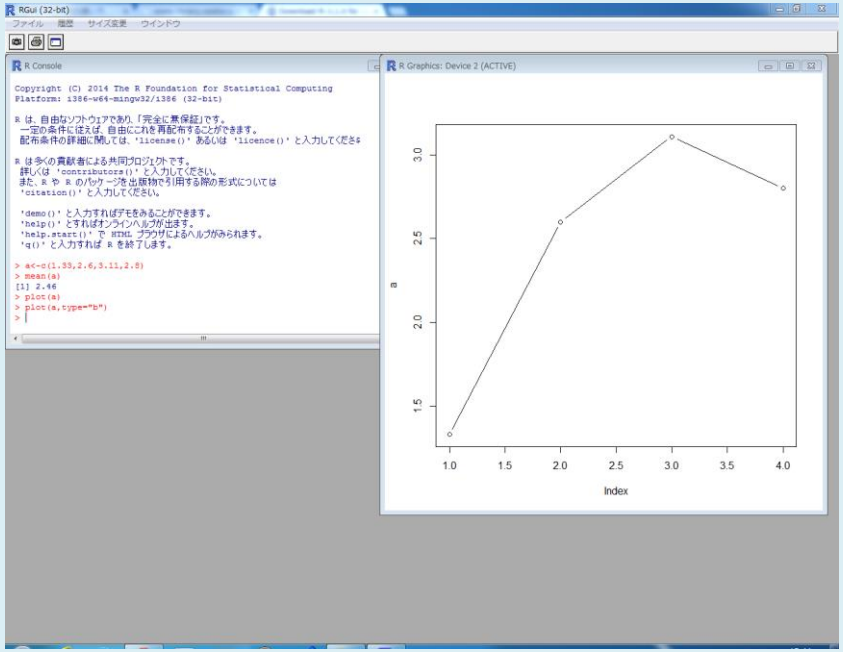
R言語って何ですか

- 統計解析用のプログラム言語

タ　ダ
無料

何ができるのか

- だいたいなんでもできる
  - 通常の四則演算+行列やベクトルの演算
  - 様々な統計分析(回帰や分散分析etc)
  - データのグラフ表示
  - 最先端の手法が使える



R実行画面

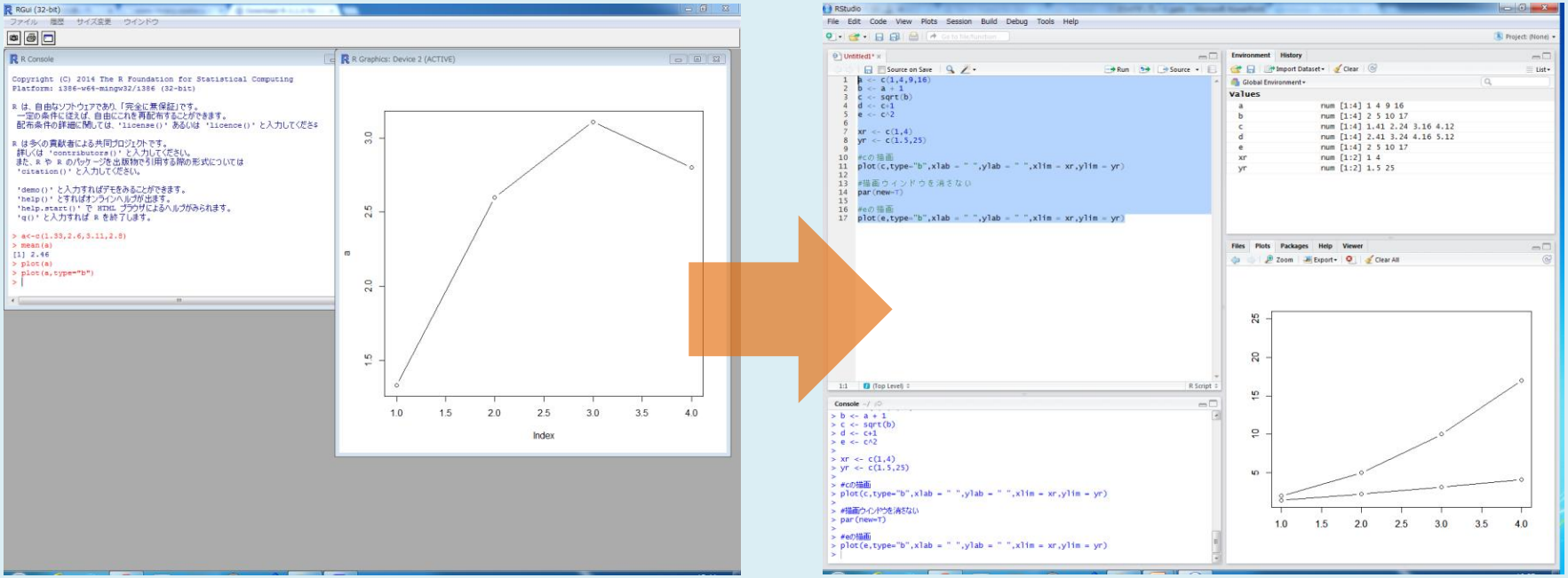
超優秀なツール RStudio

- Rのための統合開発環境(便利なツール)
- もちろん

無料（タダ）

- Rを見やすくしたようなもの.
- データを分析している感が出て恰好良い
- 導入も簡単

RStudio



RからRstudioへ

RStudioをインストールする前に,
先にRをインストールする必要があります!

インストールから実行まで

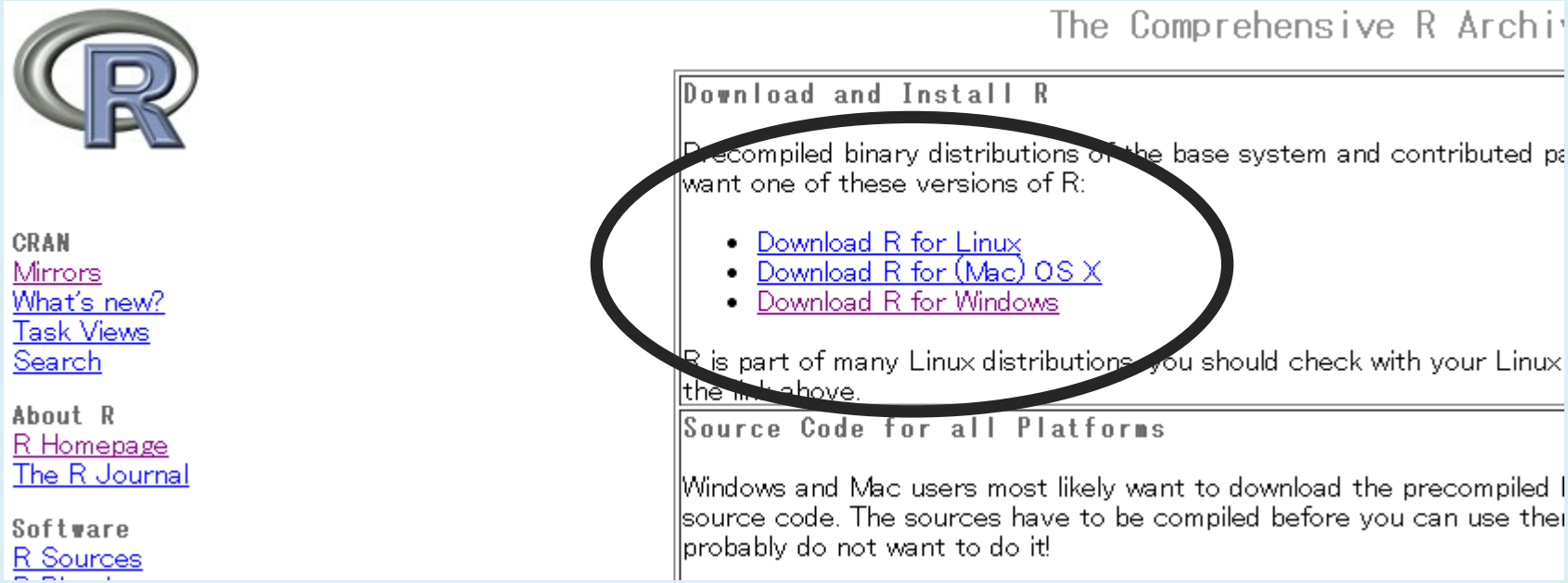
[本節の内容]

- ✓ Rをインストール
- ✓ Rstudioをインストール
- ✓ Rstudioを実行する

Rをダウンロード(1)

- CRANからプログラムをダウンロード

  http://cran.r-project.org/index.html



「Download R for (Windows)」→「base」

Rをダウンロード(2)

クリック！

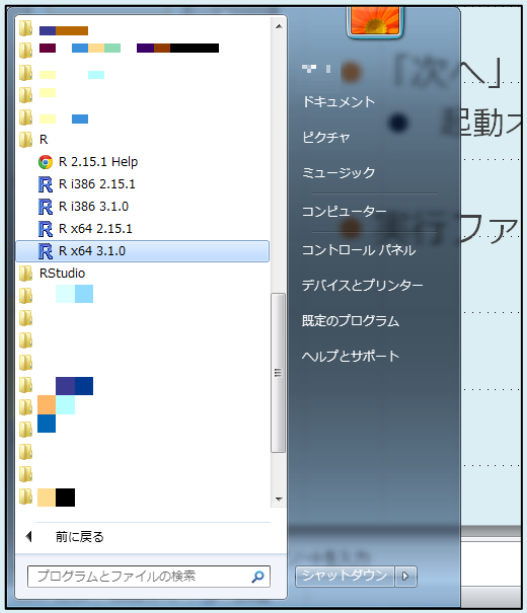
適当なフォルダに保存.
(英語版ですが, **インストール時に**日本語が選択できます.)

Rをインストール(3)

- ダウンロードしたファイルを実行.

  R-***-win.exe

- 「次へ」で進む.
  - 起動オプションなど設定する必要は無い
- 実行ファイルがインストールできました.

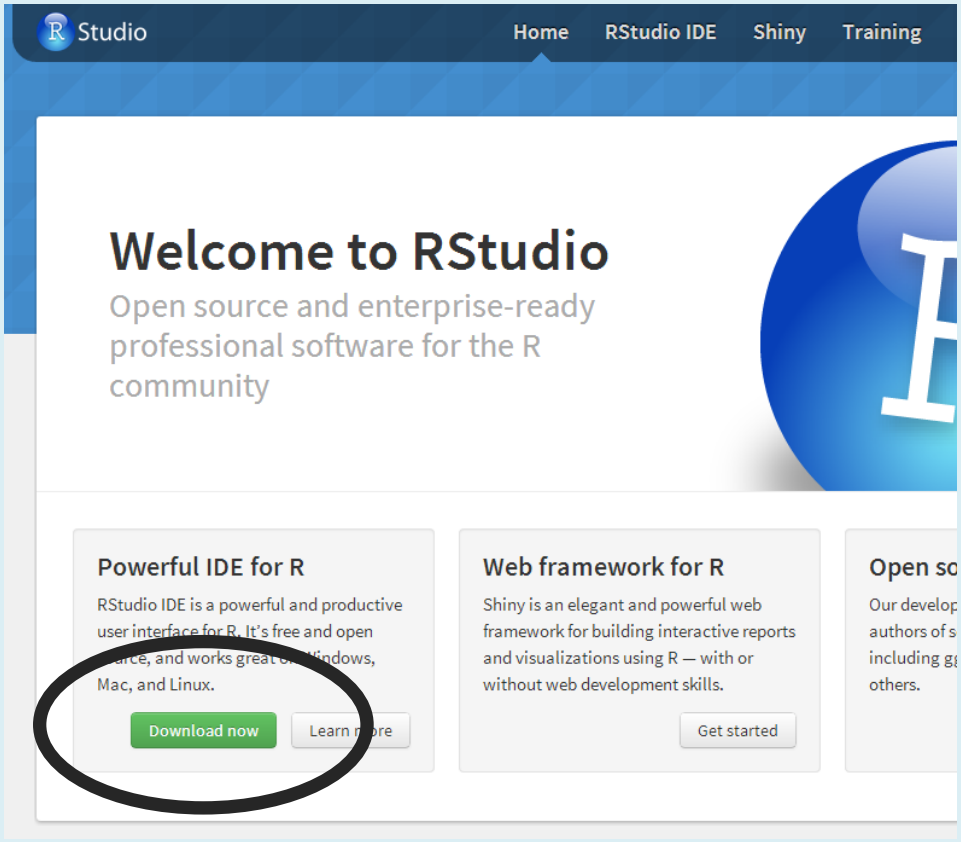


これでRが使用可能に.

Rstudioをダウンロード(1)

- 公式サイトからプログラムをダウンロードする.

https://www.rstudio.com/

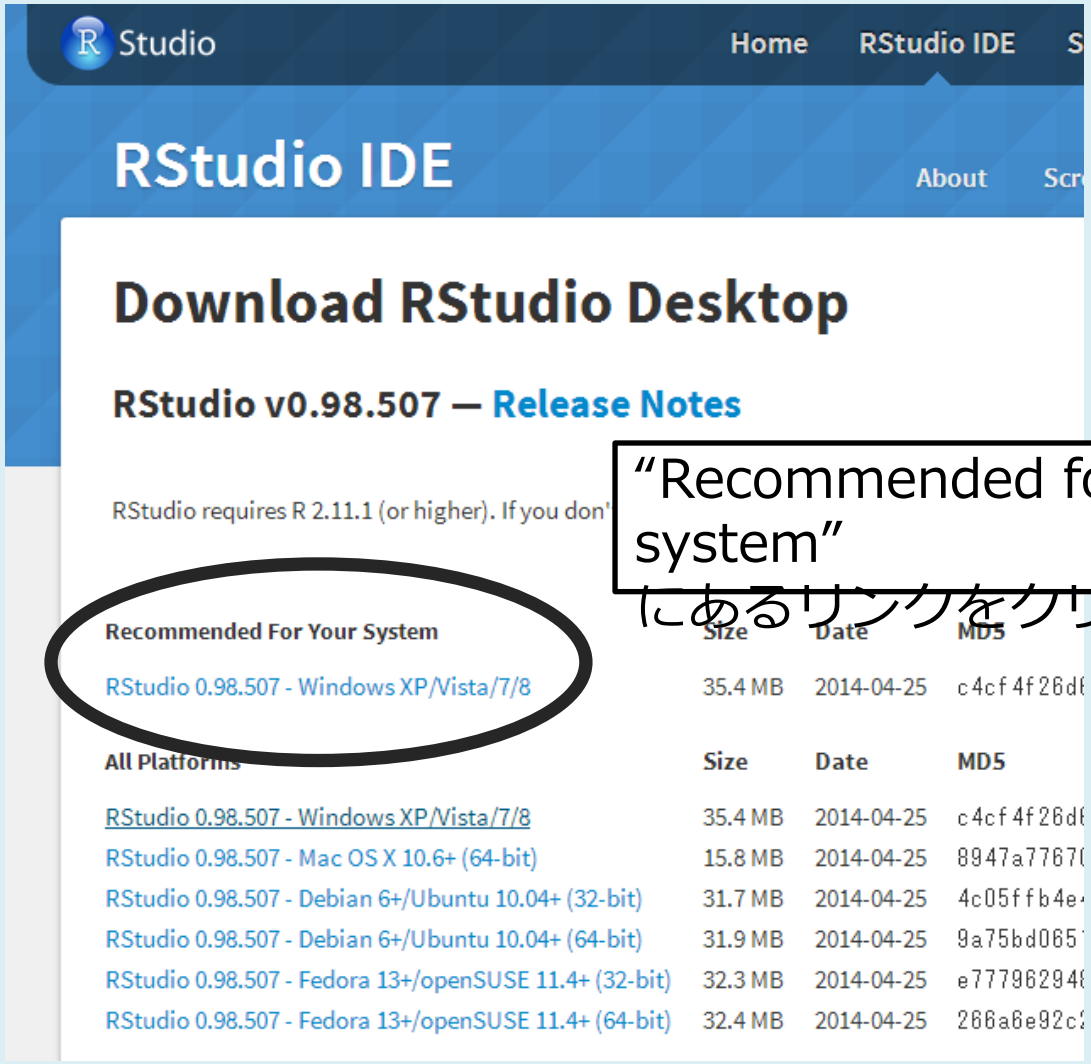


Rstudioをダウンロード(2)

on your desktop を選択
(ノートPCの方もこちら)

こっちはサーバー用

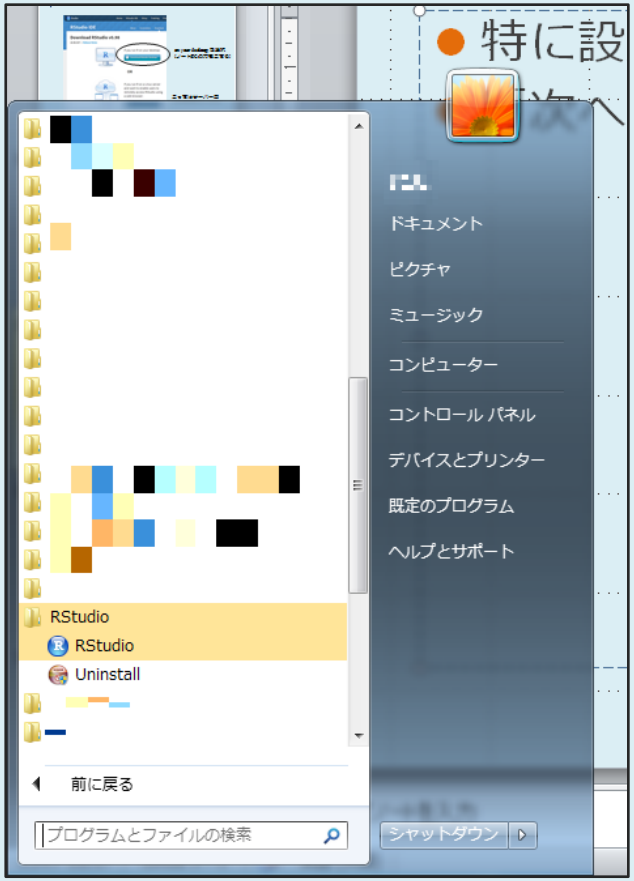
Rstudioをダウンロード(3)



“Recommended for your system”
にあるリンクをクリック

Rstudioをインストール(4)

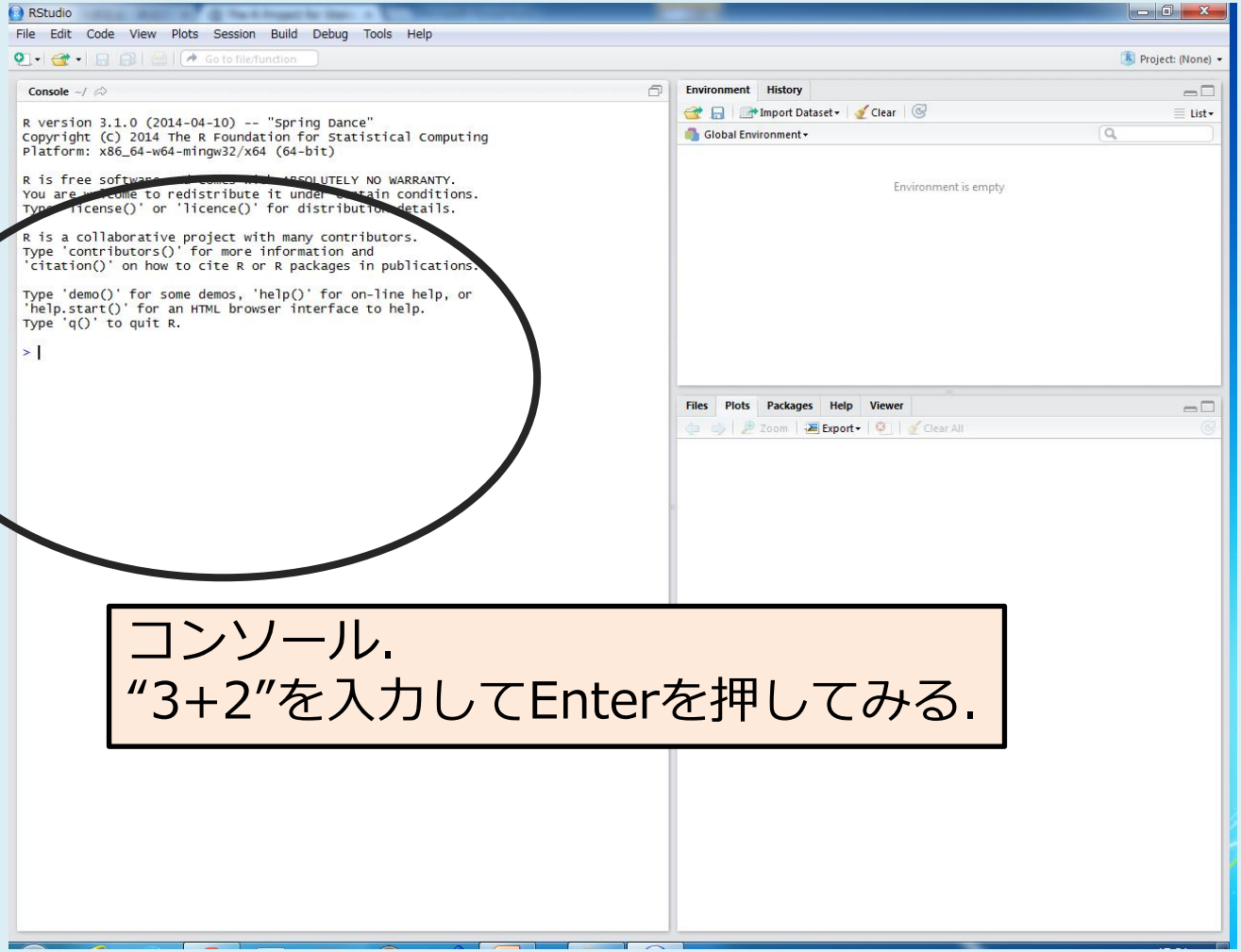
- 特に設定は必要ない.
- 「次へ」で進めばよい.



RStudioが使用可能に.

Rstudioを実行する(コンソール)

- 起動(初期画面).



コンソール.
“3+2”を入力してEnterを押してみる.

```
> 3+2
[1] 5
>
```

単純な計算機として機能している.

```
> 3*5+6/2-(2+4)/3
[1] 16
>
```

複雑な計算も可能

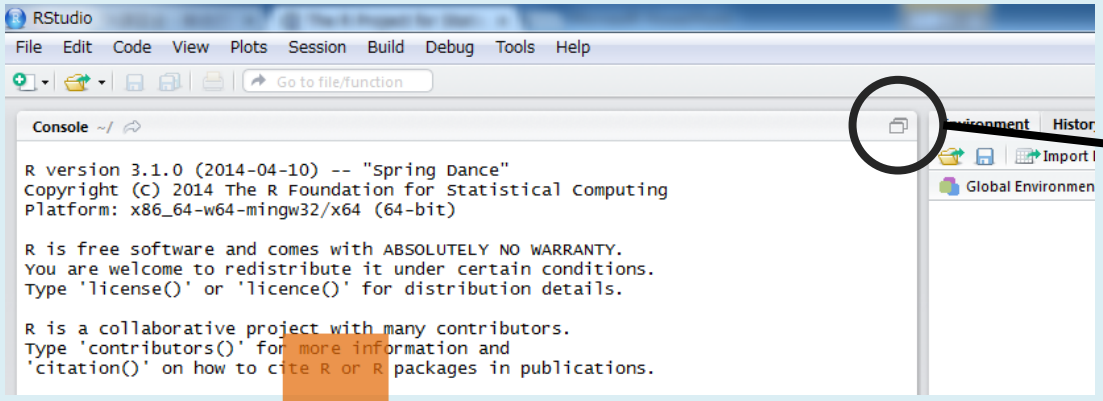
```
> a <- 3
> b <- 2
> a+b
[1] 5
```

変数も利用可能.
(変数宣言必要なし. 詳細は後で.)

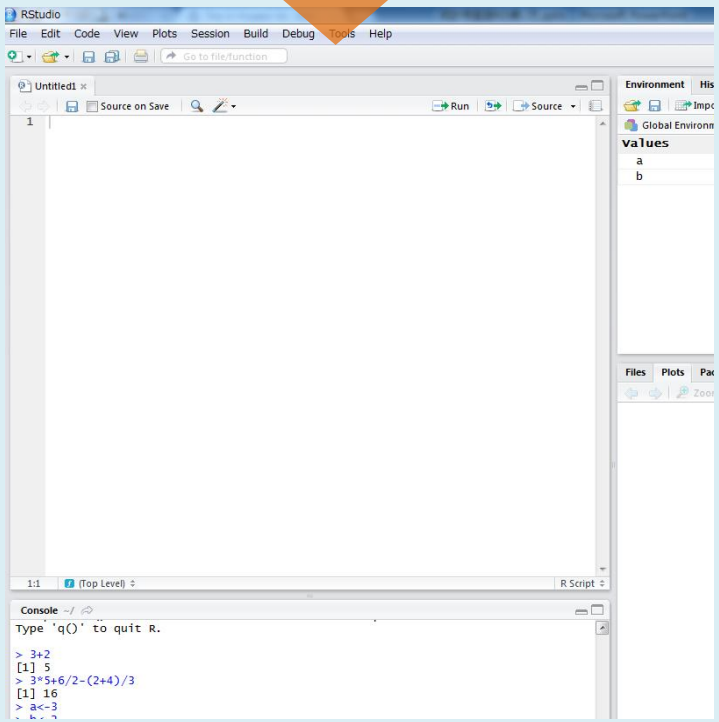
対話的に計算ができる.

電卓的利用.

もう少しプログラミングっぽい使い方



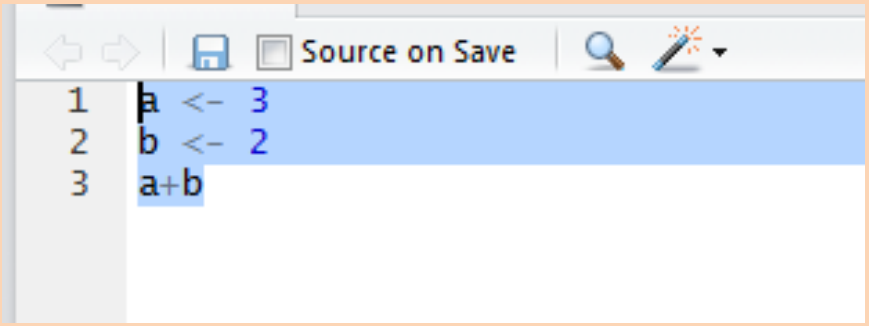
ここをクリック



左上部分に新しい画面が出てきた.
(**スクリプト**編集画面)

コンソールとスクリプト

- コンソール = 1行ずつ実行
- スクリプト = プログラムを**まとめて**実行



さっきのコードを書いて…
実行する範囲を選択,

CTRL + R

で実行!

ファイルを読み込む: getwd, setwd, read.table

- データファイルを読み込みたい
  - 現在の作業ディレクトリを確認

```
> getwd()
[1] "C:/Users/OKNAKI04/STree/takoshu"
```

  - 読み込みたいファイルのディレクトリに移動

```
> setwd("C:/Users/OKNAKI04/STree/takoshu/files")
```

  - ファイルを読み込む

```
> aa = read.table("data.txt")
```

read.csv など色々なコマンドがある. 詳細は以下:

http://cse.naro.affrc.go.jp/takezawa/r-tips/r/40.html

スクリプトの編集

- 出来るだけ**スクリプトを**使いましょう.
- 直接コンソールに打つと, 間違った時が大変.

統計っぽい解析

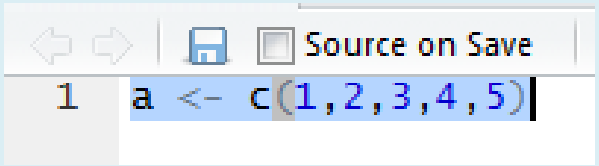
簡単な統計解析

[本節の内容]

- ✓ “ベクトル”という考え方
- ✓ 平均, 分散を計算する
- ✓ (偏差値を計算する)
- ✓ (ヒストグラムを描く)

“ベクトル”という考え方

- 配列みたいなもの.
- 数字の列ぐらいに思ってください.



aに(1,2,3,4,5)という数字の列を入れる

```
> a
[1] 1 2 3 4 5
```

aに5つの数字が入っている.

ベクトルに対して, 様々な関数が用意されています.

```
> max(a)
[1] 5
```

最大値

```
> mean(a)
[1] 3
```

平均値

```
> var(a)
[1] 2.5
```

分散

計算は“ベクトル”が基本

- 例) あるテストの平均点を求める.

	Aさん	Bさん	Cさん	Dさん	Eさん	Fさん	Gさん	Hさん
点数	55	45	65	30	85	90	95	50

1) 点数のデータをベクトルで表現

```
pts <- c(55,45,65,30,85,90,95,50)
```

2) 平均点を計算 (meanを使う)

```
> mean(pts)
[1] 64.375
```

3) 最高点等も計算できる!

```
> max(pts)
[1] 95
```

ベクトルに名前を付ける

①
```
> pts
[1] 55 45 65 30 85 90 95 50
```
←誰が何点なのか分からない

②
```
pts.names <- c("A","B","C","D","E","F","G","H")
```
名前を入れたベクトルを作成. (点数の順番と対応)

③
```
names(pts) <- pts.names
```
pts(点数一覧)に名前を入れる

④
```
> pts
 A  B  C  D  E  F  G  H
55 45 65 30 85 90 95 50
```
名前がつきました!

⑤
```
> which.max(pts)
G
7
```
最高点はGさんでした.

もっと巨大なデータの処理

200人分の点数のデータがあったとする.

```
> pts
  [1] 70 60 65 56 62 86 73 54 36 64 50 61 58 22 49 63 44 65 59 62 64 64 65 68 71
 [26] 47 56 53 51 51 57 20 39 67 63 76 62 95 49 58 38 40 73 74 37 81 74 90 48 53
 [51] 66 32 65 47 64 62 56 74 46 68 71 37 32 61 65 80 57 48 42 48 56 40 59 75 63
 [76] 49 80 67 92 48 75 36 59 69 43 68 58 78 84 72 56 40 56 67 35 66 47 61 62 51
[101] 51 50 51 50 57 62 73 43 63 71 48 66 68 68 58 60 73 54 38 40 49 58 69 51 87
[126] 54 57 52 64 57 48 64 62 64 61 80 62 93 71 34 63 62 55 75 39 53 57 42 34 45
[151] 33 25 50 52 70 76 65 85 71 57 56 81 69 32 53 53 90 47 71 69 45 73 41 75 67
[176] 68 49 56 58 74 39 71 77 73 68 40 77 47 50 39 52 82 65 56 71 59 45 64 66 47
```

mean, max, minを調べる

```
> mean(pts)
[1] 58.905
> max(pts)
[1] 95
> min(pts)
[1] 20
```

:平均点

:最高点

:最低点

手計算だとしんどい

魔法のコマンド(偏差値編)

$$\mu = \frac{1}{N}\sum_{t=1}^{N} X_i$$

$$\sigma^2 = \frac{1}{N}\sum_{t=1}^{N} (X_i - \mu)^2$$

$$Z_i := \frac{X_i - \mu}{\sigma}$$

```
normalized.pts <- scale(pts,center=TRUE,scale=TRUE)
```

$$R_i = 50 + 10Z_i$$:偏差値

```
50 + 10*normalized.pts
```

2行だけ書けば計算できる

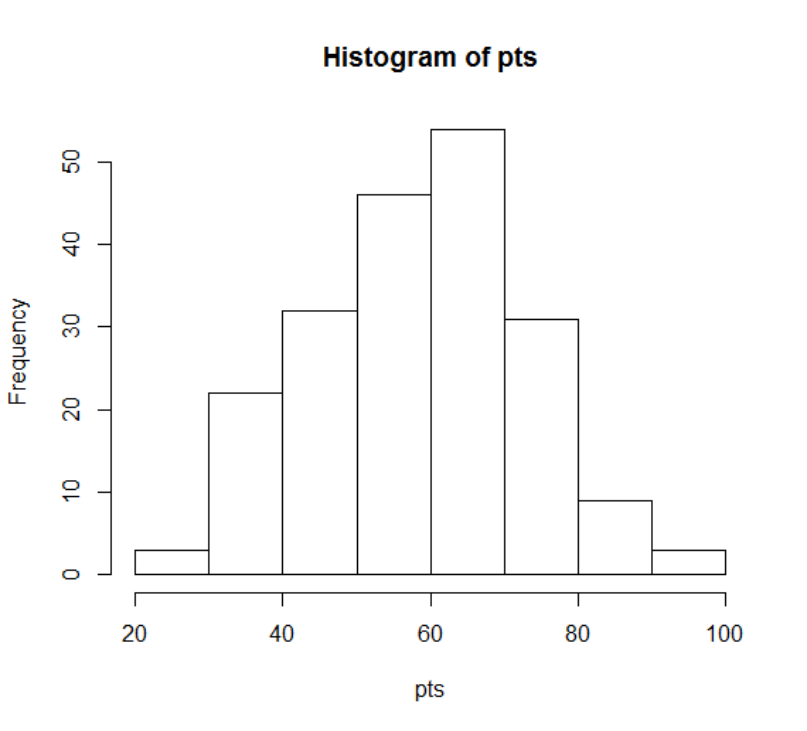
```
> 50 + 10*normalized.pts
          [,1]
[1,] 57.80125
[2,] 50.76993
[3,] 54.28559
[4,] 47.95740
[5,] 52.17619
[6,] 69.05137
[7,] 59.91065
[8,] 46.55114
```

グラフ的な処理

- さっきの点数のヒストグラムを描く

たった一言命令するだけ

```
> hist(pts)
```



まとめ

- R/Rstudioのダウンロード/インストール
- 計算はベクトルが基本
- コマンドいろいろあって便利です

グラフを描いてみよう

[本節の内容]

- ✓ データのプロット
- ✓ 線形回帰
- ✓ pairs()

データのプロット

事例）最高気温とアイスコーヒーの注文個数

最高気温（℃）	アイスコーヒー注文数（個数）
22	300
23	310
23	320
24	330
24	320
25	330
25	310
26	320
26	310
27	340
27	360
28	350
29	360
32	400
28	370
24	310
31	360
31	390
32	390
33	400
33	410
34	450
34	460
35	440
35	480

「アイスコーヒーの注文数を予測しよう」
http://markezine.jp/article/detail/16164

データを入力

- 頑張って入力します.

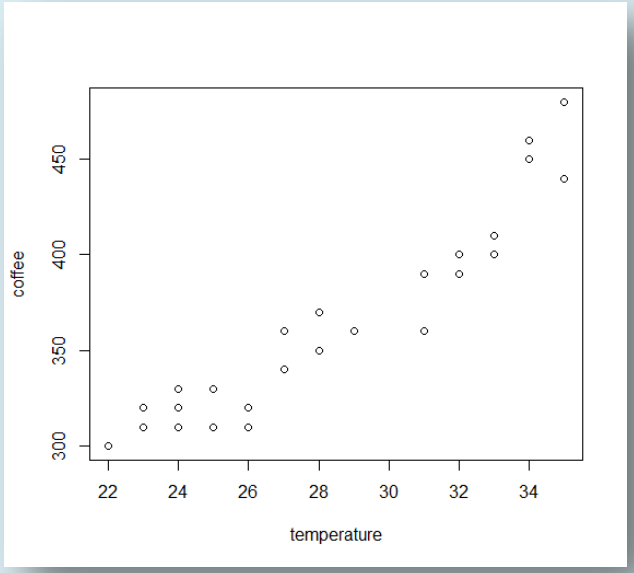
```
temperature <- c(22,23,23,24,24,25,25,26,26,27
coffee <- c(300,310,320,330,320,330,310,320,31
```

…こんな感じ

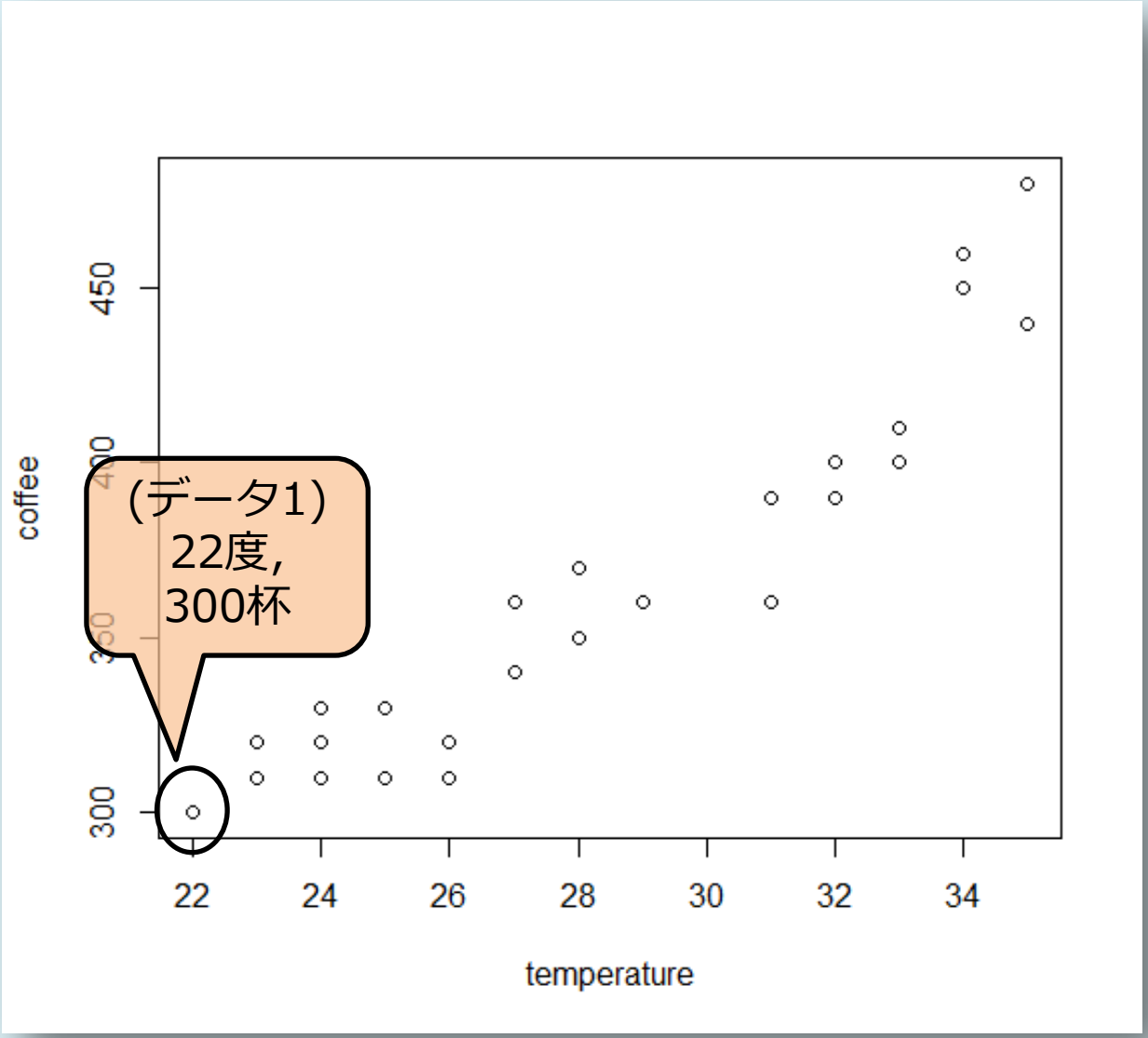
表の**上から順に**入力 (順番を変えてはいけません)

一言だけ命令

```
> plot(temperature,coffee)
```



図を拡大

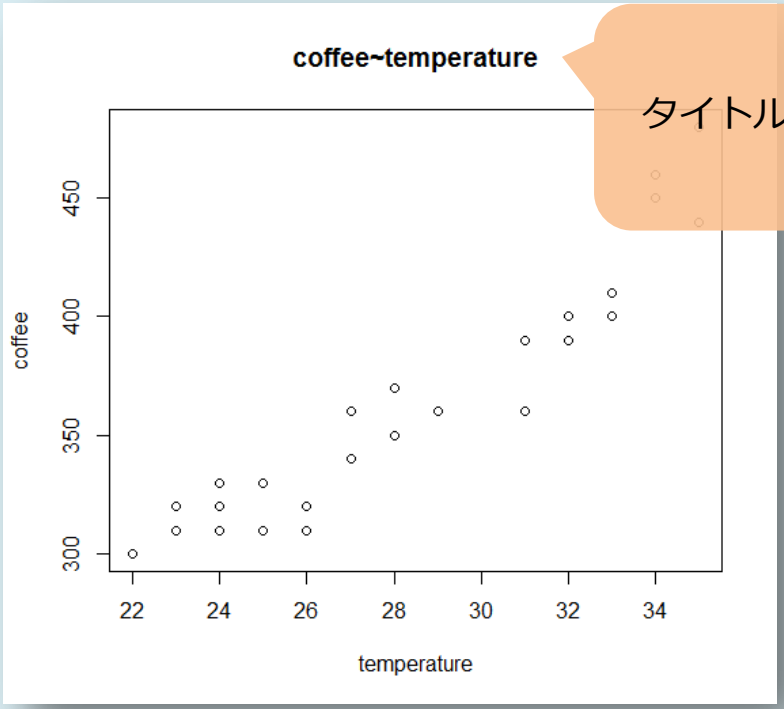


図を修正する

- plot() だと図は**自動で**調整される
- が, 手動で修正できる.

```
plot(temperature,coffee,xlim=c(22,35),
                        ylim=c(300,480),
                        main="coffee~temperature")
```

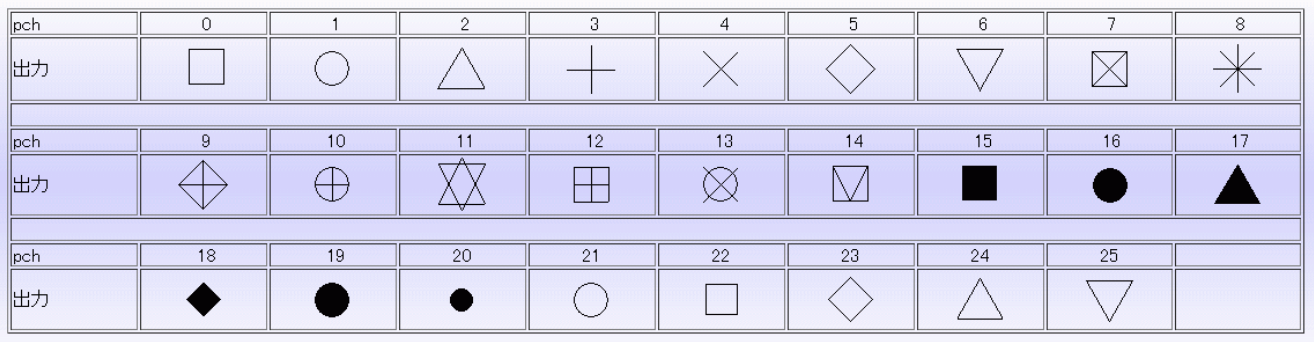
:x軸,y軸の範囲
:グラフのタイトル



色々なsettingを試してみましょう.
(点の間を線で結んでくれる設定等)

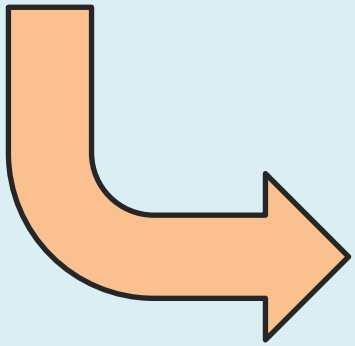
図を修正する(2)

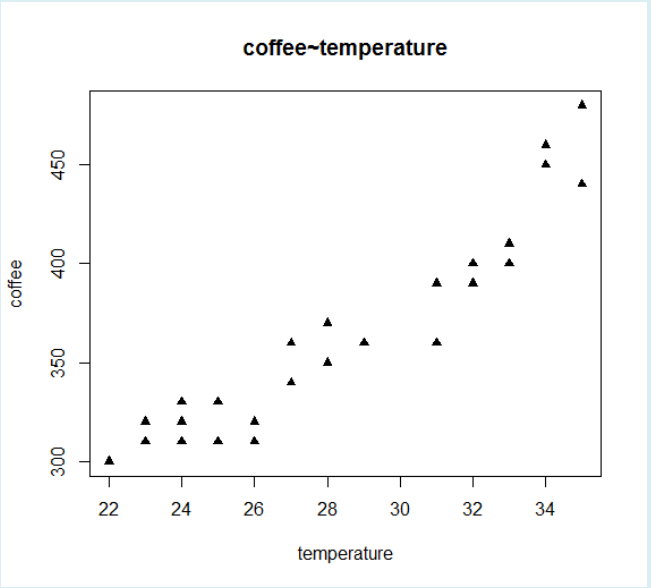
- pchとか変えてみる



[R-Source]
http://cse.naro.affrc.go.jp/takezawa/r-tips/r/53.html

```
plot(temperature,coffee,xlim=c(22,35),
                        ylim=c(300,480),
                        main="coffee~temperature",
                        pch=17)
```





相関係数を計算

- (統計を知っている人用)

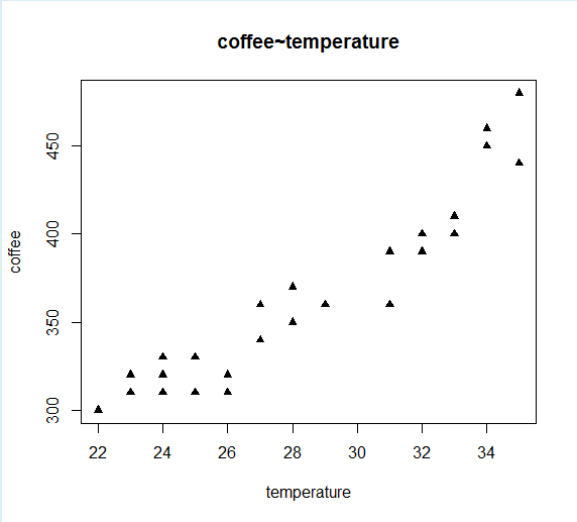
$$\mathrm{Cor}(X,Y) := \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{V}(X)\mathrm{V}(Y)}}$$

面倒な計算も一発

```
> cor(temperature,coffee)
[1] 0.9431283
```

かなり強い相関がある（ >0.8 ）
グラフから見ても線形に近似できそう

線形回帰



$$\text{coffee} = a \times \text{temperature} + b$$

計算の方法について,
興味のある人は「線形回帰」で調べましょう
省略!

また一言だけ命令:

```
> lm(coffee ~ temperature)
```

線形回帰(2)

```
> lm(coffee ~ temperature)

Call:
lm(formula = coffee ~ temperature)

Coefficients:
(Intercept)  temperature
      30.21        11.76
```

$$\text{coffee} = 11.76 \times \text{temperature} + 30.21$$

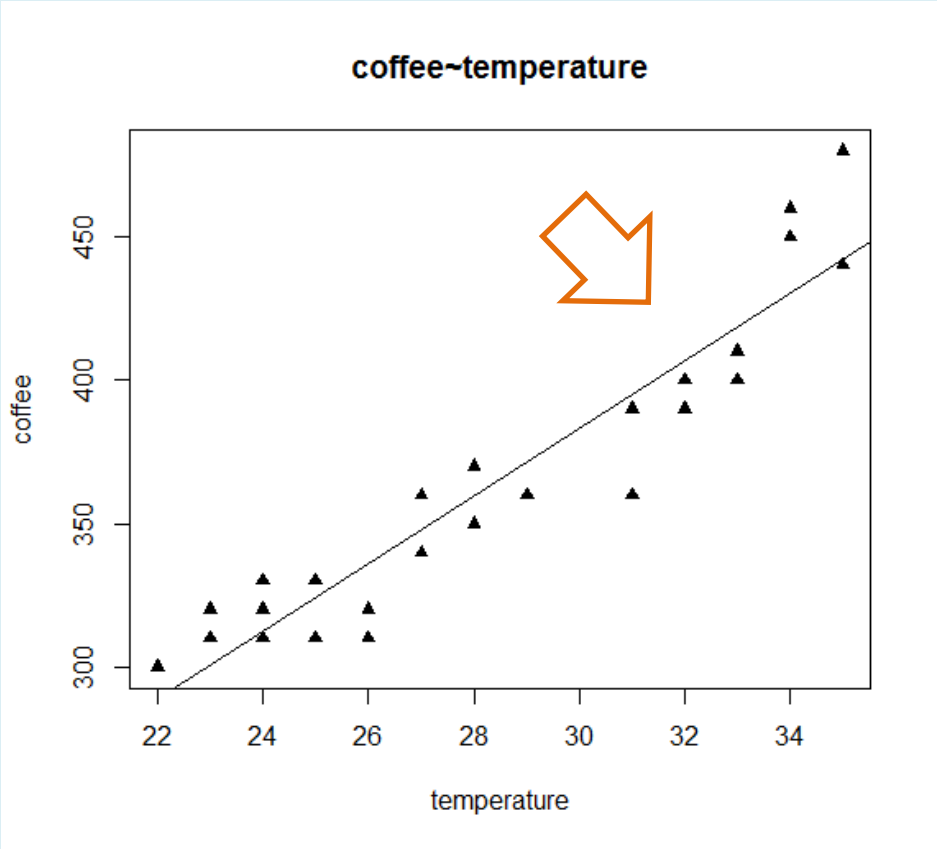
たった一言で式が求まった!

これをグラフに書き込んでみる

線形回帰(3)

回帰直線を図に書き込む

```
prd <- lm(coffee~temperature)
abline(prd)
```



pairs()

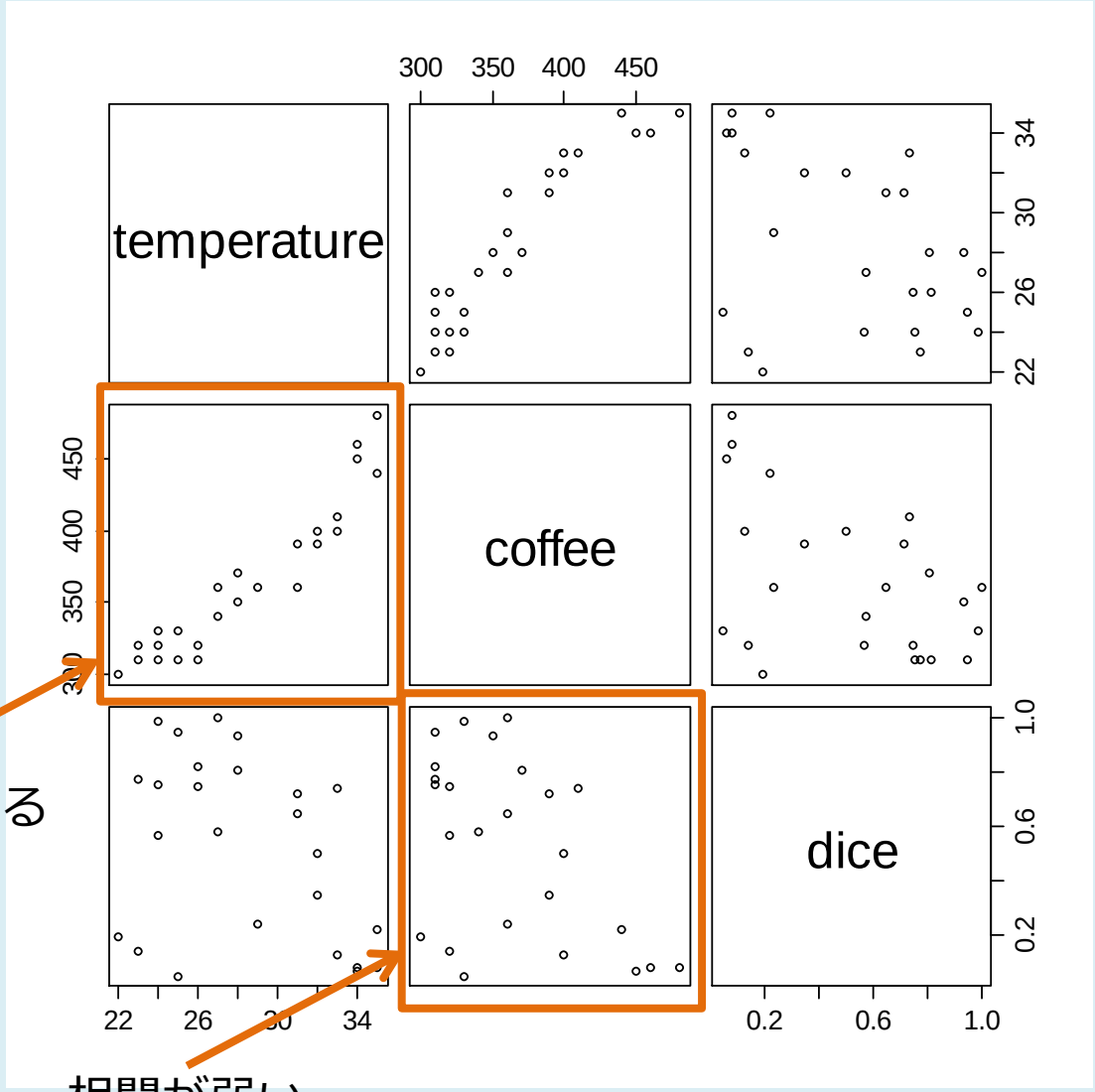
- 今あるデータ

Day	1	2	3	4	5	6	7	8	9
気温	22	23	23	24	24	25	26	26	27
コーヒー	300	310	320	330	320	330	310	320	310
乱数	0.19	0.77	0.14	0.99	0.57	0.05	0.95	0.75	0.58

データ間に関連があるか? 視覚的に見てみたい
⇒関数 pairs( )

```
> dice <- runif(25,0,1)
> pairs(cbind(temperature,coffee,dice))
```

pairs()



強い相関がある

相関が弱い

この章のまとめ

- データとplot()で図を書ける
- 手動調整用のコマンドも豊富
- 線形回帰も楽々
- pairs()とか全部勝手に書いてくれる

今日やったことのおさらい

- R/RStudioのDL/Install
- 簡単な統計解析
  - 平均や分散の計算
  - 偏差値とヒストグラム
- グラフを描いてみる
  - plot() 関数
  - 手動でのグラフの調整
  - 線形回帰

最後に

- 講習会終了後の質問は,
  - 月曜日16:00-18:00
  - 金曜日12:00-14:00
    @利用支援カウンター

- または, E-mail:
  sogo-ta52@library.osaka-u.ac.jp
  takoshu48@gmail.com まで.

おまけ

[本節の内容]

- ✓ 今日使ったソースコード
- ✓ 参考文献

今日使ったコード(1)

```
#analyze points
pts <- c(55,45,65,30,85,90,95,50)
mean(pts)
max(pts)

#big data
pts<-ceiling(rnorm(200,mean=60,sd=15))
min(pts)
max(pts)
mean(pts)

#standard score
normalized.pts <- scale(pts,center=TRUE,scale=TRUE)
50 + 10*normalized.pts

#coffee ~ temperature
temperature <-
    c(22,23,23,24,24,25,25,26,26,27,27,28,29,32,28,24,31,31,32,33,33,34,34,35,35)
coffee <- c(300,310,320,330,320,330,310,320,310,340,360,350,360,400,370,310,

            360,390,390,400,410,450,460,440,480)
```

今日使ったコード(2)

```
#simple plot
plot(temperature,coffee)

#manual
plot(temperature,coffee,xlim=c(22,35),
                        ylim=c(300,480),
                        main="coffee~temperature",
                        pch=17)

#linear regression
lm(coffee~temperature)

#plot predicted line
prd <- lm(coffee~temperature)
abline(prd)
```

参考文献

定番の本

The R tips データ解析環境Rの基本技・グラフィックス活用集 / 舟尾暢男著

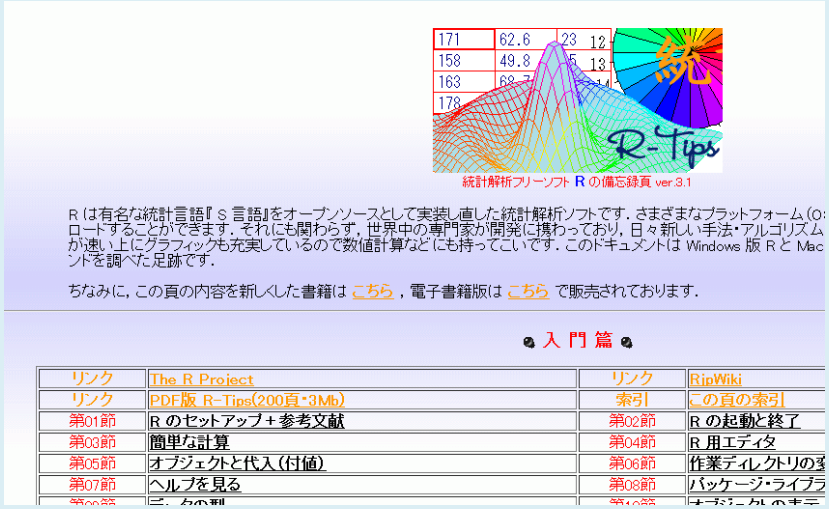
言語: Japanese

出版物情報: 東京 オーム社 2009.11.

版: 第2版

出版時期: 2009

同じ内容が無料で公開(web)



R-Tips
http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html

R-番外編

[本節の内容]

- ✓ ヘルプを読む(?mean)
- ✓ プログラミングっぽい書き方
- ✓ ベクトルの応用

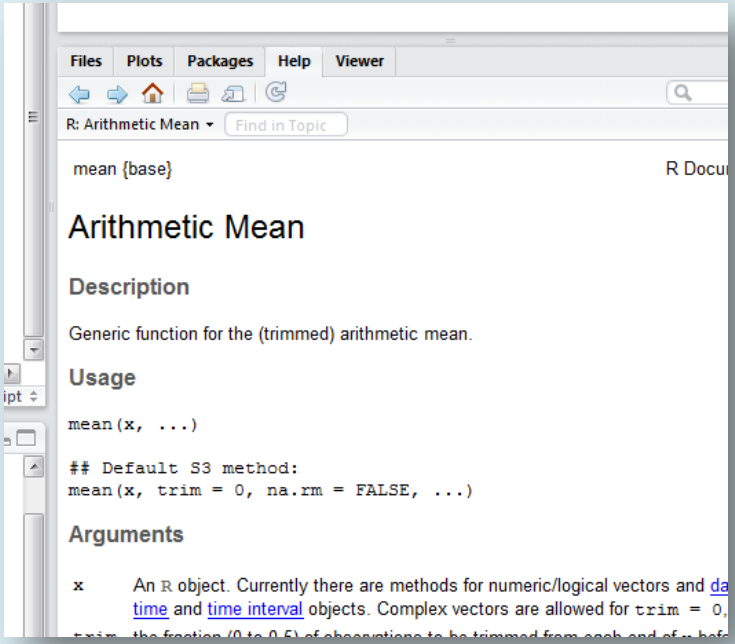
ヘルプを読む

- コマンド(mean等)のヘルプを読む.

例) mean() の使い方を調べる

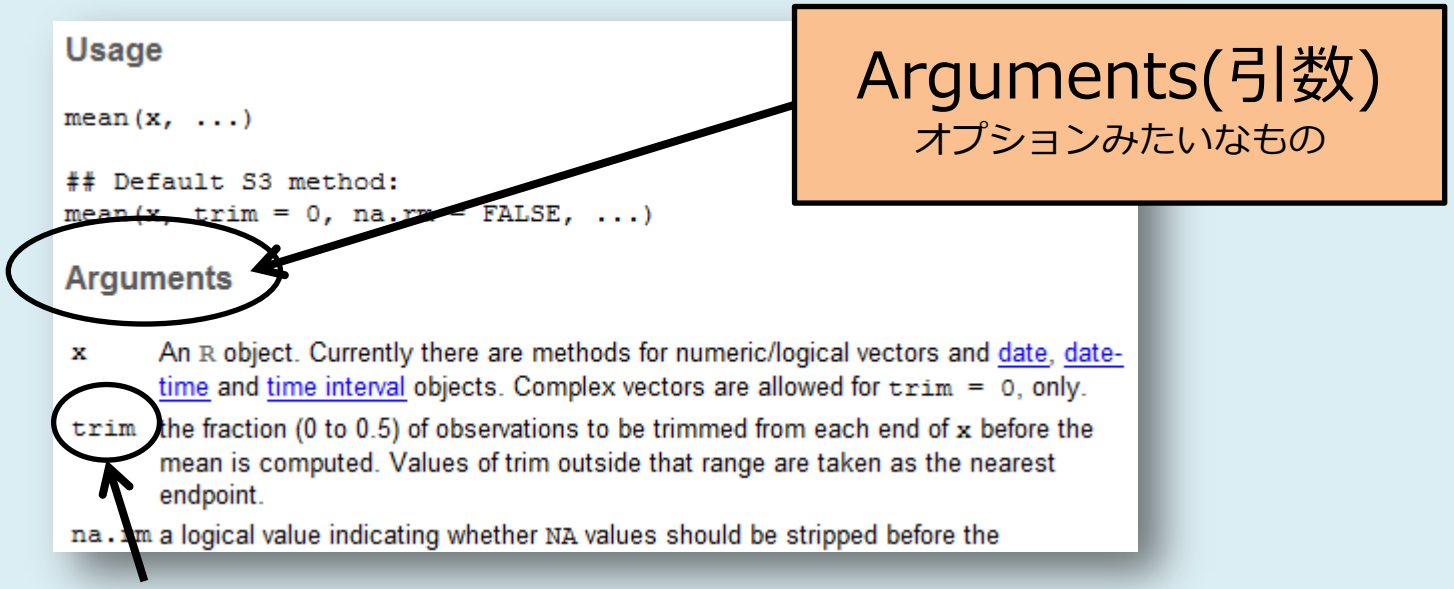
```
> ?mean
```

“**? + コマンド名**”



RStudioの右下のウインドウに
ヘルプが表示される.

mean()のhelp



- meanではtrimという引数を入力できる.
  (入力しなければデフォルト値=0)

ちょっと使ってみる.

meanの引数-trim

- ヘルプの下の方に, Exampleが載っている.

Examples

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

1行目:

xに(0,1,2,3,4,5,6,7,8,9,10,50)というベクトルを代入

```
> x<-c(0:10,50)
> x
 [1]  0  1  2  3  4  5  6  7  8  9 10 50
```

```
> mean(x)
[1] 8.75
```

```
> mean(x,trim=0.10)
[1] 5.5
```

何が違う?

trimの続き

- the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.

trim無し

(**0**,1,2,3,4,5,6,7,8,9,10,**50**)

$$\frac{0+1+2+...+10+50}{12} = 8.75$$

trim=0.1

(1,2,3,4,5,6,7,8,9,10)

$$\frac{1+2+...+10}{10} = 5.5$$

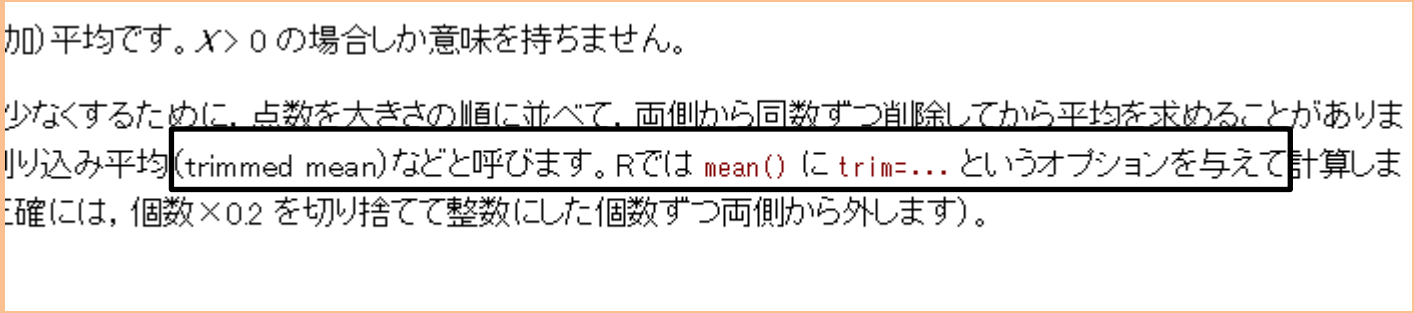
他にも色々調べてみましょう

- ?plot, ?hist, ?lm, ?abline, …等々

  けど分かりにくい場合もある.
  (英語を読むのが面倒臭い)

メジャーな関数なら…

Web検索. またはR-tipsの方が分かりやすいかも.


加)平均です。$X > 0$ の場合しか意味を持ちません。

少なくするために，点数を大きさの順に並べて，両側から同数ずつ削除してから平均を求めることがありま
刈り込み平均(trimmed mean)などと呼びます。Rでは mean() に trim=... というオプションを与えて計算しま
正確には，個数×0.2 を切り捨てて整数にした個数ずつ両側から外します)。

プログラミングっぽい書き方

1+2+3+…+100を求めたい.

方法1) ベクトルを使う方法.

```
> x <- seq(1,100,1)
> sum(x)
[1] 5050
```

方法2) forを使う方法.

```
> s <- 0
> for(i in 1:100) s <- s + i
> cat(s)
5050
```

少しプログラミング風になった.

プログラミングっぽい書き方

- for文について.
  - Rでfor文を使うと遅い.
  - forではなく**ベクトルを**使う方が良い.

```
> x <- seq(1,100,1)
> sum(x)
[1] 5050
```

速い.
(場合によっては数百倍)

>

```
> s <- 0
> for(i in 1:100) s <- s + i
> cat(s)
5050
```

通常の利用では, 単なる好みの問題

関数を作る

1変数の関数

```
f <- function(x){
  return(x+2)
}
```

```
> f(2)
[1] 4
```

2変数の関数

```
g <- function(x,y){
  return(x+3*y)
}
```

```
> g(1,2)
[1] 7
```

更に複雑な関数

```
h <- function(x){
  x.max <- max(x)
  x.min <- min(x)
  x.mean <- mean(x)
  return(list(max=x.max,
              min=x.min,
              mean=x.mean))
}
```

```
> h(pts)
$max
[1] 95

$min
[1] 30

$mean
[1] 64.375
```

ベクトル(pts)

ベクトルで計算する

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

全ての要素に適用される

```
a <- seq(1,5,1)

f <- function(x){
  return(4 + x^2 / (x+1))
}

f(a)
```

```
> a
[1] 1 2 3 4 5
> f(a)
[1] 4.500000 5.333333 6.250000 7.200000 8.166667
```

$4 + 2^2/(2+1) = 4 + 4/3 = 5.33333...$

ベクトルで計算する(発展)

$\sum_{k=1}^{100}(3k^2+2k)$ を求めるにはどうするか?

```
k <- seq(1,100,1)
sum(3*k^2 + 2*k)
```

```
> sum(3*k^2 + 2*k)
[1] 1025150
```

行列で計算する(発展)

$$\sum_{i,j=1}^{10} \frac{1}{ij}$$ をどうやって計算するか?

```
a <- seq(1,10,1)
b <- a %*% t(a)
sum(1/b)
```

```
> b
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    2    3    4    5    6    7    8    9    10
 [2,]    2    4    6    8   10   12   14   16   18    20
 [3,]    3    6    9   12   15   18   21   24   27    30
 [4,]    4    8   12   16   20   24   28   32   36    40
 [5,]    5   10   15   20   25   30   35   40   45    50
 [6,]    6   12   18   24   30   36   42   48   54    60
 [7,]    7   14   21   28   35   42   49   56   63    70
 [8,]    8   16   24   32   40   48   56   64   72    80
 [9,]    9   18   27   36   45   54   63   72   81    90
[10,]   10   20   30   40   50   60   70   80   90   100
```

```
> sum(1/b)
[1] 8.578855
```

(i,j)要素=ij

R-番外編, まとめ

- 困ったらヘルプを読みましょう.
  - Web検索, R-tips,
  - またはRから呼び出せるヘルプへ.

- プログラムっぽい書き方もできます.
  - for, while, repeat系も利用可.
  - ただしベクトルを利用した方が早い.
  - (発展)行列を使って書くのがRの醍醐味.
  - 関数も作れます. 活用しましょう.