Rブートキャンプ：エクササイズ

Ex.1 ファイル操作の基本

1. デスクトップ上に新しいフォルダを作成しなさい。フォルダ名は `応用心理学20210626` とすること
2. コマンドプロンプトを起動し、cdコマンドでカレントフォルダを確認しなさい
    - ヒント：「ファイル名を指名して実行」(`Win+R`)で `cmd` と入力
3. cdコマンドを使って、(1)で作ったフォルダに移動しなさい
    - ヒント：フォルダを右クリック→プロパティから、親フォルダの場所を確認できる
4. cdコマンドを使って、親フォルダに移動しなさい

Ex.2 RStudio

1. RStudioから新規Rスクリプトを作成しなさい
    - ヒント： File > New File > R Script
2. (1)で作ったRスクリプトをEx.1で作ったフォルダに保存しなさい。ファイル名は `r_bootcamp.r` とすること
3. 一度RStudio上でファイルを閉じ、(2)で作ったファイルを開き直しなさい
4. エディタ上でRスクリプトに `print('hello')` と入力し、実行しなさい
    - ヒント：実行したい部分を選択して `Ctrl(command) + Enter` でスクリプトを実行できる

Ex.3 はじめに覚えておくこと

1. コンソールに `# print('comment')` と入力して実行し、実行結果が **出ない** ことを確認しなさい

```
# print('comment')
```

2. エディタ上で `?mean` と入力して実行し、ヘルプが表示されることを確認しなさい

```
?mean
```

mean {base}	R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)
```

Default S3 method:

mean(x, trim = 0, na.rm = FALSE, ...)

Arguments

`x`	An `R` object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects. Complex vectors are allowed for `trim = 0`, only.
`trim`	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.
`na.rm`	a logical value indicating whether `NA` values should be stripped before the computation proceeds.
`...`	further arguments passed to or from other methods.

Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

`weighted.mean`, `mean.POSIXct`, `colMeans` for row and column means.

Examples

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

[Package *base* version 4.0.2]

3. `getwd()`を実行して作業ディレクトリを確認しなさい

```
# 割愛
```

4. setwd()関数を使ってEx.1-1で作ったフォルダを作業ディレクトリに設定しなさい

```
# 割愛
```

5. 以下を順に実行しなさい

(1) `test = 1`

```
test = 1
```

(2) `print(test)`

```
print(test)
```

```
[1] 1
```

(3) `print(TEST)` (エラーが出ることを確認する)

```
# print(TEST) # エラーが出る
```

Ex.4 演算

1. [四則演算] x = 3, y = 4, z = 5として以下の値を求めなさい

(1) xとyとzの和

```
x = 3
y = 4
z = 5
```

```
x + y + z
```

12

(2) xとyの和をzで割った値

```
(x + y) / z
```

1.4

(3) zのx乗

```
z ^ x
```

125

(4) zをxで割った商とあまり(整数除算)

```
# 商
z %/% x
# あまり
z %% x
```

1

2

2. [数学関数] x = c(1,2,3,4,5)として、以下を実行しなさい

(1) `sum(x)`

```
x = c(1, 2, 3, 4, 5)
sum(x)
```

15

(2) `mean(x)`

```
mean(x)
```

3

(3) `var(x)`

```
var(x)
```

2.5

(4) `sum((x - mean(x))^2) / 5`

```
sum((x - mean(x))^2) / 5
```

2

3. [比較演算子] x=1, y=2として以下の計算を実行しなさい

(1) `x == y`

```
x = 1
y = 2
x == y
```

FALSE

(2) `x != y`

```
x != y
```

TRUE

(3) `x > y`

```
x > y
```

FALSE

(4) `x < y`

```
x < y
```

TRUE

(5) `x %in% c(2,4)`

```
x %in% c(2, 4)

```

FALSE

(6) `y %in% c(2,4)`

```
y %in% c(2, 4)
```

TRUE

4. [論理演算その１] x = c(T, T, F, F), y = c(T, F, T, F)として、以下を実行しなさい

(1) `x & y`

```
x = c(T, T, F, F)
y = c(T, F, T, F)
x & y
```

TRUE · FALSE · FALSE · FALSE

(2) `x && y`

```
x && y
```

TRUE

(3) `x | y`

```
x | y
```

TRUE · TRUE · TRUE · FALSE

(4) `x || y`

```
x || y
```

TRUE

(5) `!x`

```
!x
```

FALSE · FALSE · TRUE · TRUE

5. [論理演算その２] x = c(T, T, T), y = c(T, T, F)として、以下を実行しなさい

(1) `all(x)`

```
x = c(T, T, T)
y = c(T, T, F)
all(x)
```

TRUE

(2) `all(y)`

```
all(y)
```

FALSE

(3) `any(x)`

```
any(x)
```

TRUE

(4) `any(y)`

```
any(y)
```

TRUE

6. [NAとの計算] 以下を実行しなさい

(1) `1 + NA`

```
1 + NA
```

<NA>

(2) `0 * NA`

```
0 * NA
```

<NA>

(3) `sum(1,2,NA)`

```
sum(1, 2, NA)
```

<NA>

(4) `T & NA`

```
T & NA
```

<NA>

(5) `F & NA`

```
F & NA
```

FALSE

(6) `T | NA`

```
T | NA
```

TRUE

(7) `F | NA`

```
F | NA
```

<NA>

1. [ベクトルの作成] 以下のベクトルを作成しなさい

(1) 1から5までの整数を順に並べたベクトル

```
c(1,2,3,4,5)
```

1 · 2 · 3 · 4 · 5

(2) a,b,c,dのそれぞれの文字を要素とするベクトル

```
c('a', 'b', 'c', 'd')
```

'a' · 'b' · 'c' · 'd'

(3) コロン演算子`:`を使って、1から10までの整数を順に並べたベクトル

```
1:10
```

1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10

(4) コロン演算子`:`を使って、8から2までの整数を大きい順に並べたベクトル

```
8:2
```

8 · 7 · 6 · 5 · 4 · 3 · 2

(5) c(1,2,3)を4つ並べたベクトル

```
rep(c(1,2,3), 4)
```

1 · 2 · 3 · 1 · 2 · 3 · 1 · 2 · 3 · 1 · 2 · 3

2. [ベクトル演算] x = 1:5, y = c(2,4), z=c(5, 3, 1)として、以下の計算をしなさい

(1) `x + 1`

```
x = 1:5
y = c(2, 4)
z = c(5, 3, 1)

x + 1
```

2 · 3 · 4 · 5 · 6

(2) `y * 2`

```
y * 2
```

4 · 8

(3) `x + y`

```
x + y
```

```
Warning message in x + y:
"longer object length is not a multiple of shorter object length"
```

3 · 6 · 5 · 8 · 7

(4) `x * z`

```
x * z
```

```
Warning message in x * z:
"longer object length is not a multiple of shorter object length"
```

5 · 6 · 3 · 20 · 15

(5) `sqrt(x)`

```
sqrt(x)
```

1 · 1.4142135623731 · 1.73205080756888 · 2 · 2.23606797749979

(6) `mean(y)`

```
mean(y)
```

3

(7) `rev(x)`

```
rev(x)
```

5 · 4 · 3 · 2 · 1

(8) `sort(z)`

```
sort(z)
```

1 · 3 · 5

3. [要素へのアクセス] x = c('a', 'b', 'c', 'd', 'e'), y = 1:6として、以下の操作をしなさい

(1) 添字番号を使ってxの3番目の要素を取り出す

```
x = c('a', 'b', 'c', 'd', 'e')
y = 1:6
x[3]
```

'c'

(2) 整数ベクトルを使ってxの2番目と5番目の要素を取り出す

```
x[c(2,5)]
```

'b' · 'e'

(3) xの1番目の要素をfに変更する

```
x[1] = 'f'
```

(4) 論理ベクトルを使ってxの1番目と4番目の要素を取り出す

```
x[c(T,F,F,T,F)]
```

'f' · 'd'

(5) yの偶数である要素を取り出す

```
y[y %% 2 == 0]
```

2 · 4 · 6

(6) yの2番目と3番目以外の要素を取り出す

```
y[-c(2,3)]
```

1 · 4 · 5 · 6

Ex.6 関数

1. [関数の作成]以下の関数を作成しなさい

(1) 実行すると'done.'と表示する関数

```
func = function(){
    print('done.')
}
func()
```

```
[1] "done."
```

(2) 数値が入力されると、入力値の2倍足す1の値を返す関数

```
func = function(number){
    return(number * 2 + 1)
}
```

```
func(1)
```

3

```
func(3)
```

7

2. [制御構文] 以下の問に答えなさい。ただし整数以外の入力値については考慮しなくてよいものとする

(1) 以下の関数を作成しなさい

(1-1) 入力された整数が偶数なら'even', 奇数なら'odd'と表示する関数

```
print_even_odd = function(number){
    if(number %% 2 == 0){
        print('even')
    }else{
        print('odd')
    }
}
```

```
print_even_odd(1)
```

```
[1] "odd"
```

```
print_even_odd(4)
```

```
[1] "even"
```

(1-2) 入力された整数を3で割った余りによって以下の表示をする関数

- 余りが2ならば'two'
- 余りが1ならば'one'
- 余りが0ならば'zero'

```
print_remainder = function(number){
    if(number %% 3 == 0){
        print('zero')
    }else if(number %% 3 == 1){
        print('one')
    }else{
        print('two')
    }
}
```

```
print_remainder(3)
```

```
[1] "zero"
```

```
print_remainder(2)
```

```
[1] "two"
```

```
print_remainder(4)
```

```
[1] "one"
```

(2) 前問(1)で作成したそれぞれの関数について、for文を使って1から20までの整数値を入力した結果を表示しなさい

```
for(i in 1:20){
    print_even_odd(i)
}
```

```
[1] "odd"
[1] "even"
[1] "odd"
[1] "even"
[1] "odd"
[1] "even"
[1] "odd"
[1] "even"
[1] "odd"
[1] "even"
[1] "odd"
```

```
[1] "even"
[1] "odd"
[1] "even"
[1] "odd"
[1] "even"
[1] "odd"
[1] "even"
[1] "odd"
[1] "even"
```

```
for(i in 1:20){
    print_remainder(i)
}
```

```
[1] "one"
[1] "two"
[1] "zero"
[1] "one"
[1] "two"
[1] "zero"
[1] "one"
[1] "two"
[1] "zero"
[1] "one"
[1] "two"
[1] "zero"
[1] "one"
[1] "two"
[1] "zero"
[1] "one"
[1] "two"
[1] "zero"
[1] "one"
[1] "two"
```

(3) 入力された数字が3で割り切れるなら'fizz', 5で割り切れるなら'buzz', 3でも5でも割り切れるなら'fizzbuzz'を、 それ以外は入力された数字を表示する関数を作成し、1から30までの整数までを入力した結果を表示しなさい

```
fizzbuzz = function(number){
    if(number %% 3 == 0){
        if(number %% 5 == 0){
            print('fizzbuzz')
        }else{
            print('fizz')
        }
    }else if(number %% 5 == 0){
        print('buzz')
    }else{
        print(number)
    }
}


```

```
for(i in 1:30){
    fizzbuzz(i)
}
```

```
[1] 1
[1] 2
[1] "fizz"
[1] 4
[1] "buzz"
[1] "fizz"
[1] 7
[1] 8
[1] "fizz"
[1] "buzz"
[1] 11
[1] "fizz"
[1] 13
[1] 14
[1] "fizzbuzz"
[1] 16
[1] 17
[1] "fizz"
[1] 19
[1] "buzz"
[1] "fizz"
[1] 22
[1] 23
```

```
[1] "fizz"
[1] "buzz"
[1] 26
[1] "fizz"
[1] 28
[1] 29
[1] "fizzbuzz"
```

3. ★[forとベクトル] Sys.time()関数は現在時刻を取得できる関数である。

処理の終了時刻と開始時刻の差をとることで、処理の実行時間を計測できる。

```
start = Sys.time() # 開始時刻

# 実行時間を測りたい処理

end = Sys.time() # 終了時刻
result = end - start # 実行時間
```

Sys.time()関数を用いて以下のそれぞれの実行時間を計測し、比較しなさい

(1) 1から100,000までの整数の和をfor文を使って求める

```
x = 0
start = Sys.time()
for(i in 1:100000){
    x = x + i
}
end = Sys.time()
result = end - start
print(result)
```

```
Time difference of 0.01099896 secs
```

(2) 1から100,000までの整数の和をsum()を使って求める

```
start = Sys.time()
x =sum(1:100000)
end = Sys.time()
result = end - start
print(result)
```

```
Time difference of 0.002995968 secs
```

Ex.7 データフレーム

1. [データフレームの操作] `excersise`フォルダに`ex7_1_a.tsv`、`ex7_1_d.tsv`が入っている（以下、ファイルADと呼ぶ）。 これはある年における講義科目の学生の成績が入っている架空のデータである。学生は10人いて、社会学(sociology)と心理学(psychology)が必修であるため全員が受講し、 経済学(economics)と政治学(politics)は選択科目のため受講していない学生もいる。以下の問に答えなさい

(1) ファイルADをデータフレームとして読み込みなさい(名前はdfadfdとすること)

```
dfa = read.delim('exercise/ex7_1_a.tsv', sep='\t')
dfb = read.delim('exercise/ex7_1_b.tsv', sep='\t')
dfc = read.delim('exercise/ex7_1_c.tsv', sep='\t')
dfd = read.delim('exercise/ex7_1_d.tsv', sep='\t')
```

```
dfa
```

A data.frame: 5 × 3

ID	sociology	psychology
<int>	<int>	<int>
1	8	7
2	5	6
3	6	10
4	1	8
5	5	10

```
dfb
```

A data.frame: 5 × 3

ID	sociology	psychology
<int>	<int>	<int>
6	3	9
7	4	3
8	9	3
9	6	8
10	2	6

```
dfc
```

A data.frame: 7 × 2

ID	economics
<int>	<int>
3	6
4	1
5	10
6	5
7	7
8	8
10	9

```
dfd
```

A data.frame: 7 × 2

ID	politics
<int>	<int>
2	4
4	2
5	9
7	5
8	9
9	9
10	7

(2) ファイルAはID1$_{\text{ID5の学生の、ファイルBはID6}}$ID10の学生の社会学と心理学の成績である。rbind()関数を用いて両者を結合し、新しいデータフレームを作りなさい（名前はdf_abとすること）

```
df_ab = rbind(dfa, dfb)
```

(3) ファイルCは経済学、ファイルDは政治学の成績である。merge()関数を用いて以下の結合結果を確かめなさい。

(3-1) 経済学と政治学の両方を受講している生徒のデータのみ使う

```
merge(dfc, dfd)
```

A data.frame: 5 × 3

ID	economics	politics
<int>	<int>	<int>
4	1	2
5	10	9
7	7	5
8	8	9
10	9	7

(3-2) 経済学を受講している生徒のデータはすべて使い、政治学のみ受講している生徒のデータは使わない

```
merge(dfc, dfd, all.x=T)
```

A data.frame: 7 × 3

ID	economics	politics
<int>	<int>	<int>
3	6	NA
4	1	2
5	10	9
6	5	NA
7	7	5
8	8	9
10	9	7

(3-3) 政治学を受講している生徒のデータはすべて使い、経済学のみ受講している生徒のデータは使わない

```
merge(dfc, dfd, all.y=T)
```

A data.frame: 7 × 3

ID	economics	politics
<int>	<int>	<int>
2	NA	4
4	1	2
5	10	9
7	7	5
8	8	9
9	NA	9
10	9	7

(3-4) ファイルC, ファイルDに含まれるすべての生徒のデータを使う

```
df_cd = merge(dfc, dfd, all=T)
print(df_cd)
```

```
   ID economics politics
1  2        NA        4
2  3         6       NA
3  4         1        2
4  5        10        9
5  6         5       NA
6  7         7        5
7  8         8        9
8  9        NA        9
9 10         9        7
```

(4) 前問(3-4)の結合結果をdf_cbと呼ぶことにする。merge()関数を用いて、df_abとdf_cbを結合したデータフレームを作成しなさい(名前はdf_abcdとする)。ただしすべてのデータを使う(all=T)こと。

```r
df_abcd = merge(df_ab, df_cd, all=T)
print(df_abcd)
```

```
   ID sociology psychology economics politics
1   1         8          7        NA       NA
2   2         5          6        NA        4
3   3         6         10         6       NA
4   4         1          8         1        2
5   5         5         10        10        9
6   6         3          9         5       NA
7   7         4          3         7        5
8   8         9          3         8        9
9   9         6          8        NA        9
10 10         2          6         9        7
```

(5) df_abcdについて、学生の成績の平均値を求め、列名を`GPA`として新しい列を追加しなさい

```r
df_abcd['GPA'] = rowMeans(df_abcd[2:5], na.rm = T)
print(df_abcd)
```

```
   ID sociology psychology economics politics      GPA
1   1         8          7        NA       NA 7.500000
2   2         5          6        NA        4 5.000000
3   3         6         10         6       NA 7.333333
4   4         1          8         1        2 3.000000
5   5         5         10        10        9 8.500000
6   6         3          9         5       NA 5.666667
7   7         4          3         7        5 4.750000
8   8         9          3         8        9 7.250000
9   9         6          8        NA        9 7.666667
10 10         2          6         9        7 6.000000
```

(6) df_abcdをtsvファイルとして書き出しなさい

```r
write.table(df_abcd, 'exercise/ex7_1.tsv', sep='\t', quote=F, row.names=F)
```

2. [データフレームと関数] `USJudgeRatings`は弁護士による米国高等裁判所における州裁判官の評価である。 なお、各変数の説明は以下の通りである：

```
[,1]    CONT    Number of contacts of lawyer with judge. # 弁護士と裁判官の接触回数
[,2]    INTG    Judicial integrity. # 司法の廉潔性
[,3]    DMNR    Demeanor. # 態度
[,4]    DILG    Diligence. # 勤勉さ
[,5]    CFMG    Case flow managing. # 事案管理
[,6]    DECI    Prompt decisions. # 即決性
[,7]    PREP    Preparation for trial. # 審理への準備
[,8]    FAMI    Familiarity with law. # 法律への熟知性
[,9]    ORAL    Sound oral rulings. # 口頭決定の健全性
[,10]   WRIT    Sound written rulings. # 書面決定の健全性
[,11]   PHYS    Physical ability. # 身体能力
[,12]   RTEN    Worthy of retention. # 留保の適切さ
```

`USJudgeRatings`をデータセットとして以下の問に答えなさい

(1) psychパッケージをインストールしていないならば`install.packages('psych')`を実行し、インストールしなさい

```r
# 割愛
```

(2) library()関数でpsychパッケージを読み込んだ上でdescribe()関数を適用し、各変数の要約統計量を算出しなさい

```r
library('psych')
describe(USJudgeRatings)
```

A psych: 12 × 13

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurto
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
CONT	1	43	7.437209	0.9408768	7.3	7.354286	0.74130	5.7	10.6	4.9	1.0483108	1.512
INTG	2	43	8.020930	0.7701447	8.1	8.097143	0.74130	5.9	9.2	3.3	-0.8135419	0.257
DMNR	3	43	7.516279	1.1437054	7.7	7.642857	1.18608	4.3	9.0	4.7	-0.9146990	0.274
DILG	4	43	7.693023	0.9008978	7.8	7.771429	1.03782	5.1	9.0	3.9	-0.7561970	0.146
CFMG	5	43	7.479070	0.8601102	7.6	7.568571	0.88956	5.4	8.7	3.3	-0.7627529	-0.096
DECI	6	43	7.565116	0.8029362	7.7	7.634286	0.74130	5.7	8.8	3.1	-0.6215587	-0.531
PREP	7	43	7.467442	0.9533702	7.7	7.542857	0.88956	4.8	9.1	4.3	-0.6572536	-0.003
FAMI	8	43	7.488372	0.9489868	7.6	7.554286	1.03782	5.1	9.1	4.0	-0.5377923	-0.365
ORAL	9	43	7.293023	1.0100437	7.5	7.388571	0.74130	4.7	8.9	4.2	-0.7530389	0.012
WRIT	10	43	7.383721	0.9611328	7.6	7.460000	0.88956	4.9	9.0	4.1	-0.6726825	-0.108
PHYS	11	43	7.934884	0.9395753	8.1	8.077143	0.59304	4.7	9.1	4.4	-1.5041764	2.159
RTEN	12	43	7.602326	1.1009711	7.8	7.731429	0.88956	4.8	9.2	4.4	-0.9373609	0.255

(3) データセットの行数と列数を求めなさい

```
ncol(USJudgeRatings) # 列数
nrow(USJudgeRatings) # 行数
```

12

43

(4) colMeans()を用いて、各変数について裁判官全体の平均値を求めなさい

```
colMeans(USJudgeRatings)
```

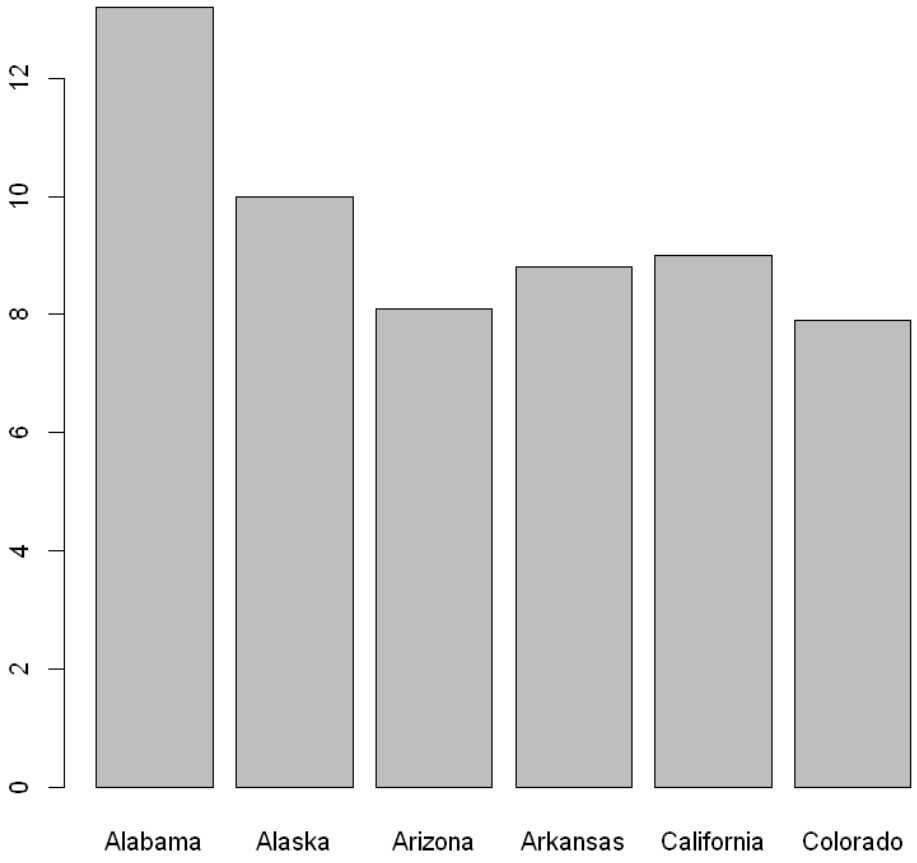
CONT: 7.43720930232558 INTG: 8.02093023255814 DMNR: 7.51627906976744 DILG: 7.69302325581395
CFMG: 7.47906976744186 DECI: 7.56511627906977 PREP: 7.46744186046512 FAMI: 7.48837209302326
ORAL: 7.29302325581395 WRIT: 7.38372093023256 PHYS: 7.93488372093023 RTEN: 7.60232558139535

Ex.8 作図

1. [棒グラフと散布図] `USArrests` は1973年の米国における10万人当たりの暴力犯罪の件数および都市人口データである。
   `USArrests` をデータセットとして用いて以下の問に答えなさい

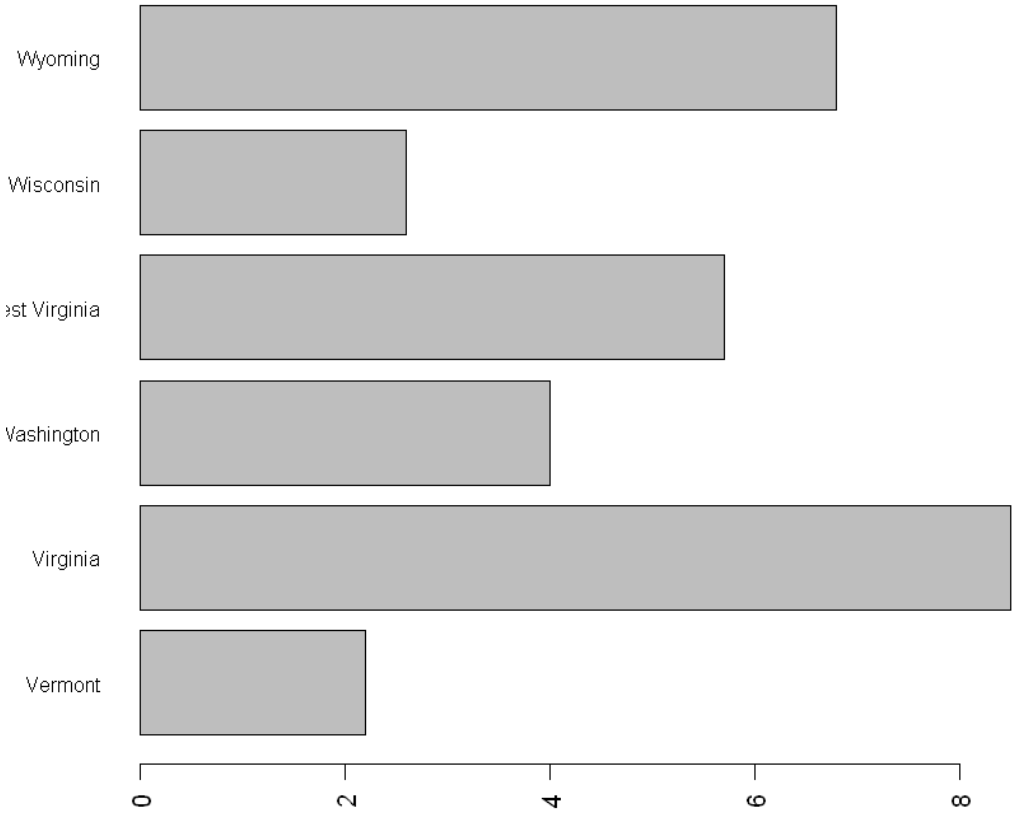
(1) データセットの先頭5件を抽出し、殺人(Murder)の値を縦棒グラフで表示しなさい

```
top = head(USArrests)
barplot(top$Murder, names.arg=rownames(top))
```



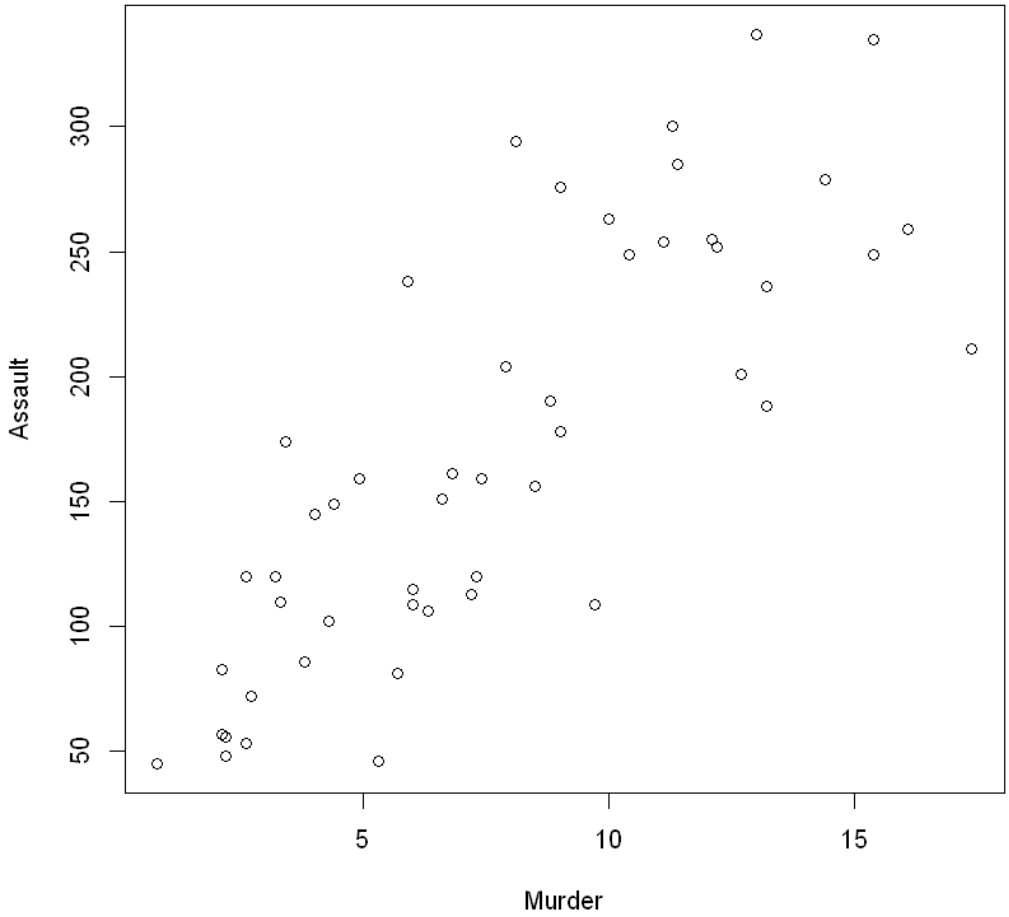
(2) データセットの末尾5件を抽出し、暴行(Assault)の値を横棒グラフで表示しなさい

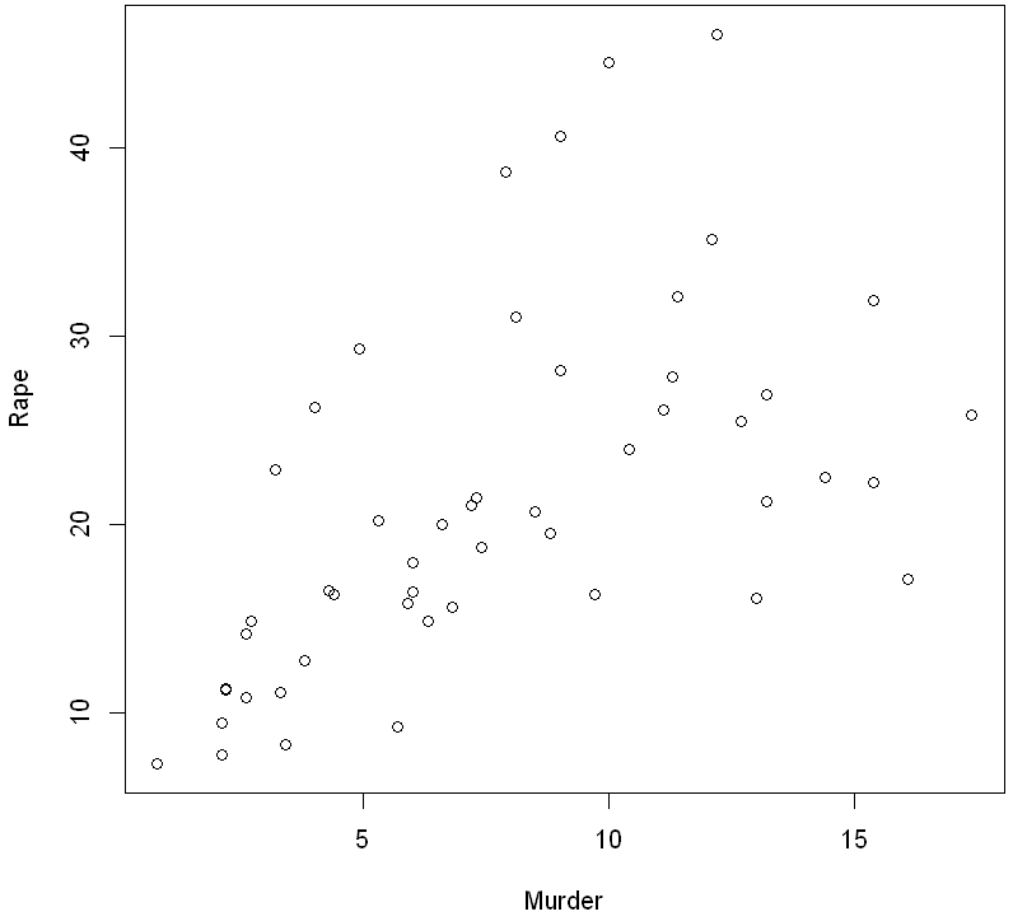
```
bottom = tail(USArrests)
barplot(bottom$Murder, names.arg=rownames(bottom), horiz=T, las=2, cex.names=0.8) # cex.namesでラベルの大きさを調整
```

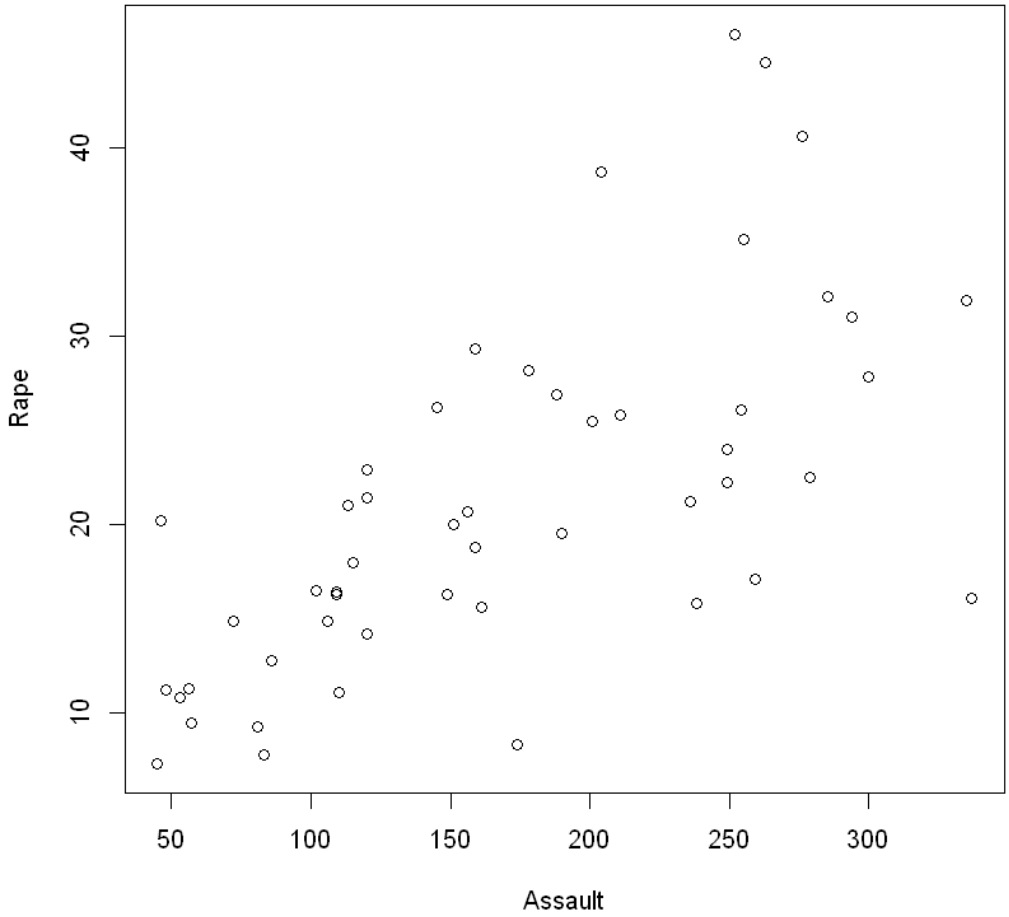


(3) plot()関数を用いて三種類の暴力犯罪(Murder, Assault, Rape)について、すべての組み合わせについて散布図を作成しなさい

```
plot(USArrests[c('Murder', 'Assault')])
plot(USArrests[c('Murder', 'Rape')])
plot(USArrests[c('Assault', 'Rape')])
```

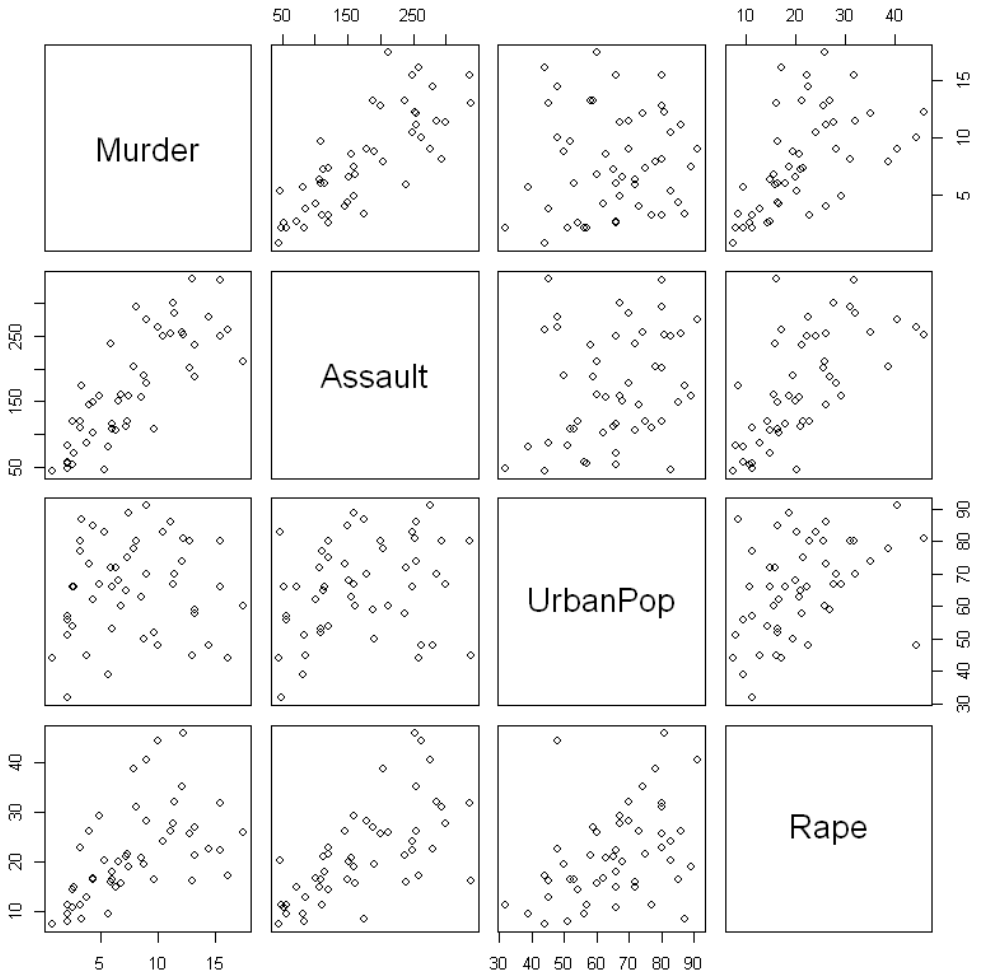

Assault
50
100
150
200
250
300
5
10
15
Murder


Rape
10
20
30
40
5
10
15
Murder



(4) pairs()関数を用いてすべての変数についての散布図行列を作成しなさい

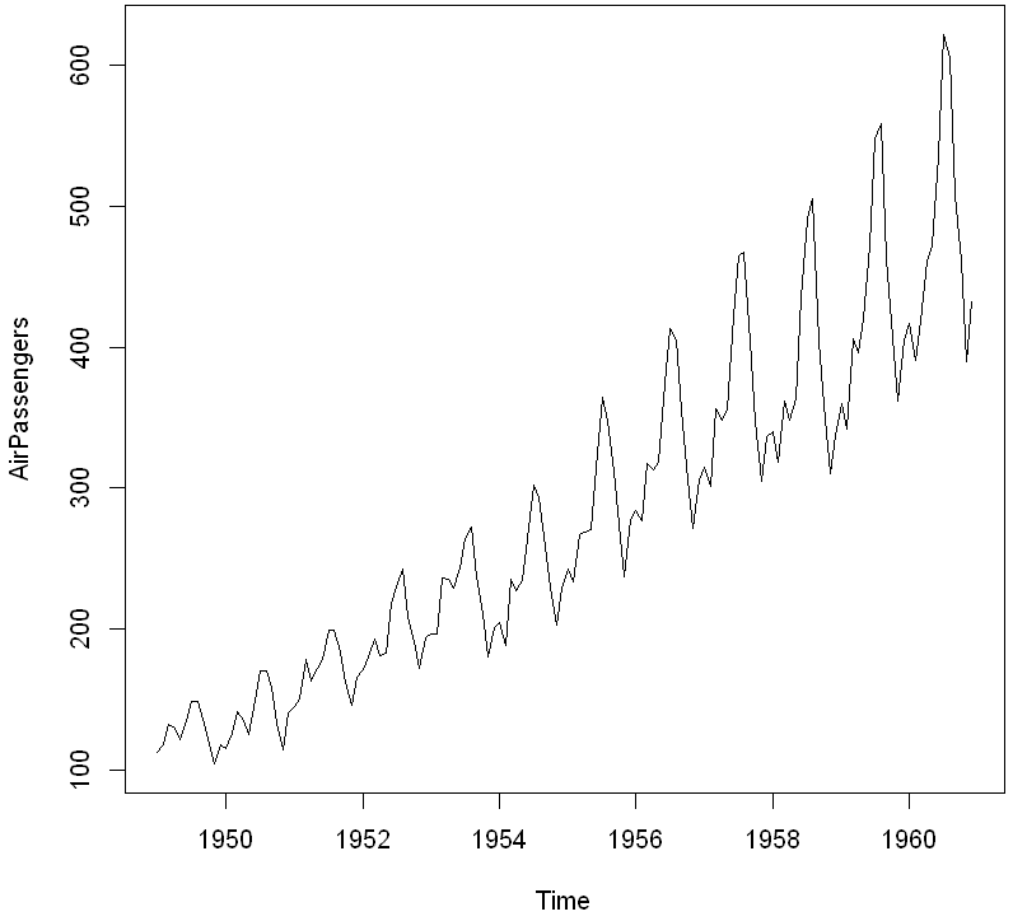
```
1 pairs(USArrests)
```



2. [時系列] `AirPassengers` は1949年から1960年にかけての月別国際航空旅客者数のデータである。

(1) plot()関数を用いて折れ線グラフを作りなさい

```
1 plot(AirPassengers)
```



(2) png()関数を用いて(1)で作ったグラフを保存しなさい

```
png('exercise/ex8_2_2.png', 400, 400)
plot(AirPassengers)
dev.off()
```

png: 2

3. [群間データ] `PlantGrowth` は植物の成長に関する実験データである。統制条件と 2 つの実験条件 の形三つの群(group)が用意され、植物の重量(weight)が測定された

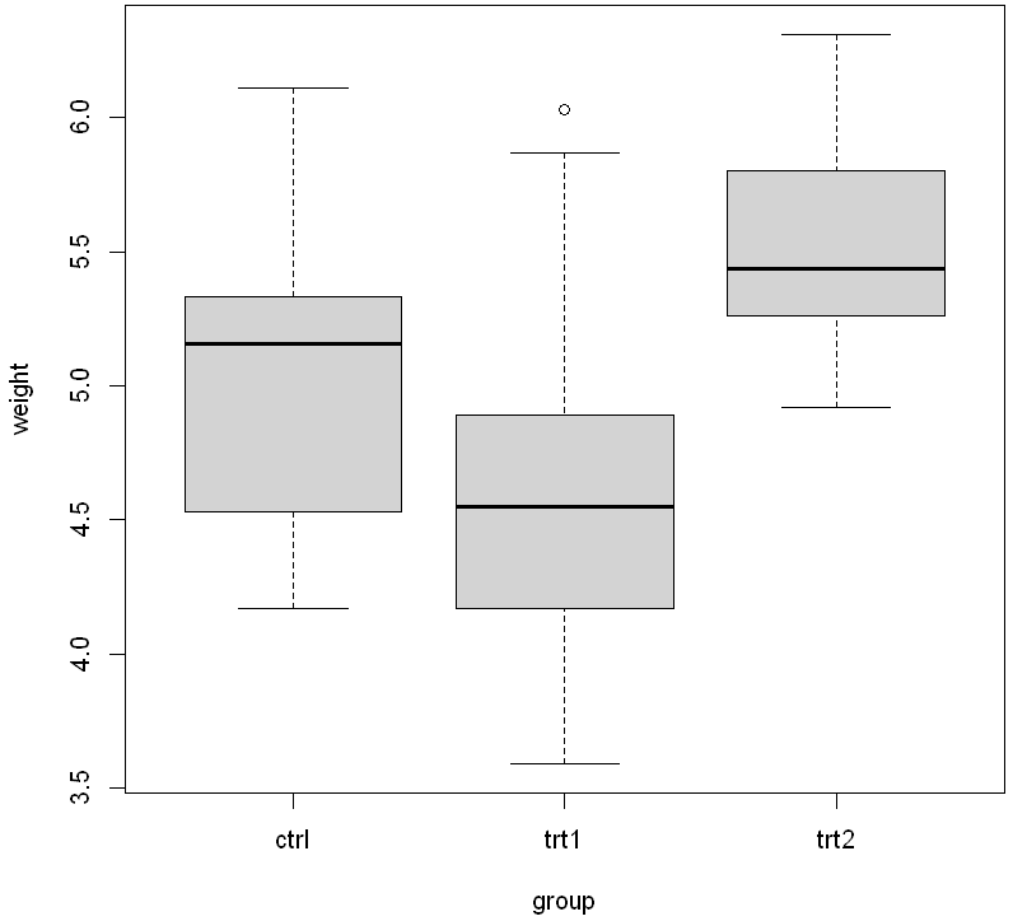
(1) table()関数を用いて各条件の標本数を求めなさい

```
table(PlantGrowth['group'])
```

```
ctrl trt1 trt2
  10   10   10
```

(2) 条件間を比較する箱ひげ図を作りなさい

```
boxplot(weight ~ group, data=PlantGrowth)
```



Ex.9 乱数

1. [sample()関数] 1から6の整数から均等な確率でひとつ数字を取り出す試行を繰り返すことを考える （ただし数字の重複は許す）。 sample()関数を使って均等な確率で1000回選び出し、以下の問に答えなさい（乱数なので答えは一定でない）

(1) 1が出た回数はいくつか

```
x = sample(1:6, 1000, replace=T)
```

```
length(x[x == 1])
# table(x) を使ってもよい
```

177

(2) 3以下の数字が出た回数はいくつか

```
length(x[x <= 3])
```

502

2. [runif()関数] runif()は0から1までの一様分布にしたがう乱数を出力できる関数である。 runif()関数を乱数を1000個生成し以下の問に答えなさい

(1) 0.2以下の数はいくつあるか

```
x = runif(1000)
```

```
length(x[x < 0.2])
```

201

```
length(x[x >= 0.3 & x <= 0.5])
```

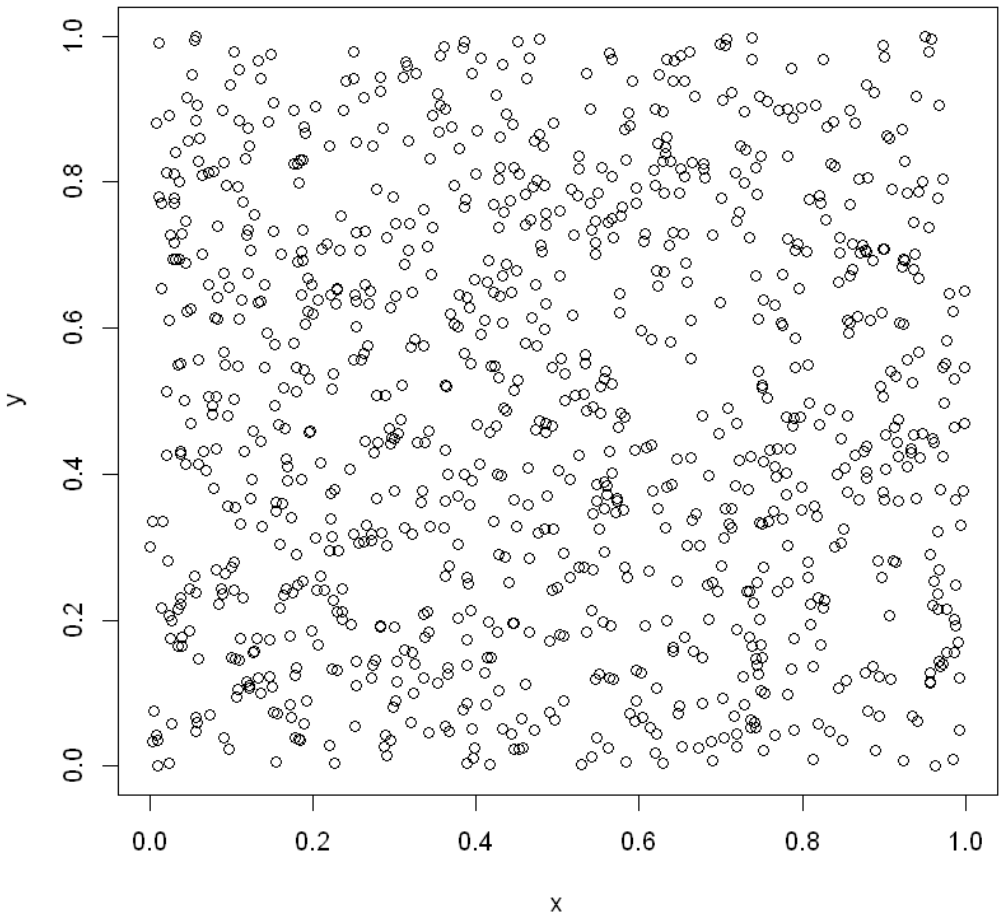
192

(2) 0.3以上、0.5以下の数はいくつあるか

3. ★[モンテカルロ・シミュレーション] runif()関数を使って近似的に円周率を求めたい。

(1) runif()関数を用いて、$0 \leq x \leq 1, 0 \leq y \leq 1$である点$(x, y)$をランダムに1000個生成し、散布図にプロットしなさい

```
x = runif(1000)
y = runif(1000)
plot(x, y)
```



(2) 原点$(0, 0)$からの距離が1以下である点の数を数えなさい。

```
# x^2 + y^2 <= 1であるものを数える
targets = (x^2 + y^2) <= 1
```

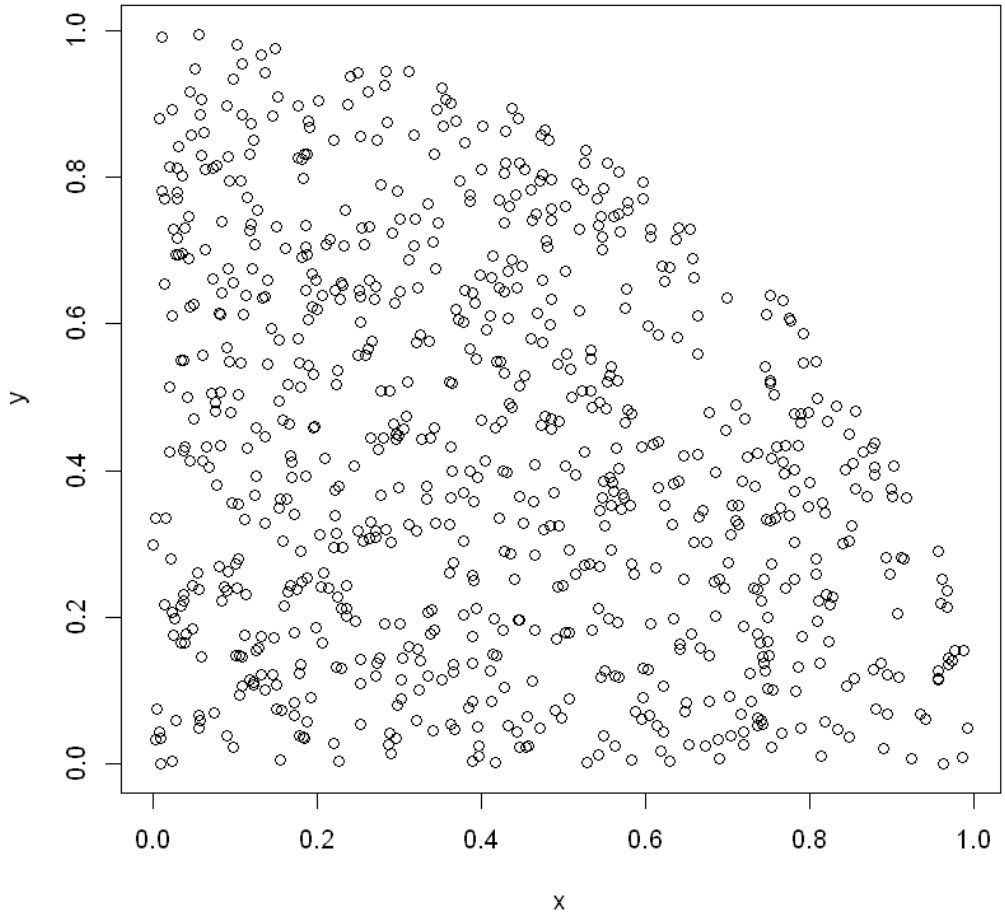
```
table(targets)
```

```
targets
FALSE  TRUE
  196   804
```

```
# 別解 TRUEが1, FALSEが0扱いであることを利用する
sum(targets)
```

804

```
plot(x[targets], y[targets], xlab='x', ylab='y')
```



(3) (1), (2)の結果から円周率の近似値を求めなさい。また、生成する点の数ををを1,000,000にしたらどうなるか試しなさい

```
# 点の総数をn, 原点(0, 0)からの距離が1以下の点の数をkとすると pi = 4*k/n
4 * sum(targets) / 1000
```

3.216

```
n = 1000000
x = runif(n)
y = runif(n)
4 * sum(x^2 + y^2 <= 1) / n
```

3.143192

- 考え方
  - ランダムに打たれた点のうち、原点$(0,0)$を中心とする半径1の円の中にある点の数をkとする。
  - このとき正方形と扇部分の面積比は$n:k$に等しい
  - 正方形の面積は1である
  - 円の面積はπr^2なので、扇部分の面積は$\frac{1}{4}\pi$である
  - $1:\frac{1}{4}\pi = n:k$の関係が成り立つので $\pi = \frac{4k}{n}$

Ex.10 dplyrによるデータハンドリング

1. `ToothGrowth`はビタミンCの用量と与え方がモルモットの歯の長さに与える影響を調べた実験データである。 これをデータセットとして用い、dplyrパッケージを読み込んだ上で以下の問に答えなさい

(1) パイプ演算子を使ってデータセットにhead()関数を適用しなさい

```
library('dplyr')
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
ToothGrowth %>% head()
```

A data.frame: 6 × 3

	len	supp	dose
	<dbl>	<fct>	<dbl>
1	4.2	VC	0.5
2	11.5	VC	0.5
3	7.3	VC	0.5
4	5.8	VC	0.5
5	6.4	VC	0.5
6	10.0	VC	0.5

(2) パイプ演算子とfilter()関数を使って以下の操作をしなさい
(2-1) `supp` が `OJ` であった個体を抽出する

```
ToothGrowth %>% filter(supp == 'OJ') %>% head()
```

A data.frame: 6 × 3

	len	supp	dose
	<dbl>	<fct>	<dbl>
1	15.2	OJ	0.5
2	21.5	OJ	0.5
3	17.6	OJ	0.5
4	9.7	OJ	0.5
5	14.5	OJ	0.5
6	10.0	OJ	0.5

(2-2) `supp` が `VC` で、 `dose` が2である個体を抽出する

```
ToothGrowth %>% filter(supp == 'VC', dose == 2) %>% head()
```

A data.frame: 6 × 3

	len	supp	dose
	<dbl>	<fct>	<dbl>
1	23.6	VC	2
2	18.5	VC	2
3	33.9	VC	2
4	25.5	VC	2
5	26.4	VC	2
6	32.5	VC	2

(2-3) `supp` が `VC` である個体、または `dose` が2である個体を抽出する

```
ToothGrowth %>% filter(supp == 'VC' | dose == 2)
```

A data.frame: 40 × 3

len	supp	dose
<dbl>	<fct>	<dbl>
4.2	VC	0.5
11.5	VC	0.5
7.3	VC	0.5
5.8	VC	0.5
6.4	VC	0.5
10.0	VC	0.5
11.2	VC	0.5
11.2	VC	0.5
5.2	VC	0.5
7.0	VC	0.5
16.5	VC	1.0
16.5	VC	1.0
15.2	VC	1.0
17.3	VC	1.0
22.5	VC	1.0
17.3	VC	1.0
13.6	VC	1.0
14.5	VC	1.0
18.8	VC	1.0
15.5	VC	1.0
23.6	VC	2.0
18.5	VC	2.0
33.9	VC	2.0
25.5	VC	2.0
26.4	VC	2.0
32.5	VC	2.0
26.7	VC	2.0
21.5	VC	2.0
23.3	VC	2.0
29.5	VC	2.0
25.5	OJ	2.0
26.4	OJ	2.0
22.4	OJ	2.0
24.5	OJ	2.0
24.8	OJ	2.0
30.9	OJ	2.0
26.4	OJ	2.0
27.3	OJ	2.0
29.4	OJ	2.0
23.0	OJ	2.0

(3) パイプ演算子とselect()関数を使って`len`列と`supp`列を抽出し、それぞれ「length」「supp_type」とリネームして表示しなさい

```
ToothGrowth %>% select(length=len, supp_type=supp) %>% head()
```

A data.frame: 6 × 2

	length	supp_type
	<dbl>	<fct>
1	4.2	VC
2	11.5	VC
3	7.3	VC
4	5.8	VC
5	6.4	VC
6	10.0	VC

(4) パイプ演算子とarrange()関数を使って、`len`列で昇順にソートした結果を表示しなさい

```
ToothGrowth %>% arrange(len) %>% head(10)
```

A data.frame: 10 × 3

	len	supp	dose
	<dbl>	<fct>	<dbl>
1	4.2	VC	0.5
2	5.2	VC	0.5
3	5.8	VC	0.5
4	6.4	VC	0.5
5	7.0	VC	0.5
6	7.3	VC	0.5
7	8.2	OJ	0.5
8	9.4	OJ	0.5
9	9.7	OJ	0.5
10	9.7	OJ	0.5

(5) パイプ演算子とsummarize()関数を使って以下の操作をしなさい

(5-1) `len`の最大値・最小値・中央値・平均をそれぞれ求める

```
ToothGrowth %>% summarise(max=max(len), min=min(len), med=median(len), mean=mean(len))
```

A data.frame: 1 × 4

max	min	med	mean
<dbl>	<dbl>	<dbl>	<dbl>
33.9	4.2	19.25	18.81333

(5-2) group_by()を`supp`列に適用し、群ごとの最大値・最小値・中央値・平均をそれぞれ求める

```
ToothGrowth %>% group_by(supp) %>% summarise(max=max(len), min=min(len), med=median(len), mean=mean(len))
```

```
`summarise()` ungrouping output (override with `.groups` argument)
```

A tibble: 2 × 5

supp	max	min	med	mean
<fct>	<dbl>	<dbl>	<dbl>	<dbl>
OJ	30.9	8.2	22.7	20.66333
VC	33.9	4.2	16.5	16.96333

Ex.11 stringrによる文字列操作

1. [基本操作] `month.name`は1年の月名のリストである。stringrを用いて以下の問に答えなさい

(1) str_subset()関数を用いて名前に'r'を含む月を抜き出しなさい

```
library('stringr')
```

```
month.name %>% str_subset('r')
```

'January' · 'February' · 'March' · 'April' · 'September' · 'October' · 'November' · 'December'

(2) str_detect()関数を用いて各月が名前に'r'を含むかどうかの論理ベクトルを得なさい

```
month.name %>% str_detect('r')
```

TRUE · TRUE · TRUE · TRUE · FALSE · FALSE · FALSE · FALSE · TRUE · TRUE · TRUE · TRUE

(3) str_length()関数を用いて各月の文字数を求めなさい

```
month.name %>% str_length()
```

7 · 8 · 5 · 5 · 3 · 4 · 4 · 6 · 9 · 7 · 8 · 8

(4) str_subset()関数は正規表現を扱える。正規表現を用いて以下の条件にあてはまる月をそれぞれ求めなさい

(4-1) 名前が'A'で始まる月

```
month.name %>% str_subset('^A')
```

'April' · 'August'

(4-2) 名前が'r'で終わる月

```
month.name %>% str_subset('r$')
```

'September' · 'October' · 'November' · 'December'

(4-3) 名前の2文字目が'a'である月

```
month.name %>% str_subset('^.a')
```

'January' · 'March' · 'May'

(4-4) 名前の最後から3番目の文字が'u'である月

```
month.name %>% str_subset('u..$')
```

'June' · 'July' · 'August'

(4-5) 名前の最初の文字が母音である月

```
month.name %>% str_subset('^[AIUEO]')
```

'April' · 'August' · 'October'

(4-6) 名前の最後の文字が'r'でも'y'でもない月

```
month.name %>% str_subset('[^ry]$')
```

'March' · 'April' · 'June' · 'August'

(4-7) 名前の末尾が'er'または'ry'で終わる月

```
month.name %>% str_subset('(er|ry)$')
```

'January' · 'February' · 'September' · 'October' · 'November' · 'December'

2. [正規表現による抽出]str_extract_all()関数は、文字列の中から正規表現にマッチした部分をすべて抜き出せる。
   sentence='comuptational social science'として以下の要素を抜き出しなさい

(1)単語(アルファベットの連続)

```
sentence = 'computational social science'
sentence %>% str_extract_all('[a-z]+')
```

1. 'computational' · 'social' · 'science'

(2) 's'で始まる単語

```
sentence %>% str_extract_all('s[a-z]*')
```

1. 'social' · 'science'

(3) 'al'で終わる単語

```
sentence %>% str_extract_all('[a-z]*al')
```

1. 'computational' · 'social'