クレジット:
UTokyo Online Education 統計データ解析 Ⅱ 2018 小池祐太

東京大学 THE UNIVERSITY OF TOKYO

CC BY NC ND

統計データ解析 II (平成３０年度)

東京大学大学院数理科学研究科
統計データ解析教育研究グループ

村田　昇 (早稲田大学, 東京大学)
吉田朋広 (東京大学)
小池祐太 (東京大学)

第 3 章 データの加工・整理と入出力

3.1. データの抽出

データから必要な部分集合を取り出すためには, 添え字を指定するのが最も基本的な方法である. 添え字の指定の仕方には, 番号を指定する以外に, 論理値で指定する方法がある. この場合, TRUE は要素の「選択」を, FALSE は要素の「除外」を意味する. また, 前にも述べたように, 要素に名前が付けられている場合は, その名前によってアクセス可能である. また, マイナス記号をつけて添え字番号を指定すると, その添え字番号の要素を除外する.

```
> ### ベクトルの例
> x <- c(4, 1, 2, 9, 8, 3, 6)
> x[c(5, 2)] # 5 番目と 2 番目の要素をこの順で抽出
[1] 8 1
> x[-c(2, 3, 7)] # 2,3,7 番目以外の要素を表示
[1] 4 9 8 3
> (idx <- x > 3) # 3 より大きい要素は TRUE, 3 以下の要素は FALSE
[1]  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE
> x[idx] # 3 より大きい要素をすべて表示
[1] 4 9 8 6
> x[x > 3] # 上と同じ
[1] 4 9 8 6
> x[-c(2, 3, 7)] # 2,3,7 番目以外の要素を表示
[1] 4 9 8 3
> x[c(2, 5)] <- c(0, 1) # 2 番目と 5 番目の要素を文字 0 と 1 に置換
> x
[1] 4 0 2 9 1 3 6
> (names(x) <- letters[1:length(x)]) # x の要素にアルファベットを順に名前をつける
[1] "a" "b" "c" "d" "e" "f" "g"
> x[c("b", "e")] # 2 番目と 5 番目の要素
b e
0 1
> ### データフレームの例
> ### R の組込みデータセット airquality を利用
> ### 詳細は help(airquality) 参照
> dim(airquality) # 大きさを確認
[1] 153   6
> names(airquality) # 列名を表示
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
> head(airquality) # 最初の 6 行を表示
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
```

```
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> str(airquality) # オブジェクトの構造を表示
'data.frame':        153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
> airquality[which(airquality$Ozone>100), ] # Ozone が 100 を超える行を抽出
    Ozone Solar.R Wind Temp Month Day
30    115     223  5.7   79     5  30
62    135     269  4.1   84     7   1
86    108     223  8.0   85     7  25
99    122     255  4.0   89     8   7
101   110     207  8.0   90     8   9
117   168     238  3.4   81     8  25
121   118     225  2.3   94     8  29
> airquality[which(airquality$Ozone>100), c("Month", "Day")]
    Month Day
30      5  30
62      7   1
86      7  25
99      8   7
101     8   9
117     8  25
121     8  29
> # Ozone が 100 を超える行の Month と Day を表示
                                                                (select.r)
```

データフレームから必要な部分集合を取り出す際に複雑な条件を指定する場合, 添え字を指定するのではコードが読みにくくなってしまう. そのような場合にも対応できるように, 関数 subset() が用意されている. 関数 subset() の基本的な書式は

```
subset(x, subset, select)
```

である. x にデータフレームを指定し, subset に抽出したい行に関する条件を, select に抽出したい列に関する条件をそれぞれ指定する.

```
> subset(airquality, Ozone>100) # Ozone が 100 を超える行を抽出
    Ozone Solar.R Wind Temp Month Day
30    115     223  5.7   79     5  30
62    135     269  4.1   84     7   1
86    108     223  8.0   85     7  25
99    122     255  4.0   89     8   7
101   110     207  8.0   90     8   9
117   168     238  3.4   81     8  25
121   118     225  2.3   94     8  29
> subset(airquality, Ozone>100, select=Wind:Day)
    Wind Temp Month Day
30   5.7   79     5  30
62   4.1   84     7   1
86   8.0   85     7  25
99   4.0   89     8   7
101  8.0   90     8   9
117  3.4   81     8  25
121  2.3   94     8  29
```

```
> # Ozone が 100 を超える行で列名が Wind〜Day のデータを抽出
> subset(airquality, !is.na(Ozone) & Day %in% c(1, 2))
    Ozone Solar.R Wind Temp Month Day
1      41     190  7.4   67     5   1
2      36     118  8.0   72     5   2
62    135     269  4.1   84     7   1
63     49     248  9.2   85     7   2
93     39      83  6.9   81     8   1
94      9      24 13.8   81     8   2
124    96     167  6.9   91     9   1
125    78     197  5.1   92     9   2
> # Ozone に欠測 (NA) がなく，Day が 1 か 2 のデータを抽出
> subset(airquality, Ozone>=120 | Temp >= 95)
    Ozone Solar.R Wind Temp Month Day
62    135     269  4.1   84     7   1
99    122     255  4.0   89     8   7
117   168     238  3.4   81     8  25
120    76     203  9.7   97     8  28
122    84     237  6.3   96     8  30
> # Ozone が 120 以上か Temp が 95 以上のデータ
> subset(airquality, Day == 1, select = -Temp)
    Ozone Solar.R Wind Month Day
1      41     190  7.4     5   1
32     NA     286  8.6     6   1
62    135     269  4.1     7   1
93     39      83  6.9     8   1
124    96     167  6.9     9   1
> # Day が 1 の行について Temp 以外の列を抽出
```

(subset.r)

その他, データフレームを特定のグループ分けに基づいて分割・結合するための関数関数 `split()`, 関数 `merge()` が用意されている (詳しい使い方はヘルプを参照). また, より高度なデータフレームの加工を実行するための関数群が `dplyr` パッケージに用意されている.

演習 3.1. R の組込データセット `airquality` (1973 年 5 月から 9 月までのニューヨークの大気の状態に関するデータ) から以下の条件を満たすデータを取り出せ.

(1) 7 月のオゾン濃度 (`Ozone`)
(2) 日射量 (`Solar.R`) に欠測 (`NA`) がないデータの月 (`Month`) と日 (`Day`)
(3) 風速 (`Wind`) が時速 10 マイル以上で, 気温 (`Temp`) が華氏 80 度以上の日のデータ

3.2. ファイルを用いたデータの読み書き

実際の解析の過程においては, 収集されたデータを読み込んだり, 整理したデータを保存したりする必要が生じる. R では一般に用いられる CSV 形式 (comma separated values) のテキストファイルと，R の内部表現を用いたバイナリーファイル (ここでは RData 形式と呼ぶ) をサポートしている．以下では，データフレームを対象として，それぞれの形式でファイルの読み書きを行うための関数を纏める.

3.2.1. 作業ディレクトリの確認と変更. R の実行は特定のフォルダ (ディレクトリ) 上で行われており, そのフォルダを**作業ディレクトリ**と呼ぶ. R のコード内でファイル名を指定した場合, 特に指定しない限り作業ディレクトリに存在するものとして扱われる. 現在の作業ディレクトリは関数 `getwd()` で確認できる. 作業ディレクトリの変更には関数 `setwd()` を利用するか, RStudio 上部の「Session」という項目から「Set Working Directory」を選び, その中の「Choose Directory...」という項目を選択すれば, 変更後のフォルダを選択できるようになる.

```
> getwd() # 作業ディレクトリの確認 (環境によって実行結果が異なるため，実行結果は省略)
> setwd("~/Documents")
> # ホームディレクトリ下の「書類」フォルダに移動
> # 環境によってディレクトリの指定の仕方が異なることに注意
```
(getwd.r)

3.2.2. CSV 形式の操作. 1 つのデータフレームを CSV 形式のファイルへ書き出すには，関数 `write.csv()` を用いる．書き出し後のファイルは特に指定しない限り作業ディレクトリ下に保存される．

```
> (mydata <- subset(airquality, Ozone > 90, select=-Temp)) # データフレームの作成
    Ozone Solar.R Wind Month Day
30    115     223  5.7     5  30
62    135     269  4.1     7   1
69     97     267  6.3     7   8
70     97     272  5.7     7   9
86    108     223  8.0     7  25
99    122     255  4.0     8   7
101   110     207  8.0     8   9
117   168     238  3.4     8  25
121   118     225  2.3     8  29
124    96     167  6.9     9   1
127    91     189  4.6     9   4
> dim(mydata) # 大きさを確認
[1] 11  5
> write.csv(mydata,file="mydata.csv") # csv ファイルとして書き出し
```
(data-write.csv.r)

CSV 形式のファイルから読み込むには，関数 `read.csv()` を用いる．読み込むファイルは，ディレクトリを明示的に指定しない限り，作業ディレクトリにある必要があることに注意．

```
> (newdata <- read.csv(file="mydata.csv",row.names=1)) # csv ファイルの読み込み
    Ozone Solar.R Wind Month Day
30    115     223  5.7     5  30
62    135     269  4.1     7   1
69     97     267  6.3     7   8
70     97     272  5.7     7   9
86    108     223  8.0     7  25
99    122     255  4.0     8   7
101   110     207  8.0     8   9
117   168     238  3.4     8  25
121   118     225  2.3     8  29
124    96     167  6.9     9   1
127    91     189  4.6     9   4
> dim(newdata) # 大きさを確認
[1] 11  5
> ## 外部 CSV データの読み込み
> ## 東京都の 2016 年の気候データによる例
> ## 気象庁のホームページより取得
> ## http://www.data.jma.go.jp/gmd/risk/obsdl/index.php
> ## 東京都の 2016 年の各日の平均気温 (℃)・降水量 (mm)・全天日射量 (MJ/u)・
> ## 平均風速 (m/s) を記録したデータセット kikou2016.csv
> kikou <- read.csv("kikou2016.csv",
```

```
+                   fileEncoding = "sjis") # ファイルの文字コードが Shift-JIS のため
> head(kikou) # データの最初の 6 行を表示
  月 日 気温 降水量 日射量 風速
1  1  1  7.5      0  11.80  2.6
2  1  2  7.3      0  11.59  1.9
3  1  3  9.3      0  10.77  1.4
4  1  4  9.2      0  11.19  1.6
5  1  5 10.9      0  10.57  1.8
6  1  6  8.9      0   4.54  1.9
> dim(kikou) # 大きさを確認
[1] 366   6
> colnames(kikou) # 列名を確認
[1] "月"     "日"     "気温"   "降水量" "日射量" "風速"
                                                                    (data-read.csv2.r)
```

オプションとして与えられている row.names=1 は，第 1 列を読み込んだデータフレームの各行の名前に割り当てることを意味している．

3.2.3. RData 形式の操作． RData 形式のファイルへの書き出しは，関数 save() を用いる．関数 write.csv() と同様に，書き出し後のファイルは特に指定しない限り作業ディレクトリ下に保存される．CSV 形式と異なり，複数のデータフレームを 1 つのファイルに同時に保存することもできる．

```
> (mydat1 <- subset(airquality, Temp>95, select=-Ozone)) # データフレームの作成
    Solar.R Wind Temp Month Day
120     203  9.7   97     8  28
122     237  6.3   96     8  30
> (mydat2 <- subset(airquality, Temp<60, select=-Ozone)) # データフレームの作成
   Solar.R Wind Temp Month Day
5       NA 14.3   56     5   5
8       99 13.8   59     5   8
15      65 13.2   58     5  15
18      78 18.4   57     5  18
21       8  9.7   59     5  21
25      66 16.6   57     5  25
26     266 14.9   58     5  26
27      NA  8.0   57     5  27
> dim(mydat1) # 大きさを確認
[1] 2 5
> dim(mydat2) # 大きさを確認
[1] 8 5
> save(mydat1,mydat2,file="mydata.rdata") # RData 形式で書き出し
                                                                    (data-save.r)
```

RData 形式のファイルからの読み込みは，関数 load() を用いる．関数 read.csv() と同様に，読み込むファイルは，ディレクトリを明示的に指定しない限り，作業ディレクトリにある必要があることに注意．

```
> (mydat1 <- subset(airquality, Ozone > 120)) # データフレームの作成
    Ozone Solar.R Wind Temp Month Day
62    135     269  4.1   84     7   1
99    122     255  4.0   89     8   7
117   168     238  3.4   81     8  25
```

```
> load(file="mydata.rdata") # RData 形式の読み込み
> mydat1 # save したときの名前で読み込まれ上書きされる
    Solar.R Wind Temp Month Day
120     203  9.7   97     8  28
122     237  6.3   96     8  30
> mydat2
   Solar.R Wind Temp Month Day
5       NA 14.3   56     5   5
8       99 13.8   59     5   8
15      65 13.2   58     5  15
18      78 18.4   57     5  18
21       8  9.7   59     5  21
25      66 16.6   57     5  25
26     266 14.9   58     5  26
27      NA  8.0   57     5  27
```

(data-load.r)

関数 `save()` では，データフレームの名前と内容が保存されるので，保存された名前が自動的に用いられる．したがって読み込む際には変数名の重複に注意が必要である．

演習 3.2. ファイル操作に慣れよう．

(1) 適当に作成したデータフレームをファイルに書き出しなさい．
(2) 表を整理するには Excel などの表計算ソフトを用いるのが簡便であり，多くの表計算ソフトは CSV 形式でもデータが保存できるようになっている．自身の利用するソフトにおいて CSV 形式で保存する方法を調べなさい．
(3) Excel 形式のファイルを直接読み込むパッケージもいくつかある．どのようなパッケージがあるか調べなさい．

3.3. 記述統計量によるデータの要約

データ解析の出発点は, 与えられたデータ全体の特徴や傾向を把握することである．そのための基本的な方法の 1 つは, データの特徴を適切に表す統計値を計算することである．そのような統計値を記述統計量, 要約統計量もしくは基本統計量と呼ぶ．

N 個のデータ $x_1, x_2, \ldots, x_N$ が与えられたとき, それらを代表する値として, **平均** *(mean)*

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{x_1 + x_2 + \cdots + x_N}{N}$$

が頻繁に利用される．平均は R では関数 `mean()` で計算できる．また, データのばらつき具合の指標として, **分散** *(variance)*

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \cdots + (x_N - \bar{x})^2}{N-1}$$

およびその平方根である**標準偏差** *(standard deviation)*

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

が広く利用されており, それぞれ関数 `var()` および関数 `sd()` で計算できる．[1]

[1] N ではなく $N-1$ で割る理由は不偏推定量を得るためである．詳細は「統計データ解析 I」の配布資料 7 参照 (`https://elf-c.he.u-tokyo.ac.jp/courses/244` よりダウンロード可能).

データの順位にもとづく記述統計量もよく利用される. 例えば, $x_1, \ldots, x_N$ の**最大値** *(maximum)* は関数 `max()` で, **最小値** *(minimum)* は関数 `min()` でそれぞれ計算できる. データを

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$$

のように昇順に並べ替えた際に中央の位置にくる値を**中央値** *(median)* または**メディアン**と呼ぶ. N が奇数の場合, 中央値は $x_{((N+1)/2)}$ であり, N が偶数の場合は $(x_{(N/2)} + x_{(N/2+1)})/2$ である. 中央値は関数 `median()` で計算できる.

中央値は平均と同様データを代表する値だと考えられるが, 平均と比較して, 計算結果がデータに含まれる異常な値 (**外れ値** *(outlier)* と呼ばれる) の影響を受けにくい.

中央値の一般化として, $\alpha \in [0, 1]$ に対して, その点以下のデータの個数が全体の**約** $100\alpha\%$になるような点を $100\alpha\%$**分位点** *(quantile)* と呼ぶ. 特に 25%分位点および 75%分位点をそれぞれ**第 1 四分位点**, **第 3 四分位点**と呼ぶ. **第 2 四分位点**は 50%分位点となるが, これは中央値のことである. ベクトル `x` の 100`alpha`%分位点は `quantile(x, alpha)` で計算できる. 分位点は一意的には定まらず, いくつかの計算方式がある: `help(quantile)` を参照すること.

```
> ### sleep データ (睡眠薬投与による睡眠時間の増減のデータ) による例
> ### 詳細は help(sleep) 参照
> (x <- sleep$extra)
 [1]  0.7 -1.6 -0.2 -1.2 -0.1  3.4  3.7  0.8  0.0  2.0  1.9  0.8  1.1  0.1 -0.1
[16]  4.4  5.5  1.6  4.6  3.4
> mean(x) # 平均
[1] 1.54
> var(x) # 分散
[1] 4.072
> sd(x) # 標準偏差
[1] 2.01792
> sqrt(var(x)) # sd(x) に一致
[1] 2.01792
> max(x) # 最大値
[1] 5.5
> min(x) # 最小値
[1] -1.6
> median(x) # 中央値
[1] 0.95
> quantile(x, 0.5) # 上と同じ
 50%
0.95
> quantile(x, c(0.25, 0.75)) # 第 1 四分位点および第 3 四分位点
   25%    75%
-0.025  3.400
> y <- c(x, 100) # データに外れ値を加えてみる
> mean(y)
[1] 6.228571
> median(y)
[1] 1.1
```

(descriptive.r)

データフレームが与えられた際には, 列 (あるいは行) ごとに記述統計量を計算したい状況が頻繁にある. そのような計算に便利な関数として関数 `apply()` がある. 関数 `apply()` は基本的に以下のような書式で利用する:

```
apply(X, MARGIN, FUN)
```

ここで, 引数 X にデータフレームを指定し, MARGIN には行ごとの計算には 1 を, 列ごとの計算には 2 を指定する. 引数 FUN には求めたい統計量を計算するための関数を指定する.

```
> ### airquality データを用いた例
> X <- airquality[ ,1:4] # 5-6 列目は月日なので除外する
> X <- na.omit(X) # 欠測 (NA) が含まれると計算できないため除外する
> apply(X, 2, mean) # 列ごとの平均
    Ozone   Solar.R      Wind      Temp
 42.09910 184.80180   9.93964  77.79279
> colMeans(X) # 上と同じ (この場合こちらの方が速い)
    Ozone   Solar.R      Wind      Temp
 42.09910 184.80180   9.93964  77.79279
> apply(X, 2, sd) # 列ごとの標準偏差
    Ozone   Solar.R      Wind      Temp
33.275969 91.152302  3.557713  9.529969
> apply(X, 2, median) # 列ごとの中央値
  Ozone Solar.R    Wind    Temp
   31.0   207.0     9.7    79.0
```

(apply.r)

複数のデータが与えられた場合, それらのデータの間の関係性を知りたい場合が頻繁に生じる. そのような目的のための最も基本的な記述統計量に**相関係数** *(correlation coefficient)* があり, これは 2 種類のデータ間の比例関係の大きさを計測する. 2 種類のデータ $x_1, x_2, \ldots, x_N$ および $y_1, y_2, \ldots, y_N$ に対して, それらの相関係数は

$$\rho = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{3.1}$$

で定義される. ここに, $\bar{x}$ および $\bar{y}$ はそれぞれ $x_1, x_2, \ldots, x_N$ および $y_1, y_2, \ldots, y_N$ の平均である. 相関係数は -1 以上 1 以下の値をとり, 1 に近いほど正の比例関係が強く, -1 に近いほど負の比例関係が強いことになる. なお, (3.1) の分子の統計量を $N-1$ で割ったものは**共分散** *(covariance)* と呼ばれる. 相関係数および共分散はそれぞれ関数 cor() および関数 cov() で計算できる. 2 種類のデータ x および y が与えられたとき, それらの相関係数は cor(x,y) で計算できる. 一方で, x がデータフレームのとき, cor(x) は (i, j) 成分が x の i 列と j 列の間の相関係数であるような行列 (**相関行列** *(correlation matrix)*) を計算する. 共分散についても同様である.

```
> ### sleep データによる例
> ### 2 種類の睡眠薬の効果の個人差に相関はあるか?
> x <- subset(sleep, group == 1, extra)
> y <- subset(sleep, group == 2, extra)
> cor(x, y)
          extra
extra 0.7951702
> ### データフレームの例: iris データ
> ### あやめのがく片 (Sepal) と花弁 (Petal) の長さ・幅,
> ### および品種 (Species) からなるデータフレーム
> ### 詳細は help(iris) 参照
> cor(iris[ ,1:4])
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

(cor.r)

演習 3.3. R の組込データセット state.x77 について, 列ごとの平均, 標準偏差, 最大値, 最小値および中央値を求めよ. また, 相関行列も求めよ.

3.4. データの可視化

記述統計量と並んでデータ全体の特徴や傾向を把握するために効果的な方法は, データの可視化である. R の基本パッケージ graphics に用意されている作図機能はきわめて多彩であり，これらを適切に組み合わせることによって様々な種類のグラフを描くことができる．以下では，いくつかの代表的な描画関数を取り上げて解説する．

描画関連の関数は色，線種や線の太さ，あるいは図中の文字の大きさなどを指定するために，多彩なオプションを用意しているので，必要に応じて関数 help()(ヘルプの表示) と example()(例題の表示) を利用して欲しい．

3.4.1. 基本的な描画. 描画において基本となるのは関数 plot() である．

関数 sin() のように 1 変数の関数として定義されているものは，定義域を指定してやればそのまま表示することができる．関数を追加するにはオプション add とともに関数 curve() を用いれば良い．

また，関数 plot() に同じ長さの二つのベクトルを与えると，同じ番号の要素からなる点の組 (x, y) をプロットして，その散布図を描くことができる．

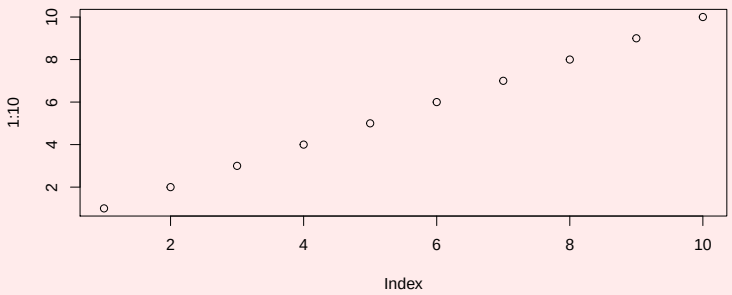
プロットの種類 (点や線) を指定するにはオプション type を用いる．'p' で点 (point)，'l' で点列を順に結んだ線 (line) が描かれる．なお，オプションに与える文字列は '(シングルクォート) か "(ダブルクォート) で囲む必要がある．

オプション col で"色の名前"を指定することにより点や線の色を変えることができる．R で指定することのできる色の名前は関数 colors() で照会することができる．

関数 plot() で描いた図中に更に線を追加するには関数 lines() を，点を追加するには関数 points() を用いる．

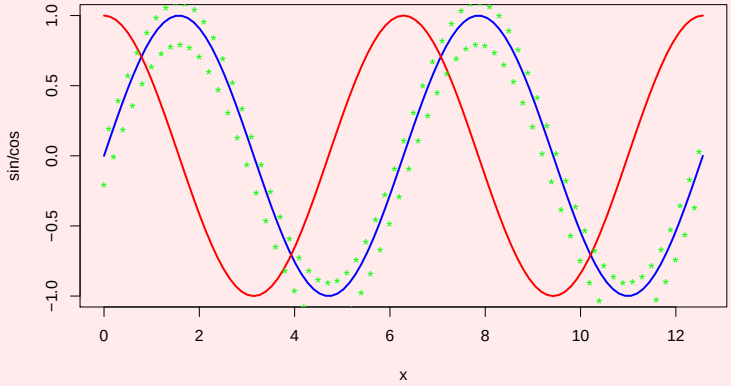
これ以外にも関数 plot() は様々なオプションを指定することができるので, help(plot) および help(plot.default) を参照して欲しい．

```
> ### ベクトルのプロット
> plot(1:10)
```

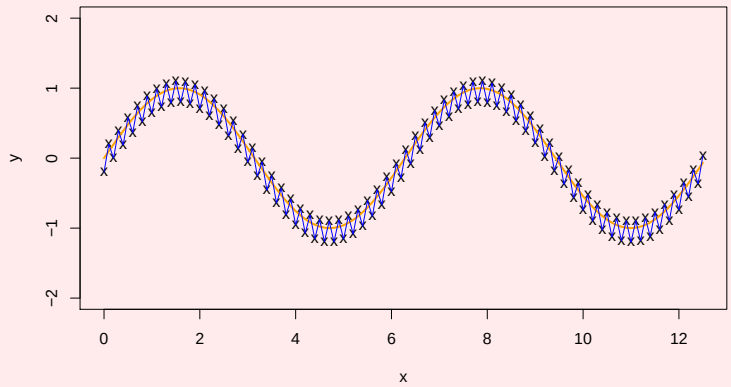



```
> ### 擬似データの作成
> x <- seq(0, 4*pi, by=0.1)
> y <- sin(x) + rep_len(c(-0.2, 0.1), length(x))
> ### 関数の描画
> plot(sin, 0, 4*pi,
+      col="blue", # グラフの線の色
+      lwd=2, # グラフの線の太さ
+      ylab="sin/cos" # y 軸のラベル
```

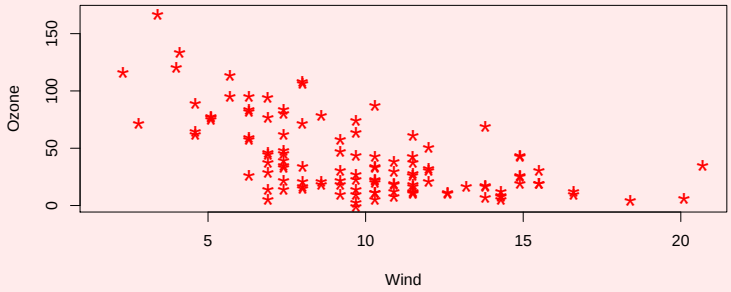
```
+       )
> curve(cos,
+         add=TRUE, # グラフを上書き
+         col="red", lwd=2)
> points(x, y, col="green", pch="*") # 点を追加. pch は点の形を指定
```




```
> ### (x,y) データの描画
> plot(x, y, type="p", pch="x", ylim=c(-2,2)) # ylim で値域を指定
> curve(sin, add=TRUE, col="orange", lwd=2)
> lines(x, y, col="blue") # 折れ線を追加
```



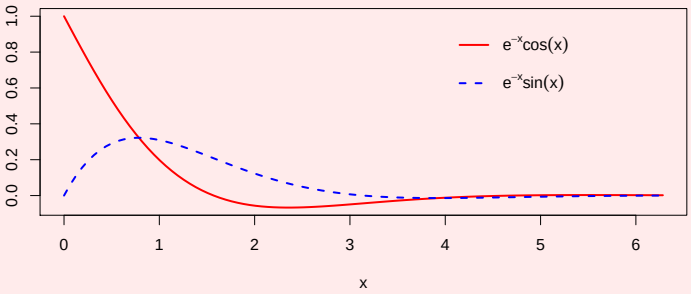

```
> ### データフレームを用いた散布図 (airquality を利用)
> plot(Ozone ~ Wind, data=airquality, pch="*", col="red", cex=2) # cex は点の大きさの倍率を指定
```



(plot3.r)

関数 legend() によってグラフに凡例を追加することができる. なお, 以下の例で見るように, R には数式を扱う機能がある. 詳細は help(plotmath) を参照してほしい.

```
> f <- function(x) exp(-x) * cos(x)
> plot(f, 0, 2 * pi, col = "red", lwd = 2, ylab = "")
> g <- function(x) exp(-x) * sin(x)
> curve(g, lty = 2, # グラフの線の形式. 2 はダッシュ線に対応
+       add = TRUE, col = "blue", lwd = 2)
> legend(4, # 凡例の左上の x 座標
+        1, # 凡例の左上の y 座標
+        legend = c(expression(e^{-x}*cos(x)), expression(e^{-x}*sin(x))), # 凡例
+        lty = c(1, 2), lwd = 2, col = c("red", "blue"), # ここのパラメーターは通常グラフと合わせる
+        bty = "n", # 凡例の枠線の形式 (オプション). "n"は書かない
+        y.intersp = 2 # 行間の指定 (オプション)
+        )
```



(legend.r)

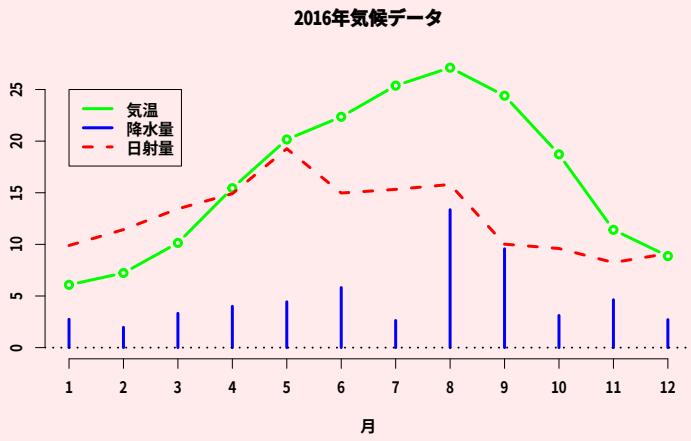
なお, OS によっては日本語を含む図を描画すると文字化けする場合がある. その場合, 関数 par() のオプション family に適当なフォントファミリーを指定することで文字化けを回避できる場合がある. 例えば, Mac OS のデフォルトの設定では日本語を含む図は文字化けしてしまうが, 以下のコマンドをコンソール上で実行することで文字化けを回避できる.

```
> par(family = "HiraginoSans-W4")
```

(family.r)

上の例ではフォントファミリーとしてヒラギノ角ゴシック W4 を指定している (数字を変えると太さが変わる).

```
> ### 東京都の 2016 年の気候データによる例
> ### 気象庁のホームページより取得
> ### http://www.data.jma.go.jp/gmd/risk/obsdl/index.php
> ### 東京都の 2016 年の各日の平均気温 (℃)・降水量 (mm)・全天日射量 (MJ/u)・
> ### 平均風速 (m/s) を記録したデータセット kikou2016.csv
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> ## 月ごとの平均をプロットする
> (x <- aggregate(kikou[ ,-c(1,2)], by = list(月 = kikou$月),
+                 FUN = "mean")) # 月ごとの平均を計算
   月       気温     降水量     日射量      風速
1   1  6.080645  2.741935  9.891290 2.393548
2   2  7.227586  1.965517 11.431034 2.889655
3   3 10.141935  3.322581 13.443226 2.812903
4   4 15.446667  4.000000 14.909667 3.263333
5   5 20.161290  4.435484 19.268065 3.383871
6   6 22.353333  5.816667 14.974000 2.926667
7   7 25.374194  2.629032 15.326129 2.674194
8   8 27.116129 13.354839 15.801935 3.096774
9   9 24.400000  9.566667 10.021000 2.436667
10 10 18.722581  3.112903  9.597742 2.441935
11 11 11.406667  4.633333  8.243000 2.466667
12 12  8.864516  2.709677  9.112581 2.641935
> plot(x$気温, type = "b", lwd = 3, col = "green", ylim = c(0, max(x$気温)),
+      xlab = "月", ylab = "", main = "2016 年気候データ", # グラフタイトル
+      axes = FALSE) # 軸を書かない
> axis(1, 1:12, 1:12) # x 軸の作成
> axis(2) # y 軸の作成
> lines(x$降水量, type = "h", lwd = 3, col = "blue")
> lines(x$日射量, lwd = 3, lty = 2, col = "red")
> abline(0, 0, lwd = 2, lty = "dotted") #  y=0 の線を引く
> legend(1, 25, legend = c("気温","降水量","日射量"),
+        col = c("green", "blue", "red"), lwd = 3,
+        lty = c(1, 1, 2))
```

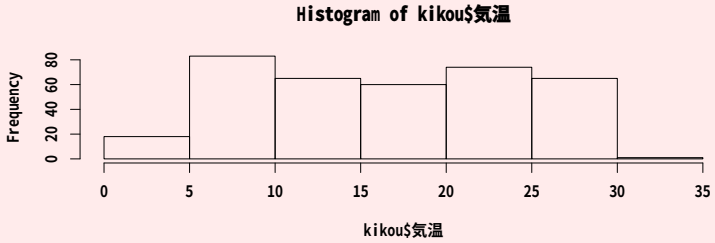


(plot-kion.r)

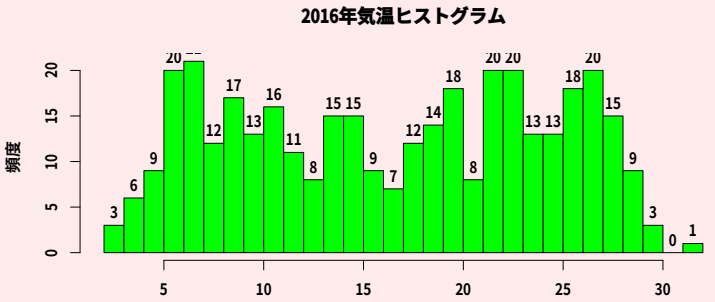
作成したグラフは保存することができる. RStudio の機能を使う場合, 右下ペインの「Plots」タブの「Export」をクリックすると, 形式やサイズを指定して保存できる (もしくはクリップボードにコピーもできる). コマンドで実行することも可能であるが, それについての詳細は `help(png)` や `help(dev.copy)` を参照してほしい.

3.4.2. ヒストグラム. データの頻度分布を表すヒストグラムを描画するには関数 hist() を用いる．これ以外にも凝ったヒストグラムを書くための関数がいくつか用意されているが，これらについては help.search("histogram") を参照して欲しい．

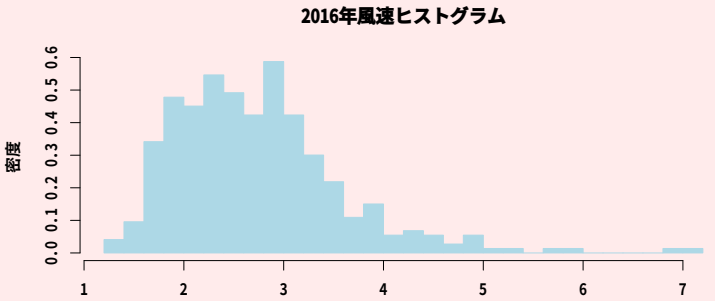
```
> ### 気候データによる例
> ### 基本的なヒストグラムの描画
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> hist(kikou$気温)
```



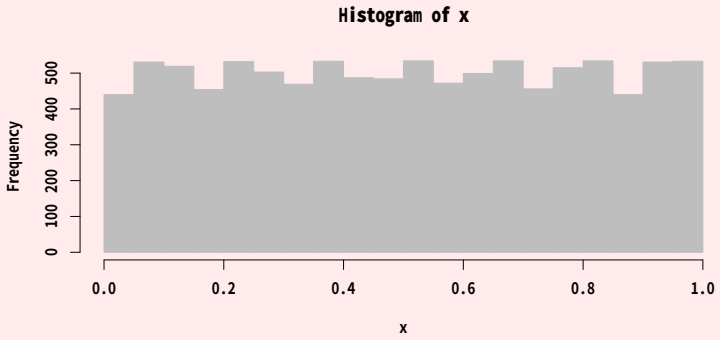

```
> hist(kikou$気温,
+      xlab = "", ylab = "頻度",
+      breaks=25, # ビンの数を約 25 に設定
+      labels = TRUE, # 各ビンの度数を表示
+      col = "green", main="2016 年気温ヒストグラム")
```




```
> hist(kikou$風速, freq = FALSE, # 全体に対する割合で表示
+      xlab = "", ylab = "密度", breaks=25, col = "lightblue",
+      border = "lightblue", # 長方形の境界の色
+      main="2016 年風速ヒストグラム")
```



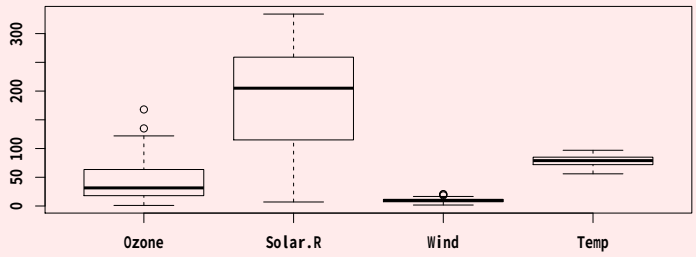
```
> ### Weyl の一様分布定理の確認
> ### a が無理数のとき，数列 a, 2a, 3a, ... の小数部分は
> ### 区間 (0,1) 上に均一に現れる
> a <- pi # 無理数
> n <- 10000
> x <- (1:n) * a
> x <- x - floor(x) # 小数部分の計算 (floor はいわゆる Gauss 記号)
> hist(x, breaks = 20, col = "gray", border = "gray")
```



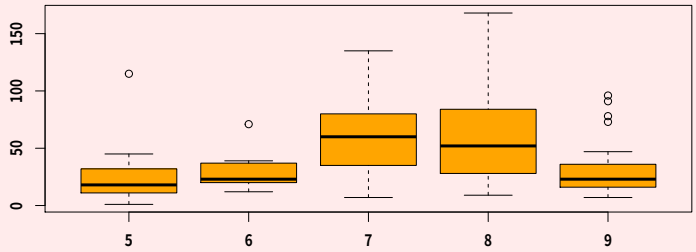
(hist3.r)

3.4.3. 箱ひげ図. 複数のデータの分布を比較する際，観測数が大きく異なるなどヒストグラムでの比較が難しい場合がある．複数のデータの分布の違いを簡便に見るには**箱ひげ図** *(boxplot)* が良く用いられるが，これは関数 `boxplot()` で描くことができる．

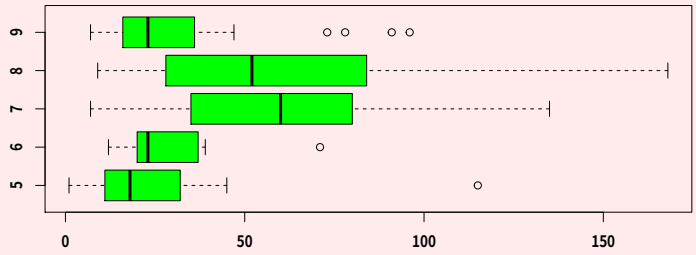
```
> ### データフレームを用いた表示例 (airquality を利用)
> boxplot(airquality[ ,1:4])
```




```
> ### Month ごとに Ozone を分類した場合
> boxplot(Ozone ~ Month, data=airquality, col="orange")
```



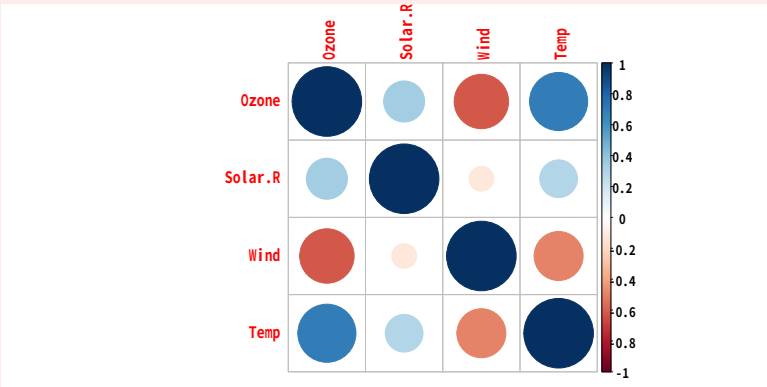

```
> boxplot(Ozone ~ Month, data=airquality, col="green", horizontal=TRUE)
```



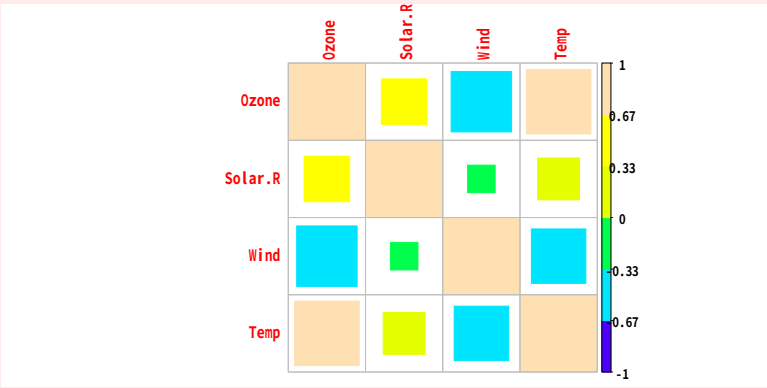
(boxplot2.r)

3.4.4. 相関行列の可視化. 列数が非常に多い大規模データフレームの変数間の相関の様子を見る場合, 相関行列の可視化が便利である. パッケージ `corrplot` には相関行列を可視化するための関数 `corrplot()` および関数 `corrplot.mixed()` が用意されている.

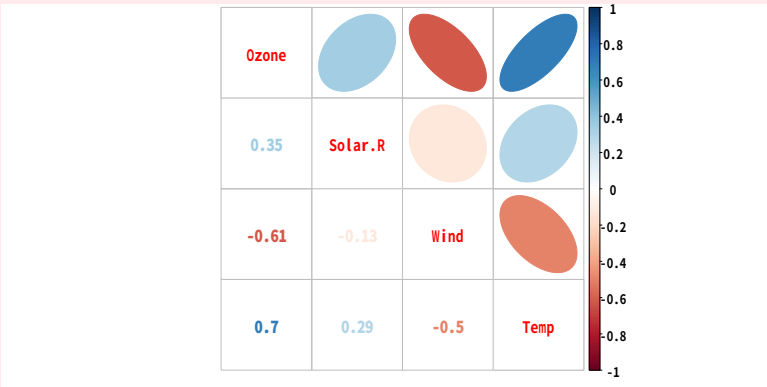
```
> ## 相関行列の可視化
> ## R の組込データセット airquality による例
> # install.packages("corrplot") # パッケージのインストール (必要な場合)
> library(corrplot) # パッケージのロード
> cor(airquality[ ,1:4]) # NA を含む列に関する結果は NA となる
        Ozone Solar.R       Wind       Temp
Ozone       1      NA         NA         NA
Solar.R    NA       1         NA         NA
Wind       NA      NA  1.0000000 -0.4579879
Temp       NA      NA -0.4579879  1.0000000
> (x <- cor(airquality[ ,1:4], use = "c")) # NA を含む行を除去して計算
             Ozone    Solar.R       Wind       Temp
Ozone    1.0000000  0.3483417 -0.6124966  0.6985414
Solar.R  0.3483417  1.0000000 -0.1271835  0.2940876
Wind    -0.6124966 -0.1271835  1.0000000 -0.4971897
Temp     0.6985414  0.2940876 -0.4971897  1.0000000
> corrplot(x) # 相関行列の可視化
```




```
> corrplot(x, method = "square", col = topo.colors(6)) # 色・形を変更
```




```
> corrplot.mixed(x, upper = "ellipse") # 下半分を数値，上半分を楕円
```

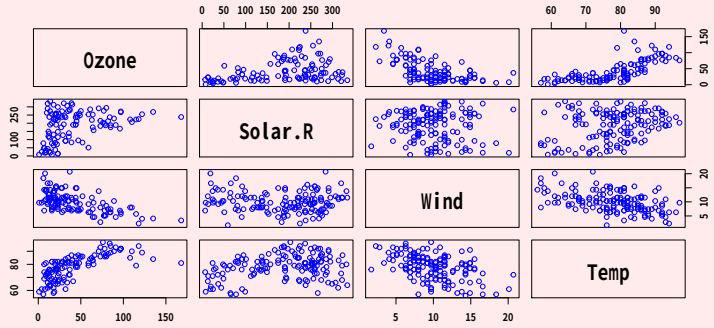


(corrplot.r)

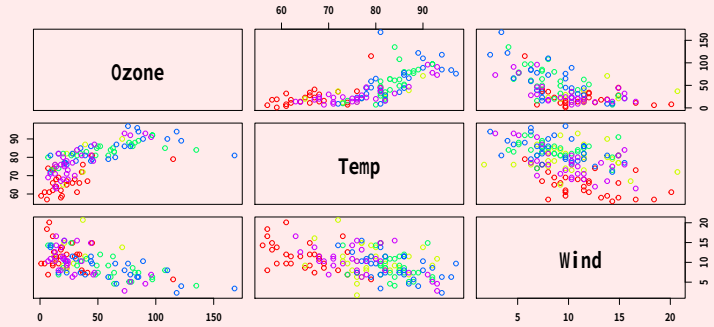
3.4.5. その他の描画関数. 多次元データの変数間の関係を概観するために，2 つの変数間の散布図を複数行列状に並べた図を用いることがある．これは関数 `pairs()` によって作成することができる．

一般的なパイチャートは関数 `pie()` で描くことができる．

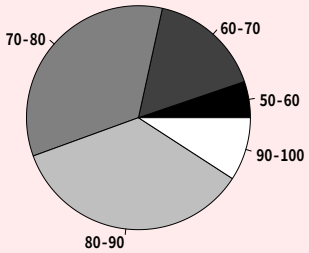
```
> ### 関数 pairs による散布図の作図
> pairs(airquality[,1:4], col="blue")
```



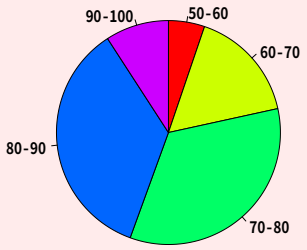

```
> pairs(~ Ozone + Temp + Wind, airquality,  # 表示する項目を指定
+ col=rainbow(5)[airquality$Month-4]) # 月毎に異なる色で表示
```




```
> ### 関数 pie によるパイチャートの作図
> z <- hist(airquality$Temp, breaks=5, plot=FALSE) # 5 つ程度に分類
> x <- z$count
> y <- z$breaks
> names(x) <- paste(y[-length(y)], y[-1], sep="-")
> pie(x, col=gray(seq(0,1,length=length(x))))
```




```
> pie(x, clockwise=TRUE, col=rainbow(length(x)))
```



(graphic-misc2.r)

R によるデータの可視化については,「統計データ解析 I」の配布資料 4 にてより詳細な説明を与えているので, 興味があればそちらも参照してほしい (https://elf-c.he.u-tokyo.ac.jp/courses/231 よりダウンロード可能).

3.5. 参考文献

1. 金明哲著「R によるデータサイエンス (第 2 版)」, 森北出版 (2017 年).
2. U. リゲス著, 石田基広訳「R の基礎とプログラミング技法」, 丸善出版 (2012 年).