クレジット:
UTokyo Online Education 統計データ解析 I 2017 小池祐太

ライセンス:
利用者は、本講義資料を、教育的な目的に限ってページ単位で利用することができます。特に記載のない限り、本講義資料はページ単位でクリエイティブ・コモンズ 表示−非営利−改変禁止 ライセンスの下に提供されています。
http://creativecommons.org/licenses/by-nc-nd/4.0/

本講義資料内には、東京大学が第三者より許諾を得て利用している画像等や、各種ライセンスによって提供されている画像等が含まれています。個々の画像等を本講義資料から切り離して利用することはできません。個々の画像等の利用については、それぞれの権利者の定めるところに従ってください。

東京大学 THE UNIVERSITY OF TOKYO

CC BY NC ND

統計データ解析Ⅰ (平成２９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 (首都大学東京, 東京大学)

UTokyo Online Education 統計データ解析Ⅰ 2017 小池祐太 CC BY-NC-ND

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
```


UTokyo Online Education 統計データ解析 I 2017 小池祐太 CC BY-NC-ND

```
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 `split()`, `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.

UTokyo Online Education 統計データ解析 I 2017 小池祐太 CC BY-NC-ND

```
> 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. データの整理

与えられたデータの総和や平均, 最大値・最小値を求めたい状況は頻繁にある．R にはこれらの操作を簡便に実行するための関数としてそれぞれ sum( ), mean( ), max( ), min( ) が用意されている．

```
> sum(1:100) # 1 から 100 までの整数の総和
[1] 5050
> ## 気候データによる例
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> kion <- kikou$気温 # 気温を取り出す
> mean(kion) # 平均
[1] 16.47022
> max(kion) # 最大値
[1] 31.9
> min(kion) # 最小値
[1] 2.8
```

(sum.r)

行列もしくはデータフレームが与えられた際には, 列 (あるいは行) ごとに平均などの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 appry( ) は基本的に以下のような書式で利用する:

```
appry(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．

なお, 総和や平均の場合には, 列/行ごとに計算するための専用の関数が用意されているため, そちらを利用した方が良い.

```
> x <- matrix(1:100, 4, 25)
> sum(x) # x の成分全ての和を計算する (mean 等も同様)
[1] 5050
> rowSums(x) # 行ごとの総和
[1] 1225 1250 1275 1300
> apply(x, 1, "sum") # 上と同じ
[1] 1225 1250 1275 1300
> ## 気候データによる例
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> x <- subset(kikou, select = -c(月, 日)) # 月日は除いておく
> colMeans(x) # 列ごとの平均
     気温     降水量     日射量       風速
16.470219  4.860656 12.681967  2.785246
> apply(x, 2, "max") # 列ごとの最大値
  気温 降水量 日射量   風速
 31.90 106.50  29.87   7.20
> sapply(x, "max") # 上と同じ
  気温 降水量 日射量   風速
 31.90 106.50  29.87   7.20
> apply(x, 2, "min") # 列ごとの最小値
  気温 降水量 日射量   風速
  2.80   0.00   1.11   1.20
> # 自作関数の適用
> apply(x, 2, function(x) sum(x > mean(x))) # 列ごとに平均より大きいデータの数を計算
  気温 降水量 日射量   風速
   188     72    157    172
```

(rowSums.r)

データフレームの各行をいくつかのグループにまとめて, グループごとの統計量を計算したい状況も頻繁に生じる. この場合に便利なのが関数 aggregate() である. 関数 aggregate() は基本的に以下のような書式で利用する:

```
aggregate(x, by, FUN)
```

ここで, 引数 x にデータフレームを指定し, by に各行が属するグループを指定するベクトルをリストで与える (複数可). 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. なお, x がベクトルの場合には関数 tapply() も利用可能である.

```
> ## 気候データによる例
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> ## 月ごとの各変数の平均値
> aggregate(kikou[ ,-(1:2)], by = list(月 = kikou$月), FUN = "mean")
   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
```

UTokyo Online Education 統計データ解析 I 2017 小池祐太 CC BY-NC-ND

```
11 11 11.406667  4.633333  8.243000 2.466667
12 12  8.864516  2.709677  9.112581 2.641935
> ## 以下のコードも同じ結果を返す
> aggregate( . ~ 月, data = subset(kikou, select = -日), FUN = "mean")
   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
> ## 月および降水の有無でグループ分け
> aggregate(kikou[ ,-(1:2)], FUN = "mean",
+           by = list(月 = kikou$月, 降水の有無 = (kikou$降水量 > 0)))
   月 降水の有無       気温     降水量     日射量      風速
1   1      FALSE  6.340741  0.000000 10.809259 2.351852
2   2      FALSE  6.747619  0.000000 12.790000 2.804762
3   3      FALSE  9.781818  0.000000 15.214091 2.654545
4   4      FALSE 16.041176  0.000000 20.606471 3.400000
5   5      FALSE 20.495652  0.000000 22.570870 3.260870
6   6      FALSE 22.550000  0.000000 21.742143 3.328571
7   7      FALSE 25.838095  0.000000 17.782857 2.700000
8   8      FALSE 28.000000  0.000000 20.305000 3.068750
9   9      FALSE 25.662500  0.000000 13.544375 2.462500
10 10      FALSE 18.761905  0.000000 11.958095 2.461905
11 11      FALSE 11.829412  0.000000 10.241176 2.411765
12 12      FALSE  7.969565  0.000000 10.114348 2.626087
13  1       TRUE  4.325000 21.250000  3.695000 2.675000
14  2       TRUE  8.487500  7.125000  7.863750 3.112500
15  3       TRUE 11.022222 11.444444  9.114444 3.200000
16  4       TRUE 14.669231  9.230769  7.460000 3.084615
17  5       TRUE 19.200000 17.187500  9.772500 3.737500
18  6       TRUE 22.181250 10.906250  9.051875 2.575000
19  7       TRUE 24.400000  8.150000 10.167000 2.620000
20  8       TRUE 26.173333 27.600000 10.998667 3.126667
21  9       TRUE 22.957143 20.500000  5.994286 2.407143
22 10       TRUE 18.640000  9.650000  4.641000 2.400000
23 11       TRUE 10.853846 10.692308  5.630000 2.538462
24 12       TRUE 11.437500 10.500000  6.232500 2.687500
```

(aggregate.r)

演習 3.3. R の組込データセット airquality について, 月日以外の変数ごとに平均, 最大値および最小値を求めよ. また, 月ごとの平均, 最大値および最小値も求めよ.

3.4. その他

CSV 形式でない通常のテキストファイルを読み込むための関数として read.table() がある. ファイルの大きさが大きい場合, 読み込みや書き出しに非常に時間がかかることがある. その場合, ファイル操作を高速化したパッケージ群がいくつか開発されているため, それらを利用するのが便利である. 例えば, 大規模データの読み込みにはパッケージ data.table の関数 fread() が便利である. CSV ファイルの書き出しにはパッケージ readr の関数 write_csv() が便利である.

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).

UTokyo Online Education 統計データ解析 I 2017 小池祐太 CC BY-NC-ND