データの整理と集計

村田 昇

2020.05.01

データフレームの操作

R に用意されているデータ構造

下記は代表的なもので，これ以外にもある

- ベクトル (vector)
- 行列 (matrix)
- リスト (list)
- **データフレーム (data frame)**
- 配列 (array)

データフレームからの項目の抽出

- 添字の番号を指定
- 要素の名前で指定
- **除外**: マイナス記号 (-) をつけて指定
- 論理値で指定
  - `TRUE`: 要素の選択
  - `FALSE`: 要素の **除外**

  (欠損値 `NA` が含まれると正しく指定できない場合があるので注意)

データ例

- `datasets::airquality` (R に準備されている)
  New York Air Quality Measurements
  - Description: Daily air quality measurements in New York, May to September 1973.
  - Format: A data frame with 153 observations on 6 variables.
    - [,1] Ozone numeric Ozone (ppb)
    - [,2] Solar.R numeric Solar R (lang)
    - [,3] Wind numeric Wind (mph)
    - [,4] Temp numeric Temperature (degrees F)
    - [,5] Month numeric Month (1–12)
    - [,6] Day numeric Day of month (1–31)
  - (`help(airquality)` または `?ariquality` で詳細を確認)

行の抽出 (1/3)

- 行番号による指定

```
## 抽出する行番号のベクトルで指定
airquality[1:10,] # 1-10 行を抽出
```

```
   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
7     23     299  8.6   65     5   7
8     19      99 13.8   59     5   8
9      8      19 20.1   61     5   9
10    NA     194  8.6   69     5  10
```

行の抽出 (2/3)

- 条件の指定

```
## 条件に合致する行は TRUE (NA は欠損値)
airquality[1:16,]$Ozone>100 # 条件の指定
airquality[1:16,]$Ozone>100 & airquality[1:16,]$Wind<=5 # 条件の AND
with(airquality[1:16,], Ozone>100 & Wind<=5) # 上と同じ (短い書き方)
with(airquality[1:24,], Ozone>100 | Wind<=5) # 条件の OR
```

```
 [1] FALSE FALSE FALSE FALSE    NA FALSE FALSE FALSE FALSE    NA FALSE FALSE
[13] FALSE FALSE FALSE FALSE
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE
 [1] FALSE FALSE FALSE FALSE    NA FALSE FALSE FALSE FALSE    NA FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

行の抽出 (3/3)

- 条件に合致する行番号の抽出

```
## 関数 which で TRUE の番号を抽出
which(with(airquality, Ozone>100 & Wind<=5)) # 全データから抽出
```

```
[1]  62  99 117 121
```

- 条件に合致する行の抽出

```
## 行の抽出
airquality[which(with(airquality, Ozone>100 & Wind<=5)), ]
```

```
    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
121   118     225  2.3   94     8  29
```

列の抽出 (1/2)

- 列番号による指定

```
## 列番号のベクトルで指定
airquality[which(with(airquality, Ozone>100 & Wind<=5)), c(1,5,6)]
```

```
    Ozone Month Day
62    135     7   1
99    122     8   7
117   168     8  25
121   118     8  29
```

列の抽出 (2/2)

- 列名による指定

```
## 複数の列の場合
airquality[which(with(airquality, Ozone>100 & Wind<=5)),
           c("Month","Day")]
```

```
    Month Day
62      7   1
99      8   7
117     8  25
121     8  29
```

```
## 1つの列の場合は以下でも良い（ただしベクトルになる）
airquality[which(with(airquality, Ozone>100 & Wind<=5)),]$Month
```

```
[1] 7 8 8 8
```

関数 subset()

複合的な条件を指定してデータを整理する

- 基本書式

```
subset(x,subset,select,drop=FALSE)
```

- 関数の引数
  - x: データフレーム
  - subset: 抽出する行の条件
  - select: 列の選択 (未指定の場合は全ての列)
  - drop: 結果が 1 行または 1 列の場合ベクトルとする (TRUE) かデータフレームとする (FALSE) か

関数 subset() の例 (1/3)

- 前出の例の書き換え

```
subset(airquality,
       subset = Ozone>100 & Wind<=5,
       select = c(1,5,6))
subset(airquality,
       Ozone>100 & Wind<=5, # 順序通りなら引数の名前は省略可
       c(Month,Day)) # 名前は$の後と同じ扱い
```

```
    Ozone Month Day
62    135     7   1
99    122     8   7
117   168     8  25
```

```
121   118     8  29
    Month Day
62      7   1
99      8   7
117     8  25
121     8  29
```

関数 subset() の例 (2/3)

- いろいろな記述の仕方

```
## Ozone に欠測 (NA) がなく，かつ Day が 5 か 10 の Wind から Day までの列を抽出
subset(airquality,
       subset = !is.na(Ozone) & Day %in% c(5,10),
       select = Wind:Day)
```

```
    Wind Temp Month Day
41  11.5   87     6  10
66   4.6   83     7   5
71   7.4   89     7  10
97   7.4   85     8   5
128  7.4   87     9   5
133  9.7   73     9  10
```

関数 subset() の例 (3/3)

- いろいろな記述の仕方

```
## Ozone が 120 以上か，または Wind が 3 以下の Temp 以外の列を抽出
subset(airquality,
       subset = Ozone>120 | Wind <= 3,
       select = -Temp)
```

```
    Ozone Solar.R Wind Month Day
53     NA      59  1.7     6  22
62    135     269  4.1     7   1
99    122     255  4.0     8   7
117   168     238  3.4     8  25
121   118     225  2.3     8  29
126    73     183  2.8     9   3
```

演習

練習問題

- datasets::airquality に対して以下の条件を満たすデータを取り出しなさい.
  - 7 月のオゾン濃度 (Ozone)
  - 風速 (Wind) が時速 10 マイル以上で，かつ気温 (Temp) が華氏 80 度以上の日のデータ
  - オゾン (Ozone) も日射量 (Solar.R) も欠測 (NA) でないデータの月 (Month) と日 (Day)

ファイルの取り扱い

データファイルの読み書き

- 実際の解析においては以下の操作が必要
  - 収集されたデータを読み込む

  - 整理したデータを保存する
- R で利用可能なデータファイル
  - CSV 形式 (comma separated values): テキストファイル
  - RData 形式: R の内部表現を用いたバイナリーファイル
  - (Excel 形式: RStudio の読み込み機能が利用可能)
- データフレームを対象とした扱いを整理する

作業ディレクトリ

- R は **作業ディレクトリ** で実行される
  - ファイルは作業ディレクトリに存在するものとして扱われる
  - それ以外のファイルを扱う場合はパスを含めて指定する
- 作業ディレクトリの確認の仕方
  - コンソールの上部の表示
  - 関数 `getwd()`
- 作業ディレクトリの変更の仕方
  - **Session** メニューの **Set Working Directory** で指定
    - 読み込んだファイルの場所を選択
    - File Pane の場所を選択
    - ディレクトリを直接選択
  - 関数 `setwd()`

関数 `getwd()/setwd()` の例

- コンソール / R Script からの作業ディレクトリの操作

```
## 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)
getwd()
## 作業ディレクトリの移動 (環境によって指定の仕方も異なる)
setwd("~/Documents") # ホームディレクトリ下の「書類」フォルダに移動
```

関数 `write.csv()`

データフレームを CSV ファイルへ書き出す

- 基本書式

```
write.csv(x, file="ファイル名")
```

- 関数の引数
  - `x`: 書き出すデータフレーム
  - `file`: 書き出すファイルの名前
    (作業ディレクトリ下，またはパスを指定)

関数 `write.csv()` の例

- CSV ファイルの書き出し

```
## 関数 write.csv の使い方
(myData <- subset(airquality,
                  subset = Ozone>120,
                  select = -Temp)) # データフレームの作成
dim(myData) # データフレームの大きさを確認
```

```
write.csv(myData,file="data/mydata.csv") # csv ファイルとして書き出し
```

```
    Ozone Solar.R Wind Month Day
62    135     269  4.1     7   1
99    122     255  4.0     8   7
117   168     238  3.4     8  25
[1] 3 5
```

関数 read.csv()

CSV ファイルからデータフレームを読み込む

- 基本書式

```
read.csv(file="ファイル名", header=TRUE,
         row.names, fileEncoding)
```

- 関数の引数
  - file: 読み込むファイルの名前
    (作業ディレクトリ下，またはパスを指定)
  - header: 1 行目を列名として使うか否か
  - row.names: 行名の指定
    (行名を含む列番号/列名，または行名の直接指定が可能)
  - fileEncoding: 文字コードの指定
    (日本語の場合，主に使うのは **“utf8”,“sjis”**)

関数 read.csv() の例

- CSV ファイルの読み込み

```
## 関数 read.csv の使い方
(newdata <- read.csv(file="data/mydata.csv",
                     row.names=1)) # 1 列目を行名に指定
dim(newdata) # 正しく読み込めたか大きさを確認
```

```
    Ozone Solar.R Wind Month Day
62    135     269  4.1     7   1
99    122     255  4.0     8   7
117   168     238  3.4     8  25
[1] 3 5
```

関数 save()

RData ファイルへ書き出す

- 基本書式

```
save(..., file="ファイル名")
```

- 関数の引数
  - ...: 保存するオブジェクト名
    (複数可，データフレーム以外も可)
  - file: 書き出すファイルの名前
    (作業ディレクトリ下，またはパスを指定)
- CSV 形式と異なり，**複数** のデータフレームを 1 つのファイルに保存することができる

関数 save() の例

- RData ファイルの書き出し

```
## 関数 save の使い方
(myDat1 <- subset(airquality, Temp>95, select=-Ozone))
(myDat2 <- subset(airquality, Temp<57, select=-Ozone))
dim(myDat1); dim(myDat2) # 大きさを確認
save(myDat1,myDat2,file="data/mydata.rdata") # RData 形式で書き出し
```

```
    Solar.R Wind Temp Month Day
120     203  9.7   97     8  28
122     237  6.3   96     8  30
  Solar.R Wind Temp Month Day
5      NA 14.3   56     5   5
[1] 2 5
[1] 1 5
```

関数 load()

RData ファイルから読み込む

- 基本書式

```
load(file="ファイル名")
```

- 関数の引数
  - file: 読み込むファイルの名前
    (作業ディレクトリ下，またはパスを指定)

関数 load() の例

```
## 関数 load の使い方
(myDat1 <- subset(airquality, Ozone > 160)) # 新たに作成
load(file="data/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
myDat2
```

```
    Ozone Solar.R Wind Temp Month Day
117   168     238  3.4   81     8  25
    Solar.R Wind Temp Month Day
120     203  9.7   97     8  28
122     237  6.3   96     8  30
  Solar.R Wind Temp Month Day
5      NA 14.3   56     5   5
```

演習

練習問題

- 以下のデータを読み込み，データの操作を行ってみよう．
  - データファイル (文字コード: utf8)
    - jpdata1.csv: 県別の対象データ
    - jpdata2.csv: 対象データの内容
    - jpdata3.csv: 県別と地域の対応関係
  - https://www.e-stat.go.jp より取得したデータ
    (地域から探す / 全県を選択 / 項目を選択してダウンロード)
  - 作業ディレクトリに置いて，以下のように読み込む

```
myData <- read.csv(file="data/jpdata1.csv", fileEncoding="utf8", row.names=1)
myItem <- read.csv(file="data/jpdata2.csv", fileEncoding="utf8")
myArea <- read.csv(file="data/jpdata3.csv", fileEncoding="utf8")
```

データの集計

集約のための関数

- データを集約するための基本的な関数は用意されている
  - 関数 sum(): 総和
  - 関数 mean(): 平均
  - 関数 max(): 最大値
  - 関数 min(): 最小値
- (これ以外にも集約を行なう関数は沢山ある)

関数の例

- 練習問題のデータの集計を行う

```
myData <- read.csv(file="data/jpdata1.csv",
                   row.names=1, fileEncoding="utf8")
## 一度読み込んでいれば上の行は不要
sum(myData$人口) # 全国の総人口（列名で選択）
mean(myData[,4]) # 面積の平均値（行列として列を選択）
median(myData[[4]]) # 面積の中央値（リストとして列を選択）
min(myData["若年"])  # 若年人口の最小値（列名で選択）
with(myData,max(老人))  # 老年人口の最大値（関数 with を利用）
```

```
[1] 126708000
[1] 793554.5
[1] 609719
[1] 72000
[1] 3160000
```

関数 apply()

列あるいは行ごとの計算を行う

- 基本書式

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

- 関数の引数
  - X: データフレーム
  - MARGIN: 行 (1) か列 (2) かを指定
  - FUN: 計算すべき統計量の関数
- 総和や平均は専用の関数も用意されている
  rowSums()/colSums(), rowMeans()/colMeans()

関数 apply() の例

- 抽出したデータの集計を行う

```
x <- subset(myData, select=婚姻:勤女) # 抽出
colMeans(x) # 各列の平均
apply(x, 2, max) # 列ごとの最大値
sapply(x, max)   # 上と同じ (help(sapply) を参照)
## 自作関数の適用 (関数に名前を付けずに利用できる)
apply(x, 2, function(z){sum(z>mean(z))}) # 平均より大きいデータ数
```

```
     婚姻      離婚      失業      勤男      勤女
4.437021  1.631064  4.221277 410.702128 296.659574
婚姻 離婚 失業 勤男 勤女
6.19  2.41  6.30 444.00 336.00
婚姻 離婚 失業 勤男 勤女
6.19  2.41  6.30 444.00 336.00
婚姻 離婚 失業 勤男 勤女
  20   22   25   27   22
```

関数 aggregate()

各行をグループにまとめて統計量を計算する

- 基本書式

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

- 関数の引数
  - x: データフレーム
  - by: 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)
  - FUN: 求めたい統計量を計算するための関数
- *(x* **がベクトルの場合には関数** *tapply()* **も利用可***)*

関数 aggregate() の例 (1/6)

- 同じ値を持つグループごとの平均値を求める

```
## 人口から面積まで地方ごとの平均値を計算
x <- subset(myData,select=人口:面積)
aggregate(x, by=list(地方=myArea$地方), FUN=mean)
```

```
    地方      人口      若年      老人      面積
1   関東 6178286 737000.0 1564000.0  463329.3
2   近畿 3204429 395714.3  898714.3  473223.6
3   九州 1795000 243875.0  511000.0  556395.0
4   四国  947000 112250.0  305750.0  470091.5
5   中国 1473800 186400.0  448600.0  638433.4
6   中部 2372889 302888.9  667555.6  742297.6
7   東北 1472667 169333.3  452666.7 1115790.7
8 北海道 5320000 588000.0 1632000.0 7842078.0
```

関数 aggregate() の例 (2/6)

- 代入せずにまとめて書くことも可能

```
aggregate(subset(myData,select=人口:面積),
          by=list(地方=myArea$地方),
          FUN=mean)
```

```
    地方      人口      若年       老人       面積
1   関東 6178286 737000.0 1564000.0  463329.3
2   近畿 3204429 395714.3  898714.3  473223.6
3   九州 1795000 243875.0  511000.0  556395.0
4   四国  947000 112250.0  305750.0  470091.5
5   中国 1473800 186400.0  448600.0  638433.4
6   中部 2372889 302888.9  667555.6  742297.6
7   東北 1472667 169333.3  452666.7 1115790.7
8 北海道 5320000 588000.0 1632000.0 7842078.0
```

関数 aggregate() の例 (3/6)

- 以下も同じ結果を返す

```
y <- data.frame(x, 地方=myArea$地方)
aggregate( . ~ 地方, data=y, FUN=mean)
```

```
    地方      人口      若年       老人       面積
1   関東 6178286 737000.0 1564000.0  463329.3
2   近畿 3204429 395714.3  898714.3  473223.6
3   九州 1795000 243875.0  511000.0  556395.0
4   四国  947000 112250.0  305750.0  470091.5
5   中国 1473800 186400.0  448600.0  638433.4
6   中部 2372889 302888.9  667555.6  742297.6
7   東北 1472667 169333.3  452666.7 1115790.7
8 北海道 5320000 588000.0 1632000.0 7842078.0
```

関数 aggregate() の例 (4/6)

- まとめて書くことも可能

```
aggregate( . ~ 地方, # 右辺で条件付けて左辺 (右辺以外) を計算
          data=data.frame(subset(myData,select=人口:面積),
                          地方=myArea$地方),
          FUN=mean)
```

```
    地方      人口      若年       老人       面積
1   関東 6178286 737000.0 1564000.0  463329.3
2   近畿 3204429 395714.3  898714.3  473223.6
3   九州 1795000 243875.0  511000.0  556395.0
4   四国  947000 112250.0  305750.0  470091.5
5   中国 1473800 186400.0  448600.0  638433.4
6   中部 2372889 302888.9  667555.6  742297.6
7   東北 1472667 169333.3  452666.7 1115790.7
8 北海道 5320000 588000.0 1632000.0 7842078.0
```

関数 aggregate() の例 (5/6)

- 複数の条件でグループ分け

```
## 地方と, 人口が中央値以下か否かでグループ分けして平均値を計算
aggregate(x, by=list(地方=myArea$地方,
                     過疎=with(myData, 人口<=median(人口))),
          FUN=mean)
```

```
    地方  過疎      人口      若年       老人       面積
1   関東 FALSE 6178285.7 737000.0 1564000.0  463329.3
2   近畿 FALSE 4681250.0 573750.0 1305500.0  517317.2
```

```
3    九州 FALSE 3436000.0 456000.0  957500.0  619800.0
4    中国 FALSE 2368000.0 305500.0  688000.0  779697.5
5    中部 FALSE 3510200.0 451400.0  973200.0  994346.8
6    東北 FALSE 2102500.0 250000.0  600000.0 1053306.0
7  北海道 FALSE 5320000.0 588000.0 1632000.0 7842078.0
8    近畿  TRUE 1235333.3 158333.3  356333.3  414432.0
9    九州  TRUE 1248000.0 173166.7  362166.7  535260.0
10   四国  TRUE  947000.0 112250.0  305750.0  470091.5
11   中国  TRUE  877666.7 107000.0  289000.0  544257.3
12   中部  TRUE  951250.0 117250.0  285500.0  427236.0
13   東北  TRUE 1157750.0 129000.0  379000.0 1147033.0
```

関数 aggregate() の例 (6/6)

- 別の書き方

```
aggregate( . ~ 地方 + 過疎, FUN=mean, # + で条件を追加
          data=data.frame(subset(myData,select=人口:面積),
                          地方=myArea$地方,
                          過疎=with(myData, 人口<=median(人口))))
```

```
    地方  過疎       人口      若年       老人       面積
1   関東 FALSE 6178285.7 737000.0 1564000.0  463329.3
2   近畿 FALSE 4681250.0 573750.0 1305500.0  517317.2
3   九州 FALSE 3436000.0 456000.0  957500.0  619800.0
4   中国 FALSE 2368000.0 305500.0  688000.0  779697.5
5   中部 FALSE 3510200.0 451400.0  973200.0  994346.8
6   東北 FALSE 2102500.0 250000.0  600000.0 1053306.0
7 北海道 FALSE 5320000.0 588000.0 1632000.0 7842078.0
8   近畿  TRUE 1235333.3 158333.3  356333.3  414432.0
9   九州  TRUE 1248000.0 173166.7  362166.7  535260.0
10  四国  TRUE  947000.0 112250.0  305750.0  470091.5
11  中国  TRUE  877666.7 107000.0  289000.0  544257.3
12  中部  TRUE  951250.0 117250.0  285500.0  427236.0
13  東北  TRUE 1157750.0 129000.0  379000.0 1147033.0
```

演習

練習問題

サンプルデータ (jpdata) の整理をしてみよう.

- 県別の人口密度を求めよ
- 地方別の人口密度を求めよ
  (県ごとに人口が異なるので単純に人口密度を平均してはいけない)
- 地方別の婚姻率・離婚率 (1000 人当たり) を概算せよ
  (「人口 1000 人当たり」とあるが，若年層は婚姻不可として除いた「婚姻可能な人口 1000 人当たり」で置き換えて計算しなさい)