ミクロ政治データ分析実習

第6回 データ構造

そん じぇひょん
宋 財 泫
関西大学総合情報学部
2021/5/20 (updated: 2021-05-13)

データ構造とは

データ構造とは

- **データ構造（Data Structure）**
  - データ型（Data Type）とは別概念: 先週の内容
- 1つ以上のベクトルで構成される。
  - ベクトルが1つのみだとベクトル
- **ベクトルの並び方**
  - 行列: 数値型「縦」ベクトルを横に並べる
  - データフレーム: 「縦」ベクトルを横に並べる
  - 配列: 行列を重ねる
  - リスト: ベクトル、データフレーム、配列、リストなどを並べる

データ構造の例

以下は代表的なデータ構造

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

ベクトル（復習）

ベクトル

同じデータ型が一つ以上格納されているオブジェクト

```
myVec1 <- "R is fun!"       # 長さ1のcharacter型ベクトル
myVec2 <- c(1, 3, 5, 6, 7) # 長さ5のnumeric型ベクトル
```

```
# 長さ6のベクトルであるが、2つのデータ型が混在しているため、
# character型が優先される
myVec3 <- c("A", "B", "C", 1, 2, 3)
myVec3
```

```
## [1] "A" "B" "C" "1" "2" "3"
```

```
# 長さ4のベクトルであるが、2つのデータ型が混在しているため、
# numeric型が優先される（TRUEは1に、FALSEは0に変換される）
myVec4 <- c(10, 20, TRUE, FALSE)
myVec4
```

```
## [1] 10 20  1  0
```

ベクトルの操作

データ型、長さなど

- `class(オブジェクト名)`: データ型
- `length(オブジェクト名)`: ベクトルの長さ（要素数）
- `nchar(オブジェクト名)`: Character型の場合、各要素の文字数

要素の抽出

- `オブジェクト名[n]`: `n`番目の要素を抽出
- `オブジェクト名[n:k]`: `n`番目から`k`番目の要素を抽出
- `オブジェクト名[c(i, j, k, ...)]`: `i`、`j`、`k`、...番目の要素を抽出
- `オブジェクト名[c(TRUE, FALSE, TRUE, ...)]`: `TRUE`に対応する位置の要素を抽出

ベクトルの演算

Numeric型ベクトルの演算

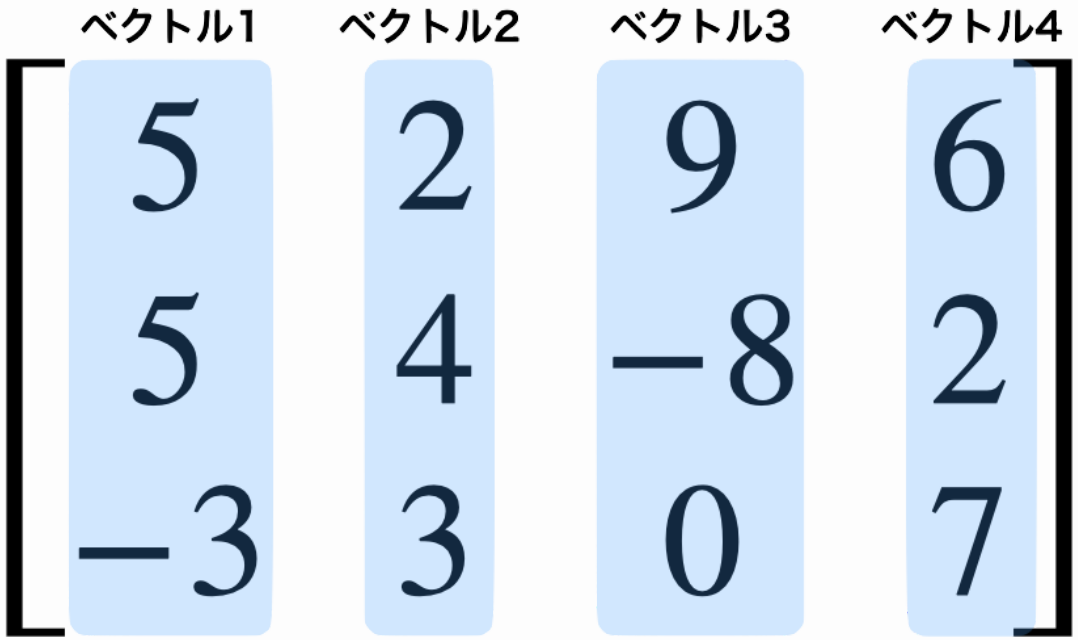
- ケース1: 同じ長さのベクトル同士
  - 同じ位置の要素同士の演算
- ケース2: 異なる長さのベクトル同士
  - 短い方のベクトルがリサイクルされる

行列と配列

行列

Numeric型、またはComplex型の縦ベクトルを横に並べたデータ構造

- 以下の例は3 × 4の行列
  - 長さ3のnumeric型**縦**ベクトルが4つ並んでいる模様
  - 長さ4のnumeric型**横**ベクトルが3つ積まれているとも読めるが、データ分析では一般的に縦ベクトルの集合として行列を捉える。



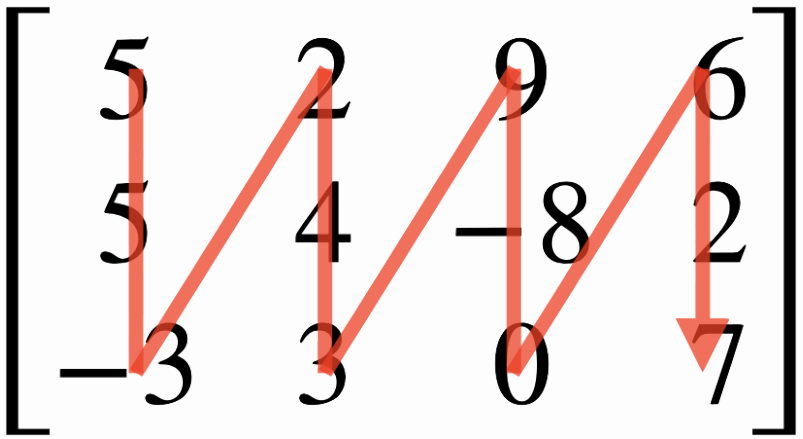
行列の作り方

`matrix()`関数を利用

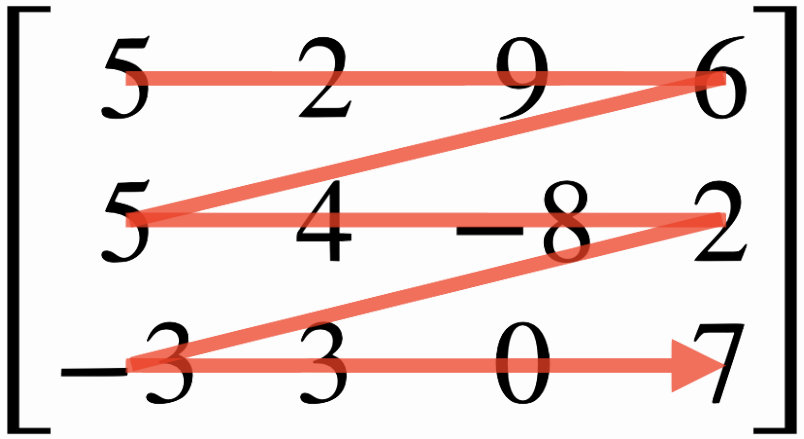
- $n \times m$ 行列を作成する場合、`nrow = n`と`ncol = m`を指定（片方のみでもOK）
- 第一引数であるベクトルは**縦方向**に入力
  - `byrow = TRUE`引数を追加すると、横方向入力
- $n \times n$ の単位行列の作成は`diag(n)`

```
matrix(numeric型ベクトル, nrow = n, ncol = m)
```

縦方向



横方向



行列の作り方

$$\begin{bmatrix} 5 & 2 & 9 & 6 \\ 5 & 4 & -8 & 2 \\ -3 & 3 & 0 & 7 \end{bmatrix}$$

行数の指定

```
My_Mat1 <- matrix(c(5, 5, -3,
                    2, 4, 3,
                    9, -8, 0,
                    6, 2, 7),
                  nrow = 3)

My_Mat1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    2    9    6
## [2,]    5    4   -8    2
## [3,]   -3    3    0    7
```

列数の指定

```
My_Mat2 <- matrix(c(5, 5, -3,
                    2, 4, 3,
                    9, -8, 0,
                    6, 2, 7),
                  ncol = 4)

My_Mat2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    2    9    6
## [2,]    5    4   -8    2
## [3,]   -3    3    0    7
```

行列の作り方

行列の要素を横方向で入力したい場合、`byrow = TRUE`を指定

横方向

```
My_Mat3 <- matrix(c(5, 2, 9, 6,
                    5, 4, -8, 2,
                    -3, 3, 0, 7),
                  nrow = 3, byrow = TRUE)

My_Mat3
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    2    9    6
## [2,]    5    4   -8    2
## [3,]   -3    3    0    7
```

行列の確認

データ構造の確認: `class()`

- データ構造がベクトルだとデータ**型**が、ベクトル以外だとデータ**構造**が出力される

```
class(My_Mat1) # 結果に"matrix"のみ出力されるとRバージョンが古い
```

```
## [1] "matrix" "array"
```

行列の大きさ: `dim()`、`nrow()`、`ncol()`

```
dim(My_Mat1) # 行列の行数と列数
```

```
## [1] 3 4
```

```
nrow(My_Mat1) # 行列の行数
```

```
## [1] 3
```

```
ncol(My_Mat1) # 行列の列数
```

```
## [1] 4
```

行列の操作

後期を含め、本講義では行列を扱う機会がほぼないため、解説はしない

要素の抽出

ベクトルの操作と基本的に同じであるが、2次元であるため、`[x, y]`のように指定する

- `x`、`y`の箇所はnumeric型かlogical型のベクトル

`n`行`m`列の要素

```
My_Mat1[2, 4] # 2行4列の要素
```

```
## [1] 2
```

`m`列

```
My_Mat1[, 2] # 2列の要素
```

```
## [1] 2 4 3
```

`n`行

```
My_Mat1[3, ] # 3行の要素
```

```
## [1] -3  3  0  7
```

`n`～`j`行（戻り値は行列）

```
My_Mat1[2:3, ] # 2～3行の要素
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    4   -8    2
## [2,]   -3    3    0    7
```

行列の足し算と引き算（１）

行列と長さ1のnumeric型ベクトル（スカラー）間の演算

- 行列の全要素に対してスカラーと足し算・引き算が行われる

```
My_Mat1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    2    9    6
## [2,]    5    4   -8    2
## [3,]   -3    3    0    7
```

足し算: `My_Mat1` の全要素に5を足す

```
My_Mat1 + 5
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   10    7   14   11
## [2,]   10    9   -3    7
## [3,]    2    8    5   12
```

引き算: `My_Mat1` の全要素に10を引く

```
My_Mat1 - 10
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -5   -8   -1   -4
## [2,]   -5   -6  -18   -8
## [3,]  -13   -7  -10   -3
```

行列の足し算と引き算（2）

行列同士の足し算と引き算は**同じ大きさの行列**のみ可能

```
My_Mat4 <- matrix(c(5, 5, -3, 2, 4, 3, 9, -8, 0), nrow = 3)
My_Mat5 <- matrix(c(-6, 5, 8, -8, -3, 4, -1, 4, 8), nrow = 3)
```

`My_Mat4`の中身

```
My_Mat4
```

```
##      [,1] [,2] [,3]
## [1,]    5    2    9
## [2,]    5    4   -8
## [3,]   -3    3    0
```

`My_Mat5`の中身

```
My_Mat5
```

```
##      [,1] [,2] [,3]
## [1,]   -6   -8   -1
## [2,]    5   -3    4
## [3,]    8    4    8
```

`My_Mat4` + `My_Mat5`

```
My_Mat4 + My_Mat5
```

```
##      [,1] [,2] [,3]
## [1,]   -1   -6    8
## [2,]   10    1   -4
## [3,]    5    7    8
```

`My_Mat4` - `My_Mat5`

```
My_Mat5 - My_Mat4
```

```
##      [,1] [,2] [,3]
## [1,]  -11  -10  -10
## [2,]    0   -7   12
## [3,]   11    1    8
```

行列の掛け算（１）

`*`演算子による掛け算は「アダマール積 (Hadamard product)」であり、**一般的に使われる掛け算ではない**ことに注意

- $A \times B$ でなく、 $A \odot B$ と表記する
- 同じ位置の要素同士の掛け算（2つの行列のサイズは同じ）
- スカラーとの掛け算だと、全要素に対して掛け算を行う

`My_Mat4` の中身

```
My_Mat4
```

```
##      [,1] [,2] [,3]
## [1,]    5    2    9
## [2,]    5    4   -8
## [3,]   -3    3    0
```

`My_Mat5` の中身

```
My_Mat5
```

```
##      [,1] [,2] [,3]
## [1,]   -6   -8   -1
## [2,]    5   -3    4
## [3,]    8    4    8
```

アダマール積

```
My_Mat4 * My_Mat5
```

```
##      [,1] [,2] [,3]
## [1,]  -30  -16   -9
## [2,]   25  -12  -32
## [3,]  -24   12    0
```

行列の掛け算（2）

一般的な行列の積は $n \times m$ 行列と $m \times p$ 行列同士で行われる

- `%*%` 演算子を使用
- 結果は $n \times p$ の行列
- $m \times p$ 行列と $n \times m$ 行列同士は出来ない
- 詳細は教科書第10.4章や線形代数の教科書、Googleなどで「行列 掛け算」で検索

```
# 2 x 3と3 x 3の積を求める
My_Mat6 <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
My_Mat7 <- matrix(c(2, 7, 17, 3, 11, 19, 5, 13, 23), nrow = 3)
```

`My_Mat6` の中身 (2×3)

```
My_Mat6
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

`My_Mat7` の中身 (3×3)

```
My_Mat7
```

```
##      [,1] [,2] [,3]
## [1,]    2    3    5
## [2,]    7   11   13
## [3,]   17   19   23
```

行列の積 (2×3)

```
My_Mat6 %*% My_Mat7
```

```
##      [,1] [,2] [,3]
## [1,]  108  131  159
## [2,]  134  164  200
```

転置行列、階数

転置行列: `t(オブジェクト名)`

```
My_Mat1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    2    9    6
## [2,]    5    4   -8    2
## [3,]   -3    3    0    7
```

```
t(My_Mat1)
```

```
##      [,1] [,2] [,3]
## [1,]    5    5   -3
## [2,]    2    4    3
## [3,]    9   -8    0
## [4,]    6    2    7
```

階数（Rank）: `qr(オブジェクト名)$rank`

- 1次方程式の場合、階数が行列の行数と一致する場合、一組の解が存在することを意味

```
qr(My_Mat1)$rank
```

```
## [1] 3
```

正方行列の操作

正方行列: $n \times n$ の行列

```
My_Mat6 <- matrix(c(2, -6, 4, 7, 2, 3, 8, 5, -1), nrow = 3)
My_Mat6
```

```
##      [,1] [,2] [,3]
## [1,]    2    7    8
## [2,]   -6    2    5
## [3,]    4    3   -1
```

行列式: `det()`

行列式 $\neq 0$だと1次方程式に1組の解が存在

```
det(My_Mat6)
```

```
## [1] -144
```

逆行列: `solve()`

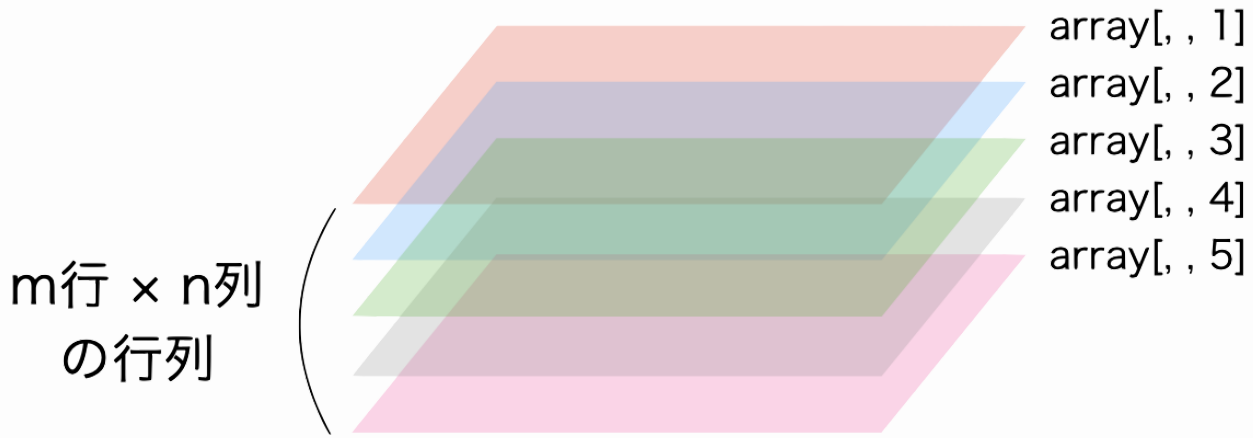
```
solve(My_Mat4)
```

```
##           [,1]     [,2]     [,3]
## [1,]  0.11806 -0.21528 -0.13194
## [2,] -0.09722  0.23611  0.40278
## [3,]  0.18056 -0.15278 -0.31944
```

配列 (array)

行列を重ねたもの

- 3次元のデータ構造であり、要素抽出の際、`[x, y, z]`で指定する必要がある。
  - 配列から行列を抽出したら、あとは行列の同じ操作
- 3番目の`z`が配列の層 (layer)を意味する
- 各層の行列の大きさは全て同じ
- 行列は層が1つのみの配列
- ほとんど使うことはないが、{rstan}でベイズ推定を行った場合、事後分布から抽出したサンプルは配列として出力される



データフレーム

データフレーム (data frame)

様々なデータ型の縦ベクトルを並べたもの

- cf) 行列: numeric型、またはcomplex型の縦ベクトル
- データフレームの拡張版として**tibble**というデータ構造もある（使い方はほぼ同じ）
  - R内蔵関数 `read.csv()` で表形式データを読み込む → data frame
  - {tidyverse}の `read_csv()` で表形式データを読み込む → tibble
- 例)Character型とNumeric型が混在
  - 各列は**変数**と呼ばれる
    - 4つの変数（Name, Foundation, Students, Faculties）
  - 各行は**ケース**と呼ばれる

Character型ベクトル（長さ4）　　**Numeric型ベクトル（長さ4）**

Name	Foundation	Students	Faculties
Kansai	1886	30141	740
Kwansei-gakuin	1889	24596	570
Doshisha	1875	30602	789
Ritsumeikan	1900	35113	1415

データフレームの作成（data.frame()）

`data.frame()`、または`tibble()`関数を使用

- `tibble()`を使用する前に予め{tidyverse}を読み込む

```
My_DF1 <- data.frame(Name       = c("Kansai", "Kwansei-gakuin",
                                    "Doshisha", "Ritsumeikan"),
                     Foundation = c(1886, 1889, 1875, 1900),
                     Students   = c(30141, 24596, 30602, 35113),
                     Faculties  = c(740, 570, 789, 1415))

My_DF1
```

```
##              Name Foundation Students Faculties
## 1          Kansai       1886    30141       740
## 2 Kwansei-gakuin       1889    24596       570
## 3        Doshisha       1875    30602       789
## 4    Ritsumeikan       1900    35113      1415
```

データフレームの作成 (tibble())

`tibble()` でデータフレームを作成すると表示形式が若干異なる

1. データ型も表示される
2. 横長のデータだと、画面に収まる分のみ表示される
3. 縦長のデータだと、最初の数十行のみ表示される
4. データ型を自動に判別してくれる

参考) `tibble()` で作成したデータフレーム

- `<dbl>` は数値型を意味する

```
## # A tibble: 4 x 4
##   Name            Foundation Students Faculties
##   <chr>                <dbl>    <dbl>     <dbl>
## 1 Kansai                1886    30141       740
## 2 Kwansei-gakuin        1889    24596       570
## 3 Doshisha              1875    30602       789
## 4 Ritsumeikan           1900    35113      1415
```

データフレームの読み込み

`.csv`や`.xlsx`などの表形式データを読み込む

- `read_csv()`や{readxl}の`read_excel()`などを使用
  - 第3回講義を参照すること（教科書第8章）
- 女子サッカーランキングデータの読み込み
  - 予め、授業サポートページからデータをダウンロードし、プロジェクト・フォルダーに保存すること
  - ここではプロジェクト・フォルダー内にDataというフォルダーを作成

```
library(tidyverse) # または、pacman::p_load(tidyverse)
My_DF2 <- read_csv("Data/Micro06.csv")
```

データフレームの確認（１）

```
My_DF2
```

```
## # A tibble: 159 x 6
##       ID Team                  Rank Points Prev_Points Confederation
##    <dbl> <chr>                <dbl>  <dbl>       <dbl> <chr>
##  1     1 Albania                 75   1325        1316 UEFA
##  2     2 Algeria                 85   1271        1271 CAF
##  3     3 American Samoa         133   1030        1030 OFC
##  4     4 Andorra                155    749         749 UEFA
##  5     5 Angola                 121   1117        1117 CAF
##  6     6 Antigua and Barbuda    153    787         787 CONCACAF
##  7     7 Argentina               32   1659        1659 CONMEBOL
##  8     8 Armenia                126   1103        1104 UEFA
##  9     9 Aruba                  157    724         724 CONCACAF
## 10    10 Australia                7   1963        1963 AFC
## # … with 149 more rows
```

データフレームの確認（2）

データ構造の確認

- `"data.frame"`が含まれていることを確認

```
class(My_DF2) # My_DF2のデータ構造
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

データフレームの大きさ

```
dim(My_DF2) # My_DF2の行数と列数
```

```
## [1] 159   6
```

```
nrow(My_DF2) # My_DF2の行数
```

```
## [1] 159
```

```
ncol(My_DF2) # My_DF2の列数
```

```
## [1] 6
```

データフレームの確認（3）

`head()`：**最初**の6行のみ出力

- `tail()`：**最後**の6行を出力
- `n = 5`を追加すると、最初の5行を出力（任意の数字）

```
head(My_DF2, n = 4) # My_DF2の最初の4行を出力
```

```
## # A tibble: 4 x 6
##      ID Team             Rank Points Prev_Points Confederation
##   <dbl> <chr>           <dbl>  <dbl>       <dbl> <chr>
## 1     1 Albania            75   1325        1316 UEFA
## 2     2 Algeria            85   1271        1271 CAF
## 3     3 American Samoa    133   1030        1030 OFC
## 4     4 Andorra           155    749         749 UEFA
```

`names()`：変数名のみ出力

- 変数が多い（横長）のデータを確認する時

```
names(My_DF2)
```

```
## [1] "ID"            "Team"          "Rank"          "Points"
## [5] "Prev_Points"   "Confederation"
```

要素の抽出（セル、行）

行列と同じ2次元データであるため、行列と同じ操作方法

- `n`行`m`列目のセルを抽出: `オブジェクト名[n, m]`

```
My_DF2[3, 2] # My_DF2の3行2列目を抽出
```

```
## # A tibble: 1 x 1
##   Team
##   <chr>
## 1 American Samoa
```

- `n`行目を抽出:: `オブジェクト名[n, ]`
  - `n:m`で`n`行〜`m`行の抽出も可能

```
My_DF2[3, ] # My_DF2の3行目を抽出
```

```
## # A tibble: 1 x 6
##      ID Team            Rank Points Prev_Points Confederation
##   <dbl> <chr>          <dbl>  <dbl>       <dbl> <chr>
## 1     3 American Samoa   133   1030        1030 OFC
```

要素の抽出（列）

列の抽出

方法1: 列の位置を指定

- `n:m`や`c(n, m, j...)`も可能

```
My_DF2[, 6]
```

```
## # A tibble: 159 x 1
##    Confederation
##    <chr>
##  1 UEFA
##  2 CAF
##  3 OFC
##  4 UEFA
##  5 CAF
##  6 CONCACAF
##  7 CONMEBOL
##  8 UEFA
##  9 CONCACAF
## 10 AFC
## # … with 149 more rows
```

方法2: 列の名前を指定

- `c("名前1", "名前2", ...)`も可能

```
My_DF2[, "Confederation"]
```

```
## # A tibble: 159 x 1
##    Confederation
##    <chr>
##  1 UEFA
##  2 CAF
##  3 OFC
##  4 UEFA
##  5 CAF
##  6 CONCACAF
##  7 CONMEBOL
##  8 UEFA
##  9 CONCACAF
## 10 AFC
## # … with 149 more rows
```

要素の抽出（列）

`$`を使う方法

- 戻り値はベクトル
- `オブジェクト名$変数名`で抽出
- 複数の行を同時に取り出すことは出来ない

```
class(My_DF2$Confederation)
```

```
## [1] "character"
```

```
My_DF2$Confederation # My_DF2からConfederationのみ抽出
```

```
##   [1] "UEFA"     "CAF"      "OFC"      "UEFA"     "CAF"      "CONCACAF"
##   [7] "CONMEBOL" "UEFA"     "CONCACAF" "AFC"      "UEFA"     "UEFA"
##  [13] "AFC"      "AFC"      "CONCACAF" "UEFA"     "UEFA"     "CONCACAF"
##  [19] "CONCACAF" "AFC"      "CONMEBOL" "UEFA"     "CAF"      "CONMEBOL"
##  [25] "UEFA"     "CAF"      "CONCACAF" "CONMEBOL" "AFC"      "AFC"
##  [31] "CONMEBOL" "CAF"      "CAF"      "CAF"      "OFC"      "CONCACAF"
##  [37] "CAF"      "UEFA"     "CONCACAF" "UEFA"     "UEFA"     "UEFA"
##  [43] "CONCACAF" "CONCACAF" "UEFA"     "CAF"      "UEFA"     "CAF"
##  [49] "CAF"      "UEFA"     "OFC"      "UEFA"     "UEFA"     "CAF"
##  [55] "CAF"      "UEFA"     "UEFA"     "CAF"      "UEFA"     "AFC"
```

抽出方法とデータ構造

- データフレームとtibbleの操作方法は**ほぼ**同じ
- ただし、抽出後のデータ構造が異なるため注意

抽出方法	data frame	tibble
1セルのみ	長さ1のベクトル	tibble
1行 (`[n, ]`)	データフレーム	tibble
複数行	データフレーム	tibble
1列 (`[, n]`)	ベクトル	tibble
1列 (`$`)	ベクトル	ベクトル
複数列	データフレーム	tibble

`class()`でデータフレームかtibbleかを確認

- データフレームの場合

```
## [1] "data.frame"
```

- tibbleの場合

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

論理演算子を用いた行の抽出

ベクトルと同様、`TRUE`、`FALSE`で抽出する行の指定が可能

```
# My_DF1から1, 3行のみを抽出
My_DF1[c(TRUE, FALSE, TRUE, FALSE), ]
```

```
##       Name Foundation Students Faculties
## 1   Kansai       1886    30141       740
## 3 Doshisha       1875    30602       789
```

```
# My_DF1からStudentsを抽出し、3万以上か否かを判定
My_DF1$Students >= 30000
```

```
## [1]  TRUE FALSE  TRUE  TRUE
```

```
# Studentsが3万以上の行を抽出
My_DF1[My_DF1$Students >= 30000, ]
```

```
##          Name Foundation Students Faculties
## 1      Kansai       1886    30141       740
## 3    Doshisha       1875    30602       789
## 4 Ritsumeikan       1900    35113      1415
```

列の追加

`オブジェクト名$新しい列名 <- ベクトル`

- **ベクトルの長さ**は**データフレームの行数**と同じ

```
# My_DF1にMainCampus列を追加し、各大学のメインキャンパス名を格納
My_DF1$MainCampus <- c("Senriyama", "Uegahara", "Imadegawa", "Kinugasa")
My_DF1
```

```
##              Name Foundation Students Faculties MainCampus
## 1          Kansai       1886    30141       740  Senriyama
## 2 Kwansei-gakuin       1889    24596       570   Uegahara
## 3        Doshisha       1875    30602       789  Imadegawa
## 4     Ritsumeikan       1900    35113      1415   Kinugasa
```

新しい列名でなく、既存の列名を指定すると**上書き**される

- `[]`を使って特定の要素のみ置換も可能

```
# My_DF1のMainCampus列の最初の要素（関大のメインキャンパス）を
#「Your Heart」に変更
My_DF1$MainCampus[1] <- c("Your Heart")
```

[参考] データフレームとtibbleの変換

{tidyverse}の`as_tibble()`とR内蔵関数`as.data.frame()`を使用

- `_`と`.`に注意

```
# My_DF1のデータ構造はデータフレーム
class(My_DF1)
```

```
## [1] "data.frame"
```

`as_tibble()`: tibbleへ変換

```
My_tbl1 <- as_tibble(My_DF1) # My_DF1をtibbleに
class(My_tbl1)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

`as.data.frame()`: データフレームへ変換

```
My_DF3 <- as.data.frame(My_tbl1) # My_tbl1をデータフレームに
class(My_DF3)
```

```
## [1] "data.frame"
```

リスト

リスト

あらゆるデータ構造を格納できるデータ構造

- リストの中にリストを入れることも可

リストの作成

```
My_List1 <- list(myVec3, My_Mat1, My_DF1)
My_List1
```

```
## [[1]]
## [1] "A" "B" "C" "1" "2" "3"
##
## [[2]]
##      [,1] [,2] [,3] [,4]
## [1,]    5    2    9    6
## [2,]    5    4   -8    2
## [3,]   -3    3    0    7
##
## [[3]]
##              Name Foundation Students Faculties MainCampus
## 1         Kansai       1886    30141       740  Senriyama
## 2 Kwansei-gakuin       1889    24596       570   Uegahara
## 3       Doshisha       1875    30602       789  Imadegawa
## 4    Ritsumeikan       1900    35113      1415   Kinugasa
```

各要素に名前が付いたリストの作成

`要素名 = 格納するオブジェクト名` で指定

```
My_List2 <- list(Vector = myVec3, Matrix = My_Mat1, DataFrame = My_DF1)
My_List2
```

```
## $Vector
## [1] "A" "B" "C" "1" "2" "3"
##
## $Matrix
##      [,1] [,2] [,3] [,4]
## [1,]    5    2    9    6
## [2,]    5    4   -8    2
## [3,]   -3    3    0    7
##
## $DataFrame
##              Name Foundation Students Faculties MainCampus
## 1          Kansai       1886    30141       740  Senriyama
## 2 Kwansei-gakuin       1889    24596       570   Uegahara
## 3        Doshisha       1875    30602       789  Imadegawa
## 4    Ritsumeikan       1900    35113      1415   Kinugasa
```

リストの確認

```
class(My_List2) # My_List2のデータ構造
```

```
## [1] "list"
```

```
length(My_List2) # My_List2の要素数
```

```
## [1] 3
```

`str()`を利用してリストの内部を簡単に確認

```
str(My_List2)
```

```
## List of 3
##  $ Vector   : chr [1:6] "A" "B" "C" "1" ...
##  $ Matrix   : num [1:3, 1:4] 5 5 -3 2 4 3 9 -8 0 6 ...
##  $ DataFrame:'data.frame':    4 obs. of  5 variables:
##   ..$ Name      : chr [1:4] "Kansai" "Kwansei-gakuin" "Doshisha" "Ritsumeikan"
##   ..$ Foundation: num [1:4] 1886 1889 1875 1900
##   ..$ Students  : num [1:4] 30141 24596 30602 35113
##   ..$ Faculties : num [1:4] 740 570 789 1415
##   ..$ MainCampus: chr [1:4] "Senriyama" "Uegahara" "Imadegawa" "Kinugasa"
```

要素の抽出（１）

抽出方法によって戻り値のデータ構造が異なる

1. `[[]]`**を利用した抽出**
   - 戻り値：要素のデータ構造
2. `[]`を利用した抽出
3. `$`を利用した抽出

要素の**位置**を指定

```
# My_List2の最初の要素
My_List2[[1]]
```

```
## [1] "A" "B" "C" "1" "2" "3"
```

```
class(My_List2[[1]])
```

```
## [1] "character"
```

要素の**名前**を指定

```
# My_List2の"Matrix"
My_List2[["Matrix"]]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    5    2    9    6
## [2,]    5    4   -8    2
## [3,]   -3    3    0    7
```

```
class(My_List2[["Matrix"]])
```

```
## [1] "matrix" "array"
```

要素の抽出（2）

抽出方法によって戻り値のデータ構造が異なる

1. `[[]]`を利用した抽出
2. **`[]`を利用した抽出**
    - 戻り値：リスト
3. `$`を利用した抽出

要素の**位置**を指定

```
# My_List2の最初の要素
My_List2[1]
```

```
## $Vector
## [1] "A" "B" "C" "1" "2" "3"
```

```
class(My_List2[[1]])
```

```
## [1] "character"
```

要素の**名前**を指定

```
# My_List2の"Matrix"
My_List2["Matrix"]
```

```
## $Matrix
##      [,1] [,2] [,3] [,4]
## [1,]    5    2    9    6
## [2,]    5    4   -8    2
## [3,]   -3    3    0    7
```

```
class(My_List2["Matrix"])
```

```
## [1] "list"
```

要素の抽出（3）

抽出方法によって戻り値のデータ構造が異なる

1. `[[]]`を利用した抽出
2. `[]`を利用した抽出
3. **`$`を利用した抽出**
    - 戻り値：要素のデータ構造
    - ただし、要素の名前が必要

```
# My_List2からDataFrameという名の要素を抽出
My_List2$DataFrame
```

```
##              Name Foundation Students Faculties MainCampus
## 1          Kansai       1886    30141       740  Senriyama
## 2 Kwansei-gakuin       1889    24596       570   Uegahara
## 3        Doshisha       1875    30602       789  Imadegawa
## 4     Ritsumeikan       1900    35113      1415   Kinugasa
```

```
class(My_List2$DataFrame)
```

```
## [1] "data.frame"
```

要素の中の要素

`[[]]`や`$`の後に`[]`や`$`が使用可能

- リストの中にリストがある場合も同様

```
# My_List2からVectorという名の要素を抽出し、更に1〜3番目の要素を抽出
My_List2[["Vector"]][1:3]
```

```
## [1] "A" "B" "C"
```

```
# My_List2からMatrixという名の要素を抽出し、更に2行目を抽出
My_List2[["Matrix"]][2, ]
```

```
## [1]  5  4 -8  2
```

```
# My_List2からDataFrameという名の要素を抽出し、更にNameという名の列を抽出
My_List2$DataFrame$Name
```

```
## [1] "Kansai"         "Kwansei-gakuin" "Doshisha"       "Ritsumeikan"
```

新しい要素の追加・置換

既にデータが入っている要素名や位置を指定すると上書きされる

1. `リスト名[["新しい要素名"]] <- 任意のデータ`
2. `リスト名[[要素の位置番号]] <- 任意のデータ`
3. `リスト名$新しい要素名 <- 任意のデータ`

```
# My_List2にMy_DF2を要素として追加し、SoccerRankingと名付ける
My_List2[["SoccerRanking"]] <- My_DF2
# My_List2に長さ3のベクトルを要素として追加し、Noodleと名付ける
My_List2$Noodle <- c("Ramen", "Udon", "Soba")
```

```
# My_List2内の要素数
length(My_List2)
```

```
## [1] 5
```

```
# My_List2内要素の名前を出力
names(My_List2)
```

```
## [1] "Vector"        "Matrix"        "DataFrame"     "SoccerRanking"
## [5] "Noodle"
```

まとめ

今回の内容

よく分からない箇所は教科書を読み返す or 宋&TAに質問 (できれば、LMSで)

- 表形式データの読み込み: 教科書第8章
- ベクトルの操作: 教科書 第7章と第9章
- その他のデータ構造の操作: 教科書 第10章

課題

1. 今回講義用のプロジェクトを作成する。
2. LMSから問題ファイル（`.Rmd`）とサンプルファイル（`.html`）をダウンロードし、プロジェクトのフォルダーに保存する。
3. **ファイル名は変更しない**こと
4. プロジェクトからRStudioを起動し、`.Rmd`ファイルを開く
    - NIIオンライン分析システムの場合、RStudio起動 > プロジェクトを開く > `.Rmd`ファイルを開く
5. 指示に従ってR Markdown文書を作成する。
6. 随時Knitし、結果を確認する。
    - Source Pane上段のKnitをクリックするか、「⌘ + Shift + K」(macOS)、「Ctrl + Shift + K」(Windows)を推す。
7. `.Rmd`ファイルを関大LMSに提出する。
8. **期限は2021年5月22日（土曜日）の23時59分とする。**
    - 時間に余裕を持って取り組むこと。期限直前に取り組み始めてPCトラブルがあっても期限延長はない
9. 答案は次回の講義までに公開する。