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

東京大学 THE UNIVERSITY OF TOKYO

CC BY NC ND

統計データ解析 (I) 第 4 回

小池祐太

2017 年 10 月 18 日

1. ファイルを用いたデータの読み書き
2. データの整理
3. データのプロット
   - 基本的な描画
   - ヒストグラム
   - 箱ひげ図

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

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, RStudio のコンソールの上部, もしくは

  ```
  getwd()
  ```

  で確認できる

作業ディレクトリの確認と変更

- 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
- 実行例 `getwd.r`

ファイルを用いたデータの読み書き

- 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
- 基本書式

```
write.csv(x, file = "")
```

  - `x`: 書き出したいデータフレーム
  - `file`: 書き出すファイルの名前
  - 他にも細かいオプションあり．ヘルプ参照
- ファイルの保存先は (指定しない限り) 作業ディレクトリとなる
- 実行例 `data-write.csv.r`

ファイルを用いたデータの読み書き

- CSV 形式のファイルから読み込むには，関数 `read.csv()` を用いる
- 基本書式

```
read.csv(file, header = TRUE, row.names)
```

  - `file`: 読み込みたいファイルの名前 (作業ディレクトリ下にある必要あり．もしくはディレクトリも指定)
  - `header`: ファイルの 1 行目をデータフレームの列名として使うか否か?
  - `row.names`: データフレームの行名を指定．(i) 行名を含む列番号/列名を指定, (ii) 行名の直接指定, というオプションがある．デフォルトでは行番号がそのまま行名になる．
  - 他にも細かいオプションあり．ヘルプ参照
- なお, より一般のテキストファイルを読み込むための関数として `read.table()`, `scan()` などがある

ファイルを用いたデータの読み書き

- 以降の講義の実行例で利用するデータ `kikou2016.csv` は, 以下の Web ページ

  `https://elf-c.he.u-tokyo.ac.jp/courses/228`

  からダウンロードできる
- 実行例 `data-read.csv2.r`

ファイルを用いたデータの読み書き

- RData 形式のファイルへの書き出しは，関数 `save()` を用いる
- 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
- 基本書式

```
save(..., file)
```

  - `...`: 保存したいオブジェクト名 (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

- ファイルの保存先は (指定しない限り) 作業ディレクトリとなる
- 実行例 `data-save.r`

ファイルを用いたデータの読み書き

- RData 形式のファイルからの読み込みは，関数 `load()` を用いる
- 基本書式

```
load(file)
```

  - `file`: 読み込みたいファイルの名前 (作業ディレクトリ下にある必要あり．もしくはディレクトリも指定)

- 実行例 `data-load.r`

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
- 実行例 `sum.r`

データの整理

- 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)
```

  - `X`: データフレーム
  - `MARGIN`: 行ごとの計算には 1 を, 列ごとの計算には 2 を指定
  - `FUN`: 求めたい統計量を計算するための関数
- 但し, 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
- 実行例 `rowSums.r`

データの整理

- 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)
```

  - `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
- なお, `x` がベクトルの場合には関数 `tapply()` も利用可能
- 実行例 `aggregate.r`

データの整理

演習 1

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. ただし, `NA` は除去して計算すること

データのプロット

- 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 にはきわめて多彩な作図機能が用意されており, ここ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おり, ここでは説明しきれないため, 必要に応じて関数 `help()`(ヘルプの表示) と `example()`(例題の表示) を参照

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
- 基本書式 (ベクトルの描画)

```
plot(x, type = "p", xlim = NULL, ylim = NULL, main =
    NULL, xlab = NULL, ylab = NULL, ...)
```

  - `x`: ベクトル
  - `type`: 描画タイプ. デフォルトは`"p"` (点プロット). 他に`"l"` (折れ線プロット) などがある
  - `xlim`: x 軸の範囲. デフォルトでは自動的に決定
  - `ylim`: y 軸の範囲. デフォルトでは自動的に決定
  - `main`: 図のタイトル. デフォルトではなし
  - `xlab`: x 軸のラベル名. デフォルトでは `Index` となる
  - `ylab`: y 軸のラベル名. デフォルトでは `x` のオブジェクト名
  - `...`: 他のオプション. 詳細は `help(par)` 参照

基本的な描画

- よく利用される plot のオプション
  - col: 描画するデータの色の指定．"red"や"blue"など．指定することのできる色の名前は関数 colors() で照会できる
  - pch: 描画される点の形．数字で指定．詳細は help(points) 参照
  - cex: 描画される文字の大きさ．デフォルトの何倍にするかで指定
  - lty: 描画される線のタイプ．実線, 破線など．タイプ名もしくは数字で指定．詳細は help(par) 参照
  - lwd: 描画される線の太さ．数字で指定
- ベクトル x に対して plot(x) を実行すれば, 横軸に成分番号, 縦軸に各成分を描画した点プロットが作成される
- 実行例 plot3.r

基本的な描画 (関数)

- 1 変数関数の描画も関数 `plot()` で可能
- 基本書式 (1 変数関数の描画)

  ```
  plot(x, y = 0, to = 1, ...)
  ```

  - `x`: 1 変数関数
  - `y`: x 軸の左端
  - `to`: x 軸の右端
  - `...`: “ベクトルの描画” と同じオプションが利用可能
- 別の関数 `f` を重ね書きをしたい場合,

  ```
  curve(f, add = TRUE, ...)
  ```

  を実行してやればよい (... には “ベクトルの描画” と同じオプションが利用可能)
- 実行例 `plot3.r`

基本的な描画 (散布図)

- 2 種類のデータ $x_1, \ldots, x_N$ および $y_1, \ldots, y_N$ が与えられたとき，点 $(x_1, y_1), \ldots, (x_N, y_N)$ を平面上に描画した図を**散布図**と呼ぶ
- 散布図も関数 `plot()` で作成できる
- 基本書式 (散布図)

```
plot(x, y = NULL, ...)
```

  - `x`: 1 種類目のデータ $x_1, \ldots, x_N$
  - `y`: 2 種類目のデータ $y_1, \ldots, y_N$
  - `...`: “ベクトルの描画” と同じオプションが利用可能

基本的な描画 (散布図)

- また, データフレーム x の変数 A と変数 B に関して散布図を作成したい場合, コマンド

  ```
  plot(B ~ A, data = x, ...)
  ```

  も利用できる
- 実行例 plot3.r

基本的な描画

演習 2

R の組込データセット sleep データ (睡眠薬投与による睡眠時間の増減のデータ・詳細は help(sleep) 参照) において, group が 1 のデータの extra を x 軸, group が 2 のデータの extra を y 軸とした散布図を描画せよ. ただし, 点の色は青, 点の形は $\times$ とし, タイトルを Sleep data, x 軸のラベルを group 1, y 軸のラベルを group 2 とせよ (適宜インターネット上の情報を参照せよ. また, x 軸, y 軸のデータはそれぞれ

```
x <- subset(sleep, group == 1, extra, drop = TRUE)
y <- subset(sleep, group == 2, extra, drop = TRUE)
```

としてつくるとよい)

基本的な描画 (凡例)

- 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, 以下の実行例で見るように, R には数式を扱う機能がある. 詳細は `help(plotmath)` を参照
- 実行例 `legend.r`

基本的な描画 (日本語を含む図の描画)

- 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")
```

  - 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)

- 実行例 `plot-kion.r`

ヒストグラム

- **ヒストグラム**
  - データの値の範囲をいくつかの区間に分割し, 各区間に含まれるデータの個数を棒グラフにしたもの
  - 棒グラフの横幅が区間に対応し, 面積が区間に含まれるデータの個数に比例するようにグラフを作成する
  - データの分布の仕方 (どのあたりに値が集中しているか, どの程度値にばらつきがあるかなど) を可視化するのに有効
- ヒストグラムは関数 `hist()` で作成できる

ヒストグラム

- 基本書式

```
hist(x, breaks, freq)
```

  - `x`: ヒストグラムを描画するベクトル
  - `breaks`: 区間の分割の仕方を指定．数字を指定するとデータ範囲をその数字に近い個数に等分割する．デフォルトの個数は Sturges の公式によって決定される．すなわち，データ数を n とすると，$\lceil \log_2 n + 1 \rceil$ である．[1] その他の指定方法もある (ヘルプ参照)
  - `freq`: `TRUE` 指定すると縦軸をデータ数にし，`FALSE` 指定すると縦軸をデータ数/全データ数とする．デフォルトは `TRUE` (`breaks` の指定によって変わる場合あり)
  - 他にも `plot` で指定できるオプションが利用可能

- 実行例 `hist3.r`

[1] $\lceil x \rceil$ は x 以下の最大の整数を表す．

箱ひげ図

- **箱ひげ図**
  - データの中心, 散らばり具合および外れ値を考察するための図 (ヒストグラムの簡易版)
  - 複数のデータの分布の比較の際に有効
  - データの第 1 四分位点を下端, 第 3 四分位点を上端とする長方形 (箱) と, 第 1 四分位点, 第 3 四分位点からそれぞれ箱の長さの 1.5 倍以内にあるデータのうちの最小の値, 最大の値を下端, 上端とする直線 (ひげ) からなる
  - ひげの外側のデータは点で表示される
  - 中央値は太線で表示される
- 箱ひげ図は関数 `boxplot()` で描画できる

箱ひげ図

- ベクトル x に対する箱ひげ図は boxplot(x,...) で描画できる (... に関数 plot( ) と同様のオプションを指定可能)
- データフレーム x に対して, boxplot(x,...) は列ごとの箱ひげ図を描画
- データフレーム x において, 変数 A が「分類」を表す変数 (性別, 植物の種類など)[2] の場合, 別の変数 B に対して,

```
boxplot(B ~ A, data = x, ...)
```

は変数 B を変数 A で分類した場合の, 分類ごとの箱ひげ図を描画する

- 実行例 boxplot3.r

[2] **質的変数**と呼ばれる