Rの基本的な操作

村田 昇

2020.04.17

R 言語の概要

R 言語とは

- 統計計算のための言語と環境の総称
- オープンソース・フリーソフトウェア
- 「パッケージ」を利用して容易に機能拡張が可能
  - パッケージの開発は非常に活発 (現在 10000 を越える)
  - 最新の技術や方法が簡単に導入できることも多い
- https://www.r-project.org/ (プロジェクトのサイト)

RStudio とは

- RStudio 社が開発している統合開発環境 (IDE)
  - R によるデータ解析や統計計算・パッケージ開発を支援
  - OS に依存しない対話型操作環境を提供
- 本講義では RStudio を用いて説明を行う
- https://www.rstudio.com/ (RStudio 社のサイト)

R の得意分野

- データの分類・集計・整理
- 記述統計量 (基本・要約統計量) の計算
- グラフによる視覚化
- プログラムによる処理の自動化
- 擬似乱数による不確定性を含む現象の模擬
  確率的シミュレーション・モンテカルロ法

データ形式の分類

- 構造化データ
  - 個々のデータが項目ごとに表形式で整理されている
  - 集計・分類・抽出・追加など整理が比較的容易
  - 国別の経済指標，学生の成績表
- 非構造化データ (本講義では扱わない)
  - データごとに形式や項目数など属性が異なる
  - データの整理や比較がそのままでは困難
  - 文書，画像，動画，音声

(参考) R および RStudio の導入方法

R のインストール手順

1. R: The R project for Statistical Computing にアクセス
   - http://www.r-project.org/ を直接入力
   - または “r install” などで検索
2. download R というリンクをクリック
3. CRAN のミラーサイトを選ぶ:
   - どれでも利用可だが “Japan” のいずれか を選ぶと良い
   - OS ごとにインストール方法が異なるので注意

Windows の場合

1. Download R for Windows をクリック
2. base をクリック
3. Download R (バージョン名) for Windows をクリック
4. ダウンロードされたインストーラを起動
5. 指示に従って R をインストールする

Mac の場合

1. Download R for (Mac) OS X をクリック
2. R-(バージョン名).pkg をクリック
3. ダウンロードされたインストーラを起動
4. 指示に従って R をインストールする

RStudio のインストール手順

1. Download RStudio - RStudio にアクセス
   - https://www.rstudio.com/products/rstudio/download/
   - “rstudio install” などで検索
2. Installers for Supported Platforms から自分の環境に合わせて OS を選択
3. ダウンロードしたインストーラを起動
4. 指示に従って RStudio をインストールする

RStudio のインターフェース

起動画面

- **以下 RStudio を用いて説明する**
- 起動すると 4 つの枠 (pane) を持つウィンドウが立ち上がる
  - 左上: エディタ (開いていない場合もある)
  - 左下: コンソール
  - 右上: 作業環境内の変数・コマンド履歴
  - 右下: パッケージ・グラフィックス・ヘルプ
- (ペインの配置や数は個別に設定することもできる)

エディタ (左上)

- コマンドを記述したファイルを扱うためのペイン
- コンソール上に入力したコマンドは直ちに実行される
- 複雑なコマンドを書いたり，後から修正するための機能
  - コマンドを実行順に記述したファイルを作成(R Script)
  - ファイルを保存
  - ファイルを実行
- (後で詳しく説明)

コンソール (左下)

- コマンドを入力するためのペイン
- 例えば，コンソール上で終了を指示する以下のコマンドを入力すればR を終了させることができる

```
q()
```

- (終了できない場合は OS の機能で強制終了)

作業ディレクトリ

- プログラムが実行されるディレクトリ (フォルダ)
- 作業ディレクトリにあるファイルの読み書きはパスを指定する必要がない
- **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

終了時の注意

- R 終了前に以下のメッセージが表示される場合がある

```
q()
```

```
Save workspace image? [y/n/c]:
```

- 作業で使った変数などをセーブするか尋ねている
  - y を入力: セーブする (yes の略)
  - n を入力: セーブしない (no の略)
  - c を入力: R の終了をキャンセルする (cancel の略)
- セーブした場合次回起動時に読み込まれる

基本的な使い方

式の入力

- 四則演算や数学関数は直感的な文法で計算可能
  - + (加算)，- (減算)，* (乗算)，/ (除算)， ^ または** (ベキ乗)
  - sin, cos, tan (三角関数)，exp (指数関数)，log (対数関数)
- コンソール上での計算例

```
1*2+3^2
```

```
[1] 11
```

```
sin(pi/4) + log(10)
```

```
[1] 3.009692
```

エディタからの実行

- 新規ファイルの作成 (以下のいずれか)
  - 左上の ＋ から **R Script** を選択
  - **File** から **New File** を選択，更に **R Script** を選択
- エディタ上でコマンドを記述
- 実行範囲の選択
  - 一行のみ: カーソルをその行に移動
  - 複数行: クリックしてながら移動して選択する
- 選択範囲の実行 (以下のいずれか)
  - 左上の **Run** をクリック
  - **Code** から **Selected Line(s)** を選択 (Ctrl/Command+Enter)

ファイルの保存

- ファイルの保存 (以下のいずれか)
  - 左上のディスクのマークをクリック
  - **File** から **Save** を選択 (Ctrl/Command+S)
- ファイル作成に関する注意
  - 保存する時にファイル名の入力が求められる
  - 拡張子は通常**.R** または**.r** を利用する
  - ＃ 以降の字列は実行されない *(***コメントを残す際に有用***)*

演習

以下の計算を行う Rscipt を作成し保存してみよう

- $123 \times 456 - 789$
- $(2^{2^5} + 1) \div 641$
- $\sin^2(\pi/3) + \cos^2(\pi/3)$
- 適当な数学関数を試す

関数の実行

- 関数の取り扱いは一般的な計算機言語とほぼ同様
- 関数は引数とその値を指定して実行
- ただし引数名は順序を守れば省略可能
- 関数の呼び出し方

```
f(arg1=value1, arg2=value2)
# arg1, arg2 は引数の名前, value1, value2 は引数に渡す値を表す
f(value1, value2) # 上と同値
```

関数呼び出しの例

- 正弦関数の計算

```
sin(x = pi/2)
sin(pi/2) # 上と同値
```

- 対数関数の計算 (x や b に適当な数値を代入せよ)

```
log(x, b) # 底を b とする対数
log(x=x, base=b) #上と同値
log(base=b, x=x) #上と同値
log(b,x) # = log(x=b,base=x)
log(x) # 自然対数 =log(x,base=exp(1))
```

オブジェクト

- R で扱うことのできる数値
  - 実数および複素数 (指数表記にも対応)
  - 無限大や不定な数など特殊なものにも対応
- 文字列を変数名として数値等を保持することができる
- 変数，関数，計算結果などを総称して **オブジェクト** と呼ぶ
- オブジェクトの情報は右上ペインの **Environment** タブで確認できる

オブジェクトの代入

- オブジェクトの内容 (情報) を別のオブジェクトに代入することができる
- 計算結果や良く使う文字列の保存に利用できる
- 代入操作

```
(foo <- 3) # 数値を変数 foo に代入
bar <- sin(2/3*pi) + cos(foo * pi/4) # 計算結果を代入
print(bar) # 変数の内容を表示
```

```
[1] 3
[1] 0.1589186
```

ヘルプ機能

- 各関数の詳細 (機能，引数名，引数の既定値，実行例など) を記述したヘルプが用意されている
- ヘルプに関連する関数:
  - `help()` (使い方や例の表示)
  - `example()` (例を実際に実行してくれる)
  - `help.search()` (キーワード検索)
- 右下ペイン **Help** タブ右上の検索窓でも参照可能

ヘルプの利用例

- ヘルプの使い方
  (ヘルプは右下のペインに表示される)

```
help(log) # 関数 log のヘルプ
?log # 上と同値
example(log) # ヘルプ内の例を実行
help.search("log") # "log"に関連する項目は？
??"log" # 上と同値
```

パッケージの操作

- 機能を拡張するために多数のパッケージが用意されている
- パッケージのインストール方法
  - RStudio の機能を利用する方法
  - コンソールから行う方法
- RStudio の機能を利用したインストール手順
  - 右下ペイン **Package** タブをクリック
  - 左上の **Install** をクリック
  - パッケージ名を入力し **Install** をクリック
- 利用可能なパッケージの情報は右下ペイン **Package** タブで確認できる

演習

講義で用いるパッケージを導入して，その中の関数について調べてみよう

- package e1071 を導入する
- 関数 kurtosis を調べる
- 関数 kurtosis を呼出す

データ構造

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)

ベクトル

ベクトルとは

- スカラー値の集合
- スカラー値として扱われるものには
  - 数値 (実数や複素数)

  - 文字列 (' や “で囲まれた文字．”foo“，”bar“など)
  - 論理値 (TRUE , FALSE)

  などが含まれる

- R オブジェクトの多くはベクトルとして扱われる
  (スカラー値は長さ 1 のベクトルとして扱われる)

ベクトルの生成と操作

- 数値や文字列の要素からなるベクトルの生成:

```
c(1,-2,3,-4,5) # 数値のベクトル
c("Alice","Bob","Cathy","David") # 文字列のベクトル
```

```
[1]  1 -2  3 -4  5
[1] "Alice" "Bob"   "Cathy" "David"
```

- ベクトルの要素の取得:

```
x[i] # x の第 i 要素 (ベクトルの添え字は 1 から始まる)
x[c(1,3,4)] # 複数の要素
            # = c(x[1],x[3],x[4])
```

- ベクトル x の長さの取得:

```
length(x)
```

ベクトルの作成と操作 (その 2)

- 実数 x から y まで 1 ずつ変化するベクトル:

```
x:y # x < y の場合は 1 ずつ増加，逆の場合は 1 ずつ減少
```

- 実数 x から y まで a ずつ変化するベクトル:

```
seq(x,y,by=a) # from=x,to=y と明示してもよい
```

- ベクトル x を n 回繰り返したベクトル:

```
rep(x,n) # 長さは length(x) * n となる
```

- ベクトル x とベクトル y の結合:

```
c(x,y)
```

- ベクトル x の反転:

```
rev(x)
```

演習

ベクトルの操作に慣れよう

- 以下に示すベクトルを作成してみよう
  - 1 から 10 までの自然数のベクトル
  - 1 以上 30 以下の奇数を昇順に並べたベクトル
  - すべての要素が 1 からなる長さ 10 のベクトル
- 作成したベクトルを操作してみよう
  - ベクトルの長さを求める
  - 3 番目の要素を取り出す
  - 最後の要素を取り出す

行列

行列の生成と操作

- すべての要素が a である $m \times n$ 行列:

```
matrix(a,m,n)
```

- 長さ mn のベクトル x を $m \times n$ 行列に変換:

```
x <- c(x11,...,xm1,x12,...,xm2,...,x1n,...,xmn)
matrix(x,m,n)
```

- 関数 as.vector() を用いたベクトル化:

```
x <- as.vector(matrix(x,m,n))
```

行列の生成と操作 (その 2)

- 行列のサイズの取得:

```
dim(X) # 長さ 2 のベクトル
nrow(X) # 行数
ncol(X) # 列数
```

- 行列の成分の取得:

```
X[i,j] # (i,j) 成分
X[i, ] # 第 i 行
X[ ,j] # 第 j 列
X[c(1,3),2:4] # 1, 3 行と, 2〜4 列からなる部分行列
```

- 長さが等しい複数のベクトルの結合:

```
rbind(x, y, ...) # 行ベクトルとして結合
cbind(x, y, ...) # 列ベクトルとして結合
```

行列の生成と操作 (補足)

- 関数 cbind()/rbind() は行数/列数が等しい行列を横/縦に結合できる
- 行列の高次元版として配列 (array) が用意されている
- 関数 rownames()/colnames() を用いると行と列に名前を付けることができる

演習

行列の操作に慣れよう

- 以下に示す行列を作成してみよう

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

- 行列を操作してみよう
  - 2 行 2 列成分を取り出す
  - 転置行列を作成する
  - 行名をつける

リスト

リストとは

- 異なる構造のデータを 1 つにまとめたもの
- リストの各要素は異なる型であって構わない
- 本講義のデータ解析ではほとんど用いない
- R の関数の操作ではときどき必要

リストの生成と操作

- リストの生成:

```
L <- list(x,y) # x,y を要素とするリスト
L[[i]] # リストの第 i 要素
```

- リストの各要素に名前を付与:

```
L1 <- list(first=x, second=y) # 方法 1
L2 <- list(x,y)               # 方法 2
names(L2) <- c("first","second")
```

- 名前によるリストの要素の取得:

```
L1$first      # 方法 1
L2[["first"]] # 方法 2
```

データフレーム

データフレームとは

- 長さの等しいベクトルを束ねたリスト
- いくつかの個体について，いくつかの属性を集計したデータ
- ある小学校の 1 年生の身長・体重・性別・血液型
  - 各成分はある個体のある属性に関する観測データ
  - 個体数は集計項目に関わらず変化しないが，集計項目によっては定量的データ・定性的データの違いが出てくるのでデータ型は変わりうる
- 実データの多くは表形式で与えられるため，実データに則したデータ構造

データフレームの生成と操作

- データフレームの生成:

```
data.frame(A=x,B=y,C=z) # x,y,z は同じ長さ，各列は A,B,C という名前
```

- データフレームは **リスト** でもある
  リストと同様にして各変数にアクセスできる
- データフレームは **行列** でもある
  行数がベクトルの長さ (個体数), 列数が変数の個数 (観測項目の数) の行列と同様にアクセスできる

演習

データフレームの操作に慣れよう

- 次の表に対応するデータフレームを作成してみよう

	国語	数学	英語
Alice	90	25	65
Bob	80	50	100
Cathy	70	75	70
David	60	100	40

- データフレームを操作してみよう
  - 数学の列を取り出す
  - Cathy の行を取り出す