Rの基本的な操作

第 1 講 - R 言語の仕様と RStudio の使い方

村田 昇

講義概要

- R 言語の概要
- RStudio の UI (ユーザインタフェース)
- R 言語の使い方
- R で用いるデータ構造 (ベクトル・データフレーム)

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 20000 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

- Posit Software 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 にほとんど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://posit.co/ (Posit Software 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
  - さまざまな統計分析 (多変量解析を含む)
- プログラムによる **処理の自動化**
- **確率的シミュレーション** (モンテカルロ法)
  - 擬似乱数による不確定性を含む現象の模擬

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 言語のオンラインコース

- RStudio の初心者向ガイド
  https://education.rstudio.com/learn/beginner/
- posit Cloud の自習コース
  https://posit.cloud/learn/recipes/
- Kaggle の R 言語入門コース
  https://www.kaggle.com/code/rtatman/getting-started-in-r-first-steps

RStudio の UI

起動画面

- 以下 **RStudio** を用いて説明する
- 複数の **タブ** (tab; つまみ) を含む 4 つの **ペイン** (pane; 枠) が立ち上がる
  - 左上 : エディタ・表など (開いていない場合もある)
  - 左下 : コンソール・ターミナルなど
  - 右上 : 作業環境内の変数・コマンド履歴など
  - 右下 : パッケージ・グラフィックス・ヘルプなど
- ペインの配置や数は個別に設定することが可能
  - メニュー : **Tools > Global Options** で設定

コンソール (左下ペイン)

- R 言語で記述されたコマンドを入力
  - 例えば以下のような計算を行うことができる

```
#' 一般的な数式を入力すれば計算機として使える．"#" 以降はコメントとして無視される
1 + 2 + 3 + 4          # 空白は無視される
sin(pi/3) / cos(pi/3) # tan(pi/3) になるはず
```

  - 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

```
#' R の終了には q() または quit() を用いる
q()
```

    * メニューの **Quit RStudio** も利用可能
    * 終了できない場合は OS の機能で強制終了する必要がある

エディタ (左上ペイン)

- コマンドを記述したファイルを操作
- 一連のコマンドをまとめたり修正しながら実行
  - コンソールに入力したコマンドは直ちに実行されてしまう
  - コマンドを実行順に記述したファイル **R Script** を作成
  - ファイル (の一部) を実行
  - ファイルを保存
- 同一ファイル内でプログラムと文書の記述も可能
  - **Quarto** (本講義でも利用)
  - R Markdown

ヘルプ (右下ペイン)

- 各関数の詳細を記述したヘルプが参照可能
  - 機能，引数名，引数の既定値，実行例など
- 右下ペイン **Help** タブ右上の検索窓から探索
- ヘルプ内の検索はその下の **Find in Topic** で可能
- コンソール内では関数 `help()` や `?` などを利用

パッケージ (右下ペイン)

- パッケージを用いて機能を拡張
- RStudio でのインストール手順
  - 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

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
- 現在の作業ディレクトリは **Console** タブで確認
- メニュー : **Session > Set Working Directory** で指定
  - 読み込んだファイルの場所を選択
  - **Files** タブ (右下ペイン) の場所を選択 (**More** からも選択可)
  - ディレクトリを直接選択

プロジェクト

- 作業環境をまとめて設定・保存する機能
  - 作成したプロジェクトは **Project** ボタン (右上) から選択可能
  - いつでもプロジェクトを中断可能
  - プロジェクト毎に履歴や変数を保存可能
  - 複数のプロジェクトを定義可能
- 一般的なプロジェクトの作成手順
  - **Project** ボタンから **New Project** を選択
  - **Create Project** ダイアログで **New Directory** を選択
    - 既にあるディレクトリを用いる場合は **Existing Directory**
    - Github などを利用する場合は **Version Control**
  - **Project Type** ダイアログで **New Project** を選択
  - **Directory Name** とその親ディレクトリを指定

終了時の注意

- 終了時にコンソール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

計算をする

- 四則演算や数学関数は直感的な文法で計算可能

Table 1: 基本的な演算と関数

加減乗除	`+, -, *, /`
冪乗	`^` または `**`
三角関数	`sin(), cos(), tan()`
逆三角関数	`asin(), acos(), atan()`
指数関数	`exp()`
対数関数	`log(), log10(), log2()`
双曲線関数	`sinh(), cosh(), tanh()`
平方根	`sqrt()`
絶対値	`abs()`

電卓として使う

- コンソール上での計算例

```r
#' 与えられた式の計算をコンソール上で実行する
#' 1 x 2 + 3^2 の計算
1 * 2 + 3^2
#' sin(2 π ) の計算
sin(2*pi)
#' √ 2 + |-0.6| の計算
sqrt(2) + abs(-0.6)
```

```
[1] 11
[1] -2.449294e-16
[1] 2.014214
```

- 入力内容は右上の **History** タブで確認可能

エディタから実行する

- 新規ファイルの作成 (以下のいずれか)
  - 左上の **+** から **R Script** (または **Quarto**) 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)

ファイルを保存する

- 以下のいずれかで保存することができる
  - 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
    - **R Script** の拡張子は通常**.R** または **.r** を利用
    - **Quarto** の拡張子は **.qmd** を利用
  - **#** 以降の文字列は実行されないのでコメントとして有用

実習

練習問題

基本的な操作に慣れよう

- 以下の計算を行う R Script/Quarto を作成し保存しなさい
  - $123 \times 456 - 789$
  - $(2^{2^5} + 1) \div 641$
  - $\sin^2(\pi/3) + \cos^2(\pi/3)$
  - 適当な数学関数を用いた計算

より進んだ使い方

関数

- 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 (引数がない場合もある)
  - 引数名は順序を守れば省略可能
- 関数の呼び出し方 (関数名を `f` とする)

```
f(arg1 = value1, arg2 = value2) # (擬似コード)
#' 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. 順序に注意
```

  - **擬似コード** = 実行しても動かないコード
- 実装されている関数の使い方は `help(関数名)` または `?関数名` でヘルプが表示される

関数の実行例

- 正弦関数の計算

```
#' 正弦関数 (引数が 1 つ) の計算例
sin(x = pi/2)
sin(pi/2)     # 引数名は省略でき, 前の行とこの行は同じ結果になる
```

- 対数関数の計算

```
#' 対数関数 (引数が 2 つ) の計算例
a <- 3; b <- 5        # a, b は適当な数値に置き換えて実行しなさい
log(a, b)             # 底を b とする a の対数
log(x = a, base = b) #上と同値
log(base = b, x = a) #上と同値 (引数名があれば順序は自由に変えられる)
log(b, a)             # = log(x=b,base=a) (引数名がなければ規定の順序で解釈される)
log(a)                # 自然対数 =log(a,base=exp(1))
```

ヘルプ機能

- 各関数の詳細を記述したヘルプが用意されている
  - Description (機能の概要)
  - Usage (関数の呼び出し方)
  - Arguments (関数の引数)
  - Value (関数の返り値)
  - Examples (実行例)
- 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** タブの利用
  - 右上にある検索窓でヘルプを参照可能
  - 左上にある検索窓はヘルプ内を検索可能

ヘルプ機能の利用例

- ヘルプの使い方

```r
#' 関数 log() に関するヘルプの例
help(log)           # Help タブに結果は表示される
?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
```

  - 右下ペインの **Help** タブに表示される

オブジェクト

- R で扱う対象を総称して **オブジェクト** と呼ぶ
  - 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
  - 一続きの文字 (文字列)
  - 計算手続きをまとめたもの (関数)
- オブジェクトには名前を付けることができる
- 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の例

- 代入操作の例

```r
#' 数値を変数 foo に代入する
(foo <- 3) # foo <- 3; print(foo) と等価
#' 変数 foo を用いて計算し，結果を bar に代入する
bar <- sin(2/3*pi) + cos(foo * pi/4) # 計算結果は表示されない
#' 変数 bar の内容を表示する
print(bar)
```

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

  - 計算結果や良く使う文字列の保存に利用できる
  - 変数名は自由に決められるが，予約語 `c`，`q`，`t`，`C`，`D`，`F`，`I`，`T` には注意が必要

パッケージの操作

- 機能を拡張する多数のパッケージが存在
- 利用可能なパッケージは右下ペインの **Package** タブに表示される
- 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
    - **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
  - コンソールから行う方法
    - 関数 `install.packages()` を利用

実習

練習問題

- パッケージを導入して，含まれている関数について調べなさい
  - `e1071` をインストールする
  - 関数 `kurtosis()` (尖度) を調べる
  - 関数 `kurtosis()` を呼び出す
- パッケージ群の導入を行いなさい
  - `tidyverse` をインストールする

R で用いるデータ構造

データ型

- R ではさまざまな数値を扱うことができる
  - 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

Table 2: 代表的なデータ型

型の名称	役割	例
`numeric`	(広義の) 実数を表す	`1, pi, NaN`
`complex`	複素数を表す	`1i, 3-4i`
`character`	文字列を表す	`"foo", "Hello World!"`
`logical`	論理値 (真偽) を表す	`TRUE, FALSE, 3<4, NA`

データ構造

- R ではさまざまなデータの集合を扱うことができる

Table 3: 代表的なデータ構造

構造の名称	役割
`vector`	1 次元配列 (ベクトル)
`matrix`	2 次元配列 (行列)
`array`	多次元配列
`data.frame` (`tibble`)	表形式のデータ集合
`list`	異なるデータ集合の集合

ベクトルとは

- 同じデータ型の値 (スカラー値) の集合
- R オブジェクトの多くはベクトルとして扱われる
- スカラーは長さ 1 のベクトルとして扱われる
- ベクトルの例

  数値 : 実数や複素数

  10.8　11.6　13.0　11.4　…

文字列 : ” や “” で囲まれた文字．“foo”，“bar” など

“this” “that” “apple” “orange” …

論理値 : TRUE , FALSE

TRUE TRUE FALSE TRUE …

ベクトルの作成

- 作成方法はいくつか用意されている
  - 関数 c() を用いて作成する
  - 演算子 : や関数 seq() を利用する (等差数列)
  - 関数 rep() を利用する (数列を繰り返す)
- ベクトルの作成の例

```
x <- c(1,-2,3,-4)                      # 数値のベクトル
y <- c("Alice","Bob","Cathy","David") # 文字列のベクトル
z <- c(TRUE,FALSE,TRUE,FALSE)          # 論理値のベクトル
1:5                                    # c(1,2,3,4,5) と同じ．seq(1,5,by=1) としても良い
rep(1:2, times=3)                      # c(1,2,1,2,1,2) と同じ
rep(1:2, each=3)                       # c(1,1,1,2,2,2) と同じ
```

ベクトルの要素の選択

- 演算子 [] を用いて指定する
  - 単一の要素を選択する
  - 複数の要素をベクトルを用いて選択することもできる
- 要素の選択の例

```
x[2]         # x の第 2 要素 (ベクトルの添え字は 1 から始まる)
y[c(1,3,4)] # 複数の要素 = c(y[1],y[3],y[4])
```

データフレーム

- 複数の個体について，いくつかの属性を集計した表
  - 長さの等しい列ベクトルをまとめたもの
  - 各列のデータ型はバラバラでも良い
- 実データの最も一般的な形式
- データフレームの例

ある小学校の 1 年生の身長・体重・性別・血液型のデータ

名前	身長 [cm]	体重 [kg]	性別	血液型
太郎	108	19	男	B
花子	116	21	女	O
次郎	130	25	男	AB
…	…	…	…	…

“base R” vs “tidyverse”

- データ操作とグラフィクスの枠組
  - **base R** : R の標準環境
  - **tidyverse** : Hadley Wickham @posit による拡張パッケージ集
    - https://www.tidyverse.org/packages/

    * https://tidyverse.tidyverse.org/

- 本講義では tidyverse を中心に説明
- パッケージ集の利用には以下が必要

```
#' 最初に一度だけ以下のいずれかを実行しておく
#'  - Package タブから tidyverse をインストール
#'  - コンソール上で次のコマンドを実行 'install.packages("tidyverse")'
#' tidyverse パッケージの読み込み
library(tidyverse)
```

データフレームの作成

- 作成方法はいくつか用意されている
  - 同じ長さのベクトルを並べる (関数 tibble::tibble())
  - データフレームを結合する (関数 dplyr::bind_cols())
  - マトリクスを変換する (全て同じ型の場合)
  - ファイルを読み込む (CSV 形式を主に扱う)
- データフレームの作成の例

```
#' 同じ長さのベクトル (関数 base::c() で作成) を並べる (関数 tibble::tibble())
#' (... <- ...) は代入した結果を表示
(foo <- tibble(one = c(1,2,3), two = c("AB","CD","EF")))
(bar <- tibble(three = c("x","y","z"), four = c(0.9,0.5,-0.3)))
#' データフレームを結合する (関数 dplyr::bind_cols())
(baz <- bind_cols(foo, bar)) # bind columns
```

データフレームの要素の選択

- 選択方法はいくつか用意されている
  - 添字の番号を指定する (マイナスは除外)
  - 論理値 (TRUE/FALSE) で指定する
  - 要素の名前で指定する
- 要素の選択の例

```
z <- tibble(one = c(1,2,3),
            two = c("AB","CD","EF"),
            three = 6:8)
z[1,2]                  # 1 行 2 列の要素を選択
z[-c(1,3),]             # 1,3 行を除外
z[c(TRUE,FALSE,TRUE),] # 1,3 行を選択
z[,"two"]               # 列名"two"を選択 (1 列のデータフレームになる)
z["two"]                # 上記と同様の結果
z[,c("one","three")]    # 列名"one"と"three"を選択 (データフレームになる)
z[c("one","three")]     # 上記と同様の結果
z[["two"]]              # 列名"two"のベクトルを選択 (1 列の場合しか使えない)
z$two                   # 上記と同様の結果
```

実習

練習問題

- 次の表に対応するデータフレームを作成しなさい

name	math	phys	chem	bio
Alice	90	25	65	70
Bob	80	50	100	50
Carol	70	75	70	30
Dave	60	100	40	80
Eve	50	80	75	100

- データフレームの要素の選択を試みなさい

次回の予定

- R 言語のデータ構造
- ベクトル・行列・リストの操作
- ベクトルと行列のさまざまな計算