Rの超速学習

1. RとRStudioのインストール

<https://cran.r-project.org/>

<https://posit.co/download/rstudio-desktop/>

1. ライブラリのインストールと使用

|  |
| --- |
| # ライブラリのインストールinstall.packages(ライブラリ名)  install.packages(tidyvers)  # インストール済みのライブラリの確認　installed.packages(ライブラリ名)  # ライブラリの使用 library(ライブラリ名)  library(tydyverse) |

1. 変数、ベクター、データフレーム（tibbleを含む）の作成

|  |
| --- |
| # 変数  user\_age <- 25 # ユーザーの年齢  user\_name <- "Alice" # ユーザー名  print(user\_age)  print(user\_name)  # ベクター  ids <- c(1, 2, 3) # IDのリスト  names <- c("Alice", "Bob", "Charlie") # ユーザー名のリスト  ages <- c(25, 30, 40) # 年齢のリスト  print(ids)  print(names)  print(ages)  # data.frameの作成  user\_data <- data.frame(  id = c(1, 2),  name = c("Alice", "Bob"),  age = c(25, 30),  stringsAsFactors = FALSE # Factor防止  )  print(user\_data)  # tibbleの作成  library(tidyverse) # tibbleの作成にはtidyverseが必要  user\_data <- tibble(  id = c(1, 2),  name = c("Alice", "Bob"),  age = c(25, 30)  )  # 確認  print(user\_data) |

参考

data.frame と tibble は、どちらもRのデータフレーム構造だが、tibble は data.frame の改良版である。tibble は tidyverse パッケージで提供され、データ分析や可視化に最適化されている。

|  |  |  |
| --- | --- | --- |
| 特徴 | data.frame | tibble |
| 印刷 (表示) 方法 | 大量データでも全行を表示する (画面が埋まる) | 自動的に先頭数行だけを表示 (スッキリ) |
| 文字列の型 | デフォルトで factor になる (stringsAsFactors = TRUE) | デフォルトで character 型 |
| 列名の扱い | 数字で始まる列名や空白の列名は許されない | 数字や空白の列名もOK |
| 行名 (row names) | デフォルトで行名が付く | 行名はデフォルトでは付かない |
| サブセット抽出 | 1列を指定すると ベクターになる | 1列を指定しても tibble のまま |
| print() 時の挙動 | すべての列を表示 | 列が多いときは一部だけ表示し、省略する |

listはベクターやデータフレームに似ているが、異なるデータ型や長さのオブジェクトを格納できる 「柔軟なコンテナ」 である。データフレーム (data.frame/tibble) と違い、各要素の長さが異なってもよい。

|  |
| --- |
| my\_list <- list(  number = 42, # 数値  name = "Alice", # 文字列  vector = c(1, 2, 3, 4), # ベクター  data = data.frame(x = 1:3, y = 3:1) # データフレーム  ) |

1. 関数の作成と使用

|  |
| --- |
| # my\_functionという関数を作る  # 関数名 <- function(引数1, 引数2) {式と出力}という書き方をする  my\_function <- function(arg1, arg2) {  result <- arg1 + arg2 # 足し算の結果をresultという変数に代入する  return(result) # 出力には return() を使用（省略も可能）　出力を戻り値ともいう  } |

Rの「関数」とは、特定の計算や処理を自動で行うプログラムのまとまりである。引数の異なる計算を繰り返す場合などに便利である。

関数は以下の3種類に分類できる。

* 標準関数 → Rに標準で組み込まれている関数
* 組み込み関数 → ライブラリ (package) に含まれる関数
* ユーザー定義関数 → ユーザーが独自に作成する関数

関数の主な用途や意義は以下の通り

* 再利用性: コードを繰り返し使える
* 保守性: 関数のロジックを1か所にまとめられる
* 柔軟性: 引数の指定により、異なる計算が可能

1. 標準データセットの利用

* data()によってデータセットの一覧が表示できる
* object.size(データセット名)によってデータのメモリサイズを確認できる
* dim(データセット名)によって次元を確認できる
* str(データセット名) によって各列のデータ型や要素の一部も確認できる
* summary(データセット名) によって列ごとの統計情報やカテゴリ数を確認できる
* colnames(データセット名) やrownames(データセット名)によって列名、行名が確認できる
* tidyverse パッケージの glimpse() を使うと、すべての列のデータ型と数値の範囲を確認できる

1. ダミーデータの作成

dummy\_data.Rファイルを用いてダミーデータを作成する

1. データの操作

今回は6で自作したダミーデータを用いたデータの操作方法を学ぶ

|  |
| --- |
| # 特定の行と列を選択する  # データフレーム[行,列]による選択  dummy\_data[4,4] # 4行目4列目のデータ  dummy\_data[4,] # 4行目のデータ  dummy\_data[,4] # 4列目のデータ  dummy\_data[1:3,4:6] #1-3行目 4-6列目のデータ  dummy\_data[c(1,3),c(4,6)] #1と3行目 4と6列目のデータ  # data.frameとtibbleでは選択した後のデータ型が異なる場合があるので注意する  # データフレーム[,"列名"]による選択　列は列番号の代わりに列名が利用できる  dummy\_data[,"name"] # name列が選択される　""で囲む  # データフレーム$列名による選択　列番号は使えない  dummy\_data$name # ベクターに変換される  # データフレーム[データフレーム$列名+条件,"列名"]による選択  dummy\_data[dummy\_data$gender=="Female","name"]　# gender=="Female"の行を選択 |

|  |
| --- |
| # filter(データフレーム, 条件)で特定の条件に当てはまるレコードを抽出  # age > 30に合致するデータを抽出  filter(dummy\_data, age>30) #第一引数はデータフレーム  filter(dummy\_data, age >= 30 & gender == "Male") # &は論理積（and）  filter(dummy\_data, age >= 30 | gender == "Female") # |は論理和（or）  filter(dummy\_data, gender != "Female") # !=は否定（not）  filter(dummy\_data, (age >= 30 & gender == "Male") | name == "Alice")  # 複雑な組み合わせは()を使って表現する |

|  |
| --- |
| # select(データフレーム, 列名1, 列名2)で特定の列を選択  # name列とage列だけを選択  select(dummy\_data, name, age) |

|  |
| --- |
| # arrange(データフレーム, 列名)で並べ替える  # 年齢順に昇順ソート  arrange(dummy\_data, age)  # 年齢順に降順ソート  arrange(dummy\_data, desc(age)) |

パイプ演算子を用いた操作の組み合わせ

|  |
| --- |
| # パイプ演算子では、第一引数に結果を引き渡すことができる  dummy\_data %>% # dummy\_dataをfilter関数の第一引数として引き渡す  filter(age >= 30) %>% # filterの結果をselect関数の第一引数として引き渡す  select(name, age) %>% # selectの結果をarrange関数の第一引数として引き渡す  arrange(desc(age))  # 通常の関数として書き換えると以下のようになり見にくい  arrange(select(filter(dummy\_data , age >= 30), name, age), desc(age)) |

|  |
| --- |
| # mutate(データフレーム, 新しい列名) による新しい列の追加  # 新しく 'height\_m' (身長m) 列を追加  mutate(dummy\_data, height\_m = height / 100) |

|  |
| --- |
| # summarize(データフレーム, 列名 = 式)によるデータの集計  summarize(dummy\_data, mean\_age = mean(age, na.rm = TRUE))  # 欠損値（NA）を無視して計算する：na.rm = TRUE  # na.rm = FALSEの場合、欠損値があるとエラーが出る |

|  |
| --- |
| # group\_by(データフレーム, 列名)によるグループ化  # summarizeと組み合わせて使用する  # 性別ごとに平均年齢を計算  dummy\_data %>%  group\_by(gender) %>%  summarize(mean\_age = mean(age, na.rm = TRUE)) |

1. データの可視化

ggplot2 を活用した視覚化

ggplot2は tidyverse パッケージに含まれる強力なデータ可視化ツール

|  |
| --- |
| # 散布図 (scatter plot)  library(ggplot2)　# ライブラリの呼び出し　すでに呼び出している場合には不要  # ggplot(データフレーム, aes(x=列1,y=列2, color = 列3)+ geom\_point()  ggplot(dummy\_data, aes(x = height, y = weight, color = gender)) +  geom\_point() +  labs(title = "身長と体重の関係", x = "身長 (cm)", y = "体重 (kg)")  # aesはaestheticsの略で美観 x軸、y軸、色分け  # geom\_point関数は点プロットを描く関数  # labs(title = "タイトル", x = "x軸ラベル", y = " y軸ラベル") |

|  |
| --- |
| # ヒストグラム (histogram)  ggplot(dummy\_data, aes(x = age)) +  geom\_histogram(binwidth = 5, fill = "skyblue", color = "black") +  labs(title = "年齢の分布", x = "年齢", y = "人数") |

* geom\_histogram関数はヒストグラムを描く関数
* binwidthビンの幅、fill塗りつぶしの色、color線の色

|  |
| --- |
| # 箱ひげ図 (boxplot)  ggplot(dummy\_data, aes(x = gender, y = age)) +  geom\_boxplot() +  labs(title = "性別ごとの年齢の分布", x = "性別", y = "年齢") |

* aesの色分けも可能だが、x軸の列名に合わせる

例　ggplot(dummy\_data, aes(x = gender, y = age, color = gender))

1. データのインポートとエクスポート

インポート　CSVファイルの読み込み

|  |
| --- |
| # read.csv("パス名")で指定されたcsvファイルを読み込む  # プロジェクトと同じディレクトリにdummy\_data.csvがある場合  # imported\_dataに代入  imported\_data <- read.csv("dummy\_data.csv") |

エクスポート　CSVファイルの書き出し

|  |
| --- |
| # read.csv(データフレーム, "パス名", row.names = FALSE)  # プロジェクトと同じディレクトリにexport\_data.csvとして書き出す場合  write.csv(dummy\_data, " export\_data.csv", row.names = FALSE) |

* row.names = FALSE は、データフレームの行番号 (行名) を出力しないためのオプション　デフォルトでは、CSV出力時に「行番号」が含まれる

csv以外も読み込み、書き出しの方法はあるが今回は取り上げない

また、その他の方法についても参考までに列挙する

読み込み

|  |  |  |
| --- | --- | --- |
| 方法 | 関数 | 特徴 |
| 基本の方法 | read.csv() | 標準Rの関数。最も一般的な方法 |
| 高速な読み込み | fread() (data.table) | 大規模データに最適。非常に高速 |
| tidyverse 流 | read\_csv() (readr) | tibble 形式での読み込み。使いやすく速い |
| 文字コード対応 | read.csv(file, fileEncoding = "UTF-8") | エンコーディング指定に対応 |
| エクセルファイルの読み込み | read.xlsx() (openxlsx) | エクセル形式のデータでもOK |

read.csvの主なオプション

|  |  |
| --- | --- |
| オプション | 説明 |
| header = TRUE | 1行目を列名として使用 (デフォルト) |
| stringsAsFactors = FALSE | 文字列が factor 型になるのを防ぐ |
| na.strings = "" | 空白を NA に変換 |
| fileEncoding = "UTF-8" | 文字コード指定 |

書き出し

|  |  |  |
| --- | --- | --- |
| 方法 | 関数 | 特徴 |
| 基本の方法 | write.csv() | 標準Rの関数。最も一般的な方法 |
| tidyverse 流 | write\_csv() (readr) | tibble をきれいに出力可能 |
| 高速な書き出し | fwrite() (data.table) | 数百万行でも素早く出力 |
| エクセル形式の出力 | write.xlsx() (openxlsx) | Excel形式に直接保存可能 |

参考：主な標準関数と組み込み関数

1. 標準関数 (Base R Functions)

基本操作

|  |  |  |
| --- | --- | --- |
| 関数 | 説明 | 使用例 |
| print() | オブジェクトを表示 | print(x) |
| cat() | 文字列を連結して表示 | cat("Hello", "World!") |
| length() | 要素数をカウント | length(c(1, 2, 3)) |
| class() | オブジェクトの型を確認 | class(iris) |
| typeof() | データ型を確認 | typeof(5) |

数値計算

|  |  |  |
| --- | --- | --- |
| 関数 | 説明 | 使用例 |
| sum() | 合計値 | sum(c(1, 2, 3)) → 6 |
| mean() | 平均値 | mean(c(1, 2, 3)) → 2 |
| median() | 中央値 | median(c(1, 2, 3)) → 2 |
| min() / max() | 最小値 / 最大値 | min(c(1, 2, 3)) → 1 |
| round() | 四捨五入 | round(3.14159, 2) → 3.14 |
| abs() | 絶対値 | abs(-5) → 5 |
| sqrt() | 平方根 | sqrt(16) → 4 |
| exp() / log() | 指数関数 / 対数関数 | log(100, base = 10) → 2 |

文字列操作

|  |  |  |
| --- | --- | --- |
| 関数 | 説明 | 使用例 |
| paste() | 文字列の結合 | paste("Hello", "World") → "Hello World" |
| substr() | 部分文字列の抽出 | substr("Hello", 1, 2) → "He" |
| toupper() / tolower() | 大文字 / 小文字変換 | toupper("hello") → "HELLO" |
| nchar() | 文字列の文字数カウント | nchar("Hello") → 5 |

条件・ループ

|  |  |  |
| --- | --- | --- |
| 関数 | 説明 | 使用例 |
| ifelse() | 条件に応じた値を返す | ifelse(x > 0, "Positive", "Negative") |
| for() | 繰り返しループ | for (i in 1:5) { print(i) } |
| while() | 条件が満たされるまで繰り返す | while (x < 5) { x <- x + 1 } |

データ操作

|  |  |  |
| --- | --- | --- |
| 関数 | 説明 | 使用例 |
| data.frame() | データフレームの作成 | data.frame(id = 1:3, name = c("A", "B", "C")) |
| tibble() | tidyverse のデータフレーム | tibble(id = 1:3, name = c("A", "B", "C")) |
| head() / tail() | 先頭 / 末尾のデータを表示 | head(iris) |
| dim() | データの次元を確認 | dim(iris) |
| summary() | 要約統計を表示 | summary(iris) |
| str() | データの構造を確認 | str(iris) |
| apply() | 行や列に関数を適用 | apply(matrix, 1, sum) |
| lapply() / sapply() | リストやベクターに関数を適用 | lapply(list, mean) |
| t() | 行列の転置 | t(matrix(1:4, 2, 2)) |

1. 組み込み関数 (Built-in Functions)

library() を使って読み込む必要がある。

* tidyverse 系の関数

|  |  |  |
| --- | --- | --- |
| 関数 | 説明 | 使用例 |
| read\_csv() | CSVの読み込み | read\_csv("data.csv") |
| write\_csv() | CSVの書き出し | write\_csv(data, "output.csv") |
| filter() | 条件指定による行の抽出 | filter(data, age > 30) |
| select() | 特定の列を選択 | select(data, name, age) |
| mutate() | 新しい列の追加 | mutate(data, bmi = weight / (height/100)^2) |
| arrange() | データの並べ替え | arrange(data, desc(age)) |
| group\_by() + summarize() | グループごとの集計 | group\_by(data, gender) %>% summarize(avg\_age = mean(age)) |

* ggplot2 系の関数

|  |  |  |
| --- | --- | --- |
| 関数 | 説明 | 使用例 |
| ggplot() | グラフ作成のベース | ggplot(data, aes(x, y)) |
| geom\_point() | 散布図 | ggplot(data, aes(x, y)) + geom\_point() |
| geom\_histogram() | ヒストグラム | ggplot(data, aes(x)) + geom\_histogram() |
| geom\_boxplot() | 箱ひげ図 | ggplot(data, aes(x, y)) + geom\_boxplot() |