

医療データ科学実習

Practice of Biomedical Data Science

第2回

Slidoで質疑応答に参加しよう

医療データ科学実習第2回

Q&A

11 Polls

医療データ科学実習第2回
2025/04/15~2025/04/22
#2974 065

ライブインタラクシ...

Slido を切り替え

ダークモード

Slido について

Slido を無料で試す

質問を入力

人気 最近

質問投稿部分

1 個の質問

匿名
6 日前

0 0

テスト質問です

Moderator
6 日前

テスト質問了解です

人の質問に投票できる
人気の質問は上位に表示される

- 匿名で質疑応答に参加できるプラットフォーム
- スマホからでも簡単に利用できます
- 講義後1週間解放しておくので自由に質問やコメントを投稿してください

主催者としてログイン -
プレゼンテーション モード
許可可能な使用 - Slido のプライバシー
Cookie の設定
© 2012-2025 Slido - 67.29.3

slido

参加

URL : <https://app.sli.do/event/ntA1oiNfNnSgbddJ8sUhiQ>



Chap1. R言語事始め(続き)

- データフレームの作成
- 外部ファイルへの出力
- 外部ファイルの読み込み

R上でデータフレームを作成してみよう

- Rで表形式のデータ(いわゆる表計算ソフトのような行列状のデータ)を作成するための基本的な関数に `data.frame()` 関数がある
- 基本構文(各列には同じ長さのベクトルを指定する):

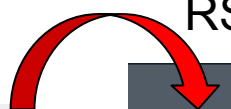
```
data.frame(列名1 = ベクトル1, 列名2 = ベクトル2, ...)
```

ベクトルの定義: `c(要素1, 要素2, 要素3, ...)`

- 例(各列はカンマ区切りであることに注意)

```
my_data <- data.frame(  
  列名 Name = c("Alice", "Bob", "Charlie"),  
  Age = c(25, 30, 22),  
  Score = c(90.5, 88.0, 95.0)  
)
```

RStudioでの実行結果



	Name	Age	Score
1	Alice	25	90.5
2	Bob	30	88.0
3	Charlie	22	95.0

演習: RStudioで上のコードを実行してデータフレームを表示しよう

R上のデータフレームを外部ファイルに出力しよう

- Rでは `write()` や `cat()`, `write.csv()` 関数などを使って文字列や変数, データフレームの内容を外部ファイルに出力できる.
- 外部ファイルは `readLines()` や `read.csv()` でR内に読み込むことができる

演習: 次のコードをRStudioで実行し, csvファイルの出力結果を確認しよう

```
write.csv(my_data, file="my_data_output.csv", row.names=FALSE)
```

出力するデータフレーム

出力先のファイル名("" で囲む)

TRUE → 行番号を出力する

FALSE → 行番号を出力しない

出力されたファイルの場所が分からないとき

→ `getwd()` を実行すると現在のワーキングフォルダが表示されるので, フォルダを開いて `my_data_output.csv` を探す

実行結果をエクセルで開いたもの:

	A	B	C
1	Name	Age	Score
2	Alice	25	90.5
3	Bob	30	88
4	Charlie	22	95

外部ファイルをRに読み込んで表示しよう

- Rでは `write()` や `cat()`, `write.csv()` 関数などを使って文字列や変数, データフレームの内容を外部ファイルに出力できる.
- 外部ファイルは `readLines()` や `read.csv()` でR内に読み込むことができる

演習: 次のコードをRStudioで実行し, 結果を確認しよう

```
loaded_data <- read.csv("my_data_output.csv")  
print(loaded_data)
```

読み込むファイル名("" で囲む)

コンソール

```
> loaded_data <- read.csv("my_data_output.csv")  
> print(loaded_data)  
  Name Age Score  
1  Alice  25  90.5  
2   Bob   30  88.0  
3 Charlie  22  95.0  
> |
```

実行結果

オブジェクト内容

Data		
loaded_data	3 obs. of 3 variables	<input type="checkbox"/>
my_data	3 obs. of 3 variables	<input type="checkbox"/>

読み込んだデータフレーム

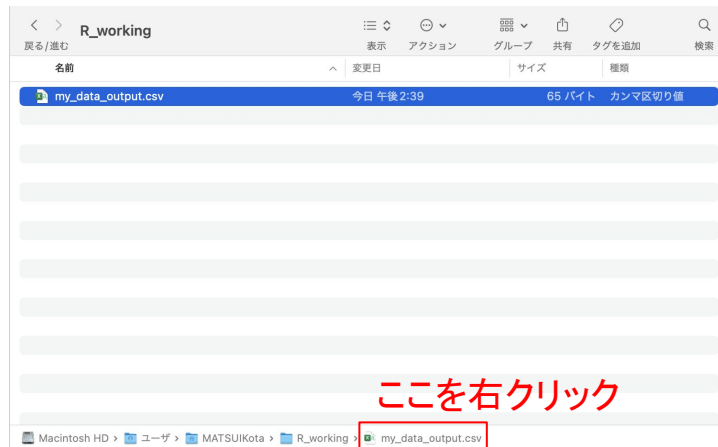
外部ファイルをRに読み込んで表示しよう

macの場合

(注意)読み込むファイルが保存されているフォルダとワーキングフォルダが違っているとエラーが出る

[対策1] read.csvするときの " " の中にファイルのパスを全て書く

- windowsの場合
 - a. ファイルをShiftキーを押しながら右クリックし、表示された一覧から「パスのコピー」をクリック
 - b. read.csvの " " の中に貼り付け
- macの場合
 - a. フォルダを開いたら「表示」メニューから「パスバーを表示」をクリック
 - b. ファイル名をクリックするとフォルダ下部にパスが表示されるので、ファイル名部分を右クリックして「ファイル名"のパス名をコピー」をクリック
 - c. read.csvの " " の中に貼り付け



外部ファイルをRに読み込んで表示しよう

(注意) 読み込むファイルが保存されているフォルダとワーキングフォルダが違っているとエラーが出る

[対策2] `setwd()` 関数でワーキングフォルダを設定する

```
setwd("フォルダのパス")
```

例(松井の場合):

```
setwd("/Users/MATSUIKota/Dropbox/workspace/Matsui_Lab/2025/医療データ科学実習（前期）/第1回/r_project_demo")
```

演習: RStudioで以下のコードを実行してワーキングフォルダを設定してみよう

```
getwd() 現在のワーキングフォルダを確認. パスをどこかに保存しておく  
setwd("~/") ワーキングフォルダをホームディレクトリに変更  
getwd() ワーキングフォルダがホームディレクトリに変更されているか確認  
setwd("元のワーキングフォルダのパス") はじめに保存しておいたパス  
getwd() 元のワーキングフォルダに戻ってきているか確認
```


Chap2. R言語事始め(続き)

- 組み込み関数の利用
- 外部パッケージのインストール
- 外部パッケージのインポートと利用

組み込み関数

- Rに最初から含まれている関数で, 追加パッケージをインストールせずすぐ使える

カテゴリ	代表的な組み込み関数			
算術演算	<code>sum()</code>	<code>mean()</code>	<code>sqrt()</code>	<code>abs()</code>
統計処理	<code>var()</code>	<code>sd()</code>	<code>median()</code>	<code>quantile()</code>
データ構造生成	<code>c()</code>	<code>matrix()</code>	<code>list()</code>	<code>data.frame()</code>
乱数生成	<code>rnorm()</code>	<code>runif()</code>	<code>sample()</code>	<code>set.seed()</code>

- 組み込み関数の一覧を確認する方法: `library(help = "base")`

組み込みの `base` パッケージにある関数の一覧を表示

組み込み関数を使った操作の例

- ベクトルの定義と操作

演習: RStudioで以下のコードを実行してベクトルを定義し、いろいろな操作をしてみよう

```
x <- c(1, 2, 3, 4, 5) ベクトル x の定義
sum(x) x の要素の合計
mean(x) x の要素の算術平均
sqrt(x) x の「各要素」の平方根 → 要素ごとの演算になる
sample(x, 3) x の要素の中から3つランダム抽出
```

- 乱数生成

演習: RStudioで以下のコードを実行していろいろな乱数を生成してみよう

```
runif(5, min = 0, max = 1) 0~1の範囲で一様乱数を5個生成
rnorm(5, mean = 100, sd = 15) 平均100, 標準偏差15の正規分布に従う乱数を5個生成
rbinom(5, size = 10, prob = 0.5) 成功確率0.5, 試行回数10回の二項分布に従う乱数を5個生成
```

Rにおける乱数生成と乱数シード

- 乱数シードとは？

乱数生成の出発点となる初期値. これを設定することで同じ乱数列を再現することができる

- 乱数シードの固定方法: `set.seed(数値)`

乱数シードを固定しない

```
> runif(5, min = 0, max = 1)
[1] 0.05352104 0.98155767 0.15084077 0.78914458 0.68673811
> runif(5, min = 0, max = 1)
[1] 0.2685128 0.7562047 0.3970674 0.1572252 0.4856097
> runif(5, min = 0, max = 1)
[1] 0.62152182 0.08209173 0.95623535 0.06640249 0.51158993
> |
```

毎回異なる結果が得られる

乱数シードを固定

```
> set.seed(46)
> runif(5, min = 0, max = 1)
[1] 0.1843486 0.2433627 0.5839976 0.3456296 0.2331262
> set.seed(46)
> runif(5, min = 0, max = 1)
[1] 0.1843486 0.2433627 0.5839976 0.3456296 0.2331262
> set.seed(46)
> runif(5, min = 0, max = 1)
[1] 0.1843486 0.2433627 0.5839976 0.3456296 0.2331262
>
```

毎回同じ結果が得られる

なぜ乱数シードが重要か？

- 同じコードを別の人が実行しても結果が一致する
- シミュレーションや検証実験で公平な比較ができる
- 学術論文やレポートで再現可能性 (reproducibility) が求められるときに有効

Rにおける乱数生成と乱数シード

- 乱数シードとは？
乱数生成の出発点となる初期値. これを設定することで同じ乱数列を再現することができる
- 乱数シードの固定方法: `set.seed(数値)`

演習: RStudioで, 以下の乱数生成を3回ずつ行い, 結果を比較してみよう

- 乱数シードを固定せず, 平均100, 標準偏差15の正規分布に従う乱数を5個生成
- 乱数シードを固定して, 平均100, 標準偏差15の正規分布に従う乱数を5個生成



演習: [Rの乱数生成](#) 



Rによる数値計算とベクトル演算

演習: Rのベクトル演算

- Rでは, **ベクトル演算**を活用することで繰り返し処理を大幅に効率化できる(むしろこれを使わないと非常に処理が遅い)

例: 1千万要素の和の計算

ベクトル演算を使用

```
# ベクトルを準備
x <- runif(1e7)
y <- runif(1e7)

# 2. ベクトル演算による加算
system.time({
  result_vec <- x + y
})
```

計算時間(colab): 0.164 (sec)

forループで計算

```
system.time({
  result_loop <- numeric(1e7) # 初期化
  for (i in 1:n) {
    result_loop[i] <- x[i] + y[i]
  }
})
```

計算時間(colab): 1.492 (sec)

外部パッケージのインストール

- 外部パッケージをインストールする方法:
コンソールで以下を実行

```
install.packages("パッケージ名", repos="ミラーサイトのURL")
```

- 例えば `ggplot2` パッケージをインストールしたい場合を実行

```
install.packages("ggplot2", repos="https://cloud.r-project.org")
```

RStudio社提供のグローバルミラーで非常に安定・高速

- インストールが完了したら `library(パッケージ名)` でインポートできる

```
library(ggplot2) ← library() で呼び出すときは " " は不要な点に注意
```

演習: RStudioで上記のコードを実行して `ggplot2` をインストールしてみよう

インストールした `ggplot2` を使ってみよう

演習: RStudioで下記のコードを実行して `ggplot2` を使ってみよう

1. `ggplot2` のインポート: `library(ggplot2)` を実行
2. データの準備: 以下を実行

```
my_data <- data.frame(  
  x = c(1, 2, 3, 4, 5),  
  y = c(2, 4, 1, 8, 7)  
)
```

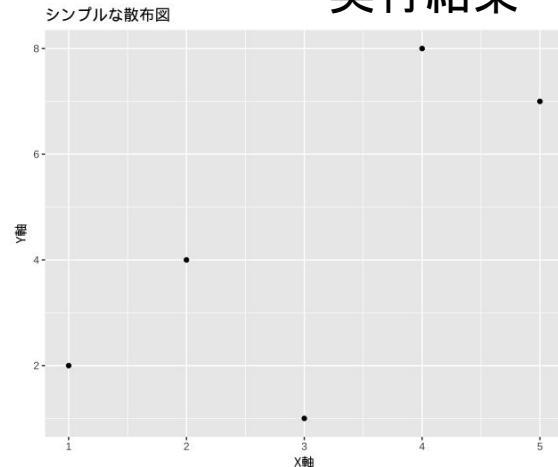
3. 散布図を描く: 以下を実行

```
① ggplot(data = my_data, aes(x = x, y = y)) + ② geom_point()  
  + labs(title = "シンプルな散布図", x = "X軸", y = "Y軸")  
③
```

- ①: データ+プロットの美術 (aesthetics) を指定
- ②: プロットの幾何オブジェクト (geom)
- ③: オプションの指定

} これらを + 演算子で繋げていくだけ

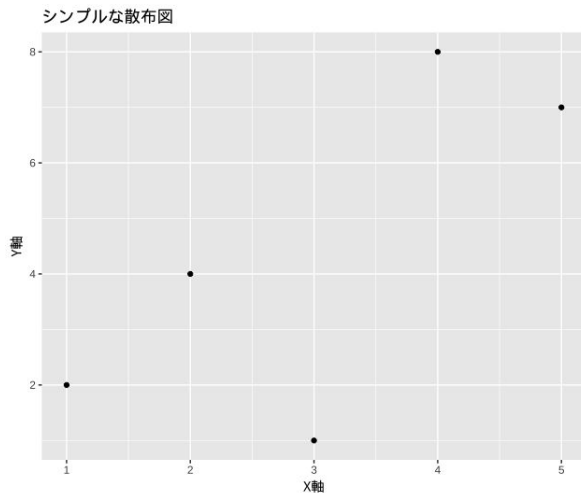
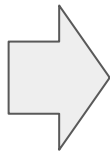
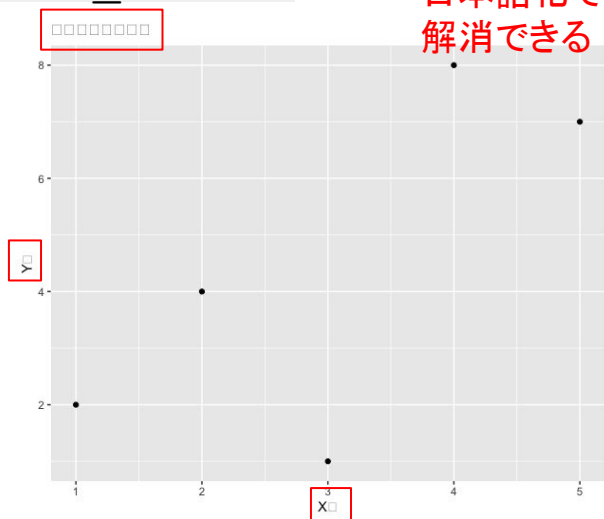
実行結果



インストールした `ggplot2` を使ってみよう(補足)

- プロットが文字化けするとき
→ 以下を実行して日本語フォントをインポート・有効化する

```
install.packages("showtext", repos="https://cloud.r-project.org")  
library(showtext)  
showtext_auto()
```



Rによる乱数生成とプロット

演習: RStudioで実行してみよう

1. 平均が50, 分散が10の正規分布から乱数を1000個生成し, ヒストグラムとしてプロットしてみよう

ヒント

- 正規分布に従う乱数の生成は `rnorm()` 関数で
- `ggplot2` でヒストグラムを描画するためには `geom_histogram()` 関数を使う

2. 乱数シードを変えて実行し, ヒストグラムがどう変わるか見てみよう

* サンプルコードは講義後に公開します

[サンプルコード](#)