2023.12.14

短期集中研修
『Rデータ解析自由自在（入門編）』

# ③データ操作と可視化

三村　喬生

統計数理研究所
医療健康データ科学研究センター

# データ科学

対象に内在する構造を体系的に切り分け
共有可能な知識に変換すること、その方法

(ある視点から)同じものを同じとし、
違うものを違うものとして分類整理する

情報(実存の写像)のうち意思伝達・解釈・
処理に適した再利用可能なもの

# プログラミング言語を使おう

**基本的な姿勢**

結果ではなく解析の**工程**を
明示的に組み上げる

- 自ずと結果も共有される。
- 追加処理が容易になる。
- デバックしやすい。

# R = スクリプト言語

# R = スクリプト言語

スクリプト

```
## packages ----
library(tidyverse)

## parameters ----
r <- 2
x_center <- 1
y_center <- 3

## data ----
df <-
  tibble(
    theta = seq(0, 2 * pi, length = 361),
    X = r * cos(theta) + x_center,
    Y = r * sin(theta) + y_center
  )

## visualization ----
ggplot(data = df) +
  aes(x = X, y = Y) +
  geom_path() +
  coord_fixed()
```

# ■ Rを始めよう

- 基礎知識(オブジェクト, 関数)
- データの読み書き
- データの操作
- データの可視化

# ■ Rを始めよう

- **基礎知識**

→ オブジェクトは**名詞**

→ 関数は**動詞**

- **データの読み書き** ⬅

+ プロジェクト管理 in RStudio

+ パッケージの利用

+ パイプ演算子

+ if文, for文, ディレクトリ操作など

# ■ Rを始めよう

- 基礎知識(オブジェクト, 関数)
- データの読み書き
- **データの操作**
- データの可視化

「先生、ちょっと待って。スクリプトがごちゃごちゃになってきました！」

「先生、ちょっと待って。スクリプトがごちゃごちゃになってきました！」

「R Markdownを使うんじゃよ。」

# File -> New File -> R Markdown...

Knit Run

Source Visual Outline

```
---
title: "test"
output: html_document
date: "2023-12-04"
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

## R Markdown

This is an R Markdown document. Markdown is a
simple formatting syntax for authoring HTML, PDF,
and MS Word documents. For more details on using
R Markdown see <http://rmarkdown.rstudio.com>
```

Knit
Run
Source
Visual
Outline
Knit
---
title: "test"
output: html_document
date: "2023-12-04"
---
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
## R Markdown
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using

data
r-training.Rproj
204 B
test.html
690.9 KB
test.R
335 B
test.Rmd
811 B

data
r-training.Rproj
204 B
690.9 KB
335 B
811 B
Open in Editor
View in Web Browser

# test

2023-12-04

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

## Including Plots

You can also embed plots, for example:

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

# {rmarkdown}パッケージ

## test

2023-12-04

### R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

### Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

# プログラミング

CSV
CSV
Input
code
# comment
(for PC)
PNG
PNG
CSV
CSV
Output
document
(for human)

# “文学的”プログラミング Literate Programming

# “文学的”プログラミング Literate Programming

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to humans what we want the computer to do.

— Donald E. Knuth, Literate Programming, 1984

プログラム構築に関する伝統的な態度を切り替えてみましょう。
コンピュータに何をするべきかを指図するのが主な仕事だと考えるのではなく、
むしろ**コンピュータに何をして欲しいかを人間に説明する**ことに集中しよう。

— ドナルド・E・クヌース, 文芸的プログラミング, 1984

# “文学的”プログラミング in R

WAB → Noweb

↓

Sweave

2002

knitr

2012

rmarkdown

2014

Quato?

2021

Yihui Xie
@xieyihui

**R Markdown: The Definitive Guide**

**Yihui Xie, J. J. Allaire, Garrett Grolemund, 2022**

https://bookdown.org/yihui/rmarkdown/#ref-xie2015

# {rmarkdown}パッケージ

test

2023-12-04

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

## Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

レポートの中にRのコードとその実行結果を埋め込むことができる。

レポートの記述にMarkdownを使用することができる。

# Markdownの基本

| テキスト表示 | ブラウザ表示 |
| --- | --- |
| *This* | *This* |
| **This** | **This** |
| ***This*** | ***This*** |
| ~~This~~ | ~~This~~ |
| <u>This</u> | <u>This</u> |

# Markdownの基本

テキスト表示 → ブラウザ表示

```
# Title

## Title

### Title

#### Title

##### Title
```

# Title

## Title

### Title

#### Title

##### Title

# Markdownの基本

テキスト表示 → ブラウザ表示

```
$$
y \sim Normal(\mu, \sigma^2)
$$
```

$$y \sim Normal(\mu, \sigma^2)$$

```
|ID|title||
|--|--|--|
|kilometer00|R Markdown|link|
```

| ID | title | |
|---|---|---|
| kilometer00 | R Markdown | link |

# Markdown

テキスト表示 → ブラウザ表示

```
Click [here](sample.html)!!
```

Click here!!

```
![](img/logo_TokyoR.png){width=30%}
```

# Markdown

テキスト表示 → ブラウザ表示

```
```r
library(palmerpenguins)

penguins[1, ]
```


```
# A tibble: 1 × 8
  species island  bill_…¹ bil
  <fct>   <fct>     <dbl>   <
1 Adelie  Torger…    39.1
# … with abbreviated variable
#   ²bill_depth_mm, ³flipper_
```
```

```r
library(palmerpenguins)

penguins[1, ]
```

```
# A tibble: 1 × 8
  species island  bill_…¹ bill_…² flipp…³ body_…⁴ sex    year
  <fct>   <fct>     <dbl>   <dbl>   <int>   <int> <fct> <int>
1 Adelie  Torger…    39.1    18.7     181    3750 male   2007
# … with abbreviated variable names ¹bill_length_mm,
#   ²bill_depth_mm, ³flipper_length_mm, ⁴body_mass_g
```

# R短期研修メモ

2023-12-04

## データを連番で書き出すスクリプト

```
library(tidyverse)

path <- "data"

for(i in 1:10){
  df <-
    data.frame(x = c(1, 2, 3),
               y = c("a", "b", "c"))

  path_output <-
    i %>%
    formatC(width = 3, flag = "0") %>%
    paste0(path, "/sample_", ., ".csv")

  df %>%
    write_csv(path_output)
}
```

- data.frameオブジェクトを作成するためのベクトルは同じ長さである必要がある。
- `paste0()` 関数は `stringr::str_c()` 関数でも良い。
- `readr::write_csv()` 関数は自動的に `row.names = FALSE` になる。

## データをまとめて読み込むスクリプト

```
## packages ----
library(tidyverse)

## parameters ----
path_folder <- "data"
key <- "sample_"

## path ----
list_path <-
  path_folder %>%
  list.files(full.names = TRUE) %>%
  str_subset(pattern = key)

## import ----
df <-
  list_path %>%
  read_csv(id = "file_path")
```

```
df
```

```
## # A tibble: 30 × 3
##    file_path               x y
##    <chr>               <dbl> <chr>
##  1 data/sample_001.csv     1 a
##  2 data/sample_001.csv     2 b
##  3 data/sample_001.csv     3 c
##  4 data/sample_002.csv     1 a
##  5 data/sample_002.csv     2 b
```

# .Rmdを書く！

.Rmd

documentを設定する
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く

# .Rmdを書く！

.Rmd

documentを設定する
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く

ここは平和な世界

- **No** コピペ
- **No** 作業窓切り替え

# .Rmdを書く！

.Rmd

documentを設定する
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く

## ここは平和な世界

- **No** コピペ
- **No** 作業窓切り替え

R codeに頑張って説明用のドキュメントをつける

⬇

R codeをドキュメントの中に埋め込んで一緒に保管する

# .Rmdを書く！

.Rmd

documentを設定する
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く

```
---
title: "R短期研修"
output:
  html_document: default
---
```

YAML header
(YAML Ain't a Markup Language.)

ドキュメント全体に掛かる設定を指定できる。

# .Rmdを書く！

.Rmd

documentを設定する
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く

```
---
title: "R短期研修"
output:
  html_document:
    keep_md: TRUE
---
```

YAML header
(YAML Ain't a Markup Language.)

ドキュメント全体に掛かる設定を指定できる。

Knit
Knit to HTML
Knit to PDF
Knit to Word
.Rmd
knitr
.md
Pandoc
.html
.pdf
.docx

# .Rmdをknitする！

英語で書くなら

| | |
|---|---|
| .Rmd → .html | 簡単！素晴らしい！！ |
| .Rmd → .pdf | 良い感じ。便利。 |
| .Rmd → .docx | あー、必要ならどうぞ。 |

# .Rmdをknitする！

日本語で書くなら

| | |
|---|---|
| .Rmd → .html | 簡単！素晴らしい！！ |
| .Rmd → .pdf | 闇。 |
| .Rmd → .docx | **深い闇。** |

# 闇(やみ)

**闇**（やみ）とは、光の無い状態のこと[1][2]。**暗闇**（くらやみ）とも、**暗黒**とも。

https://ja.wikipedia.org/wiki/%E9%97%87

As soon as Anakin came before him on Coruscant, Yoda could sense darkness in his soul.

--- *The Phantom Menace*

コルサントでアナキンがヨーダの前に現れた時、
ヨーダはすぐに彼の魂の中にある**闇**を見出した。

# .Rmdを書く！

.Rmd

documentを設
↓
documentを書
↓
Rを書く
↓
Rをテストする
↓
documentを書
↓
Rを書く
↓
Rをテストする
↓
documentを書く

```
---
title: "R短期研修メモ"
output:
  pdf_document:
    latex_engine: xelatex
documentclass: bxjsarticle
classoption: xelatex,ja=standard
mainfont: Hiragino Kaku Gothic Pro
---
```

日本語PDFをknitするための
YAML header
(環境に**強く**依存！)

# .Rmdを書く！

## .Rmd

documentを設定する
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く

チャンク chunk

Chunk name
(必須ではない)

```{r chunk1}

```

チャンク chunk

```{r chunk2}

```

# チャンク・オプション

**chunkの表示方法・実行の有無などを細かく指定できる。**

Language　Name　Options

```
```{r chunk1, ...}

```
```

II

```
```{r chunk1}
#| options...

```
```

# チャンク・オプション

```{r chunk1}
# コード非表示、結果表示
#| echo = FALSE

# コード表示、結果非表示、実行されない
#| eval = FALSE

# コード非表示、結果非表示、実行される
#| include = FALSE

# コード表示、結果非表示、実行される
#| results = FALSE
```

# チャンク・オプション

```{r chunk1}
# コードと結果を1つのブロックで表示
#| collapse = TRUE

# 警告を表示しない
#| warning = FALSE

# メッセージを表示しない
#| message = FALSE

# 図の横幅を指定(縦幅はfig.height)
#| fig.width = 3
```

# .Rmdを書く！

.Rmd

documentを設定する
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く
↓
Rを書く
↓
Rをテストする
↓
documentを書く

セットアップ・チャンク
- setupという名前のチャンク
- 必須(docの冒頭に書こう)
- doc内に1つだけ
- テンプレには自動で挿入

```{r setup}
#| include = FALSE

# このdoc内のchunk optionsを
# 一括でグローバルに指定できる
knitr::opts_chunk$set(
  echo = TRUE,
  ...
)
```

# R短期研修メモ

2023-12-04

## データを連番で書き出すスクリプト

```
library(tidyverse)

path <- "data"

for(i in 1:10){
  df <-
    data.frame(x = c(1, 2, 3),
               y = c("a", "b", "c"))

  path_output <-
    i %>%
    formatC(width = 3, flag = "0") %>%
    paste0(path, "/sample_", ., ".csv")

  df %>%
    write_csv(path_output)
}
```

- data.frameオブジェクトを作成するためのベクトルは同じ長さである必要がある。
- `paste0()` 関数は `stringr::str_c()` 関数でも良い。
- `readr::write_csv()` 関数は自動的に `row.names = FALSE` になる。

## データをまとめて読み込むスクリプト

```
## packages ----
library(tidyverse)

## parameters ----
path_folder <- "data"
key <- "sample_"

## path ----
list_path <-
  path_folder %>%
  list.files(full.names = TRUE) %>%
  str_subset(pattern = key)

## import ----
df <-
  list_path %>%
  read_csv(id = "file_path")
```

```
df
```

```
## # A tibble: 30 × 3
##    file_path              x y
##    <chr>              <dbl> <chr>
##  1 data/sample_001.csv     1 a
##  2 data/sample_001.csv     2 b
##  3 data/sample_001.csv     3 c
##  4 data/sample_002.csv     1 a
##  5 data/sample_002.csv     2 b
```

やってみよう

# ■ Rを始めよう

- 基礎知識
- データの読み書き
- **レポーティングの基礎** ⬅
- データの操作
- データの可視化

# ■ Rを始めよう

- 基礎知識
- データの読み書き
- レポーティングの基礎
- **データの操作** ⬅
- データの可視化

# ■ データの加工

【目標】
data.frame (もしくはtibble)を
自由自在に変形する

# ■ データの加工

【目標】
data.frame (もしくはtibble)を
自由自在に変形する

1. データの縦型⇔横型変換
2. 行, 列方向のデータ加工
3. データの結合

# ■ 縦型⇔横型変換

{tidyr}
パッケージ

# 横型データ

**やってみよう**

```
library(tidyverse)
set.seed(71)

df_wide <-
  tibble(
    患者 = str_c("患者", 1:5),
    検査1 = runif(n = 5, 0, 1),
    検査2 = c(T, F, T, T, F),
    検査3 = runif(n = 5, 100, 200)
  )
```

※ 乱数の関連（set.seed()関数とrunif()関数）は明日やります。

# 横型データ

```
> df_wide
# A tibble: 5 × 4
  患者  検査1  検査2  検査3
  <chr> <dbl> <lgl> <dbl>
1 患者1 0.118 TRUE   178.
2 患者2 0.762 FALSE  113.
3 患者3 0.142 TRUE   181.
4 患者4 0.767 TRUE   190.
5 患者5 0.842 FALSE  193.
```

# 縦型データ

```
> df_long
# A tibble: 15 × 3
   患者  検査    value
   <chr> <chr>   <dbl>
 1 患者1 検査1   0.118
 2 患者1 検査2   1
 3 患者1 検査3 178.
 4 患者2 検査1   0.762
 5 患者2 検査2   0
 6 患者2 検査3 113.
 7 患者3 検査1   0.142
 8 患者3 検査2   1
 9 患者3 検査3 181.
```

# 横型 → 縦型

```
df_wide %>%
  pivot_longer(
    # 縦方向に展開する列
    cols = c(検査1, 検査2, 検査3),
    # 水準を格納する列の列名
    names_to = "検査",
    # 値を格納する列の列名
    values_to = "value"
  )
```

# 横型 → 縦型

```
df_wide %>%
  pivot_longer(
    # 縦方向に展開する列
    cols = str_c("検査", 1:3),
    # 水準を格納する列の列名
    names_to = "検査",
    # 値を格納する列の列名
    values_to = "value"
  )
```

# 横型 → 縦型

```
df_wide %>%
  pivot_longer(
    # 縦方向に展開する列
    cols = 検査1:検査3,

    # 水準を格納する列の列名
    names_to = "検査",

    # 値を格納する列の列名
    values_to = "value"
  )
```

# 横型 → 縦型

```
df_wide %>%
  pivot_longer(
    # 縦方向に展開する列
    cols = !患者,
    # 水準を格納する列の列名
    names_to = "検査",
    # 値を格納する列の列名
    values_to = "value"
  )
```

# 横型 → 縦型

```
df_wide %>%
  pivot_longer(
    # 縦方向に展開する列
    cols = contains("検査"),
    # 水準を格納する列の列名
    names_to = "検査",
    # 値を格納する列の列名
    values_to = "value"
  )
```

# 横型 → 縦型

やってみよう

```
df_long <-
  df_wide %>%
  pivot_longer(
    cols = contains("検査"),
    names_to = "検査",
    values_to = "value"
  )
```

# 列選択お助け関数群

## Select help functions

```
starts_with("s") # 先頭一致
ends_with("s") # 末尾一致
contains("se") # 部分一致
matches("^.e") # 正規表現
any_of(c("tag", "B")) # 複数列
everything() # 全ての列
```

# 縦型データ

```
> df_long
# A tibble: 15 × 3
   患者  検査    value
   <chr> <chr>   <dbl>
 1 患者1 検査1   0.118
 2 患者1 検査2   1
 3 患者1 検査3 178.
 4 患者2 検査1   0.762
 5 患者2 検査2   0
 6 患者2 検査3 113.
 7 患者3 検査1   0.142
 8 患者3 検査2   1
 9 患者3 検査3 181.
```

# 縦型 → 横型

```
df_long %>%
  pivot_wider(
    # 水準が格納されている列の列名
    names_from = "検査",
    # 値が格納されている列の列名
    values_from = "value"
  )
```

# 横型データ

```
# A tibble: 5 × 4
  患者   検査1  検査2 検査3
  <chr> <dbl> <dbl> <dbl>
1 患者1 0.118     1  178.
2 患者2 0.762     0  113.
3 患者3 0.142     1  181.
4 患者4 0.767     1  190.
5 患者5 0.842     0  193.
```

# 横型データ

```
# A tibble: 5 × 4
  患者   検査1 検査2 検査3
  <chr> <dbl> <dbl> <dbl>
1 患者1 0.118     1  178.
2 患者2 0.762     0  113.
3 患者3 0.142     1  181.
4 患者4 0.767     1  190.
5 患者5 0.842     0  193.
```

やってみよう

```
> df_wide
# A tibble: 5 × 4
  患者  検査1  検査2  検査3
  <chr> <dbl> <lgl> <dbl>
1 患者1 0.118 TRUE   178.
2 患者2 0.762 FALSE  113.
3 患者3 0.142 TRUE   181.
4 患者4 0.767 TRUE   190.
5 患者5 0.842 FALSE  193.
```

logical型

何で？

```
# A tibble: 5 × 4
  患者  検査1  検査2  検査3
  <chr> <dbl> <dbl> <dbl>
1 患者1 0.118     1  178.
2 患者2 0.762     0  113.
3 患者3 0.142     1  181.
4 患者4 0.767     1  190.
5 患者5 0.842     0  193.
```

double型

# ■ リスト list

```
x <- list(c(1, 2, 3), c(4, "a"))
```

```
> x
[[1]]
[1] 1 2 3

[[2]]
[1] "4" "a"
```

※ 1つベクトルの中身は1つの型v

```
> df_wide
# A tibble: 5 × 4
  患者  検査1  検査2  検査3
  <chr> <dbl> <lgl> <dbl>
1 患者1 0.118 TRUE   178.
2 患者2 0.762 FALSE  113.
3 患者3 0.142 TRUE   181.
4 患者4 0.767 TRUE   190.
5 患者5 0.842 FALSE  193.
```

logical型

戻せる？

```
# A tibble: 5 × 4
  患者   検査1  検査2 検査3
  <chr> <dbl> <dbl> <dbl>
1 患者1 0.118     1  178.
2 患者2 0.762     0  113.
3 患者3 0.142     1  181.
4 患者4 0.767     1  190.
5 患者5 0.842     0  193.
```

double型

# 縦型 → 横型

```
df_long %>%
  pivot_wider(
    names_from = "検査",
    values_from = "value"
  ) %>%
  mutate(検査2 = as.logical(検査2))
```

dplyr::mutate()関数 = 列方向のデータ加工関数
後ほど詳しく解説します。

# データ型 type

| | |
|---|---|
| `integer` | 整数 |
| `double` | 実数（整数を含む） |
| `complex` | 複素数 |
| `logical` | 論理型（TRUE, FALSE, NA, NaN） |
| `character` | 文字列 |

# データ型の変換

```
vec <- c(1.2, 0.7, 0, -1.1)
as.logical(vec) # ゼロ以外TRUE
#> [1]  TRUE  TRUE FALSE  TRUE
as.integer(vec) # 小数点以下切り捨て
#> [1]  1  0  0 -1
as.character(vec)
#> [1] "1.2"  "0.7"  "0"    "-1.1"
```

# データ型の変換

```
as.double(c(T, T, F, T))
#> [1] 1 1 0 1
as.double(c("b", "a", "c"))
#> [1] NA NA NA
#> Warning message: NAs introduced
#> by coercion as.character
```

# データ型の変換

```
c("b", "a", "c") %>%
  factor() %>%
  as.double()
#> [1] 2 1 3
```

# データ型の変換

```
c("b", "a", "c") %>%
  factor()
#> [1] b a c
#> Levels: a b c
```

辞書順

# データ型の変換

```
c("b", "A", "c") %>%
  factor()
#> [1] b A c
#> Levels: A b c
```

辞書順

# データ型の変換

```
c("b", "A", "c") %>%
  factor() %>%
  levels()
#> [1] "A" "b" "c"
```

水準の抽出

# データ型の変換

```
c("Feb", "Jan", "Mar", "Apr") %>%
  factor()
#> [1] Feb Jan Mar Apr
#> Levels: Apr Feb Jan Mar
```

辞書順

# データ型の変換

```
c("Feb", "Jan", "Mar", "Apr") %>%
  factor() %>%
  as.double()
#> [1] 2 3 4 1
```

辞書順

# データ型の変換

```
levs <-
  c("Jan", "Feb", "Mar", "Apr")


c("Feb", "Jan", "Mar", "Apr") %>%
  factor(levels = levs) %>%
  as.double()
#> [1] 2 1 3 4
```

指定順

```
> df_wide
# A tibble: 5 × 4
  患者  検査1  検査2  検査3
  <chr> <dbl> <lgl> <dbl>
1 患者1 0.118 TRUE   178.
2 患者2 0.762 FALSE  113.
3 患者3 0.142 TRUE   181.
4 患者4 0.767 TRUE   190.
5 患者5 0.842 FALSE  193.
```

logical型

戻せる？

```
# A tibble: 5 × 4
  患者   検査1  検査2 検査3
  <chr> <dbl> <dbl> <dbl>
1 患者1 0.118     1  178.
2 患者2 0.762     0  113.
3 患者3 0.142     1  181.
4 患者4 0.767     1  190.
5 患者5 0.842     0  193.
```

double型

# dplyr::mutate()関数

```
df_long %>%
  pivot_wider(
    names_from = "検査",
    values_from = "value"
  ) %>%
  mutate(検査2 = as.logical(検査2))
```

# データ操作 data manipulation

# データ操作 data manipulation

# データ操作 data manipulation

# vignette("dplyr")

## Single table verbs

dplyr aims to provide a function for each basic verb of data manipulation. These verbs can be organised into three categories based on the component of the dataset that they work with:

- Rows:
  - `filter()` chooses rows based on column values.
  - `slice()` chooses rows based on location.
  - `arrange()` changes the order of the rows.
- Columns:
  - `select()` changes whether or not a column is included.
  - `rename()` changes the name of columns.
  - `mutate()` changes the values of columns and creates new columns.
  - `relocate()` changes the order of the columns.
- Groups of rows:
  - `summarise()` collapses a group into a single row.

```
vignette("dplyr")
```

dplyr aims to provide a function
for each **basic verb** of data manipulation.

{dplyr}パッケージの目的は、
データ操作の**基本動詞**のそれぞれに
対応した関数を提供することだ。

# Grammar of data manipulation

By **constraining** your options, it helps you think about your data manipulation challenges.

Introduction to dplyr
https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

# Grammar of data manipulation

選択肢を制限することで、
試みたいデータ操作について
シンプルに捉えられるように
なります。

Introduction to dplyr
https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

# データ操作 data manipulation

## 基本動詞ランキング【必修】

1. **mutate()**関数: 列の操作
2. **filter()**関数: 行の選択
3. **select()**関数: 列の選択
4. **group_by()**関数: グループ化
5. **summarise()**関数: 要約

# mutate() # 列の操作

```
mutate(dat, C = fun(A, B))
```

# mutate() # 列の操作

```
dat %>% mutate(C = fun(A, B))
```

## スクリプト

```
## packages ----
library(tidyverse)

## parameters ----
r <- 2
x_center <- 1
y_center <- 3

## data ----
df <-
  tibble(
    theta = seq(0, 2 * pi, length = 361),
    X = r * cos(theta) + x_center,
    Y = r * sin(theta) + y_center
  )

## visualization ----
ggplot(data = df) +
  aes(x = X, y = Y) +
  geom_path() +
  coord_fixed()
```

```r
df <-
  data.frame(
    theta = seq(0, 2 * pi, length = 361),
    X = r * cos(theta) + x_center,
    Y = r * sin(theta) + y_center
  )
#> Error: object 'theta' not found
```

```r
df <-
  data.frame(
    theta = seq(0, 2 * pi, length = 361),
    X = r * cos(theta) + x_center,
    Y = r * sin(theta) + y_center
  )
#> Error: object 'theta' not found
```

```r
df <-
  tibble(
    theta = seq(0, 2 * pi, length = 361),
    X = r * cos(theta) + x_center,
    Y = r * sin(theta) + y_center
  )
# tibble()関数では、遅延評価される
```

```r
df <-
  data.frame(
    theta = seq(0, 2 * pi, length = 361),
    X = r * cos(theta) + x_center,
    Y = r * sin(theta) + y_center
  )
#> Error: object 'theta' not found
```

```r
df <-
  data.frame( # 一旦theta列を作っておいて
    theta = seq(0, 2 * pi, length = 361)
  ) %>%
  mutate( # それを使って後から加工する
    X = r * cos(theta) + x_center,
    Y = r * sin(theta) + y_center
  )
```

# filter() # 行の選択

```
dat %>% filter(tag %in% c(1, 3, 5))
```

# ブール演算子 Boolean Algebra

# equal to
A == B

# not equal to
A != B

# or
A | B

# and
A & B

# is A in B?
A %in% B

George Boole
1815 - 1864

# ブール演算子 Boolean Algebra

```
# is A in B?
"a" != "b"
```

```
[1] TRUE
```

```
# is A in B?
1 %in% 10:100
```

```
[1] FALSE
```

# George Boole
# 1815 - 1864

Mathematician

&

Philosopher

A Class-Room Introduction to Logic
https://niyamaklogic.wordpress.com/category/laws-of-thoughts/

# select() # 列の選択

dat %>% select(tag, B)

| tag | A | B | C |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

→

| tag | B |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |

# 列選択お助け関数群

## Select help functions

```
starts_with("s") # 先頭一致
ends_with("s") # 末尾一致
contains("se") # 部分一致
matches("^.e") # 正規表現
any_of(c("tag", "B")) # 複数列
everything() # 全ての列
```

```
dat %>% select(tag, B)

dat %>% select("tag", "B")
```

select()関数は引数に文字列を取れる

あとで(たっぷり)話します。

# データ操作 data manipulation

## 基本動詞ランキング【必修】

1. **mutate()**関数: 列の操作
2. **filter()**関数: 行の選択
3. **select()**関数: 列の選択
4. **group_by()**関数: グループ化
5. **summarise()**関数: 要約

列名に日本語が入るとイライラ💢するので作り直し

```
df_long <-
  tibble(
    ID = rep(str_c("患者", 1:5), each = 3),
    exam = rep(str_c("検査", 1:3), 5),
    value = df_long$value
  )
```

# group_by() # グループ化

```
df_long %>% head()
#> # A tibble: 6 × 3
#>   ID    exam    value
#>   <chr> <chr>   <dbl>
#> 1 患者1 検査1   0.615
#> 2 患者1 検査2   1
#> 3 患者1 検査3 146.
#> 4 患者2 検査1   0.287
#> 5 患者2 検査2   0
#> 6 患者2 検査3 142.
```

# group_by() # グループ化

```
df_long %>%
  group_by(exam)
#> # A tibble: 15 × 3
#> # Groups:   exam [3]
#>   ID    exam    value
#>   <chr> <chr>   <dbl>
#> 1 患者1 検査1   0.118
#> 2 患者1 検査2   1
#> 3 患者1 検査3 178.
#> 4 患者2 検査1   0.762
#> 5 患者2 検査2   0
```

# group_by() # グループ化

# summarise() # 要約

```
df_long %>%
  group_by(exam) %>%
  summarise(avg = mean(value))
#> # A tibble: 3 × 2
#>   exam      avg
#>   <chr>   <dbl>
#> 1 検査1   0.405
#> 2 検査2   0.6
#> 3 検査3 148.
```

# group_by() # グループ化

# summarise() # 要約

```
df_long %>%
  group_by(ID) %>%
  summarise(avg = mean(value))
```

```
#> # A tibble: 5 × 2
#>   ID      avg
#>   <chr> <dbl>
#> 1 患者1  49.3
#> 2 患者2  47.3
#> 3 患者3  44.9
#> 4 患者4  43.7
#> 5 患者5  63.0
```

# group_by() # グループ化

# summarise() # 要約

```
df_long %>%
#  group_by(ID) %>%
  summarise(avg = mean(value))
#> # A tibble: 1 × 1
#>     avg
#>   <dbl>
#> 1  49.6
```

```r
df_long %>%
  group_by(exam) %>%
  summarise(
    avg = mean(value),
    sd = sd(value)
  )
```

```
# A tibble: 3 × 3
  exam      avg     sd
  <chr>   <dbl>  <dbl>
1 検査1   0.405  0.239
2 検査2   0.6    0.548
3 検査3 148.    23.7
```

```
df_long %>%
  group_by(exam) %>%
  summarise(
    avg = mean(value),
    sd = sd(value)
  )
```

summarise()関数を使わずに
同じ結果を得るにはどうする？

```r
df_long %>%
  group_by(exam) %>%
  mutate(
    avg = mean(value),
    sd = sd(value)
  )
```

```
# A tibble: 15 × 5
# Groups:   exam [3]
   ID    exam    value     avg     sd
   <chr> <chr>   <dbl>   <dbl>  <dbl>
 1 患者1 検査1   0.615   0.405  0.239
 2 患者1 検査2   1       0.6    0.548
 3 患者1 検査3 146.    148.    23.7
 4 患者2 検査1   0.287   0.405  0.239
 5 患者2 検査2   0       0.6    0.548
```

```r
df_long %>%
  group_by(exam) %>%
  mutate(
    avg = mean(value),
    sd = sd(value)
  ) %>%
  arrange(exam) # 指定列の辞書順で並び替え
```

```
# A tibble: 15 × 5
# Groups:   exam [3]
   ID    exam    value     avg     sd
   <chr> <chr>   <dbl>   <dbl>  <dbl>
 1 患者1 検査1   0.615   0.405  0.239
 2 患者2 検査1   0.287   0.405  0.239
 3 患者3 検査1   0.708   0.405  0.239
 4 患者4 検査1   0.220   0.405  0.239
 5 患者5 検査1   0.193   0.405  0.239
 6 患者1 検査2   1       0.6    0.548
 7 患者2 検査2   0       0.6    0.548
```

```r
df_long %>%
  group_by(exam) %>%
  mutate(
    avg = mean(value),
    sd = sd(value)
  ) %>%
  arrange(exam) %>% # 並び替え
  select(exam, avg, sd) # 列選択
```

```
# A tibble: 15 × 3
# Groups:   exam [3]
   exam      avg     sd
   <chr>   <dbl>  <dbl>
 1 検査1   0.405  0.239
 2 検査1   0.405  0.239
 3 検査1   0.405  0.239
 4 検査1   0.405  0.239
 5 検査1   0.405  0.239
 6 検査2   0.6    0.548
```

```r
df_long %>%
  group_by(exam) %>%
  mutate(
    avg = mean(value),
    sd = sd(value)
  ) %>%
  arrange(exam) %>% # 並び替え
  select(!c(ID, value)) # 列選択
```

```
# A tibble: 15 × 3
# Groups:   exam [3]
   exam      avg     sd
   <chr>   <dbl>  <dbl>
 1 検査1   0.405  0.239
 2 検査1   0.405  0.239
 3 検査1   0.405  0.239
 4 検査1   0.405  0.239
 5 検査1   0.405  0.239
 6 検査2   0.6    0.548
```

```r
df_long %>%
  group_by(exam) %>%
  mutate(
    avg = mean(value),
    sd = sd(value)
  ) %>%
  arrange(exam) %>% # 並び替え
  select(!c(ID, value)) %>% # 列選択
  distinct() # 重複列の除去
```

```
# A tibble: 3 × 3
# Groups:   exam [3]
  exam      avg     sd
  <chr>   <dbl>  <dbl>
1 検査1   0.405  0.239
2 検査2   0.6    0.548
3 検査3 148.    23.7
```

```r
df_long %>%
  group_by(exam) %>%
  mutate(
    avg = mean(value),
    sd = sd(value)
  ) %>%
  arrange(exam) %>% # 並び替え
  select(!c(ID, value)) %>% # 列選択
  distinct() %>% # 重複列の除去
  ungroup() # グループ化の解除
```

```
# A tibble: 3 × 3
  exam      avg     sd
  <chr>   <dbl>  <dbl>
1 検査1   0.405  0.239
2 検査2   0.6    0.548
3 検査3 148.    23.7
```

```
df_long %>%
  group_by(exam) %>%
  mutate(
    avg = mean(value),
    sd = sd(value)
  ) %>%
#  arrange(exam) %>% # 並び替え
  select(!c(ID, value)) %>% # 列選択
  distinct() %>% # 重複列の除去
  ungroup() # グループ化の解除
```

出力結果には影響しない

```
# A tibble: 3 × 3
  exam      avg     sd
  <chr>   <dbl>  <dbl>
1 検査1   0.405  0.239
2 検査2   0.6    0.548
3 検査3 148.    23.7
```

```r
df_long %>%
  group_by(exam) %>%
  mutate(
    avg = mean(value),
    sd = sd(value)
  ) %>%
  select(!c(ID, value)) %>% # 列選択
  distinct() %>% # 重複列の除去
  ungroup() # グループ化の解除
```

```r
df_long %>%
  group_by(exam) %>%
  summarise(
    avg = mean(value),
    sd = sd(value)
  )
```

# データ操作 data manipulation

## 基本動詞ランキング【必修】

1. **mutate()**関数: 列の操作
2. **filter()**関数: 行の選択
3. **select()**関数: 列の選択
4. **group_by()**関数: グループ化
5. **summarise()**関数: 要約

```
dat %>% select(tag, B)

dat %>% select("tag", "B")
```

select()関数は引数に文字列を取れる

あとで(たっぷり)話します。

```
dat %>% select(tag, B)

dat %>% select("tag", "B")
```

select()関数は引数に文字列を取れる

**非標準評価** NSE
(Non-Standard Evaluation)

```
df_long %>% group_by(exam)
#> # A tibble: 15 × 3
#> # Groups:   exam [3]
#>   ID    exam    value
#>   <chr> <chr>   <dbl>
#> 1 患者1 検査1   0.615
#> 2 患者1 検査2   1
#> 3 患者1 検査3 146.
#> 4 患者2 検査1   0.287
#> 5 患者2 検査2   0
#> 6 患者2 検査3 142.
...

exam
# Error: object 'exam' not found
```

- examというオブジェクトは定義されていないので見つからない。
- group_by()関数は勝手に引数のdf_longデータの中からexamという名前の列を探してくれる。

```
df_long %>% group_by("exam")
#> # A tibble: 15 × 4
#> # Groups:   "exam" [1]
#>   ID    exam    value `"exam"`
#>   <chr> <chr>   <dbl> <chr>
#> 1 患者1 検査1   0.615 exam
#> 2 患者1 検査2   1     exam
#> 3 患者1 検査3 146.    exam
#> 4 患者2 検査1   0.287 exam
#> 5 患者2 検査2   0     exam
#> 6 患者2 検査3 142.    exam
...
```

- “exam”という列が新たに定義されデータ全体が1つの水準でグループ化される
- “exam”という明示的な文字列はdf_long$examとは解釈されずに別のオブジェクトとして認識される

```
df_long %>% mutate(x = "exam")
#> # A tibble: 15 × 4
#>   ID    exam    value x
#>   <chr> <chr>   <dbl> <chr>
#> 1 患者1 検査1   0.615 exam
#> 2 患者1 検査2   1     exam
#> 3 患者1 検査3 146.    exam
#> 4 患者2 検査1   0.287 exam
...
```

文字型オブジェクトとして認識される

```
df_long %>% mutate(x = exam)
#> # A tibble: 15 × 4
#> ID    exam    value x
#>   <chr> <chr>   <dbl> <chr>
#> 1 患者1 検査1   0.615 検査1
#> 2 患者1 検査2   1     検査2
#> 3 患者1 検査3 146.    検査3
#> 4 患者2 検査1   0.287 検査1
...
```

df_long$examとして認識される

# {dplyr}パッケージの非標準評価

## 1. Data masking

.data引数に指定されたデータフレームの列名を関数内で参照する際に、データフレーム名$を省略できる。

```
df_long %>% filter(ID == "患者1")
#> # A tibble: 3 × 3
#>  ID    exam    value
#>  <chr> <chr>   <dbl>
#> 1 患者1 検査1   0.615
#> 2 患者1 検査2   1
#> 3 患者1 検査3 146.
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価

## 1. Data masking

.data引数に指定されたデータフレームの列名を関数内で参照する際に、データフレーム名$を省略できる。

```
filter(.data = df_long, ID == "患者1")
```

```
df_long %>% filter(ID == "患者1")
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価

## 1. Data masking

.data引数に指定されたデータフレームの列名を関数内で参照する際に、データフレーム名$を省略できる。

```
df_long %>% filter(ID == "患者1")
```

**標準評価**

```
df_long[ID == "患者1",]
#> Error: object 'ID' not found
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価

## 1. Data masking

.data引数に指定されたデータフレームの列名を関数内で参照する際に、データフレーム名$を省略できる。

```
df_long %>% filter(ID == "患者1")
```

**標準評価**

オブジェクト名[行番号, 列番号]の応用版

```
df_long[df_long$ID == "患者1",]

df_long$ID == "患者1"
#>  [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
#> [11] FALSE FALSE FALSE FALSE FALSE
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価

## 1. Data masking

.data引数に指定されたデータフレームの列名を関数内で参照する際に、データフレーム名$を省略できる。

```
df_long %>% filter(ID == "患者1")
```

### 標準評価

```
a <- df_long$ID == "患者1"
#> > a
#>  [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
#> [11] FALSE FALSE FALSE FALSE FALSE

df_long[a,]
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価

## 1. Data masking

.data引数に指定されたデータフレームの列名を関数内で参照する際に、データフレーム名$を省略できる。

Data maskingが適用される関数

| | |
|---|---|
| `mutate()` | `filter()` |
| `group_by()` | `summarise()` |
| `arrange()` | `count()` |

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価

## 1. Data masking

.data引数に指定されたデータフレームの列名を関数内で参照する際に、データフレーム名$を省略できる。

## 2. Tidy selection

```
select()        relocate()
rename()        across()
pull()
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価

## 2. Tidy selection

列選択に係る非標準評価のこと

```
df_long %>% rename(a = ID)
#> # A tibble: 15 × 3
#>   a     exam    value
#>   <chr> <chr>   <dbl>
#> 1 患者1 検査1   0.615
#> 2 患者1 検査2   1
#> 3 患者1 検査3 146.
```

data maskingと同じように
data.frame名を省略できる

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価

## 2. Tidy selection

列選択に係る非標準評価のこと

```
df_long %>% rename(a = "ID")
#> # A tibble: 15 × 3
#>   a     exam    value
#>   <chr> <chr>   <dbl>
#> 1 患者1 検査1   0.615
#> 2 患者1 検査2   1
#> 3 患者1 検査3 146.
```

文字列に一致する列を
探してくれる

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価

## 2. Tidy selection

列選択に係る非標準評価のこと

```
df_long %>% rename(a = contains("ID"))
#> # A tibble: 15 × 3
#>   a     exam    value
#>   <chr> <chr>   <dbl>
#> 1 患者1 検査1   0.615
#> 2 患者1 検査2   1
#> 3 患者1 検査3 146.
```

列選択お助け関数群
{tidyselect}が裏で動いてる

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価

## 1. Data masking

.data引数に指定されたデータフレームの列名を関数内で参照する際に、データフレーム名$を省略できる。

## 2. Tidy selection

列選択に係る非標準評価のこと

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価の困りどころ

これはいいんだけど、

```
df_long %>% mutate(x = ID)
#> # A tibble: 15 × 4
#>   ID    exam    value x
#>   <chr> <chr>   <dbl> <chr>
#> 1 患者1 検査1   0.333 患者1
#> 2 患者1 検査2   1     患者1
#> 3 患者1 検査3 195.    患者1
#> ...
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価の困りどころ

関数の中でNSEを使おうとすると...

```
f <- function(data, column){
  data %>% mutate(x = column)
}

f(data = df_long, column = ID)

#> Error in `mutate()`:
#> ℹ In argument: `x = column`.
#> Caused by error:
#>! object 'ID' not found
#> Run `rlang::last_trace()` to see where the error occurred.
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価の困りどころ

関数の中でNSEを使おうとすると...

```
f <- function(data, column){
  data %>% mutate(x = column)
}

f(data = df_long, column = "ID")
```

```
#> # A tibble: 15 × 4
   ID    exam    value x
   <chr> <chr>   <dbl> <chr>
 1 患者1 検査1   0.333 ID
 2 患者1 検査2   1     ID
 3 患者1 検査3 195.    ID ...
```

違う！
こうじゃない

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価の困りどころ

```
f_00 <- function(data, column){
  data %>% mutate(x = {{ column }})
}
```

この中のはdata masking として解釈される

```
f_00(data = df_long, column = ID)
```

```
# A tibble: 15 × 4
   ID    exam    value x
   <chr> <chr>   <dbl> <chr>
 1 患者1 検査1   0.333 患者1
 2 患者1 検査2   1     患者1
 3 患者1 検査3 195.    患者1
 4 患者2 検査1   0.555 患者2
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価の困りどころ

```
f_01 <- function(data, new_name, column){
  data %>%
    mutate("{{ new_name }}" := {{ column }})
}
```

列名(左辺)を新しく定義したい時は
定義演算子:=を使う。

```
f_01(data = df_long, new_name = aa, column = ID)
```

```
# A tibble: 15 × 4
   ID    exam    value aa
   <chr> <chr>   <dbl> <chr>
 1 患者1 検査1   0.333 患者1
 2 患者1 検査2   1     患者1
 3 患者1 検査3 195.    患者1
 4 患者2 検査1   0.555 患者2
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

**やってみよう**

次の結果を返すf_03()関数を定義してください

```
df_long %>% f_03(exam, value)
#> # A tibble: 3 × 3
#>   exam  avg_value sd_value
#>   <chr>     <dbl>    <dbl>
#> 1 検査1     0.349    0.126
#> 2 検査2     0.6      0.548
#> 3 検査3   165.      26.9
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

# {dplyr}パッケージの非標準評価の困りどころ

```
f_03 <- function(data, grp, column){
  data %>%
    group_by({{ grp }}) %>%
    summarise(
      "avg_{{ column }}" := mean({{ column }}),
      "sd_{{ column }}" := sd({{ column }})
    )
}
```

rf. **Programming with dplyr,** https://dplyr.tidyverse.org/articles/programming.html

```r
palmerpenguins::penguins %>%
  na.omit() %>% # NAを含む行の除去
  f_03(grp = species, column = bill_length_mm)
#> # A tibble: 3 × 3
#>  species  avg_bill_length_mm sd_bill_length_mm
#>  <fct>                 <dbl>             <dbl>
#> 1 Adelie                38.8              2.66
#> 2 Chinstr…              48.8              3.34
#> 3 Gentoo                47.6              3.11
```

ちゃんと使えます！

# データ操作 data manipulation

# Grammar of data manipulation

選択肢を制限することで、
試みたいデータ操作について
シンプルに捉えられるように
なります。

Introduction to dplyr
https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

# ■ Rを始めよう

- 基礎知識
- データの読み書き
- レポーティングの基礎
- **データの操作** ⬅
- データの可視化

# ■ Rを始めよう

- 基礎知識
- データの読み書き
- レポーティングの基礎
- データの操作
- **データの可視化** ⬅

# 写像 mapping

ある情報の集合の要素を、別の情報の集合のただ1つの要素に対応づけるプロセス

**【写像】**
**ある集合の要素を他の集合のただ1つの要素に対応づける規則**

実存

写像

情報

(画像)

赤色 (性質)

りんご (生物学的分類)

果物 (商業的分類)

1 (数量)

実存
情報
写像
(画像)
赤色 (性質)
りんご (生物学的分類)
果物 (商業的分類)
1 (数量)
チャネル
channel

# データ可視化

実存

↓ 写像（観察）

データ

↓ 写像（データ可視化）

グラフ

**審美的チャネル**

aesthetic channels

| $X$ | $Y$ |
|---|---|
| $x_1$ | $y_1$ |
| $x_2$ | $y_2$ |

data

mapping

# データ可視化

| $X$ | $Y$ |
|---|---|
| $x_1$ | $y_1$ |
| $x_2$ | $y_2$ |

写像
mapping

# データ可視化

| $X$ | $Y$ |
|---|---|
| $x_1$ | $y_1$ |
| $x_2$ | $y_2$ |

写像
mapping

**審美的チャネル**
aesthetic channels

x axis, y axis, color, fill, shape, linetype, alpha…

# データ可視化

| $X$ | $Y$ |
|---|---|
| $x_1$ | $y_1$ |
| $x_2$ | $y_2$ |

写像
mapping

{ggplot2}による作図

```
ggplot(data = my_data) +
  aes(x = X, y = Y)) +
  goem_point()
```

**審美的チャネル**
aesthetic channels

x axis, y axis, color, fill, shape, linetype, alpha...

```r
library(palmerpenguins)

penguins
#> # A tibble: 344 × 8
#>   species island    bill_length_mm bill_depth_mm flipper_length_mm
#>   <fct>   <fct>              <dbl>         <dbl>             <int>
#> 1 Adelie  Torgersen           39.1          18.7               181
#> 2 Adelie  Torgersen           39.5          17.4               186
#> 3 Adelie  Torgersen           40.3          18                 195
#> 4 Adelie  Torgersen           NA            NA                  NA
#> 5 Adelie  Torgersen           36.7          19.3               193
#> 6 Adelie  Torgersen           39.3          20.6               190
#> 7 Adelie  Torgersen           38.9          17.8               181
#> 8 Adelie  Torgersen           39.2          19.6               195
#> 9 Adelie  Torgersen           34.1          18.1               193
#> 10 Adelie  Torgersen           42            20.2               190
#> # ℹ 334 more rows
#> # ℹ 3 more variables: body_mass_g <int>, sex <fct>, year <int>
#> # ℹ Use `print(n = ...)` to see more rows
```

## License

Data are available by **CC-0** license in accordance with the Palmer Station LTER Data Policy and the LTER Data Access Policy for Type I data.

https://allisonhorst.github.io/palmerpenguins/

```r
library(tidyverse)
library(palmerpenguins)

df <-
  penguins %>%
  na.omit() # NAを含む行の除去
```

```
#> # A tibble: 333 × 8
#>   species island    bill_length_mm bill_depth_mm flipper_length_mm
#>    <fct>   <fct>              <dbl>         <dbl>             <int>
#> 1 Adelie  Torgersen           39.1          18.7               181
#> 2 Adelie  Torgersen           39.5          17.4               186
#> 3 Adelie  Torgersen           40.3          18                 195
...
```

```r
library(tidyverse)
library(palmerpenguins)

df
#> # A tibble: 333 × 8
#>   species island    bill_length_mm bill_depth_mm flipper_length_mm
#>    <fct>   <fct>              <dbl>         <dbl>             <int>
#> 1 Adelie  Torgersen           39.1          18.7               181
#> 2 Adelie  Torgersen           39.5          17.4               186
#> 3 Adelie  Torgersen           40.3          18                 195
...
```

```r
ggplot() +
  geom_point(
    data = df,
    mapping = aes(x = bill_length_mm,
                  y = flipper_length_mm)
  )
```

```r
ggplot() +
  geom_point(
    data = df,
    mapping = aes(x = bill_length_mm,
                  y = flipper_length_mm)
  )
```

描画開始を宣言

+で繋ぐ

グラフの種類に合わせて様々な
geom_*()関数が用意されている

data引数にはdata.frame(もしくはtibble)を指定

```
ggplot() +
  geom_point(
    data = df,
    mapping = aes(x = bill_length_mm,
                  y = flipper_length_mm)
  )
```

非標準評価で列名を指定する

aes()関数の引数は審美的チャネル

# データ可視化: 初めての{ggplot2}

```
ggplot() +
  geom_point(
    data = df,
    mapping = aes(x = bill_length_mm,
                  y = flipper_length_mm,
                  color = species)
  )
```

aes()関数で指定できる審美的チャネルは色々ある

離散量の場合、水準に従ってfactor扱いになる。

# データ可視化: 初めての{ggplot2}

```
ggplot() +
  geom_point(
    data = df,
    mapping = aes(x = bill_length_mm,
                  y = flipper_length_mm,
                  color = body_mass_g)
  )
```

aes()関数で指定できる審美的チャネルは色々ある

連続量の場合、水準に従ってグラデーションがつく

# データ可視化: 初めての{ggplot2}

```
ggplot() +
  geom_point(
    data = df,
    mapping = aes(x = bill_length_mm,
                  y = flipper_length_mm,
                  color = body_mass_g)
  ) +
  geom_path(
    data = df,
    mapping = aes(x = bill_length_mm,
                  y = flipper_length_mm),
    color = "red",
    alpha = 0.5
  )
```

データ点をプロットする

データ点をつなぐ線を描く

dfに依存しない審美的チャネルは
aes()関数の外側で指定する

透明度

# データ可視化: 初めての{ggplot2}

```
ggplot() +
  geom_point(
    data = df,
    mapping = aes(x = bill_length_mm,
                  y = flipper_length_mm
  ) +
  geom_
    d
    m
    c
    a
  )
```

データ点をプロットする

# データ可視化: 初めての{ggplot2}

共通項を繰り返し描くのは手間

```
ggplot() +
  geom_point(
    data = df,
    mapping = aes(x = bill_length_mm,
                  y = flipper_length_mm,
                  color = body_mass_g)
  ) +
  geom_path(
    data = df,
    mapping = aes(x = bill_length_mm,
                  y = flipper_length_mm),
    color = "red",
    alpha = 0.5
  )
```

# データ可視化: 初めての{ggplot2}

ggplot()関数に括り出す

```
ggplot(
  data = df,
  mapping = aes(x = bill_length_mm,
                y = flipper_length_mm)
  ) +
  geom_point(
    mapping = aes(color = body_mass_g)
  ) +
  geom_path(
    color = "red",
    alpha = 0.5
  )
```

個別の指定は残す

# データ可視化: 初めての{ggplot2}

```
ggplot(data = df) +
  aes(
    x = bill_length_mm,
    y = flipper_length_mm
  ) +
  geom_point(
    mapping = aes(color = body_mass_g)
  ) +
  geom_path(
    color = "red",
    alpha = 0.5
  )
```

**aes()関数はggplot()関数の外側における**

# データ可視化: 初めての{ggplot2}

```
ggplot(data = df) +
  aes(
    x = bill_length_mm,
    y = flipper_length_mm
  ) +
  geom_point(
    mapping = aes(color = body_mass_g)
  ) +
  geom_path(
    color = "red",
    alpha = 0.5
  )
```

# データ可視化: 初めての{ggplot2}

```
ggplot(data = df) +
  aes(
    x = bill_length_mm,
    y = flipper_length_mm
  ) +
  geom_point(
    mapping = aes(color = body_mass_g)
  ) +
  geom_path(
    color = "red",
    alpha = 0.5
  )
```

# データ可視化: 初めての{ggplot2}

```
ggplot(data = df) +
  aes(
    x = bill_length_mm,
    y = flipper_length_mm
  ) +
  geom_path(
    color = "red",
    alpha = 0.5
  ) +
  geom_point(
    mapping = aes(co
  )
```

表記順＝重ね順

# データ可視化: 初めての{ggplot2}

```
g <-
  ggplot(data = df) +
  aes(x = bill_length_mm,
      y = flipper_length_mm) +
  geom_path(
    color = "red",
    alpha = 0.5
  ) +
  geom_point(
    mapping = aes(color = body_mass_g)
  )
```

```
ggsave(filename = "fig/fig_penguins.png",
       plot = g, width = 5, height = 3,
       dpi = 300, units = "in")
```

解像度(dot per inch)

サイズの基本はinch。cm, mm, pxも指定可能。

# gemo_*()関数群

rf. https://www.rstudio.com/resources/cheatsheets/

## GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

**a + geom_blank()**
(Useful for expanding limits)

**b + geom_curve(**aes(yend = lat + 1, xend=long+1,curvature=z)**)** - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

**a + geom_path(**lineend="butt", linejoin="round", linemitre=1**)**
x, y, alpha, color, group, linetype, size

**a + geom_polygon(**aes(group = group)**)**
x, y, alpha, color, fill, group, linetype, size

**b + geom_rect(**aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)**)** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

**a + geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900)**)** - x, ymax, ymin, alpha, color, fill, group, linetype, size

## LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

**b + geom_abline(**aes(intercept=0, slope=1)**)**
**b + geom_hline(**aes(yintercept = lat)**)**
**b + geom_vline(**aes(xintercept = long)**)**

**b + geom_segment(**aes(yend=lat+1, xend=long+1)**)**
**b + geom_spoke(**aes(angle = 1:1155, radius = 1)**)**

## ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

**c + geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size

**c + geom_density(**kernel = "gaussian"**)**
x, y, alpha, color, fill, group, linetype, size, weight

**c + geom_dotplot()**
x, y, alpha, color, fill

**c + geom_freqpoly()** x, y, alpha, color, group, linetype, size

**c + geom_histogram(**binwidth = 5**)** x, y, alpha, color, fill, linetype, size, weight

**c2 + geom_qq(**aes(sample = hwy)**)** x, y, alpha, color, fill, linetype, size, weight

### discrete

d <- ggplot(mpg, aes(fl))

**d + geom_bar()**
x, alpha, color, fill, linetype, size, weight

## TWO VARIABLES

### continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

**e + geom_label(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

**e + geom_jitter(**height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size

**e + geom_point()**, x, y, alpha, color, fill, shape, size, stroke

**e + geom_quantile()**, x, y, alpha, color, group, linetype, size, weight

**e + geom_rug(**sides = "bl"**)**, x, y, alpha, color, linetype, size

**e + geom_smooth(**method = lm**)**, x, y, alpha, color, fill, group, linetype, size, weight

**e + geom_text(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**, x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

### discrete x , continuous y

f <- ggplot(mpg, aes(class, hwy))

**f + geom_col()**, x, y, alpha, color, fill, group, linetype, size

**f + geom_boxplot()**, x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

**f + geom_dotplot(**binaxis = "y", stackdir = "center"**)**, x, y, alpha, color, fill, group

**f + geom_violin(**scale = "area"**)**, x, y, alpha, color, fill, group, linetype, size, weight

### discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

**g + geom_count()**, x, y, alpha, color, fill, shape, size, stroke

### continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

**h + geom_bin2d(**binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

**h + geom_density2d()**
x, y, alpha, colour, group, linetype, size

**h + geom_hex()**
x, y, alpha, colour, fill, size

### continuous function

i <- ggplot(economics, aes(date, unemploy))

**i + geom_area()**
x, y, alpha, color, fill, linetype, size

**i + geom_line()**
x, y, alpha, color, group, linetype, size

**i + geom_step(direction = "hv")**
x, y, alpha, color, group, linetype, size

### visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

**j + geom_crossbar(**fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

**j + geom_errorbar()**, x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

**j + geom_linerange()**
x, ymin, ymax, alpha, color, group, linetype, size

**j + geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

### maps

data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

**k + geom_map(**aes(map_id = state), map = map**)**
**+ expand_limits(**x = map$long, y = map$lat**)**,
map_id, alpha, color, fill, linetype, size

## THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))

**l + geom_contour(aes(z = z))**
x, y, z, alpha, colour, group, linetype, size, weight

**l + geom_raster(**aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill

**l + geom_tile(**aes(fill = z)**)**, x, y, alpha, color, fill, linetype, size, width

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...

ggplot(data = anscombe)
```

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...

ggplot(data = anscombe) +
  geom_point(mapping = aes(x = x1, y = y1),
             color = "blue")
```

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...

ggplot(data = anscombe) +
  geom_point(mapping = aes(x = x1, y = y1),
             color = "blue") +
  geom_point(mapping = aes(x = x2, y = y2),
             color = "red") +
  geom_point(mapping = aes(x = x3, y = y3),
             color = "green") +
  geom_point(mapping = aes(x = x4, y = y4),
             color = "black")
```

# データ可視化: 複数の系列

実存

↓ 写像（観察）

データ

↓ 写像（データ可視化）

グラフ

**審美的チャネル**
aesthetic channels

raw data

↓ 変形

**可視化に適したdata形式**

図の1つの審美的チャネルが
データの1つの変数(列)に対応

↓ 写像

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...
```

```
ggplot(data = anscombe_long) +
  aes(x = X, y = Y, color = key) +
  geom_point()
```

描画コードを
こう書きたい

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...
```

```
anscombe_long
#>   key       x     y
#> 1 1        10  8.04
#> 2 2        10  9.14
#> 3 3        10  7.46
#> ...
```

それに合う
データが必要

```
ggplot(data = anscombe_long) +
  aes(x = X, y = Y, color = key) +
  geom_point()
```

描画コードを
こう書きたい

# データ可視化: 複数の系列

**やってみよう**

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...
```

データの操作
横型→縦型の変形

```
anscombe_long
#>   key       x     y
#> 1 1        10  8.04
#> 2 2        10  9.14
#> 3 3        10  7.46
#> ...
```

それに合う
データが必要

```
ggplot(data = anscombe_long) +
  aes(x = X, y = Y, color = key) +
  geom_point()
```

描画コードを
こう書きたい

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...
```

```
anscombe_long <-
  anscombe %>%
  pivot_longer(
    cols = ***,
    names_to = ***,
    values_to = ***,
  )
```

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...
```

```
anscombe_long <-
  anscombe %>%
  pivot_longer(
    cols = everything(),
    names_to = ***,
    values_to = ***,
  )
```

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...
```

```
anscombe_long <-
  anscombe %>%
  pivot_longer(
    cols = everything(),
    names_to = "param",
    values_to = "value",
  )
```

# データ可視化: 複数の系列

```
anscombe_long <-
  anscombe %>%
  pivot_longer(
    cols = everything(),
    names_to = "param",
    values_to = "value",
  )
```

```
# A tibble: 88 × 2
   param value
   <chr> <dbl>
 1 x1    10
 2 x2    10
 3 x3    10
 4 x4     8
 5 y1     8.04
 6 y2     9.14
...
# ℹ 78 more rows
# ℹ Use `print(n = ...)` to see more rows
```

# データ可視化: 複数の系列

```
anscombe_long <-
  anscombe %>%
  pivot_longer(
    cols = everything(),
    names_to = "param",
    values_to = "value",
  ) %>%
  mutate(
    key = str_sub(param, start = 2, end = 2),
    param = str_sub(param, start = 1, end = 1)
  )
```

```
# A tibble: 88 × 3
   param value key
   <chr> <dbl> <chr>
 1 x     10    1
 2 x     10    2
 3 x     10    3
...
```

# データ可視化: 複数の系列

```
anscombe_long <-
  anscombe %>%
  pivot_longer(
    cols = everything(),
    names_to = "param",
    values_to = "value",
  ) %>%
  mutate(
    key = str_sub(param, start = 2, end = 2),
    param = str_sub(param, start = 1, end = 1)
  ) %>%
  pivot_wider(
    names_from = param,
    values_from = value
  )
```

# データ可視化: 複数の系列

```
# A tibble: 4 × 3
  key   x           y
  <chr> <list>      <list>
1 1     <dbl [11]> <dbl [11]>
2 2     <dbl [11]> <dbl [11]>
3 3     <dbl [11]> <dbl [11]>
4 4     <dbl [11]> <dbl [11]>
Warning message:
Values from `value` are not uniquely identified;
output will contain list-cols.
• Use `values_fn = list` to suppress this
  warning.
• Use `values_fn = {summary_fun}` to summarise
  duplicates.
• Use the following dplyr code to identify
  duplicates.
  {data} %>%
  dplyr::group_by(key, param) %>%
  dplyr::summarise(n = dplyr::n(), .groups =
  "drop") %>%
  dplyr::filter(n > 1L)
```

# データ可視化: 複数の系列

```
anscombe_long <-
  anscombe %>%
  rowid_to_column(var = "rowid") %>%
  pivot_longer(
    cols = !rowid,
    names_to = "param",
    values_to = "value",
  ) %>%
  mutate(
    key = str_sub(param, start = 2, end = 2),
    param = str_sub(param, start = 1, end = 1)
  ) %>%
  pivot_wider(
    names_from = param,
    values_from = value
  )
```

# データ可視化: 複数の系列

```
> anscombe_long
# A tibble: 44 × 4
   rowid key       x     y
   <int> <chr> <dbl> <dbl>
 1     1 1        10  8.04
 2     1 2        10  9.14
 3     1 3        10  7.46
 4     1 4         8  6.58
 5     2 1         8  6.95
 6     2 2         8  8.14
 7     2 3         8  6.77
 8     2 4         8  5.76
 9     3 1        13  7.58
10     3 2        13  8.74
# ℹ 34 more rows
# ℹ Use `print(n = ...)` to see more rows
```

# データ可視化: 複数の系列

```
anscombe_long <-
  anscombe %>%
  rowid_to_column(var = "rowid") %>%
  pivot_longer(
    cols = !rowid,
    names_to = "param",
    values_to = "value",
  ) %>%
  mutate(
    key = str_sub(param, start = 2, end = 2),
    param = str_sub(param, start = 1, end = 1)
  ) %>%
  pivot_wider(
    names_from = param,
    values_from = value
  )
```

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...
```

```
anscombe_long <-
  anscombe %>%
  pivot_longer(
    cols = everything(),
    names_to = c(".value", "key"),
    names_pattern = c("(.)(.)")
  )
```

```
# A tibble: 44 × 3
   key       x     y
   <chr> <dbl> <dbl>
 1 1        10  8.04
 2 2        10  9.14
 3 3        10  7.46
 4 4         8  6.58
```

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...
```

```
anscombe_long <-
  anscombe %>%
  pivot_longer(
    cols = everything(),
    names_to = c(".value", "key"),
    names_pattern = c("(.)(.)")
  )
```

2つの要素の最初の要素はそのまま列名として採用するという指定

2つめの要素はkey列に格納

「2文字を1文字ずつ分割」という指定（正規表現）

```
# A tibble: 44 × 3
   key       x     y
   <chr> <dbl> <dbl>
 1 1        10  8.04
 2 2        10  9.14
 3 3        10  7.46
 4 4         8  6.58
```

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...
```

```
anscombe_long <-
  anscombe %>%
  pivot_longer(
    cols = everything(),
    names_to = c(".value", "key"),
    names_pattern = c("([a-z])([0-9])")
  )
```

2つの要素の最初の要素はそのまま列名として採用するという指定

2つめの要素はkey列に格納

「英小文字1文字，数字1文字で分割」という指定（正規表現）

```
# A tibble: 44 × 3
   key       x     y
   <chr> <dbl> <dbl>
 1 1        10  8.04
 2 2        10  9.14
 3 3        10  7.46
 4 4         8  6.58
```

# データ可視化: 複数の系列

```
anscombe00
#>     x_1 x_2 x_3 x_4    y_1   y_2    y_3    y_4
#>  1   10  10  10   8   8.04  9.14   7.46   6.58
#>  2    8   8   8   8   6.95  8.14   6.77   5.76
#>  3   13  13  13   8   7.58  8.74  12.74   7.71
#>  ...
```

もし列名が明示的にセパレートされていれば

```
anscombe_long <-
  anscombe00 %>%
  pivot_longer(
    cols = everything(),
    names_to = c(".value", "key"),
    names_sep = "_"
  )
```

正規表現を使う必要がない

```
# A tibble: 44 × 3
   key       x     y
   <chr> <dbl> <dbl>
 1 1        10  8.04
 2 2        10  9.14
 3 3        10  7.46
 4 4         8  6.58
```

# データ可視化: 複数の系列

```
anscombe
#>     x1 x2 x3 x4    y1   y2    y3    y4
#>  1  10 10 10  8  8.04 9.14  7.46  6.58
#>  2   8  8  8  8  6.95 8.14  6.77  5.76
#>  3  13 13 13  8  7.58 8.74 12.74  7.71
#>  ...
```

```
anscombe_long
#>   key       x     y
#>   <chr> <dbl> <dbl>
#> 1 1        10  8.04
#> 2 2        10  9.14
#> 3 3        10  7.46
#> ...
```

keyは文字型なので離散量扱い
(aes()関数の中で、自動的にfactorで認識される)

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point()
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point()
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key)
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key,
             nrow = 1)
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key,
             nrow = 1) +
  coord_fixed()
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key,
             nrow = 1) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 20)) +
  scale_y_continuous(limits = c(0, 20))
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key,
             nrow = 1) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 20)) +
  scale_y_continuous(limits = c(0, 20)) +
  labs(title = "Anscombe data")
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key,
             nrow = 1) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 20)) +
  scale_y_continuous(limits = c(0, 20)) +
  labs(title = "Anscombe data") +
  theme(legend.title = element_blank())
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key,
             nrow = 1) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 20)) +
  scale_y_continuous(limits = c(0, 20)) +
  labs(title = "Anscombe data") +
  theme(legend.position = "none")
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key,
             nrow = 1) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 20)) +
  scale_y_continuous(limits = c(0, 20)) +
  labs(title = "Anscombe data", tag = "A") +
  theme(legend.position = "none")
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key,
             nrow = 1) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 20)) +
  scale_y_continuous(limits = c(0, 20)) +
  labs(title = "Anscombe data", tag = "A") +
  theme_classic() +
  theme(legend.position = "none")
```

# データ可視化: 複数の系列

```
ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key,
             nrow = 1) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 20)) +
  scale_y_continuous(limits = c(0, 20)) +
  labs(title = "Anscombe data", tag = "A") +
  theme_classic() +
  theme(legend.position = "none",
        strip.background = element_rect(color = NA))
```

# データ可視化: 複数の系列

```
# 以後の作図全てに同じthemeを指定したい場合
theme_set(
 theme_classic() +
  theme(legend.position = "none",
        strip.background = element_rect(color = NA))
)

ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key, nrow = 1) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 20)) +
  scale_y_continuous(limits = c(0, 20)) +
  labs(title = "Anscombe data", tag = "A")
```

# データ可視化: 複数の系列

```
# 以後の作図全てに共通のthemeを指定したい場合
theme_set(
 theme_classic() +
  theme(legend.position = "none",
        strip.background = element_rect(color = NA))
)

ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key, nrow = 1) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 20)) +
  scale_y_continuous(limits = c(0, 20)) +
  scale_color_manual(
    values = c("black", "red", "blue", "green")
  ) +
  labs(title = "Anscombe data", tag = "A")
```

# データ可視化: 複数の系列

```
# 以後の作図全てに共通のthemeを指定したい場合
theme_set(
 theme_classic() +
  theme(legend.position = "none",
        strip.background = element_rect(color = NA))
)

ggplot(data = anscombe_long) +
  aes(x = x, y = y, color = key) +
  geom_point() +
  facet_wrap(facets = . ~ key, nrow = 1) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 20)) +
  scale_y_continuous(limits = c(0, 20)) +
  scale_color_manual(
    values = c("black", "red", "blue", "green")
  ) +
  labs(title = "Anscombe data", tag = "A")
```

プロットの構築

装飾
（思う存分）

# 何にコストを注ぎたいか問題

1. 作戦を建てる

2. 手を動かす

## スクリプト

```
## packages ----
library(tidyverse)

## parameters ----
r <- 2
x_center <- 1
y_center <- 3

## data ----
df <-
  tibble(
    theta = seq(0, 2 * pi, length = 361),
    X = r * cos(theta) + x_center,
    Y = r * sin(theta) + y_center
  )

## visualization ----
ggplot(data = df) +
  aes(x = X, y = Y) +
  geom_path() +
  coord_fixed()
```

# R = スクリプト言語

# ■ Rを始めよう

- 基礎知識
- データの読み書き
- レポーティングの基礎
- データの操作
- **データの可視化** ⬅

# 今日の内容

- データ科学とは
- データ科学のツール
- Rを始めよう ⬅

# 明日の内容

- 確率の話
- 回帰モデルの話
- 分散分析の話