第 1 章　平均値

1　平均値

1.1　準備

1.1.1　必要なライブラリのインストール

下の行は console で実行する。

```
# console
install.packages("tidyverse")
```

1.1.2　必要なライブラリを読み込む

以下のライブラリを読み込む。

```
# code
library(tidyverse)
library(systemfonts)
library(ragg)
```

1.2　平均値とは何かを知る (1.1)

> 「平均値について復留しましょう。小学校で答う算数でおなじみの公式（平均値＝合計 ÷ 個数）です。合計の値をデータの数で割ったものが平均値です。」（テキスト 3）

1.3　平均値 (1.2)

1.3.1　各部門の表

> 「部門は全部で 7 つあります。平均値の公式は合計 ÷ 個数です。ここでの「合計」とは、各部門の総人数のことです。「個数」は部門数、つまり 7 です。」（テキスト 3）

1.3.2　合計値を個数で割る

割り算の記号は、/である。÷ ではないので注意すること。

```r
# code
(123 + 154 + 190 + 30 + 85 + 51 + 60) / 7
```

```
## [1] 99
```

1.4 ひとまとまりのデータ

ひとまとまりのデータを扱う方法は複数ある。

- ベクトル: 同じデータ型の要素を 1 次元に並べたもの【授業で使う】
- リスト: 異なるデータ型の要素を格納できる、より柔軟なデータ構造
- データフレーム
  - data frame: R の基本パッケージ (base R) に含まれる、最も基本的なデータ構造
  - tibble: tidyverse パッケージ群に含まれる、よりモダンなデータフレーム【授業で使う】

この授業では、ベクトルと tibble を主に使用する。

なお、Excel のワークシートの場合

- セル範囲 `A1:A7` のようにセル範囲を選択
- テーブルとして書式設定
- 名前付き範囲（定義された名前）
- スピル

などがある。

1.5 c() ベクトル

ベクトルを作成して平均を出してみる。

- `c()`: ベクトルを作成する

その他の関数

- `sum()`：合計を計算する
- `length()`: データの個数を数える

```r
# code
vec_data <- c(123, 154, 190, 30, 85, 51, 60)
sum(vec_data) / length(vec_data)
```

```
## [1] 99
```

変数の型を確認しておく。

```
# code
class(vec_data)
```

```
## [1] "numeric"
```

numeric (数値型) であることがわかる。

1.6 データフレームを使用

データフレーム tibble（表）を作成して計算してみる。

- tibble() : ベクトルからデータフレーム（表）を作成する
- read_csv(): CSV ファイルを読み込む

というやり方がある。

1.6.1 tibble()

ベクトルからデータフレーム（表）を作成する。

- tibble(): tibble(ティブル) と呼ばれるデータフレームを作成
- c(): 複数の値を結合してベクトルを作成するために使用

```
# code
df_from_vec <- tibble(
  bumon = c(
    "繊維部門", "機械部門", "造船部門",
    "新規事業部門", "環境部門", "デザイン部門",
    "広告部門"
  ),
  ninzuu = c(123, 154, 190, 30, 85, 51, 60)
)
df_from_vec
```

```
## # A tibble: 7 x 2
##   bumon        ninzuu
##   <chr>         <dbl>
## 1 繊維部門        123
## 2 機械部門        154
## 3 造船部門        190
## 4 新規事業部門     30
## 5 環境部門         85
## 6 デザイン部門     51
```

```
## 7 広告部門          60
```

変数の型を確認しておく。

```
# code
class(df_from_vec)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

これは、tibble であることを意味する。

1.6.2 CSV ファイルの作成

`read_csv()` を使う準備をする。

Excel で CSV ファイルを作成する（今後主にこのやり方でデータフレームを作成する）。

- 列名（ヘッダー）は、半角アルファベットと半角数字にする。スペースは使用しないで「_」で置き換える。
- 表の左上（セル A1）から入力する。
- セルを結合しない。
- セル内で改行しない。
- 欠損値は「NA」と入力しておく。
- ファイルは「保存形式」で「CSV UTF-8(コンマ区切り)」を選択して保存する。

CSV ファイルの名前の例: `zenki_chap_01_01_utf8.csv`

CSV ファイルの中身

```
bumon,ninzuu
繊維部門,123
機械部門,154
造船部門,190
（以下省略）
```

1.6.3 ファイルのアップロード

保存したファイルを RStudio で読み込みこむ。

- RStudio の右下のペイン（「Files/Plots/Packages/Help/Viewer pane」）の [upload] ボタンをクリック
- ファイルを選択して読み込み
- 右下のペインにそのファイルがアップロードされたことを確認

1.6.4 read_csv() CSV ファイルの読み込み

CSV ファイルを変数に代入する。

`read_csv()`: CSV ファイルを読み込む。(ウィッカム 2024: 94)

```
# code
df_from_csv <- read_csv("zenki_chap_01_01_utf8.csv")
```

```
## Rows: 7 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): bumon
## dbl (1): ninzuu
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

`df_from_csv` の中身を確認してみる。

```
# code
df_from_csv
```

```
## # A tibble: 7 x 2
##   bumon        ninzuu
##   <chr>         <dbl>
## 1 繊維部門        123
## 2 機械部門        154
## 3 造船部門        190
## 4 新規事業部門     30
## 5 環境部門         85
## 6 デザイン部門     51
## 7 広告部門         60
```

1.7 glimpse() データフレームの確認

すべての変数と各変数の最初のいくつかの観測値を表示するは、`glimpse()` を使用する。(ウィッカム 2024: 5)

```
# code
glimpse(df_from_csv)
```

```
## Rows: 7
```

```
## Columns: 2
## $ bumon  <chr> "繊維部門", "機械部門", "造船部門", "新規事業部門", "環境部門", "デザイン部門", "広
## $ ninzuu <dbl> 123, 154, 190, 30, 85, 51, 60
```

1.7.1 head() tail()

この表はデータ件数が少ないためあまり役に立たないが、`head()` や `tail()` でデータの始めや終わりを表示することができる。

データの最初の数件を表示する

```
# code
head(df_from_csv)
```

```
## # A tibble: 6 x 2
##   bumon        ninzuu
##   <chr>         <dbl>
## 1 繊維部門        123
## 2 機械部門        154
## 3 造船部門        190
## 4 新規事業部門     30
## 5 環境部門         85
## 6 デザイン部門     51
```

データの最後の数件を表示する

```
# code
tail(df_from_csv)
```

```
## # A tibble: 6 x 2
##   bumon        ninzuu
##   <chr>         <dbl>
## 1 機械部門        154
## 2 造船部門        190
## 3 新規事業部門     30
## 4 環境部門         85
## 5 デザイン部門     51
## 6 広告部門         60
```

1.8 列を取り出す

データフレームの特定の列を表示してみる。

1.8.1 $

$ を使うやり方がある。【重要】今後このやり方を多く用いる。

- データフレーム名 $ 列名

```
# code
df_from_csv$ninzuu
```

```
## [1] 123 154 190  30  85  51  60
```

変数の型を確認する。

```
# code
class(df_from_csv$ninzuu)
```

```
## [1] "numeric"
```

今回 tibble から抽出した列は numeric である。

```
# code
df_from_csv$bumon
```

```
## [1] "繊維部門"     "機械部門"     "造船部門"     "新規事業部門" "環境部門"
## [6] "デザイン部門" "広告部門"
```

変数の型を確認する。

```
# code
class(df_from_csv$bumon)
```

```
## [1] "character"
```

今回 tibble から抽出した列は character（文字列）である。

1.8.2 [[“ ”]] pull()

その他の列を取り出すやり方を挙げておく。

- [[]]

```
# code
df_from_csv[["ninzuu"]]
```

```
## [1] 123 154 190  30  85  51  60
```

- pull(): データフレームから特定の列をベクトルとして抽出する

```
# code
pull(df_from_csv, ninzuu)
```

```
## [1] 123 154 190  30  85  51  60
```

1.9 |> パイプ

- |>: これは「パイプ」を意味していいます。R のコードをより読みやすく、そして効率的に書くための強力なツールである。（古い書き方としては「%>%」がある。）

「パイプは左側のものを受け取り、それを右側の関数に渡します」（ウィッカム 2024: 35）

この時、第 1 引数は書かない。

pull(df_from_csv, ninzuu) をパイプを用いて書けば以下のように書ける。

```
# code
df_from_csv |>
  pull(ninzuu)
```

```
## [1] 123 154 190  30  85  51  60
```

1.10 平均を求める

1.10.1 mean()

- mean(): 平均を求める

R で平均を求める関数は mean() です。average() ではないので注意すること。

```
# code
mean(df_from_csv$ninzuu)
```

```
## [1] 99
```

1.10.2 summarize() mean()

- summarize(): データを要約するために使用する

summarize() を使うやり方もあります。今後も出てくる。summarize() の使い方の詳細は、ウィッカム (2024: 49-50) にある。

```
# code
df_from_csv |>
  summarize(
```

```
    mean_ninzuu = mean(ninzuu)
  )
```

```
## # A tibble: 1 x 1
##   mean_ninzuu
##         <dbl>
## 1          99
```

1.11 章末問題

1.11.1 問題

次の気温の平均値（平均気温）を求めてください。

CSV ファイル作成のヒント

- Excel で CSV ファイルは作成できる。
- 1 行目は列の見出しをつける（半角英数スペースなし）
- CSV ファイルは文字コードを UTF-8 にする。
- `read_csv()` を用いて読み込む。

1.11.2 解き方の例

複数の解き方を解説します。

1.11.3 ベクトルを使う場合

```
# code
vec_shoumatu_data <- c(12, 14, 13, 17, 14)
mean(vec_shoumatu_data)
```

```
## [1] 14
```

1.11.4 ベクトルからデータフレームを作成する場合

```
# code
df_shoumatu <- tibble(
  day_name = c("月", "火", "水", "木", "金"),
  temp = c(12, 14, 13, 17, 14)
)
```

glimps()

```
# code
glimpse(df_shoumatu)
```

```
## Rows: 5
## Columns: 2
## $ day_name <chr> "月", "火", "水", "木", "金"
## $ temp     <dbl> 12, 14, 13, 17, 14
```

mean()

```
# code
mean(df_shoumatu$temp)
```

```
## [1] 14
```

summarize()

```
# code
df_shoumatu |>
  summarize(
    mean_temp = mean(temp)
  )
```

```
## # A tibble: 1 x 1
##   mean_temp
##       <dbl>
## 1        14
```

1.11.5 CSV ファイルを作成する場合

ファイル名の例: zenki_chap_01_shoumatu_utf8.csv

ファイルの内容

```
day_name,temp
月,12
火,14
水,13
（以下省略）
```

```
# code
df_shoumatu_2 <- read_csv("zenki_chap_01_shoumatu_utf8.csv")
```

```
## Rows: 5 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): day_name
## dbl (1): temp
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

ファイルの中身を確認

```
# code
glimpse(df_shoumatu_2)
```

```
## Rows: 5
## Columns: 2
## $ day_name <chr> "月", "火", "水", "木", "金"
## $ temp     <dbl> 12, 14, 13, 17, 14
```

平均

```
# code
mean(df_shoumatu_2$temp)
```

```
## [1] 14
```

1.12 課題提出の準備

1.12.1 ファイルの保存

- Rmd ファイルを保存する（[File]-[Save] または [File]-[Save as...]）
- ファイル名は、半角英数字のみ用いる。
- ファイル名にスペースは入れてはならない。
- ファイルの拡張子は「`.Rmd`」である。これを変えてはならない。

1.12.2 コードのチェック

- Console で `install.packages("lintr")`
- Console で `library(lintr)`
- Console で `lint("")` カッコの中には先ほど保存したファイル名を入れる。例 `lint("kadai-01.Rmd")`
- 現在のディレクトリにある R ファイルをチェックする場合は、`lint_dir()`

1.12.3 コードの自動整形

- Console で `install.packages("styler")`
- Console で `library(styler)`
- Console で `style_file("")` カッコの中には先ほど保存したファイル名を入れる。例 `style_file("kadai-01.Rmd")`
- 現在のディレクトリにある R ファイルを自動整形する場合は、`style_dir()`

1.13 テキストと参考文献

1.13.1 テキスト

- 玄場公規, 湊宣明, 豊田裕貴, 2016, 『Excel で学ぶビジネスデータ分析の基礎ビジネス統計スペシャリスト・エクセル分析ベーシック対応』, オデッセイコミュニケーションズ.

1.13.2 参考文献

- ウィッカム, 2024, 『R ではじめるデータサイエンス第 2 版』, オライリー・ジャパン.