# ミクロ政治データ分析実習

## 第9回 データハンドリング（2）

そん じぇひょん
宋 財 泫
関西大学総合情報学部
2021/6/10 (updated: 2021-06-03)

# データのグルーピングと要約

# 目標

- 変数の記述統計量を計算する
  - 記述統計量: 平均値、中央値、標準偏差、分散、ケース数など
- グループごとに記述統計量を計算する

例) 大陸ごとに政治的権利（`FH_PR`）と市民的自由（`FH_CL`）の平均値を計算する

```
## # A tibble: 5 x 3
##   Continent Politial_Right Civil_Liberty
##   <chr>               <dbl>         <dbl>
## 1 Africa               15.8          25.8
## 2 America              29.4          42.6
## 3 Asia                 14.7          24.2
## 4 Europe               31.9          47.6
## 5 Oceania              31.8          47.2
```

# 記述統計量

変数が持つ情報を要約した数値

元の情報:

```
MathScore <- c(82, 45, 69, 94, 88, 73, NA, 51, 90, 63)
```

`MathScore`を代表する値

```
# 平均値
mean(MathScore, na.rm = TRUE)
```

```
## [1] 72.77778
```

```
# 中央値
median(MathScore, na.rm = TRUE)
```

```
## [1] 73
```

`MathScore`のばらつきの具合

```
# 不偏分散
var(MathScore, na.rm = TRUE)
```

```
## [1] 302.4444
```

```
# 不偏標準偏差
sd(MathScore, na.rm = TRUE)
```

```
## [1] 17.39093
```

```
# 四分位範囲
IQR(MathScore, na.rm = TRUE)
```

```
## [1] 25
```

# summarise()の使い方

`summarise()`: 記述統計量を計算する{dplyr}の関数

```
データフレーム %>%
  summarise(記述統計の関数(変数名, ...))
```

記述統計の関数の例

- `mean()`: 平均値
- `median()`: 中央値
- `sd()`: 不偏標準偏差
- `var()`: 不偏分散
- `IQR()`: 四分位範囲
- `min()`、`max()`: 最小値と最大値
- `n()`: ケース数（引数不要）
- など（教科書第14.1.2章）

# 記述統計量の計算（1）

```
library(tidyverse)
df <- read_csv("Data/Micro08.csv") # 第8回の実習用データ
```

例) `df` の `Population` と `Area` の平均値（`mean()`）を計算

```
df %>%
  summarise(mean(Population),
            mean(Area))
```

```
## # A tibble: 1 x 2
##   `mean(Population)` `mean(Area)`
##                <dbl>        <dbl>
## 1          41737773.      696069.
```

# 記述統計量の計算（2）

`summarise()`内に異なる関数を使うことも可能

例) `df`の`Population`と`Area`の平均値（`mean()`）と標準偏差（`sd()`）を計算

```
df %>%
  summarise(mean(Population),
            sd(Population),
            mean(Area),
            sd(Area))
```

```
## # A tibble: 1 x 4
##   `mean(Population)` `sd(Population)` `mean(Area)` `sd(Area)`
##                <dbl>            <dbl>        <dbl>      <dbl>
## 1           41737773.       151270298.       696069.   1872412.
```

# 出力された結果をより見やすく

`summarise()`内に`出力される結果の列名 = 関数()`を指定

例) `df`の`Population`と`Area`の平均値（`mean()`）と標準偏差（`sd()`）を計算し、結果の列名を`Mean_Pop`、`SD_Pop`などとする

```
Pop_Area_df <- df %>%
  summarise(Mean_Pop  = mean(Population),
            SD_Pop    = sd(Population),
            Mean_Area = mean(Area),
            SD_Area   = sd(Area))
Pop_Area_df
```

```
## # A tibble: 1 x 4
##    Mean_Pop     SD_Pop Mean_Area  SD_Area
##       <dbl>      <dbl>     <dbl>    <dbl>
## 1 41737773. 151270298.   696069. 1872412.
```

`summarise()`から得られた結果のデータ構造はデータフレーム/tibble

```
class(Pop_Area_df)
```

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

# グループごとの記述統計量（1）

{dplyr}を使わずに大陸ごとの`PPP_per_capita`の平均値を計算する例

- `PPP_per_capita`は欠損値が含まれているため、`na.rm = TRUE`を指定（指定しないと結果は`NA`となる）

```
mean(df$PPP_per_capita[df$Continent == "Africa"], na.rm = TRUE)
```

```
## [1] 5667.087
```

```
mean(df$PPP_per_capita[df$Continent == "America"], na.rm = TRUE)
```

```
## [1] 18100.29
```

```
mean(df$PPP_per_capita[df$Continent == "Asia"], na.rm = TRUE)
```

```
## [1] 22728.13
```

```
mean(df$PPP_per_capita[df$Continent == "Europe"], na.rm = TRUE)
```

```
## [1] 37782.59
```

```
mean(df$PPP_per_capita[df$Continent == "Oceania"], na.rm = TRUE)
```

```
## [1] 27572.65
```

# グループごとの記述統計量（2）

{dplyr}の`group_by()`を使用

```
データフレーム名 %>%
  group_by(グループ化する変数名) %>%
  summarise(...)
```

例) `df`の`Continent`でデータをグループ化し、`PPP_per_capita`の平均値を計算

```
df %>%
  group_by(Continent) %>%
  summarise(Mean_PPP = mean(PPP_per_capita, na.rm = TRUE))
```

```
## # A tibble: 5 x 2
##   Continent Mean_PPP
##   <chr>         <dbl>
## 1 Africa        5667.
## 2 America      18100.
## 3 Asia         22728.
## 4 Europe       37783.
## 5 Oceania      27573.
```

# 複数の変数でグルーピング

例) `df` の `Continent` と `G20` でデータをグループ化し、`HDI_2018` の平均値を計算

```
df %>%
  group_by(Continent, G20) %>%
  summarise(Mean_HDI = mean(HDI_2018, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'Continent'. You can override using the `
```

```
## # A tibble: 10 x 3
## # Groups:   Continent [5]
##    Continent   G20 Mean_HDI
##    <chr>     <dbl>    <dbl>
##  1 Africa        0    0.550
##  2 Africa        1    0.705
##  3 America       0    0.727
##  4 America       1    0.84
##  5 Asia          0    0.710
##  6 Asia          1    0.798
##  7 Europe        0    0.859
##  8 Europe        1    0.877
##  9 Oceania       0    0.729
## 10 Oceania       1    0.938
```

# グルービング後のsummarise()

謎のメッセージが出力される

```
## `summarise()` has grouped output by 'Continent'. You can override using
the `.groups` argument.
```

とりあえず、`group_by()`の後に`summarise()`を使う場合、`summarise()`の最後に`.groups = "drop"`を追加する。

- 理由は割愛するが、詳細は教科書第14.2章を参照
- 多くの場合、メッセージが出力されるだけで、問題が生じることはあまりない。
- しかし、複数の変数でグルーピングしたり、記述統計量が複数計算される関数（`quantile()`など）を使う場合、問題が生じる可能性あり
  - 特に`summarise()`の後に更にパイプ（`%>%`）を使って計算を続ける場合
- とりあえず、`.groups = "drop"`をしておけば安全

# .groups = "drop"を追加する

謎のメッセージが出力されなくなる

```
df %>%
  group_by(Continent, G20) %>%
  summarise(Mean_HDI = mean(HDI_2018, na.rm = TRUE),
            .groups  = "drop")
```

```
## # A tibble: 10 x 3
##    Continent   G20 Mean_HDI
##    <chr>     <dbl>    <dbl>
##  1 Africa        0    0.550
##  2 Africa        1    0.705
##  3 America       0    0.727
##  4 America       1    0.84
##  5 Asia          0    0.710
##  6 Asia          1    0.798
##  7 Europe        0    0.859
##  8 Europe        1    0.877
##  9 Oceania       0    0.729
## 10 Oceania       1    0.938
```

# グループごとのケース数を計算

`summarise()`の中に`n()`を使用

```
df %>%
  group_by(Continent) %>%
  summarise(Mean_PPP = mean(PPP_per_capita, na.rm = TRUE),
            SD_PPP   = sd(PPP_per_capita, na.rm = TRUE),
            Cases    = n())
```

```
## # A tibble: 5 x 4
##   Continent Mean_PPP SD_PPP Cases
##   <chr>         <dbl>  <dbl> <int>
## 1 Africa        5667.  6015.    54
## 2 America      18100. 12601.    36
## 3 Asia         22728. 24067.    42
## 4 Europe       37783. 21276.    50
## 5 Oceania      27573. 21984.     4
```

# おまけ: 効率的な方法

`across()`関数を利用: 詳細は教科書第14.1章を参照

例) `df`の`Population`から`PPP`列まで平均値を計算し、結果の変数名は`元の変数名_Mean`とする

```
df %>%
  summarise(across(Population:PPP,
                   .fns = list(Mean = ~mean(.x, na.rm = TRUE))))
```

```
## # A tibble: 1 x 4
##   Population_Mean Area_Mean GDP_Mean PPP_Mean
##             <dbl>     <dbl>    <dbl>    <dbl>
## 1       41737773.   696069.  473031.  717953.
```

# 変数の計算

# 変数の計算

`mutate()`: データフレームの変数を用いた計算を行い、新しい列として追加

- 新しい列名として既存の列名を指定すると上書きされる
- 新しく追加された列は最後に位置する（指定可能）

```
データフレーム名 %>%
  mutate(新しい列名 = 計算式)
```

例) `df` の `Population` を `Area` で割り（=人口密度）、`Density` という名の列として追加する

```
# {dplyr}を使わない方法
df$Density <- df$Population / df$Area

# {dplyr}を使う方法
df %>%
  mutate(Density = Population / Area)
```

# {dplyr}の例

例) `df`の`Population`を`Area`で割り（人口密度）、`Density`という名の列として追加する

```
df %>%
  mutate(Density = Population / Area) %>%
  print(n = 5) # 最初の5行のみ出力
```

```
## # A tibble: 186 x 19
##   Country    Population    Area    GDP     PPP GDP_per_capita PPP_per_capita
##   <chr>           <dbl>   <dbl>  <dbl>   <dbl>          <dbl>          <dbl>
## 1 Afghanis…   38928346  652860 1.91e4  82737.           491.          2125.
## 2 Albania      2877797   27400 1.53e4  39658.          5309.         13781.
## 3 Algeria     43851044 2381740 1.70e5 496572.          3876.         11324.
## 4 Andorra        77265     470 3.15e3     NA           40821.            NA
## 5 Angola      32866272 1246700 9.46e4 218533.          2879.          6649.
## # … with 181 more rows, and 12 more variables: G7 <dbl>, G20 <dbl>,
## #   OECD <dbl>, HDI_2018 <dbl>, Polity_Score <dbl>, Polity_Type <chr>,
## #   FH_PR <dbl>, FH_CL <dbl>, FH_Total <dbl>, FH_Status <chr>,
## #   Continent <chr>, Density <dbl>
```

# 新しい列の位置指定

`mutate()`内に`.after`、または`.before`を指定

- `.after = 変数名`: 指定した変数の**後**に新しい変数を入れる
- `.before = 変数名`: 指定した変数の**前**に新しい変数を入れる

**例)** `df`から`Country`～`Area`列を抽出し、`Population`を`Area`で割り（人口密度）、`Density`という名の列として追加する。新しい列の位置は`Area`の後にする。

```
df %>%
  mutate(Density = Population / Area,
         .after  = Area)
```

```
## # A tibble: 186 x 19
##    Country        Population    Area Density     GDP      PPP GDP_per_capita
##    <chr>               <dbl>   <dbl>   <dbl>   <dbl>    <dbl>          <dbl>
##  1 Afghanistan      38928346  652860    59.6   1.91e4  8.27e4           491.
##  2 Albania           2877797   27400   105.    1.53e4  3.97e4          5309.
##  3 Algeria          43851044 2381740    18.4   1.70e5  4.97e5          3876.
##  4 Andorra             77265     470   164.    3.15e3 NA              40821.
##  5 Angola           32866272 1246700    26.4   9.46e4  2.19e5          2879.
##  6 Antigua and B…      97929     440   223.    1.73e3  2.08e3         17643.
##  7 Argentina        45195774 2736690    16.5   4.50e5  1.04e6          9949.
```

# 変数の計算いろいろ（1）

1. `df`の`Population`の合計を`Total_Pop`という列として追加する。
2. `Population`を`Total_Pop`で割り、100を掛ける。結果は`Share_Pop`という名の列として`Population`後に追加する。
3. `Country`から`Share_Pop`までの列のみ残す。
4. `Total_Pop`列を除外する。
5. `Share_Pop`が大きい順で行を並び替える

# 変数の計算いろいろ（1）

```
df %>%
  # Total_Popを作らずにShare_Pop作成時に直接sum(Population)を入れてもOK
  mutate(Total_Pop = sum(Population),
         Share_Pop = Population / Total_Pop * 100,
         .after    = Population) %>%
  select(Country:Share_Pop) %>%
  select(!Total_Pop) %>%
  arrange(desc(Share_Pop))
```

```
## # A tibble: 186 x 3
##    Country       Population Share_Pop
##    <chr>              <dbl>     <dbl>
##  1 China         1447470092     18.6
##  2 India         1380004385     17.8
##  3 United States  334308644      4.31
##  4 Indonesia      273523615      3.52
##  5 Pakistan       220892340      2.85
##  6 Brazil         212559417      2.74
##  7 Nigeria        206139589      2.66
##  8 Bangladesh     164689383      2.12
##  9 Russia         145934462      1.88
```

# 変数の計算いろいろ（2）

1. `df`を利用する
2. `Developed`という列を追加し、`G7`、`G20`、`OECD`のいずれかに加盟した国なら`"先進国"`、それ以外なら`"その他"`とする。
3. 人口密度を`Density`という名の列として追加する。
4. `HDI_2018`と`Polity_Score`のいずれかが欠損した行を除外する。
5. `Developed`変数でデータをグルーピングする。
6. `HDI_2018`、`Polity_Score`、`Density`の平均値を求める。
7. `df2`という名前のオブジェクトとして作業環境内に格納する。

# 変数の計算いろいろ（2）

```
df2 <- df %>%
  mutate(Developed = G7 + G20 + OECD,
         Developed = if_else(Developed > 1, "先進国", "その他"), # 上書き
         Density   = Population / Area) %>%
  filter(!is.na(HDI_2018), !is.na(Polity_Score)) %>%
  group_by(Developed) %>%
  summarise(Density = mean(Density),
            HDI     = mean(HDI_2018),
            Polity  = mean(Polity_Score))

df2
```

```
## # A tibble: 2 x 4
##   Developed Density   HDI Polity
##   <chr>       <dbl> <dbl>  <dbl>
## 1 その他       197. 0.695   3.92
## 2 先進国       174. 0.892   7.91
```

# summarise()の結果を並び替えたい

`df2`を`"先進国" > "その他"`の順番で表示させたい。

- `summarise()`を行う場合、グルーピング変数のアルファベット順で表示される。
- ただし、日本語の場合、50音順にはならない。
  - ひらがな、カタカナなら50音順になるが、感じは ×
- したがって、`summarise()`の前にグルーピング変数を**Factor型に変換**する必要がある。
  - Factor型: 順序付きの文字型

# DevelopedをFactor型に

```
df %>%
  mutate(Developed = G7 + G20 + OECD,
         Developed = if_else(Developed > 1, "先進国", "その他"), # 上書き
         Density   = Population / Area) %>%
  filter(!is.na(HDI_2018), !is.na(Polity_Score)) %>%
  group_by(Developed) %>%
  # ここでなく、filter()前のmutate()内でやるのが効率的
  mutate(Developed = factor(Developed, levels = c("先進国", "その他"))) %>%
  summarise(Density = mean(Density),
            HDI     = mean(HDI_2018),
            Polity  = mean(Polity_Score))
```

```
## # A tibble: 2 x 4
##   Developed Density   HDI Polity
##   <fct>       <dbl> <dbl>  <dbl>
## 1 先進国       174. 0.892   7.91
## 2 その他       197. 0.695   3.92
```

# 変数のリコーディング: 2値の例

`mutate()` 内に `if_else()` を使用（または、 `ifelse()` ）

- `df` の `OECD` が `1` なら `"OECD加盟国"` 、それ以外なら `"OECD非加盟国"` に変換し、 `OECD_J` という列として追加

```
df %>%
  mutate(OECD_J = if_else(OECD == 1, "OECD加盟国", "OECD非加盟国"))
```

例) 変換前

```
df %>%
  group_by(OECD) %>%
  summarise(PPP = mean(PPP_per_capita, na.rm = TRUE),
            HDI = mean(HDI_2018, na.rm = TRUE),
            FH  = mean(FH_Total, na.rm = TRUE))
```

```
## # A tibble: 2 x 4
##    OECD    PPP   HDI    FH
##   <dbl>  <dbl> <dbl> <dbl>
## 1     0 14229. 0.667  49.9
## 2     1 46000. 0.894  89.1
```

# 変数のリコーディング: 2値の例

例) 変換後

```
df %>%
  mutate(OECD_J = if_else(OECD == 1, "OECD加盟国", "OECD非加盟国")) %>%
  group_by(OECD = OECD_J) %>% # 「=」で列名の変更が可能
  summarise(PPP = mean(PPP_per_capita, na.rm = TRUE),
            HDI = mean(HDI_2018, na.rm = TRUE),
            FH  = mean(FH_Total, na.rm = TRUE))
```

```
## # A tibble: 2 x 4
##   OECD             PPP   HDI    FH
##   <chr>          <dbl> <dbl> <dbl>
## 1 OECD加盟国    46000. 0.894  89.1
## 2 OECD非加盟国 14229. 0.667  49.9
```

# 変数のリコーディング: 3値以上の例

`mutate()` 内に `case_when()` を使用

```
データフレーム名 %>%
  mutate(新しい変数名 = case_when(条件1 ~ 新しい値,
                                  条件2 ~ 新しい値,
                                  ...
                                  TRUE  ~ 新しい値))
```

- `TRUE ~ 新しい値` は「上記の条件全てが満たされる場合の値」を意味する

# 変数のリコーディング: 3値以上の例

例) `df` の `Continent` を日本語にし、`Continent_J` として追加

```
df %>%
  mutate(Continent_J = case_when(Continent == "Africa"  ~ "アフリカ",
                                 Continent == "America" ~ "アメリカ",
                                 Continent == "Asia"    ~ "アジア",
                                 Continent == "Europe"  ~ "ヨーロッパ",
                                 TRUE                   ~ "オセアニア")) %>%
  group_by(大陸 = Continent_J) %>%
  # 日本語は非推奨だが、一応使える（_と.を除く特殊記号不可）
  summarise(OECD加盟国比率 = mean(OECD),
            国家数         = n())
```

```
## # A tibble: 5 x 3
##   大陸        OECD加盟国比率 国家数
##   <chr>                <dbl>   <int>
## 1 アジア              0.0714      42
## 2 アフリカ            0           54
## 3 アメリカ            0.139       36
## 4 オセアニア          0.5          4
## 5 ヨーロッパ          0.54        50
```

# 変数のリコーディング: 応用

例) `Continent`が`AP`列を追加し、`"Asia"`か`"Oceania"`、`"America"`なら1、以外は0

方法1: `if_else()`利用

```
df %>%
  mutate(AP = if_else(Continent %in% c("Asia", "America", "Oceania"),
                      1, 0))
```

方法2: `case_when()`利用1

```
df %>%
  mutate(AP = case_when(Continent == "Asia"    ~ 1,
                        Continent == "America" ~ 1,
                        Continent == "Oceania" ~ 1,
                        TRUE                   ~ 0))
```

方法3: `case_when()`利用2

```
df %>%
  mutate(AP = case_when(Continent %in% c("Asia", "America", "Oceania") ~ 1,
                        TRUE                                           ~ 0))
```

# 注意) 欠損値を指定する場合

世論調査などの場合、欠損値が`NA`でなく、`9`や`99`、`""`などの場合がある。

例) `my_data`の例

- `YoungAge`変数を作成し、`Age`が39以下なら`1`、それ以外は`0`にする。ただし、999なら`NA`とする。
- `HighEduc2`変数を作成し、`HighEduc`が1なら`"大卒以上"`、それ以外は`"大卒未満"`にする。ただし、9なら`NA`とする。

```
## # A tibble: 10 x 3
##       ID   Age HighEduc
##    <int> <dbl>    <dbl>
##  1     1    32        1
##  2     2    35        0
##  3     3    57        0
##  4     4   999        1
##  5     5    74        0
##  6     6    66        9
##  7     7   999        1
##  8     8    49        1
##  9     9    78        9
## 10    10    67        9
```

# 注意) 欠損値を指定する場合

**注意:** `条件 ~ NA` ではエラーが発生する。

- `if_else()` も同様。ただし、`ifelse()` は `NA` で作動

```
my_data %>%
  mutate(YoungAge  = case_when(Age == 999 ~ NA,
                               Age <=  39 ~ 1,
                               TRUE       ~ 0),
         HighEduc2 = case_when(HighEduc == 9 ~ NA,
                               HighEduc == 1 ~ "大卒以上",
                               TRUE          ~ "大卒未満"))
```

```
## Error: Problem with `mutate()` column `YoungAge`.
## ℹ `YoungAge = case_when(Age == 999 ~ NA, Age <= 39 ~ 1, TRUE ~ 0)`.
## x must be a logical vector, not a double vector.
```

# 注意) 欠損値を指定する場合

`NA`でなく、**生成される列のデータ型**に応じて`NA_real_`（numeric型）、または`NA_character_`（character型）を使用

```
my_data %>%
  mutate(YoungAge  = case_when(Age == 999 ~ NA_real_,
                               Age <=  39 ~ 1,
                               TRUE       ~ 0),
         HighEduc2 = case_when(HighEduc == 9 ~ NA_character_,
                               HighEduc == 1 ~ "大卒以上",
                               TRUE          ~ "大卒未満"))
```

```
## # A tibble: 10 x 5
##       ID   Age HighEduc YoungAge HighEduc2
##    <int> <dbl>    <dbl>    <dbl> <chr>
##  1     1    32        1        1 大卒以上
##  2     2    35        0        1 大卒未満
##  3     3    57        0        0 大卒未満
##  4     4   999        1       NA 大卒以上
##  5     5    74        0        0 大卒未満
##  6     6    66        9        0 <NA>
##  7     7   999        1       NA 大卒以上
```

# 特定の値を欠損値にコーディング場合

特定の値を欠損値とし、それ以外の値は元も値にする場合

- 主に`if_else()`を使用し、条件に合致した場合は`NA_real_`か`NA_character_`を、合致しない場合は元の変数のままにする。

```
my_data %>%
  mutate(Age       = if_else(Age == 999, NA_real_, Age),
         HighEduc = if_else(HighEduc == 9, NA_real_, HighEduc))
```

```
## # A tibble: 10 x 3
##       ID   Age HighEduc
##    <int> <dbl>    <dbl>
##  1     1    32        1
##  2     2    35        0
##  3     3    57        0
##  4     4    NA        1
##  5     5    74        0
##  6     6    66       NA
##  7     7    NA        1
##  8     8    49        1
##  9     9    78       NA
## 10    10    67       NA
```

# 特定の値を欠損値にコーディング場合

`if_else()`でなく、`case_when()`を使うことも可能

- 欠損を意味する値が複数の場合、`case_when()`を使うか、OR演算子（`|`）を用いた`if_else()`を使用する。

```
my_data %>%
  mutate(Age       = case_when(Age == 999 ~ NA_real_,
                               TRUE       ~ Age),
         HighEduc = case_when(HighEduc == 9 ~ NA_real_,
                               TRUE          ~ HighEduc))
```

```
## # A tibble: 10 x 3
##       ID   Age HighEduc
##    <int> <dbl>    <dbl>
##  1     1    32        1
##  2     2    35        0
##  3     3    57        0
##  4     4    NA        1
##  5     5    74        0
##  6     6    66       NA
##  7     7    NA        1
##  8     8    49        1
```

# 特定の値を欠損値にコーディング場合

{naniar}パッケージの`replace_with_na()`関数を利用

- 引数はリスト型オブジェクトであり、リストの中には`変数名 = 欠損値の値`
- 欠損値の値が複数の場合、`変数名 = c(値1, 値2, ...)`
- 似たような関数として{expss}の`na_if()`関数

```
library(naniar) # 事前に install.package(naniar) でインストール
my_data %>%
  replace_with_na(list(Age = 999, HighEduc = 9))
```

```
## # A tibble: 10 x 3
##       ID   Age HighEduc
##    <int> <dbl>    <dbl>
##  1     1    32        1
##  2     2    35        0
##  3     3    57        0
##  4     4    NA        1
##  5     5    74        0
##  6     6    66       NA
##  7     7    NA        1
##  8     8    49        1
##  9     9    78       NA
```

# まとめ

# 今回の内容

よく分からない箇所は教科書を読み返す or 宋&TAに質問 (できれば、**LMSの質問コーナー**で)

- データのグルーピングと要約: 教科書第14.1章と第14.2章
- 変数の計算: 教科書第14.3章と第14.4章
- Factor型変数の扱い: 教科書第15章
    - Factor型については可視化の時にも解説する。詳しい内容は教科書を参照すること

# 次回の内容

- データの結合: 教科書第14.5章
- 整然データ構造: : 教科書第16章

# 課題

1. 今回講義用のプロジェクトを作成する。
2. LMSからデータ（`.csv`）問題ファイル（`.Rmd`）とサンプルファイル（`.html`）をダウンロードし、プロジェクトのフォルダーに保存する。
   - **ファイル名は変更しない**こと
3. プロジェクトを開き、`.Rmd`ファイルを開く
4. サンプルファイルと同じ結果が得られるようにR Markdown文書を作成する。
5. 随時Knitし、結果を確認する。
   - Knitできないファイルは評価の対象外
6. `.Rmd`ファイルを関大LMSに提出する。
7. **期限は2021年6月12日（土）の23時59分とする。**
   - 時間に余裕を持って取り組むこと。期限直前に取り組み始めてPCトラブルがあっても期限延長はない
8. 答案は次回の講義までに公開する。