医療データ科学実習
Practice of Biomedical Data Science

第3回

前回のSlidoの質問に対する回答

Q1. データフレームの命名方法に規則はありますか？増えると整理するのに苦労しそうです。
A1. Rの文法的に使えない命名規則以外に厳密なルールはありませんが，中身のデータを端的に表すような命名が望ましいです（多少名前が長くなっても）

Q2. 同じ名前で異なるデータフレームを作ると上書きされますか？
A2. その通りです．後に作った方で上書きされてしまうので注意してください

Q3. エクセルではなく、csvで出力するのはどうしてですか？
A3. 第2回の資料「Microsoft Excelで作られたデータの取り扱い」を参照してください

Q4. エディタの内容を保存する方法はありますか？
A4. RStudioのメニューで File -> save as を選ぶと.Rという拡張子でエディタの内容が保存できます

前回のSlidoの質問に対する回答

Q5. Showtextは一度アクティベートすればその後のコード全てに反映されますか？
A5. 反映されます。RやRstudioを再起動すると再度実行が必要です

Q6. ggplot等の外部パッケージはRStudioを起動する度に毎回libraryで有効化する必要があるのでしょうか。自動で有効化する機能はありますでしょうか。
A6. 起動のたびに有効化する必要があります。Rの設定ファイルを編集することで自動化もできるようですが、コードの可読性や関数名の重複の問題(異なるパッケージで同じ名前の関数を使用していることがあり、後から読み込んだものが優先される)を避けるために、毎回明示的にとのパッケージを使用したかをわかるように一行ずつ書くことをおすすめします

Q7. 日付データはどういう形式で保存するのがおすすめですか
A7. RにはDate型が存在します。文字型のオブジェクトに対してas.Dateでデータ型を変換できます。このas.Date関数のデフォルトがyyyy-mm-dd(例: 2025-04-01)もしくはyyyy/mm/dd(例: 2025/04/01)ですのでこのどちらかの形式が推奨です

前回のSlidoの質問に対する回答

Q8. 列名で1月とかならJanなどにするのがいいでしょうか？
A8. はい，そのような命名で良いと思います．注意点として，混乱を避けるために同じ種類の列は命名規則も同じようにすることが推奨されます（1月がJanなら2月はFebに...という感じ）

Q9. macでcsvを開くと、numbersで開きます。numbersが使いにくく感じるのですが、csvを開くのにおすすめはありますか？
A9. 松井は普段csvを直接閲覧する場合はエクセルで開いています．numbersよりは使いやすい気がします（個人の感覚ですが...）

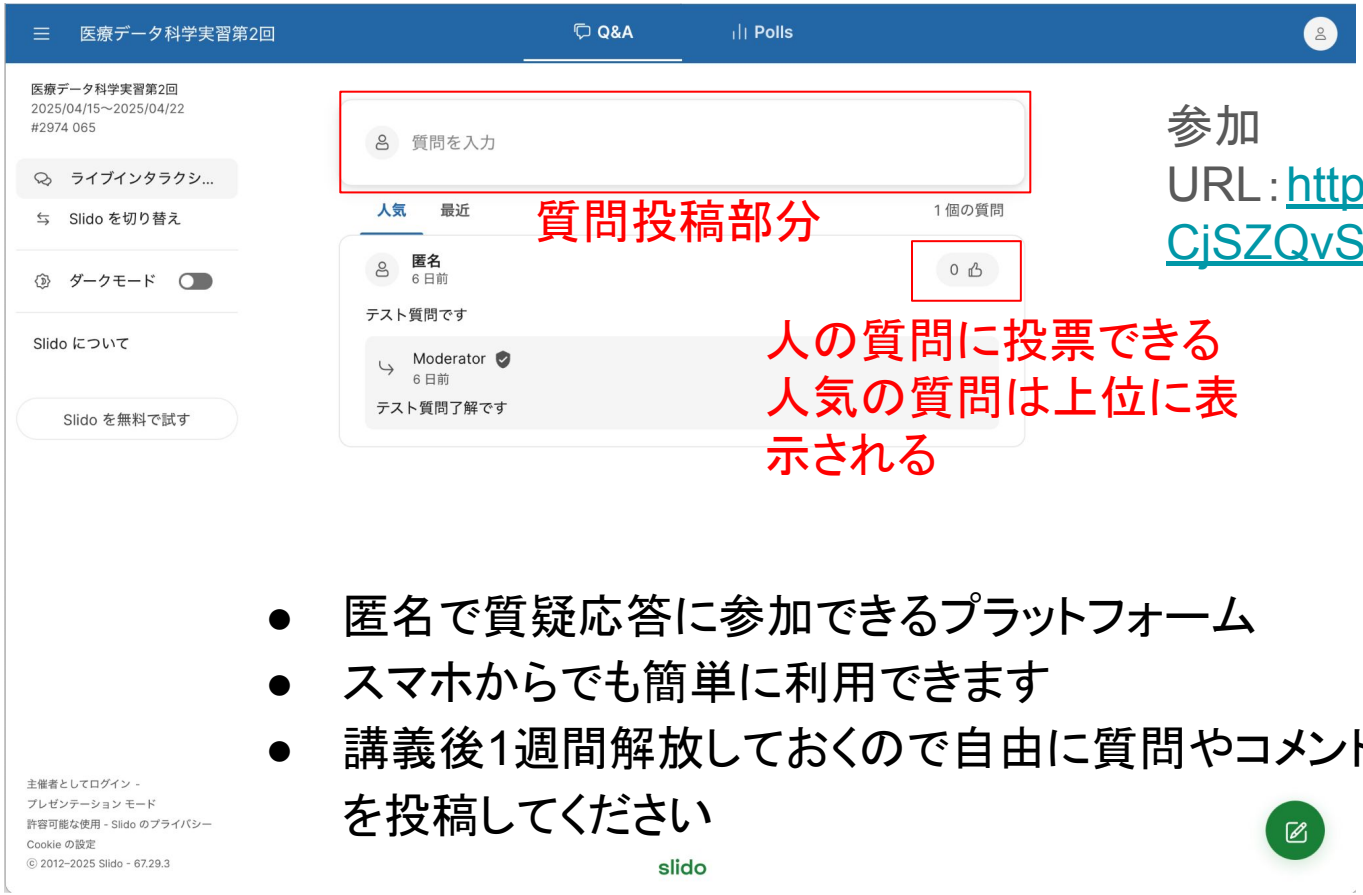
Q10. エクセルで複数シートあるファイルをR上で開くとどのようになりますか
A10. 授業で扱ったread_excel関数では引数にsheet="シート名"と指定するもしくはsheet=2などで何番目のシートかを指定することで特定のシートを読み込めます。指定しない場合は先頭のシートが読み込まれます。

前回のSlidoの質問に対する回答

Q11. summary() 関数を使用した際の、"Class" と "Mode" との違いは何でしょうか？
A11. R上でのオブジェクトのデータ構造を示すのがmodeでR上でのオブジェクトの"役割"のようなものを表す属性を示すのがclassです。それぞれmode()関数、class()関数を各列に使用した時の結果が示されています。この二つの関数は、例えばread.csvで読み込まれたオブジェクトに対してはmode()は"list", class()は"data.frame"を返すといった違いがあり、R上ではlist構造で保存されており、printなどでオブジェクトを見る際にはdata.frameとして表示するという設定になっています。

Q12. 演習問題のように京都、滋賀などのデータの中にまた区分があるようなデータはどのような形で使用するのがいいのでしょうか、縦持ちにしても横持ちにしても行か列が2行になってしまいますし、かといって「京都の〇〇数」とかすると総数を計算するのが難しくなりそうで。
A12. 次回授業の冒頭で演習問題の解説をします(後ろの問題についてはサンプルコードを共有します)。

Slidoで質疑応答に参加しよう



質問投稿部分

人の質問に投票できる
人気の質問は上位に表示される

参加
URL：https://app.sli.do/event/6sQCjSZQvSq31Bvj2Q9aUa

- 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

Rにおける乱数生成と乱数シードに関する補足

- 乱数シードとは？
  - 乱数生成の初期値
  - コンピュータによる乱数は完全なランダムではなく決まったルール（アルゴリズム）に従って生成される「擬似乱数」
  - そのため初期値（乱数シード）が同じならば常に同じ乱数の並びが得られる

- なぜ乱数シードを固定しないと毎回結果が変わるのか？
  - RやPythonなどではデフォルトでは毎回異なる乱数シードが設定される（例えばシステム時刻などを使ってシードを変えている）
  - そのため，乱数生成やサンプリングは何度実行しても毎回異なる結果になる

Chap1. Rの組み込み関数を用いたデータの集計

- 1次元の離散変数の頻度集計，連続変数の頻度集計
- 2次元の離散変数のクロス集計
- 要約統計量算出（連続変数の平均，中央値，分散，標準偏差など）
- 層別の集計，要約統計量の計算

江本先生のスライド
「ベース機能を使っての表作成」へ

Chap2. 外部パッケージを使ってのデータ集計

- `dplyr` と `janitor` を使ったデータ操作・集計，クロス集計や頻度表の作成
- 組み込み関数を用いた集計と外部パッケージを用いた集計の違い

`dplyr` と `janitor` によるデータの集計

`dplyr` データ操作・集計の基本ツール

`janitor` クロス集計や頻度表をきれいに出力してくれるツール

大きな特徴：Base Rの方法に比べて追加の処理なく綺麗に集計を表示できる

演習：RStudioで下記のコードを実行してインストール・インポートしてみよう

- コンソールで以下を実行

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

- インストールが完了したら `library(dplyr)` , `library(janitor)` でインポート

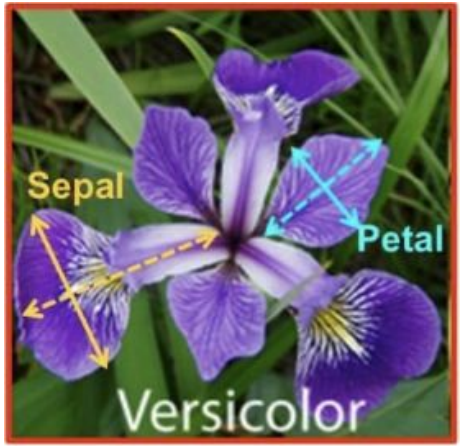
`dplyr` と `janitor` の標準的な使い方

- Rの組み込みデータセット `iris` を使って標準的な使い方を見てみよう
  - `iris` は初めからRに含まれているので `print(iris)` で内容が確認できる

```
print(iris)
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
```

がく片の長さ，幅　　花弁の長さ，幅　　種類



画像の出典（RPubs）

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その1：一次元離散変数の頻度・相対割合の集計

  例　`iris` データセットの `Species` 列の集計

  集計の目的：
  `Species`列に含まれるカテゴリ（setosa, versicolor, virginicaの3種）ごとに

  - 件数（頻度）
  - 全体に占める割合（%表示）

  を見やすく表示する

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その1：一次元離散変数の頻度・相対割合の集計

  例 `iris` データセットの `Species` 列の集計

  - サンプルコード

```r
# 例:iris データセットの Species 列(離散変数)
iris %>%
  tabyl(Species) %>%
  adorn_pct_formatting(digits = 1)  # 割合を%表示
```

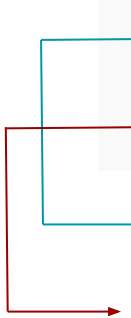
dplyr と janitor の標準的な使い方

- 単純な集計その1：一次元離散変数の頻度・相対割合の集計

  例 iris データセットの Species 列の集計

  - サンプルコード

```
# 例:iris データセットの Species 列(離散変数)
iris %>%
  tabyl(Species) %>%
  adorn_pct_formatting(digits = 1)   # 割合を%表示
```



iris 内の Species 列の頻度表（件数 (n) と割合 (percent)）を作る

集計した percent 列を「小数 -> パーセント表記」に変換

digits = 1 なので 小数点1桁まで表示

dplyr と janitor の標準的な使い方

- 単純な集計その1：一次元離散変数の頻度・相対割合の集計

  例 iris データセットの Species 列の集計

  - サンプルコード

```
# 例:iris データセットの Species 列(離散変数)
iris %>%
  tabyl(Species) %>%
  adorn_pct_formatting(digits = 1)   # 割合を%表示
```

「前の処理の結果を次の関数に渡す」という意味

iris -> tabyl() -> adorn_pct_formatting()
と処理がつながる

iris 内の Species 列の頻度表（件数 (n) と割合 (percent)）を作る

集計した percent 列を「小数 -> パーセント表記」に変換

digits = 1 なので 小数点1桁まで表示

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その1：一次元離散変数の頻度・相対割合の集計

  例 `iris` データセットの `Species` 列の集計

  - 実行結果（**演習**：RStudioでコードを実行して集計を表示してみよう）

```
> # 例：iris データセットの Species 列（離散変数）
> iris %>%
+   tabyl(Species) %>%
+   adorn_pct_formatting(digits = 1)  # 割合を%表示
     Species  n percent
      setosa 50    33.3%
  versicolor 50    33.3%
   virginica 50    33.3%
>
```

各種類の割合は33.3%

各種類ごとに50個のデータ

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その2：一次元連続変数の階級分けと頻度の集計

  例　`iris` データセットの `Sepal.Length` 列を階級に分けて集計

  集計の目的：
  `Sepal.Length`（がく片の長さ）という連続値を一定の幅で区切り（階級分け），各階級にいくつのデータが入っているかを集計して見やすく表示

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その2：一次元連続変数の階級分けと頻度の集計

  例　`iris` データセットの `Sepal.Length` 列を階級に分けて集計

  - サンプルコード

```
# Sepal.Length を階級に分けて集計
iris %>%
  mutate(Sepal_cat = cut(Sepal.Length, breaks = seq(4, 8, 0.5))) %>%
  tabyl(Sepal_cat) %>%
  adorn_pct_formatting()
```

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その2：一次元連続変数の階級分けと頻度の集計

  例 `iris` データセットの `Sepal.Length` 列を階級に分けて集計

  - サンプルコード

```
# Sepal.Length を階級に分けて集計
iris %>%
  mutate(Sepal_cat = cut(Sepal.Length, breaks = seq(4, 8, 0.5))) %>%
  tabyl(Sepal_cat) %>%
  adorn_pct_formatting()
```

`cut()` 関数で連続変数 `Sepal.Length` を階級化して，新しい列 `Sepal_cat` を作成

- `breaks = seq(4, 8, 0.5)` は 4, 4.5, 5.0, …, 8.0 という0.5間隔の区切り
- `cut()` はこの区切りに基づいて `Sepal.Length` を分類（カテゴリ化）

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その2：一次元連続変数の階級分けと頻度の集計

  例 `iris` データセットの `Sepal.Length` 列を階級に分けて集計

  - 実行結果（**演習**：RStudioでコードを実行して集計を表示してみよう）

```
> # Sepal.Length を階級に分けて集計
> iris %>%
+   mutate(Sepal_cat = cut(Sepal.Length, breaks = seq(4, 8, 0.5))) %>%
+   tabyl(Sepal_cat) %>%
+   adorn_pct_formatting()
 Sepal_cat  n percent
   (4,4.5]  5    3.3%
   (4.5,5] 27   18.0%
   (5,5.5] 27   18.0%
   (5.5,6] 30   20.0%
   (6,6.5] 31   20.7%
   (6.5,7] 18   12.0%
   (7,7.5]  6    4.0%
   (7.5,8]  6    4.0%
>
```

各階級ごとのデータの割合

各階級ごとのデータ数

`dplyr` と `janitor` の標準的な使い方

演習：RStudioで `MedDataSets` データセットに対して `dplyr` と `janitor` を使って以下の集計を実行してみよう

1. `MedDataSets` の `gender` 列に含まれるカテゴリ（FemaleとMaleの2種）ごとに件数（頻度）および全体に占める割合（%表示）を集計して表示
2. `MedDataSets` の `age` 列を 10 歳~100歳まで5歳刻みで階級分けし，各階級ごとに件数（頻度）および全体に占める割合（%表示）を集計して表示

解答（サンプルコード）は講義中に示します

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その3:二次元離散変数のクロス集計

  例 `iris` データの `Species` ごとに花のサイズ(仮のカテゴリ変数)を集計
  (ここでは既存の変数をカテゴリ変数に変換)

  集計の目的:
  `iris` データセットを使い、「種別（`Species`）」と、「がく片の長さによるカテゴリ（新たなカテゴリ）」の関係を表形式で分析

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その3：二次元離散変数のクロス集計

  例 `iris` データの `Species` ごとに花のサイズ（仮のカテゴリ変数）を集計
  （ここでは既存の変数をカテゴリ変数に変換）

  - サンプルコード

```
iris %>%
  mutate(SizeCat = ifelse(Sepal.Length > 5.5, "Large", "Small")) %>%
  tabyl(Species, SizeCat) %>%
  adorn_percentages(denominator = "row") %>%
  adorn_pct_formatting(digits = 1) %>%
  adorn_ns()
```

dplyr と janitor の標準的な使い方

- 単純な集計その3：二次元離散変数のクロス集計

例 iris データの Species ごとに花のサイズ（仮のカテゴリ変数）を集計
（ここでは既存の変数をカテゴリ変数に変換）

- サンプルコード

```
iris %>%
  mutate(SizeCat = ifelse(Sepal.Length > 5.5, "Large", "Small")) %>%
  tabyl(Species, SizeCat) %>%
  adorn_percentages(denominator = "row") %>%
  adorn_pct_formatting(digits = 1) %>%
  adorn_ns()
```

連続変数 Sepal.Length をカテゴリ化して新しい変数 SizeCat を作成

- Sepal.Length > 5.5 → "Large"
- それ以外 → "Small"

dplyr と janitor の標準的な使い方

- 単純な集計その3：二次元離散変数のクロス集計

例 iris データの Species ごとに花のサイズ（仮のカテゴリ変数）を集計
（ここでは既存の変数をカテゴリ変数に変換）

- サンプルコード

```
iris %>%
  mutate(SizeCat = ifelse(Sepal.Length > 5.5, "Large", "Small")) %>%
  tabyl(Species, SizeCat) %>%
  adorn_percentages(denominator = "row") %>%
  adorn_pct_formatting(digits = 1) %>%
  adorn_ns()
```

adorn_ns() → パーセントの横にデータ数(n)を括弧付きで表示

Species（setosa, versicolor, virginica）× SizeCat（Large, Small）のクロス集計表を作成

各行（Species）の中で、Large / Small の割合を計算

dplyr と janitor の標準的な使い方

- 単純な集計その3：二次元離散変数のクロス集計

例 iris データの Species ごとに花のサイズ（仮のカテゴリ変数）を集計
（ここでは既存の変数をカテゴリ変数に変換）

- 実行結果（**演習**：RStudioでコードを実行して集計を表示してみよう）

```
> iris %>%
+   mutate(SizeCat = ifelse(Sepal.Length > 5.5, "Large", "Small")) %>%
+   tabyl(Species, SizeCat) %>%
+   adorn_percentages(denominator = "row") %>%
+   adorn_pct_formatting(digits = 1) %>%
+   adorn_ns()
    Species       Large       Small
     setosa  6.0%  (3) 94.0% (47)
 versicolor 78.0% (39) 22.0% (11)
  virginica 98.0% (49)  2.0%  (1)
>
```

各階級ごとのデータの割合

各階級ごとのデータ数

`dplyr` と `janitor` の標準的な使い方

演習：RStudioで `MedDataSets` データセットに対して `dplyr` と `janitor` を使って以下の集計を実行してみよう

1. `MedDataSets` の `amt_weekends` 列を「20本以下」と「20本より多い」の2階級に分け，各階級ごとに件数（頻度）および全体に占める割合（%表示）を欠損を考慮して集計して表示
2. `MedDataSets` の `amt_weekends` 列を「20本以下」と「20本より多い」の2階級に分け，smoke 列とのクロス集計表を作成

解答（サンプルコード）は講義中に示します

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その4：要約統計量の計算

  例 `iris` データの `Sepal.Length` 列の
  - 平均
  - 中央値
  - 分散
  - 標準偏差
  - 最大値，最小値

  を計算する

  集計の目的：
  `iris` データセットを使い、「がく片の長さ（`Sepal.Length`）」という連続変数の要約統計量を計算する

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その4：要約統計量の計算

  例 `iris` データの `Sepal.Length` 列の
  - 平均
  - 中央値
  - 分散
  - 標準偏差
  - 最大値，最小値

  を計算する

サンプルコード

```
iris %>%
  summarise(
    mean = mean(Sepal.Length),
    median = median(Sepal.Length),
    sd = sd(Sepal.Length),
    var = var(Sepal.Length),
    min = min(Sepal.Length),
    max = max(Sepal.Length)
  )
```

dplyr と janitor の標準的な使い方

- 単純な集計その4：要約統計量の計算

例 iris データの Sepal.Length 列の
- 平均
- 中央値
- 分散
- 標準偏差
- 最大値，最小値

を計算する

サンプルコード

```
iris %>%
  summarise(
    mean = mean(Sepal.Length),
    median = median(Sepal.Length),
    sd = sd(Sepal.Length),
    var = var(Sepal.Length),
    min = min(Sepal.Length),
    max = max(Sepal.Length)
  )
```

- データ全体から要約統計量を計算して1行1列の結果を返す関数
- 複数の統計量を同時に計算できる

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その4：要約統計量の計算

例 `iris` データの `Sepal.Length` 列の
  - 平均
  - 中央値
  - 分散
  - 標準偏差
  - 最大値，最小値

を計算する

（注）中身の計算関数は組み込み関数なので，欠損は `na.rm=TRUE` で同様に対処できる

実行結果（**演習**：RStudioでコードを実行して集計を表示してみよう）

```
> iris %>%
+   summarise(
+     mean = mean(Sepal.Length),
+     median = median(Sepal.Length),
+     sd = sd(Sepal.Length),
+     var = var(Sepal.Length),
+     min = min(Sepal.Length),
+     max = max(Sepal.Length)
+   )
      mean median        sd       var min max
1 5.843333    5.8 0.8280661 0.6856935 4.3 7.9
>
```

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その5：層別の要約統計量の計算

  例　`iris` データを `Species` で層別化し，各層ごとに `Sepal.Length` 列の要約統計量（平均・標準偏差・件数）を計算

サンプルコード

```
iris %>%
  group_by(Species) %>%
  summarise(
    mean = mean(Sepal.Length),
    sd = sd(Sepal.Length),
    n = n()
  )
```

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その5：層別の要約統計量の計算

  例　`iris` データを `Species` で層別化し，各層ごとに `Sepal.Length` 列の要約統計量（平均・標準偏差・件数）を計算

サンプルコード

```
iris %>%
  group_by(Species) %>%
  summarise(
    mean = mean(Sepal.Length),
    sd = sd(Sepal.Length),
    n = n()
  )
```

- `iris` データフレームを `Species` 列で層別化
- "setosa", "versicolor", "virginica"の3グループに分かれる

`dplyr` と `janitor` の標準的な使い方

- 単純な集計その5：層別の要約統計量の計算

例　`iris` データを `Species` で層別化し，各層ごとに `Sepal.Length` 列の要約統計量（平均・標準偏差・件数）を計算

実行結果（**演習**：RStudioでコードを実行して集計を表示してみよう）

```
> iris %>%
+   group_by(Species) %>%
+   summarise(
+     mean = mean(Sepal.Length),
+     sd = sd(Sepal.Length),
+     n = n()
+   )
# A tibble: 3 × 4
  Species     mean    sd     n
  <fct>      <dbl> <dbl> <int>
1 setosa      5.01 0.352    50
2 versicolor  5.94 0.516    50
3 virginica   6.59 0.636    50
>
```

base Rによる集計と `dplyr` / `janitor` による集計の比較

観点	base R	dplyr / janitor
記述スタイル	関数中心、やや複雑	パイプ `%>%` による直感的な構文
可読性	やや低い（ネストが多い）	高い（パイプで流れが明快）
拡張性	基本的な処理には十分	グループ集計・ラベル整形に強い
表の整形	`table()`, `prop.table()`	`tabyl()`, `adorn_*()` で柔軟に整形
出力形式	配列や表形式	データフレーム（tibble）