19 データ分析プログラミング (4,5,6)

花澤楓 学籍番号: 2125242

2023/11/27

1　表の作成

分析レポートや論文における表では、記述統計、回帰分析表の 2 種類がよく使われる。

1. 記述統計
   table 1 では、よくコントロール群とトリートメント群の比較がなされ、これを balance table と呼ぶ。
2. 回帰分析
   回帰分析表では、カテゴリー変数 × カテゴリー変数 × アウトカム、の形で表が作られる。例えば、コントロール群（C）とトリートメント群（T）の列と、行にそれぞれの説明変数の点推定値とその標準誤差が報告される。また、diff 列に T と C の差分とその標準誤差を報告することもある（統計的有意な場合に ∗ ∗ ∗ を数値につけるかどうかは各 journal による)。見やすい表を作成するには、以下の 3 つのコツがある。
   - 縦線を入れない
   - 横線を入れすぎない
   - 報告される数値の桁数を調整する

例えば、R には `kableExtra` というパッケージがあり、これを使うと簡単に表を作成することができる。R のデータセット `airquality` を使用して、1973 年 5 月から 9 月までのニューヨークにおけるオゾン料を、各説明変数

- 温度
- 太陽放射の量
- 風力

から回帰分析した結果を表にまとめる。

```
# データの読み込み
# 簡単のため、欠損値を含む行を除外する
df <- as_tibble(airquality) |>
```

表 1　Regression Results

	estimate	std.error	statistic	p.value
(Intercept)	-64.34	23.05	-2.79	0.01
Temp	1.65	0.25	6.52	0.00
Solar.R	0.06	0.02	2.58	0.01
Wind	-3.33	0.65	-5.09	0.00

```
  drop_na()

# 回帰分析
# formula: Ozone = Temp + Solar.R + Wind
model <- lm(Ozone ~ Temp + Solar.R + Wind, data = df) |>
  tidy() |>
  mutate_if(is.numeric, round, 2) |>
  rename(" " = term)

# 表の作成
my_table <- model |>
  kbl(booktabs = T, caption = "Regression Results") |>
  kable_styling(full_width = F, position = "center") |>
  kable_classic_2()

my_table
```

2　衛星・安全性の確保

2.1　生データの問題

生データを扱う際には、主に以下の 3 つの問題がある。

1. 欠測値（missing value）
   欠測値には、
   - 明示的（explicit）
   - 暗示的（implicit）

   なものの 2 種類がある。明示的な欠測値は、NA や NULL などで表される。暗示的な欠測値は、例えば、0 や空白などで表される。欠測値があると、データの分析ができないため、欠測値の処理が必要に

なる。明示的な欠測値に比べて、暗示的な欠測値の方が、欠測値の処理が難しい。そのような場合にはデータを直で確認することが必要になる。

2. 重複値（duplication）
3. 外れ値（outliers）
    - 真の値
    - エントリー・ミス

重複値や、外れ値についてもデータを直接確認する必要がある。なぜなら、それらが真の値なのか、エントリー・ミスなのかを判断する必要があるからである。

2.2 プログラムの一貫性の問題

プログラムを通じて一貫性がない場合、プログラムのメンテナンスが困難になる。そのため、一貫性を持たせつつ、プログラムを統括するようなスクリプトを作成することが重要になる。

2.3 パッケージの整合性の問題

開発者によってパッケージがアップデートされていくが、ある別のパッケージがそれに対応していないためにプログラムが動かせない問題が生じることがある。そのため、パッケージ管理のためのパッケージである `renv` などを使用することが必要になる。また、関数のマスキングのための `box` パッケージもある。

3 エラーへの向き合い方

長いプログラムコードの中で、1 文字でも誤りがあるとエラーが発生する。そのため、プログラミングの 50% の時間が、エラーへの対応に費やされる。例えば、すでに誰かが完成されているコードを自らのコンテクストで応用しようとする際にもさまざまなエラーが発生する。R のバージョン環境の違いや、自分のデータとの整合性などが異なる場合などである。

エラーを確認するための原則として、「エラーの原因が見つかるまで、自分の理解が正しいことを確認する作業」を繰り返すことが重要になる。そのためには、以下の 3 つのステップを踏むことが重要になる。

1. どこにエラーがあるのかについて仮説を立てる
    - エラーメッセージを確認
    - 変数や関数のスペルミスを確認
2. エラーメッセージを確認する
    - エラーメッセージをコピーして、検索エンジンで検索する
    - エラーメッセージをコピーして、Stack Overflow で検索する
    - エラーメッセージをコピーして、GitHub で検索する
    - エラーメッセージをコピーして、RStudio Community で検索する

エラーについてネット上で質問する際には、自分の環境を他の人も再現できるような最小の単位でのプログラム（Minimal Working Example）を示すことで、より多くの人に質問に答えてもらえる可能性が高くなる。

3. 小さなテスト・ケースを書いて、仮説を検証する