Tutorial-05 Solution

Lecturer - Kate Saunders

Table of contents

Solutions

```
library(tidyverse)
library(here)
```

Exercise 1: Swiss exports data

The file `swiss_exports.csv` contains the export data for Switzerland. Each row represents a different date. The first column is the `Date` variable, the second column is the `Year` only and each remaining column measures exports to a different country. The country names are represented using 2 letter code.

1. Read the data into R.

```
swiss_wide <- read_csv(here('data/swiss_exports.csv'))
swiss_wide
```

2. Get the data into long form using the `pivot_longer` function.

Solutions

```
swiss_long <- swiss_wide |>
  pivot_longer(cols = -c(Date, Year), names_to = "Country", values_to = "Exports")
swiss_long
```

3. Using `group_by` and `summarise` create a new data set of yearly aggregate exports to each country. Does having a long form data set help with this?

Solutions

```
annual_exports <- swiss_long |>
  group_by(Year, Country) |>
  summarise(annual_exports = sum(Exports))
annual_exports
```

4. Now produce a scatter plot on a log-log scale of 1988 exports against 2018 exports.

Solutions

```
annual_exports_wide <- annual_exports |>
  filter(Year %in% c(1988, 2018)) |>
  pivot_wider(names_from = Year, values_from = annual_exports)
annual_exports_wide |>
  ggplot(aes(x = `1988`, y = `2018`)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```

5. Produce the same plot but remove all countries for which exports are zero in either 1988 or 2018.

Solutions

```
annual_exports_wide |>
  filter(`1988` != 0, `2018` != 0) |>
  ggplot(aes(x = `1988`, y = `2018`)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```

Exercise 2: Options data

The following example uses Options data from Yahoo Finance. The owner of an put option has the right (but not the obligation) to sell stocks at a predetermined price (the `Strike Price`) on some fixed date (the `Expiry date`). A call option is the same but gives the owner the right to buy stocks.

The objective of this exercise is to produce the well-known *volatility smile* result from finance. This result states that for a given `Expiry date`, a plot of `Implied Volatility` against `Strike Price` is U-shaped. Implied volatility is the volatility of a stock that is computed from stock option data assuming a specific pricing model.

The standard naming in R is snake case (variable_name), where words are separated with underscores. The names in this data set are not saved in snake_case - they have spaces between the words! To use them in R code, you need to put the name of the variable in ticks `` `variable name` ``. You can find this symbol at the top left-hand corner of your keyboard. Working with names having spaces like this is quite difficult and prone to errors. You could try modifying the column names to make them into snake case at the end of the tutorial.

1. Read the data from this csv file into R.

Solutions

```
apple <- read_csv(here("data/apple_options.csv"), show_col_types = FALSE)
apple
```

2. The `Implied Volatility` has been imported as a character variable. To plot this it must be converted to a numeric variable. Create this using the `mutate` function.

Hint: The following code removes the percentage sign, converts to numeric and divides by 100.

```
str_replace('25%', '%', "") %>% as.numeric() / 100
str_replace('1.32%', '%', "") %>% as.numeric() / 100
```

Create the new variable.

Solutions

```
apple <- apple |>
  mutate(`Implied volatility` = str_replace(`Implied volatility`, '%', "") %>% as.numeric()
```

3. The volatility smile is best observed when options with a single expiry date are used. To use as much data as possible, find the expiry date that has the most 'put' options. To do this, you might use the `n()` function, which counts the number of observations in each group.

Solutions

```
apple |>
  filter(Type == 'Put') |>
  group_by(`Expiry date`) |>
  summarise(count = n())
```

The expiry date with the most options is 2020-05-29.

4. Options that are very far *out of the money* (very low strike price for a put option) should be excluded from the analysis. Building on previous answers, construct a data frame that only keeps put options from the expiry date in your answer to question 3, and that have a `Strike Price` above 250.

Note

Note that the `filter` function could use the `&` operator as well.

Solutions

```
q4 <- apple |>
  filter(Type == 'Put', `Expiry date` == "2020-05-29", `Strike Price` > 250)
```

5. Using the data constructed in question 4, find the median value of `Implied Volatility` for each `Strike Price`.

Solutions

```
q5 <- q4 |>
  group_by(`Strike Price`) |>
  summarise(median_iv = median(`Implied volatility`))
```

6. Plot `Implied Volatility` against `Strike Price` using a line plot.

Solutions

```
q5 |>
  ggplot(aes(x = `Strike Price`, y = median_iv)) +
  geom_line()
```

Exercise 3: First Normal Form

Discuss whether the following databases satisfy first normal form.

Database A:

Database B:

Solutions

- Database A has the same issue as seen in lectures. Kamal Usman has two social media accounts so the entry is not atomic.
- Database B resolves this issue. However `Social Media Username` is still not atomic. There are two separate pieces of information, the social media platform and the username. A better database would store these as two separate variables.
- Another point worth mentioning is that the variable `Name` might not be considered to be atomic. It may be better to separate family name from given name

Extra: Clean variable names in options data

Install the `janitor` package and load it into R. Learn how to use the `clean_names` function to create (clean) new column names.

Solutions

```
library(janitor)
apple |>
  clean_names()
```