Lab 3

Table of contents

Introduction

In this lab you'll build the data wrangling and visualization skills you've developed so far and data tidying and joining to your repertoire.

> **Note**
>
> This lab assumes you've completed the labs so far and doesn't repeat setup and overview content from those labs. If you have not yet done those, you should go back and review the previous labs before starting on this one.

Learning objectives

By the end of the lab, you will...

- Be able to pivot/reshape data using `tidyr`
- Continue developing your data wrangling skills using `dplyr`
- Build on your mastery of data visualizations using `ggplot2`
- Get more experience with data science workflow using R, RStudio, Git, and GitHub
- Further your reproducible authoring skills with Quarto
- Improve your familiarity with version control using Git and GitHub

Getting started

Log in to RStudio, clone your `lab-3` repo from GitHub, open your `lab-3.qmd` document, and get started!

Click here if you prefer to see step-by-step instructions

Step 1: Log in to RStudio

- Go to https://cmgr.oit.duke.edu/containers and log in with your Duke NetID and Password.
- Click `STA198-199` under My reservations to log into your container. You should now see the RStudio environment.

Step 2: Clone the repo & start new RStudio project

- Go to the course organization at github.com/sta199-s25 organization on GitHub. Click on the repo with the prefix **lab-3**. It contains the starter documents you need to complete the lab.
- Click on the green **CODE** button, select **Use SSH** (this might already be selected by default, and if it is, you'll see the text **Clone with SSH**). Click on the clipboard icon to copy the repo URL.
- In RStudio, go to *File* *New Project* *Version Control* *Git*.
- Copy and paste the URL of your assignment repo into the dialog box *Repository URL*. Again, please make sure to have *SSH* highlighted under *Clone* when you copy the address.
- Click *Create Project*, and the files from your GitHub repo will be displayed in the *Files* pane in RStudio.

- Click *lab-3.qmd* to open the template Quarto file. This is where you will write up your code and narrative for the lab.

Step 3: Update the YAML

In `lab-3.qmd`, update the `author` field to your name, render your document and examine the changes. Then, in the Git pane, click on **Diff** to view your changes, add a commit message (e.g., "Added author name"), and click **Commit**. Then, push the changes to your GitHub repository, and in your browser confirm that these changes have indeed propagated to your repository.

! Important

If you run into any issues with the first steps outlined above, flag a TA for help before proceeding.

Packages

In this lab we will work with the **tidyverse** package, which is a collection of packages for doing data analysis in a "tidy" way.

```
library(tidyverse)
```

- **Run** the code cell by clicking on the green triangle (play) button for the code cell labeled `load-packages`. This loads the package to make its features (the functions and datasets in it) be accessible from your *Console*.
- Then, **render** the document which loads this package to make its features (the functions and datasets in it) be available for other code cells in your Quarto document.

Guidelines

As we've discussed in lecture, your plots should include an informative title, axes and legends should have human-readable labels, and careful consideration should be given to aesthetic choices.

Additionally, code should follow the tidyverse style. Particularly,

- there should be spaces before and line breaks after each `+` when building a `ggplot`,
- there should also be spaces before and line breaks after each `|>` in a data transformation pipeline,

- code should be properly indented,
- there should be spaces around = signs and spaces after commas.

Furthermore, all code should be visible in the PDF output, i.e., should not run off the page on the PDF. Long lines that run off the page should be split across multiple lines with line breaks.

> **!** Important
>
> Continuing to develop a sound workflow for reproducible data analysis is important as you complete the lab and other assignments in this course. There will be periodic reminders in this assignment to remind you to **render, commit, and push** your changes to GitHub. You should have at least 3 commits with meaningful commit messages by the end of the assignment.

Questions

Part 1

NEED TO UPDATE ALL THIS!

All about `group_by()`!

Use the following data frame for Question 1 and Question 2:

```
df <- tibble(
  var_1 = c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120),
  var_2 = c("Pizza", "Burger", "Pizza", "Pizza", "Burger", "Burger",
            "Burger", "Pizza", "Burger", "Pizza", "Pizza", "Burger"),
  var_3 = c("Apple", "Apple", "Pear", "Pear", "Banana", "Apple",
            "Apple", "Apple", "Pear", "Pear", "Banana", "Apple")
)

df
```

```
# A tibble: 12 x 3
   var_1 var_2  var_3
   <dbl> <chr>  <chr>
 1    10 Pizza  Apple
 2    20 Burger Apple
 3    30 Pizza  Pear
```

```
 4    40 Pizza  Pear
 5    50 Burger Banana
 6    60 Burger Apple
 7    70 Burger Apple
 8    80 Pizza  Apple
 9    90 Burger Pear
10   100 Pizza  Pear
11   110 Pizza  Banana
12   120 Burger Apple
```

Question 1

Grouping by a single variable.

a. What does the following code chunk do? Run it, analyze the result, and articulate in words what `arrange()` does.

```
df |>
  arrange(var_2)
```

```
# A tibble: 12 x 3
   var_1 var_2  var_3
   <dbl> <chr>  <chr>
 1    20 Burger Apple
 2    50 Burger Banana
 3    60 Burger Apple
 4    70 Burger Apple
 5    90 Burger Pear
 6   120 Burger Apple
 7    10 Pizza  Apple
 8    30 Pizza  Pear
 9    40 Pizza  Pear
10    80 Pizza  Apple
11   100 Pizza  Pear
12   110 Pizza  Banana
```

b. What does the following code chunk do? Run it, analyze the result, and articulate in words what `group_by()` does. Also, comment on how it's different from the `arrange()` in part (a).

```
df |>
  group_by(var_2)
```

```
# A tibble: 12 x 3
# Groups:   var_2 [2]
   var_1 var_2  var_3
   <dbl> <chr>  <chr>
 1    10 Pizza  Apple
 2    20 Burger Apple
 3    30 Pizza  Pear
 4    40 Pizza  Pear
 5    50 Burger Banana
 6    60 Burger Apple
 7    70 Burger Apple
 8    80 Pizza  Apple
 9    90 Burger Pear
10   100 Pizza  Pear
11   110 Pizza  Banana
12   120 Burger Apple
```

c. What does the following code chunk do? Run it, analyze the result, and articulate in words what the pipeline does.

```
df |>
  group_by(var_2) |>
  summarize(mean_var_1 = mean(var_1))
```

```
# A tibble: 2 x 2
  var_2  mean_var_1
  <chr>       <dbl>
1 Burger       68.3
2 Pizza        61.7
```

d. Compare this behavior to the following code chunk. Run it, analyze the result, and articulate in words what the pipeline does, and how it's behavior is different from part c.

```
df |>
  summarize(mean_var_1 = mean(var_1))
```

```
# A tibble: 1 x 1
  mean_var_1
       <dbl>
1         65
```

Question 2

Grouping by two variables.

a. What does the following code chunk do? Run it, analyze the result, and articulate in words what the pipeline does. Then, comment on what the message says.

```
df |>
  group_by(var_2, var_3) |>
  summarize(mean_var_1 = mean(var_1))
```

```
`summarise()` has grouped output by 'var_2'. You can override using the
`.groups` argument.
```

```
# A tibble: 6 x 3
# Groups:   var_2 [2]
  var_2  var_3  mean_var_1
  <chr>  <chr>       <dbl>
1 Burger Apple        67.5
2 Burger Banana       50
3 Burger Pear         90
4 Pizza  Apple        45
5 Pizza  Banana      110
6 Pizza  Pear         56.7
```

b. What does the following code chunk do? Run it, analyze the result, and articulate in words what the pipeline does, especially mentioning what the `.groups` argument does. How is the output different from the one in part (a)?

```
df |>
  group_by(var_2, var_3) |>
  summarize(mean_var_1 = mean(var_1), .groups = "drop")
```

```
# A tibble: 6 x 3
  var_2  var_3  mean_var_1
  <chr>  <chr>       <dbl>
1 Burger Apple        67.5
2 Burger Banana       50
3 Burger Pear         90
4 Pizza  Apple        45
5 Pizza  Banana      110
6 Pizza  Pear         56.7
```

c. What do the following pipelines do? Run both, analyze their results, and articulate in words what each pipeline does. How are the outputs of the two pipelines different?

```
df |>
  group_by(var_2, var_3) |>
  summarize(mean_var_1 = mean(var_1), .groups = "drop")
```

```
# A tibble: 6 x 3
  var_2  var_3  mean_var_1
  <chr>  <chr>       <dbl>
1 Burger Apple        67.5
2 Burger Banana       50
3 Burger Pear         90
4 Pizza  Apple        45
5 Pizza  Banana      110
6 Pizza  Pear         56.7
```

```
df |>
  group_by(var_2, var_3) |>
  mutate(mean_var_1 = mean(var_1))
```

```
# A tibble: 12 x 4
# Groups:   var_2, var_3 [6]
   var_1 var_2  var_3  mean_var_1
   <dbl> <chr>  <chr>       <dbl>
 1    10 Pizza  Apple        45
 2    20 Burger Apple        67.5
 3    30 Pizza  Pear         56.7
 4    40 Pizza  Pear         56.7
 5    50 Burger Banana       50
 6    60 Burger Apple        67.5
 7    70 Burger Apple        67.5
 8    80 Pizza  Apple        45
 9    90 Burger Pear         90
10   100 Pizza  Pear         56.7
11   110 Pizza  Banana      110
12   120 Burger Apple        67.5
```

Render, commit, and push your changes to GitHub with the commit message "Added answers for Questions 1 and 2".

Make sure to commit and push all changed files so that your Git pane is empty afterward.