Methods Camp 2025: Day 2

Christina Pao, Sofia Avila, Florencia Torche

September 5, 2025

Table of contents

Intro

Reflections from Week 1!

Quick check-ins: How are folks doing?

Pair/share :)

Outline for Today

1. Working with Data
2. Logistics for Markdown
3. Loops
4. Data Visualization

Working with Data

Social Science Data

What is the typical way that you depict a social science data set?

Draw out a fake data set:

- how would you represent person IDs?
- time?
- demographic characteristics?
- social outcomes of interest?

What if we wanted to change the “shape” of the data? What does that mean, and why might we need to do this?

R Basics: Code-along

Exercise/Live Demo:

1. Open up the `day2_morning_codealong.qmd`
2. Practice annotating your code!

My AI of the day: Claude helped me produce data for your lesson on merging and provided useful exercises for us to do together for pivoting. I used Claude to debug a deprecated function issue!

The `mtcars` dataset

`mtcars` contains info about 32 car models from a 1974 car magazine. We used it a bit yesterday!

Read in the data. **What is the current format of the data?**

```r
library(tidyr)
library(dplyr)

data(mtcars)
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

pivot_longer: Wide to Long

```r
mtcars_long <- mtcars %>%
  tibble::rownames_to_column("car_model") %>%
  pivot_longer(cols = -car_model,
               names_to = "specification",
               values_to = "value")

head(mtcars_long)
```

```
# A tibble: 6 × 3
  car_model specification  value
  <chr>     <chr>          <dbl>
1 Mazda RX4 mpg            21
2 Mazda RX4 cyl             6
3 Mazda RX4 disp          160
4 Mazda RX4 hp            110
5 Mazda RX4 drat            3.9
6 Mazda RX4 wt              2.62
```

pivot_wider: Long to Wide

```r
mtcars_wide <- mtcars_long %>%
  pivot_wider(names_from = specification,
              values_from = value)
head(mtcars_wide)
```

```
# A tibble: 6 × 12
  car_model       mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear
carb
  <chr>         <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl>
1 Mazda RX4      21       6   160   110  3.9   2.62  16.5     0     1     4
4
2 Mazda RX4 W…   21       6   160   110  3.9   2.88  17.0     0     1     4
4
3 Datsun 710     22.8     4   108    93  3.85  2.32  18.6     1     1     4
1
4 Hornet 4 Dr…   21.4     6   258   110  3.08  3.22  19.4     1     0     3
1
5 Hornet Spor…   18.7     8   360   175  3.15  3.44  17.0     0     0     3
```

A (contrived) application of long to wide to long

Task: Calculate the mean value for each specification across all cars, then create a single-row wide-format summary.

1. Use the long-format data.
2. Calculate the mean for each specification.
3. Pivot the data wider.

How many columns should there be? Double check with your resulting dataframe!

```r
mtcars_summary <- mtcars_long %>%
  group_by(specification) %>%
  summarise(mean_value = mean(value, na.rm = TRUE)) %>%
  pivot_wider(names_from = specification,
              values_from = mean_value,
              names_prefix = "avg_")

print(mtcars_summary)
```

```
# A tibble: 1 × 11
  avg_am avg_carb avg_cyl avg_disp avg_drat avg_gear avg_hp avg_mpg avg_qsec
   <dbl>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>  <dbl>   <dbl>    <dbl>
1  0.406     2.81    6.19     231.     3.60     3.69   147.    20.1     17.8
# ℹ 2 more variables: avg_vs <dbl>, avg_wt <dbl>
```

Merging data

Outside of changing the shape of the data, we may need to merge data sets together.

Does anyone have examples of when you may want to merge data?

Types of joins

- **Inner Join:** Returns only rows that have matching values in both datasets
- **Left Join:** Returns all rows from the left dataset, and matched rows from the right
- **Right Join:** Returns all rows from the right dataset, and matched rows from the left
- **Full Join:** Returns all rows when there’s a match in either dataset

Load in some fake data[1]

1. Load in the `mtcars` data again (without modification), ensuring that the first column has a name ("car_model")

```r
mtcars_clean <- mtcars %>%
  tibble::rownames_to_column("car_model")
```

2. Load in the three auxiliary tribbles (i.e., row-wise created tibbles). Run the code chunk in your .qmd file!

`inner_join`

Task: Merge `mtcars_clean` with manufacturer information using an inner join.

```r
mtcars_with_manufacturer <- mtcars_clean %>%
  inner_join(manufacturer_info, by = "car_model")

nrow(mtcars_clean)
```

```
[1] 32
```

```r
nrow(mtcars_with_manufacturer)
```

```
[1] 32
```

`left_join` and missing data

Task: Merge mtcars with fuel efficiency ratings using a left join to keep all cars.

```
            car_model  mpg
1       Mazda RX4 Wag 21.0
2   Hornet Sportabout 18.7
3             Valiant 18.1
4          Duster 360 14.3
5            Merc 280 19.2
6           Merc 280C 17.8
7          Merc 450SE 16.4
8          Merc 450SL 17.3
9         Merc 450SLC 15.2
10 Cadillac Fleetwood 10.4
11 Lincoln Continental 10.4
12  Chrysler Imperial 14.7
13   Dodge Challenger 15.5
```

Multiple joins with different names

Task: Merge all three datasets, handling the different column name in market-values

```r
# Step 1: First merge mtcars with manufacturer info
step1 <- mtcars_clean %>%
  left_join(manufacturer_info, by = "car_model")

# Step 2: Add fuel efficiency data
step2 <- step1 %>%
  left_join(fuel_ratings, by = "car_model")

# Step 3: Add market values (note different column name)
final_dataset <- step2 %>%
  left_join(market_values, by = c("car_model" = "model_name"))
```

Merging in a single chain

```r
complete_dataset <- mtcars_clean %>%
  left_join(manufacturer_info, by = "car_model") %>%
  left_join(fuel_ratings, by = "car_model") %>%
  left_join(market_values, by = c("car_model" = "model_name"))

nrow(complete_dataset)
```

```
[1] 32
```

```r
ncol(complete_dataset)
```

```
[1] 21
```

Key Takeaways for Data Merging[1]

1. **Always check your data dimensions** before and after merging to ensure you haven't lost or duplicated rows unexpectedly.
2. **Handle missing data thoughtfully**: left joins preserve all original data but may introduce NAs that need to be handled in analysis.
3. **Column name mismatches are common**: use by = c("col1" = "col2") syntax when key columns have different names.
4. **Chain multiple joins** efficiently using the pipe operator, but be mindful of the order and type of each join.
5. **Validate your merges** by checking for unexpected duplicates, missing values, or changes in data distribution.

Saving and Exporting Data

Now, with our complete data, we may want to save our data to our working directory!

```r
# Save our merged dataset
library(readr)
write_csv(complete_dataset, "mtcars_complete_analysis.csv")

# We can do the same in excel and save to a full workbook
library(writexl)
datasets_list <- list(
  "Complete_Data" = complete_dataset,
  "Country_Summary" = country_analysis,
  "Performance_Value" = performance_value)

write_xlsx(datasets_list, "mtcars_comprehensive_analysis.xlsx")
```

Other ways to save data…

You may see…

- RDS files
  - R-only analsys and preserves complex data (not readable elsewhere)
- RData files
  - Saves entire workspace, preserves environment (can become very large)

Some tips…

1. Use good file naming practices

```r
current_date <- Sys.Date()
timestamp <- format(Sys.time(), "%Y%m%d_%H%M")
write_csv(complete_dataset, paste0("mtcars_analysis_", current_date, ".csv"
```

2. Handle missing values with care!

```r
write_csv(complete_dataset, "mtcars_with_nas.csv", na = "")
```

3. Ensure proper directory organization!

```r
dir.create("mtcars_analysis_output/csv_files", showWarnings = FALSE)
write_csv(complete_dataset, "mtcars_analysis_output/csv_files/complete_data
```

Logistics for Markdown (and problem sets!)

Why Quarto?

- Quarto, like RMarkdown, allows you to render documents that have text, code, and outputs (figures and tables)!
    - Many people use it to write articles since you can also link with co-authors using GitHub and can input citations (e.g., from Zotero).
- Quarto is flexible to exporting (e.g., PDF/HTML/Word)
- Easy for reproducibility since code and results stay together!

Using Quarto

In our summer assignments, we had you **bold**, *italicize*, use bullet points, and had you do inline/display $math$.

However, there are several other key features:

- Linking text
- Using block quotes
- Adding external images/figures
- Changing page formatting or inserting page breaks, etc.
- Making an external (non-R) table! (See this website for a way to do this easily.)

Task: Discuss each of these with your neighbor. If neither of you know how to perform these tasks, practice asking ChatGPT!

R Chunk Options

Parameter	Description
echo: false	Hide the code, show only output
eval: false	Show code but don’t run it
include: false	Run code but hide everything
warning: false	Suppress warnings
message: false	Suppress messages
tidy: true	Reformats source code

You can also label your code chunks! e.g., {r problem-1a}

Inline Code

Do not copy and paste numeric answers from your code chunks as raw text!

You can use inline code by using the backtick quotes, having the letter “r” and then the calculation you want to add:

- The mean mpg of the `mtcars` data is 20.090625 and the standard deviation is 6.0269481.
- The number of cars in the `mtcars` dataset is 32.

Changing the YAML header

The basic YAML header has the title, author, and format of your document.

You can change this to include the date (in different formats), different default fonts/font sizes, different page orientation, and much more!

Note of warning… the YAML header is extremely finicky!

Debugging Tips[1]

1. **YAML header errors:** Check indentation and syntax
2. **Code chunk issues:** Verify chunk syntax and options (so many people have had eval = F when turning in their problem sets to us in the past!)
3. **Math rendering:** Ensure proper $\LaTeX$ syntax in dollar signs
4. **File paths:** Use relative paths and check file locations

Questions?