Homework 3

YOUR NAME HERE

2026-01-05

Introduction

In this assignment you will practice (1) choosing appropriate plots based on variable type and analytic goal, and (2) using `dplyr` pipelines to move from raw data → transformed data → a final plot.

You will use two data sets:

- `ncbirths` (from the `openintro` package)
- `flights` (from the `nycflights13` package)

Use the following code chunk to load the necessary packages the data listed above.

Part I: Creating plots (ncbirths)

1. Create appropriate visualizations for the `marital` status, and the mothers age (`mage`).

a. `marital` status

b. Mothers age (`mage`)

2. Bivariate Relationships

a. Is birth weight related to length of pregnancy? Create a scatterplot of `weight` versus `weeks`. Add **one** smoother to your plot (either a loess smoother *or* a linear model line).

b. Do babies of smokers tend to have different birth weights than babies of non-smokers? Create a plot that compares the distribution of `weight` across levels of `habit`.

c. Does smoking habit differ by mothers maturity status? Create a table to display the distribution of smoking `habit` by the mothers maturity status (`mature`).

d. Create a side by side bar chart that visually reflects the comparison above.

Part II: Efficient Data Management → Plot (NYC Flights)

Each question starts with the full `flights` data set and will use multiple dplyr verbs so chaining is advised. Your final result should be shown as a printed tibble.

1. Create a new data set that contains only flights that:

- departed from **JFK**
- traveled at least **1000 miles**
- have non-missing `air_time`

Keep only the variables `origin`, `dest`, `distance`, and `air_time`. Then create a new variable called `speed` defined as `distance / air_time * 60`. Finally, sort from fastest to slowest `speed` and display the first 10 rows.

2. What are the most popular destinations for each airport?

> This question is broken down into a few smaller parts to help you ensure that you are on the right track before moving to the next step. Only add one command at a time and "trust but verify" to make sure your code works as intended. You do not need to report the results of each step, just the final result.

a. Use `filter()` to keep only flights with a non-missing distance, then `group_by(origin, dest)` to group the data by origin first, then destination within origin.
b. Use `summarise()` to calculate, for each origin–destination pair:

   - the number of flights as `n_flights`
   - the average distance as `avg_distance`

c. Keep only the top 3 destination using `slice_head(n=3)` after sorting by the number of flights.
d. Use `relocate()` so the columns appear in the order: `origin`, `avg_distance`, `dest`, n_flights‘, and display the resulting table.

3. Exploring travel delays

Typically we don't care if a departure delay is a few minutes late, but anything longer could be detrimental to connecting flights. Use `case_when` to bin the `dep_delay` variable into categories based on departure delay status.

- Create **at least three categories**
- Choose and briefly describe your cutoff values
- Use category names that correspond to intuitive delay types (e.g., "on time", "short," "medium," "long").

- A reasonable approach is to look at the distribution of the departure delay first (histogram), then choose cutoffs based on where the distribution naturally changes.

Display a frequency table showing how many flights fall into each delay category.