COGS 536 Recitation Worksheet

Today's Goals

- Use dplyr to filter, arrange, group, and summarise datasets.
- Handle missing values (count, filter, remove) responsibly.
- Conduct one-sample z-test and one-sample t-test in R.
- Perform sample size calculation for a one-sample t-test using pwr.
- Assess normality using QQ plots and the Shapiro–Wilk test.
- Interpret outputs and write brief conclusions in plain English.

0) Setup

```
# Install (if needed) and load packages
# Create a list of packages we want to use
pkgs <- c("tidyverse", "pwr")

# Check which of these packages are not yet installed on this computer
to_install <- setdiff(pkgs, rownames(installed.packages()))

# If there are any missing packages, install them from the CRAN cloud repository
if (length(to_install)) install.packages(to_install, repos = "https://cloud.r-
project.org")

# Load the packages into the current R session
library(tidyverse)
library(pwr)

# Set a random seed so that random results can be repeated later
set.seed(42)
```

1) Toy data (for practice)

We will use 'mtcars' and 'iris'. To illustrate missing-data operations, we will also create a small copy with some injected `NA`s.

```
# Convert the built-in 'mtcars' dataset into a tibble (a tidy data frame)
# and keep the row names in a new column called "model"
cars <- as_tibble(mtcars, rownames = "model")

# Convert the built-in 'iris' dataset into a tibble (for a cleaner, modern
format)
iris_tbl <- as_tibble(iris)

# Inject NAs into a copy of mtcars for missing-data practice
cars_na <- cars %>%
  mutate(
```

```
    # Replace 'mpg' (miles per gallon) with NA every 7th row
    mpg = if_else(row_number() %% 7 == 0, NA_real_, mpg),

    # Replace 'hp' (horsepower) with NA every 11th row
    hp  = if_else(row_number() %% 11 == 0, NA_real_, hp),

    # Change the 'am' (transmission) column from 0/1 to readable labels
    am  = factor(am, labels = c("Automatic", "Manual")),

    # Convert 'cyl' (number of cylinders) into a factor (categorical variable)
    cyl = factor(cyl))
```

2) Filtering, grouping, transforming, and summarising

2.1 Filter rows

```
# Cars with mpg > 25 and weight (wt) < 2.5
cars %>% filter(mpg > 25, wt < 2.5)
```

- **Exercise 1:** Filter 'cars' to keep only manual transmission ('am == 1') and 4-cylinder cars ('cyl == 4'). Show 'model', 'mpg', 'hp' sorted by descending 'mpg'.

2.2 Select and rename columns

```
cars %>%
  # Keep only the columns we want to look at
  select(model, mpg, hp, wt) %>%

  # Rename columns to have more descriptive names:
  # 'wt' becomes 'weight_1000lb' (weight in thousands of pounds)
  # 'hp' becomes 'horsepower'
  rename(weight_1000lb = wt, horsepower = hp)
```

- **Exercise 2:** From 'iris_tbl', keep 'Species', 'Sepal.Length', 'Petal.Length' and rename them to species, sepal_len, petal_len.

2.3 Create/transform variables with 'mutate()'

```
cars %>%
  # Create a new data frame with selected and calculated columns
  transmute(model, mpg, hp,
    # create a new column that shows horsepower divided by weight
    power_to_weight = hp / wt)
```

- **Exercise 3:** On 'iris_tbl', create 'sepal_ratio = Sepal.Length / Sepal.Width' and keep 'Species', 'sepal_ratio'. Which species has the highest average 'sepal_ratio'?

2.4 Grouping & summarising (descriptives)

```
cars %>%
  # Turn 'cyl' (number of cylinders) into a factor so it's treated as a category
  mutate(cyl = factor(cyl)) %>%

  # Group the data by the number of cylinders
  group_by(cyl) %>%

  # For each cylinder group, calculate summary statistics
  summarise(
    n = n(),                      # number of cars in each group
    mean_mpg = mean(mpg),         # average miles per gallon
    sd_mpg = sd(mpg),             # standard deviation of mpg
    median_mpg = median(mpg),     # median miles per gallon
    .groups = "drop"              # remove the grouping after summarizing
  )
```

- **Exercise 4:** For 'iris_tbl', compute per-species 'n', mean, sd, and median of 'Sepal.Length'. Sort by 'mean' descending.

2.5 Multiple group variables

```
cars %>%
  # Convert 'cyl' (number of cylinders) and 'am' (transmission type)
  # into factors so they are treated as categorical variables
  mutate(cyl = factor(cyl), am = factor(am)) %>%

  # Group the data by both cylinder count and transmission type
  group_by(cyl, am) %>%

  # For each combination of cylinder and transmission, calculate summaries
  summarise(
    n = n(),                  # number of cars in each group
    mean_hp = mean(hp),       # average horsepower
    mean_mpg = mean(mpg),     # average miles per gallon
    .groups = "drop"          # drop grouping after summarizing
 )
```

- **Exercise 5:** For 'iris_tbl', compute mean 'Petal.Width' by 'Species' and 'cut_number(Sepal.Length, 3)' bins.

Reference: Data transformation with dplyr (R for Data Science)

3) Counting and filtering 'NA' values

3.1 Count NA per column

```
cars_na %>%
  # Summarize the dataset by counting missing (NA) values in each column
  summarise(
    across(
      everything(),        # apply the following operation to every column
      ~ sum(is.na(.))      # count how many NAs (missing values) are in each
column))
```

3.2 Count complete vs missing rows

```
cars_na %>%
  summarise(
    # Count the total number of rows in the dataset
    n_rows = n(),

    # Count how many rows have no missing (NA) values
    complete_rows = sum(complete.cases(.)),

    # Calculate how many rows contain at least one missing value
    rows_with_any_na = n_rows - complete_rows
  )
```

3.3 Filter out rows with any NA (drop NAs)

```
# Remove any rows that contain missing (NA) values
cars_no_na <- cars_na %>% drop_na()

# Show how many rows are in the original dataset (with NAs)
nrow(cars_na)

# Show how many rows remain after removing rows with any NAs
nrow(cars_no_na)
```

3.4 Filter rows requiring certain columns to be non-missing

```
# Create a new dataset by removing only the rows
# where 'mpg' or 'hp' have missing (NA) values
cars_req <- cars_na %>% drop_na(mpg, hp)
```

- **Exercise 6:** In 'cars_na', compute the mean 'mpg' after dropping rows where 'mpg' is 'NA'. Compare with the mean computed on 'cars' (no NAs).

Reference: Removing NA with dplyr

4) One-sample tests, sample size, QQ plot, and normality

In this section we will:

(a) run a one-sample z-test (requires known population SD),
(b) run a one-sample t-test,
(c) compute sample size for a one-sample t-test using 'pwr',
(d) draw QQ plots, and
(e) test normality via Shapiro–Wilk.

4.1 One-sample z-test (known population sigma)

Assume (for illustration) that miles-per-gallon in the population has a known standard deviation of 6. We test whether the mean 'mpg' of our sample ('cars$mpg') equals 20.

$$\text{Test statistic: } z = \frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}}$$

```
# We assume that the population standard deviation (sigma) is known (σ = 6)
# and test whether the average mpg (miles per gallon) in our sample equals 20.

# Take the 'mpg' column from the cars dataset
x <- cars$mpg

# Set the hypothesized population mean (H₀: μ = 20)
mu0 <- 20

# Set the known population standard deviation
sigma_pop <- 6  # assumed known for this example

# Find the number of observations in the sample
n <- length(x)

# Calculate the z-test statistic using the formula:
# z = (sample_mean - population_mean) / (sigma / sqrt(n))
z_stat <- (mean(x) - mu0) / (sigma_pop / sqrt(n))

# Calculate the two-sided p-value
# 'pnorm()' gives the cumulative probability under the normal curve
# 'abs(z_stat)' ensures symmetry for two-tailed tests
p_val <- 2 * (1 - pnorm(abs(z_stat)))

# Display both the z-statistic and the p-value
c(z_stat = z_stat, p_value = p_val)
```

- Exercise 7: Repeat the same test for 'hp' with hypothesised mean 'mu0 = 150' and population SD 'sigma_pop = 40'.

Reference: One-sample z-test in R

4.2 One-sample t-test

Now we drop the "known sigma" assumption and test whether mean 'mpg' equals 20 using a t-test.

```
# We no longer assume that the population standard deviation is known.
# Instead, we estimate it from our sample using the t-test.

# Test whether the mean miles per gallon (mpg) equals 20
t.test(cars$mpg,
       mu = 20,                      # hypothesized population mean (H₀: μ = 20)
       alternative = "two.sided")    # test if the mean is different (not just higher
                                       or lower)
```

- **Exercise 8:** Test whether the mean 'Sepal.Length' equals '5.8' in 'iris_tbl' (two-sided). Then repeat with an alternative '"greater"'.

Reference: One-sample t-test

4.3 Sample size calculation (one-sample t-test)

We want to detect a mean difference of 'd = 2' units, with assumed SD 'sd = 6', at power '0.8' and alpha '0.05'. The standardised effect size is

$$d_{Cohen} = \frac{2}{6} = 0.333\overline{3}$$

```
# Goal: Find how many samples we need to detect a mean difference of 2 units
# given a standard deviation (SD) of 6, desired power of 0.8, and alpha of 0.05.

# Step 1: Calculate Cohen's d (effect size)
# Formula: d = mean_difference / SD = 2 / 6 = 0.3333
effect_size <- 2 / 6

# Step 2: Use the power analysis function for a one-sample t-test
pwr.t.test(
  d = effect_size,           # standardized effect size (Cohen's d)
  power = 0.8,               # desired statistical power (80%)
  sig.level = 0.05,          # significance level (alpha = 0.05)
  type = "one.sample",       # type of t-test
  alternative = "two.sided"  # test if mean is different in either direction
)
```

- **Exercise 9:** What sample size do you need to detect a +0.2 difference from 'mu0' if the SD is 1.0 with power 0.9 at alpha 0.05?

Reference: Sample size with 'pwr'

4.4 QQ plots (visual normality check)

```
# Step 1: Set up the plotting area to show two plots side by side
par(mfrow = c(1, 2))

# Step 2: Create a QQ plot for 'mpg' from the cars dataset
# plot sample quantiles vs. theoretical normal quantiles
qqnorm(cars$mpg, main = "QQ plot of mpg")
# add a reference line for a perfect normal distribution
qqline(cars$mpg)

# Step 3: Create a QQ plot for 'Sepal.Length' from the iris dataset
qqnorm(iris_tbl$Sepal.Length, main = "QQ plot of Sepal.Length")
qqline(iris_tbl$Sepal.Length)

# Step 4: Reset the plotting layout back to one plot per window
par(mfrow = c(1, 1))
```

- **Exercise 10:** Create a QQ plot for `cars$hp` and add a reference line.

Reference: QQ plots in base R

4.5 Shapiro–Wilk normality test

```
# This test checks whether the data come from a normal distribution.
# Test for 'mpg' (miles per gallon) from the cars dataset
shapiro.test(cars$mpg)

# Test for 'Sepal.Length' from the iris dataset
shapiro.test(iris_tbl$Sepal.Length)
```

> Note: Shapiro–Wilk is sensitive in large samples; combine with visual checks (QQ plots) and domain reasoning.

- **Exercise 11:** Run 'shapiro.test' for 'cars$hp'. Interpret with alpha = 0.05.

Reference: Shapiro–Wilk test in R

Exercise Solutions

- **Exercise 1:**

```
cars %>%
  filter(am == 1, cyl == 4) %>%
  arrange(desc(mpg)) %>%
  select(model, mpg, hp)
```

- **Exercise 2:**

```
iris_tbl %>%
  select(Species, Sepal.Length, Petal.Length) %>%
  rename(species = Species, sepal_len = Sepal.Length, petal_len = Petal.Length)
```

- **Exercise 3:**

```
iris_tbl %>%
  mutate(sepal_ratio = Sepal.Length / Sepal.Width) %>%
  group_by(Species) %>%
  summarise(mean_ratio = mean(sepal_ratio), .groups = "drop") %>%
  arrange(desc(mean_ratio))
```

- **Exercise 4:**

```
iris_tbl %>%
  group_by(Species) %>%
  summarise(
    n = n(),
    mean_sepal = mean(Sepal.Length),
    sd_sepal = sd(Sepal.Length),
    median_sepal = median(Sepal.Length),
    .groups = "drop") %>%
  arrange(desc(mean_sepal))
```

- **Exercise 5:**

```
iris_tbl %>%
  mutate(sepal_len_bin = cut_number(Sepal.Length, 3)) %>%
  group_by(Species, sepal_len_bin) %>%
  summarise(mean_petal_w = mean(Petal.Width), n = n(), .groups = "drop")
```

- **Exercise 6:**

```
mean(cars_na$mpg, na.rm = TRUE)
cars_na %>% drop_na(mpg) %>% summarise(mean_mpg = mean(mpg))
mean(cars$mpg)
```

- **Exercise 7:**

```
x <- cars$hp; mu0 <- 150; sigma_pop <- 40; n <- length(x)
z_stat <- (mean(x) - mu0) / (sigma_pop / sqrt(n))
p_val <- 2 * (1 - pnorm(abs(z_stat)))
c(z_stat = z_stat, p_value = p_val)
```

- **Exercise 8:**

```
t.test(iris_tbl$Sepal.Length, mu = 5.8, alternative = "two.sided")
t.test(iris_tbl$Sepal.Length, mu = 5.8, alternative = "greater")
```

- **Exercise 9:**

```
pwr.t.test(d = 0.2/1.0, power = 0.9, sig.level = 0.05, type = "one.sample",
alternative = "two.sided")
```

- **Exercise 10:**

```
qqnorm(cars$hp, main = "QQ plot of hp")
qqline(cars$hp)
```

- **Exercise 11:**
  - If 'p-value < 0.05', reject normality (evidence against normality).
  - If 'p-value >= 0.05', do not reject normality (no strong evidence against normality).

Appendix: Tips & Cheats

- 'summarise(across(where(is.numeric), mean, na.rm = TRUE))' to compute means of all numeric columns.
- 'count(var, sort = TRUE)' quickly counts frequencies.
- 'drop_na(col1, col2)' to drop rows with NA in specified columns only.
- Prefer factors for grouping variables: 'mutate(group = factor(group))'.
- z-test requires known population SD; otherwise use t-test.
- Always pair numerical tests with plots and context.