Midterm 1: Practice Test

This exam is 75 minutes long and is made up of 20 short-answer questions, each worth 5 points for 100 points in total. Grab your pencil (no calculators needed!) and get ready to show what you know. Good luck—you're going to do great!

1) Quiz 1A Question 4

How do you find out how many elements are in a vector? What function would you use?

2) Quiz 1B Question 4

If you wanted to find the median GDP per capita for each continent, which two dplyr functions would you need to use together?

3) Quiz 1C Question 1

In ggplot, what does `aes` stand for? What does it do? Give an example for how to use it.

4) Quiz 1D Question 1

Write an anonymous function that multiplies its input by 3.

5) Quiz 1D Question 2

How would you plot the function $y = x^2$ using ggplot2?

6) Quiz 2A Question 6)

For a continuous random variable, what is the probability that it takes on exactly one specific value?

7) Quiz 2B Question 1

In the equation $y_i = \beta_0 + \beta_1 x_i + u_i$, explain what β_0 and β_1 represent. How are they different from $\widehat{\beta}_0$ and $\widehat{\beta}_1$?

8) Quiz 2B Question 2

Explain why the statement $\sum_i x_i y_i = \sum_i x_i \sum_i y_i$ is false. Give a simple numerical example to demonstrate.

9) Quiz 2C Question 1

What does `map()` do in R?

10) Quiz 2C Question 4

Define model selection using forward selection.

11) Koan 1 Question 6)

Create a random character vector that draws "heads" or "tails".

12) Koan 4 Question 1)

Filter gapminder for all the observations from Europe in 2007.

13) Koan 9 Question 2)

Draw a scatterplot comparing gdpPercap and lifeExp, where different continents are drawn with different colors.

14) Assignment 1.5 Question 1.3)

Are male contestants, on average, older than female contestants?

15) Assignment 1.6 Question 5)

Create a function called `summarize_vector` that takes a numeric vector `x` and returns vector containing:

- The length of the vector (call this `x_len`)
- The sum of all values (call this `x_sum`)
- The mean of all values (call this `x_mean`)

Use curly brackets and a return statement.

```
# ___ <- function(x) {
#   ___ <- ___
#   ___ <- ___
#   ___ <- ___
#   return(___)
# }
```

16) Assignment 2.1 Question 6)

Let X be the random variable "the number of likes you get on a social media post" where you get 0-4 likes per post each with equal probability. What is the variance of X?

17) Assignment 2.3 Question 10)

Consider the linear model without an intercept $y = \beta_0 x + u$. Recall that OLS minimizes the sum of squared residuals. Write down what that means in this context mathematically, and then take first order conditions. Show that the OLS estimator is $\hat{\beta}_0 = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$.

18) Assignment 2.4 Question 1)

Suppose the regression output shows $\hat{\beta}_1 = 6$ with a standard error of 2.

a) Calculate the t-statistic for $\hat{\beta}_1$.
b) The sample size of $n = 96$ is large. Is the t-statistic greater than 1.96?
c) Based on your answers to (a) and (b), do we reject $H_0 : \beta_1 = 0$ at the 5% significance level? That is, does X have a statistically significant effect on Y?

19) Assignment 2.7 Question b)

Demonstrating Bias

Finish the code chunk to run 1000 simulations that:

- Generates data using your function `generate_education_data`
- Fits a model of `earnings ~ education` (omitting ability)
- Extracts the education coefficient
- Creates a density plot of the estimates for each of the 1000 simulations

```
# map(
#   1:1000,
#   function(x) {
#     generate_education_data() %>%
#       lm(_____) %>%
#       broom::tidy() %>%
#       slice(_____) %>%
#       select(_____)
#   }
# ) %>%
#   list_rbind() %>%
#   ggplot(aes(x = _____)) +
#   geom_density() +
#   geom_vline(xintercept = 1)
```

20) Assignment 2.8 Question c)

Fill in the missing pieces to implement KNN from scratch for the first observation of the test data set.

```
k <- 2 # We'll find the 2 nearest neighbors. You could also make this value 3, 4, etc.

# Step 1: For each test point x, we need to:
#   a) Calculate distances to all training points: |x - x_train|
#   b) Find the k closest training points
#   c) Take the mean of their y values as our prediction (mean(y_train))

x_test <- nonlinear_test %>%
  slice(1) %>%
  pull(x)

nonlinear_train %>%
```

```
  mutate(distance = abs(x - _____)) %>%
  _____(distance) %>%
  _____(1:k) %>%
  _____(prediction = mean(y)) %>%
  pull(prediction)
```