Practice Exam #1 (150 pts)

SDS 270 — Programming for Data Science in R, Spring 2026

Name: _____

Exam Rules

- You have **75 minutes** to complete this exam.
- No computers, cellphones, or other internet-connected devices. This exam is closed-note and to be completed individually.
- Metadata for the dataset specified in the questions can be found at the back of the exam.
- Give your best effort and attempt every question, laying out your thought-process for questions/parts that may be ambiguous for you. Partial credit may be awarded.

Honor Pledge

Smith College expects all students to be honest and committed to the principles of academic and intellectual integrity in their preparation and submission of coursework and examinations. All submitted work of any kind must be the original work of the student who must cite all the sources used in its preparation.

By signing my name on this exam, I confirm that I have completed this exam with integrity and honesty. I did not use any non-approved materials, electronic devices, or other aids to assist me on the exam.

Signature: _____

This page left intentionally blank.

Q1: Data Preparation (40 pts)

(a) (8 pts) Highlight the main differences between atomic vector and list objects.

(b) (8 pts) Let the `Area` and `Perimeter` be the first two columns in the dataset. Describe two ways we can access the areas and perimeters of all observations with an `Area` between 30,000 and 40,000 (inclusive).

(c) (9 pts) Describe how memory is allocated for `data2` and `data3` when we run the following code at each of the checkpoints (labeled `#CP _`).

```
data2 <- data
# CP 1
data3 <- data2
# CP 2
data2[[2]] <- rep(NA, nrow(data2))
# CP 3
```

(d) (8 pts) Write pseudo-code/code that would print the summary of the `AspectRatio` for each group of the `Class` column in our `data` object.

(e) (7 pts) Describe what would happen in the moment and with the `data` if we ran the following command:

```
data$ShapeTogether <- ifelse(data$ShapeFactor1 < 0.007, 1, "yes")
```

Q2: Sorting (40 pts)

In this question, you will be using the `order()` and `sort()` functions to rearrange and/or sort the observations in the dataset. Here are summaries of the two functions with their arguments:

- `sort(x, decreasing)`: Sorts a vector/factor into ascending or descending order.
  - `x`: vector/factor
  - `decreasing`: defaults as `FALSE`, can also be `TRUE`. Dictates which direction vector/factor is sorted.
- `order(x, decreasing)`: Returns an atomic vector specifying the indices of the elements in ascending or descending order.
  - `x`: vector/factor
  - `decreasing`: defaults as `FALSE`, can also be `TRUE`. Dictates which direction vector/factor is sorted.

As an example, if we have `x <- c(2.5, 3.7, 9.6, 4.3)`, then:

- `sort(x)` would return 2.5, 3.7, 4.3, 9.6
- `order(x)` would return 1, 2, 4, 3

Use this information to answer the following parts.

(a) (8 pts) Using pseudocode or code, describe how we can use one/both of the aforementioned functions to retrieve the bottom 5 observations from the `data` based on its perimeter.

(b) (8 pts) Using pseudocode or code, describe how we can use one/both of the aforementioned functions to retrieve the highest 100 values of the eccentricity vector.

(c) (8 pts) Suppose we want to store the bottom 5 observations based on each non-character variable in a pre-allocated list through the use of a for-loop. Critique the code below and spot areas in which problems may occur.

```
5bottom <- vector(mode = "list", length = ncol(data))
ind = 1
for (i in 1:ncol(data)){
  if (typeof(data[i, ]) != "character"){
    5bottom[[ind]] <- # correct code used from part (a) for ind
  }
}
```

(d) (8 pts) Discuss how we can use a functional to perform the same task as seen in part (c).

(e) (8 pts) Address the memory implications of the process seen in parts (c) and (d), including potential alternatives or fixes.

Q3: Evaluating Changes in Script (35 pts)

(a) (24 pts) Suppose we create code that defines new objects in R and creates a new function to run through a calculation:

```
within2SDs <- function(x){
  meanX <- mean(x)
  sdX <- sd(x)
  within2 <- mean(x > meanX - 2 * sdX & x < meanX + 2 * sdX)
  return(within2)
}

x <- c(1, 8, 5, 2, 9)
```

For each sub-part (6 pts each), describe the code's behavior when running the following chunk in R:

(i)
```
mean <- function(x){(min(x) + max(x)) / 2}
within2SDs(x)
```

(ii)
```
sdX <- 10
within2SDs(x)
```

(iii)
```
y <- c(2, 2, 2, 2, 2)
within2SDs(y)
```

(iv)
```
within2SDs()
```

(b) (11 pts) Rewrite the `within2SDs` function such that we can access additional arguments in the `mean()` function.

Q4: Building A Whole Script (35 pts)

Sketch out a script (of code/pseudocode and comments) that accomplishes the following:

I want to create an object in R that houses the following information for each group of `Class`:

- Percentage of areas that are below 30,000 or above 60,000
- A summary of the compactness variable
- A scatter plot of area vs. perimeter

In this sketch, specify what data/coding structures you will use and how they will accomplish the task. Consider appropriate ways to accomplish these tasks.

Q4: Building A Whole Script (35 pts) (cont.)

Background

The dataset we will focus our analysis on is a dataset regarding seven different dry beans and their characteristics over time. You can see a real synposis here:

> Seven different types of dry beans were used in this research, taking into account the features such as form, shape, type, and structure by the market situation. A computer vision system was developed to distinguish seven different registered varieties of dry beans with similar features in order to obtain uniform seed classification. For the classification model, images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. Bean images obtained by computer vision system were subjected to segmentation and feature extraction stages, and a total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains. Link

- `Area`: The area of a bean zone and the number of pixels within its boundaries (double)
- `Perimeter`: Bean circumference is defined as the length of its border (double)
- `MajorAxisLength`: The distance between the ends of the longest line that can be drawn from a bean (double)
- `MinorAxisLength`: The longest line that can be drawn from the bean while standing perpendicular to the main axis (double)
- `AspectRatio`: Defines the relationship between MajorAxisLength and MinorAxisLength (double)
- `Eccentricity`: Eccentricity of the ellipse having the same moments as the region (double)
- `ConvexArea`: Number of pixels in the smallest convex polygon that can contain the area of a bean seed (integer)
- `EquivDiameter`: Equivalent diameter: The diameter of a circle having the same area as a bean seed area (double)
- `Extent`: The ratio of the pixels in the bounding box to the bean area (double)
- `Solidity`: Also known as convexity. The ratio of the pixels in the convex shell to those found in beans (double)
- `Roundness`: Calculated with the following formula: $\frac{4\pi A}{P^2}$ (double)
- `Compactness`: Measures the roundness of an object (units of Ed/L, double)

- `ShapeFactor1`: (double)
- `ShapeFactor2`: (double)
- `ShapeFactor3`: (double)
- `ShapeFactor4`: (double)
- `Class`: Seker, Barbunya, Bombay, Cali, Dermosan, Horoz, and Sira (character)