Lab 4: Data Manipulation and Visualization

INSERT YOUR NAME HERE (INSERT YOUR UW NETID HERE)

Due by 23:59pm on Feb 13, 2024

Total Points: 70 pts

Part 1. Pipes to Base R (2+2+2+2 pts)

Loading the `tidyverse` package:

```
library(tidyverse)
```

For each of the following code blocks, which are written with pipes, write equivalent code in base R (to do the same thing).

1. Base R code:

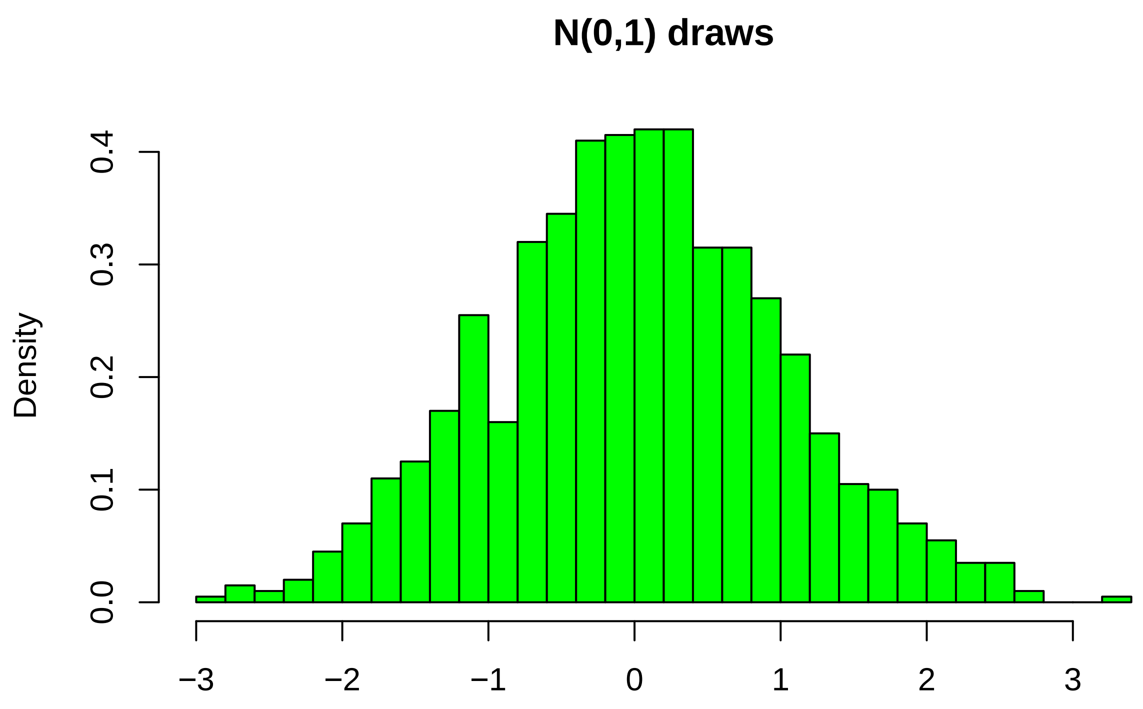
```
letters %>%
  toupper %>%
  paste(collapse = "+")
```

```
## [1] "A+B+C+D+E+F+G+H+I+J+K+L+M+N+O+P+Q+R+S+T+U+V+W+X+Y+Z"
```

```
# Your code here
```

2. Base R code:

```
set.seed(123)
rnorm(1000) %>%
  hist(breaks = 30, main = "N(0,1) draws", col = "green", prob = TRUE)
```



.

```
# Your code here
```

3. Base R code:

```
set.seed(123)
rnorm(1000) %>%
  hist(breaks=30, plot=FALSE) %>%
  `[[`("density") %>%
  max
```

```
## [1] 0.42
```

```
# Your code here
```

4. Base R code:

```
"  Ceci n'est pas une pipe  " %>%
gsub("une", "un", .) %>%
trimws
```

```
## [1] "Ceci n'est pas un pipe"
```

```
# Your code here
```

Part 2. Base R to Pipes (2+2+2+2 pts)

For each of the following code blocks, which are written in base R, write equivalent code with pipes (to do the same thing).

1. Hint: Use the dot ., as seen above in 1-4, or in the lecture slides.

```
paste("Your grade is", sample(c("A","B","C","D","R"), size = 1))
```

```
## [1] "Your grade is A"
```

```
# Your code here
```

2. Hint: use the dot `.` again, in order to index `state.name` directly in the last pipe command. In addition, you can view `state.x77[,"Illiteracy"]` as the entire object when writing your pipes code.

```
state.name[which.max(state.x77[,"Illiteracy"])]
```

```
## [1] "Louisiana"
```

```
# Your code here
```

3. Note: `str.url` is defined for this question and the next one; you can simply refer to it in your solution code (it is not part of the code you have to convert to pipes). In addition, you can ignore the difference between the headers from the output of your pipes code and the original base R code.

```
str.url = "https://github.com/zhangyk8/zhangyk8.github.io/raw/master/_teaching/file_stat302/Data/king.t

lines = readLines(str.url)
text = paste(lines, collapse=" ")
words = strsplit(text, split="[[:space:]]|[[:punct:]]")[[1]]
wordtab = table(words)
wordtab = sort(wordtab, decreasing=TRUE)
head(wordtab, 10)
```

```
## words
##        of  the   to  and    a   be will that   is
##   203   98   98   58   40   37   32   25   24   23
```

```
# Your code here
```

4. Hint: the only difference between this and the last part is the line `words = words[words != ""]`. This is a bit tricky line to do with pipes: use the dot `.`, once more, and manipulate it as if were a variable name.

```
lines = readLines(str.url)
text = paste(lines, collapse=" ")
words = strsplit(text, split="[[:space:]]|[[:punct:]]")[[1]]
words = words[words != ""]
wordtab = table(words)
wordtab = sort(wordtab, decreasing=TRUE)
head(wordtab, 10)
```

```
## words
##   of  the   to  and    a   be will that   is   we
##   98   98   58   40   37   32   25   24   23   21
```

```
# Your code here
```

Part 3. Practice with `dplyr` Verbs (2+2+2+2+2+3+3 pts)

We will look at a data set on 97 men who have prostate cancer (from the book The Elements of Statistical Learning). There are 9 variables measured on these 97 men:

1. `lpsa`: log PSA score.
2. `lcavol`: log cancer volume.
3. `lweight`: log prostate weight.
4. `age`: age of patient.
5. `lbph`: log of the amount of benign prostatic hyperplasia.
6. `svi`: seminal vesicle invasion (0 means no SVI).

7. `lcp`: log of capsular penetration.
8. `gleason`: Gleason score.
9. `pgg45`: percent of Gleason scores 4 or 5.

Read the prostate cancer data into R as a data frame.

```r
pros_df = read.table("https://github.com/zhangyk8/zhangyk8.github.io/raw/master/_teaching/file_stat302/
```

In the following, you are required to use pipes and `dplyr` verbs to answer questions on `pros_df`.

1. Among the men whose `lcp` value is equal to the minimum value (across the entire data set), report the range (min and max) of `lpsa`.

```r
# Your code goes here
```

2. Order the rows by decreasing `age`, then display the rows from men who are strictly older than 70 and without SVI.

```r
# Your code goes here
```

3. Order the rows by decreasing `age`, then decreasing `lpsa` score, and display the rows from men who are strictly older than 70 and without SVI, but only the `age`, `lpsa`, `lcavol`, and `lweight` columns. Hint: `arrange()` can take two arguments, and the order you pass in them specifies the priority.

```r
# Your code goes here
```

Below, we read in a dataset of the 1000 fastest times ever recorded for the 100m sprint in women's track.

```r
sprint_w = read.table(
  file="https://github.com/zhangyk8/zhangyk8.github.io/raw/master/_teaching/file_stat302/Data/sprint.w.
  sep="\t", quote="", header=TRUE)
```

In the following, use pipes and `dplyr` verbs to answer questions on `sprint_w`.

4. Order the rows by increasing `Wind` value, and then display only the women who ran at most 10.7 seconds.

```r
# Your code goes here
```

5. Order the rows by terms of increasing `Time`, then decreasing `Wind`, and again display only the women who ran at most 10.7 seconds, but only the `Time`, `Wind`, `Name`, and `Date` columns.

```r
# Your code goes here
```

6. Select a data frame from `sprint_w` with only `Time` and `Wind` columns using data where `Wind` values that are nonpositive. Then, coerce all the entries of the selected data frame to numeric values and subset those rows without `NA`. Print the number of rows in the final data frame and plot `Time` versus `Wind` columns. Hint: note that for a data frame, `df` with columns `colX` and `colY`, you can use `plot(colY ~ colX, data = df)`, to plot `df$colY` (y-axis) versus `df$colX` (x-axis). Hint: `drop_na()` in the `tidyr` package allows you to drop rows based on `NA` values.

```r
# Your code goes here
```

7. Based on the selected data frame in the last subquestion, further subset a data frame with the single fastest `Time` per `Wind` value. Add this scatterplot to the plot in the last subquestion using different color and point type. In addition, set the point size of the plot in the last subquestion as `cex=2` and the point size of the added scatterplot as `cex=1` (That is, your new data frame and added plot should be as in the last part, but among points that share the same x value, only the point with the lowest y value should be drawn.)

```r
# Your code goes here
```

Part 4: Practice with `tidyr` Verbs (2+3+3+4+2+3+3 pts)

Below, we read in a dataset of the 1000 fastest times ever recorded for the 100m sprint in men's track.

```r
sprint_m = read.table(
  file="https://github.com/zhangyk8/zhangyk8.github.io/raw/master/_teaching/file_stat302/Data/sprint.m.
  sep="\t", quote="", header=TRUE)
```

In the following, use pipes and `dplyr` and `tidyr` verbs to answer questions on `sprint_m`. In some parts, it might make more sense to use direct indexing, and that's perfectly fine.

1. Compute, for each country, the fastest time among men athletes who come from that country. Display the first 7 rows of the result.

```r
# Your code goes here
```

2. Confirm that the `Time` column is stored as character data type (store it as a variable `Time_org` and print out the data type). Then, convert the `Time` column to numeric. After converting to numeric, there will be `NA` values; look at the positions with `NA` values and revisit the original `Time` column. Explain in words why it was stored as character type in the first place.

```r
# Your code goes here
```

3. Define a reduced data frame `dat_reduced` as follows. For each athlete and each city, keep the fastest of all times that they recorded in this city. Then keep all rows without `NA` values in the `Time` column and drop duplicated rows. Your new data frame `dat_reduced` should have 600 rows and 3 columns (`Name`, `City`, `Time`). Confirm that it has these dimensions, and display its first 7 rows.

```r
# Your code goes here
```

4. The data frame `dat_reduced` is said to be in "long" format: it has observations on the rows, and variables (`Name`, `City`, `Time`) on the columns. Arrange the rows alphabetically by city; convert this data frame into "wide" format; and then order the rows so that they are alphabetical by athlete name. Call the result `dat_wide`. To be clear, here the first column should be the athlete names, and the remaining columns should correspond to the cities. Confirm that your data frame has dimension 141×152. Find the lengths of unique athletes and unique cities in `dat_reduced` and argue why these dimensions make sense.

```r
# Your code goes here
```

5. Not counting the names in the first column, how many non-`NA` values does `dat_wide` have? How could you have guessed this number ahead of time, directly from `dat_reduced` (before any pivoting at all)?

```r
# Your code goes here
```

6. From `dat_wide`, look at the row for the world record keeper "Usain Bolt", and determine the city names that do not have `NA` values. These should be the cities in which he raced. Determine these cities directly from `dat_reduced`, and confirm that they match using logical operations.

```r
# Your code goes here
```

7. Convert `dat_wide` back into "long" format, and call the result `dat_long`. Remove rows that have `NA` values (hint: you can do this by setting `values_drop_na = TRUE` in the call to the pivoting function), and order the rows alphabetically by athlete and city name. Once you've done this, `dat_long` should have matching entries to the ordered `dat_reduced`; confirm that this is the case by outputting a logical `TRUE`/`FALSE`.

```r
# Your code goes here
```

Part 5: Practice with `ggplot2` (2+2+2+3+3+6 pts)

We are going to create some plots using `ggplot2` based on the data frame `babies`. Let's load this data frame into R, whose column descriptions can be found in Table 3.1 on Page 112 of this document.

```r
load(url("https://github.com/zhangyk8/zhangyk8.github.io/raw/master/_teaching/file_stat302/Data/babies.
library(ggplot2)
```

There are at least 5 "notions" to a ggplot:

- The plot object (created by `ggplot()`). That can be used to identify the data frame and set up the aesthetic mapping of variables to aspects of the plot.
- The aesthetic mapping which connects variables in a data frame to plotting features such as the x axis, y axis, color, line type, etc.
- Layers - These contain visual components that are added to the plot. A layer consists of a geom (visual representation) and a statistic (operation on variable). Each geom has a default statistic and we typically use the default. We can specify a layer with `layer()`, but more typically we use the shortcut functions `geom_VisualPiece()` and the default values that are set up with this function.
- Scales - These allow us to control how an aesthetic is represented in the plot. Through the `scale_XXX_XXX()` functions we can, e.g., specify axis labels (name), axis limits, transformations, colors, line types, etc.
- Themes - The theme allows us to control non-data aspects of the graphic such as font size, location of legend, background colors, etc.

We will demonstrate these various pieces as we make 4 plots of the babies data. These include a histogram, boxplot, bar plot, and scatter plot.

1. We begin by making the histogram on the variable `parity`. Set the `binwidth` to be 0.8 for the plotted histogram.

```r
# Your code goes here
```

2. Create a boxplot (with `geom_boxplot()`) of the birth weight where we have a separate box for each level of smoking status. (Hint: What is the x-axis aesthetic? What is the y-axis aesthetic?)

```r
# Your code goes here
```

One problem here is that there are some `NA`s in somking status. We want to eliminate them. We can drop any rows in babies that have `NA` for smoking status when we specify the data frame to use. In addition, set the x-axis label as "Smoking Status" and y-axis label as "Birth Weight" in the new boxplot.

```r
# Your code goes here
```

3. We want to know the numbers of mothers in each smoking status. Make a bar plot for smoking status and drop the `NA`s. In addition, set the x-axis label as "Smoking Status" and y-axis label as "Number of mothers" in the bar plot.

```r
# Your code goes here
```

4. We want to make a scatter plot that shows the relationship between the baby's birth weight and the mothers height. In particular, we will only look at Never and Current smokers.

Begin by making a scatter plot of these points and using color to identify the group that each baby belongs to. Make sure that all the `NA`s in the baby's birth weight and the mothers height of the `babies` data frame are dropped. Consider also jitterring the points, shrinking the points, and using transparency to reduce the problems with overplotting.

```r
# Your code goes here
```

5. Now we can add another layer to the above plot in 4 that includes two linear regression lines (Hint: `method = "lm"` in the corresponding `geom_***` function), for each smoking status. Fix the x and y labels to be “Baby Weight” and “Mother Height”, respectively.

```
# Your code goes here
```

6. Plot the density of the baby’s birth weight for each smoking type of the entire `babies` data frame in the same plot. Add the rug plot. Change the color legend title to be “Smoking Type” and the colors to be `c("purple", "red","orange", "cyan")`. Finally, fix the x and y labels to be “Baby Weight” and “Densities”, respectively.

```
# Your code goes here
```