R Lab 0 Exercises
Basic Data Manipulation & Plotting in R

Datasaurus

A. Load in `datasaurus.csv` (on the Canvas page for R Lab 0; more information can be found at github.com/stephlocke/datasauRus) using the following command (change the `file` argument as necessary to point to the correct file path for the computer you're using):

```
dino <- read.csv(file = "./RLab0/datasaurus.csv", header = T)
```

Notes: An alternative function to read in a comma separated file is `read.table()`, adding a `sep = ","` argument. Also, if you have RStudio, note that "`dino`" now appears within the Environment tab on the upper-right of the screen.

B. Run the command `head(dino)`, followed by `names(dino)`.

C. Run the commands `hist(dino$x)` , `boxplot(dino$x)`, and `plot(dino$x)`.

D. Run the command `plot(dino$x, dino$y)`.

E. Run the command `?plot`. Use the results to add a title of "T-Rex" and a subtitle of "technology is truly amazing" to your plot in D.

F. Run the following code, which replaces the first two entries in the `dino$x` vector. Explain in language understandable to an R beginner why the output of the first line differs from the third line.

```
typeof(dino$x)
dino$x[1:2] <- c("hello", "there")
typeof(dino$x)
```

G. Why does the following command return TRUE?

```
identical(dino$y, dino[ , 2])
```

H. What `typeof()` R vector is returned by the following command?

```
dino$y < 20
```

I. With the last question in mind, run the following two commands. What do you conclude about ways you can index things in R?

```
dino_subset_1 <- dino[dino$y < 20, ]
dino_subset_2 <- dino[c(1, 3, 6:10), ]
```

J. Combining code from Part B and the second line of Part F, change the column names of `dino` to your first name and last name. Run `head(dino)` again to make sure that the changes worked.

Traffic Accidents

The file `denvertraffic.txt` contains traffic accident data from the City of Denver open data portal (`denvergov.org/opendata/dataset/city-and-county-of-denver-traffic-accidents`) from July 2017.

A. Read in `denvertraffic.txt`. Name the resulting data frame `traffic`.

B. Print the column names of the dataset. (Revisit the command you used in Part A if you get `V1, V2, …, V20`)

C. Run the following command:

```
dia_only <- traffic[traffic$NEIGHBORHOOD_ID == "DIA", ]
head(dia_only)
```

D. Run `table(traffic$OFFENSE_TYPE_ID)`

E. What does the following code do?

```
traffic[traffic$OFFENSE_TYPE_ID == "TRAF-ACCIDENT-",
        "OFFENSE_TYPE_ID"] <- "TRF-ACCIDENT"
```

F. Use code in the style of Part C to make a data frame consisting of only the offenses marked as a DUI. Save this data frame as `dui_only`.

G. Use `plot()` to make a scatter plot of all traffic accidents, with longitude on the x-axis and latitude on the y-axis. Add the argument `col = "grey"` to make the points grey.

H. If we were to run another plotting command, by default R would overwrite our last plot. To avoid this, try out the command `points()` instead. `points()` takes very

similar arguments to `plot()`.

Use `points()` to add the geographical locations of accidents in `dia_only` data with `col = "blue".` Similarly, add accidents associated with `dui_only` in red.

I. A pound sign (`#`) can be used in R to specify comments. Run the following commands, then annotate it by writing what each line does.

```
# write comment here
bike_accidents <- table(traffic$NEIGHBORHOOD_ID,
                        traffic$BICYCLE_IND)

# write comment here
length(bike_accidents)

# write comment here
prop <- bike_accidents[,2] / (bike_accidents[,1]+bike_accidents[,2])

# write comment here
max(prop)

# write comment here
names(prop)[ prop == max(prop) ]
```

Note: while these examples include comments on their own line, you can also add comments at the end of a line of code. However, any functions included after the pound sign will not run. For example,

```
max(prop) #comment after the max() function

max(prop) #in this case the min() function won't run: min(prop)
```