Lab Session 8

1. mtcars dataset

a. Check to see if you have the mtcars dataset by entering the command

```
mtcars.
```

b. What class is mtcars? `class(mtcars)`

c. How many observations (rows) and variables (columns) are in the mtcars dataset?

```
dim(mtcars)
nrow(mtcars)
ncol(mtcars)
```

d. Copy mtcars into an object called cars and rename mpg in cars to MPG. Use rename().

```
cars=mtcars
cars=rename(cars, MPG=mpg)
```

e. Convert all the column names of cars to upper case. Use rename_all, and the toupper command (or colnames).

```
Mt_upper=rename_all(cars,toupper)
toupper(colnames(cars))
```

f. Convert the rownames of cars to a column called car using rownames_to_column. Subset the columns from cars that end in "p" and call it pvars using ends_with().

```
cars = tibble::rownames_to_column(mtcars, var = "car")
head(cars)
pvars = select(cars, ends_with("p"))
head(pvars)
```

g. Create a subset cars that only contains the columns: wt, qsec, and hp and assign this object to carsSub. What are the dimensions of carsSub? (Use select() and dim().)

```
carsSub = select(cars, wt, qsec, hp)
dim(carsSub)
```

h. Convert the column names of carsSub to all upper case. Use rename_all(), and toupper() (or colnames()).

```
carsSub = rename_all(carsSub, toupper)
```

i. Subset the rows of cars that get more than 20 miles per gallon (mpg) of fuel efficiency. How many are there? (Use filter().)

```
cars_mpg = filter(cars, mpg > 20)
select(cars_mpg,mpg,hp)
```

j. Subset the rows that get less than 16 miles per gallon (mpg) of fuel efficiency and have more than 100 horsepower (hp). How many are there? (Use filter().)

```
nrow(filter(cars, mpg < 16 & hp > 100))
```

k. Create a subset of the cars data that only contains the columns: wt, qsec, and hp for cars with 8 cylinders (cyl) and reassign this object to carsSub. What are the dimensions of this dataset?

```
carsSub = filter(cars, cyl == 8)
carsSub = select(carsSub, wt, qsec, hp, car)
dim(carsSub)
```

l. Re-order the rows of carsSub by weight (wt) in increasing order. (Use arrange().)

```
carsSub = arrange(carsSub, wt)
```

m. Create a new variable in carsSub called wt2, which is equal to wt^2, using mutate() and piping %>%.

```
carsSub %>% mutate(wt2 = wt^2)
```

2. Bike_Lane dataset

```
bike = read.csv("http://johnmuschelli.com/intro_to_r/data/Bike_Lanes.csv")
bike
```

1. How many bike “lanes” are currently in Baltimore? You can assume each observation/row is a different bike “lane”

```
dim(bike)
```

2. How many (a) feet and (b) miles of bike “lanes” are currently in Baltimore?

(a) `sum(bike$length)`

```
sum(bike$length)/5280
```

3. **How many types of bike lanes are there? Which type has (a) the most number of lanes and (b) longest average bike lane length?**

```
colnames(bike)
length(unique(bike$type))   #  n_distinct(bike$type)
m1=bike %>%   group_by(type) %>%   summarise(number_of_rows = n(),
          mean_lane = mean(length)) %>%   arrange(desc(mean_lane))
m1[1,]
filter(m1,number_of_rows==max(number_of_rows)) %>% select(type, number_of_rows)
filter(m1,mean_lane==max(mean_lane)) %>% select(type, mean_lane)
```

4. **How many different projects (project) do the bike lanes fall into? Which project category has the longest average bike lane length?**

```
length(unique(bike$project))
avg = bike %>%   group_by(project) %>%   summarize(mn = mean(length,
na.rm = TRUE)) %>%   filter(mn == max(mn))
avg
```

5. **What was the average bike lane length per year that they were installed? (Be sure to first set dateInstalled to NA if it is equal to zero.)**

```
bike = bike %>% mutate(
  dateInstalled = ifelse( dateInstalled == 0, NA, dateInstalled))
print(mean(bike$length[ !is.na(bike$dateInstalled)]))
```

6. (a)Numerically and (b) graphically describe the distribution of bike lane lengths (length).

```
# Numeric summary
quantile(bike$length)
hist(bike$length)
boxplot(bike$length~bike$type)
```

Data visualization with ggplot2

ggplot() creates a coordinate system that you can add layers to. The first argument of ggplot() is the dataset to use in the graph. You complete your graph by adding one or more layers to ggplot(). The function geom_point() adds a layer of points to your plot, which creates a scatterplot. The mapping argument is always paired with aes(), and the x and y arguments of aes() specify which variables to map to the x and y-axes.

```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy)) #geom_point() adds
 a layer of points to the plot which creates a scatterplot
```

The general plotting functions of `ggplot2` is `ggplot` and is very powerful using the **g**rammar of **g**raphics. When creating a plot, there are two essential attributes of the plot you need to specify: **aesthetics and geoms**
Aesthetics are mappings between the variables in the data and visual properties in the plots. Aesthetics are set in the `aes()` function and the most common aesthetics are

- **x**
- y
- color/colour
- size

- fill
- shape
- linetype
- group

If you set these in `aes`, then you set them to a variable. If you want to set them for all values, set them in a `geom`.
The other essential element of a `ggplot` is a `geom` layer to determine how the data will be plotted.

- `geom_point` **- add points**
- `geom_line` **- add lines**
- `geom_density` **- add density plot**
- `geom_histogram` **- add a histogram**
- `geom_smooth` **- add a smoother**
- `geom_boxplot` **- add a boxplot**
- `geom_bar` **- add a bar chart**
- `geom_tile` **- rectangles/heatmaps**

You add these layer with `+` sign. If you assign a plot to an object, you must call `print` to display it (this is the same a submitting the name of the object to the console).

```
install.packages("tidyverse")

library(tidyverse)

g=mpg%>%ggplot(aes(x = displ, y = hwy))

g

g+geom_line()

gg <- g + geom_line() +
  labs(x = "displacement", y = "hwy", title = "disp vs hwy" )
gg
s1=mpg %>% filter(year %in% 1999)
s1
g = s1 %>% ggplot(aes(x = displ, y = cyl, group = year))
g + geom_line()

ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

```
ggplot(data = mpg) +

  geom_line(mapping = aes(x = displ, y = hwy))

ggplot(data = mpg) +

  geom_smooth(mapping = aes(x = displ, y = hwy))

ggplot(data = mpg) +

  geom_boxplot(mapping = aes(x = displ, y = hwy))
```

```
qplot(x, y=NULL, data, geom="auto", xlim = c(NA, NA), ylim
=c(NA, NA))
```

- **x** : x values
- **y** : y values (optional)
- **data** : data frame to use (optional).
- **geom** : Character vector specifying geom to use. Defaults to “point” if x and y are specified, and “histogram” if only x is specified.
- **xlim**, **ylim**: x and y axis limits

Other arguments including *main*, *xlab*, *ylab* and *log* can be used also:

- **main**: Plot title
- **xlab, ylab**: x and y axis labels

```
# Use data from numeric vectors
x <- 1:10
 y = x*x
# Basic plot qplot(x,y)
# Add line qplot(x, y, geom=c("point", "line"))
# Use data from a data frame qplot(mpg, wt,
data=mtcars)
# Smoothing
```

```
qplot(mpg, wt, data = mtcars, geom = c("point", "smooth"))
```

Smoothed line by groups

The argument **color** is used to tell **R** that we want to color the points by groups:

```
# Linear fits by group
qplot(mpg, wt, data = mtcars, color = factor(cyl),geom=c("point", "smooth"))
```

```
qplot(mpg, data=mtcars, geom=”histogram”,xlab=”miles per gallon”,ylab=”count”, main=”histogram”)
```

Change the shape and the size of points

Like color, the **shape** and the **size** of points can be controlled by a continuous or discrete variable.

```
# Change the size of points according to
  # the values of a continuous variable
qplot(mpg, wt, data = mtcars, size = mpg)
# Change point shapes by groups
qplot(mpg, wt, data = mtcars, shape = factor(cyl))
```