STAT 113: ASSOCIATION BETWEEN QUANTITATIVE VARIABLES

The Consumer Reports 1999 New Car Buying Guide contains lots of information for a large number of new (at that time!) car models. Some of the data for 109 of these cars has been extracted. This activity will focus on the relationships among several of these variables including:

`Weight`	Weight of the car (in pounds)
`CityMPG`	EPA's estimated miles per gallon for city driving
`FuelCap`	Size of the gas tank (in gallons)
`QtrMile`	Time (in seconds) to go 1/4 mile from a standing start
`Acc060`	Time (in seconds) to accelerate from zero to 60 mph
`PageNum`	Page number on which the car appears in the buying guide

1. Consider the relationship you would expect to see between each the following pairs of variables for the car data. Place the letter for each pair on the chart below to indicate your guess as to the direction (negative, neutral, or positive) and strength of the association between the two variables. Note: You may have more than one letter at about the same spot.

   (a) `Weight` vs. `CityMPG`
   (b) `Weight` vs. `FuelCap` (acity)
   (c) `PageNum` (ber) vs. `FuelCap` (acity)
   (d) `Weight` vs. `QtrMile` Time
   (e) `Acc060` vs. `QtrMile` Time
   (f) `CityMPG` vs. `QtrMile` Time

Strong Negative	Moderate Negative	Weak Negative	No Association	Weak Positive	Moderate Positive	Strong Positive

2. Open RStudio, create a new project, and open a new Markdown document from the template. Create a new code chunk with the following code to load the dataset and the `ggformula` package.

```
library(ggformula)
library(mosaic)
Cars99 <- read.file("http://colindawson.net/data/Cars99.csv")
```

3. For each of the six pairs of variables above, make a scatterplot by typing

```
gf_point(Y ~ X, data = Cars99)
```

but with `Y` and `X` replaced by the variable names. For some pairs you will get a warning that rows containing missing values have been removed. This is ok; it's because not every car has every variable recorded.

4. Now that you have looked at the data, revise your initial guesses. How did you do?

Strong Negative	Moderate Negative	Weak Negative	No Association	Weak Positive	Moderate Positive	Strong Positive

5. Compute correlation coefficients for each pair of variables by typing:

```
cor(Y ~ X, data = Cars99, use = "pairwise.complete")
```

(again replacing `Y` and `X` with variable names).

R note: Missing data. The last argument in the command above is needed in this case because the dataset has some missing values for some variables, and the default behavior of the `cor()` function is to report that the correlation is undefined if any cases have missing values for either the `X` or `Y` variables. The `"pairwise.complete"` option tells it to just ignore cases where one of the variables does not have a value.

Record the correlations in the table below (you can round to two decimal places). For a few cases try swapping which variable is `X` and which is `Y`. What happens to the correlation?

Pair	Correlation
(a) `Weight` vs. `CityMPG`	
(d) `Weight` vs. `QtrMile` Time	
(b) `Weight` vs. `FuelCap` (acity)	
(e) `Acc060` vs. `QtrMile` Time	
(c) `PageNum` (ber) vs. `FuelCap` (acity)	
(f) `CityMPG` vs. `QtrMile` Time	

6. Suppose Consumer Reports wanted to convert these variables to the metric system, for non-U.S. readers. To convert weight from pounds to kilograms, we want to divide by about 2.20. To convert miles to kilometers, multiply by about 1.61. To convert gallons to liters, multiply by 3.79 (and to convert from miles / gallon to kilometers / liter, multiply by 1.61 / 3.79). We can create some additional variables in R using the `mutate()` function as follows:

```
## The line breaks in the command below are just for readability,
## and so the command will fit on the page.  They don't actually
## matter to R.
Cars99 <- mutate(Cars99,
                 WeightKg = Weight / 2.20,
                 CityKmPL = CityMPG * (1.61 / 3.79))
### Check what Cars99 looks like now
head(Cars99)
```

R **note: The `mutate()` function**. What is actually going on in the command above? The `mutate()` function is used to create new variables out of old ones. Its first argument is the name of a dataset, and then it can take an arbitrary number of additional arguments (as always, separated by commas) in which the argument name is whatever you want to call the new variable, and the argument value is the formula that defines the new variable. The return value of `mutate()` is a data frame that contains all the old variables in the first argument, plus some additional columns corresponding to the new variables. If we want to actually use these variables in later commands, we need to assign (save) the result to a named variable. In this case, since we are using the same name, we are just overwriting the original dataset. If we had used a different name we would have two copies of the data, one with only the original variables, and one with both old and new variables.

7. Having created some new metric unit variables, recompute the correlation between weight and mileage. What do you notice? You might want to try different combinations of metric and U.S. units.

8. Record three things you notice about sample correlations and their interpretations.

   A.

   B.

   C.