R Notebook: The Linear Model and ggplot in R

Chelsea Estancona

```
rm(list=ls())
library(foreign)
library(ggplot2)
library(xtable)
library(stargazer)
```

When you save the notebook, a PDF file containing the code and output will be saved alongside it. You can also save this as an HTML file- note the settings in 'output' above. What we're doing when we save our file is to create/update two files in our working directory: the 'rmd' file (what we see/edit in R studio) and the pdf file, which is the notebook produced by our code.

Let's dive in. First, we'll simulate some data to use:

```
set.seed(7856)
#allows our simulation to be replicable. Good practice to do this.
x1 <- rnorm(20)
#What is the length of this object?
x2 <- runif(20,5,95) #how about this one? What is its range?

b0 <- 17 #Intercept
b1 <- 0.5 #Beta 1
b2 <- -0.037 #Beta 2
sigma <- 1.5 #variance

s <- rnorm(length(x1),0,sigma) #Let's simulate some random error, too.
y <- b0 + b1*x1  + b2*x2 + s
```

Now that we have simulated independent variables and set up a 'true' relationship between our independent variables (x1, x2) and dependent variable, we can use R's built-in functions to evaluate our linear model.

```
m1<-lm(y~x1+x2) #the basic syntax here: lm(DV~IV1+IV2)
summary(m1) #summarize the 'model object' we've created
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0046 -1.3178 -0.6585  1.5710  4.0946
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 18.01193    1.10216  16.342 7.89e-12 ***
## x1           0.34797    0.66303   0.525   0.6065
## x2          -0.05133    0.01797  -2.857   0.0109 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.185 on 17 degrees of freedom
## Multiple R-squared:  0.3276, Adjusted R-squared:  0.2485
```

```
## F-statistic: 4.141 on 2 and 17 DF,  p-value: 0.03426
```

Let's use some of R's included data to play around with this:

```
data() #I recommend airquality, mtcars, swiss (all have multiple possible IVs)
m2<-lm(mpg~disp+cyl, data=mtcars) #can also specify data within the lm function
summary(m2)
```

```
##
## Call:
## lm(formula = mpg ~ disp + cyl, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4213 -2.1722 -0.6362  1.1899  7.0516
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.66099    2.54700  13.609 4.02e-14 ***
## disp        -0.02058    0.01026  -2.007   0.0542 .
## cyl         -1.58728    0.71184  -2.230   0.0337 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.055 on 29 degrees of freedom
## Multiple R-squared:  0.7596, Adjusted R-squared:  0.743
## F-statistic: 45.81 on 2 and 29 DF,  p-value: 1.058e-09
```

It should be evident that the 'summary' function provides lots of valuable information about the model we've created. We can also extract different information if we need, for example, just the coefficients, or just the residuals:

```
m2$coefficients #extract coefficients
m2$residuals #extract residuals
m2$fitted.values #extract y hat
```

What if we wanted to include an interaction term?

```
m3<-lm(mpg~disp+cyl+disp*cyl, data=mtcars) #use * for interaction- include both variables
summary(m3)
```

```
##
## Call:
## lm(formula = mpg ~ disp + cyl + disp * cyl, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0809 -1.6054 -0.2948  1.0546  5.7981
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 49.037212   5.004636   9.798 1.51e-10 ***
## disp        -0.145526   0.040002  -3.638 0.001099 **
## cyl         -3.405244   0.840189  -4.053 0.000365 ***
## disp:cyl     0.015854   0.004948   3.204 0.003369 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 2.66 on 28 degrees of freedom
## Multiple R-squared:  0.8241, Adjusted R-squared:  0.8052
## F-statistic: 43.72 on 3 and 28 DF,  p-value: 1.078e-10
```

How might we present these results in a table?

```
stargazer(m3)
xtable(m3)
#These produce code for a .pdf that we can copy and paste.
```

EXERCISE 1: Fill in the chunk below with your R code and output for the following:
-Access an R dataset (airquality, mtcars, swiss)
-Fit a linear model with at least two independent variables and an interaction term.
-Include a comment about the relationship (or lack thereof) in your model.
-Summarize your linear model. Include the code to create a table from it.

Moving on to plotting with ggplot. Use install.packages("ggplot2") if you haven't already). R has its own built-in plot function, but ggplot is far more flexible and creates far more descriptive (and thus useful!) plots.

For a comprehensive 'cheat sheet', see: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

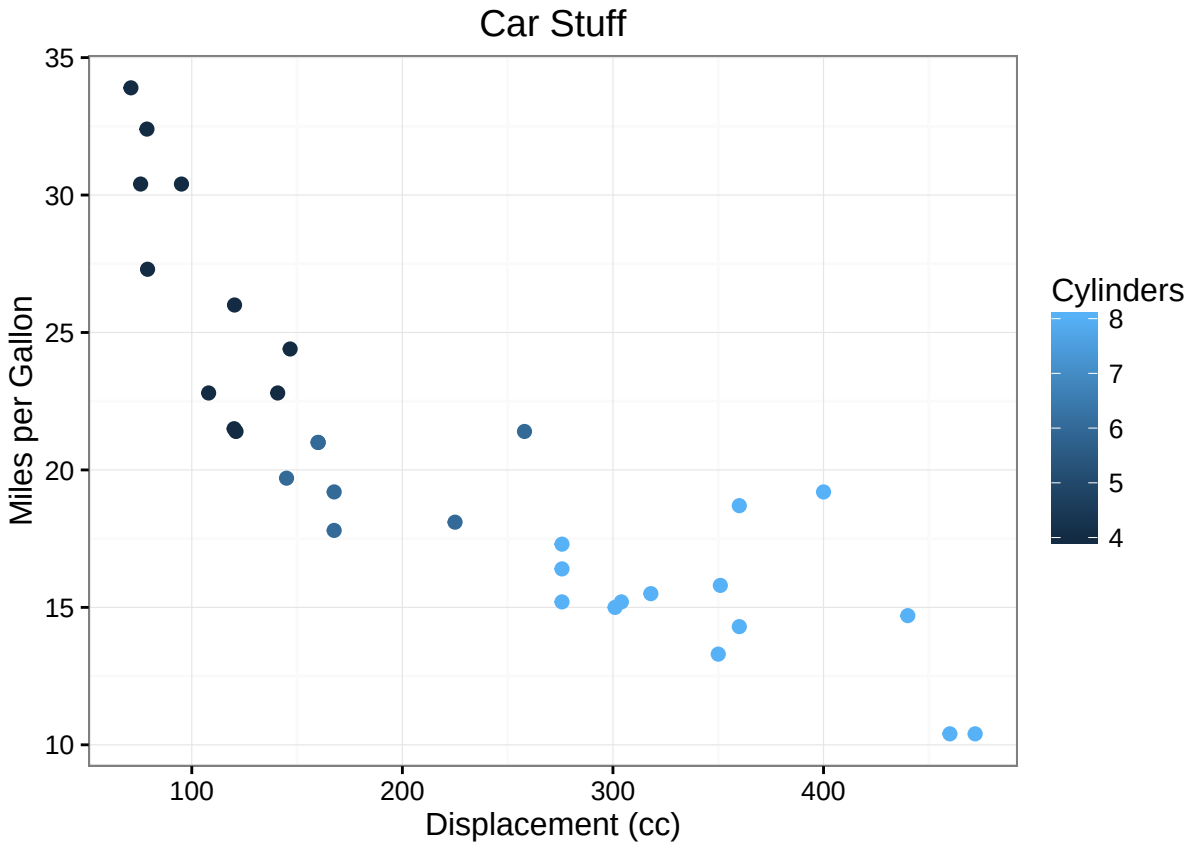
To begin, we can use the 'qplot' or 'ggplot' commands. I prefer 'ggplot' for flexibility.

```
qplot(x=disp, y=mpg, data=mtcars, color=cyl, geom="point")
#x and y are independent + dependent variables,
#data can be specified (or use mtcars$var),
#color distinguishes a characteristic over which to map color
#and geom is a plot type

ggplot(data=mtcars, aes(x=disp, y=mpg)) + geom_point(aes(color=cyl))
#Note the similarities above.
#ggplot is more flexible in that you map to different `aesthetics'-
#hence why we use 'aes.' Plenty of examples below to dig into this more!
```

What are some ways we can make this plot clearer?

Axes? Labels? Etc?

```
ggplot(data=mtcars, aes(x=disp, y=mpg)) + #we add 'layers' with +
  geom_point(aes(colour=cyl), size=2) + #point=scatterplot
  labs(title="Car Stuff", x="Displacement (cc)", y="Miles per Gallon")+
  #label x and y axes, give the plot a main title. Note the quotes!
  scale_colour_continuous(name="Cylinders")+ #label the scale
  theme_bw() #make the plot background blank w/grid lines
```

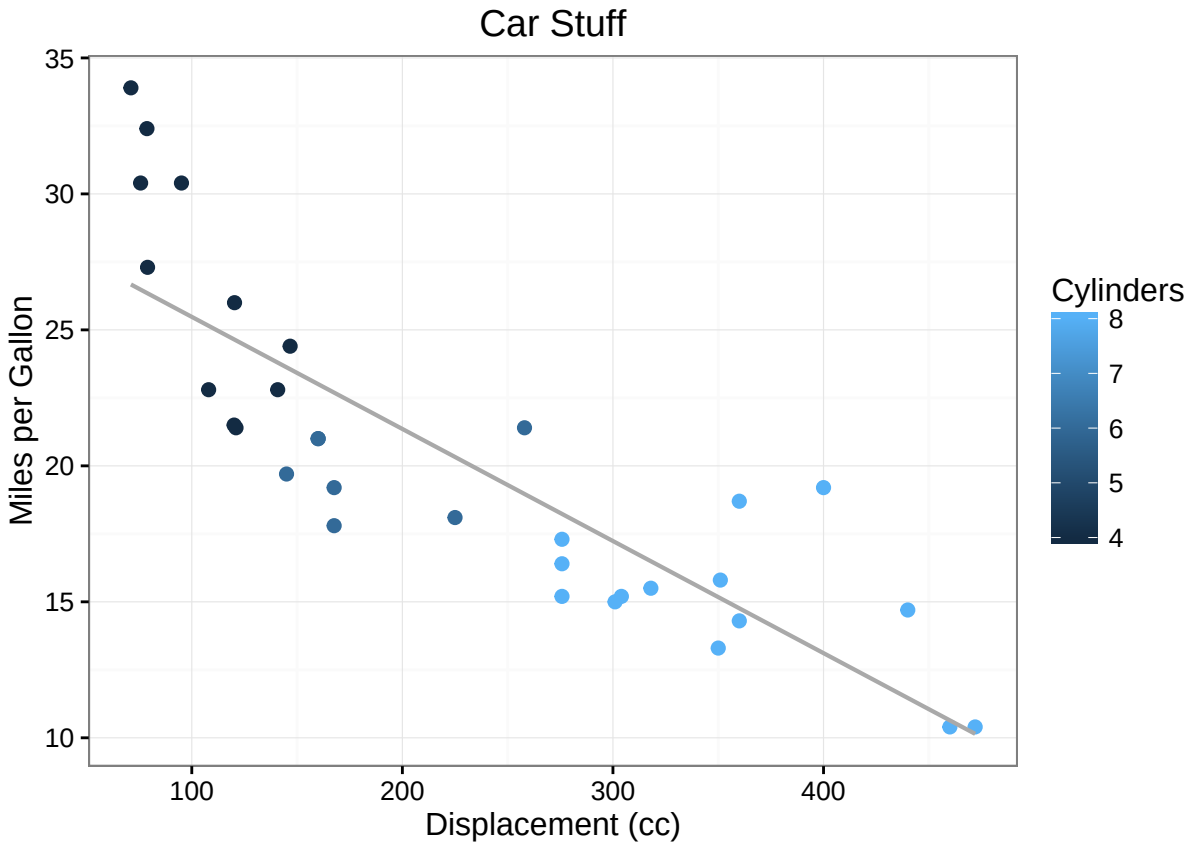


Reminders/basics: assign this plot to an object and save that object as a .pdf.

```
carsplot<-ggplot(data=mtcars, aes(x=disp, y=mpg)) +
  geom_point(aes(colour=cyl), size=2) +
  labs(title="Car Stuff", x="Displacement (cc)", y="Miles per Gallon")+
  scale_colour_continuous(name="Cylinders")+
  theme_bw()

#One method for saving plots:
#pdf("Plot1.pdf")
#carsplot
#dev.off()

#Where do these plots go?!

#Unique to ggplot:
#ggsave("Plot1a.pdf") #can edit image sizes as well
```

What if we wanted to plot the linear relationship between our x and y?

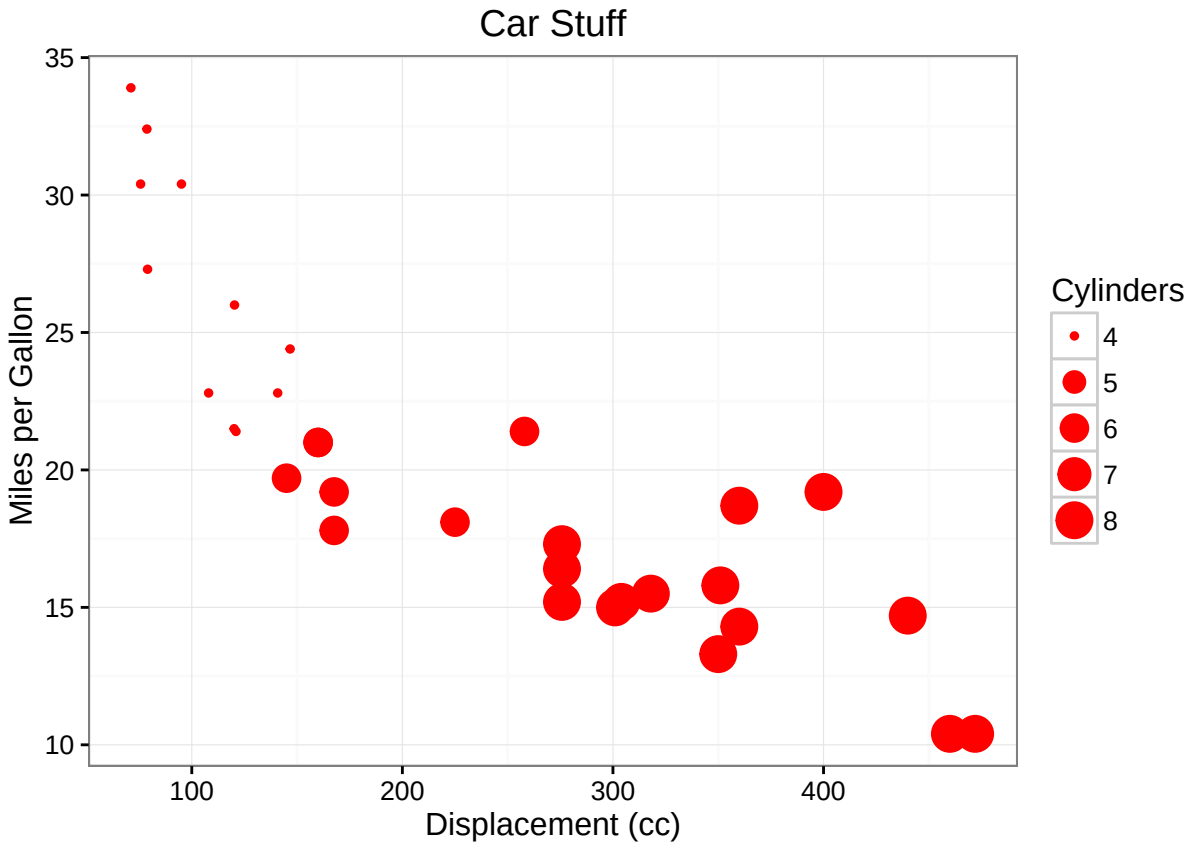
```
carsplot+geom_smooth(method="lm", se=FALSE, color="darkgrey", size=0.75)
```



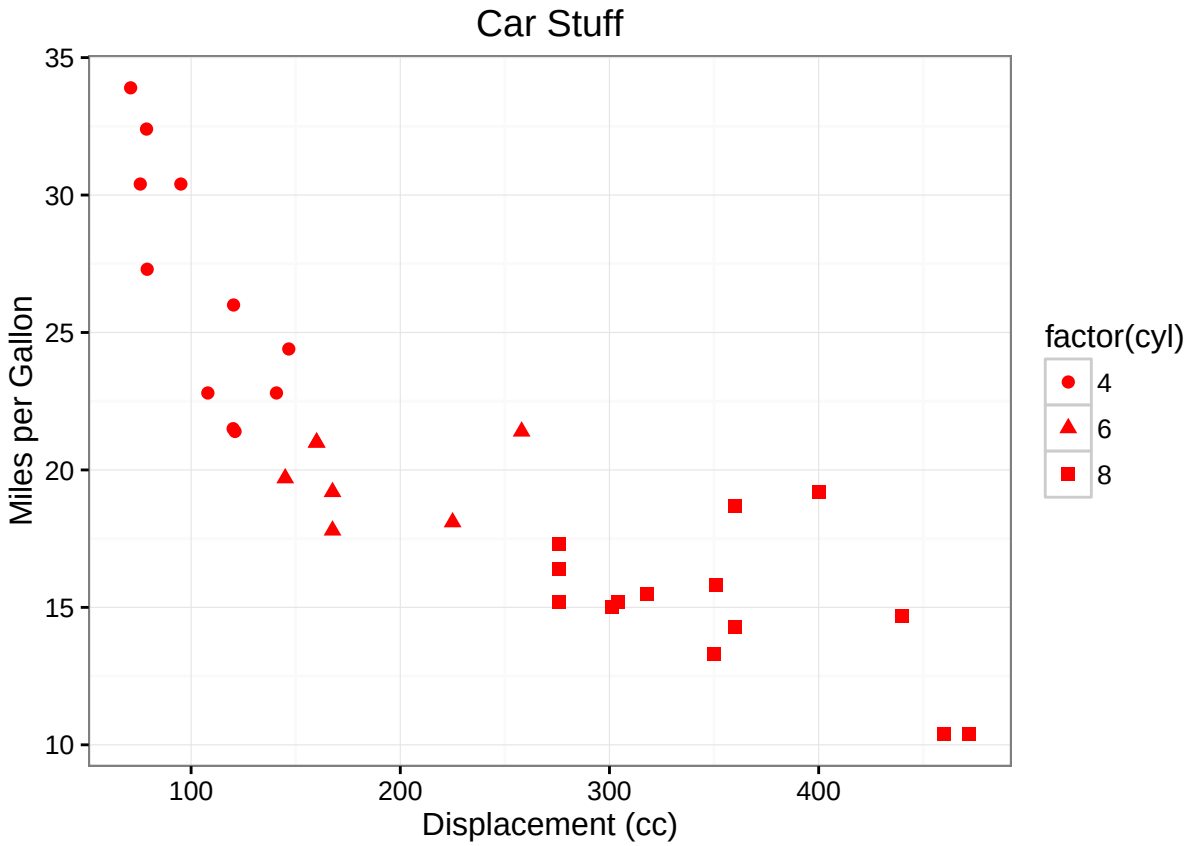

```
#geom_smooth is for fitted lines.
#se=TRUE plots confidence intervals
#We can adjust the color and size of the line plotted.
```

Right now, we're using color as the aesthetic mapped to a specific independent variable (color represents the number of cylinders). We can play around with this:

```
ggplot(data=mtcars, aes(x=disp, y=mpg)) +
  geom_point(aes(size=cyl), colour="red") +
  labs(title="Car Stuff", x="Displacement (cc)", y="Miles per Gallon")+
  scale_size_continuous(name="Cylinders")+
  theme_bw()
```



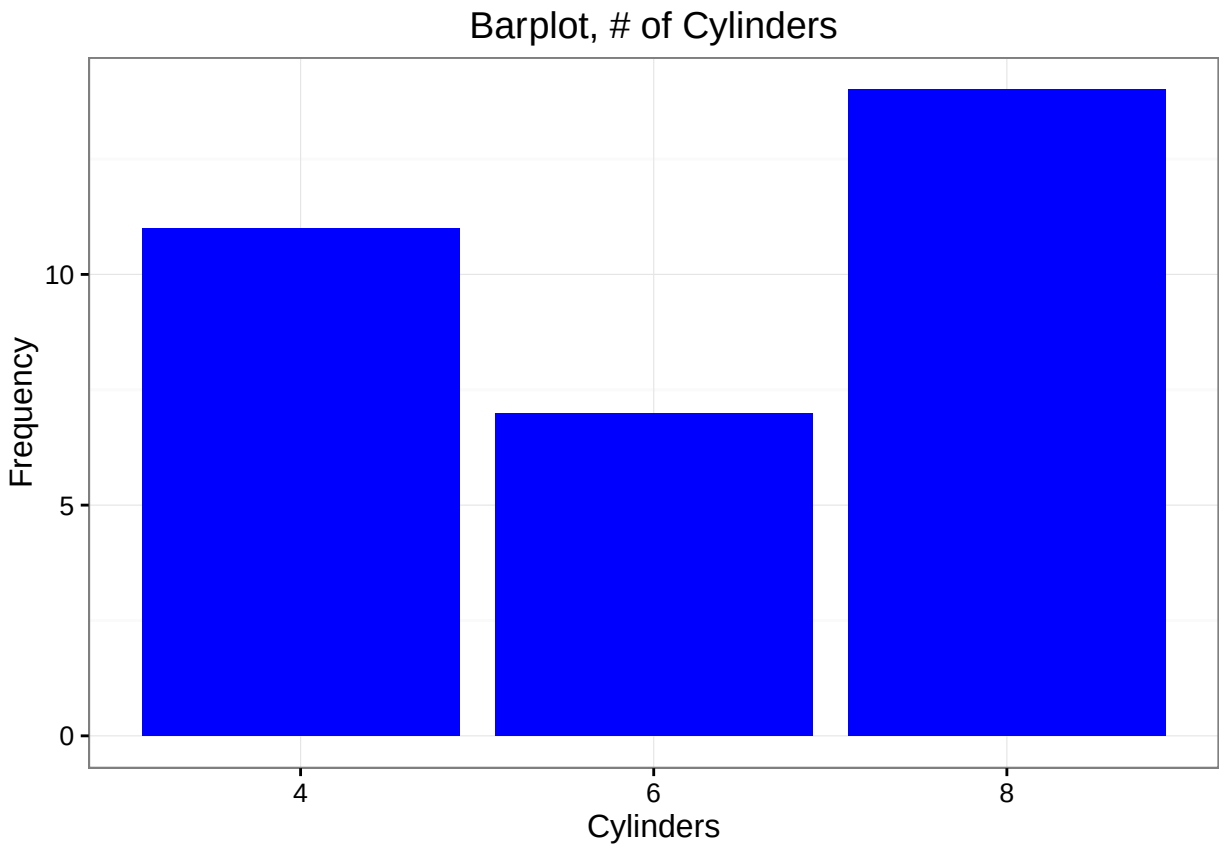

```
ggplot(data=mtcars, aes(x=disp, y=mpg)) +
  geom_point(aes(shape=factor(cyl)), colour="red", size=2) +
  labs(title="Car Stuff", x="Displacement (cc)", y="Miles per Gallon")+
  scale_size_continuous(name="Cylinders")+
  theme_bw()
```



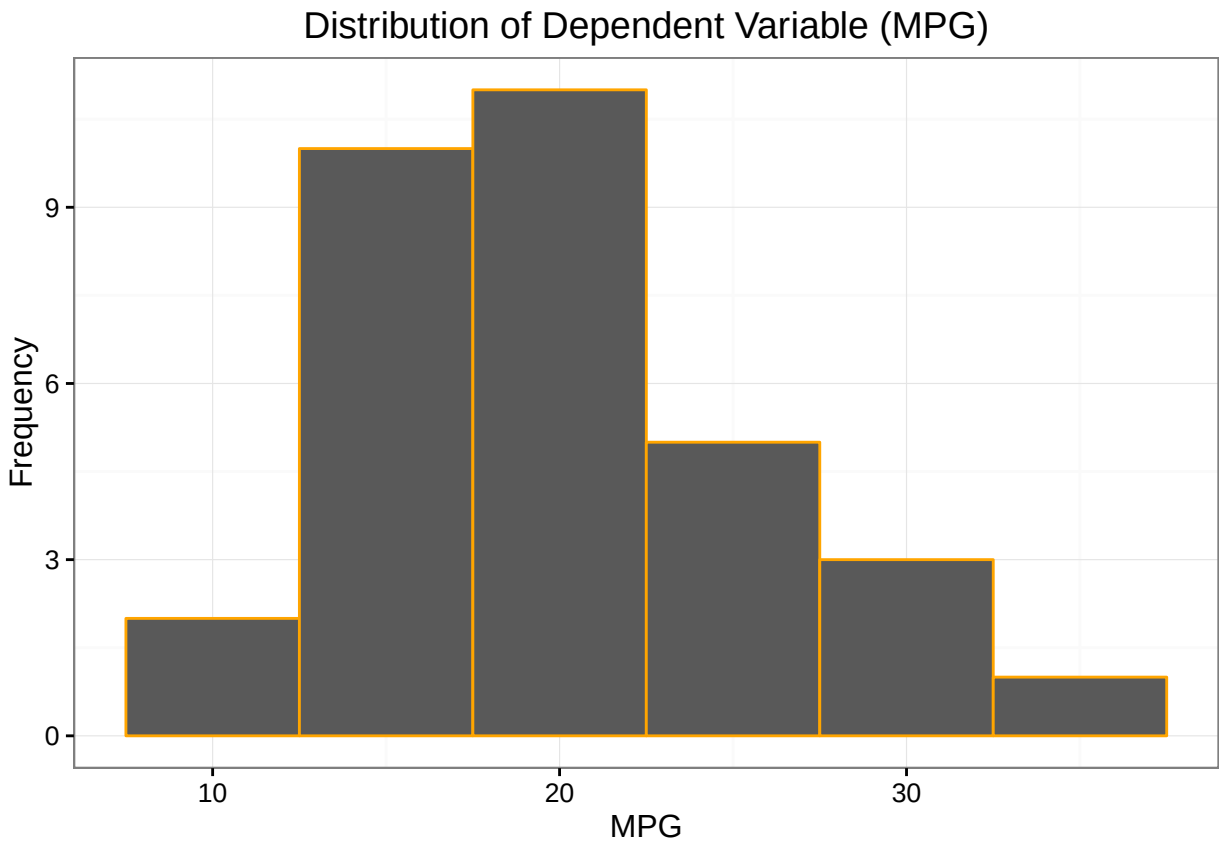
What are the benefits/drawbacks of each of these strategies? Think about how well the plot communicates information about the types of variables plotted as well as the relationship between them.

What if we're actually more interested in the relationship between # of cylinders and the miles per gallon for each car type? We might first want to plot each of these quantities separately and see their respective distributions (helpful for descriptive stats both of your DV and IVs)!

```
ggplot(mtcars, aes(x=factor(cyl))) +
  #factor(cyl) ensures that the variable is treated categorically
  geom_bar(fill="blue") +
  labs(title="Barplot, # of Cylinders", x="Cylinders", y="Frequency")+
  theme_bw()
```

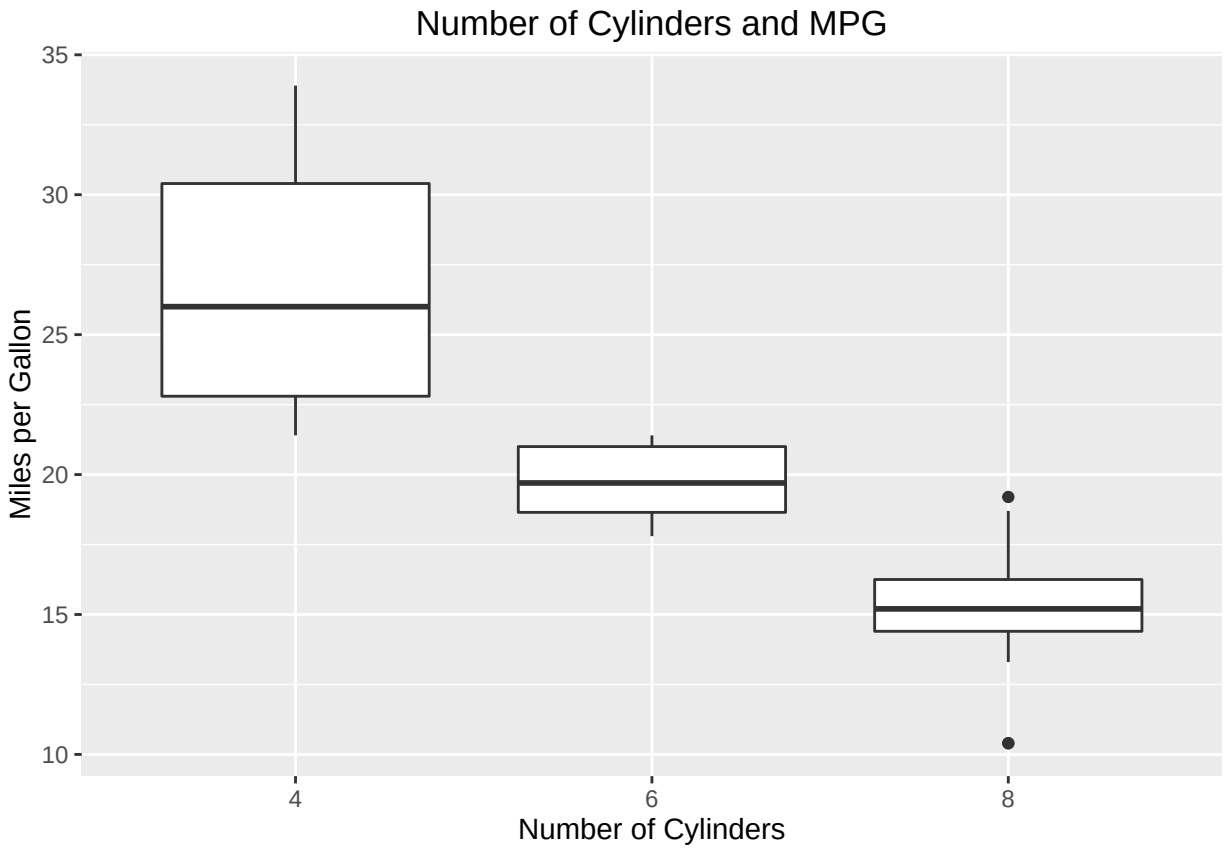



```
ggplot(mtcars, aes(x=mpg)) +
  geom_histogram(stat="bin", binwidth=5, color="orange")+
  labs(title="Distribution of Dependent Variable (MPG)", x="MPG", y="Frequency")+
  theme_bw()
```



We'd also need to adjust a plot that primarily looks at the relationship between the two variables.

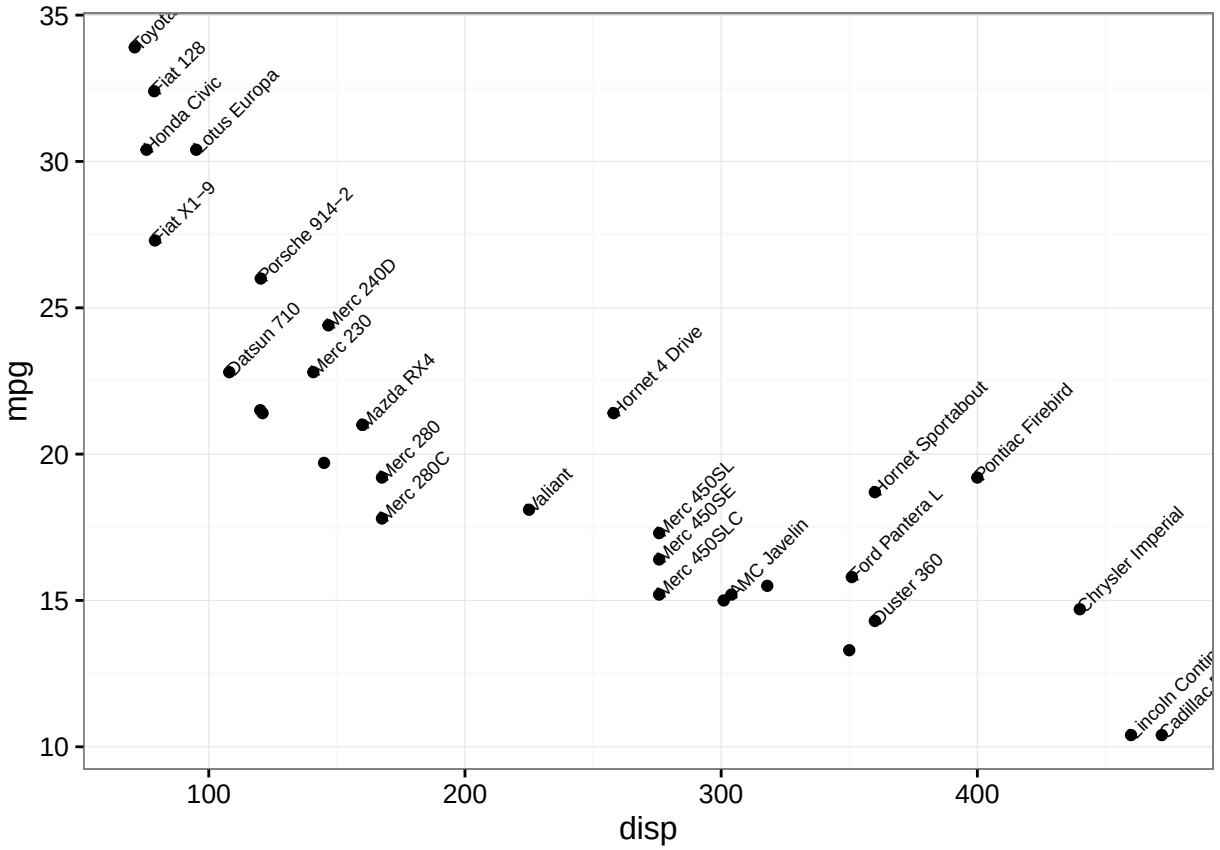
```
box<-ggplot(data=mtcars, aes(x=factor(cyl), y=mpg))+
geom_boxplot()+
labs(title="Number of Cylinders and MPG", x="Number of Cylinders", y="Miles per Gallon")

box
```




```
#If we wanted, we could add the same line as before with our data.
#box+geom_smooth(method="lm", se=FALSE, color="darkgrey", size=0.75, aes(group=1))
```

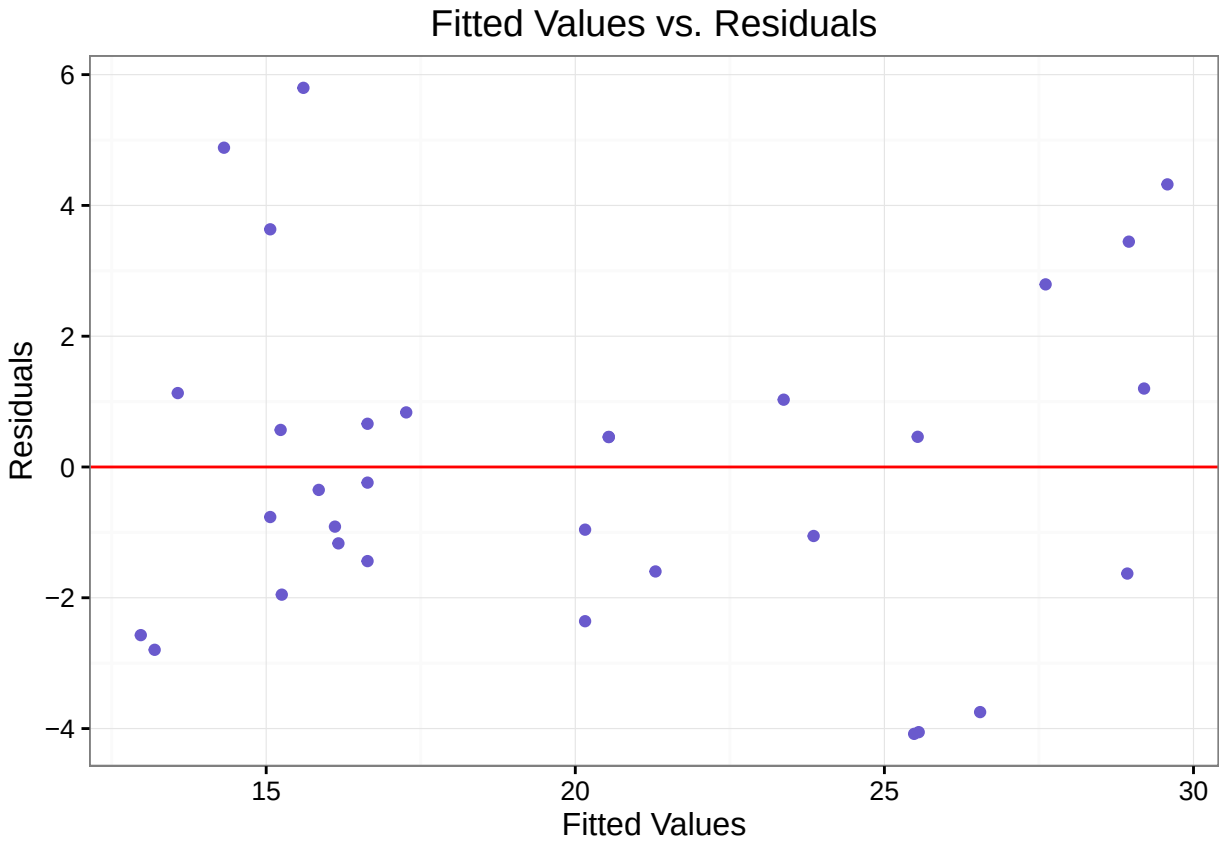
What if we have qualitative characteristics (like labels) we're interested in plotting?

```
ggplot(mtcars, aes(disp, mpg, label = rownames(mtcars)))+
  geom_point()+
  geom_text(angle=45, check_overlap=TRUE, size=2.5, hjust=0, nudge_x=0.1)+
  #angle of text, overlap, text size, height away from bullet, nudge horizontally
  theme_bw()
```



Finally, let's make a plot of our residuals vs. our fitted values - a good way to check for any disturbing tendencies in our data!

```
test<-as.data.frame(cbind(m3$fitted.values,m3$residuals))

ggplot(data=test, aes(x=V1, y=V2)) + #we add 'layers' with +
  geom_point(colour="slateblue") + #point=scatterplot, color specified
  geom_hline(yintercept=0, colour="red")+
  labs(title="Fitted Values vs. Residuals", x="Fitted Values", y="Residuals")+
  #label x and y axes, give the plot a main title. Note the quotes!
  theme_bw()
```



EXERCISE 2: Make at least 4 ggplots using your R data from above. Adjust color, shape, and size as desired, and give each plot meaningful axes/main plot labels.

-A plot showing a linear relationship of interest
-A plot describing the distribution of one of your IVs
-A plot showing the distribution of your DV
-A diagnostic plot (fitted vals vs. residuals is a good example)