Lab 6 - Diagnostics and simulation based inferences: Answers

TA: Eric Parajon

This lab has been adapted from code by Simon Hoellerbauer and Isabel Laterzo.

So far in lab, we have often simulated data to allow us to test out various methods. Simulation can help us do much more, as well. Simulations are a useful tool that is being used more and more often within the discipline. They can help us: 1) Construct a hypothetical DGP and then see if the methods we use/develop can successfully retrieve the parameters of that DGP 2) Make our data go further with re-sampling 3) Create plots to help us assess our results when our variance/covariance matrix gets complex (when distributional assumptions can be tricky)

Today, we will use simulation to help us assess the uncertainty around our parameter estimates and will then incorporate this uncertainty into the predicted values of our outcome variable.

```
#clear global environ
rm(list=ls())

#the packages we will need

#Installing and loading packages for use
pacman::p_load(MASS,#required for multivariate normal
               tidyverse,
               AER)
```

Download the "aid_data_95_ee.csv" in the Labs folder on Sakai and then read it into R:

```
#remember to set your working directory!

#read_csv comes from readr from the tidyverse
aid_data_95_ee <- read_csv("aid_data_95_ee.csv")
```

Let's take a look at this data very quickly:

```
#checking dimensions
dim(aid_data_95_ee)
```

```
## [1] 477  76
```

```
#that's a lot of variables!

#which variables?
names(aid_data_95_ee)
```

```
##  [1] "country_iso3c"           "year"
##  [3] "recipient"               "civ_soc_aid"
##  [5] "gov_aid"                 "other_aid"
##  [7] "dem_aid"                 "dem_aid_all"
##  [9] "country_id"              "elec_dem"
## [11] "lib_dem"                 "delib_dem"
## [13] "egal_dem"                "participatory_dem"
```

```
## [15] "corruption_index"         "civ_soc_strength"
## [17] "civ_soc_part"             "GDPpc_vdem"
## [19] "exchange_rate"            "gini_vdem"
## [21] "pop_mil_vdem"             "pop_vdem"
## [23] "iso2c"                    "country"
## [25] "GDPpc_WB_2010"            "GDPpc_ppp_WB_2011"
## [27] "GDPpc_growth_WB"          "GINI_WB"
## [29] "pop_WB"                   "region"
## [31] "elec_dem2"                "lib_dem2"
## [33] "egal_dem2"                "delib_dem2"
## [35] "participatory_dem2"       "civ_soc_aid_cumul"
## [37] "gov_aid_cumul"            "other_aid_cumul"
## [39] "civ_soc_strength_l1"      "civ_soc_aid_l1"
## [41] "gov_aid_l1"               "other_aid_l1"
## [43] "elec_dem_l1"              "elec_dem2_l1"
## [45] "lib_dem_l1"               "lib_dem2_l1"
## [47] "delib_dem_l1"             "delib_dem2_l1"
## [49] "egal_dem_l1"              "egal_dem2_l1"
## [51] "participatory_dem_l1"     "participatory_dem2_l1"
## [53] "GDPpc_ppp_WB_2011_l1"     "civ_soc_aid_cumul_l1"
## [55] "gov_aid_cumul_l1"         "other_aid_cumul_l1"
## [57] "EU_accession"             "civ_soc_str_3yr"
## [59] "civ_soc_str_2yr"          "civ_soc_str_4yr"
## [61] "civ_soc_str_diff1"        "civ_soc_aid_diff1"
## [63] "civ_soc_cumul_diff1"      "gov_aid_diff1"
## [65] "gov_cumul_diff1"          "other_aid_diff1"
## [67] "other_cumul_diff1"        "civ_soc_cumul_diff1_l1"
## [69] "gov_cumul_diff1_l1"       "other_cumul_diff1_l1"
## [71] "civ_soc_str_3yr_l1"       "civ_soc_str_2yr_l1"
## [73] "civ_soc_aid_l1_log"       "gov_aid_l1_log"
## [75] "other_aid_l1_log"         "GDPpc_ppp_WB_2011_l1_log"
```

```
#which countries?
unique(aid_data_95_ee$recipient)
```

```
##  [1] "Albania"            "Armenia"            "Azerbaijan"
##  [4] "Bulgaria"           "Bosnia-Herzegovina" "Belarus"
##  [7] "Estonia"            "Georgia"            "Croatia"
## [10] "Hungary"            "Kazakhstan"         "Kyrgyz Republic"
## [13] "Lithuania"          "Latvia"             "Moldova"
## [16] "Macedonia, FYR"     "Montenegro"         "Poland"
## [19] "Romania"            "Russia"             "Serbia"
## [22] "Slovak Republic"    "Slovenia"           "Tajikistan"
## [25] "Turkmenistan"       "Ukraine"            "Uzbekistan"
## [28] "Kosovo"
```

```
#for which years?
unique(aid_data_95_ee$year)
```

```
##  [1] 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009
## [16] 2010 2011 2012 2013
```

```
#how many countries in years? dplyr can help us here
aid_data_95_ee %>% group_by(recipient) %>% tally
```

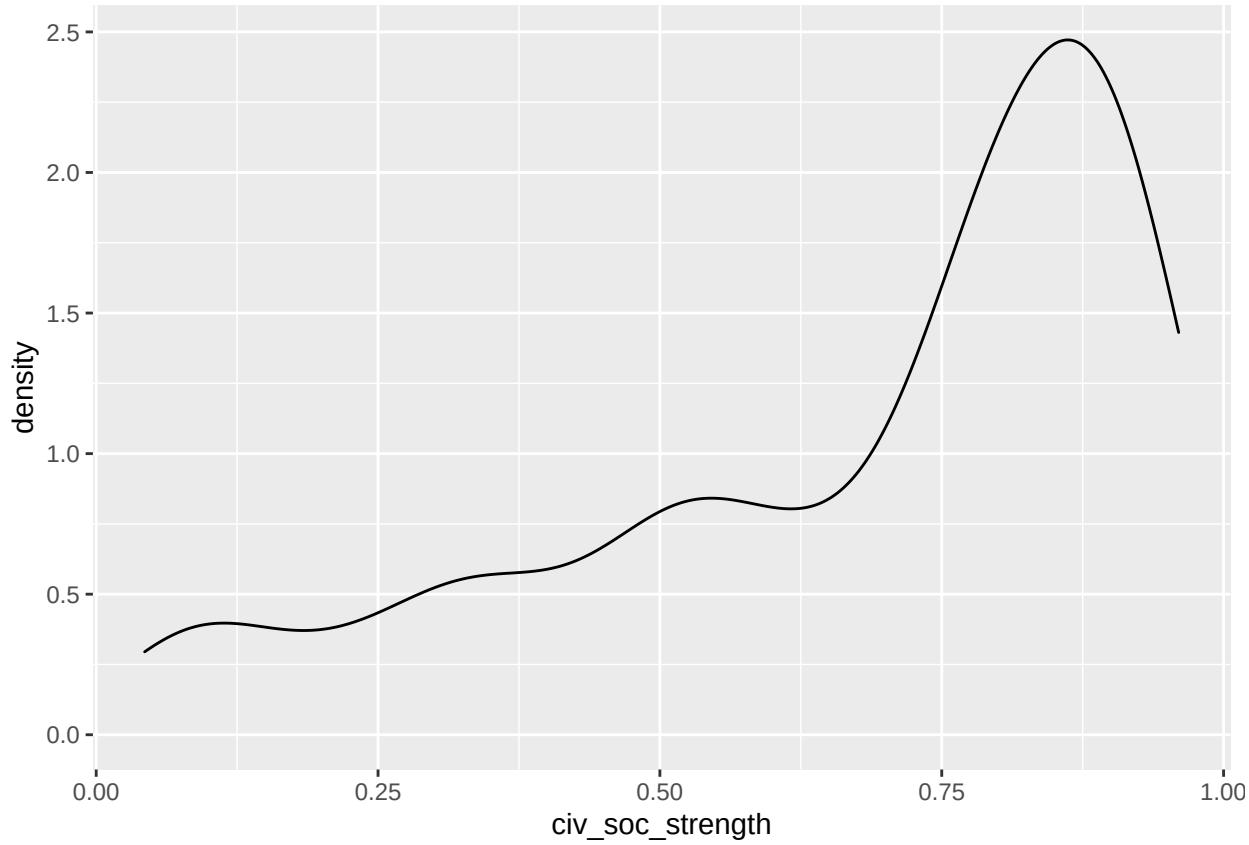
```
## # A tibble: 28 x 2
```

```
##    recipient              n
##    <chr>              <int>
##  1 Albania               19
##  2 Armenia               19
##  3 Azerbaijan            19
##  4 Belarus               19
##  5 Bosnia-Herzegovina    19
##  6 Bulgaria              17
##  7 Croatia               17
##  8 Estonia               11
##  9 Georgia               19
## 10 Hungary               16
## # ... with 18 more rows
## # i Use `print(n = ...)` to see more rows
```

We are going to see how civil society strength in Eastern Europe varies according to lagged civil society assistance, EU accession leverage, and lagged GDP per capita.

Let's look at the distribution of the outcome variable (with some simple, barebones graphs):

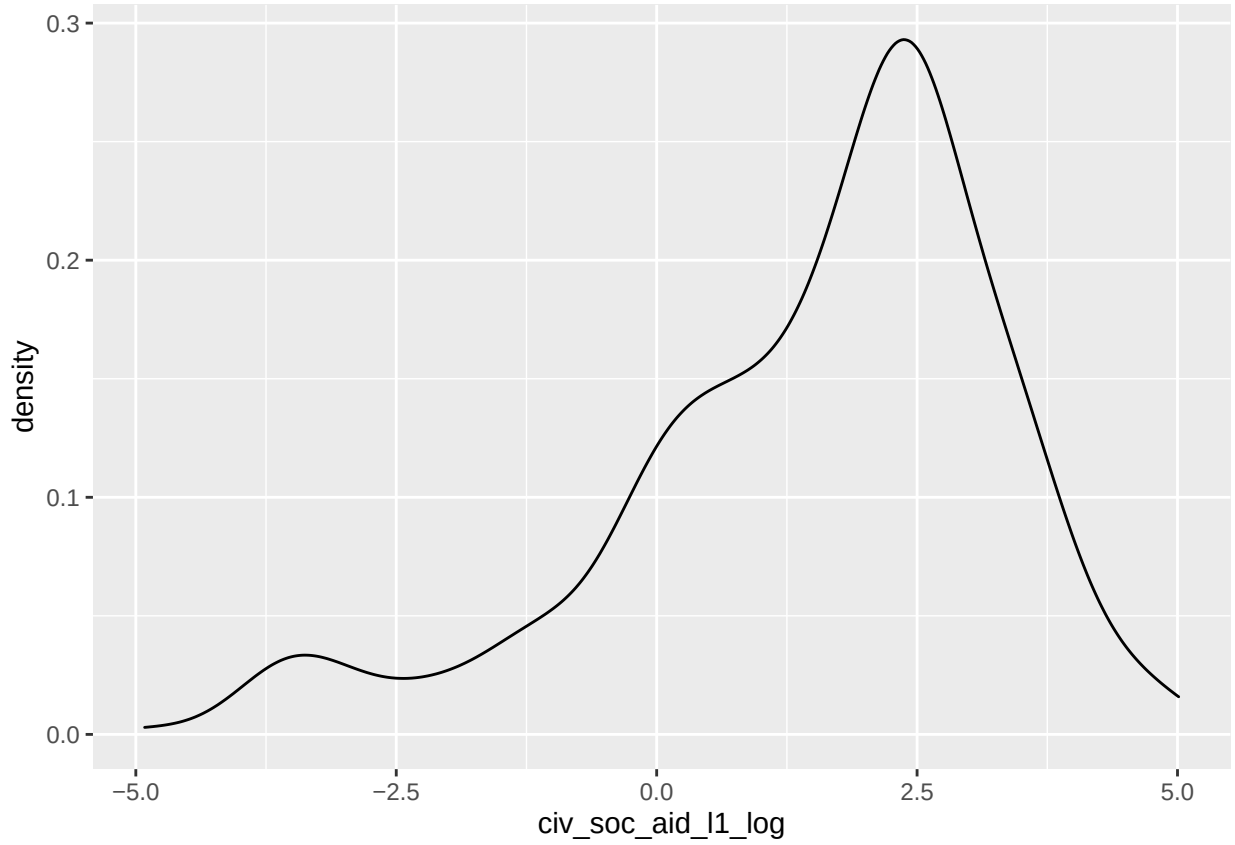
```
#distribution of civil society strength
ggplot(data = aid_data_95_ee, aes(x = civ_soc_strength)) +
  geom_density()
```




```
#Is it necessarily a problem that our outcome variable is skewed?

#what does our main predictor of interest look like?
ggplot(data = aid_data_95_ee, aes(x = civ_soc_aid_l1_log)) +
```

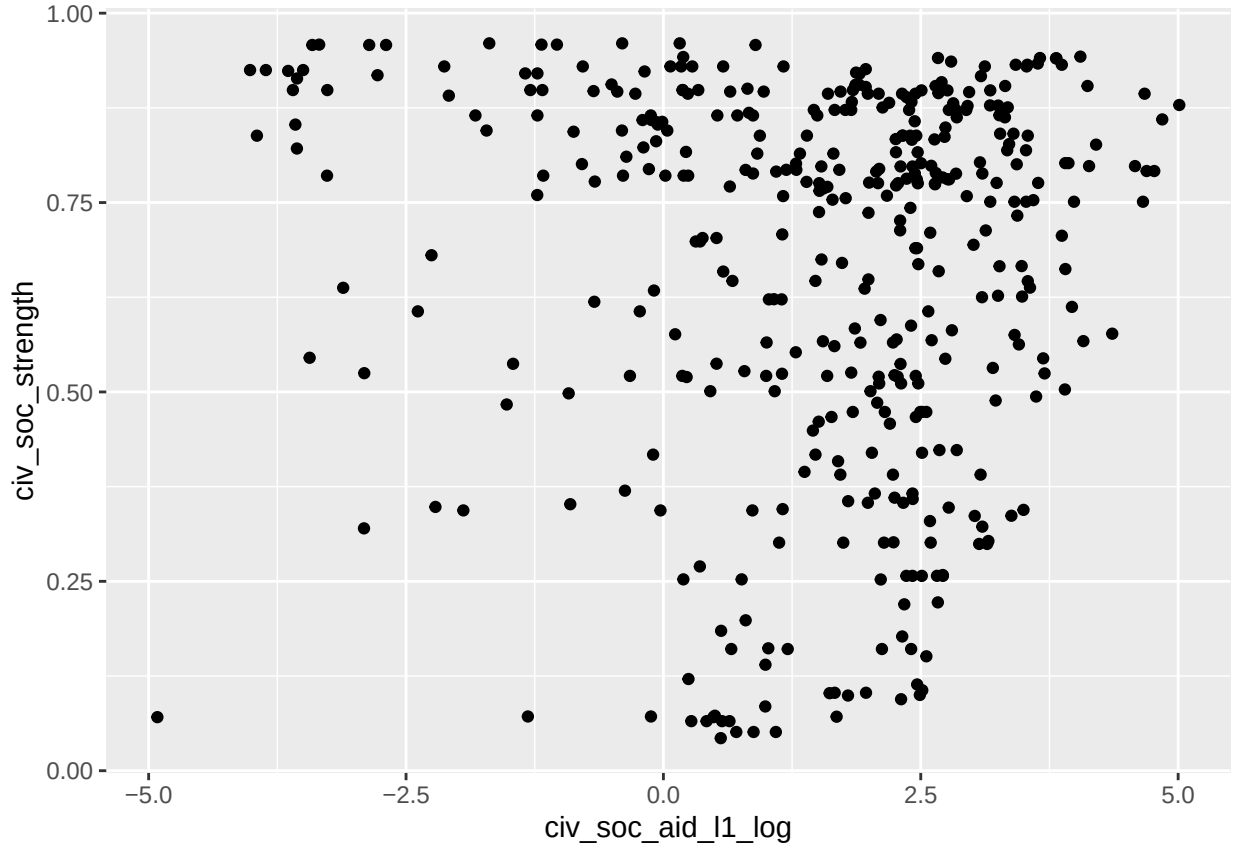
```
  geom_density()
```

```
## Warning: Removed 68 rows containing non-finite values (stat_density).
```

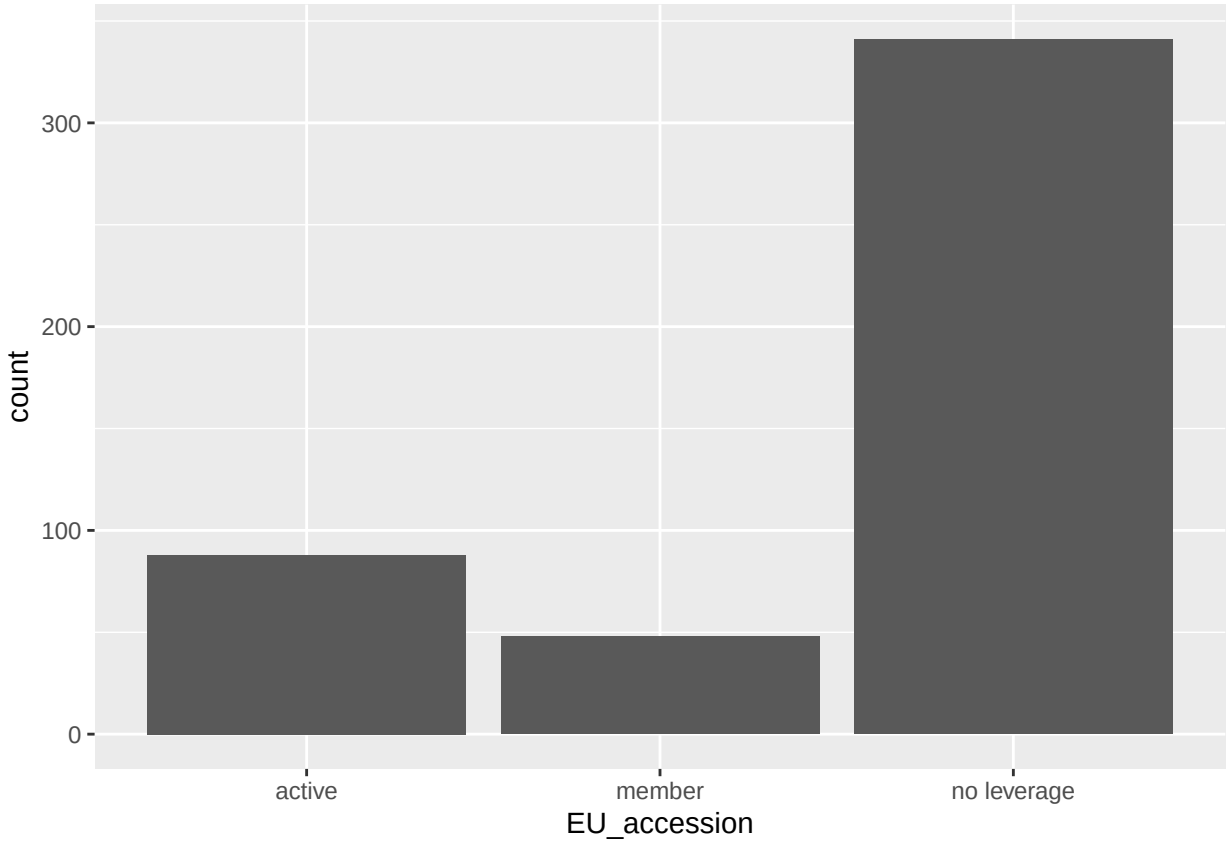



```
#why do you think we log civil society aid?

#let's look at relationship between the two
ggplot(data = aid_data_95_ee, aes(x = civ_soc_aid_l1_log, y = civ_soc_strength)) +
  geom_point()
```

```
## Warning: Removed 68 rows containing missing values (geom_point).
```




```
#what about EU accession?
ggplot(data = aid_data_95_ee, aes(x = EU_accession)) + geom_bar()
```



Let's fit our model:

```
#first we have to make sure that EU_accession is a factor, and that the
#reference category makes sense for us
aid_data_95_ee$EU_accession <- factor(aid_data_95_ee$EU_accession) %>%
  relevel(ref = "no leverage")
levels(aid_data_95_ee$EU_accession) #First one will be the reference catagory
```

```
## [1] "no leverage" "active"      "member"
```

```
#Estimating the model
cs_model <- lm(civ_soc_strength ~ GDPpc_ppp_WB_2011_l1_log +
                 EU_accession*civ_soc_aid_l1_log, data = aid_data_95_ee)
summary(cs_model)
```

```
##
## Call:
## lm(formula = civ_soc_strength ~ GDPpc_ppp_WB_2011_l1_log + EU_accession *
##     civ_soc_aid_l1_log, data = aid_data_95_ee)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52359 -0.11256  0.02016  0.16938  0.45719
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          0.218656   0.148281   1.475  0.14110
## GDPpc_ppp_WB_2011_l1_log             0.034771   0.017227   2.018  0.04422 *
```

```
## EU_accessionactive                       0.339604   0.036321   9.350  < 2e-16 ***
## EU_accessionmember                       0.336745   0.074468   4.522 8.08e-06 ***
## civ_soc_aid_l1_log                       0.035276   0.007794   4.526 7.93e-06 ***
## EU_accessionactive:civ_soc_aid_l1_log -0.041988   0.015400  -2.727  0.00668 **
## EU_accessionmember:civ_soc_aid_l1_log -0.032931   0.031997  -1.029  0.30401
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2165 on 400 degrees of freedom
##   (70 observations deleted due to missingness)
## Multiple R-squared:  0.2939, Adjusted R-squared:  0.2833
## F-statistic: 27.75 on 6 and 400 DF,  p-value: < 2.2e-16
```

Now simulation comes in! We can use simulation to get confidence intervals. Here, we will be using the function `mvrnorm` to simulate some data. Look at the help file for that function and what it does.

```
#set seed for replication
set.seed(123)

#We will take 500 samples from the sampling distribution
#of our model's coefficients
samp_beta <- mvrnorm(500,
                     coef(cs_model),
                     vcov(cs_model))
dim(vcov(cs_model))
```

```
## [1] 7 7
```

```
dim(samp_beta)
```

```
## [1] 500   7
```

```
# 87% Monte Carlo/simulation based confidence interval:
t(apply(samp_beta,
        2, #why 2? look at help file and the "MARGIN" argument
        quantile,
        probs = c(0.065, 0.935)))
```

```
##                                             6.5%       93.5%
## (Intercept)                           0.003178624  0.45130749
## GDPpc_ppp_WB_2011_l1_log              0.006387482  0.05970041
## EU_accessionactive                    0.282523334  0.40207168
## EU_accessionmember                    0.222259417  0.45629586
## civ_soc_aid_l1_log                    0.025003918  0.04649095
## EU_accessionactive:civ_soc_aid_l1_log -0.065507423 -0.02064103
## EU_accessionmember:civ_soc_aid_l1_log -0.086996118  0.01539446
```

Let's compare that to CIs generated from the function `confint()`. Both produce asymptotic confidence intervals, but the first uses resampling/Monte Carlo methods. I suggest using these methods, as it is easier for us to get CIs for slightly more complicated models via simulation.

```
confint(cs_model, level = 0.87)
```

```
##                                              6.5 %      93.5 %
## (Intercept)                           -0.006319173  0.44363131
## GDPpc_ppp_WB_2011_l1_log               0.008633669  0.06090779
## EU_accessionactive                     0.284496766  0.39471026
## EU_accessionmember                     0.223760575  0.44972868
```

```
## civ_soc_aid_l1_log                       0.023451120  0.04710138
## EU_accessionactive:civ_soc_aid_l1_log -0.065353020 -0.01862332
## EU_accessionmember:civ_soc_aid_l1_log -0.081477035  0.01561493
```

We are now going to use the "average value approach" to simulating predicted outcomes and uncertainty around them. We want to see how predicted civil society strength varies according to lagged civil society assistance. In order to do this, there are a few thing we have to do first:

```
# we are going to let lagged log civil society assistance vary along its
# inner-quartile range. These will be the values over which we plot the predicted
# civil society strength
cs_aid_sim <- with(aid_data_95_ee,
                   #quantile() here produces sample quantities based on the given
                   #probability we provide it, here 0.25
                   seq(quantile(civ_soc_aid_l1_log, 0.25, na.rm = T),
                       quantile(civ_soc_aid_l1_log, 0.75, na.rm = T),
                       length.out = 100))

# we now need to create a matrix of hypothetical predictors. We will use the
# mean of lagged log GDP per capita; in order to see the effect of EU accession
# leverage, we will also allow that to vary (otherwise, you usually use the mode
# of categorical variables)
x_mat_sim <- with(aid_data_95_ee,
                  model.matrix(~ GDPpc_ppp_WB_2011_l1_log +
                                 EU_accession*civ_soc_aid_l1_log,

                               data = data.frame(GDPpc_ppp_WB_2011_l1_log =
                                 mean(GDPpc_ppp_WB_2011_l1_log,
                                      na.rm = T), #mean of lagged log GDP

                                      EU_accession = relevel(
                                 factor(c("no leverage",
                                          "active", # vary EU accession
                                          "member")),
                                 ref = "no leverage"),

                                      #calling above data for civ soc assist
                                      civ_soc_aid_l1_log = rep(cs_aid_sim,
                                                               each = 3))))


#let's check dimensions
dim(x_mat_sim)
```

```
## [1] 300   7
```

```
#how does it look? note the interaction effects included!
head(x_mat_sim)
```

```
##   (Intercept) GDPpc_ppp_WB_2011_l1_log EU_accessionactive EU_accessionmember
## 1           1                 8.971782                  0                  0
## 2           1                 8.971782                  1                  0
## 3           1                 8.971782                  0                  1
## 4           1                 8.971782                  0                  0
## 5           1                 8.971782                  1                  0
## 6           1                 8.971782                  0                  1
```

```
##   civ_soc_aid_l1_log EU_accessionactive:civ_soc_aid_l1_log
## 1          0.5162972                                0.0000000
## 2          0.5162972                                0.5162972
## 3          0.5162972                                0.0000000
## 4          0.5380981                                0.0000000
## 5          0.5380981                                0.5380981
## 6          0.5380981                                0.0000000
##   EU_accessionmember:civ_soc_aid_l1_log
## 1                             0.0000000
## 2                             0.0000000
## 3                             0.5162972
## 4                             0.0000000
## 5                             0.0000000
## 6                             0.5380981
```

Now we have to do XBeta to get our predicted values! But instead of beta being a k x 1 vector, it is now a k x 500 (number of samples from our multivariate normal) matrix. So, what will the dimensions of this new object be?

```
cs_str_hat <- x_mat_sim %*% t(samp_beta)

dim(cs_str_hat)
```

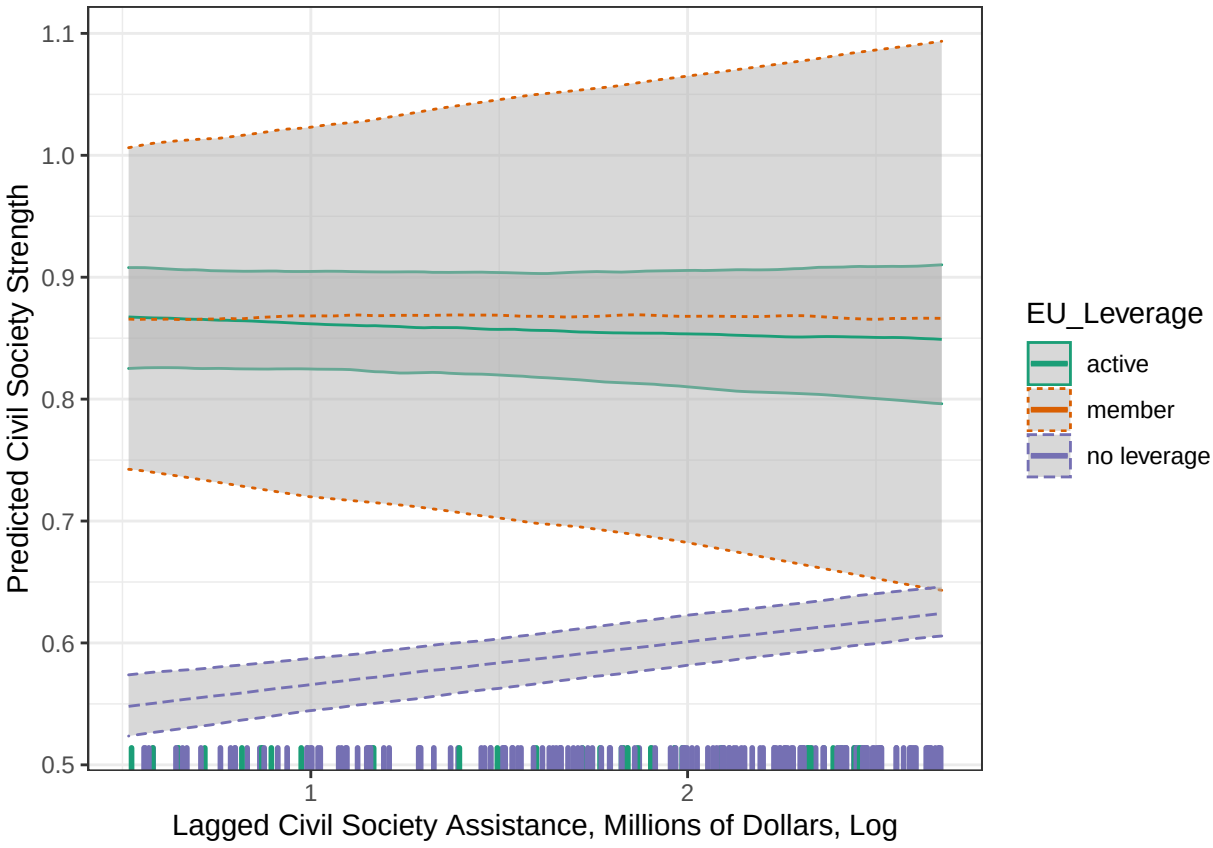
```
## [1] 300 500
```

```
#we only need quantiles for our confidence intervals
cs_str_hat_quant <- apply(cs_str_hat, 1, #why 1? again look at MARGIN
                      quantile, c(0.065, 0.5, 0.935))

#creating data frame of predictions
pred_df <- data.frame(PCS = cs_str_hat_quant[2, ], #why calling 2, 3, and 1?
                      UB = cs_str_hat_quant[3, ],
                      LB = cs_str_hat_quant[1, ],
                      cs_aid = x_mat_sim[, "civ_soc_aid_l1_log"],
                      EU_Leverage = rep(c("no leverage", "active", "member"),
                                        100)) #why can we do this?

#for rug plot
cs_aid_IQ <- dplyr::select(aid_data_95_ee, civ_soc_aid_l1_log, EU_accession) %>%
  filter(civ_soc_aid_l1_log > quantile(civ_soc_aid_l1_log, 0.25, na.rm = T) &
           civ_soc_aid_l1_log < quantile(civ_soc_aid_l1_log, 0.75, na.rm = T)) %>%
  rbind(data.frame(civ_soc_aid_l1_log = rep(NA, nrow(pred_df) - nrow(.)),
                   EU_accession = rep(NA, nrow(pred_df) - nrow(.))))
```

Now let's graph this!

```
#a library with different colors for ggplot2
library("RColorBrewer")


ggplot(data = pred_df, aes(x = cs_aid, y = PCS, lty = EU_Leverage)) +

  geom_ribbon(aes(ymin = LB, ymax = UB, color = EU_Leverage),
              alpha = 0.5,
              fill = "gray70") +
```

```
geom_line(aes(color = EU_Leverage)) +
#what do you think this does?

geom_rug(data = cs_aid_IQ, aes(x = civ_soc_aid_l1_log, color = EU_accession),
         size = 1, inherit.aes = F) +
#rug plot

scale_color_brewer(palette = "Dark2") +
#if you want to have some fun, change around the colors
#scale_color_brewer(palette = "Set3") +
#scale_color_brewer(palette = "Greys") +
#scale_color_brewer(palette = "PuOr") +
#scale_color_brewer(palette = "Set1") +

xlab("Lagged Civil Society Assistance, Millions of Dollars, Log") +
ylab("Predicted Civil Society Strength") +
#labels

theme_bw()
```




```
#theme
```

Regression Diagnostics

First we'll talk about (statistical) leverage! Points that are unusual (aka they are *outliers*) with respect to the *predictors*—that is, all of the $\mathbf{X}$—are said to have high *leverage* because they can impact our coefficients. One way to assess this is called "Cook's Distance," or a measure of an observation's influence.

We can write a function for Cook's Distance that computes a standard regression diagnostic and plots the results all in one. Useful if we want flexibility with different models and different observations!

$$D_i = \frac{\sum_{j=1}^{n}(y_{\hat{j(i)}} - \hat{y_j})^2}{k\hat{\sigma}^2}$$

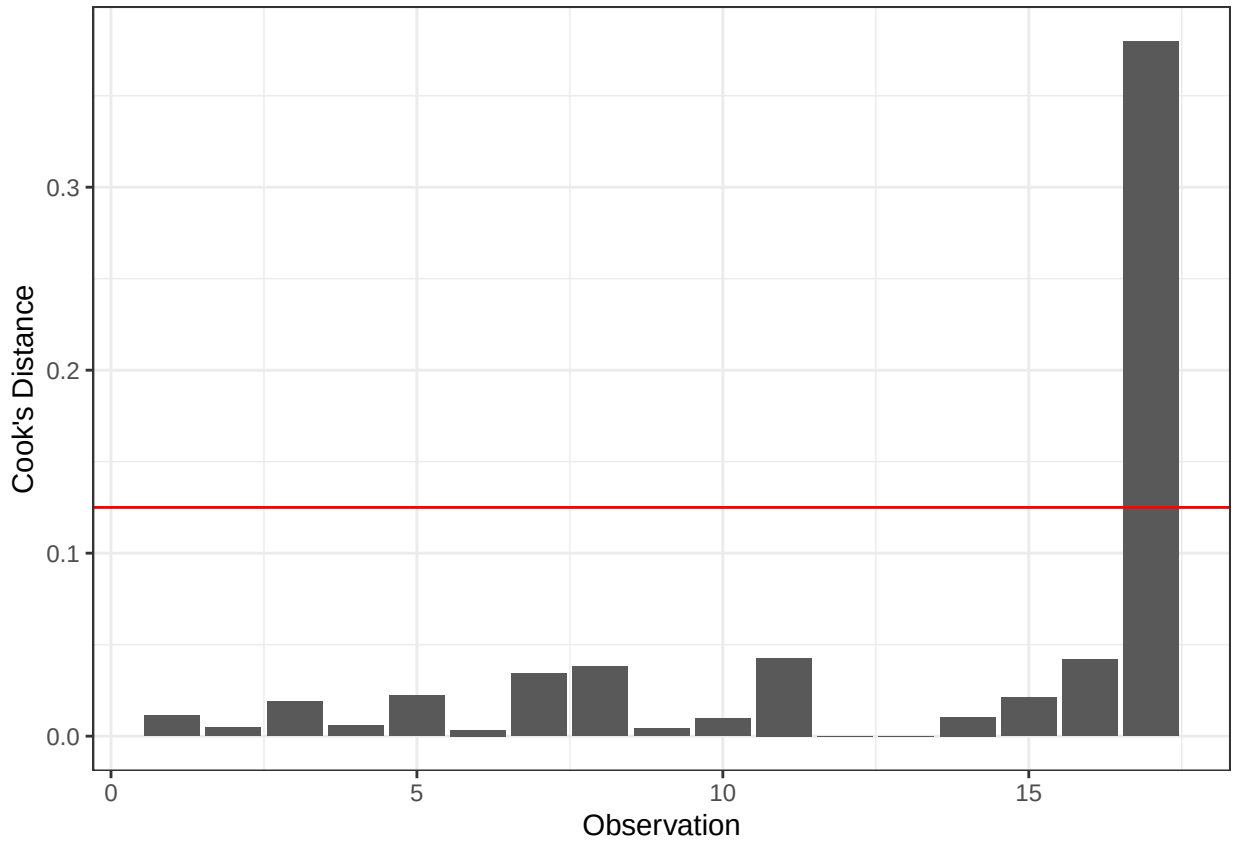
Where k is our number of coefficients, σ^2 is the variance of our model, and the numerator is an estimated mean of $\hat{y}$ when removing observation i.

When we have multiple regressors in our model (and with an eye to using R for computing Cook's D), it's useful to write this formula as such:

$$D_i = \frac{e_i^2}{k\hat{\sigma}^2}\frac{h_i}{(1-h_i)^2}$$

Where h_i is the i'th diagonal element of the hat matrix - and each observation's leverage.

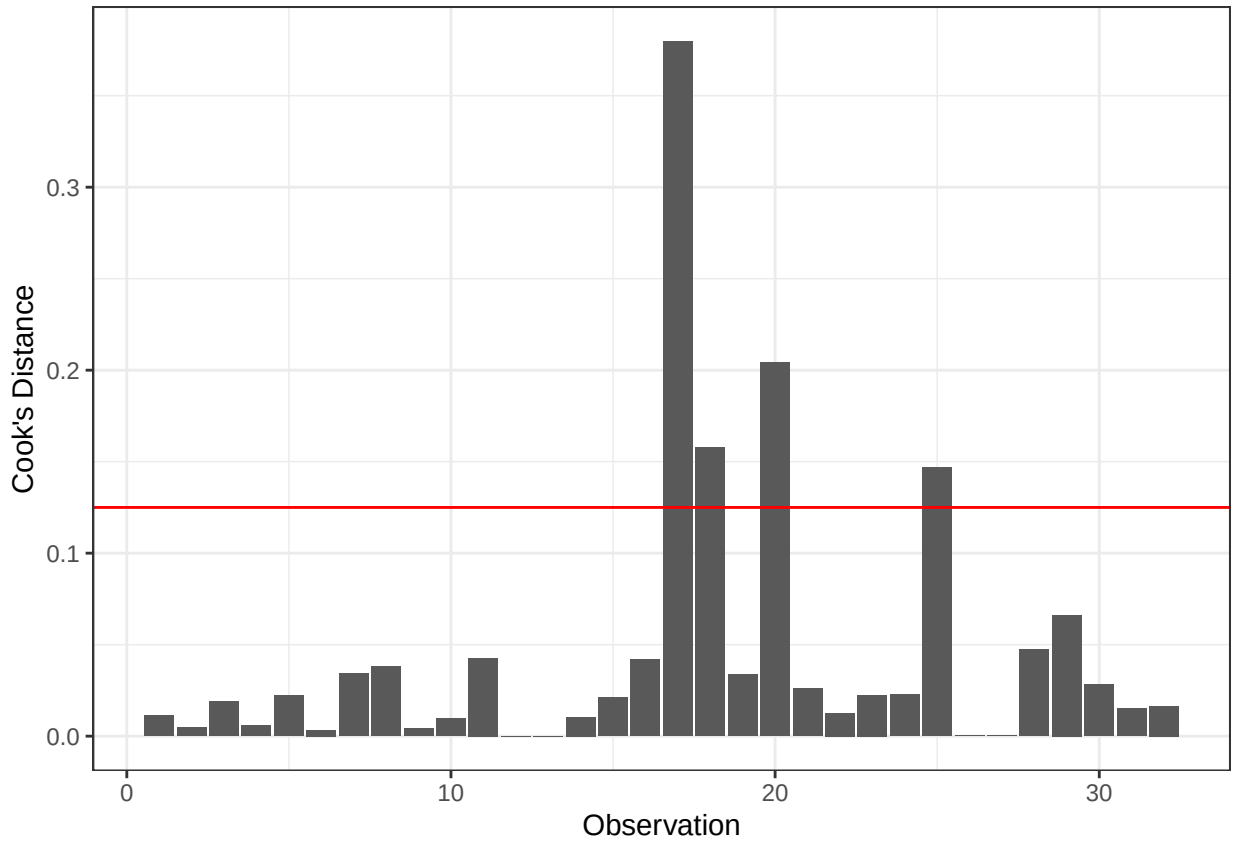
```
m1 <- lm(mpg ~ disp + wt, data = mtcars) #simple regression model

#Now, the function
cooks <- function(LM, x = "all") {
  #browser()
  #2 arguments
  if (is.character(x) && x == "all") x <- 1:nrow(model.matrix(LM))

  resid <- LM$residuals
  k <- length(LM$coefficients) #number of coefficients estimated
  num_1 <- resid^2 #squared residuals
  denom_1 <- mean(resid^2) * k #setting up the numerator and denominator for Cook's Dist
  num_2 <- hatvalues(LM)
  denom_2 <- (1 - hatvalues(LM))^2
  cooks_d <- ((num_1 / denom_1) * (num_2 / denom_2))[x] #Calculate Cook's D
  plot_dat <- data.frame(x = x,
                         cooks_d = cooks_d) #x allows for flex in the # of obsvs
  cooks_plot <- ggplot(plot_dat, aes(x = x, y = cooks_d)) +
  geom_bar(stat = "identity") +
  geom_hline(yintercept = (4 / nrow(model.matrix(LM))), colour = "red") +
  labs(x = "Observation", y = "Cook's Distance") +
  theme_bw() #Plot with a standard cutoff for concern
  return(list(plot = cooks_plot, data = plot_dat))
}

#Apply function for our model and 1st 20 observations
test <- cooks(m1, 1:17)
test$plot
```




```
test$data
```

```
##                      x      cooks_d
## Mazda RX4            1 1.127967e-02
## Mazda RX4 Wag        2 4.801561e-03
## Datsun 710           3 1.899854e-02
## Hornet 4 Drive       4 5.784270e-03
## Hornet Sportabout    5 2.237290e-02
## Valiant              6 3.408999e-03
## Duster 360           7 3.415049e-02
## Merc 240D            8 3.799309e-02
## Merc 230             9 4.165976e-03
## Merc 280            10 9.593086e-03
## Merc 280C           11 4.264584e-02
## Merc 450SE          12 4.882816e-06
## Merc 450SL          13 1.468001e-04
## Merc 450SLC         14 1.043682e-02
## Cadillac Fleetwood  15 2.119325e-02
## Lincoln Continental 16 4.187383e-02
## Chrysler Imperial   17 3.797096e-01
```

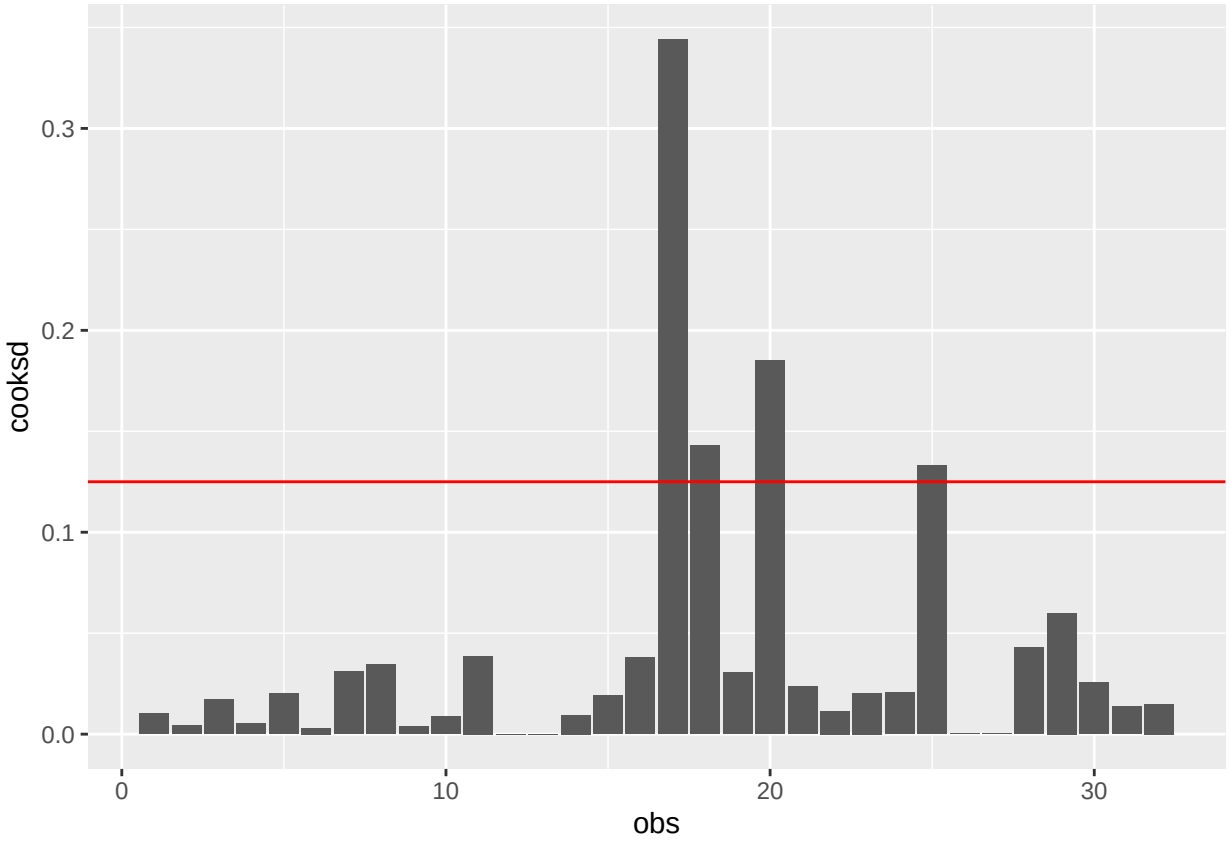
```
cooks(m1, "all")$plot
```



This is one rule of thumb with Cook's distance: looking at points above $4/n$, where n is our number of observations. Sometimes, you'll see this as $4/n - k$.

Although it is useful to be able to write our own Cook's Distance function, do note there are pre-existing "canned" functions. Your models won't always be compatible with these functions, but in simple cases they are useful.

```
#cooks.distance function from base R

test2 <- as.data.frame(cooks.distance(m1)) #make this into a df
test2$ident <- seq(1, nrow(test2)) #add observation index
colnames(test2) <- c("cooksd", "obs") #add meaningful column names


#plot
cooks_plot <- ggplot(test2, aes(y = cooksd, x = obs)) +
  geom_bar(stat= "identity") +
  geom_hline(yintercept = (4 / nrow(test2)), colour = "red")

cooks_plot
```

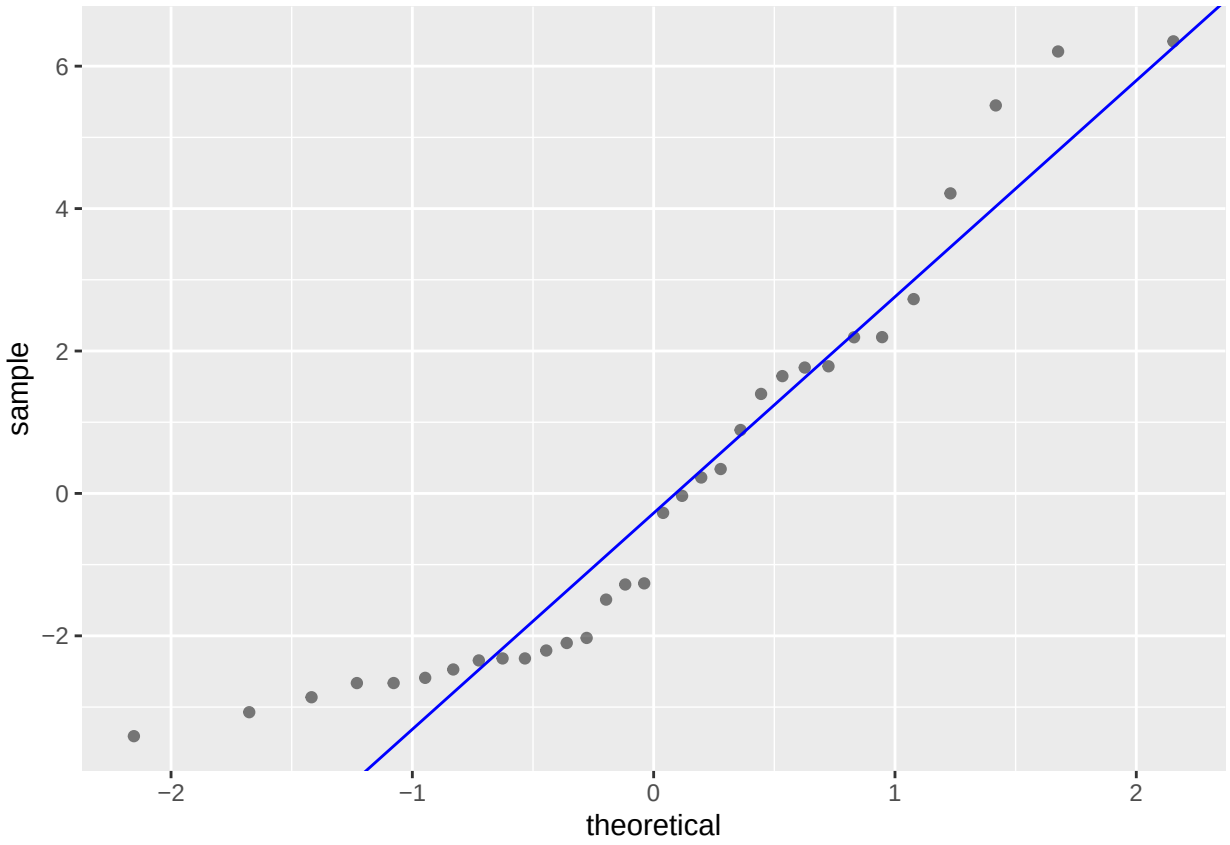


More Assumptions

Another assumption we may want to check is that our residuals (errors) are normally distributed (normality of errors assumption). We can do this by separating our residuals into quantiles and comparing their distribution to points produced randomly by a normal distribution (a q-q plot).

If both sets of quantiles came from the same distribution, we should see the points forming a line that's roughly straight.

```
ggQQ <- function(LM) { # argument: a linear model
     y <- quantile(LM$resid[!is.na(LM$resid)], c(0.25, 0.75))
     x <- qnorm(c(0.25, 0.75))
     slope <- diff(y)/diff(x)
     int <- y[1] - slope * x[1]
     p <- ggplot(LM) +
     stat_qq(aes(sample=.resid), alpha = 0.5) +
     geom_abline(slope = slope, intercept = int, color="blue")
     return(p)
    }

ggQQ(m1)
```



As stated before, it's often useful to be able to write these functions ourselves. Moving forward (particularly with your own data!) you might be in a situation with missingness, multilevel data, or other data quirks, and 'canned' functions aren't always able to give us reliable regression diagnostics.

That being said, in addition to the Cook's Distance canned function shown before, there are some other useful functions to be aware of: like `tidy` and `augment` from the `broom` package.

1) `tidy` converts your model into a tidy tibble, and provides useful estimates about your model components. This includes the coefs and p-values for each term in the regression. It is similar to `summary`, but in a new format that is easier to work with, particularly if you want to do further analysis with your data.

2) `augment` adds columns to the original data that was modeled, providing information about predictions, residuals, and Cook's Distance! As you can see, it shows this information for each type of the original points in your regression, in this case the type of car. This function *augments* the original data with more information from the model.

```
library(broom)
```

```
## Warning: package 'broom' was built under R version 4.2.2
```

```
tidy(m1)
```

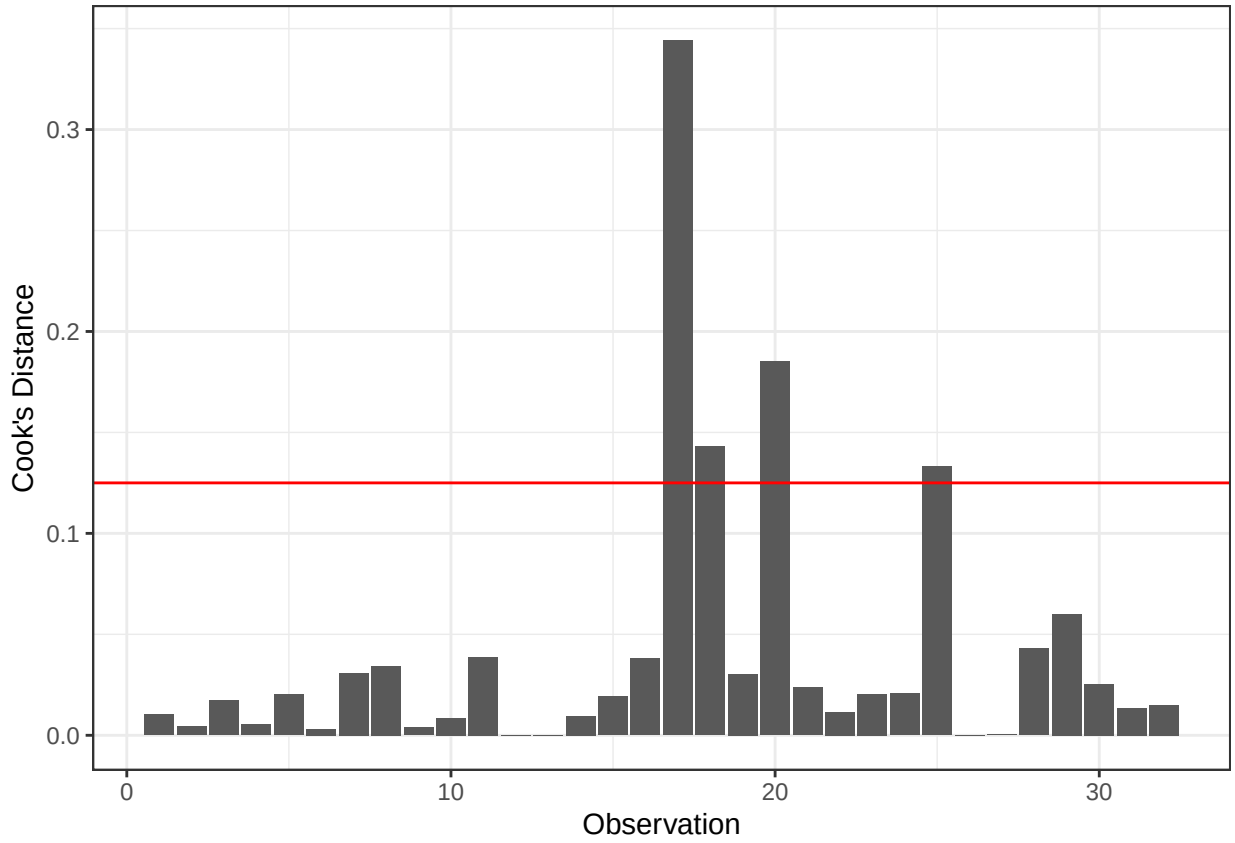
```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  35.0      2.16        16.2  4.91e-16
## 2 disp         -0.0177   0.00919     -1.93 6.36e- 2
## 3 wt           -3.35     1.16        -2.88 7.43e- 3
```

```
head(augment(m1))
```

```
## # A tibble: 6 x 10
##   .rownames          mpg  disp    wt .fitted .resid   .hat .sigma .cooksd .std.~1
##   <chr>            <dbl> <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>   <dbl>
## 1 Mazda RX4         21     160  2.62    23.3  -2.35 0.0434   2.93 0.0102   -0.822
## 2 Mazda RX4 Wag     21     160  2.88    22.5  -1.49 0.0455   2.95 0.00435  -0.523
## 3 Datsun 710        22.8   108  2.32    25.3  -2.47 0.0631   2.93 0.0172   -0.876
## 4 Hornet 4 Drive    21.4   258  3.22    19.6   1.79 0.0388   2.95 0.00524   0.624
## 5 Hornet Sportab~   18.7   360  3.44    17.1   1.65 0.141    2.95 0.0203    0.609
## 6 Valiant           18.1   225  3.46    19.4  -1.28 0.0441   2.96 0.00309  -0.448
## # ... with abbreviated variable name 1: .std.resid
```

Neat, huh? So, this means that for our plot above (Cook's distance) we could also do the following:

```
vals <- augment(m1)

ggplot(data=vals, aes(seq_along(.cooksd),y=.cooksd))+
    geom_bar(stat="identity")+
    geom_hline(yintercept=4/length(vals$.cooksd), color="red")+
    labs(x="Observation", y="Cook's Distance") +
    theme_bw()
```

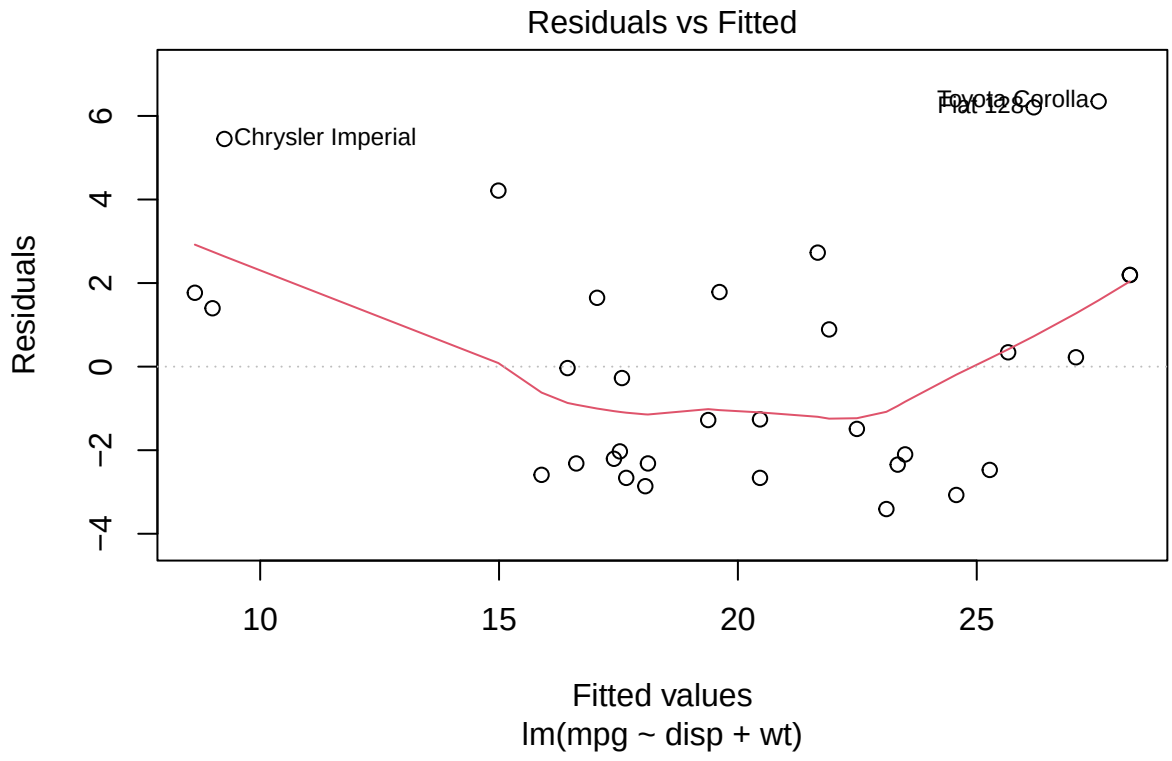


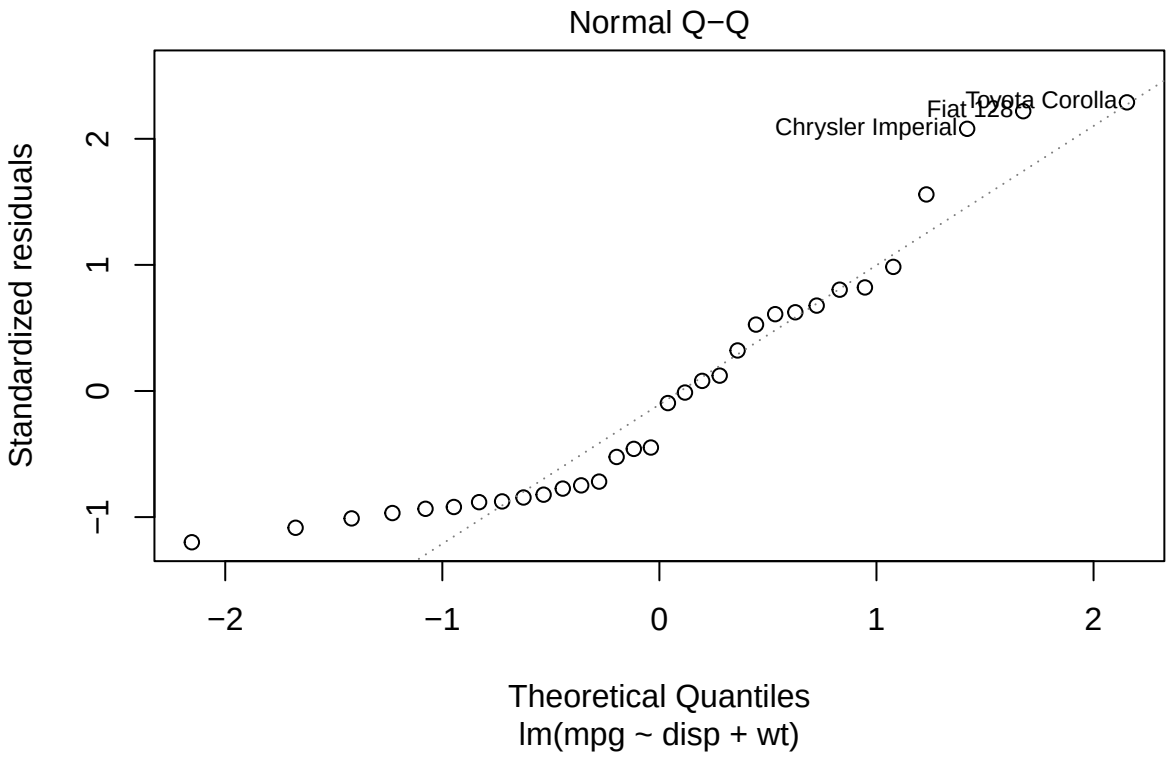
Other 'canned' functions that can be useful are found in the `stats` package and the `car` package (although most of these values can be gotten just by using `augment`):

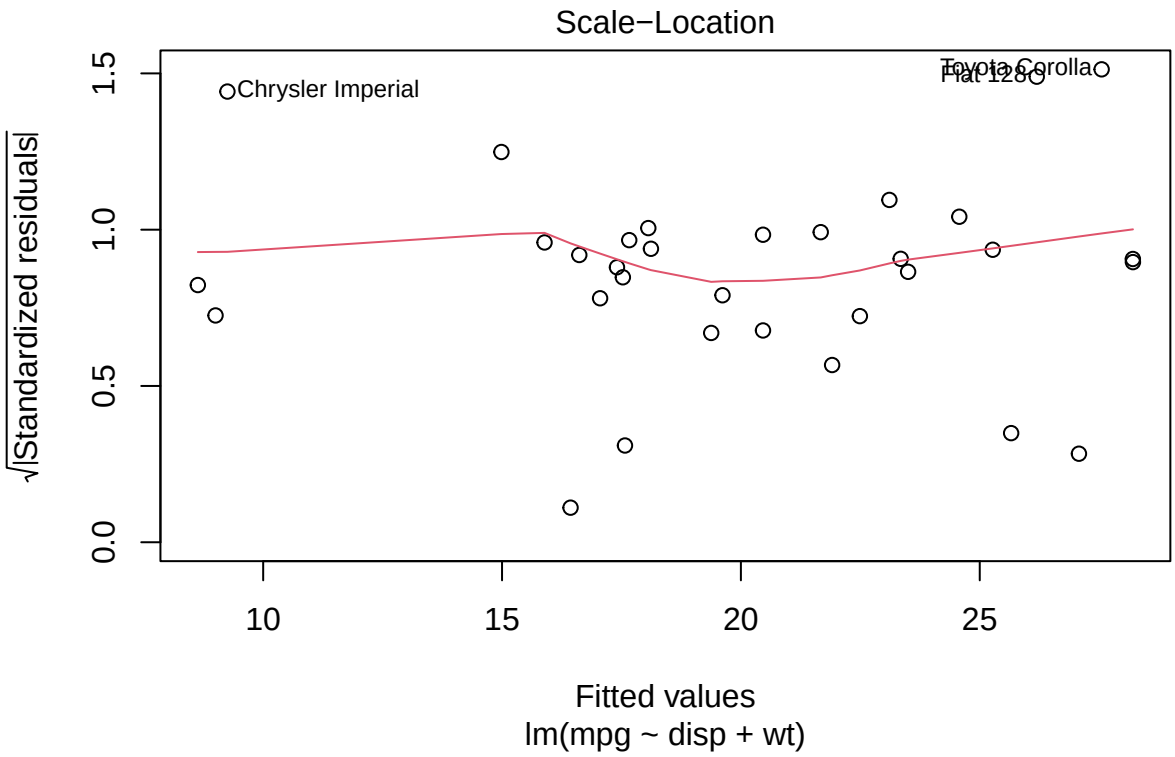
```
library(stats)

#Another cook's distance
cooks.distance(m1)

#dfbetas
dfbetas(m1)

#or, to get them all:
influence.measures(m1)
#note that this "marks" influential observations for you
#with an asterisk

library(car)
#variance inflation factors
#quantifies the severity of multicollinearity
vif(m1)
sqrt(vif(m1)) > 2 #look into this, is this a problem?
```

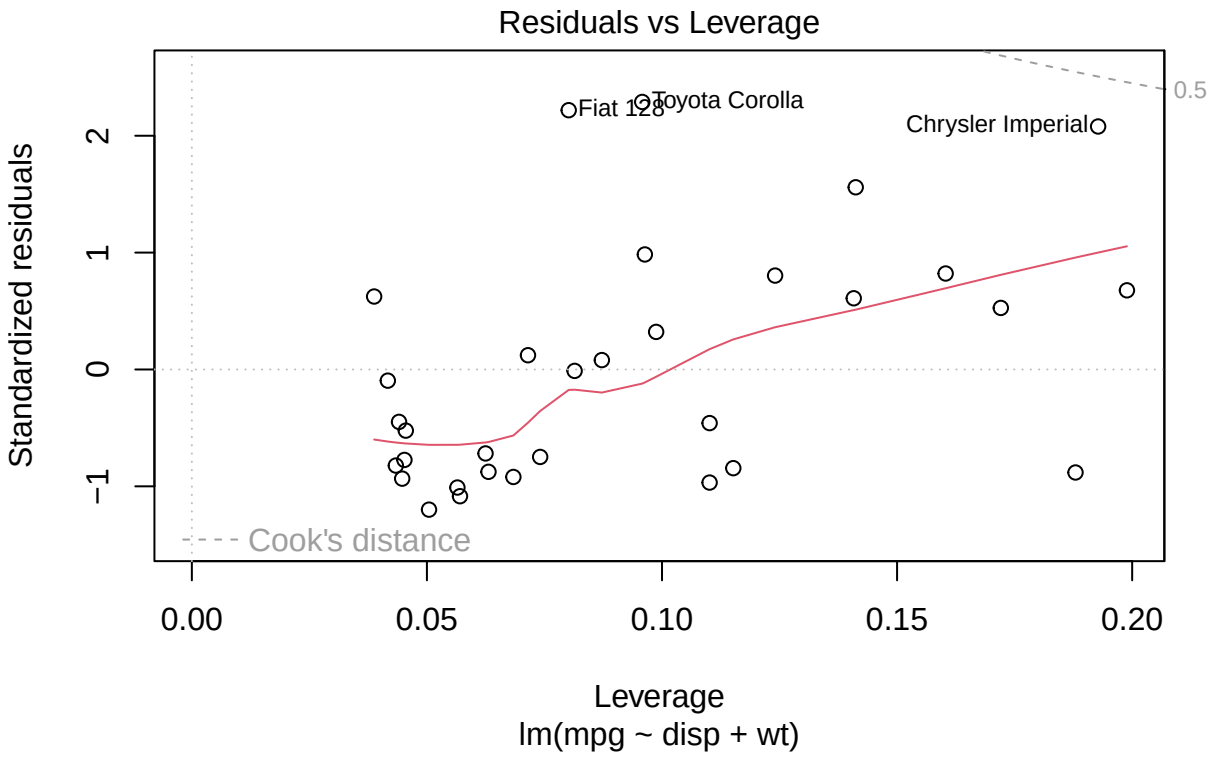
Other useful diagnostics include fitted value vs. residual plots and component + residual plots. The `plot` function allows you to get 4 standard diagnostic plots easily.

```
plot(m1)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(mpg ~ disp + wt)
Toyota Corolla
Fiat 128
Chrysler Imperial
−2
−1
0
1
2


Scale−Location
Chrysler Imperial
Toyota Corolla
Fiat 128
√|Standardized residuals|
0.0
0.5
1.0
1.5
10
15
20
25
Fitted values
lm(mpg ~ disp + wt)



1) The Residuals vs Fitted values plot can help us with two things: 1) it can help us see if the constant variance assumption is appropriate (most often, if it isn't we would see a "funnel" in one direction or another), and 2) it can tell us whether we might need to transform the outcome variable in some way (if the line of best fit isn't reasonably straight.) In an ideal scenario, the red line should show no fitted pattern and be approximately horizontal at zero. Is this the case here?

2) We discussed the Q-Q plot above! It helps us visually check the normality assumption. How does this one look?

3) The Scale-Location plot is basically the first plot folded in half and compressed. It doesn't tell us much more, to be quite honest, but it does make it more clear if our residuals are spread equally along the range of our predictors. It is good to see a horizontal line with all points spread out equally. Is this the case here?

4) The Residuals vs Leverage plot, as the name suggests, plots the standardized residuals of each observation against its leverage (remember only if both are high do we worry about how it affects our model). You can't really see it here, but R plots .5 and 1 Cook's Distance "contours" on this plot as well, which indicates where Cook's Distances of .5 and 1 would be. Some people say a Cook's Distance of > 1 is concerning, but some people say that is extremely conservative. In addition, you'll see in this plot it identifies the top 3 most extreme points - Toyota Corolla, Fiat, and Chrysler Imperial - which have standardized residuals > 2. As a rule of thumb, some say that observations with a standardized residual > 3 are concerning and possible outliers - we don't see any here. Good news!

Handily, all of these plots point out observations of which you may want to be wary.

And to wrap up, a couple other tests which will be useful for you to know!

1) The Durbin Watson test to see if we have autocorrelated errors (violating independence of errors). The DW statistic will always be between 0 and 4, with a value of 2 signifying that there is NO autocorrelation

detected in your sample. Values < 2 means there is positive autocorrelation, values > 2 indicate there is negative autocorrelation.

2) Shapiro-Wilk's normality test. This test, generally speaking, examines normality. We can apply it to our residual terms to check for normality of errors. If the p value is > 0.05 the distribution of the data is NOT significantly different from the normal distribution (the normality assumption holds).

```
#durbin watson
durbinWatsonTest(m1) #is this a problem?
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1        0.341622      1.276569   0.024
##  Alternative hypothesis: rho != 0
```

```
#shapiro test
shapiro.test(m1$residuals) #how does it look?
```

```
##
##  Shapiro-Wilk normality test
##
## data:  m1$residuals
## W = 0.89097, p-value = 0.003677
```

To do on your own:

Take the `Guns` dataset in the `AER` package. Take a look at the dataset and use `?Guns` to get the codebook. Regress the robbery rate on the percent of the state population that is male, the real per capita personal income in the state, and then an interaction between the population density and whether a shall carry law was in effect that year.

Use Monte Carlo simulation with 10,000 samples from the sampling distribution to get 80% confidence intervals around the coefficient estimates.

Then, use simulation to plot the predicted robbery rate as the population density varies from its 10th percentile to its 90th percentile dependent on if a shall carry law is in effect (don't worry about the rug plot).

```
library(AER)
data(Guns)

#estimate model
guns_model <- lm(robbery ~ male + income + density*law,
                 data = Guns)

set.seed(53)

#sampling betas
samp_beta_guns <- mvrnorm(10000,
                          coef(guns_model),
                          vcov(guns_model))

#confidence intervals
t(apply(samp_beta_guns, 2, quantile, prob = c(.1, .9)))
```

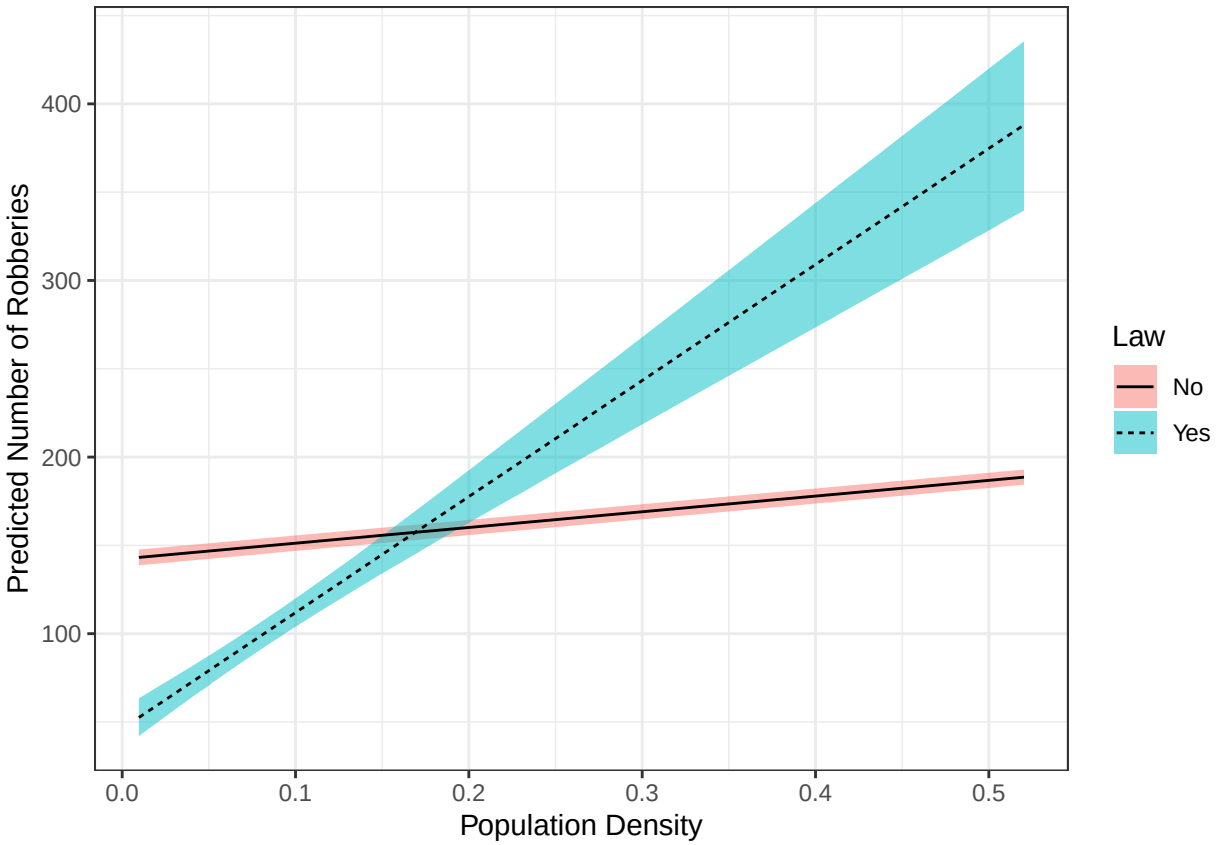
```
##                          10%          90%
## (Intercept)    -1.317127e+02  -9.72173240
## male            6.266225e-01   5.93038083
## income          9.824894e-03   0.01352996
```

```
## density         8.600604e+01  91.91279878
## lawyes         -1.082471e+02 -83.60493956
## density:lawyes  4.590982e+02 673.56117863
```

```
confint(guns_model, level = .8)
```

```
##                         10 %         90 %
## (Intercept)    -1.317624e+02  -8.86131129
## male            5.944492e-01   5.97040494
## income          9.802649e-03   0.01350774
## density         8.600847e+01  91.89936094
## lawyes         -1.085605e+02 -83.40076614
## density:lawyes  4.586914e+02 674.37298743
```

```
#getting range of density
density_sim <- with(Guns,
                    seq(quantile(density, .1), quantile(density, .9),
                        length.out = 100))

#hypothetical predictor matrix
 X_sim_guns <- with(Guns,
                   model.matrix(~ male + income + density*law,
                      data = data.frame(male = mean(male),
                                        income = mean(income),
                                        law = c(0, 1),
                                        density = rep(density_sim, each = 2))
                   ))

robbery_hat <- X_sim_guns %*% t(samp_beta_guns)

#we only need quantiles for our confidence intervals
robbery_hat_quant <- apply(robbery_hat, 1, #why 1?
                      quantile, c(0.1, 0.5, 0.9))

pred_df_guns <- data.frame(PP = robbery_hat_quant[2, ],
                      UB = robbery_hat_quant[3, ],
                      LB = robbery_hat_quant[1, ],
                      Density = X_sim_guns[, "density"],
                      Law = c("No", "Yes")[X_sim_guns[, "law"] + 1]) #why can we do this?  `

ggplot(data = pred_df_guns, aes(x = Density, y = PP, lty = Law)) +
  geom_ribbon(aes(ymin = LB, ymax = UB, fill = Law), alpha = 0.5) +
  geom_line() + #what do you think this does?
  xlab("Population Density") +
  ylab("Predicted Number of Robberies") +
  theme_bw()
```


Predicted Number of Robberies
400
300
200
100
0.0
0.1
0.2
0.3
0.4
0.5
Population Density
Law
No
Yes