Lab 5 - LM Diagnostics

TA: Isabel Laterzo

February 24, 2021

This week's lab, as always, has been adapted from code and text by Chelsea Estancona, Simon Hoellerbauer, and various online sources.

Today we will be examining **Regression Diagnostics**, or ways to examine if our regression meets necessary assumptions, and if anything weird (e.g., outliers, etc.) is going on. Remember the four key assumptions of a linear regression:

1) Linearity of the data - the relationship between x and y is linear
2) Normality of residuals - the residual errors are normally distributed
3) Homogeneity of residual variance - the residuals have CONSTANT variance (homoscedasticity)
4) Independence of residual error terms

The following tests will help us see if these assumptions hold, and will examine a number of possibe issues. These include (but are not limited to) non-linearity of the data, heteroscedasticity (non-constant variance in the errors), and presence of influential values (outliers and high-leverage points).

Let's begin!

Load necessary packages and data, clear your environment:

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

#ggplot, my old friend
library(ggplot2)

#cars! again!
data(mtcars)
```

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 **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!

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

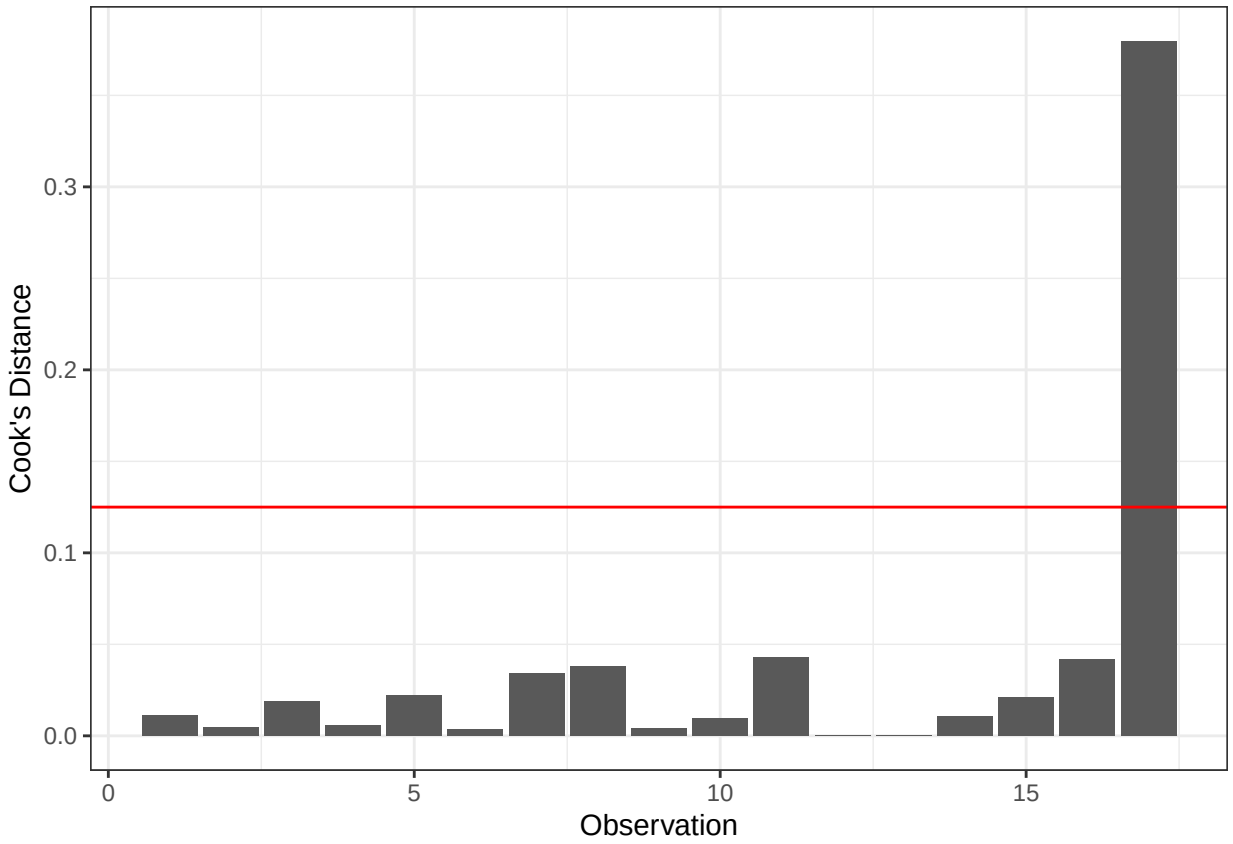
Where k is our number of coefficients, σ^2 is the variance of our model, and the numerator is an estimated mean of $\hat{y}$ when removing observation i.

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.

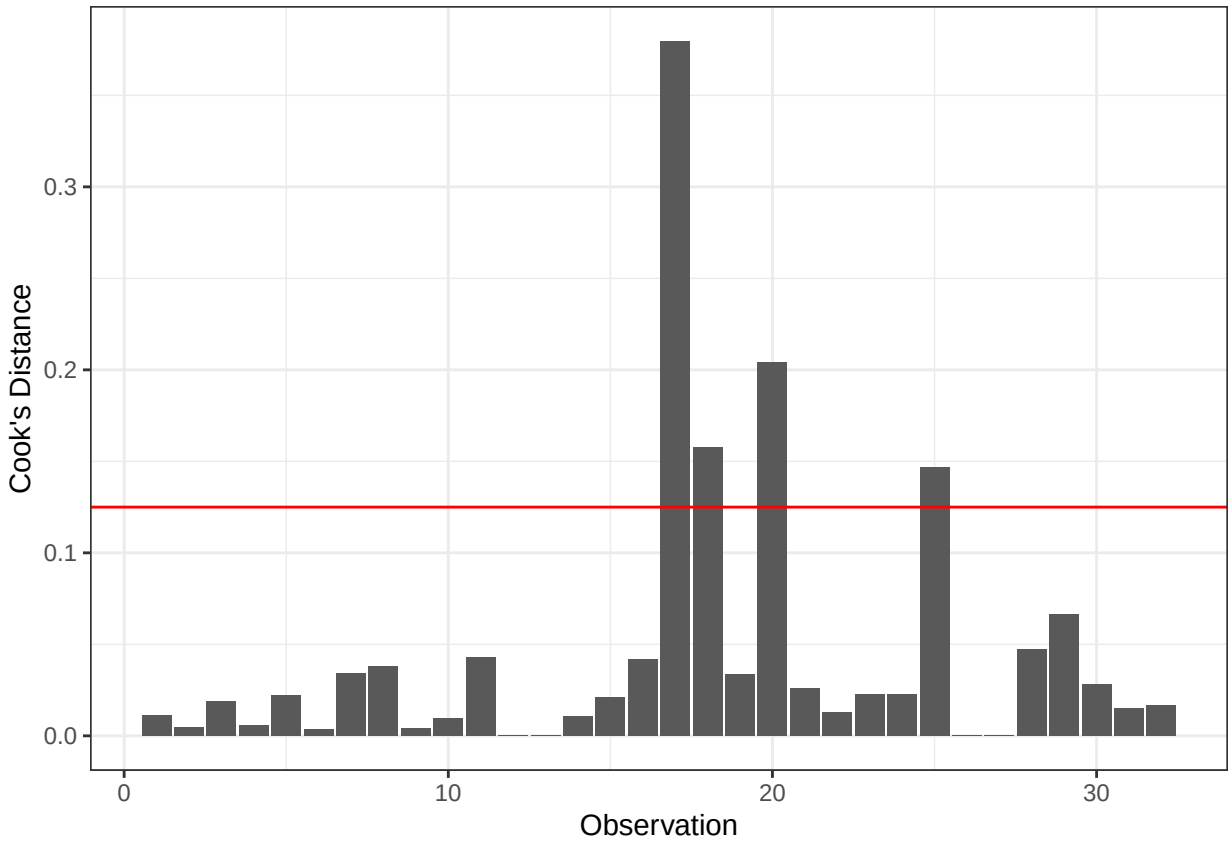
```
m1 <- lm(mpg ~ disp + wt, data = mtcars) #simple regression model

#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
```

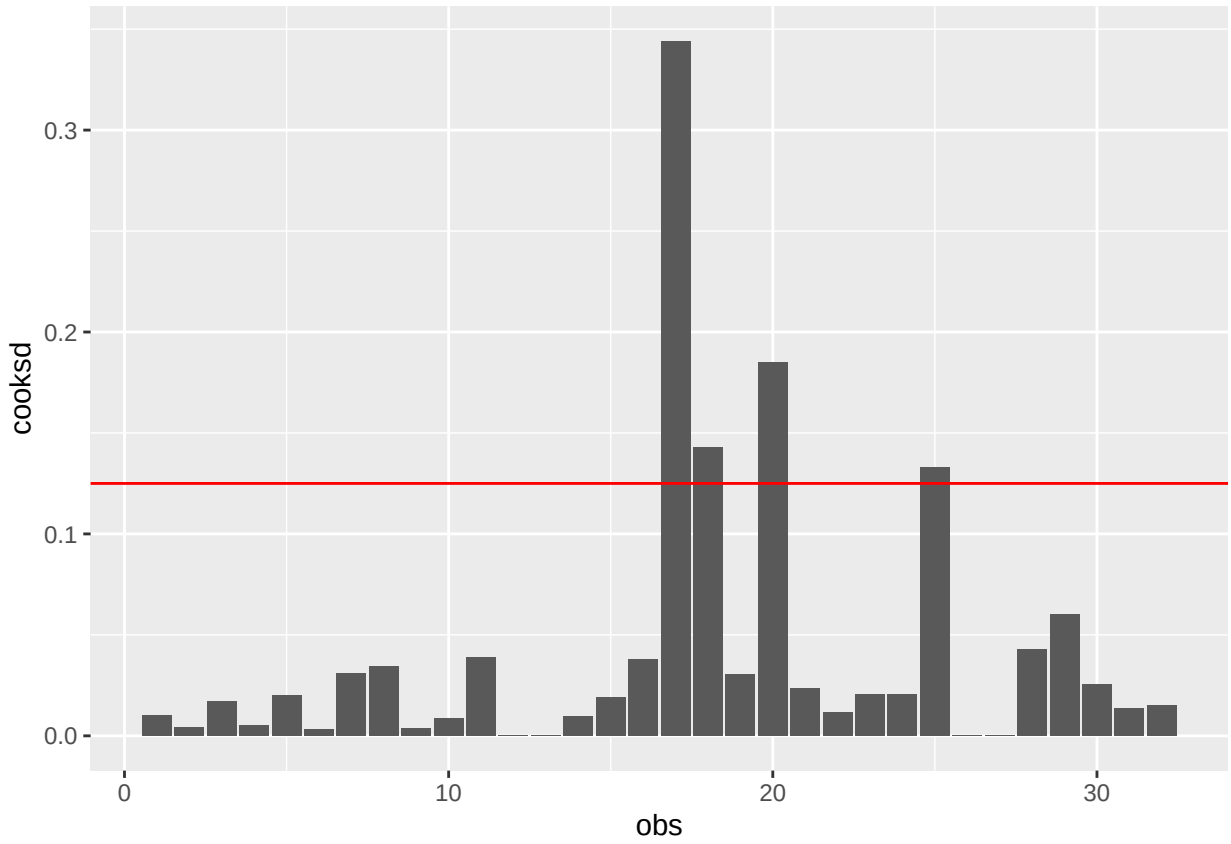
```
cooks(m1, "all")$plot
```



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
```

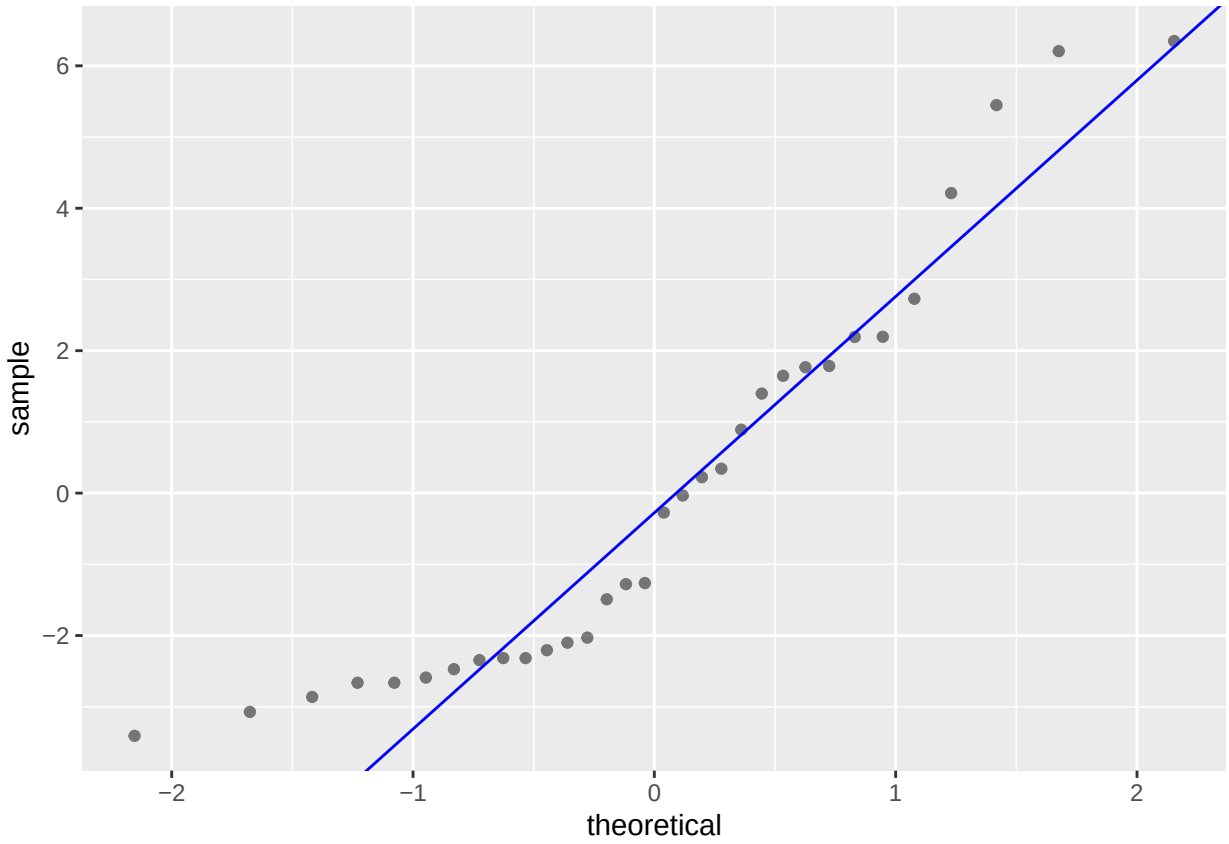


More Assumptions

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
      #browser()
     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)
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
```

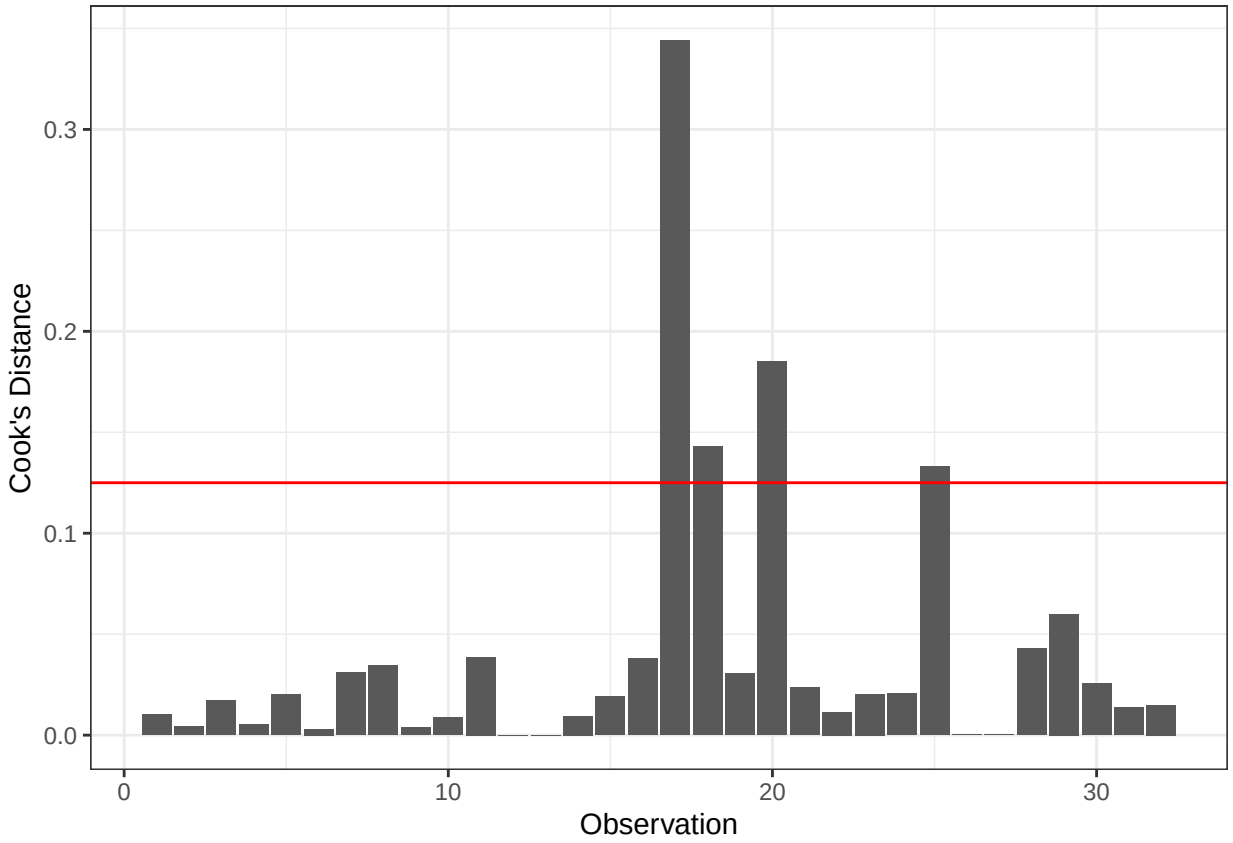
```
head(augment(m1))
```

```
## # A tibble: 6 x 10
##   .rownames      mpg  disp    wt .fitted .resid .std.resid   .hat .sigma .cooksd
```

```
##   <chr>        <dbl> <dbl> <dbl>   <dbl>  <dbl>      <dbl>  <dbl>  <dbl>   <dbl>
## 1 Mazda RX4     21     160  2.62    23.3  -2.35     -0.822 0.0434   2.93 0.0102
## 2 Mazda RX4 W~  21     160  2.88    22.5  -1.49     -0.523 0.0455   2.95 0.00435
## 3 Datsun 710    22.8   108  2.32    25.3  -2.47     -0.876 0.0631   2.93 0.0172
## 4 Hornet 4 Dr~  21.4   258  3.22    19.6   1.79      0.624 0.0388   2.95 0.00524
## 5 Hornet Spor~  18.7   360  3.44    17.1   1.65      0.609 0.141    2.95 0.0203
## 6 Valiant       18.1   225  3.46    19.4  -1.28     -0.448 0.0441   2.96 0.00309
```

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()
```



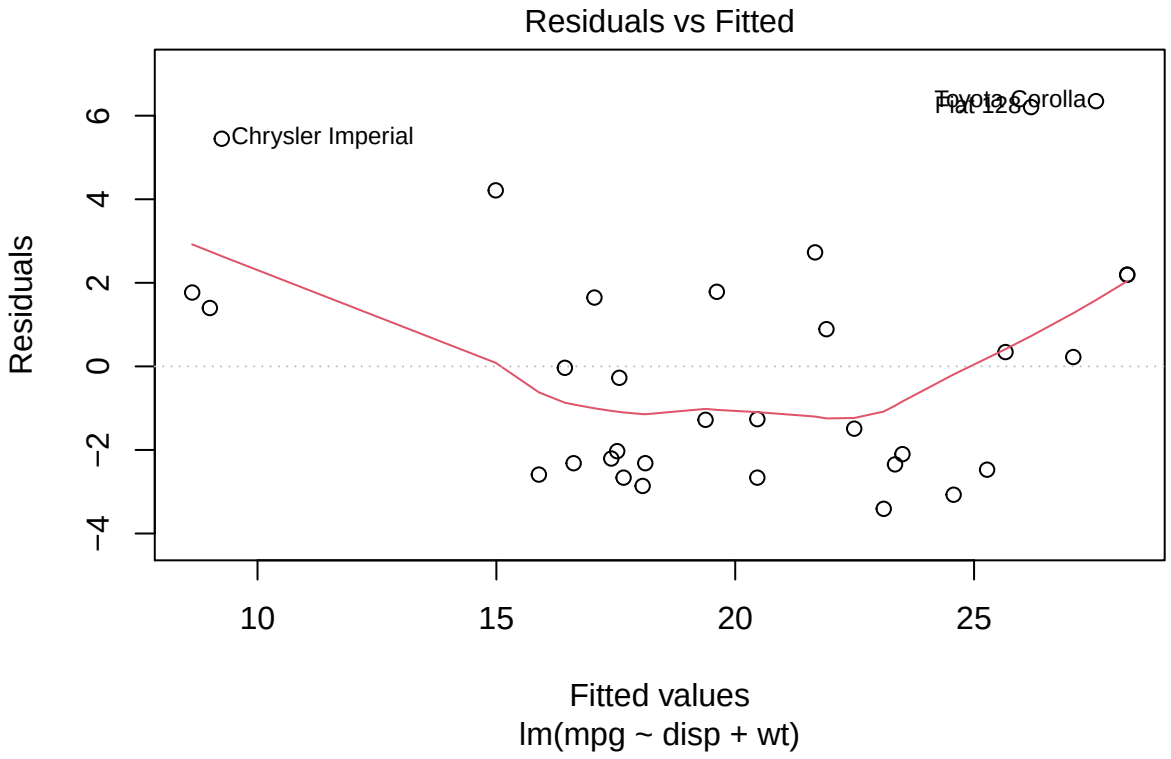
Other 'canned' functions that can be useful are found in the `stats` package and the `car` package (although most of these values can be gotten just by using `augment`):

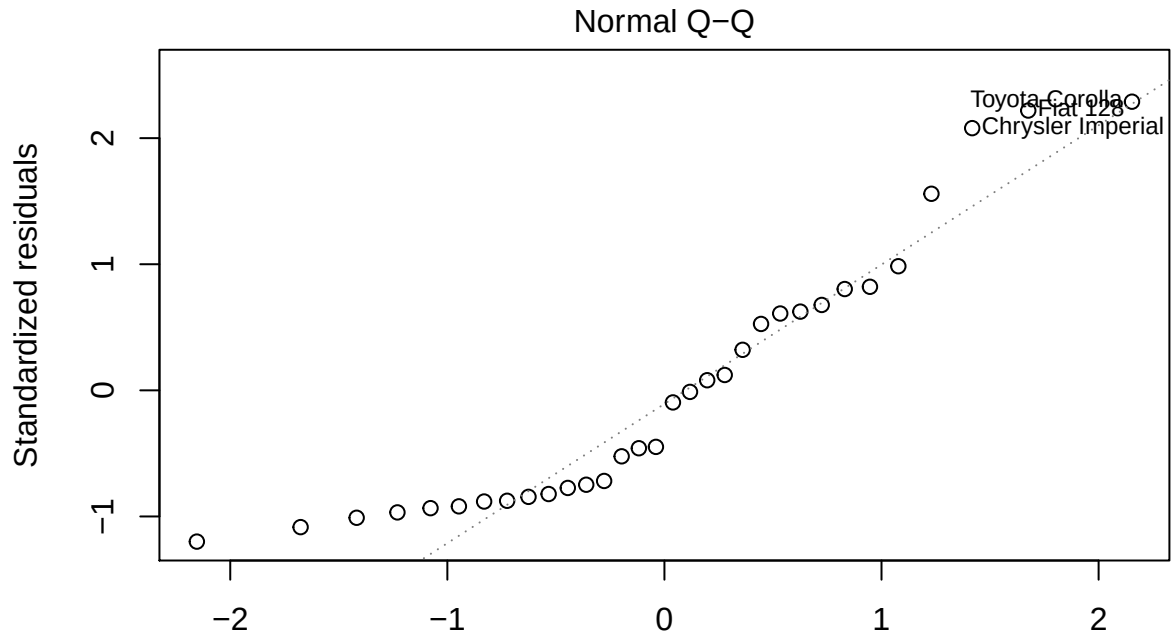
```
library(stats)

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

#dfbetas
dfbetas(m1)
```

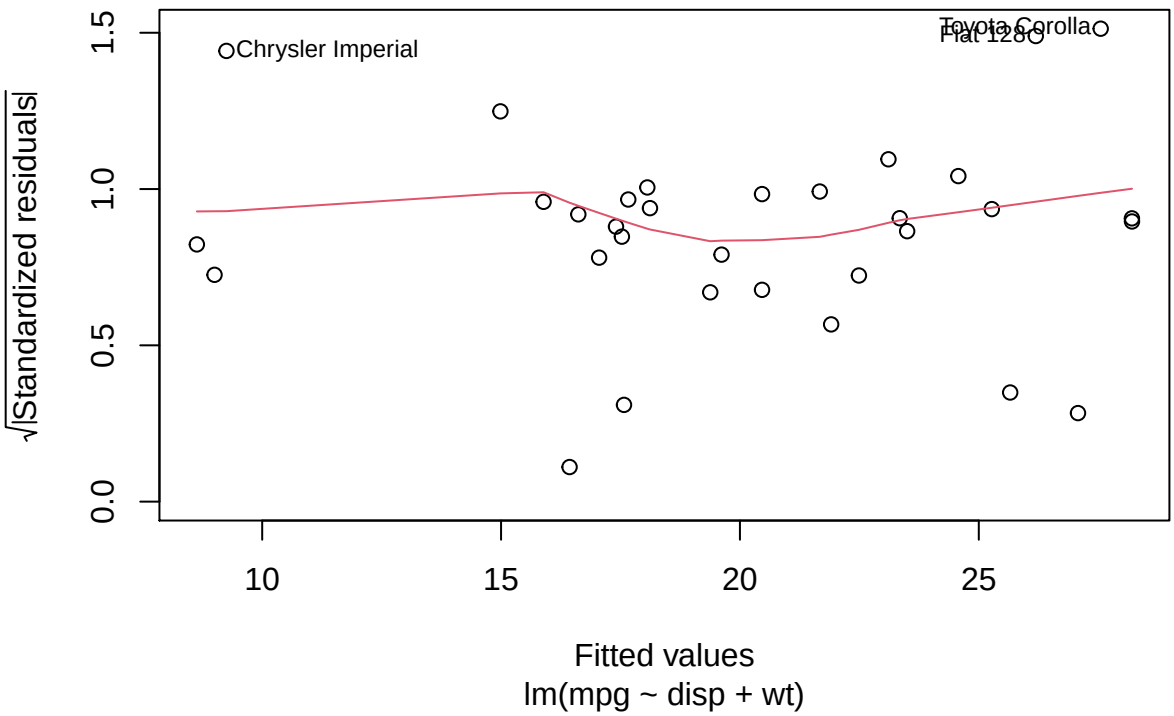
```
#or, to get them all:
influence.measures(m1)
#note that this "marks" influential observations for you
#with an asterisk

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

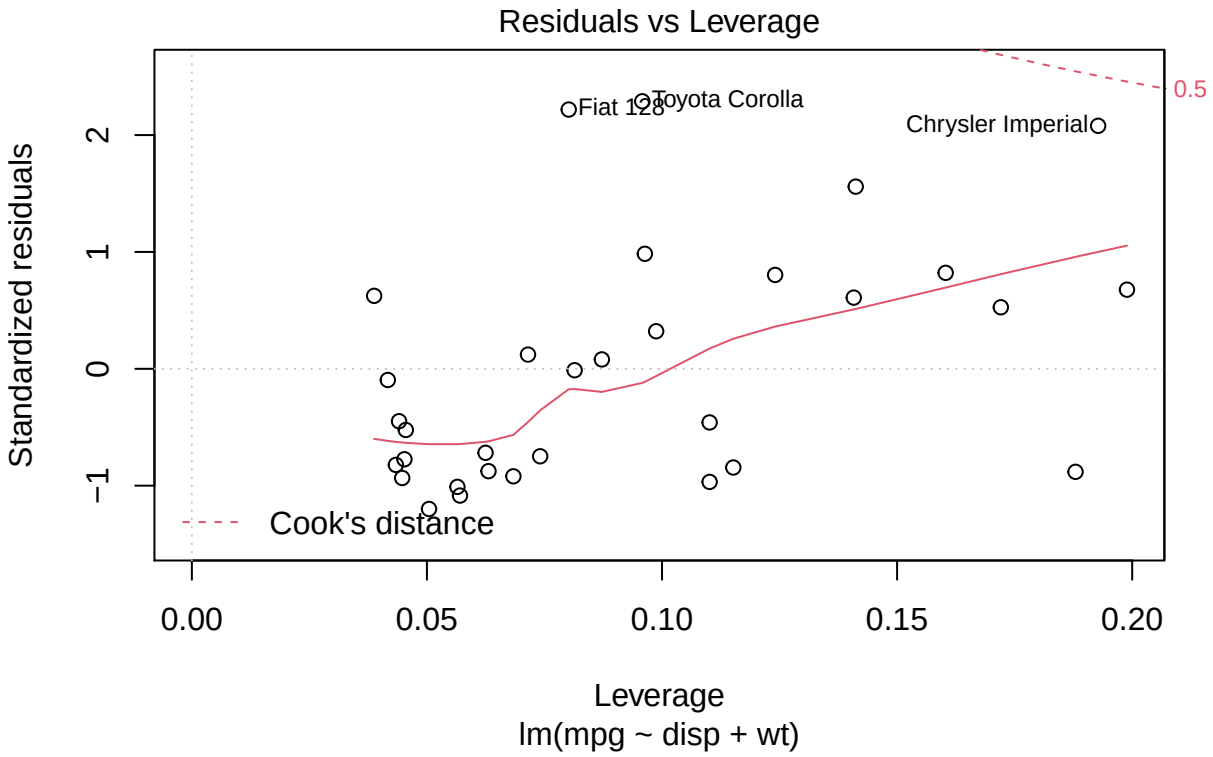
Other useful diagnostics include fitted value vs. residual plots and component + residual plots. The `plot` function allows you to get 4 standard diagnostic plots easily.

```
plot(m1)
```




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


Scale−Location
√|Standardized residuals|
Chrysler Imperial
Toyota Corolla
Fiat 128
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    0.01
##  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:

Consider the data for your upcoming deliverable (your reproduction project plan). Get these data into R, estimate any relevant linear model (doesn't have to be the model used in the paper!) and then assess the influence of observations and the normality assumption. Provide plots.