STAT350 Tutorial 4

22/09/2020

In this weeks tutorial we'll be going over some issues that can arise when performing multiple linear regression, followed by assessment of the model.

We're going to be using the same dataset as last week, but I'll add some complexity to the model by including disp as a predictor. I've fit the model and printed out the model summary below.

```
mdl <- lm(mpg ~ wt + hp + disp, data = auto_mpg)
(mdl_sum <- summary(mdl))
```

```
##
## Call:
## lm(formula = mpg ~ wt + hp + disp, data = auto_mpg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.3347  -2.8028  -0.3402   2.2037  16.2409
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 44.8559357  1.1959200  37.507  < 2e-16 ***
## wt          -0.0053516  0.0007124  -7.513 4.04e-13 ***
## hp          -0.0416741  0.0128139  -3.252  0.00125 **
## disp        -0.0057688  0.0065819  -0.876  0.38132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.241 on 388 degrees of freedom
## Multiple R-squared:  0.707,  Adjusted R-squared:  0.7047
## F-statistic:   312 on 3 and 388 DF,  p-value: < 2.2e-16
```

Hidden Extrapolation

I talked about this briefly in last weeks tutorial. This week we'll compute a metric known as the regressor variable hull (RVH) which will allow us to check whether or not a new data point of interest lies within the initial design.

First we compute the hat matrix and save the largest diagonal element as h_max.

```
X <- cbind(rep(1, nrow(auto_mpg)), auto_mpg$wt, auto_mpg$hp, auto_mpg$disp)
H <-  X %*% solve(t(X) %*% X) %*% t(X)
(h_max <- max(diag(H)))
```

```
## [1] 0.1386801
```

Now we will define a new point x_0 and check it's location relative to the RVH.

$$\boldsymbol{x}_0'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}_0 \leq h_max$$

```
x_0 <- data.frame(wt = 2500, hp = 150, disp = 300)
x_0 <- as.matrix(cbind(1, x_0), nrow = 1)
(h_00 <- x_0 %*% solve(t(X) %*% X) %*% t(x_0))
```

```
##             [,1]
## [1,] 0.05459918
```

Since $0.0545992 < 0.1386801$, we conclude that the new data point would fall under interpolation and is safe to make a prediction at given our current model.

Standardized Regression Coefficients

The purpose of standardizing the regression coefficients is to allow us to more directly compare the effects of each individual predictor on the response. To do do this we will scale the data. I will be doing unit normal scaling, but the steps used here will be the same for every type of scaling.

Unit normal scalling essentially turns all variables in the data to a zero-mean normal variable with unit variance

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}$$

A useful function that we could is `apply()`. It allows to apply a function, `FUN` to a dataframe or matrix for each of its rows or columns. If we want to apply the function across columns we set the second argument equal to 2, otherwise setting equal to 1 will apply the function to the rows. This allow us to compute the means and standard deviations all variables in one line.

```
auto_means <- apply(auto_mpg, 2, FUN = mean)
auto_sds <- apply(auto_mpg, 2, FUN = sd)
```

We could then use this to scale the data manually, but there's another useful function `scale()` which will do this type of scaling for us. To ensure that the output is in the right format I wrap this function in `as.data.frame()`.

```
auto_scaled <- as.data.frame(scale(auto_mpg, center = TRUE, scale = TRUE))
```

As a check, we can make sure the means of each column in the data are zero:

```
apply(auto_scaled, 2, FUN = mean)
```

```
##           mpg          disp            hp            wt           acc
##  1.569283e-16 -5.786680e-17 -1.767977e-16 -8.020330e-18 -2.319512e-16
```

and that their standard deviations are one:

```
apply(auto_scaled, 2, FUN = sd)
```

```
##  mpg disp   hp   wt  acc
##    1    1    1    1    1
```

So, we have our scaled data and can now fit a new linear regression model on this data, remembering to remove the intercept from the model.

```
mdl_scaled <- lm(mpg ~ -1 + wt + hp + disp, data = auto_scaled)
(mdl_scaled_sum <- summary(mdl_scaled))
```

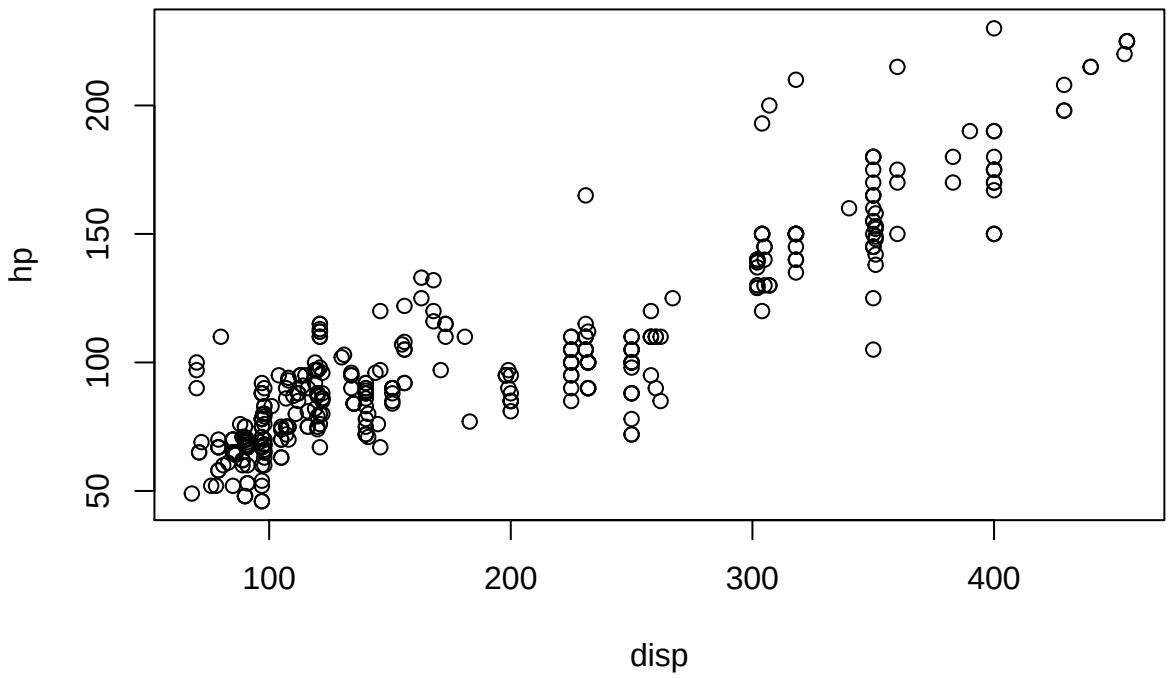
```
##
## Call:
## lm(formula = mpg ~ -1 + wt + hp + disp, data = auto_scaled)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45223 -0.35910 -0.04359  0.28234  2.08083
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## wt   -0.58240    0.07742  -7.522 3.77e-13 ***
## hp   -0.20552    0.06311  -3.256  0.00123 **
## disp -0.07734    0.08813  -0.878  0.38070
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5427 on 389 degrees of freedom
## Multiple R-squared:  0.707,  Adjusted R-squared:  0.7047
## F-statistic: 312.8 on 3 and 389 DF,  p-value: < 2.2e-16
```

With this we can more easily compare the effect of each predictor on the response, remembering that these effect are dependent on the other regressors being included in the model.

Multicollinearity

This was alluded to in last weeks tutorial - in the pairs plot we saw that was a near linear relationship between some of the regressor variables in the data, e.g. between hp and disp as seen below.



The presence of multicollinearity can affect our ability to estimate the coefficients in the model. Sometimes a visual check like above is enough, but a diagnostic is always nice to have - we'll use the variance inflation factor (VIF):

$$VIF_j = \frac{1}{1 - R_j^2}.$$

Here R_j^2 is the coefficient of determination for a linear regression model fit with $\boldsymbol{x}_j$ as the response and the other regression variables as the predictors.

Below I calculate the VIF for each predictor in our initial model:

```
wt_mdl <- lm(wt ~ hp + disp, data = auto_mpg)
wt_sum <- summary(wt_mdl)
wt_vif <- 1/(1-wt_sum$r.squared)



hp_mdl <- lm(hp ~ wt + disp, data = auto_mpg)
hp_sum <- summary(hp_mdl)
hp_vif <- 1/(1-hp_sum$r.squared)
```

```r
disp_mdl <- lm(disp ~ wt + hp, data = auto_mpg)
disp_sum <- summary(disp_mdl)
disp_vif <- 1/(1-disp_sum$r.squared)

c(wt_vif, hp_vif, disp_vif)
```

```
## [1]  7.957383  5.287295 10.310539
```

Your textbook suggests VIF values larger than 10 are a serious issue. For our data we see that the VIF for disp is slightly larger than 10 and therefore a problem. Chapter 9 covers how to deal with multicollinearity in the data.

Model Assessment

In this section we'll will be assessing whether or not the modelling assumptions are met. Those assumptions are:

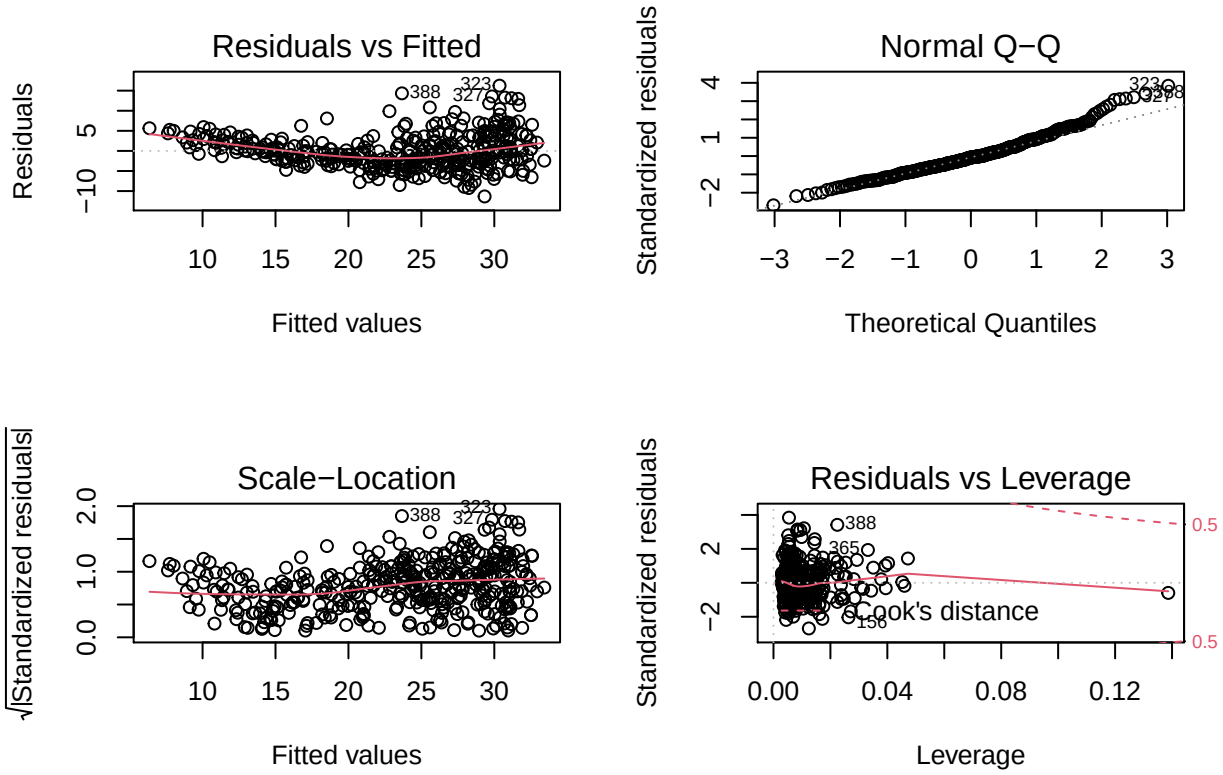
1. The response in linearly related to the predictors.
2. The errors are i.i.d. normally distributed, with zero mean and constant variance.

Residual Analysis

Today will be going over graphic analysis of the residuals to check the parts of assumption 2 above, as well as any unusual observations. With this type of analysis it is important not to read too much into what you see, e.g. in one of the following plots the location of one or two points should not change our perception, we want to point out only obvious deviations from the assumptions.

Remember: the residual is defined to be the difference between the true response value and the predicted value from the model, $e = y_i - \hat{y}_i$. In R we can get residual plots simply by plotting the model object.

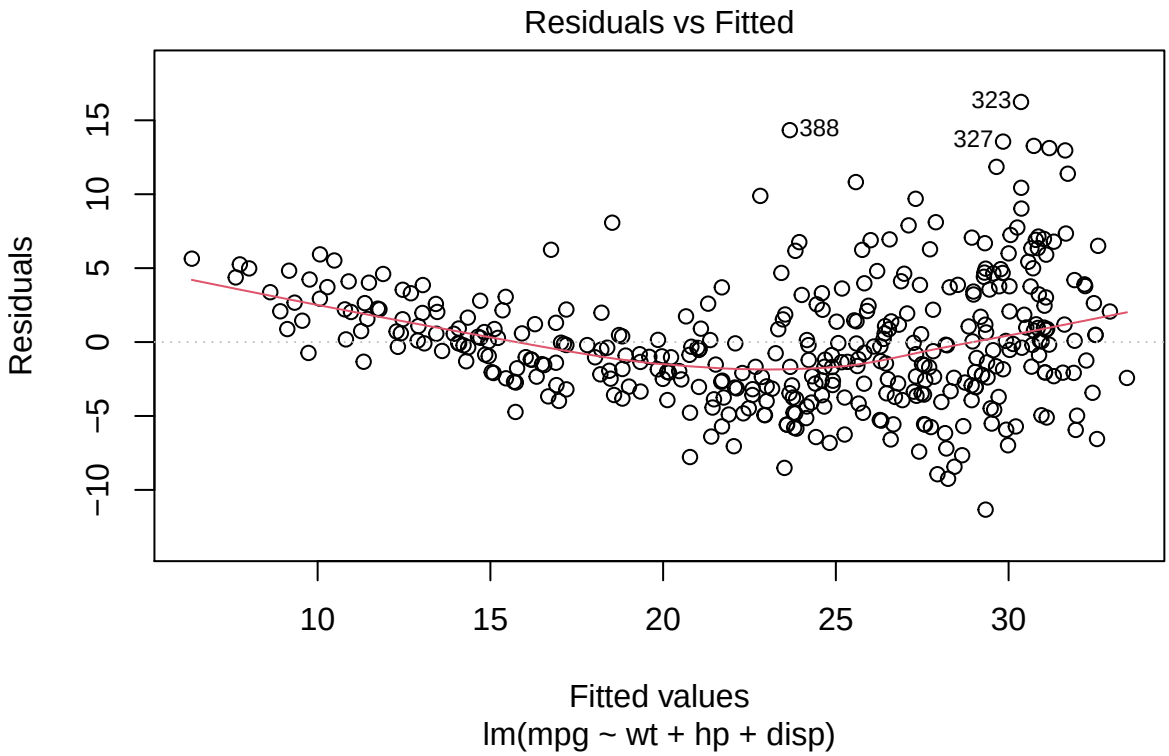
```r
par(mfrow = c(2, 2))
plot(mdl)
```



Residuals vs. Fitted

First we'll look at the residuals versus fitted values plot. With this plot we want to look for any pattern in the data. Ideally, we should see points distributed around zero, with the same variance throughout the whole space.

```
plot(mdl, which = 1)
```



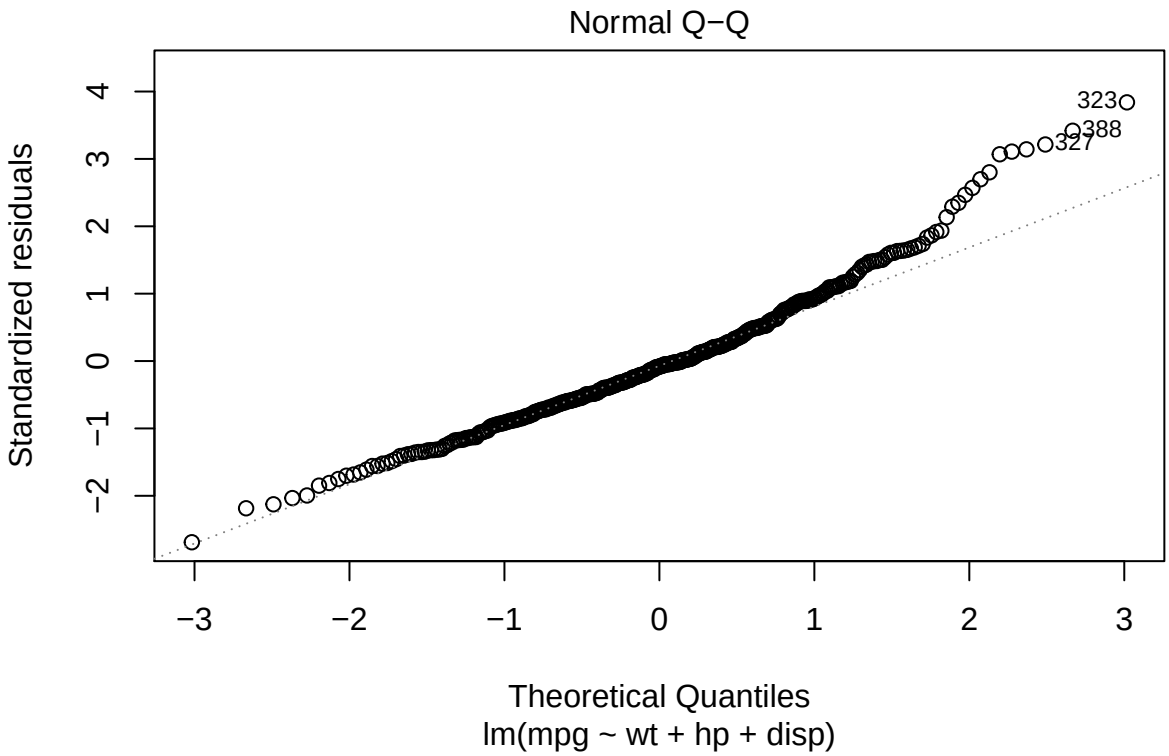
Looking at the plot above, our model seems to be adequate with respect to the linear relationship assumption. There does seem to be a light nonlinearity, but again, we only want to point out obvious deviations.

When looking at the variance of the residuals, it's important to keep in mind the number of data points in each part of the plot. Here on the right we have quite a bit more data than on the left, so we don't want to read too far into the appearance of more variation in the residuals on the right than on the left. It could be that if we had a few more data points whose fitted values are less than 20 that the variation would appear even out. We can corroborate our conclusion from this plot with the scale-location plot.

Normal Q-Q

Now, we'll look at the Normal Q-Q (quantile-quantile) plot to check the normality of errors assumption. Ideally the points should fall on the straight dotted line.

```
plot(mdl, which = 2)
```

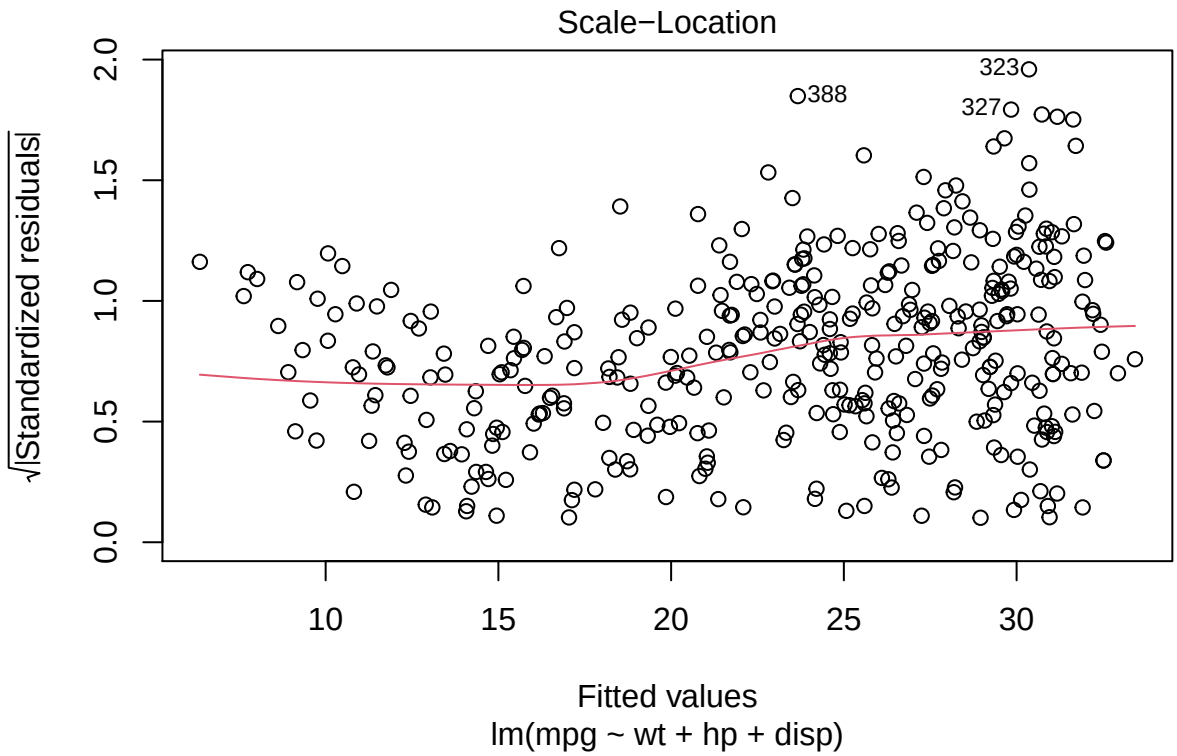


Here we observe that for our model the residuals do adequately meet the normality assumption. The deviation seen on the right of this plot suggest a slight positive skew to the distribution of the residuals, but it is not large enough to be of concern.

Scale-Location

Next we look at the scale-location plot. This is used to check that the residuals display equal variance along the model plane. With this plot we want to see a horizontal line with points evenly spread above and below it.

```
plot(mdl, which = 3)
```



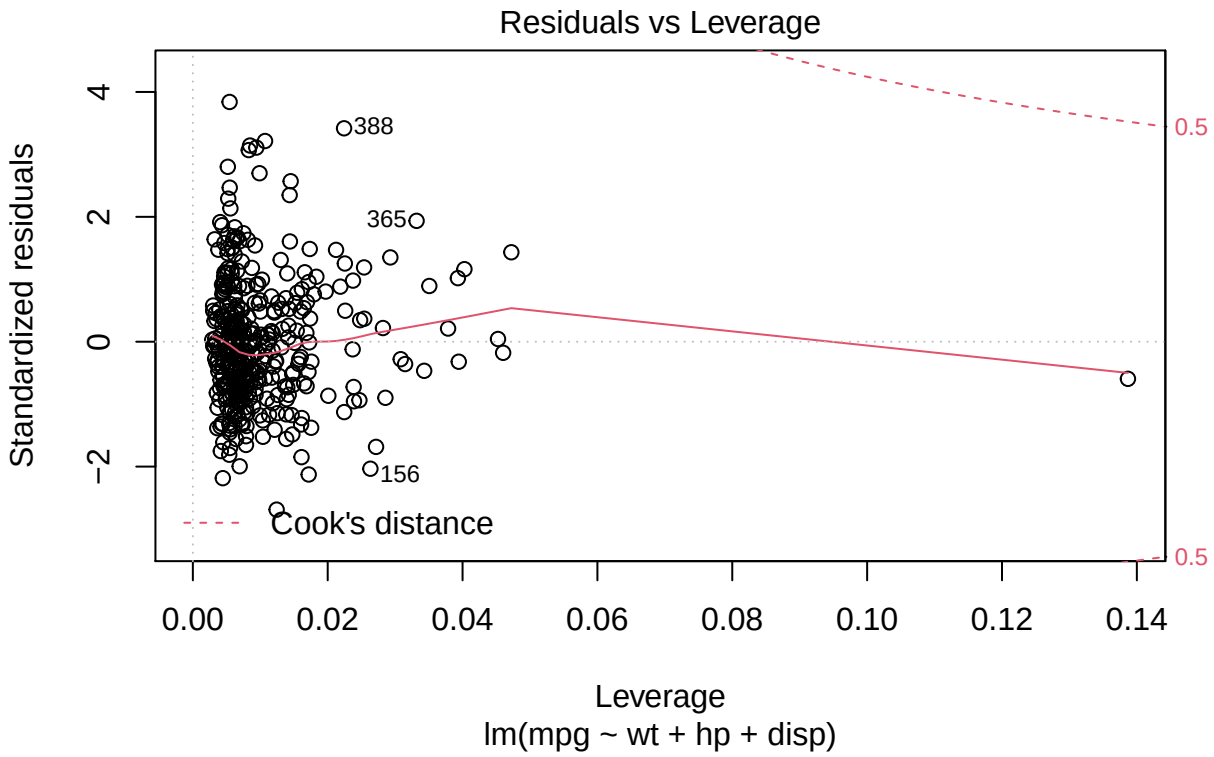
So, for our model this plot looks pretty good. Again there seems to be less variation on the left, but there is also less data so we say the assumption of constant variance is met.

Residuals vs. Leverage

Now, we look at the residuals vs. leverage plot. This is used to find influential observations in the dataset, i.e. an observation that, if removed, will have a great affect on our fit model. Influential points may or may not be outliers, there are other ways to check for outliers, though this is a useful plot to identify possible outliers.

Influential points will appear outside of Cooks Distance line marked by the dotted red line.

```
plot(mdl, which = 5)
```

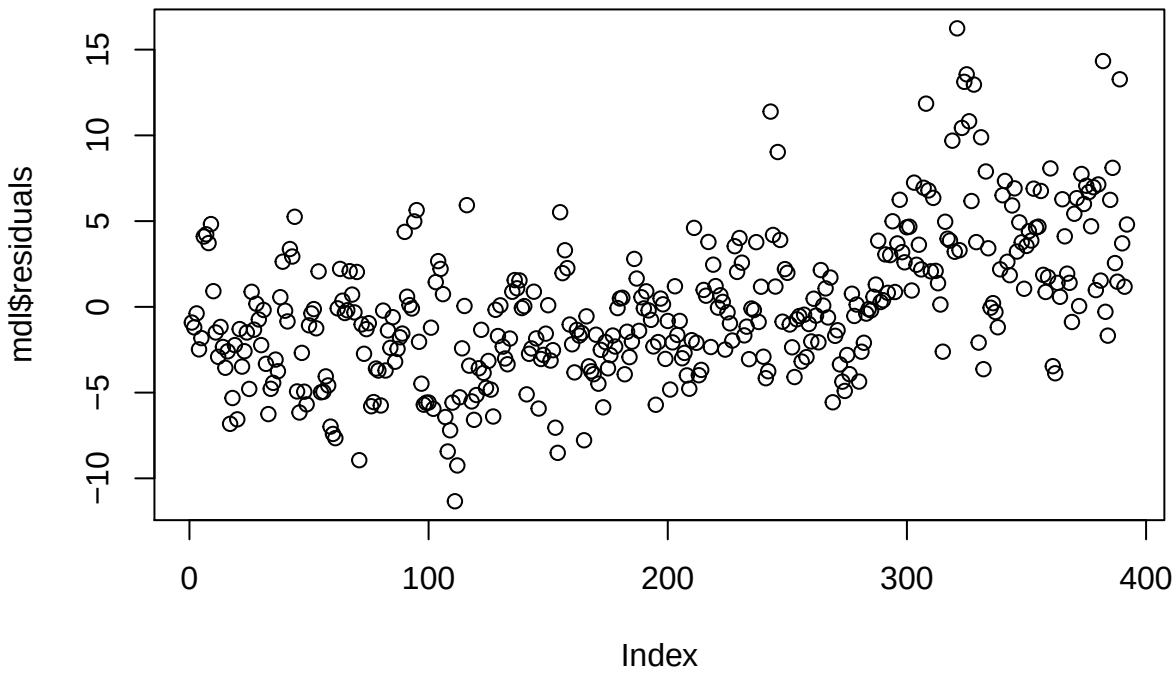


For our model we see that none of the points in our dataset has high leverage.

Residual vs. Index

This last plot with residuals on the y-axis and their index on the x-axis allows us to check for correlation between the residuals. Again here we want to to see an even scattering around $e = 0$

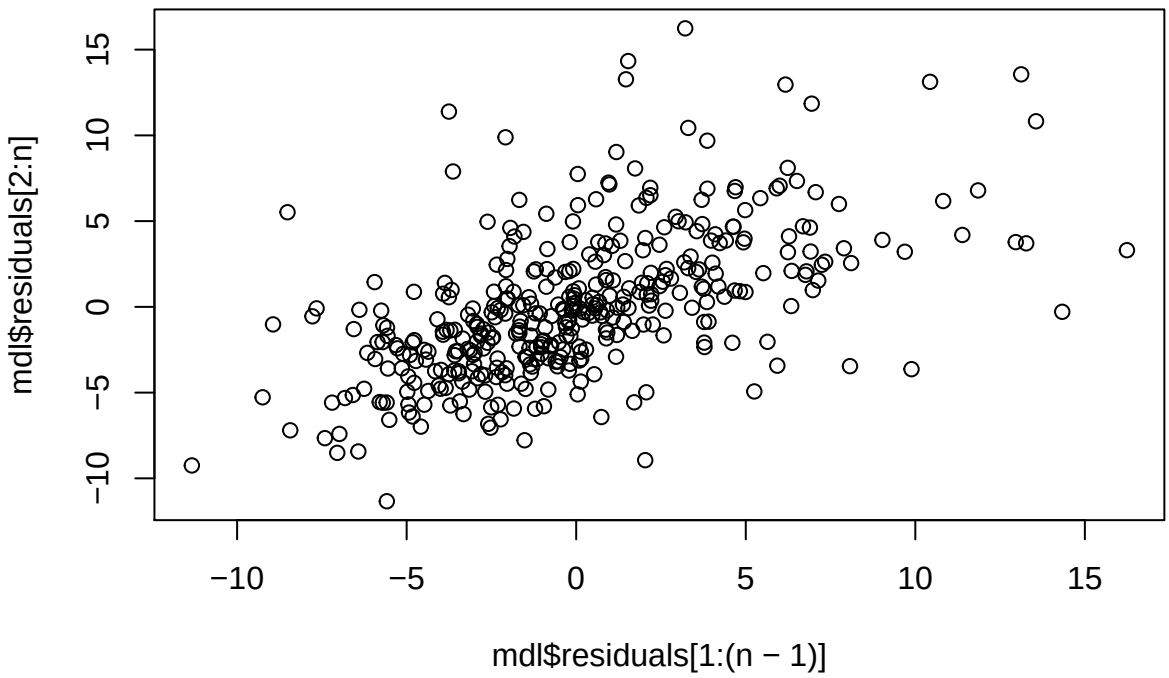
```
plot(mdl$residuals)
```



So, for our model we do get somewhat of an even scattering, though there is a slight rise in the value after the 200th residual.

We can also plot succesive pairs of residuals. With this plot we would like to see a "shotgun blast", indicating there is no correlation.

```
n <- nrow(auto_mpg)
plot(mdl$residuals[1:(n-1)], mdl$residuals[2:n])
```



Here we see that the points are scattered throughout the space. There does seem to be a very slight positive correlation between the residuals here but I don't think we need to worry about it. There are tests that allow us to check for the significance of this correlation which we will get into in future tutorials.

Final Remarks

In this tutorial we went over some common issues with multiple regression models, followed by graphical methods to asses the model. I added an extra regressor to our regression model from last week. When consideringthe predictors in the model we found that there was some issue with multicollinearity between regressors. Through residual analysis we concluded that the model assumptions were approximately met, but there are some questions; this is likely due to the presence of multicollinearity. In the coming tutorials we will cover how to deal with these issues, along with what to do when model assumtpions are not met.