Regression Diagnostics

Dr. Sarah Hunter

10/19/2020

Regression Assumptions

Linear regression makes several assumptions about the data when you choose to estimated the model. These assumptions come in two sets: mathematical assumptions behind the model and assumptions that allow us to make inference. I will briefly list those assumptions here, but you can see your textbook for more details.

Gauss-Markov Assumptions

Linear regression has several mathematical assumptions included. Violations of these assumptions can cause either inefficiency (inflated standard errors) or bias (incorrect parameter estimates). The Gauss-Markov Theorem states that when the following assumptions are met, Ordinary Least Squares (OLS) is **BLUE**(Best Linear Unbiased Estimator):

- The model is linear in the parameters
- X values are fixed and independent of the error term
- The model has the correct functional form
- Errors are Homoscedastic
- There is no autocorrelation
- N>k (N is the number of observations and k is the number of estimated parameters)
- There must be variation in X.

Outside of the Gauss-Markov assumptions, OLS also assumes that the variables are correctly measures, that random sampling was used, and there is no multicollinearity.

The Normality Assumption and Statistical Inference

The Gauss-Markov assumptions are necessary for OLS to be an efficient and unbiased estimator. However, there is one additional assumption that must be met in order to use OLS for statistical inference. The errors must be normally distributed in order to invoke the Central Limit Theorem and therefore do statistical inference.

Bias v. Inefficiency

There are many possible violations of OLS assumptions which can lead to one of two issues: bias or inefficiency. Bias means that our actual estimates of our regression coefficients is wrong, or does not reflect the actual relationship between X and Y in the population. Inefficiency occurs when the standard errors are inflated over what they otherwise would be.

The assumption violations that lead to bias, and are therefore far more of an issue in OLS are:

- Functional form issues
- Autocorrelation

- Measurement error in the independent variable

The assumption violations that lead to inefficiency are:

- Heteroscedasticy
- Multicollinearity
- Measurement error in the dependent variable

In this document, I will defining each assumption violation in turn and demonstrating how to diagnose and fix the issue. First, I estimate a model, using the Prestige dataset from the `car` library. These data are on 1980s Canadian occupational prestige (a score ranging from 0 to 100, where higher numbers represent more prestige). The model code is as follows:

```
library(car)
```

```
## Loading required package: carData
```

```
#A usual multiple regression model.
mod1<-lm(prestige~income+women+education, data=Prestige)

summary(mod1)
```

```
##
## Call:
## lm(formula = prestige ~ income + women + education, data = Prestige)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.8246  -5.3332  -0.1364   5.1587  17.5045
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.7943342  3.2390886  -2.098   0.0385 *
## income       0.0013136  0.0002778   4.729 7.58e-06 ***
## women       -0.0089052  0.0304071  -0.293   0.7702
## education    4.1866373  0.3887013  10.771  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.846 on 98 degrees of freedom
## Multiple R-squared:  0.7982, Adjusted R-squared:  0.792
## F-statistic: 129.2 on 3 and 98 DF,  p-value: < 2.2e-16
```

Normality of the Risiduals

The residuals (or the errors) must be normally distributed in order to make inference about our results. To do this, we must first extract the residuals from R. Remember that residuals are defined as:

$\hat{u}_i = Y_i - \hat{Y}_i$

Where Y_i is the actual value of Y for observation i and $\hat{Y}_i$ is the predicted value of Y for observation i.

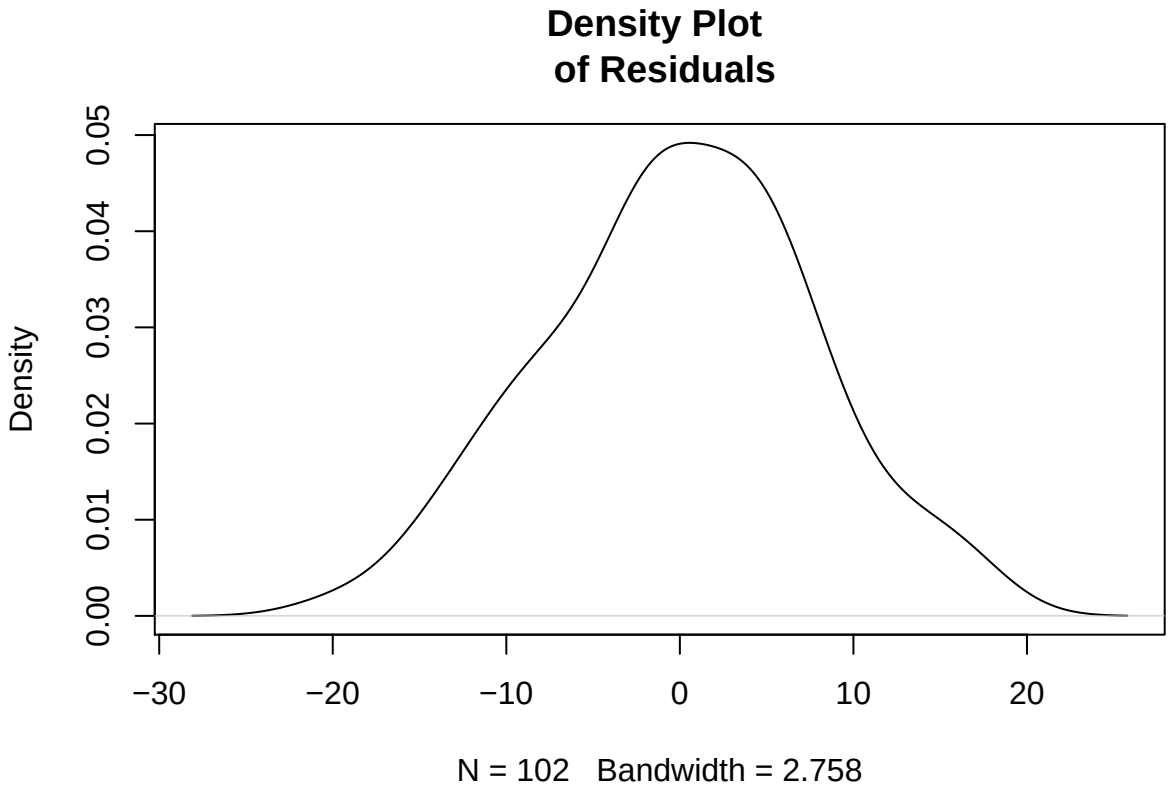
To get R to calculate the residuals for us, we use the following code:

```
resid<-mod1$residuals
```

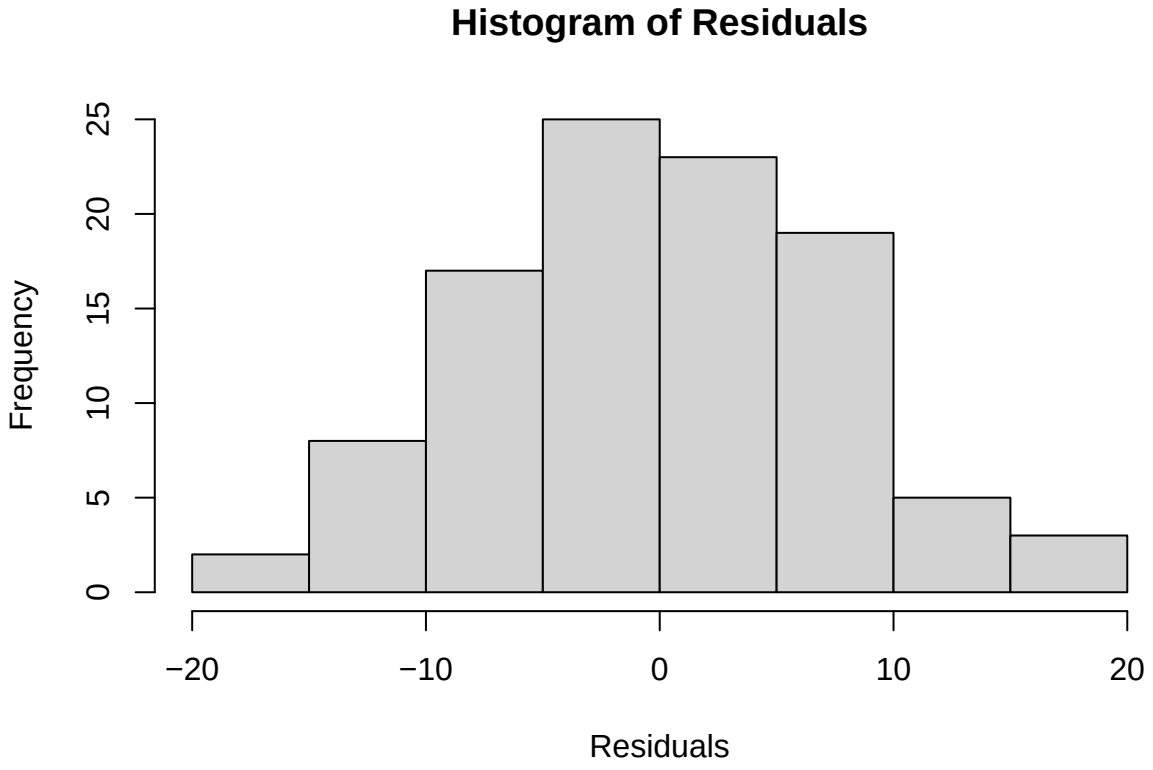
To check the normality of the residuals, there are several graphical options: the QQplot, a density plot, and a histogram. Remember that a normal distribution is a symmetric distribution where the mean, median, and

mode are all equal. Therefore, we are looking for the same to be true of our density plots and histograms for the residuals. The following code creates both types of plots:

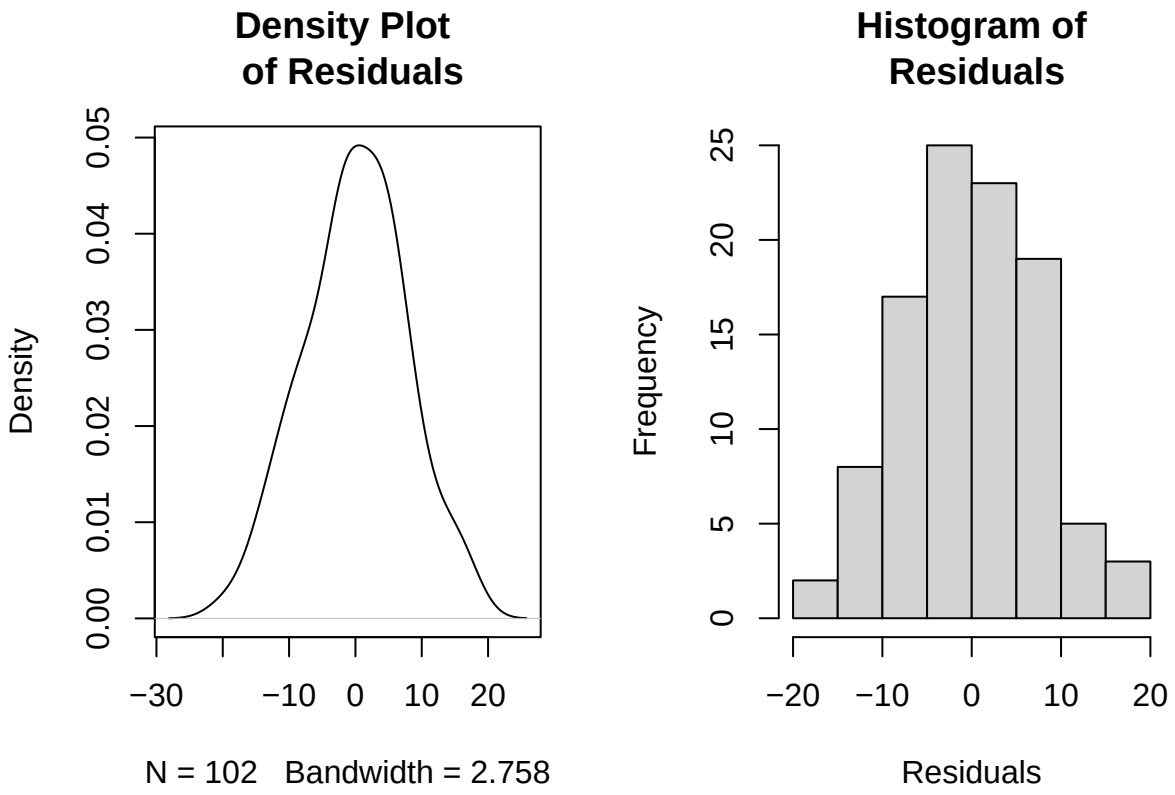
```
#the density plot

plot(density(resid), main="Density Plot \n of Residuals")
```




```
#the histogram

hist(resid, main="Histogram of Residuals", xlab="Residuals")
```



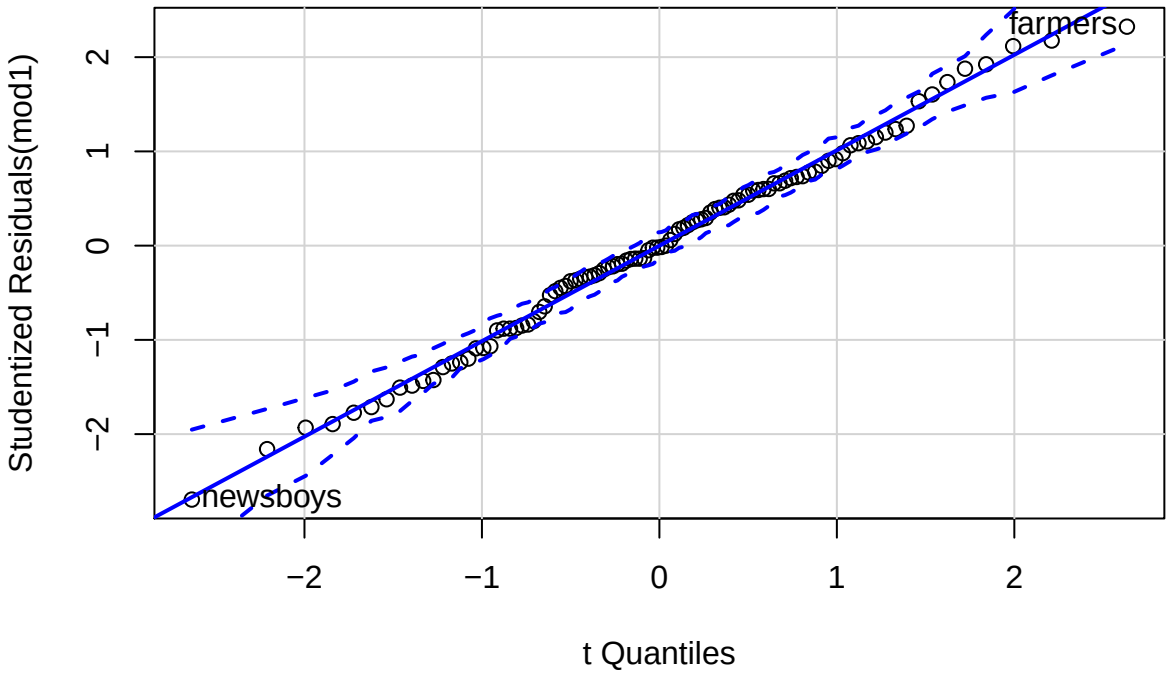

```
#Putting both figures on one

par(mfrow=c(1,2))
plot(density(resid), main="Density Plot \n of Residuals")
hist(resid, main="Histogram of\n Residuals", xlab="Residuals")
```



The way to interpret these plots is to look at the shape of the plots. Do they look normal-ish? I would argue that these residuals look to be roughly normally distributed. You should also consider the number of observations (102). The larger the number of observations, the more clearly normal we would expect the residuals to look.

If you want a diagnostic that is more precise, you can look to the QQplot. This is a plot designed to see if the residuals and the normal distribution quantiles match. The code is as follows:

```
#This is also in the car library
library(car)

qqPlot(mod1)
```



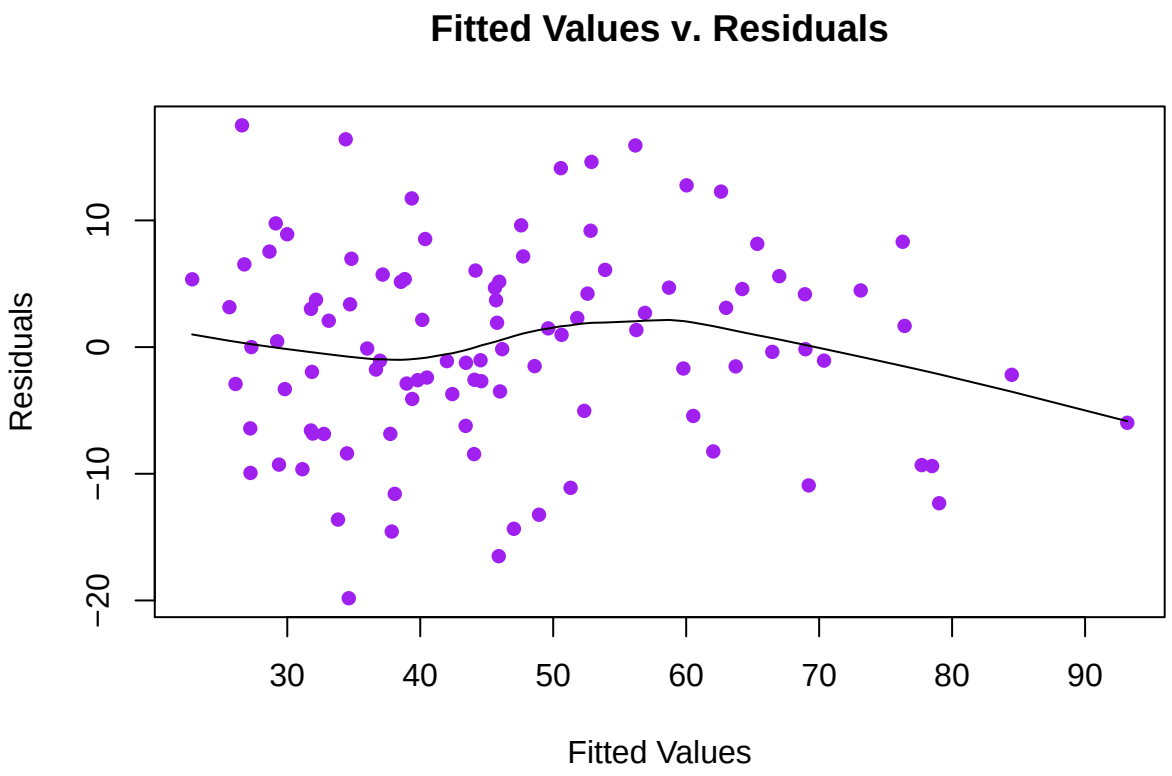

```
## newsboys  farmers
##       53       67
```

To interpret this figure, you are looking for most of the points to be on or as close as possible to the line. This plot also includes a margin of error. Therefore, as long as a vast majority of your residuals are inside the dashed lines (margin of error), you have normally distributed residuals. This command also gives you the highest absolute value residuals.

Function Form Diagnostics

Several issues can cause you to have the incorrect functional form of your model. These issues range from omitted variable bias to no correctly characterizing the relationship between X and Y (i.e. failing to include a squared term when the relationship between X and Y is quadratic rather than linear). Non-normal residuals can often point to functional form issues, however other residual analysis can also help diagnose said functional form issues. The following code plots the residuals against the fitted value of Y:

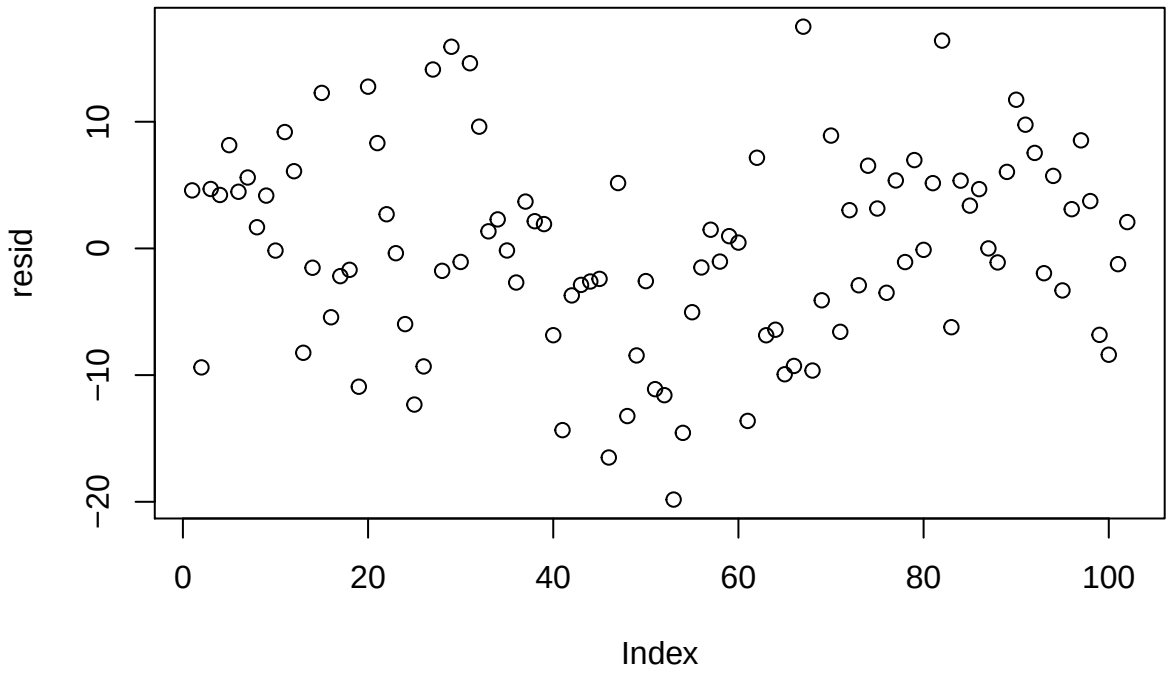
```
#First find the fitted values of Y

fitted<-fitted.values(mod1)

#Residual v. fitted

plot(x=fitted, y=resid, pch=16, col="purple", xlab="Fitted Values", ylab="Residuals", main="Fitted Valu
lines(lowess(fitted, resid))
```



In the above plot, you are looking for a systematic relationship between the residuals and the fitted values. The line drawn in the plot is called a lowess line, which follows the data (without being a straight line). You want the lowess to look roughly like a straight line at 0. If there looks to be a curve or a pattern, then you should be concerned. I would say that is nothing to be concerned about here. While the line does bend a bit, it does not do so dramatically.

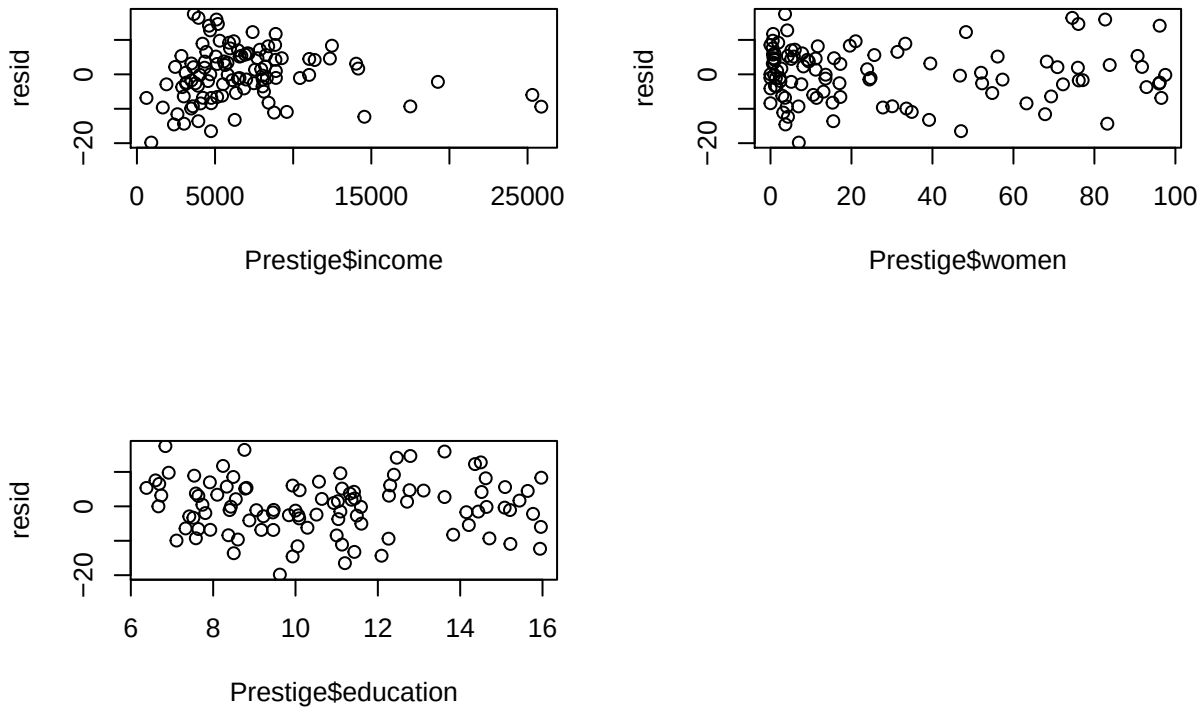
You can also look at a plot of just the residuals. This plot shows the value of the residual for each observation in order. You are looking for the points to be randomly distributed around 0 the entire length of the indices.

```
plot(resid)
```



The last of the residual analysis that should be conducted is to see if the residuals correlate with any of the independent variables from the model. The following code shows how to do this. Remember that the `par(mfrow=c(1,2))` command puts two graphs on one line. Since we have three plots, we will need to adjust the command to put 2 plots on each row for 2 rows:

```
par(mfrow=c(2,2))
plot(x=Prestige$income, y=resid)
plot(x=Prestige$women, y=resid)
plot(x=Prestige$education, y=resid)
```



Looking at the above figures, I see no evidence of correlation between the independent variables and the

residuals. To be sure, you can always find the correlation coefficient between the two. See the `cor.test()` command from the Bivariate Hypothesis Testing R Markdown file.

Autocorrelation

Autocorrelation occurs when the residual of one observation is correlated with the residual at the next observation. More formally:

$Cov(u_i, u_j) = 0$ when $i \neq j$

OLS assumes that the residuals are independent of each other. When there is autocorrelation present, the residuals are instead correlated with each other. This problem is most present in either time series or spatial data. Autocorrelation can cause OLS to estimate the incorrect standard errors, making them smaller than they should be. This can, in turn, lead to incorrect inference (what we learn about the population from our sample). We can test for autocorrelation in many ways, but this lesson focuses on two statistical tests: the Durbin-Watson Test and the Breusch-Godfrey test.

The Durbin-Watson test is designed to look for serial autocorrelation. This is a kind of hypothesis test, just like a t test, therefore it is important to understand the null and alternative hypotheses being tested here. In the Durbin-Watson test:

- H_0 = No serial autocorrelation
- H_a = Serial autocorrelation

This means that we do not want to reject the null hypothesis. If we get a statistically significant p-value (less than .05), then we reject the null hypothesis that there is no serial autocorrelation. To run this diagnostic in R, you need to first install the `lmtest` library:

```
#You need the lmtest library for this command
#install.packages("lmtest")

#Once installed,  you need to load the library
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
#The actual test
dwtest(mod1)
```

```
##
##  Durbin-Watson test
##
## data:  mod1
## DW = 1.6869, p-value = 0.04103
## alternative hypothesis: true autocorrelation is greater than 0
```

The results of the Durbin-Watson test have a p-value of less than .05, which means that we reject the null hypothesis of no serial autocorrelation. However, these are not time series data, therefore we had not prior expectation for serial autocorrelation. As a result, we should use another diagnostic to be sure.

The other autocorrelation diagnostic test we can use the Breusch-Godfrey test. With this test, the null and alternative hypothesis remain the same as the Durbin-Watson test:

- H_0 = No serial autocorrelation
- H_a = Serial autocorrelation

The difference between the Durbin-Watson test and the Breusch-Godfrey test is that the Durbin-Watson test can only be used on nonstochastic regressors and can only test for first order autocorrelation. With neither restrictions, the Breusch-Godfrey test is a more general test. To conduct the test in R:

```
#Also from the lmtest library

bgtest(mod1)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  mod1
## LM test = 2.4634, df = 1, p-value = 0.1165
```

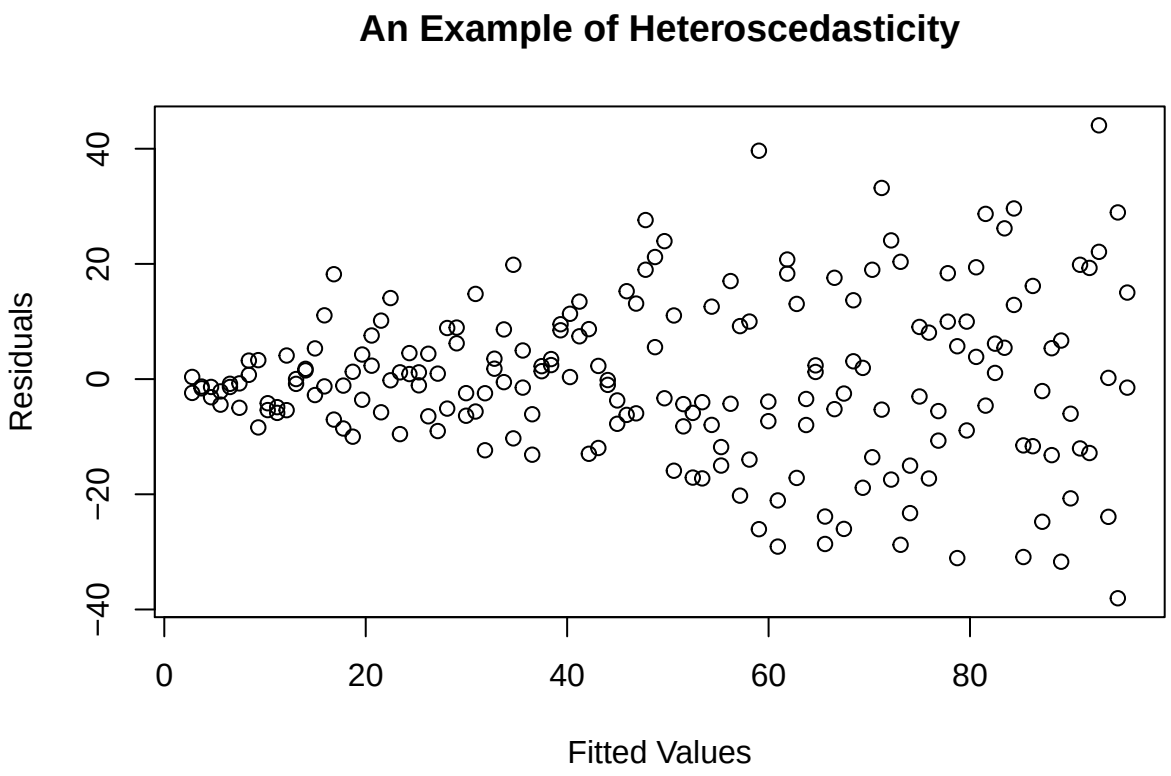
The Breusch-Godfrey test produces a p-value of greater than .05, therefore we cannot reject the null hypothesis of no serial autocorrelation.

The problem we face in this circumstance is the conflicting test statistics. In this case, I would usually rely on logic and the residual plots. As the residual plots have shown nothing of concern, and these data are not time series, I would go with the Breusch-Godfrey test and say there is no evidence of serial autocorrelation.

Heteroscedasticity

Heteroscedasticity simply means non-constant error variance. Our regression model assumes that we have homoscedasticity, meaning we have constant error variance. When heteroscedasticity is present, the standard errors are incorrectly estimated in your regression model, usually making them larger than they should be.

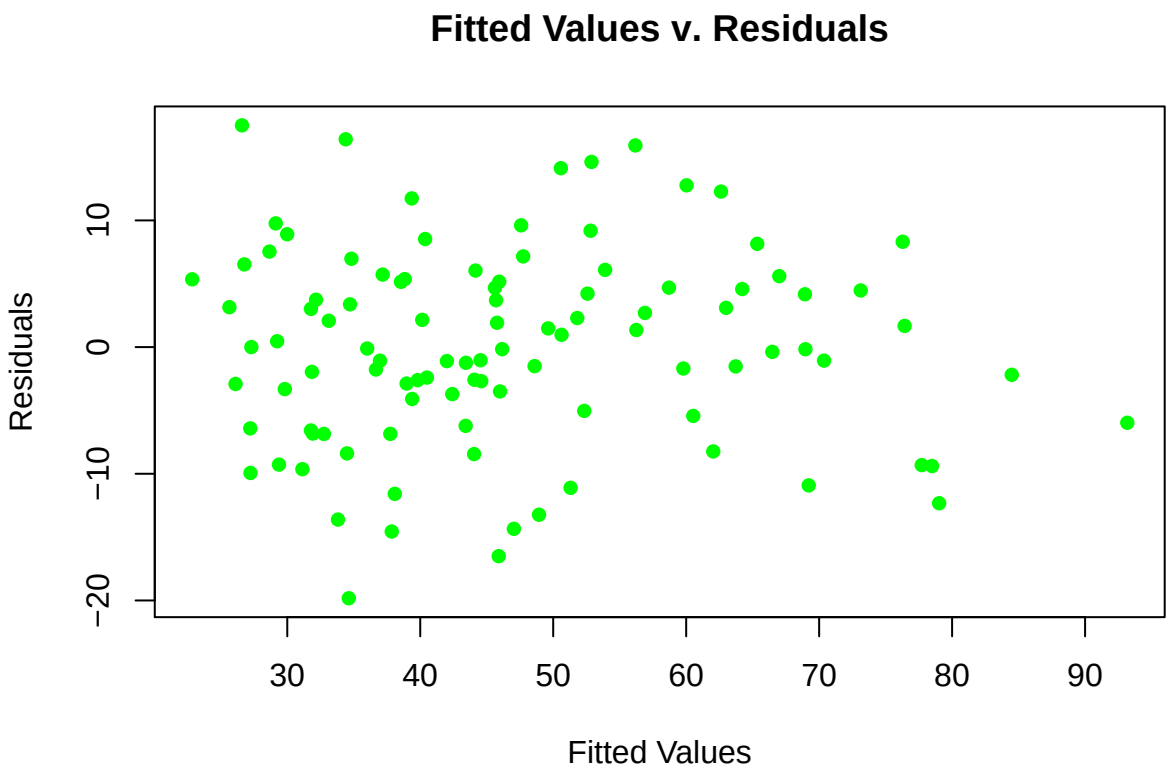
The best way to check for heteroscedasticity in R is to plot the residuals against the fitted values of the model. You are looking for a "trumpet" shape. The figure below shows what heteroscedasticity would look like:



Notice that in the figure above, the residuals at the low end of the fitted values are much closer together, clustered around 0, than at the higher end, where they are all scattered. If your residual plot looks like this, you have heteroscedasticity.

Below is an example of how to diagnose our model from above for heteroscedasticity:

```
plot(x=fitted, y=resid, pch=16, col="green", xlab="Fitted Values", ylab="Residuals", main="Fitted Value
```



This figure shows no evidence of heteroscedasticity. Therefore, we do not need to anything to our model. The most concerning feature of this plot is the few points that are on the upper end of the fitted values, but more on that later.

Multicollinearity

Another assumption of multiple regression with OLS is no multicollinearity. Multicollinearity happens when one or more of independent variables are closely related. Perfect multicollinearity happens when one of the independent variables is a linear function of another independent variable. For example, if you included both age and year of birth into a model, that would be perfect multicollinearity. However, problematic multicollinearity is not always that obvious.

The most common causes of multicollinearity are a small sample and high correlation between independent variables. The first indication that you have multicollinearity is when you have a high R^2 value, but few statistically significant coefficients. If this occurs, you should check for multicollinearity by calculating the * variance inflation factor * (VIF). This test will essentially estimate several models where each X variable is the dependent variable and the other X variables are the independent variables. The VIF takes the R^2 from each of these models and calculates:

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

Where VIF_j is the VIF for the model ran with X_j as the dependent variable and the other Xs as the independent variables. R_j^2 is the R^2 for the model ran with X_j as the dependent variable and the other Xs as the independent variables. This will give you a VIF score for each X variable in your model. R can estimate the VIF for you with one line of code from the `car` library:

```
vif(mod1)
```

```
##    income     women education
##  2.282038  1.526593  1.845165
```

If any X gives a $VIF > 5$, then you have high enough multicollinearity to be concerning. If this does occur, the best fix is to make sure you do not have two measures of the same concept. If that is not the case, could you use a different measure of something? Are all your variables necessary?

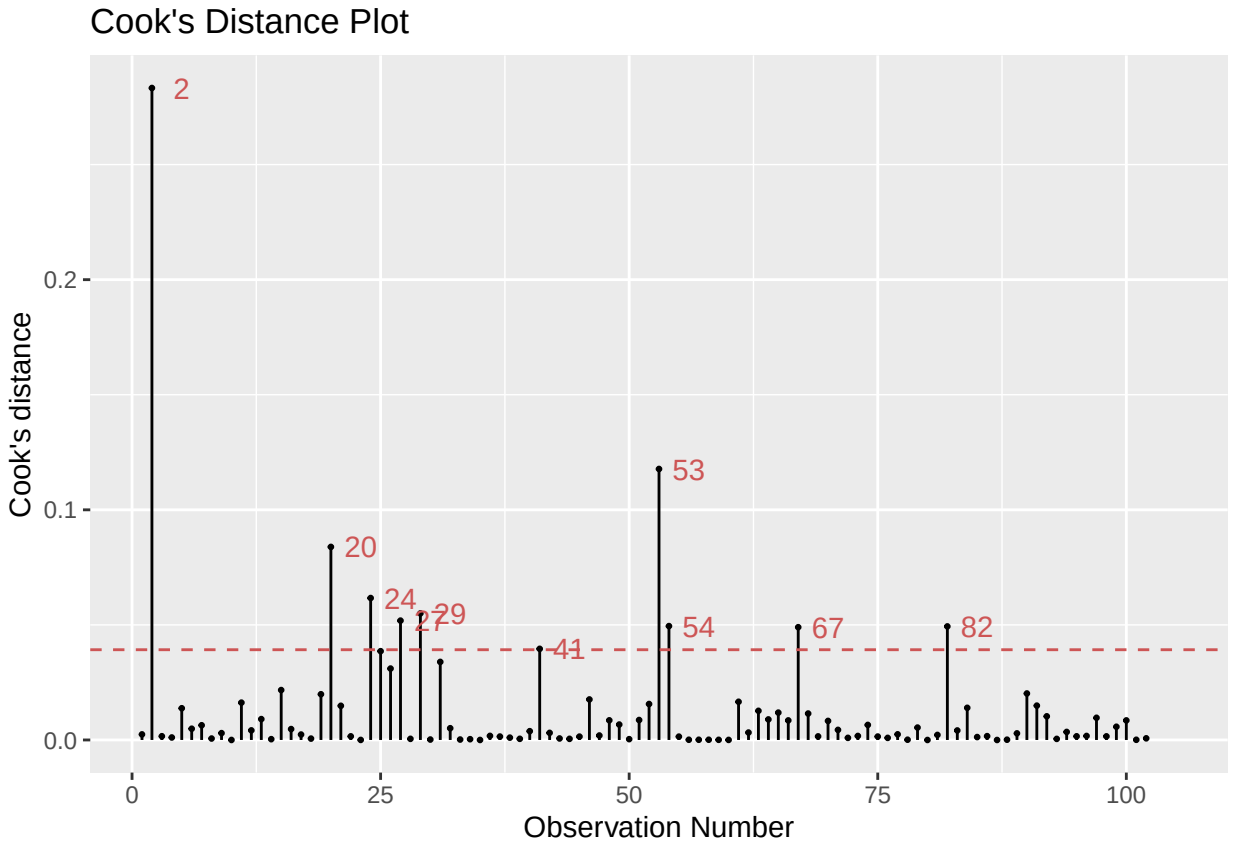
In the end, multicollinearity will be mostly likely to make your model inefficient (i.e. have larger standard errors) rather than biased.

Influential Observations

Influential observations are those that have sufficiently unusual values of the independent variables (* leverage *) and a high residual value. Commonly called outliers, observations are only considered influential if they have both of those qualities (high leverage and a high residual). Influential observations can actually bias regression results. You can check for influential observations in many different ways.

The first way to test for influential observations is through the Cook's distance. Cook's D calculates the sum of all changes in a regression model when one observation is removed. Each observation gets a Cook's D score. If an observation has a $D > \frac{4}{n-k-1}$, that observation is considered influential, where n is the total number of observations and k is the number of parameters to estimate. You can get Cook's D scores in a figure with the following code:

```
#Need this package
#install.packages("lindia")

#Load the package
library(lindia)

#Get the Cook's D plot
gg_cooksd(mod1)
```



In this plot, any of the points above the line indicate influential observations. The plot also gives the row number of the observation that is an influential observation. You can use the row number to find the observation that is mentioned by using the following code:

```
#use the name of the data and [] to get the
#obs that matches that row.

#the [] notation looks like: [row no., column no.]

Prestige[2,]
```

```
##                   education income women prestige census type
## general.managers     12.26  25879  4.02     69.1   1130 prof
```

So this tells us that general managers have a large amount of influence on our model. The next two larges Cook's D scores are row numbers 53 and 20 which correspond to:

```
Prestige[53,]
```

```
##          education income women prestige census type
## newsboys      9.62    918     7     14.8   5143 <NA>
```

```
Prestige[20,]
```

```
##           education income women prestige census type
## ministers      14.5   4686  4.14     72.8   2511 prof
```

Therefore, newsboys and ministers also have influence on our model.

We can also get Cook's Distance scores for each observation and then only select the ones that are greater than the cutoff. To do this, we first create a new object called `cutoff` that uses the equation from above

($D > \frac{4}{n-k-1}$). From there, we can use the command `cooks.distance` to get the Cook's D score. Then, you can use the which command to find the observations with Cook's D scores higher than the cutoff:

```
#Getting the "max" value of Cook's D

cutoff<- 4/(length(Prestige$income)-4-1)

#Finding only the observations with Cooks D> cutoff
Prestige[which(cooks.distance(mod1)>cutoff),]
```

```
##                           education income women prestige census type
## general.managers              12.26  25879  4.02     69.1   1130 prof
## ministers                     14.50   4686  4.14     72.8   2511 prof
## physicians                    15.96  25308 10.56     87.2   3111 prof
## nurses                        12.46   4614 96.12     64.7   3131 prof
## physio.therapsts              13.62   5092 82.66     72.1   3137 prof
## newsboys                       9.62    918  7.00     14.8   5143 <NA>
## service.station.attendant      9.93   2370  3.69     23.3   5145   bc
## farmers                        6.84   3643  3.60     44.1   7112 <NA>
## electronic.workers             8.76   3942 74.54     50.8   8534   bc
```

The second route to take to find the DF Beta score for each observation. This statistic functions much like Cook's distance, but instead of looking at the overall model, it looks at how each observation changes each estimated regression coefficient. The difference in one coefficient between what was estimated with and without that observation is the DF Beta score for that observation for that particular coefficient. Like Cook's D, there is a cutoff value where we should be concerned about influence. This cutoff dependents on the size of the dataset. To be concerned about influence:

- For small/medium data sets: $DFBeta > |1|$
- For large datasets: $DFBeta > |\frac{2}{\sqrt{n}}|$

To get the DFBeta scores in R, the command is simply `dfbetas` in the `car` library. We can then use the same `which` command to sort through the DFBetas score and only get those observations with scores over the cutoff value.

```
#DFBetas

dfb<-dfbetas(mod1)

#Easier way to sort them

Prestige[which(abs(dfbetas(mod1)[,'education']) > 1),]
```

```
## [1] education income    women     prestige  census    type
## <0 rows> (or 0-length row.names)
```

```
Prestige[which(abs(dfbetas(mod1)[,'income']) > 1),]
```

```
##                   education income women prestige census type
## general.managers     12.26  25879  4.02     69.1   1130 prof
```

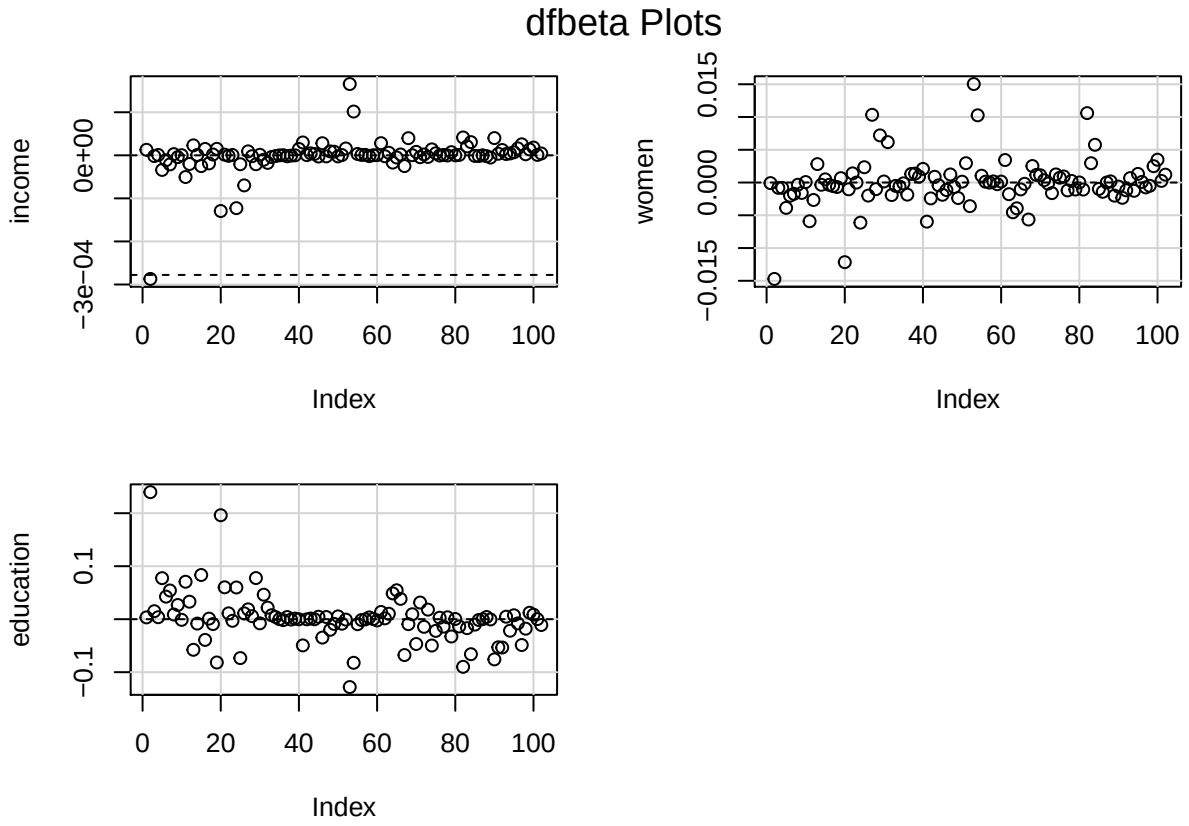
```
Prestige[which(abs(dfbetas(mod1)[,'women']) > 1),]
```

```
## [1] education income    women     prestige  census    type
## <0 rows> (or 0-length row.names)
```

This shows that the only influential observation that impacts our slope coefficients is general managers, which impacts the value of the coefficient on income.

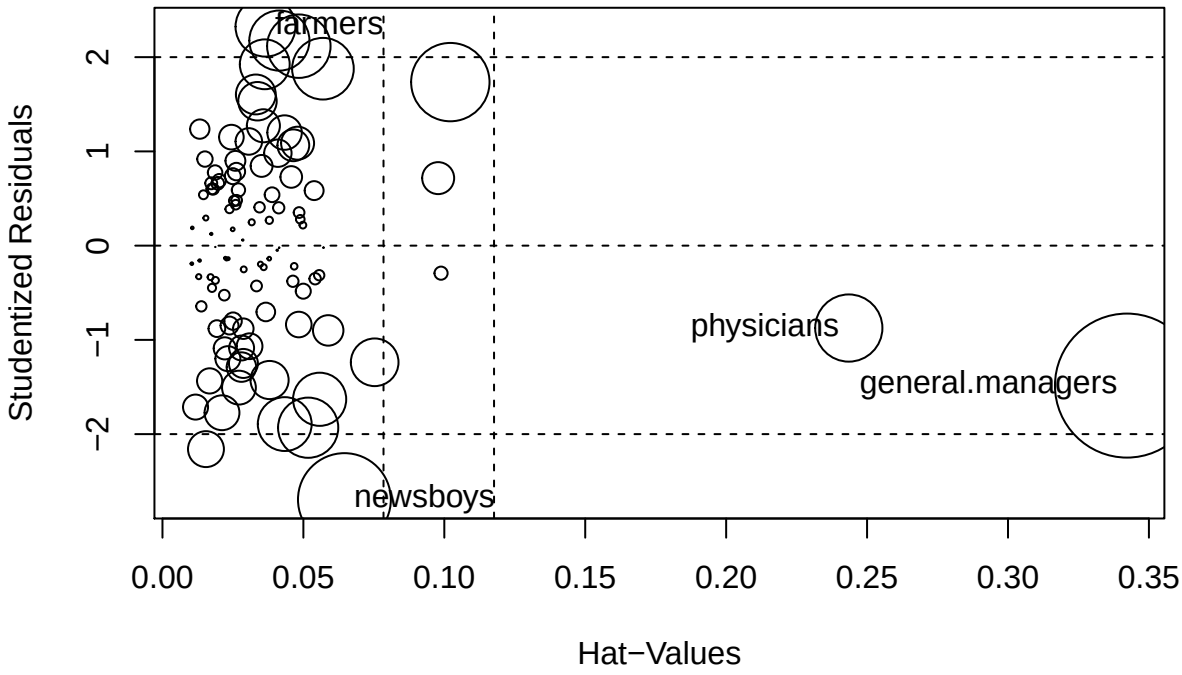
Another way to use the DFBetas is to plot the DF Betas for each estimated coefficient:

```
dfbetaPlots(mod1)
```



Another way to look at influential observations is called an Influence Plot, which shows both leverage and residuals. The larger circles in the top and bottom right corners of the plot are the influential observations, as calculated by the Cook's D score. The code for this figure is below:

```
influencePlot(mod1)
```




```
##                     StudRes        Hat      CookD
## general.managers -1.4848372 0.34221178 0.28326973
## physicians       -0.8743077 0.24351491 0.06166504
## newsboys         -2.6944419 0.06456007 0.11774271
## farmers           2.3232364 0.03655496 0.04899856
```