Prestige Dataset

P.Weber, S. Matter J.Peter, R.Pircher, L. Schaffluetzel

May 23, 2019

Linear Regression - Predicting Occupational Prestige for Human Resources & Career Planning

Description of the Dataset & Business Understanding

This data set contains an analysis of different occupations, which were obtained from Fox, J. and Weisberg, S. (2011) An R Companion to Applied Regression, Second Edition. The Prestige data frame has 102 rows and 6 columns where each observation represents an occupation. The columns relate to predictors such as average years, percentage of woman in the occupation, prestige of the occupation etc.

Occupational prestige or also known as job prestige is a way for sociologists to determine the relative social class that people have. This basically refers to the consensual nature of rating a job based on the belief of its worthiness. Transferring this basic idea into our vast and fast shifting business society we see more and more people not only relying on their judgment and instincts, but proving their hypothesis by using KPIs. Prestige scores are used by head-hunters, recruiters in HR to rank business schools, banking firms and consulting companies to use it as a criterion to hire or not. However, this dataset does not score companies and schools or universities, but general occupation classes.

Attributes

- education: Average education of occupational incumbents, years in 1971.
- women: Percentage verage education of occupational incumbents, years in 1971.
- income: Average iof incumbents who are woman.
- prestige: Pineo-Porter prestige score for occupation, from a social survey conducted in the mid 1960s.
- census: Canadian Census occupational code.
- type: Type of occupation. A factor with levels. (note: out of order) bc: Blue Collar;prof, Professional, Managerial and Technical; wc, White Collar.

Environment Installation

In this section we are going to instal all relevant libraries. This task can be skipped if you have already installed the Packages.

```
#install.packages('caTools') #Create Training and Testing dataset
#install.packages('dplyr') #For Data Manipulation
#install.packages('ggplot2') # for some amazing looking graphs
#install.packages('caret') # for confusion matrix
#install.packages('corrplot') #Correlation Plot
#install.packages('car') # This is the library containing the dataset
#install.packages('randomForest') # for RandomForest Alogrithm
#install.packages('cowplot') #to use plot_grid
#install.packages('gridExtra') #plot for cooks distance
```

The next step is to activate the Libraries.

```
options(warn = -1) # Suppress Warnings
library(caTools)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(cowplot)
```

```
##
## Attaching package: 'cowplot'

## The following object is masked from 'package:ggplot2':
##
##     ggsave
```

```
library(carData) # Loading Prestige dataset
```

Load the Dataset

For the sake of training we decided to use the Library carData instead of the csv. Now we split the dataset into train and test sets. But first we have a look at the dataset with the head() command (same as view()).

```
dataset <- (Prestige)
head(dataset)
```

```
##                     education income women prestige census type
## gov.administrators      13.11  12351 11.16     68.8   1113 prof
## general.managers        12.26  25879  4.02     69.1   1130 prof
## accountants             12.77   9271 15.70     63.4   1171 prof
## purchasing.officers     11.42   8865  9.11     56.8   1175 prof
## chemists                14.62   8403 11.68     73.5   2111 prof
## physicists              15.64  11030  5.13     77.6   2113 prof
```

```
#trainingset_x <- select (trainingset_x, -c(census, education)) to get rid of columns
```

Data Understanding

With the commands below we are going to explore the dataset.

```r
str(dataset) # it will return a data frame with four columns: variables, class, levels and examples
```

```
## 'data.frame':    102 obs. of  6 variables:
##  $ education: num  13.1 12.3 12.8 11.4 14.6 ...
##  $ income   : int  12351 25879 9271 8865 8403 11030 8258 14163 11377 11023 ...
##  $ women    : num  11.16 4.02 15.7 9.11 11.68 ...
##  $ prestige : num  68.8 69.1 63.4 56.8 73.5 77.6 72.6 78.1 73.1 68.8 ...
##  $ census   : int  1113 1130 1171 1175 2111 2113 2133 2141 2143 2153 ...
##  $ type     : Factor w/ 3 levels "bc","prof","wc": 2 2 2 2 2 2 2 2 2 2 ...
```

```r
summary(dataset) #showes the quartiles and Median, this is good for skewness
```

```
##    education          income          women           prestige
##  Min.   : 6.380   Min.   :  611   Min.   : 0.000   Min.   :14.80
##  1st Qu.: 8.445   1st Qu.: 4106   1st Qu.: 3.592   1st Qu.:35.23
##  Median :10.540   Median : 5930   Median :13.600   Median :43.60
##  Mean   :10.738   Mean   : 6798   Mean   :28.979   Mean   :46.83
##  3rd Qu.:12.648   3rd Qu.: 8187   3rd Qu.:52.203   3rd Qu.:59.27
##  Max.   :15.970   Max.   :25879   Max.   :97.510   Max.   :87.20
##      census       type
##  Min.   :1113   bc  :44
##  1st Qu.:3120   prof:31
##  Median :5135   wc  :23
##  Mean   :5402   NA's: 4
##  3rd Qu.:8312
##  Max.   :9517
```

Skewness is the degree of distortion from symmetrical normal distribution. If the median is larger than the mean, the variable is left-skewed. If it's lower, then it's right-skewed.

To calculate the skewness we use the Pearson Coefficient. So in addition to the statistical KPIs provided above, we need the standard deviation.

```r
sd(Prestige$education) # 2.728444
```

```
## [1] 2.728444
```

```r
sd(Prestige$income) # 4245.922
```

```
## [1] 4245.922
```

```r
sd(Prestige$women) # 31.72493
```

```
## [1] 31.72493
```

```r
sd(Prestige$prestige) # 17.20449
```

```
## [1] 17.20449
```

```r
sd(Prestige$census) # 2644.993
```

```
## [1] 2644.993
```

Now that we have the standard deviation for each variable, let's continue with calculate the actual Pearson Coefficient to make a more detailed statement about the skewness. In order to do so, we write a function passing the median, mean and standard deviation as a parameter.

```r
calculate_pearson_skewness <- function(median, mean, sd){

  pearson_skewness <- ((median - mean) * 3) / sd
  return(pearson_skewness)
```

```
}

education_pc <- calculate_pearson_skewness(median(Prestige$education), mean(Prestige$education), sd(Pres

income_pc <- calculate_pearson_skewness(median(Prestige$income), mean(Prestige$income), sd(Prestige$inco

women_pc <- calculate_pearson_skewness(median(Prestige$women), mean(Prestige$women), sd(Prestige$women)

prestige_pc <- calculate_pearson_skewness(median(Prestige$prestige), mean(Prestige$prestige), sd(Prestig

census_pc <- calculate_pearson_skewness(median(Prestige$census), mean(Prestige$census), sd(Prestige$cens
```

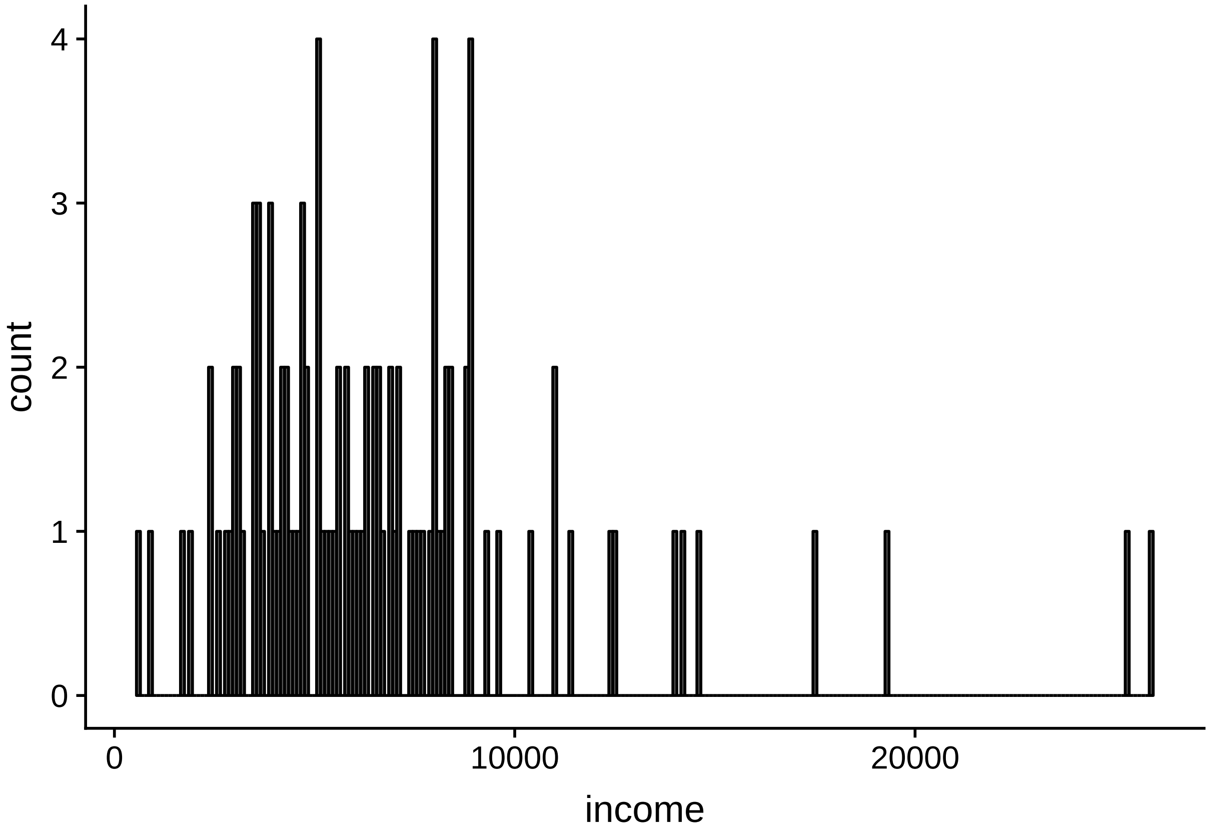
As interpretation of the results, we use the following table:

-0.5 to 0.5: Fairly symmetrical
-1 to -0.5 OR 0.5 to 1: Moderately skewed
less than -1 OR higher than 1: Highly skewed

"education": Fairly symmetrical data
"income": Moderately right-skewed
"women": Highly right-skewed
"prestige": Moderately right-skewed
"census:" Fairly symmetrical

So what does this tell us? Let's take the variable "income" for example. We can now say for certain, that the majority of people have an income below the average! To illustrate this and prove our point, we make a very simply graph to show the distribution of income.

```
ggplot(Prestige, aes(income))+
  geom_histogram(binwidth=100, color="black")
```



```r
anyNA(dataset) #Check if there areany missing Values
```

```
## [1] TRUE
```

```r
anyDuplicated(dataset) #Check if there are duplicated rows in the dataset
```

```
## [1] 0
```

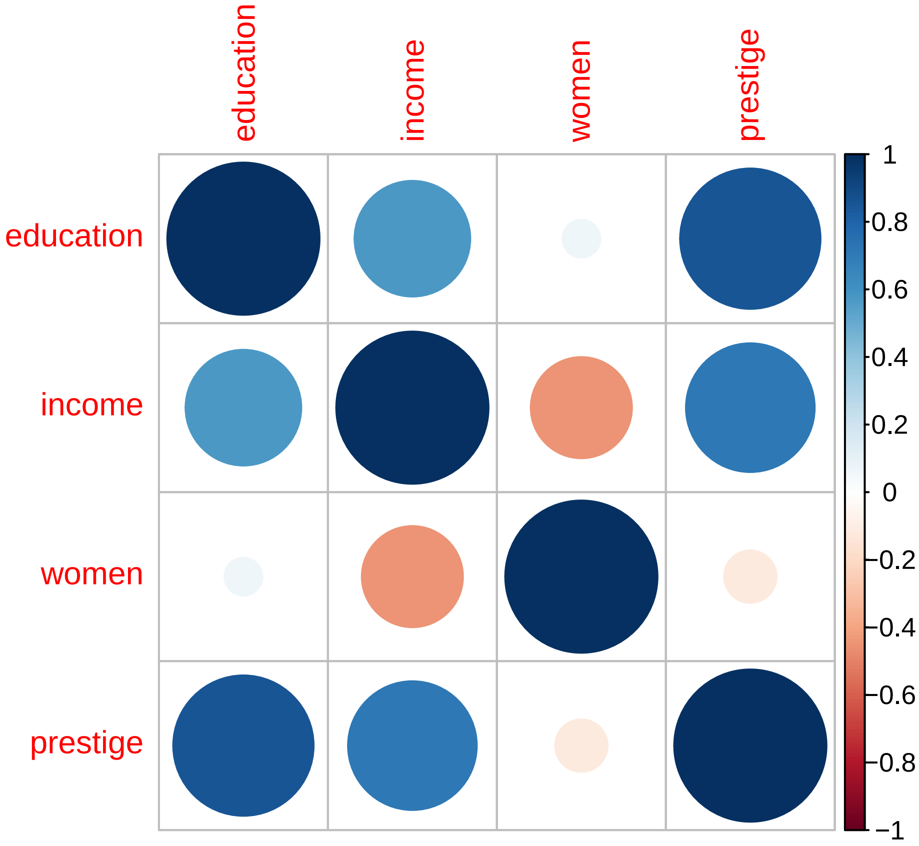
There's data on a total of 102 different jobs. We can notice that the "type" variable has 4 missing values. We have to keep this in mind whilst building our regression model, as long as we want to use the type variable.

Data Visualization

Corrplot

The best and easiest way to indentify the variables that are highly correlated is using a corrplot, short for correlation plot. Attention: Let's keep in mind we have two classes, namely numeric / integer and factor.

```r
corrplot(cor(dataset[,-c(5:6)])) #Let's get rid of the factor 'type'
```



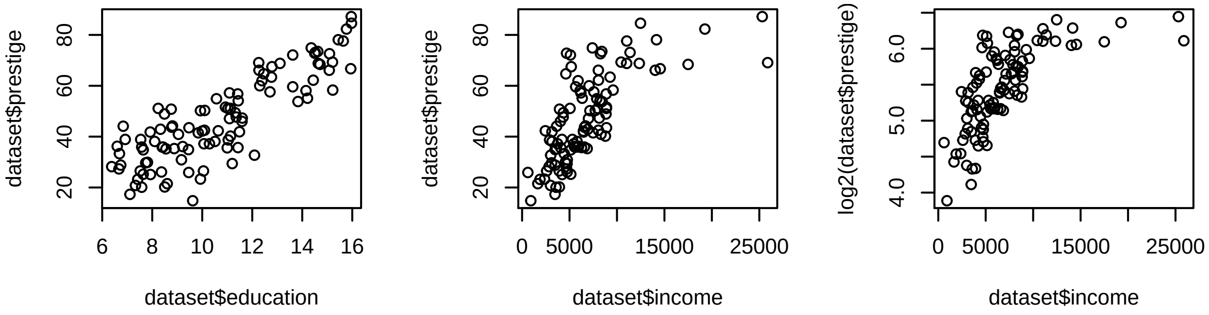
According to the independent Variables, we notice that income and education are highly positively correlated with prestige. When we look at the census, we see that it's negatively correlated with prestige. Further, there is no use for the variable "women" as it shows little to no correlation at all. The higher the correlation, the more accurate a prediction can be in linear regression. Therefore, only the highly correlating variables should be used. But for multiple linear regression another assumption should be fulfilled. The independent variables should have a linear independence. This means that it can be problematic if there is a strong correlation between two or more independent variables. Because then an independent variable can be predicted to a large degree from another variable.

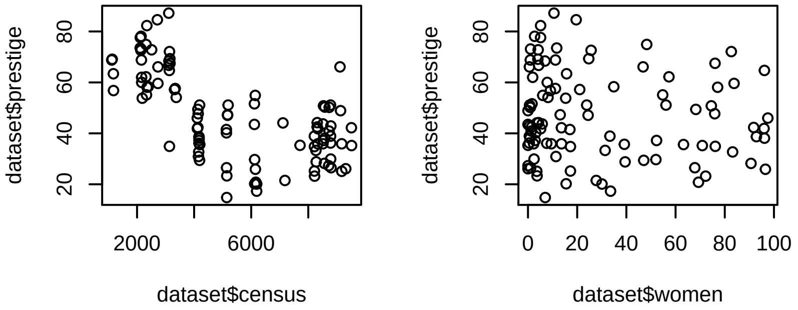
This exploration has provided us great insight into the data, but let's move on to the data preprocessing stage.

Simple Plots

Let's keep it simple and plot two variables against each other. So we can look for outliers and non-linear relationships.

```
# Arrange 4 Plots in 2 rows and 2 columns
par(mfrow=c(2,3))
plot(dataset$education, dataset$prestige)
plot(dataset$income, dataset$prestige)
plot(dataset$income, log2(dataset$prestige))
plot(dataset$census, dataset$prestige)
plot(dataset$women, dataset$prestige)
```

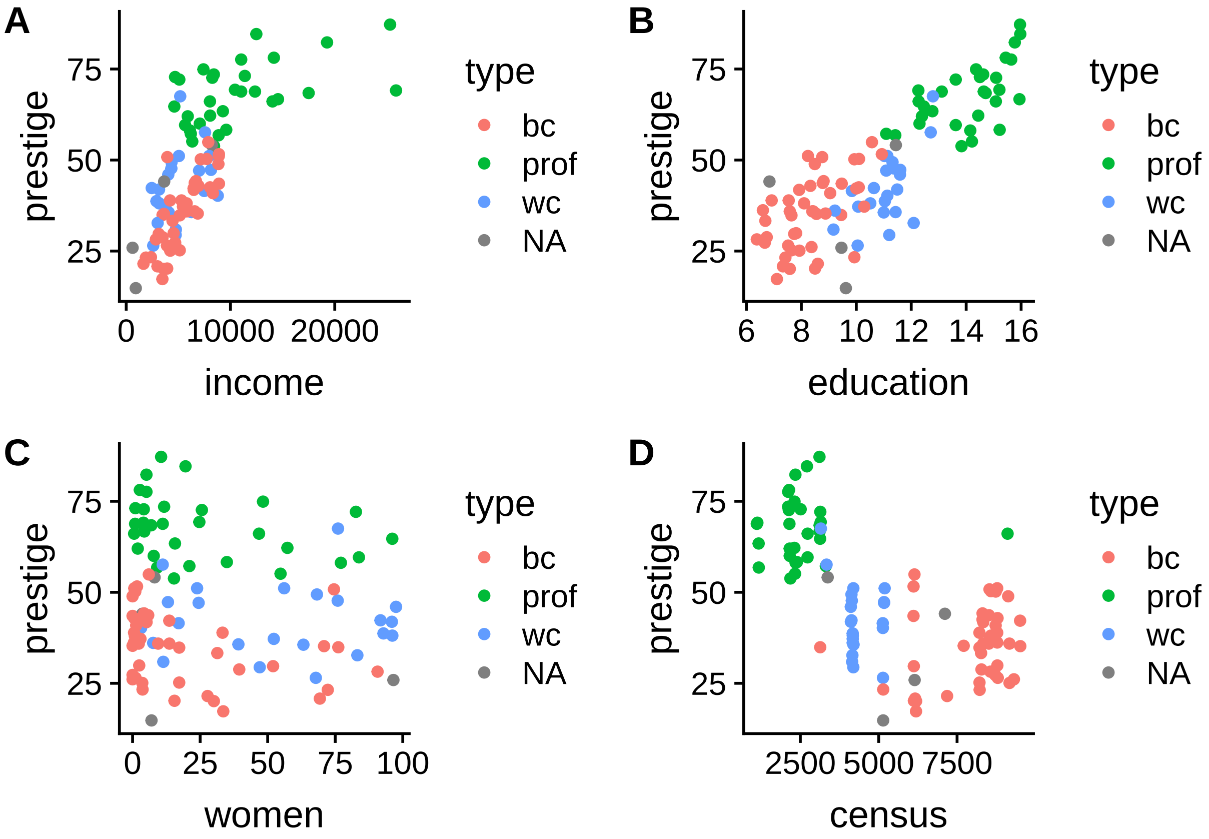




As we can conclude from the five plots, the relationship between education and prestige shares a more linear relationship that income and prestige. The census shows us how much data was collected. Hence it makes no sense to use it in our modelling section.

More detailed ggplot

```
plotEducation <- ggplot(data = dataset, aes(x = education, y = prestige, col = type)) + geom_point()
plotIncome <- ggplot(data = dataset, aes(x = income, y = prestige, col = type)) + geom_point()
plotCensus <- ggplot(data = dataset, aes(x = census, y = prestige, col = type)) + geom_point()
plotWomen <- ggplot(data = dataset, aes(x = women, y = prestige, col = type)) + geom_point()
#Show the 4 Plots in a Grid
plot_grid(plotIncome, plotEducation, plotWomen, plotCensus, labels = "AUTO")
```



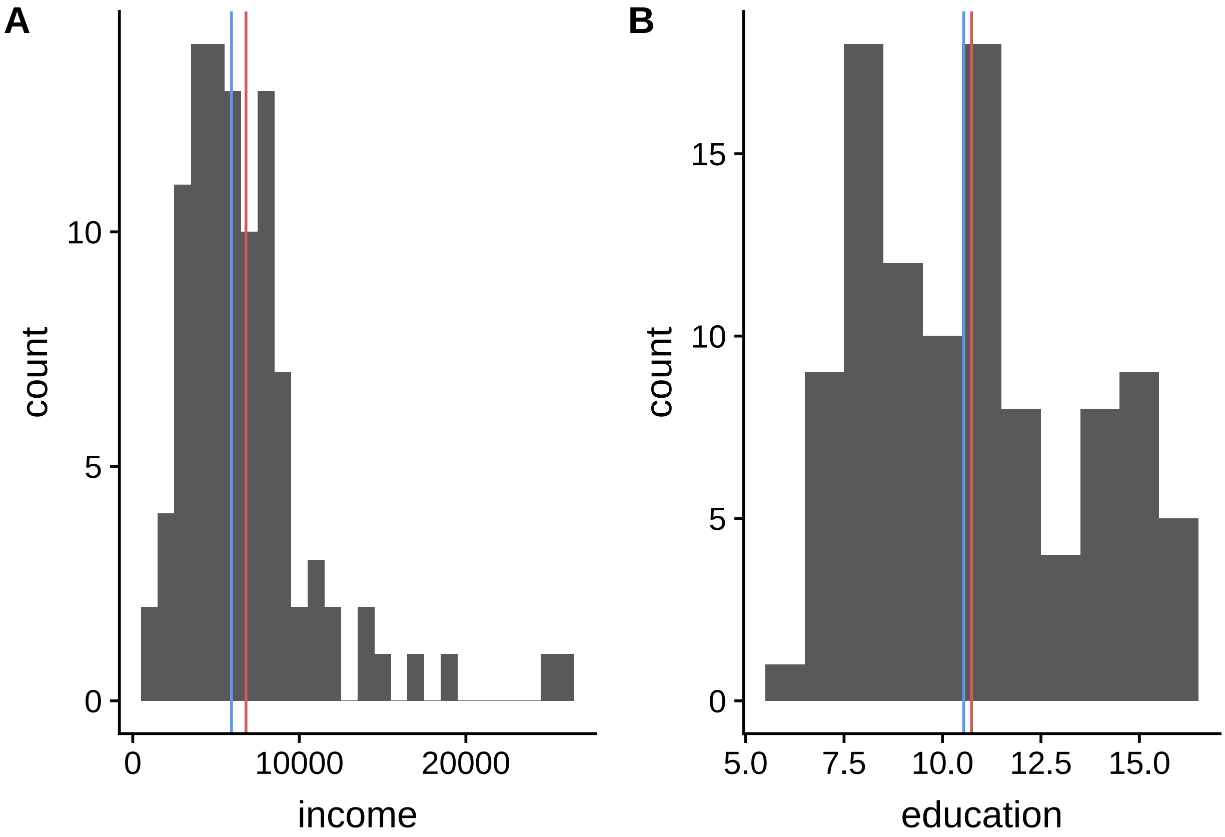
The more detailed ggplot gives us a better insight into the relationship of the "income" and "education" rather than "women" and "census".

Data Preprocessing

Univariate analysis - Histogram

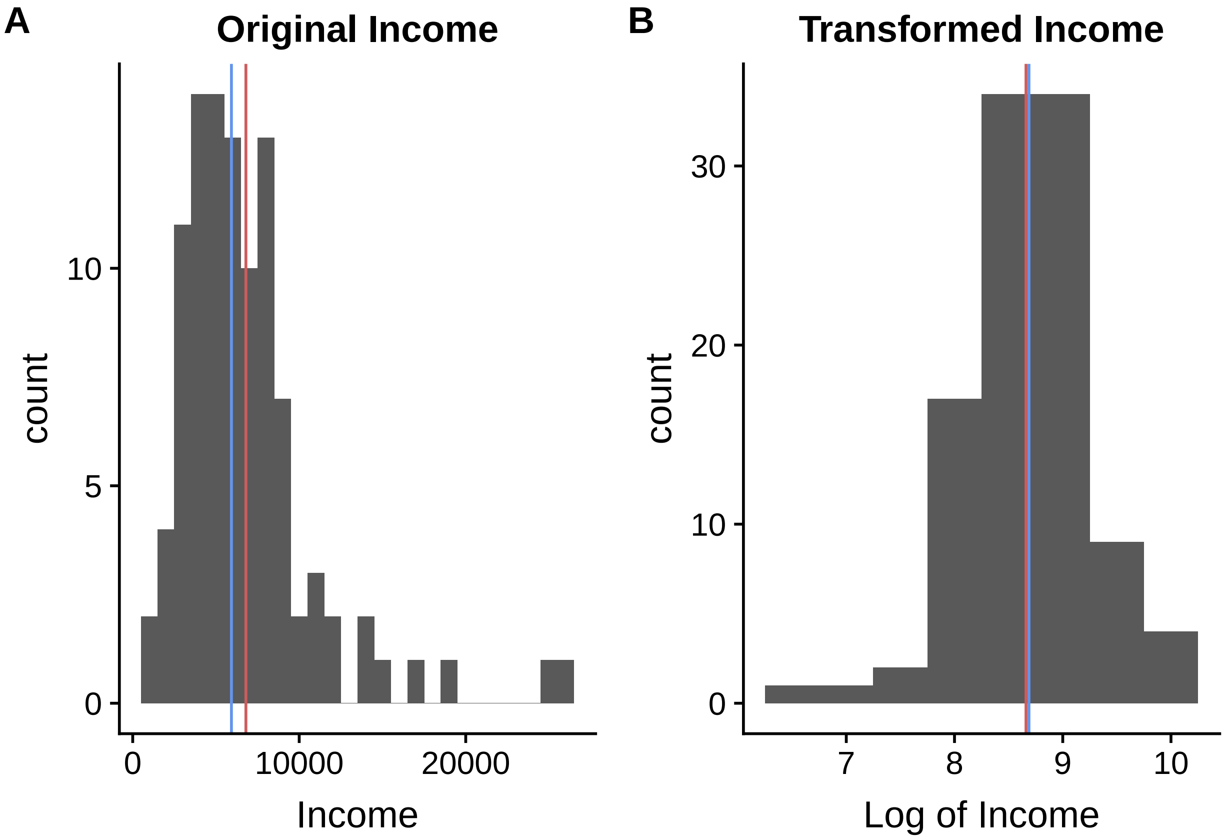
As a next step we'd like to make sure that we check the variable before feeding it into to simple regression algorithm; Thus we are looking for the data distribution of "income" and "education" variables through a histogram with bins and compare it against mean and median values.

```
histIncome <- ggplot(Prestige, aes(x = income)) + geom_histogram(binwidth = 1000) +
  geom_vline(xintercept = mean(Prestige$income), color = "indianred") +
  geom_vline(xintercept = median(Prestige$income), color = "cornflowerblue")
histEducation <- ggplot(Prestige, aes(x = education)) + geom_histogram(binwidth = 1) +
  geom_vline(xintercept = mean(Prestige$education), color = "indianred") +
  geom_vline(xintercept = median(Prestige$education), color = "cornflowerblue")
#Show the two histogram together side by side
plot_grid(histIncome, histEducation, labels = "AUTO")
```

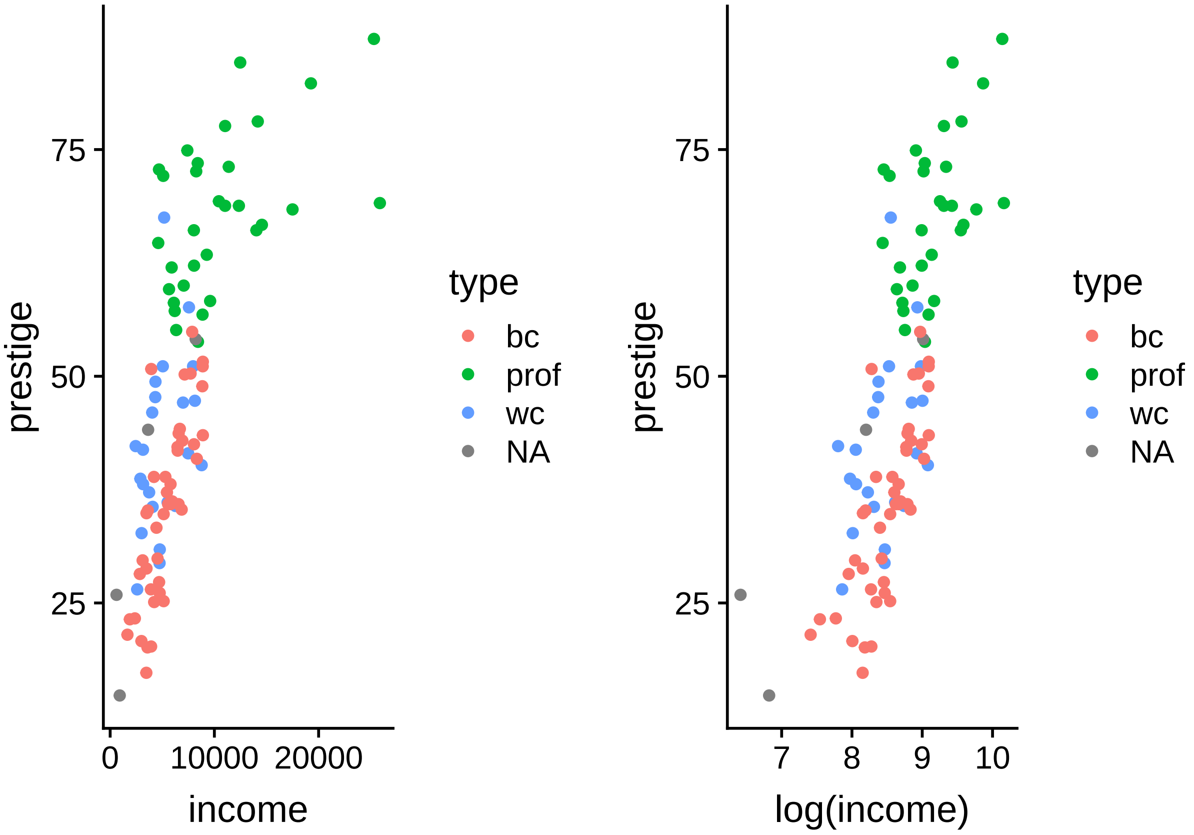


The “income” variable is right skewed and the varibale “education” is not representing normal distribution (also known as Gaussian distribution). To transform this into a normal distribution we are going to use Log2 for the “income” variable and scale the value of the variable on its mean.

```r
# Comparing original income histogram against log of income histogram
hist_income <- ggplot(Prestige, aes(x = income)) + geom_histogram(binwidth = 1000) +
  labs(title = "Original Income") + labs(x ="Income") +
  geom_vline(xintercept = mean(Prestige$income), color = "indianred") +
  geom_vline(xintercept = median(Prestige$income), color = "cornflowerblue")
hist_trnsf_income <- ggplot(Prestige, aes(x = log(income))) + geom_histogram(binwidth = 0.5) +
  labs(title = "Transformed Income") + labs(x ="Log of Income") +
  geom_vline(xintercept = mean(log(Prestige$income)), color = "indianred") +
  geom_vline(xintercept = median(log(Prestige$income)), color = "cornflowerblue")
plot_grid(hist_income, hist_trnsf_income, labels = "AUTO")
```



```
plotlogIncome <- ggplot(data = dataset, aes(x = log(income), y = prestige, col = type)) + geom_point()
plot_grid(plotIncome, plotlogIncome)
```



Remove rows containing na's values

```
dataset <- na.omit(dataset) # remove rows containing NA's values via omit function
```

Data Modelling - Linear Regression to Predict Prestige

Having made all the steps above, we are ready to build a linear regression model. Our approach is applying different regressors step by step and eliminating the variables that are not significant enough to improve the performance of the algorithm. This section shall also show us that our findings from the Data Understanding and Data Visualisation above of "women" and "census" variable are not sharing any relationship with "prestige" attribute. However, adding these minor attributes may give us in the final steps better accuracy, so let's find it out.

Regression Model - RAW

```
# First Raw Regressor without the Factor variable
lm_raw = lm(prestige ~ education + log(income) + women , data = dataset)
summary(lm_raw) # run a summary of its results
```

```
##
## Call:
## lm(formula = prestige ~ education + log(income) + women, data = dataset)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -16.8202  -4.7019   0.0696   4.2245  17.6833 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -129.16790   18.95716  -6.814 8.97e-10 ***
## education      3.59404    0.38431   9.352 4.39e-15 ***
## log(income)   15.60547    2.43246   6.416 5.62e-09 ***
## women          0.06481    0.03270   1.982   0.0504 .  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 6.828 on 94 degrees of freedom
## Multiple R-squared:  0.8454, Adjusted R-squared:  0.8405 
## F-statistic: 171.4 on 3 and 94 DF,  p-value: < 2.2e-16
```

By looking at our first raw concatenated algorithm, we notice that the adjusted R2 is 82% which is good. Nevertheless, the p-values from “women” is high. For the p-Value we can examine a linear model to be statistically significant only when the pre-determined statistical significance level of 0.05. When reflecting this at the “income” and “education”; We can see that they have very considerable p-values; Hence they are playing the most important rule in our regressor. For our second step, we are going to get rid of “women” and “census” variables. Then we build a new model and again check its accuracy.

Regression Model - Significant Variables

```
# Fit a linear model with education and income variables
lm_modified1 = lm(prestige ~ education + log(income), data = dataset)
summary(lm_modified1) # run a summary of its results
```

```
## 
## Call:
## lm(formula = prestige ~ education + log(income), data = dataset)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -16.6775  -4.0506  -0.2538   4.0648  16.7992 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -101.1882    12.8490  -7.875 5.51e-12 ***
## education      4.0375     0.3173  12.726  < 2e-16 ***
## log(income)   12.0564     1.6719   7.211 1.33e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 6.932 on 95 degrees of freedom
## Multiple R-squared:  0.8389, Adjusted R-squared:  0.8356 
## F-statistic: 247.4 on 2 and 95 DF,  p-value: < 2.2e-16
```

Little has changed in the adjusted R2. However the can see that the intercept went down significantly. The value is now at -95. Imagine that now our linear regressor starts 95 below the coordinate system. It is hard to overlook that this model makes sense. It is hard to prove whether this model makes sense or not. To address this issue, we will create a new variable called “education.c” where we centre the value of “education” on its mean. Therefore, the mean of "education" becomes the new zero. This transformation should bring us a more meaningful interpretation of its intercept estimate.

This link showes how z transformation in R works: http://www.statistics4u.info/fundstat_germ/ee_ztransform.html http://rtutorialseries.blogspot.com/2012/03/r-tutorial-series-centering-variables.html

Center Varibles and Create Train and Test

```
prstg_df = dataset # creating a new dataset copy of Prestige or other manipulations

# scaling the value of education to its mean value
set.seed(1)
education.c = scale(prstg_df$education, center=TRUE, scale=FALSE)
prstg_df = cbind(prstg_df, education.c)
```

As a next step (this could also be done in the data preprocessing part) we are splitting our prestaged dataset into Train and Test set. The SplitRatio is high at 95% so that we have around 5 testing values.

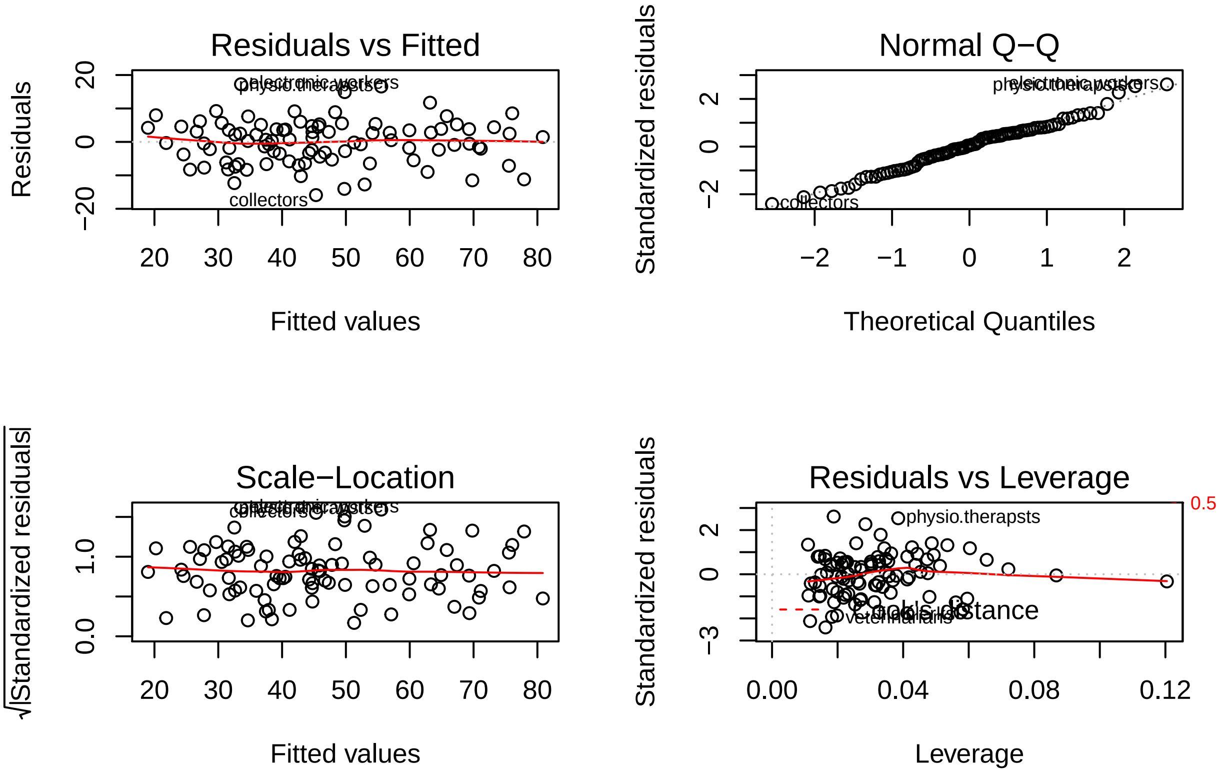
```
# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(prstg_df$prestige, SplitRatio = 0.95)
training_set = subset(prstg_df, split == TRUE)
test_set = subset(prstg_df, split == FALSE)
```

Regression Model - Education Centred

```
# Fit a linear model with centered education and income variables
lm_modified3 = lm(prestige ~ education.c + log(income), data = training_set)
summary(lm_modified3) # run a summary of its results
```

```
##
## Call:
## lm(formula = prestige ~ education.c + log(income), data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.911  -4.422   0.296   4.212  17.241
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -64.6588    14.6362  -4.418 2.77e-05 ***
## education.c   3.8475     0.3193  12.049  < 2e-16 ***
## log(income)  12.8086     1.6796   7.626 2.39e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.664 on 90 degrees of freedom
## Multiple R-squared:  0.8422, Adjusted R-squared:  0.8387
## F-statistic: 240.1 on 2 and 90 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2, 2))  # Split the plotting panel into a 2 x 2 grid
plot(lm_modified3)  # Plot the income model information
```



To better analyse our model we plot residual values of our last regressor. The residuals are the difference between the actual and the predicted values. Let's take some time to refresh on what kind of properties a residual plot should have:

1. They're symmetrically distributed, tending to cluster towards the middle of the plot
2. They're gathered around the lower single digits of the y-axis (e.g., 0.5 or 1.5, not 30 or 150)
3. In general, there aren't clear patterns

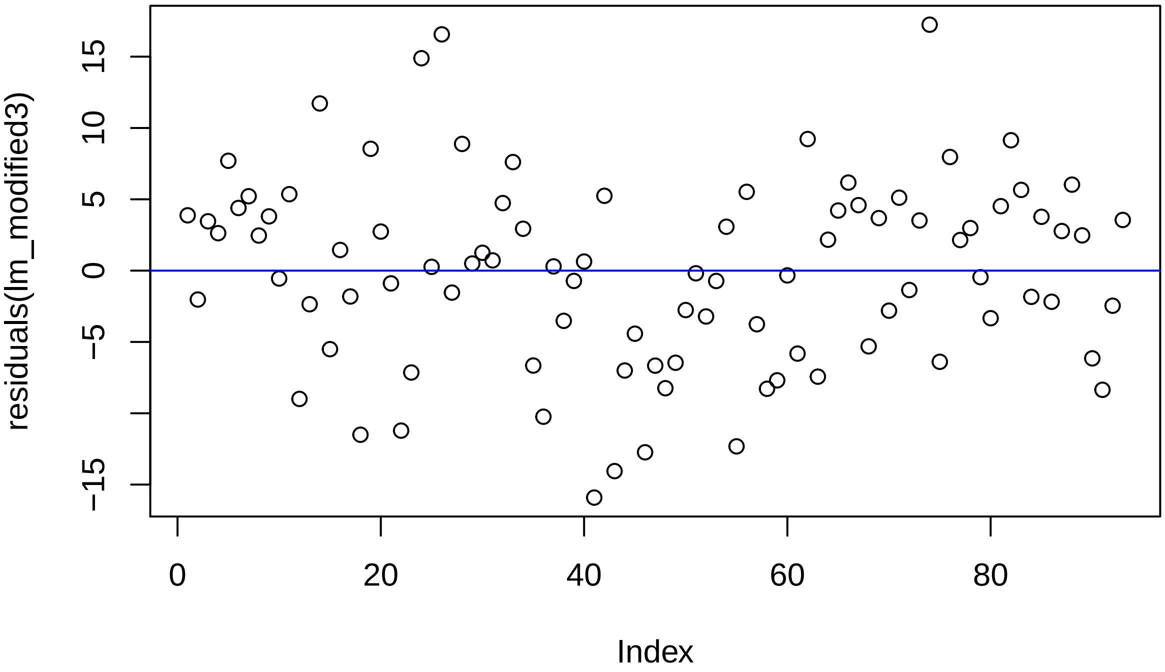
From the above summary, we can conclude that residuals are ranging from -17 to 18 and you can see them on the Q-Q plot that our data are evenly distributed. A Q-Q plot is a scatterplot which plots two sets of quantiles against one another. Q-Q plots take your sample data, sort it in ascending order, and then plot them versus quantiles calculated from a theoretical distribution. The number of quantiles is selected to match the size of your sample data.

Residual Plots

```
# Compare Actual, Predicted and Residual values of prestige from Education model
prstg_df_pred = as.data.frame(training_set$prestige) # Save the actual values
prstg_df_pred$predicted <- predict(lm_modified3) # Save the predicted values
prstg_df_pred$residuals <- residuals(lm_modified3) # Save the residual values
head(prstg_df_pred)
```

```
##   training_set$prestige predicted residuals
## 1                  68.8  64.92386  3.876138
## 2                  69.1  71.12794 -2.027942
## 3                  63.4  59.94162  3.458380
## 4                  56.8  54.17392  2.626077
## 5                  73.5  65.80038  7.699624
## 6                  77.6  73.20915  4.390851
```

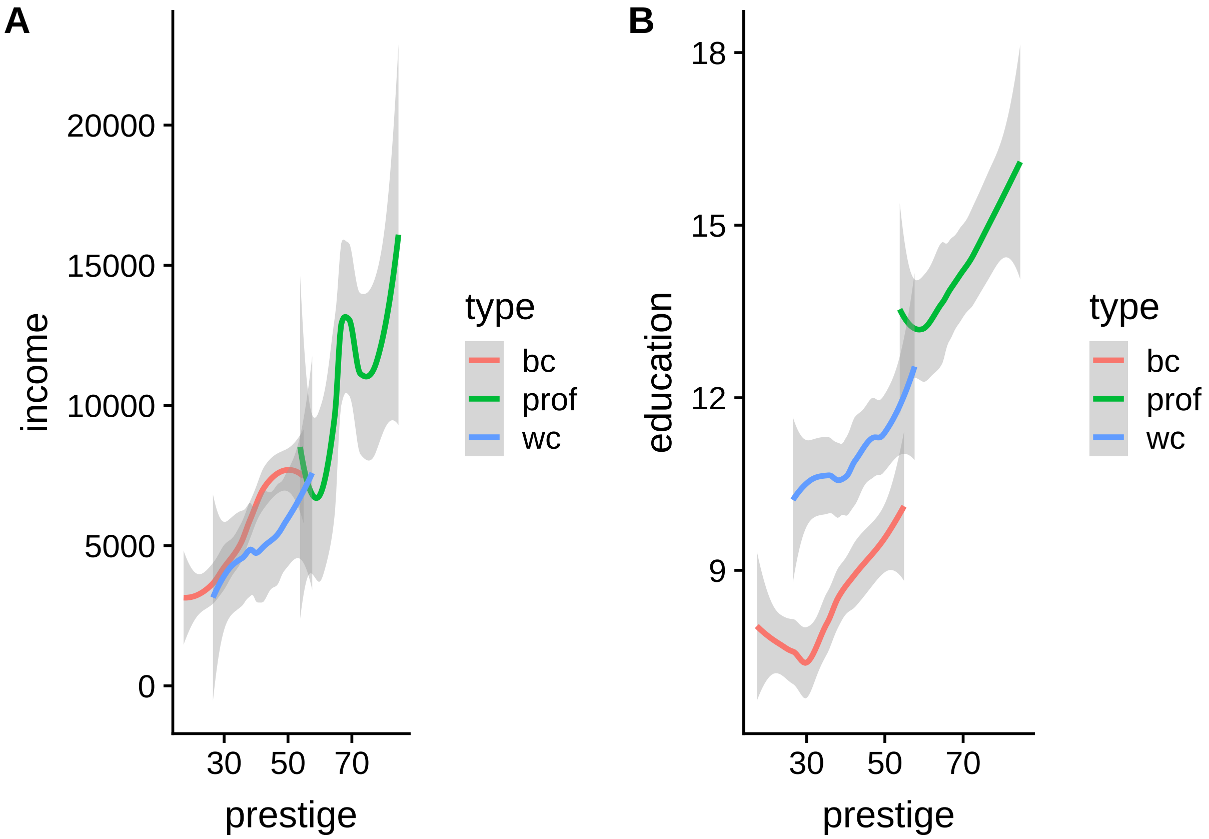
```
plot(residuals(lm_modified3)) # Residual distribution of the model
abline(a=0,b=0,col='blue')
```



Now, let’s see if we can further improve the model. If you remember, above we had done scatterplot for “income” and “education” against “prestige” with “type” variable as category and you have seen that for each category the linearity is different. Let’s look at that from a different perspective. But before that, let’s handle the NA values in “type”variable.

```
ggplot_income <- ggplot(data = training_set, aes(x = prestige, y = income, col = type)) + geom_smooth()
ggplot_educ <- ggplot(data = training_set, aes(x = prestige, y = education, col = type)) + geom_smooth(
plot_grid(ggplot_income, ggplot_educ, labels = "AUTO")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The regressor looks good but let's think about how we can improve the model? We are going back to the detailed ggplots where we included the types to give us nice boundaries to look at. This actually shows us that each and every type is different. So let us integrate the factors and examine whether the model has improved.

Linear Regressor - Type Factor Included

```
# Fit a linear model with centered education, log of income and type variables
lm_modified4 = lm(prestige ~ education.c + log(income) + type, data = training_set)
summary(lm_modified4) # run a summary of its results
```

```
##
## Call:
## lm(formula = prestige ~ education.c + log(income) + type, data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.977  -4.132   1.124   4.072  16.806
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -51.4778    15.1137  -3.406 0.000995 ***
## education.c   3.0110     0.6016   5.005 2.84e-06 ***
## log(income)  11.0635     1.7094   6.472 5.31e-09 ***
## typeprof      7.4346     3.5700   2.083 0.040195 *
## typewc       -1.3734     2.2811  -0.602 0.548662
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.316 on 88 degrees of freedom
## Multiple R-squared:  0.8614, Adjusted R-squared:  0.8551
## F-statistic: 136.7 on 4 and 88 DF,  p-value: < 2.2e-16
```

We can notice that R2 has increased to 85% and the residuals are ranging from -13 to 17.

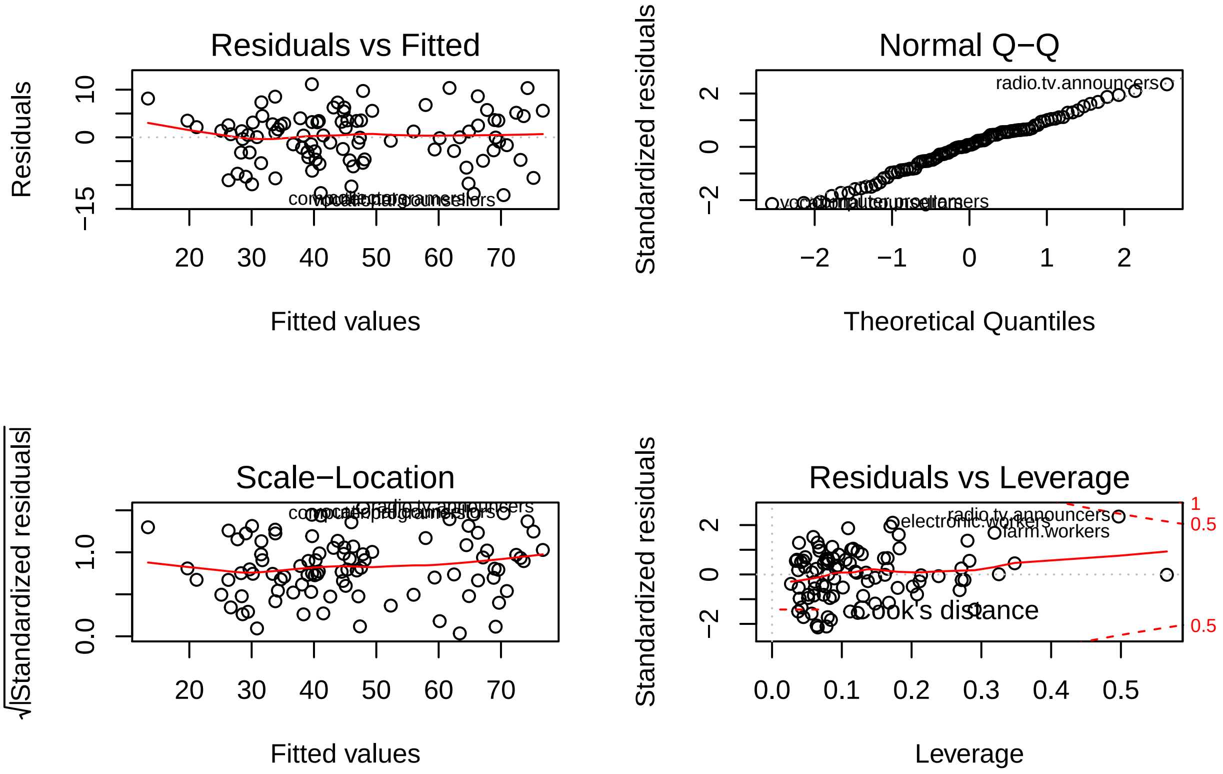
Linear Regressor - Adding Variables Another Way + Women

Here we are just adding the variable type in another way to the regressor model. As a final step we included the attribute "women" to see if it could make our model even more accurate. Adding the "women" into the brackets gives us a improvement of around 2%.

```
lm_modified5 = lm(prestige ~ type * (education.c + log2(income) + women) , data = training_set)
summary(lm_modified5) # run a summary of its results
```

```
##
## Call:
## lm(formula = prestige ~ type * (education.c + log2(income) +
##     women), data = training_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.1160  -3.1987   0.4034   3.5185  11.1381
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -146.03410   27.11441  -5.386 6.89e-07 ***
## typeprof               139.94307   49.60075   2.821  0.00601 **
## typewc                   2.57125   77.84937   0.033  0.97373
## education.c              2.70895    0.86982   3.114  0.00255 **
## log2(income)            15.07398    2.10467   7.162 3.27e-10 ***
## women                    0.14965    0.04617   3.241  0.00173 **
## typeprof:education.c     0.03453    1.22615   0.028  0.97760
## typewc:education.c      -0.60501    2.14849  -0.282  0.77897
## typeprof:log2(income)  -10.21642    3.74972  -2.725  0.00789 **
## typewc:log2(income)     -0.68329    5.99782  -0.114  0.90958
## typeprof:women          -0.14642    0.07667  -1.910  0.05970 .
## typewc:women             0.01993    0.10804   0.184  0.85410
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.847 on 81 degrees of freedom
## Multiple R-squared:  0.8906, Adjusted R-squared:  0.8758
## F-statistic: 59.97 on 11 and 81 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2, 2))
plot(lm_modified5)
```

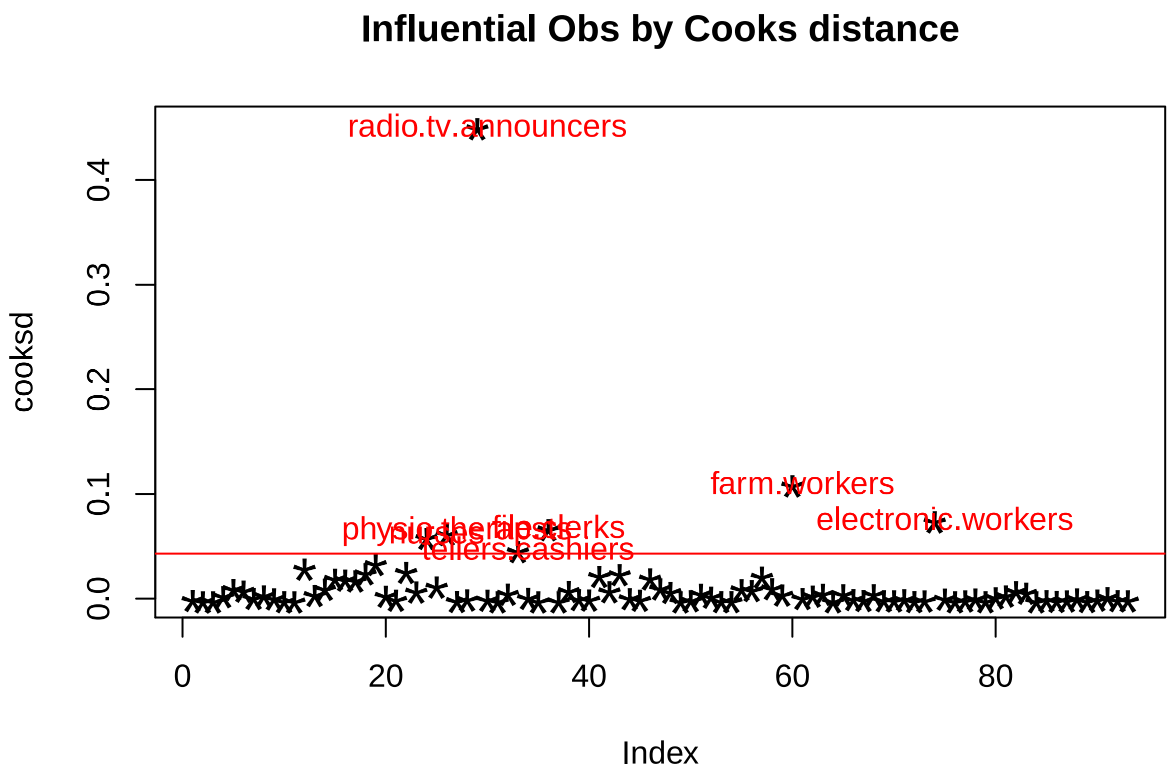


Our model provides us now with an accuracy of 87.5 %, so we can conclude that including the type increases the predictor, as well as the attribute "women" in our final model. In the end, we were playing around with the brackets and improved our model (increase of prediction by roughly 2.5%).

Outlier Detection - Cooks distance

An additional input after today's input from the Professor.

```
cooksd <- cooks.distance(lm_modified5)
sample_size <- nrow(training_set)
plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")# plot cook's distance
abline(h = 4/sample_size, col="red")  # add cutoff line
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4/sample_size, names(cooksd),""), col="red")
```



```
# Detecting outliers in cars dataset;
fit<- lm(prestige ~ type * (education.c + log2(income) + women) , data = training_set)
training_set$cooksd <- cooks.distance(fit);
# Defining outliers based on 4/n criteria;
training_set$outlier <- ifelse(training_set$cooksd < 4/nrow(training_set), "keep","delete")
training_set_cleared <- subset(training_set, outlier=='keep')

lm_modified6<- lm(prestige ~ type * (education.c + log2(income) + women) , data = training_set_cleared)

summary(lm_modified6)
```

```
##
## Call:
## lm(formula = prestige ~ type * (education.c + log2(income) +
##     women), data = training_set_cleared)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.7762  -3.1899   0.8667   3.2715  11.1461
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -167.58150   29.12506  -5.754 1.85e-07 ***
## typeprof              166.70680   47.12192   3.538 0.000701 ***
## typewc                -73.87719   93.51135  -0.790 0.432032
## education.c             2.11739    0.80360   2.635 0.010246 *
## log2(income)           16.68574    2.25647   7.395 1.79e-10 ***
## women                   0.13745    0.04658   2.951 0.004240 **
## typeprof:education.c    1.08128    1.12790   0.959 0.340848
## typewc:education.c     -3.65712    2.98265  -1.226 0.224036
## typeprof:log2(income)  -12.23846    3.57224  -3.426 0.001003 **
## typewc:log2(income)      5.06905    7.16321   0.708 0.481386
## typeprof:women          -0.21908    0.07720  -2.838 0.005859 **
## typewc:women             0.17770    0.12826   1.385 0.170079
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.198 on 74 degrees of freedom
## Multiple R-squared:  0.9144, Adjusted R-squared:  0.9016
## F-statistic: 71.84 on 11 and 74 DF,  p-value: < 2.2e-16
```

Backward Elimination

Here is a quick test of building a model with Backward Elimination. This algorithm can be summarized by five steps:

1. Step: Select a significance level to stay in the model (e.g. SL= 0.05)
2. Step: Fit the model with all possible predictors
3. Step: Consider the predictor with the highest P-value. if P > SL, go to step 4, otherwhise go to END
4. Step: Remove the predictor
5. Step: Fit the model without this variable –> goto Step 3

```
#Function backwarElimination
backwardElimination <- function(x, sl) {
    numVars = length(x)
    for (i in c(1:numVars)){
      regressorBE = lm(formula = prestige ~ ., data = x)
      maxVar = max(coef(summary(regressorBE))[c(2:numVars), "Pr(>|t|)"])
      if (maxVar > sl){
        j = which(coef(summary(regressorBE))[c(2:numVars), "Pr(>|t|)"] == maxVar)
        x = x[, -j]
      }
      numVars = numVars - 1
    }
    return(summary(regressorBE))
  }
#Call function + declate variables
  SL = 0.001
  x <- select (training_set_cleared,-c(education, census, type, outlier, cooksd))
  backwardElimination(x, SL)
```

```
##
## Call:
## lm(formula = prestige ~ ., data = x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.3792  -3.8796   0.2193   5.2683  13.5911
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.595e+01  1.793e+00  20.046  < 2e-16 ***
## income      1.525e-03  2.299e-04   6.632 3.14e-09 ***
## education.c 3.945e+00  3.269e-01  12.069  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.599 on 83 degrees of freedom
## Multiple R-squared:  0.8452, Adjusted R-squared:  0.8415
## F-statistic: 226.6 on 2 and 83 DF,  p-value: < 2.2e-16
```

With the Backward Elimination approach we ended up that there are just one, or two possible predictors, namely the income and the centered education. However we are not sure if this automatic function worked entierly corretly.

Evaluation

This is our final section where we feed the algorithm with our Test set and perform K-Fold.

K-Fold Cross Validation

The problem of the validation set approach is that the test error estimation depends on the selection of test sets. Thus, the model has low robustness. The test set might contain outliers which influence the result.

In order to reach a higher robustness, we use the k-fold cross validation method which evaluates the model performance on different subset of the training data and then calculate the average prediction error rate. The algorithm is as follow:

1. Randomly split the data set into k-subsets (or k-fold) (for example 5 subsets)
2. Reserve one subset and train the model on all other subsets
3. Test the model on the reserved subset and record the prediction error
4. Repeat this process until each of the k subsets has served as the test set.
5. Compute the average of the k recorded errors. This is called the cross-validation error serving as the performance metric for the model.

We set k = 5. In practice, one typically performs k-fold cross-validation using k = 5 or k = 10, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.

In R it looks like this. We took the lm_modified3 model which has the most significance.

```
model_cv <- train(prestige ~ education.c + log(income), training_set,
                  method = "lm",
                  trControl = trainControl(
                    method = "cv", number = 5,
                    verboseIter = TRUE
                  ))
```

```
## + Fold1: intercept=TRUE
## - Fold1: intercept=TRUE
## + Fold2: intercept=TRUE
## - Fold2: intercept=TRUE
## + Fold3: intercept=TRUE
## - Fold3: intercept=TRUE
## + Fold4: intercept=TRUE
## - Fold4: intercept=TRUE
## + Fold5: intercept=TRUE
## - Fold5: intercept=TRUE
## Aggregating results
## Fitting final model on full training set
```

```r
print(model_cv)
```

```
## Linear Regression
##
## 93 samples
##  2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 75, 75, 75, 74, 73
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   6.753576  0.8497972  5.5002
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
summary(model_cv)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.911  -4.422   0.296   4.212  17.241
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -64.6588    14.6362  -4.418 2.77e-05 ***
## education.c     3.8475     0.3193  12.049  < 2e-16 ***
## `log(income)`  12.8086     1.6796   7.626 2.39e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.664 on 90 degrees of freedom
## Multiple R-squared:  0.8422, Adjusted R-squared:  0.8387
## F-statistic: 240.1 on 2 and 90 DF,  p-value: < 2.2e-16
```

Here K-Fold Cross Validation is explained in more detail.

We see that not much has changed in the result, but it gives us confirmation that the model is good fitted.

The Test Set

```r
head(test_set)
```

```
##                     education income women prestige census type
## surveyors               12.39   5902  1.91     62.0   2161 prof
## ministers               14.50   4686  4.14     72.8   2511 prof
## physicians              15.96  25308 10.56     87.2   3111 prof
## medical.technicians     12.79   5180 76.04     67.5   3156   wc
## carpenters               6.92   5299  0.56     38.9   8781   bc
##                     education.c
## surveyors              1.594898
## ministers              3.704898
## physicians             5.164898
## medical.technicians    1.994898
## carpenters            -3.875102
```

Prediction Regressor

In this section we are only going to test the algorithm against our Original Algorithm, Algorithm with type and the final algorithm with the strongest prediction. In y__pred__interval we let the prediction and the 95% confidence interval be displayed. This means we are 95% confident the predicted units will fall between "lwr" und "upr".

Linear Regressor - log2(income) + education.c

```
y_pred = predict(lm_modified3, newdata =test_set)
y_pred
```

```
##           surveyors           ministers          physicians 
##            52.69521            57.85833            85.07791 
## medical.technicians          carpenters 
##            52.56286            30.26897
```

```
y_pred_interval <-predict(lm_modified3, newdata =test_set, interval = "prediction", level = 0.95)
y_pred_interval
```

```
##                          fit      lwr      upr
## surveyors           52.69521 39.34036 66.05006
## ministers           57.85833 44.22353 71.49314
## physicians          85.07791 71.20337 98.95245
## medical.technicians 52.56286 39.15118 65.97455
## carpenters          30.26897 16.78001 43.75792
```

Linear Regressor - education.c + log(income) + type

```
y_pred = predict(lm_modified4, newdata =test_set)
y_pred
```

```
##           surveyors           ministers          physicians 
##            56.82369            60.62443            83.67951 
## medical.technicians          carpenters 
##            47.77642            31.72654
```

```
y_pred_interval <-predict(lm_modified4, newdata =test_set, interval = "prediction", level = 0.95)
y_pred_interval
```

```
##                          fit      lwr      upr
## surveyors           56.82369 43.87241 69.77496
## ministers           60.62443 47.58012 73.66874
## physicians          83.67951 70.47393 96.88508
## medical.technicians 47.77642 34.76719 60.78564
## carpenters          31.72654 18.89554 44.55755
```

Linear Regressor - type * (education.c + log(income) + women)

```r
y_pred = predict(lm_modified5, newdata =test_set)
y_pred
```

```
##           surveyors           ministers          physicians 
##            59.14136            63.32047            79.16593 
## medical.technicians          carpenters 
##            51.19195            30.04009 
```

```r
y_pred_interval <-predict(lm_modified5, newdata =test_set, interval = "prediction", level = 0.95)
y_pred_interval
```

```
##                          fit      lwr      upr
## surveyors           59.14136 45.51031 72.77241
## ministers           63.32047 48.44599 78.19495
## physicians          79.16593 65.50127 92.83060
## medical.technicians 51.19195 38.01127 64.37263
## carpenters          30.04009 17.85466 42.22552
```

```r
# Drop the columns of the dataframe X
library(dplyr)
x <- select (training_set,-c(education, census, cooksd, outlier))
xlog <- x
xlog[, 1] <- log2(x[,1])
```

Test Dataset with Random Forest

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

## 
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
## 
##     margin

## The following object is masked from 'package:dplyr':
## 
##     combine
```

```r
set.seed(123)
regressorRF = randomForest(xlog[,-3],
                           y = xlog$prestige,
                           ntree = 175)
print(regressorRF)
```

```
## 
## Call:
##  randomForest(x = xlog[, -3], y = xlog$prestige, ntree = 175) 
##                Type of random forest: regression
##                      Number of trees: 175
## No. of variables tried at each split: 1
```

```
##
##           Mean of squared residuals: 47.85168
##                     % Var explained: 82.43
```

```
#attributes(regressorRF)
importance(regressorRF)
```

```
##             IncNodePurity
## income           6638.788
## women            2161.744
## type             7333.276
## education.c      7188.359
```

```
#regressorRF$rsq[25]
#regressorRF$rsq[100]
regressorRF$rsq[175] #Accuracy
```

```
## [1] 0.8242723
```

MSS is here too big, the model RF does not fit well but we can see again the importance of the attributes in another way.

```
# Predicting a new result with Random Forest Regression
y_pred = predict(regressorRF, newdata =test_set)
y_pred
```

```
##           surveyors           ministers          physicians
##            68.01540            70.75032            74.29612
## medical.technicians          carpenters
##            55.97409            44.99644
```

Conclusion

With our multivariate model, we were able to improve R2 even further and also lowered the boundaries of the residuals of the sum of squared errors of our original log(income) and education.c model. We suggest the combination of preprocessed variables as well as adding the factor type into it. However, the added attribute "women" showed little increase but was added anyway for safety's sake.

If we had more time, it would be interesting to find out in which types the predictor shows most robust accuracy and how an entirely other algorithm performs, but that might be a task for the future.