

Price and Performance in Central Processing Units

Alan Liu

11/19/2020

Introduction

```
# Load Library
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.0 --

## v ggplot2 3.3.2      v purrr  0.3.4
## v tibble  3.0.3      v dplyr  1.0.2
## v tidyr   1.1.1      v stringr 1.4.0
## v readr   1.3.1      v forcats 0.5.0

## Warning: package 'ggplot2' was built under R version 4.0.3

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()     masks stats::lag()

# install.packages(readr)
library(readr)

# install.packages(car) on a separate R script
library(car)

## Warning: package 'car' was built under R version 4.0.3

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.0.3

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##      recode

## The following object is masked from 'package:purrr':
##
##      some

# install.packages(psych) on a separate R script
library(psych)
```

```
##
## Attaching package: 'psych'

## The following object is masked from 'package:car':
##
##      logit

## The following objects are masked from 'package:ggplot2':
##
##      %+%, alpha

# install.packages('readr','leaps')
library(leaps)

# Import data
cpudata <- read_csv("researchproject_data.csv")

## Parsed with column specification:
## cols(
##   MSRP = col_double(),
##   Year = col_double(),
##   Year_Fixed = col_double(),
##   Benchmark_Result = col_double(),
##   Brand = col_character(),
##   Processor = col_character(),
##   Chipset = col_character()
## )
```

Variables Analysis

MSRP - Currency, numerical variable that displays the original price a chip was marketed for

Year - Date, numerical variable that is the original year of release for the chip

Benchmark_Result - Rating, numerical variable that calculates the performance of a chip

Brand - Name, categorical variable that displays the branding of a chip

Processor - Name, categorical variable that displays the associated chip

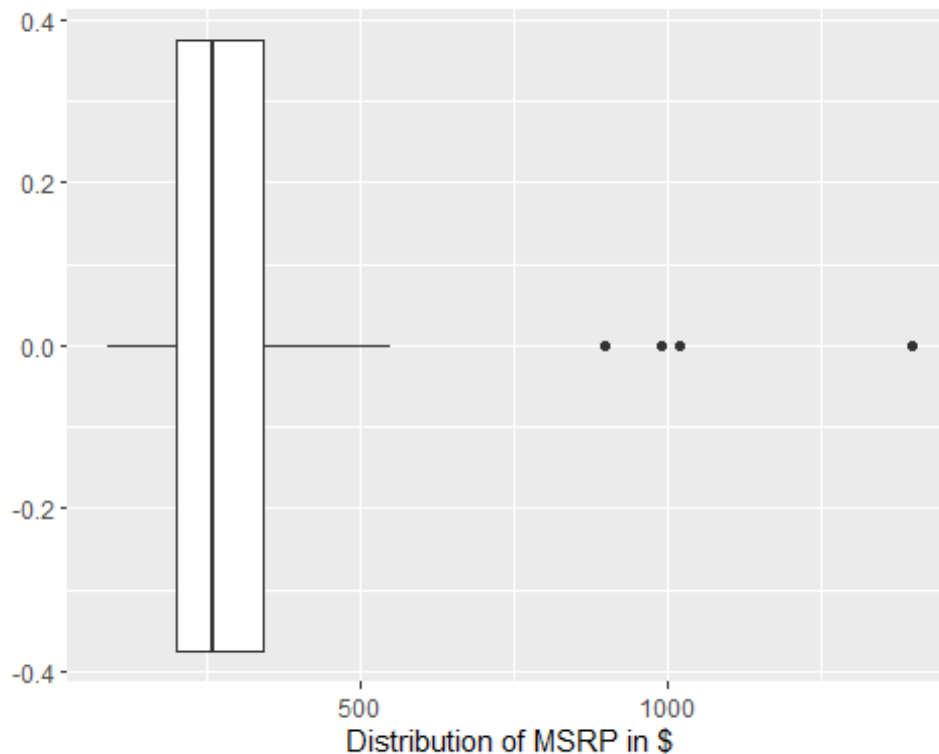
Chipset - Name, categorical variable that displays the chipset the processor was built on

```
# Univariate Stats
summary.extended <- cpudata %>%
  select(MSRP,Year,Benchmark_Result) %>%
  psych::describe(fast = TRUE) %>%
  as_tibble(rownames="rowname") %>%
  print(summary.extended)

## # A tibble: 3 x 9
##   rowname      vars      n  mean    sd  min  max range      se
##   <chr>      <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 MSRP          1    61  313.  234.    88 1399  1311  30.0
## 2 Year          2    61 2016.   2.71 2010 2020    10  0.346
## 3 Benchmark_Result 3    61 11611. 9255.  1792 64205 62413 1185.
```

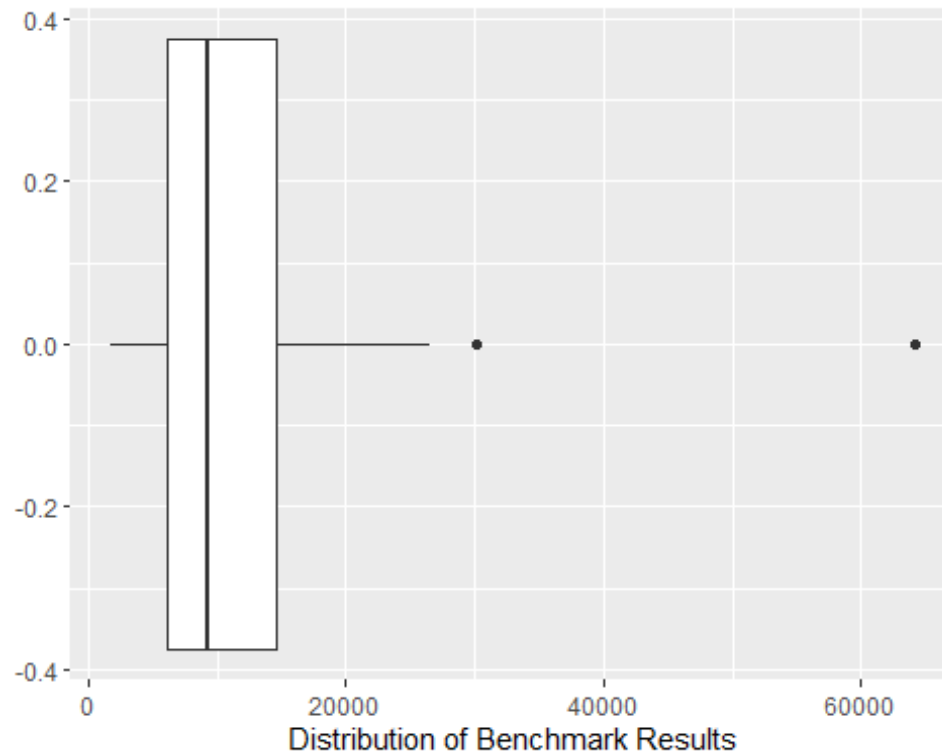
Variable Regression Analysis

```
# Visualize the predictor variable
# Creates the box plot for MSRP
ggplot(cpudata, aes(y=MSRP)) +
  geom_boxplot() + coord_flip() +
  labs(y = "Distribution of MSRP in $")
```



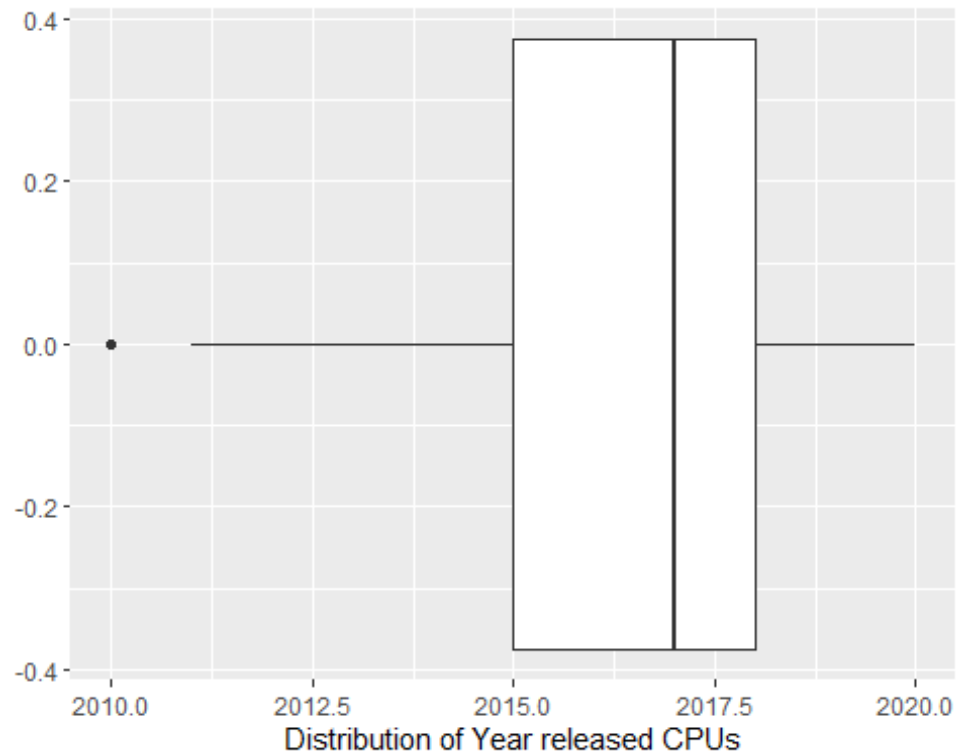
This box plot shows a distribution of the MSRP across the 60 CPUs in the data.

```
# Visualize the predictor variable
# Creates the box plot for Benchmark_Results
ggplot(cpudata, aes(y=Benchmark_Result)) + geom_boxplot() + coord_flip() +
  labs(y = "Distribution of Benchmark Results")
```



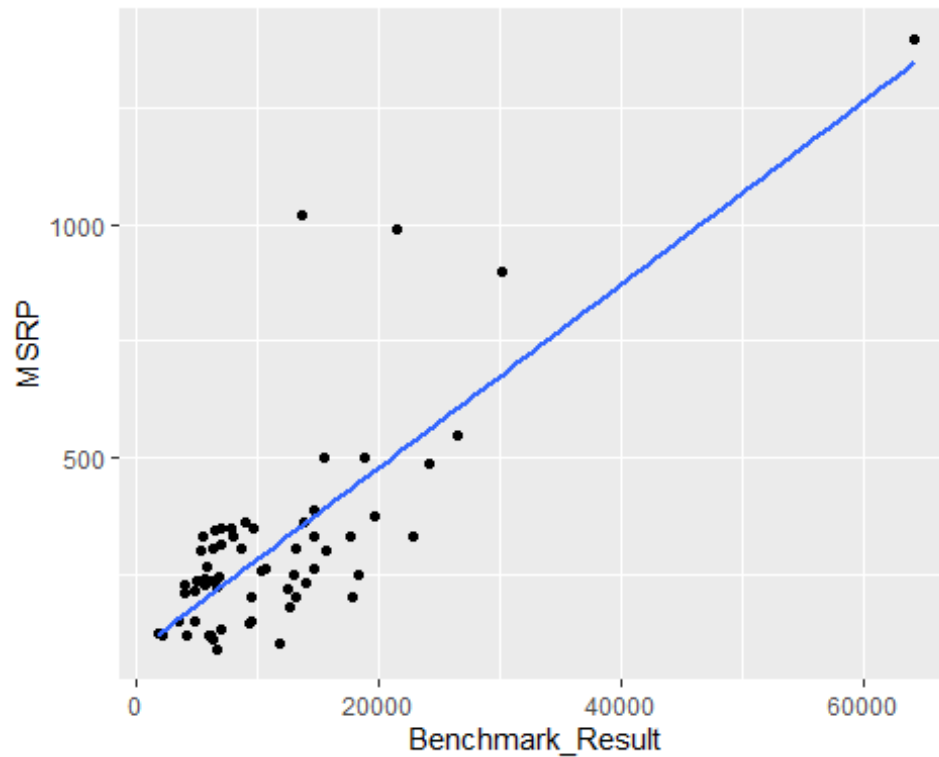
This box plot shows a distribution of the Benchmark Results for all the CPUs. The higher the score, the better performance the CPU has.

```
# Visualize the predictor variable  
# Creates the box plot for Year  
ggplot(cpdata, aes(y=Year)) + geom_boxplot() + coord_flip() +  
  labs(y = "Distribution of Year released CPUs")
```



This box plot shows a distribution of the years released for all the CPUs.

```
# Visualize the data with the regression line
# Creates the scatterplot for MSRP
ggplot(cpudata, aes(x=Benchmark_Result, y=MSRP)) +
  geom_point() +
  geom_smooth(method='lm', se = FALSE)
## `geom_smooth()` using formula 'y ~ x'
```

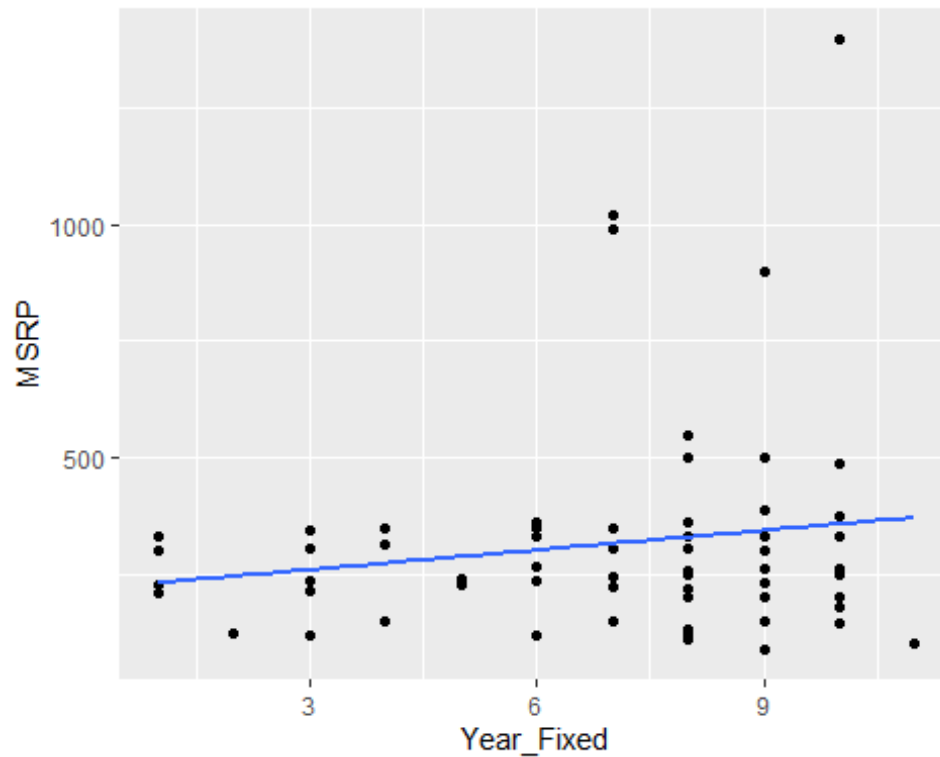


```
# Summary statistics : Correlation coefficient
# Calculates the correlation coefficient for Air Permeability
cor(cpdata$MSRP,cpdata$Benchmark_Result)

## [1] 0.7791367

# Visualize the data with the regression line
# Creates the scatterplot for Year
ggplot(cpdata, aes(x=Year_Fixed, y=MSRP)) +
  geom_point() +
  geom_smooth(method='lm', se = FALSE)

## `geom_smooth()` using formula 'y ~ x'
```

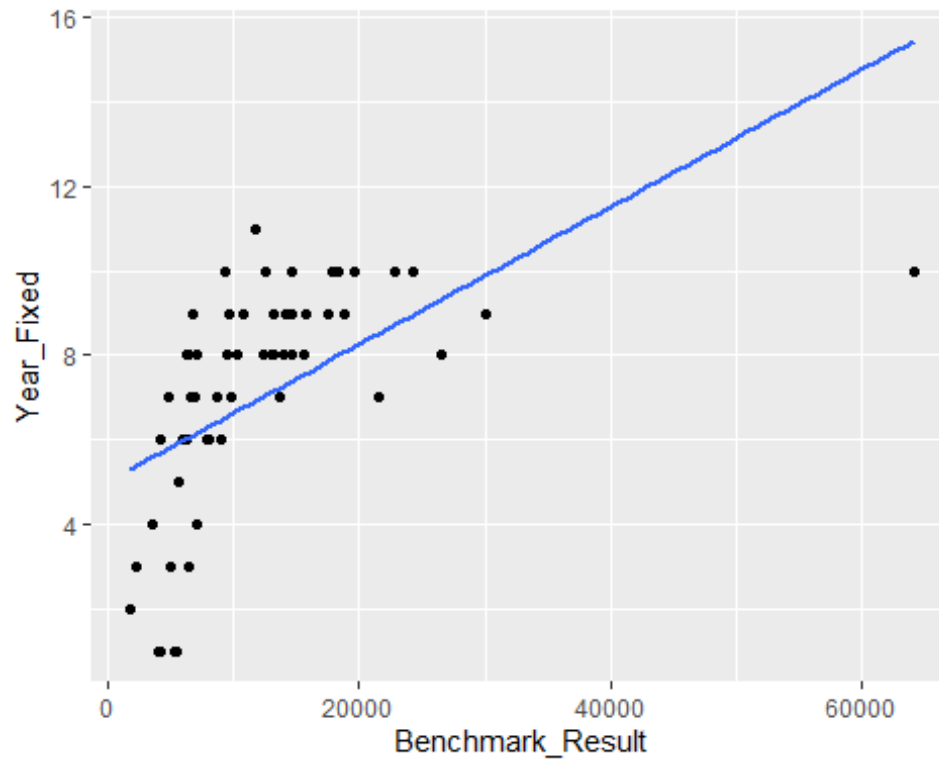


```
# Summary statistics : Correlation coefficient
# Calculates the correlation coefficient for Air Permeability
cor(cpdata$MSRP,cpdata$Year_Fixed)

## [1] 0.1625529

# Visualize the data with the regression line
# Creates the scatterplot for Predictors
ggplot(cpdata, aes(x=Benchmark_Result, y=Year_Fixed)) +
  geom_point() +
  geom_smooth(method='lm', se = FALSE)

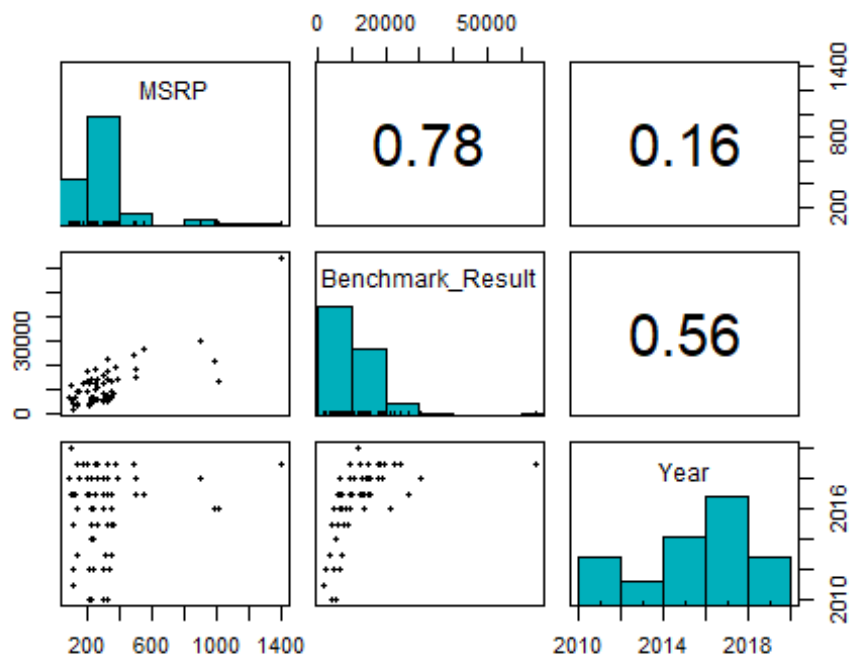
## `geom_smooth()` using formula 'y ~ x'
```



```
# Summary statistics : Correlation coefficient
# Calculates the correlation coefficient for Air Permeability
cor(cpudata$Year_Fixed,cpudata$Benchmark_Result)

## [1] 0.5551998

# A fancy scatterplot matrix
pairs.panels(cpudata[c("MSRP","Benchmark_Result","Year")],
method = "pearson", # correlation method
hist.col = "#00AFBB", # color of histogram
smooth = FALSE, density = FALSE, ellipses = FALSE)
```

Graphs indicate the correlation coefficients among all three variables and provide various plots that indicate point distributions (via histograms and scatterplots).

Model Building Strategy

```
# Fit the regression model with 1 predictor, Benchmark Results
reg <- lm(MSRP ~ Benchmark_Result + Year_Fixed, cpudata)
```

```
# Display the summary table for the regression model
summary(reg)
```

```
##
## Call:
## lm(formula = MSRP ~ Benchmark_Result + Year_Fixed, data = cpudata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -166.93  -63.79  -23.16   40.69   658.42
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    253.73198    45.18730     5.615 5.82e-07 ***
## Benchmark_Result  0.02517     0.00214    11.763 < 2e-16 ***
## Year_Fixed     -33.74488     7.31845    -4.611 2.25e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 127.6 on 58 degrees of freedom
## Multiple R-squared:  0.7125, Adjusted R-squared:  0.7025
## F-statistic: 71.85 on 2 and 58 DF,  p-value: < 2.2e-16
```

```
# Display the correlation coefficient
coefficients(reg)
```

```
##      (Intercept) Benchmark_Result      Year_Fixed
##      253.73197738      0.02517316      -33.74488399
```

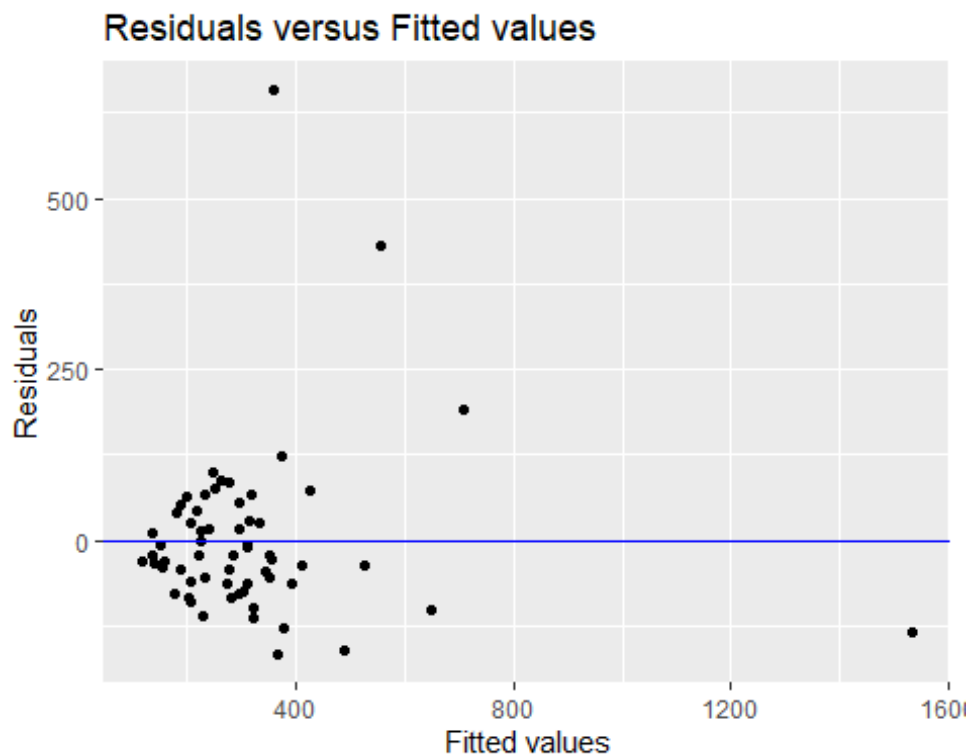
Assumptions Check

```
# Residuals versus Fitted values
```

```
cpudata$resids <- residuals(reg)
```

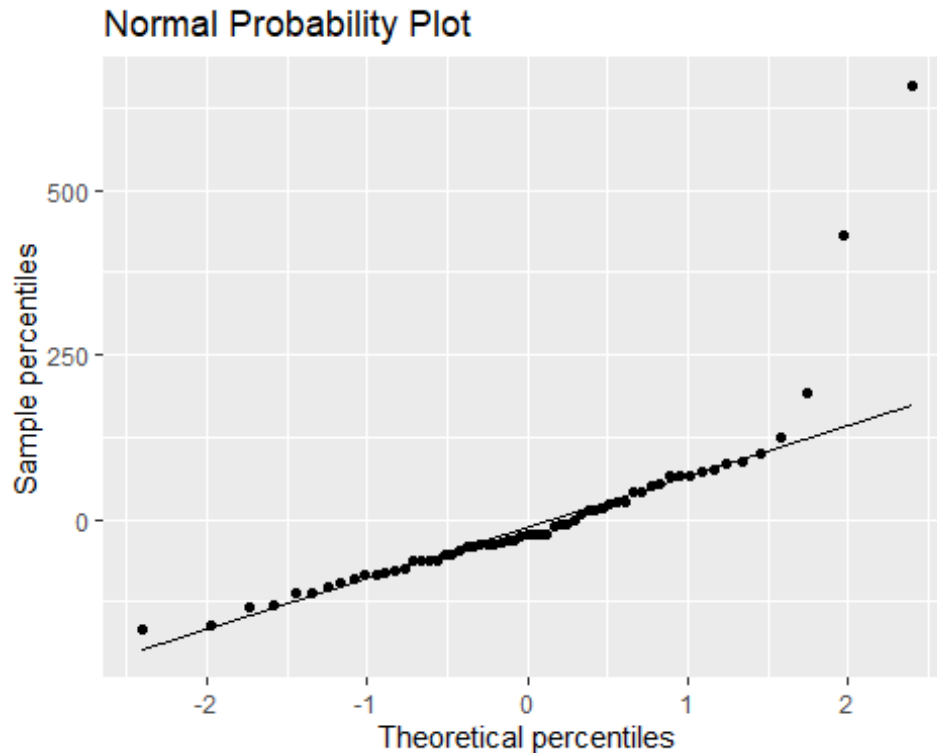
```
cpudata$predicted <- predict(reg)
```

```
ggplot(cpudata, aes(x=predicted, y=resids)) + geom_point() +
geom_hline(yintercept=0, color = "blue") +
labs(title = "Residuals versus Fitted values", x = "Fitted values", y
="Residuals")
```



```
# Normal probability plot
```

```
ggplot(cpudata, aes(sample = resids)) + stat_qq() + stat_qq_line() +
labs(title = "Normal Probability Plot", x = "Theoretical percentiles", y =
"Sample percentiles")
```



Check the interaction effect of all variables

`vif(reg)`

```
## Benchmark_Result      Year_Fixed
##           1.445602           1.445602
```

Will have to ejected some outliers in order to make the assumptions checks pass. Too many outliers that could lead to incorrect assumptions.

Deciding on the Best Model

Find the best model for each number of predictors (with 3 predictors maximum)

```
models <- regsubsets(MSRP ~ Benchmark_Result + Year_Fixed, cpudata, nvmax = 3)
```

```
models.sum <- summary(models)
```

Create four plots within a 2x2 frame to compare the different criteria

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

SSE

```
plot(models.sum$rss, xlab = "Number of predictors", ylab = "SSE", type = "l")
```

R2

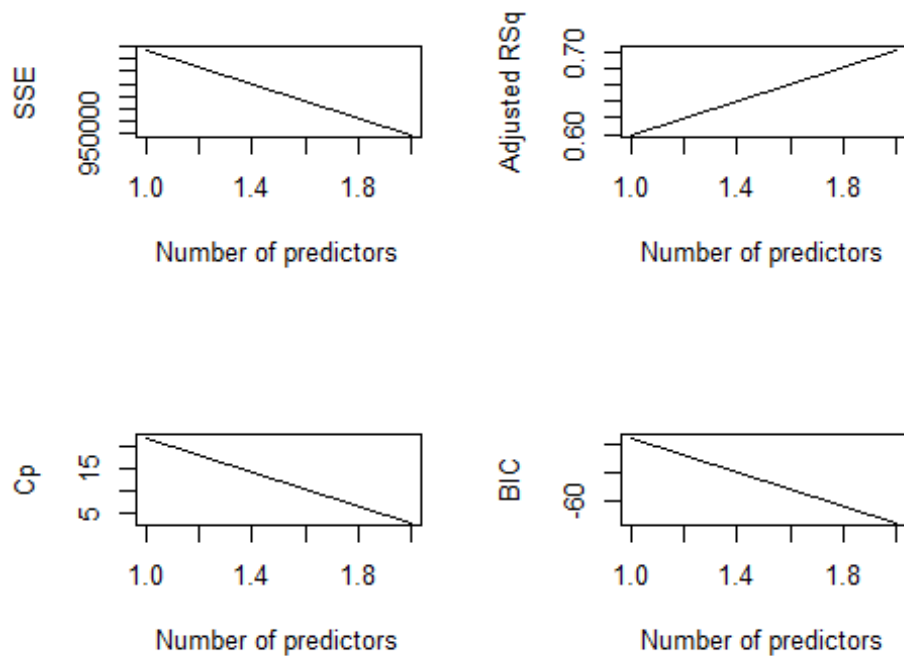
```
plot(models.sum$adjr2, xlab = "Number of predictors", ylab = "Adjusted RSq", type = "l")
```

Mallows' Cp

```
plot(models.sum$cp, xlab = "Number of predictors", ylab = "Cp", type = "l")
```

BIC

```
plot(models.sum$bic, xlab = "Number of predictors", ylab = "BIC", type = "l")
```



Calculate the squared predictor variables to include in the model and the interaction term:

```
cpudata <- cpudata %>%
```

```
mutate(bench2 = Benchmark_Result^2,
```

```
bench.year = Benchmark_Result*Year_Fixed)
```

Fit the polynomial regression model

```
reg2 <- lm(MSRP ~ Year_Fixed + Benchmark_Result + bench2 + bench.year,  
cpudata)
```

Display the summary table for the regression model

```
summary(reg2)
```

```
##
```

```
## Call:
```

```
## lm(formula = MSRP ~ Year_Fixed + Benchmark_Result + bench2 +
```

```
## bench.year, data = cpudata)
```

```
##
```

```
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
```

```
## -305.43  -43.84   -6.61   32.58  529.29
```

```
##
```

```
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  -1.248e+02  7.450e+01  -1.675   0.0996 .
```

```
## Year_Fixed     7.194e+00  1.164e+01   0.618   0.5390
```

```
## Benchmark_Result  9.490e-02  1.164e-02   8.152 4.34e-11 ***
```

```
## bench2         9.386e-08  8.796e-08   1.067   0.2905
```

```
## bench.year     -7.826e-03  1.424e-03  -5.497 9.85e-07 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 100.9 on 56 degrees of freedom
## Multiple R-squared:  0.8263, Adjusted R-squared:  0.8139
## F-statistic: 66.59 on 4 and 56 DF,  p-value: < 2.2e-16
```

Choose the best model.

*# Display the best model (selected predictors are indicated by *) for each number of predictors*

```
models.sum$outmat
```

```
##           Benchmark_Result Year_Fixed
## 1  ( 1 ) "*"                " "
## 2  ( 1 ) "*"                "**"
```

Creating the final model

Printing the final model with the best number of predictor variables

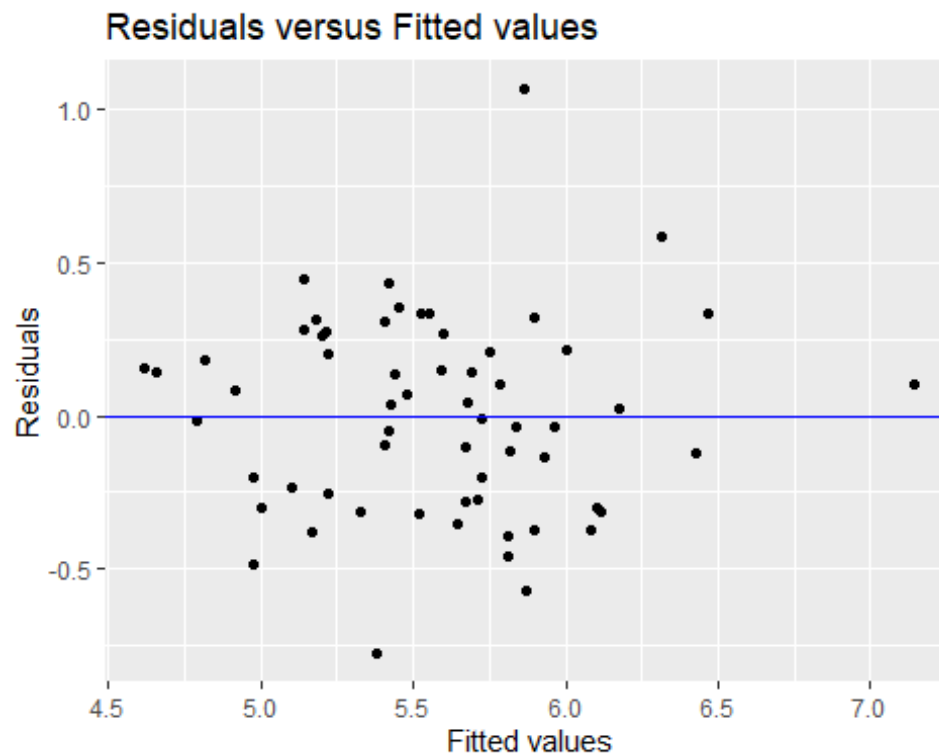
```
lm_log.model = lm(log1p(MSRP) ~ log1p(Benchmark_Result) + log1p(Year_Fixed),
data = cpudata)
summary(lm_log.model)
```

```
##
## Call:
## lm(formula = log1p(MSRP) ~ log1p(Benchmark_Result) + log1p(Year_Fixed),
##     data = cpudata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.77414 -0.27192  0.02001  0.21139  1.06391
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.89780    0.67994  -2.791   0.0071 **
## log1p(Benchmark_Result)  0.99690    0.09037  11.032 7.20e-16 ***
## log1p(Year_Fixed)     -0.83145    0.12480  -6.662 1.08e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.326 on 58 degrees of freedom
## Multiple R-squared:  0.6806, Adjusted R-squared:  0.6696
## F-statistic: 61.79 on 2 and 58 DF,  p-value: 4.225e-15
```

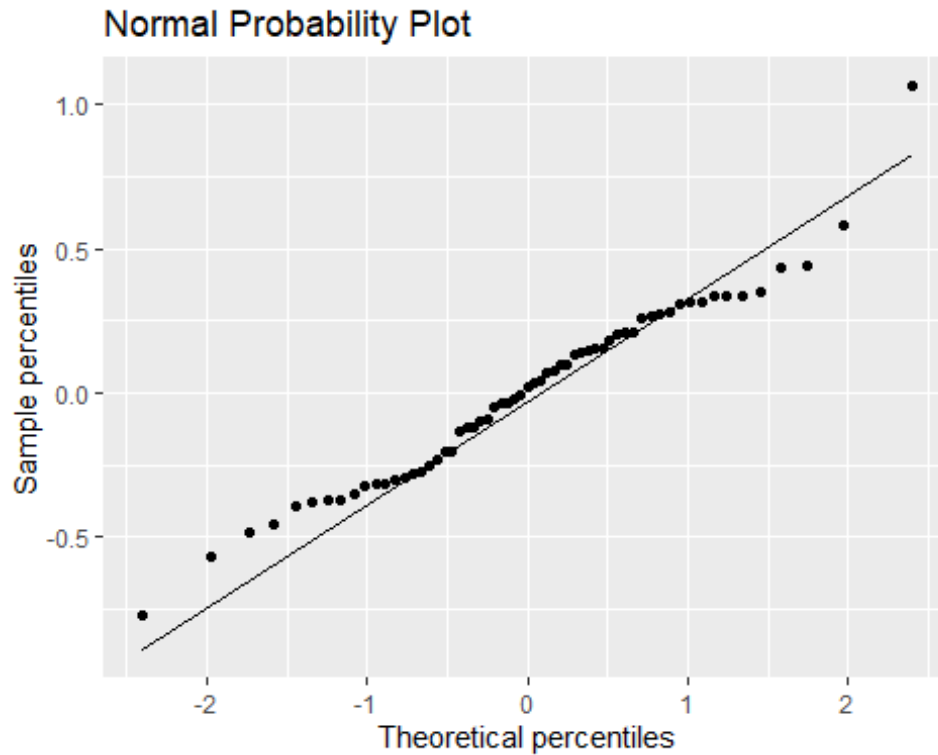
Residuals versus Fitted values

```
cpudata$resids2 <- residuals(lm_log.model)
cpudata$predicted2 <- predict(lm_log.model)
ggplot(cpudata, aes(x=predicted2, y=resids2)) + geom_point() +
geom_hline(yintercept=0, color = "blue") +
```

```
labs(title = "Residuals versus Fitted values", x = "Fitted values", y = "Residuals")
```



```
# Normal probability plot  
ggplot(cpu_data, aes(sample = resid2)) + stat_qq() + stat_qq_line() +  
labs(title = "Normal Probability Plot", x = "Theoretical percentiles", y =  
"Sample percentiles")
```



Check the interaction effect of all variables

`vif(lm_log.model)`

## log1p(Benchmark_Result)	log1p(Year_Fixed)
## 1.928061	1.928061

The equation for the final model is without outliers removed in logit form:

$MSRP = -1.897 + 0.997\log_{1p}(\text{Benchmark_Result}) - 0.832\log_{1p}(\text{Year_Fixed})$

The coefficient of determination is 0.681