



Probability & Statistic-MT2013

Assignment's Report

Analyses The Relationships between Various CPU Specifications
with Multi-factor ANOVA Test
and Multiple Linear Regression Models using R

Lecturer: Nguyễn Tiến Dũng
Class: CC03
Group: 03
Students: Cao Minh Quang - 2052221
Trần Cao Duy Trường - 2052299
Lâm Quang Khải - 2052128
Hoàng Cao Quốc Thắng - 2050020
Trương Huỳnh Đăng Khoa - 2053145



Contents

1	Member list and Workload	3
2	Introduction	4
3	Data Import	6
4	Data cleaning	6
5	Data Visualization	8
5.1	Transformation	8
5.2	Descriptive Statistics	9
5.3	Graphs	9
5.3.1	Histogram for The Number of Cores and Threads	9
5.3.2	Base Frequency, Turbo Frequency and Bus speed related to The Number of Cores and Threads	10
5.3.3	Box Plot for The Cache Size, Maximum Memory Size and Bus Speed	12
6	Theoretical Basis	14
6.1	Multi-factor ANOVA Test	14
6.1.1	Basic Concept of Two-way ANOVA	14
6.1.2	Find the best-fit model	15
6.1.3	Levene Test for Homoscedasticity of Variance	15
6.1.4	Tukey's Honestly Significant Difference (Tukey's HSD) post-hoc test	16
6.2	Kruskal-Wallis Test when countering Assumptions' Failures in ANOVA	16
6.2.1	Kruskal-Wallis Test	16
6.2.2	The Epsilon-Squared Scale	17
6.2.3	Dunn's Test for Multiple Comparisons	18
6.3	Multiple Linear Regression Model	18
6.3.1	Basic Concept	18
6.3.2	Interpreting Linear Regression Model Output in R	19
6.3.3	Breusch-Pagan Test for Heteroscedasticity in Regression Models	20
6.3.4	The Usage of Correlation Matrix	21
6.3.5	Multicollinearity Check with VIFs	21
6.4	Shapiro-Wilk Test for Normality	22
6.5	Residuals vs Leverage	22
6.6	Scale-Location Graph	24
6.7	Residuals vs Fitted Graph	24
6.8	Normal Q-Q Graph	26
7	ANOVA Test for The Effect of The Numbers of Cores and Threads on other CPU Specifications	26
7.1	Overview	26
7.2	Effects on The Processor Base Frequency	27
7.3	Effects on The Turbo Frequency	34
7.4	Effects on The Maximum Temperature	38
7.5	Effects on The Bus Speed	41
7.6	Summary	46



8	Fitting Linear Regression Models	47
8.1	Overview	47
8.2	Diagnostic for Regression with Correlation Matrix	47
8.3	Fitting Cache Size in Linear Regression Models	48
8.3.1	With Cores and The Others	48
8.3.2	With Threads and The Others	50
8.4	Fitting Maximum Memory Size in Linear Regression Models	52
8.4.1	With Cores and The Others	52
8.4.2	With Threads And The Others	54
8.5	Fitting Bus Speed In A Linear Regression Model	55
8.5.1	With Cores And The Others	55
8.5.2	With Threads and The Others	57
8.6	Fitting Base Frequency in Linear Regression Models	59
8.6.1	With Cores and The Others	59
8.6.2	With Threads and The Others	60
8.7	Fitting Turbo Frequency in Linear Regression Models	61
8.7.1	With Cores and The Others	61
8.7.2	With Threads and The Others	63
8.8	Summary	65
9	References	66



1 Member list and Workload

No.	Full name	Student ID	Task	Contribution
1	Cao Minh Quang	2052221	Tasks 6 + 7.1 + 7.5 + 8.1 + 8.2 + 8.3.1 + 8.8	25%
2	Trần Cao Duy Trường	2052299	Tasks 3 + 4 + 7.2 + 8.7	17%
3	Lâm Quang Khải	2052128	Task 2 + 4 + 5.3.1 + 7.1 + 7.4 + 8.5	21%
4	Hoàng Cao Quốc Thắng	2050020	Tasks 5.1 + 5.2 + 5.3.2 + 7.3 + 8.4	18%
5	Trương Huỳnh Đăng Khoa	2053145	Tasks 2 + 5.3.3 + 7.4 + 8.3.2 + 8.6	19%

2 Introduction

Our group uses a Kaggle dataset that contains a collection of Intel CPUs released from 2010 through 2021 for our data analysis project. We hope to demonstrate the impact of cores, threads, bus speed, cache size, maximum memory, and maximum temperature on our computers' performance by analyzing the data from this dataset. We'll explain what these features are and how they relate to one another to help you transition into this computing subject.

To begin, "CPU" stands for "Central Processing Unit," which is also referred to as a central processor. This electronic circuitry performs basic arithmetic, logic, controlling, and input/output operations specified by the instructions in the program. We can think of these actions as adding and removing data, as well as moving data around. A CPU used to be just one processing unit that could only handle one instruction at a time.

However, since operating systems and programs have a lot more data and provide considerably more instructions for CPUs, we now have multiple processing units in one processor which are referred as cores. This means one processor can process multiple instructions at a time which significantly increases the speed of the CPU. For even better performance and multi-tasking, cores are split into threads. And its meaning is the same as slitting the central processor into cores.

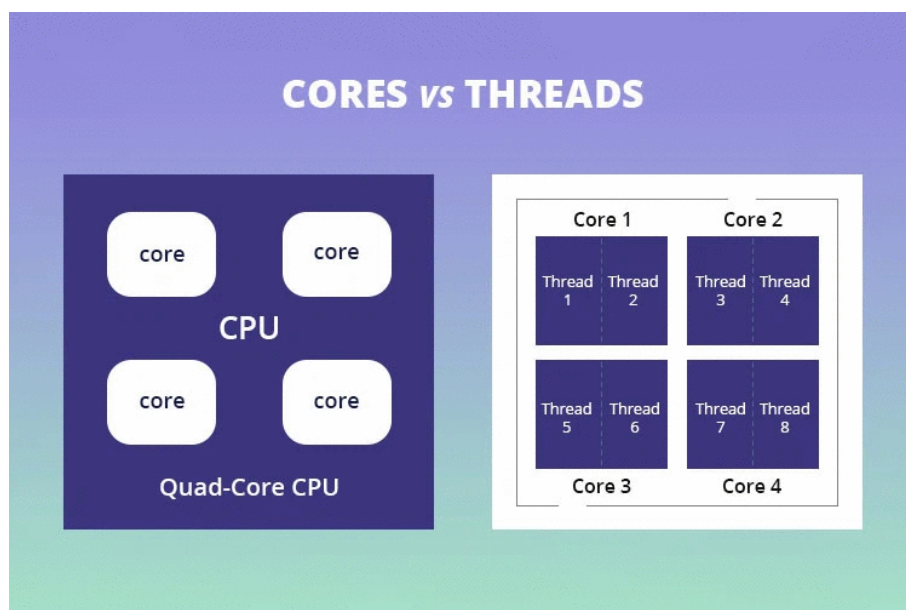


Figure 1: Cores and threads illustration.

The figure about bus in our dataset refers to the front side bus (FSB), which connect the CPU to the memory controller to manage the flow of data going to and from the computer's main memory(RAM/ROM). Therefore, the higher speed of FSB, the better computer's performance we have.

What is **cache** ? Cache is a small amount of memory which is a part of the CPU - closer to the CPU than RAM. It is used to temporarily hold instructions and data that the CPU

is likely to reuse. The CPU control unit automatically checks cache for instructions before requesting data from RAM. This saves fetching the instructions and data repeatedly from RAM – a relatively slow process which might otherwise keep the CPU waiting. Transfers to and from cache take less time than transfers to and from RAM. The more cache there is, the more data can be stored closer to the CPU.

Cache is graded as Level 1 (L1), Level 2 (L2) and Level 3 (L3):

- **L1** is usually part of the CPU chip itself and is both the smallest and the fastest to access. Its size is often restricted to between 8 KB and 64 KB.
- **L2** and **L3** caches are bigger than **L1**. They are extra caches built between the CPU and the RAM. Sometimes L2 is built into the CPU with L1. L2 and L3 caches take slightly longer to access than **L1**. The more **L2** and **L3** memory available, the faster a computer can run.

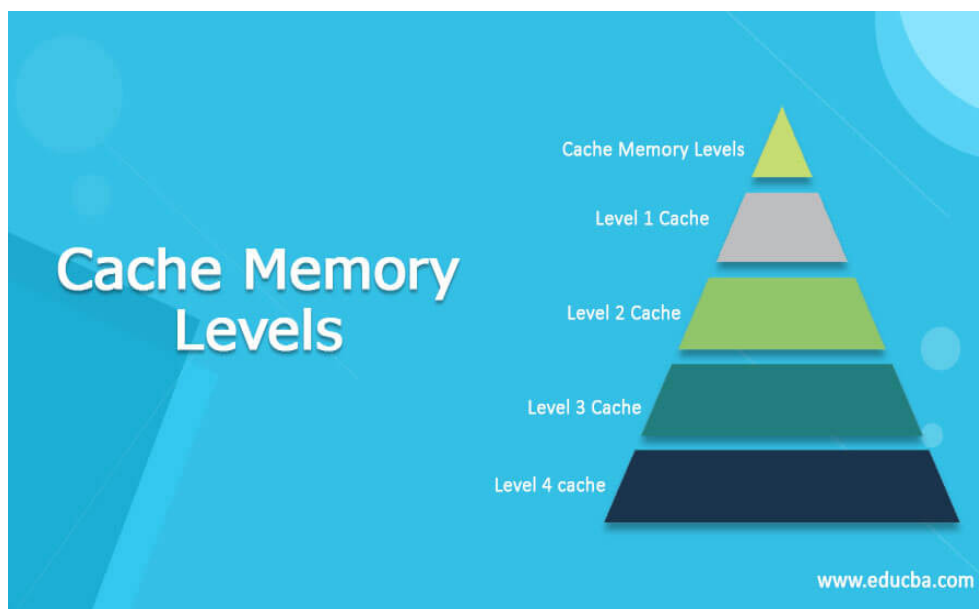


Figure 2: Cache memory level.

Not a lot of physical space is allocated for cache. There is more space for RAM, which is usually larger and less expensive.

Max Memory is the maximum amount of RAM this CPU will work with. For example, if we use 32GB max memory CPU, we can plug in 64GB RAM, but only 32GB will be used.

Max Temperature is the maximum amount of degree this CPU can be sustained and work properly. If somehow the temperature of CPU is over the maximum range, it will reduce the CPU performance just to cool it down and maybe break the CPU if the temperature is too hot.

3 Data Import

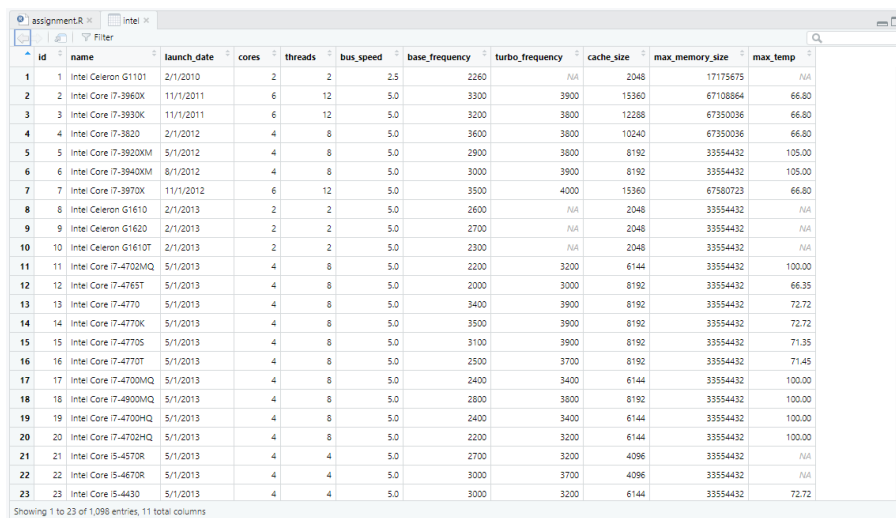
In order to import data from .csv file and store it in an object the following code segment will be used:

```
> setwd("C:/Users/Truong/Downloads/Documents")
> intel <- read.csv("IntelProcessors.csv", header = TRUE, sep = ",")
> View(intel)
```

We used "setwd" instruction to set the directory to a folder where store our .csv file. Furthermore, we should also put our R-script file in the same path.

The instruction read.csv then will import data from file "IntelProcessors.csv" and store it in an object named "Intel".

The instruction View allow us to take a look at our data frame.



id	name	launch_date	cores	threads	bus_speed	base_frequency	turbo_frequency	cache_size	max_memory_size	max_temp
1	Intel Celeron G1101	2/1/2010	2	2	2.5	2260	N/A	2048	17175675	N/A
2	Intel Core i7-3960X	11/1/2011	6	12	5.0	3300	3900	15360	67108864	66.80
3	Intel Core i7-3930K	11/1/2011	6	12	5.0	3200	3800	12288	67350036	66.80
4	Intel Core i7-3820	2/1/2012	4	8	5.0	3600	3800	10240	67350036	66.80
5	Intel Core i7-3920XM	5/1/2012	4	8	5.0	2900	3800	8192	33554432	105.00
6	Intel Core i7-3940XM	8/1/2012	4	8	5.0	3000	3900	8192	33554432	105.00
7	Intel Core i7-3970K	11/1/2012	6	12	5.0	3500	4000	15360	67580723	66.80
8	Intel Celeron G1610	2/1/2013	2	2	5.0	2600	N/A	2048	33554432	N/A
9	Intel Celeron G1620	2/1/2013	2	2	5.0	2700	N/A	2048	33554432	N/A
10	Intel Celeron G1610T	2/1/2013	2	2	5.0	2300	N/A	2048	33554432	N/A
11	Intel Core i7-4702MQ	5/1/2013	4	8	5.0	2200	3200	6144	33554432	100.00
12	Intel Core i7-4765T	5/1/2013	4	8	5.0	2000	3000	8192	33554432	66.35
13	Intel Core i7-4770	5/1/2013	4	8	5.0	3400	3900	8192	33554432	72.72
14	Intel Core i7-4770K	5/1/2013	4	8	5.0	3500	3900	8192	33554432	72.72
15	Intel Core i7-4770S	5/1/2013	4	8	5.0	3100	3900	8192	33554432	71.35
16	Intel Core i7-4770T	5/1/2013	4	8	5.0	2500	3700	8192	33554432	71.45
17	Intel Core i7-4700MQ	5/1/2013	4	8	5.0	2400	3400	6144	33554432	100.00
18	Intel Core i7-4800MQ	5/1/2013	4	8	5.0	2800	3800	8192	33554432	100.00
19	Intel Core i7-4700HQ	5/1/2013	4	8	5.0	2400	3400	6144	33554432	100.00
20	Intel Core i7-4702HQ	5/1/2013	4	8	5.0	2200	3200	6144	33554432	100.00
21	Intel Core i5-4570R	5/1/2013	4	4	5.0	2700	3200	4096	33554432	N/A
22	Intel Core i5-4670R	5/1/2013	4	4	5.0	3000	3700	4096	33554432	N/A
23	Intel Core i5-4430	5/1/2013	4	4	5.0	3000	3200	6144	33554432	72.72

Figure 3: After cleaning data

Our data set contains information of 1098 Intel CPUs from 2010 to 2021.

4 Data cleaning

After reading the input file, next thing we have to do is to check if the data contain empty cells. And because of that we have written code to clean each variables.

```
# Data Cleaning
> install.packages("tidyverse")
> library(tidyverse)
> cleanIntel <- drop_na(intel)
> View(cleanIntel)
```

The drop-na instruction will sequentially check whether each row has empty cells or not. If there is, the whole row will be deleted from the dataframe. A new object name "cleanIntel" will hold our data after this process.

As we can see, almost half of the data frame has been removed due to empty entries.

Data	
cleanIntel	521 obs. of 11 variables
intel	1098 obs. of 11 variables

Figure 4: After cleaning data

Now we start to list out the range of each variable and its data type in "cleanIntel". These information may be helpful in later section. In order to do that, we repeatedly call 2 functions **table** and **is.numeric** to for easy further examination.

```
> is.numeric(cleanIntel$name)
[1] FALSE

> is.numeric(cleanIntel$launch_date)
[1] FALSE

> table(cleanIntel$cores)
 2   4   6   8  10  12  14  16  18  24  28  32  38
98 216 100  60  20   6   4   4   5   3   3   1   1
> is.numeric(cleanIntel$cores)
[1] TRUE

> table(cleanIntel$threads)
 2   4   6   8  12  16  20  24  28  32  36  48  56  64  76
2 157  25 162  75  53  20   6   4   4   5   3   3   1   1
> is.numeric(cleanIntel$threads)
[1] TRUE

> table(cleanIntel$bus_speed)
 0   4   5   8
3  85 127 306
> is.numeric(cleanIntel$bus_speed)
[1] TRUE

> table(cleanIntel$base_frequency)
 700  800 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900
 1     1     4     9     6     6     5     9    16    10    15     9
2000 2100 2200 2300 2400 2500 2600 2700 2800 2900 3000
 15     7    19    21    25    23    26    25    34    28    25
3100 3200 3300 3400 3500 3600 3700 3800 3900 4000 4100 4200
 25    18    22    15    24    26    20    12     4    10     4     1
4300
 1
> is.numeric(cleanIntel$base_frequency)
[1] TRUE

> table(cleanIntel$turbo_frequency)
```



```

1900 2000 2300 2400 2600 2700 2800 2900 3000 3100 3200 3300
  3    2    2    1    2    7    3    9    10   10   19   19
3400 3500 3600 3700 3800 3900 4000 4100 4200 4300 4400
 24   26   27   18   30   38   25   26   16   17   32
4500 4600 4700 4800 4900 5000 5100 5200 5300
 30   27   16   21   15   18   11    8    9
> is.numeric(cleanIntel$turbo_frequency)
[1] TRUE

> table(cleanIntel$cache_size)
2048  3072  4096  6144  8192  8448  9216 10240 11264 12288
   2    38    56   116    93    8    27    2    2    74
14080 15360 16384 16896 19712 20480 21504 22528 24576
   2    5    44    6   11   15    1    3    3
25344 33792 36864 39424 49152 58368
   5    2    1    3    1    1
> is.numeric(cleanIntel$cache_size)
[1] TRUE

> table(cleanIntel$max_memory_size)
16777216  33554432  67108864 67350036.48  67580723.2
   52    124    121    2    1
134217728  268435456  536870912 1073741824
  188    4    8    13
2147483648  4294967296
   3    5
> is.numeric(cleanIntel$max_memory_size)
[1] TRUE

> table(cleanIntel$max_temp)
 59   61   62   64   65   66 66.35 66.8   68   70
  1    3    2    6    2    5    6   10    5    1
71 71.35 71.45  72 72.72 73 74.04  76  77
  8    9    5    2   11    1    1    2    2
78   80   82   84   85   86   88   92   94   95
  2    5    2    1    2    2    1    3    3    3
98   99  100  102  105
  1    1  391    1   21
> is.numeric(cleanIntel$max_temp)
[1] TRUE

```

5 Data Visualization

5.1 Transformation

In order to analyze the data in the dataset we have to remove several unwanted categories such as index of each object, the name and launch data.

```

> newIntel <- cleanIntel[,c("cores", "threads", "bus_speed",
  "base_frequency", "turbo_frequency", "cache_size",
  "max_memory_size", "max_temp")]
> head(newIntel)

```

By doing this, We only selected some attributes to compute the statistics. In this case "cores", "threads", "bus_speed", "base_frequency", "turbo_frequency", "cache_size", "max_memory_size", "max_temp" is selected in new data frame called "newIntel".

The Result is:

Data	
cleanIntel	521 obs. of 11 variables
intel	1098 obs. of 11 variables
newIntel	521 obs. of 8 variables

Figure 5: Comparison amongst 3 data frames.

As we can see, the number of variable between newIntel and clean Intel is reduce by 3 where "id", "name" and "launch_date" have been dropped.

5.2 Descriptive Statistics

Since considering Statistic, there are some crucial values that we need to compute such as min, max, median, mean, var, sum and so on.

```
> install.packages("pastecs")
> library(pastecs)
> stat.desc(newIntel[, c(1,2,3,4,5,6,7,8)]) %>% round(4)
```

And the result is:

```
> stat.desc(newIntel[, c(1,2,3,4,5,6,7,8)]) %>% round(4)
      cores threads bus_speed base_frequency turbo_frequency cache_size
nbr.val 521.0000 521.0000 521.0000 521.0000 521.0000 5.210000e+02
nbr.null 0.0000 0.0000 3.0000 0.0000 0.0000 0.000000e+00
nbr.na 0.0000 0.0000 0.0000 0.0000 0.0000 0.000000e+00
min 2.0000 2.0000 0.0000 700.0000 1900.0000 2.048000e+03
max 38.0000 76.0000 8.0000 4300.0000 5300.0000 5.836800e+04
range 36.0000 74.0000 8.0000 3600.0000 3400.0000 5.632000e+04
sum 2848.0000 5242.0000 3423.0000 1412800.0000 2090200.0000 5.164544e+06
median 4.0000 8.0000 8.0000 2800.0000 4000.0000 8.192000e+03
mean 5.4664 10.0614 6.5701 2711.7083 4011.9002 9.912752e+03
SE.mean 0.1777 0.3684 0.0776 32.7860 29.6969 2.851942e+02
CI.mean.0.95 0.3492 0.7238 0.1524 64.4094 58.3407 5.602744e+02
var 16.4609 70.7193 3.1340 560035.7301 459473.4977 4.237591e+07
std.dev 4.0572 8.4095 1.7703 748.3554 677.8447 6.509678e+03
coef.var 0.7422 0.8358 0.2695 0.2760 0.1690 6.567000e-01

      max_memory_size max_temp
nbr.val 5.210000e+02 521.0000
nbr.null 0.000000e+00 0.0000
nbr.na 0.000000e+00 0.0000
min 1.677722e+07 59.0000
max 4.294967e+09 105.0000
range 4.278190e+09 46.0000
sum 8.583319e+10 49331.4600
median 6.710886e+07 100.0000
mean 1.647470e+08 94.6861
SE.mean 2.047797e+07 0.5140
CI.mean.0.95 4.022973e+07 1.0099
var 2.184800e+17 137.6685
std.dev 4.674184e+08 11.7332
coef.var 2.837200e+00 0.1239
```

Figure 6: Some significant descriptive Statistics.

5.3 Graphs

5.3.1 Histogram for The Number of Cores and Threads

We will use **histogram graph** to describe the numbers of cores and threads in the dataset.

```
> #Histograms of Cores and Threads
> gghistogram(cleanIntel, x = "cores", fill = "blue",
  add = "mean", rug = TRUE, add_density = TRUE)
> gghistogram(cleanIntel, x = "threads", fill = "red",
  add = "mean", rug = TRUE, add_density = TRUE)
```

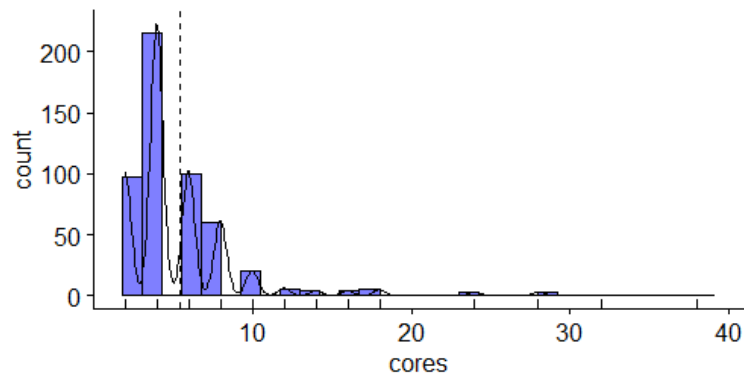


Figure 7: Histogram of Cores.

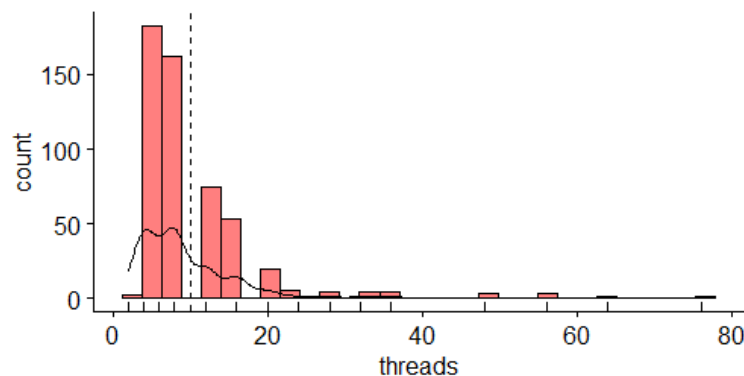


Figure 8: Histogram of Threads.

5.3.2 Base Frequency, Turbo Frequency and Bus speed related to The Number of Cores and Threads

The relationship between each pair Cores and Threads with Base Frequency, Turbo Frequency and Bus speed also play a crucial role in analyzing the data set. In each pair, we will use strip chart to illustrate the connection amongst them.

```
#strip chart for cores
> ggstripchart(newIntel, x = 'turbo_frequency', y = 'cores', color = 'cores')
> ggstripchart(newIntel, x = 'base_frequency', y = 'cores', color = 'cores')
> ggstripchart(newIntel, x = 'bus_speed', y = 'cores', color = 'cores')
```

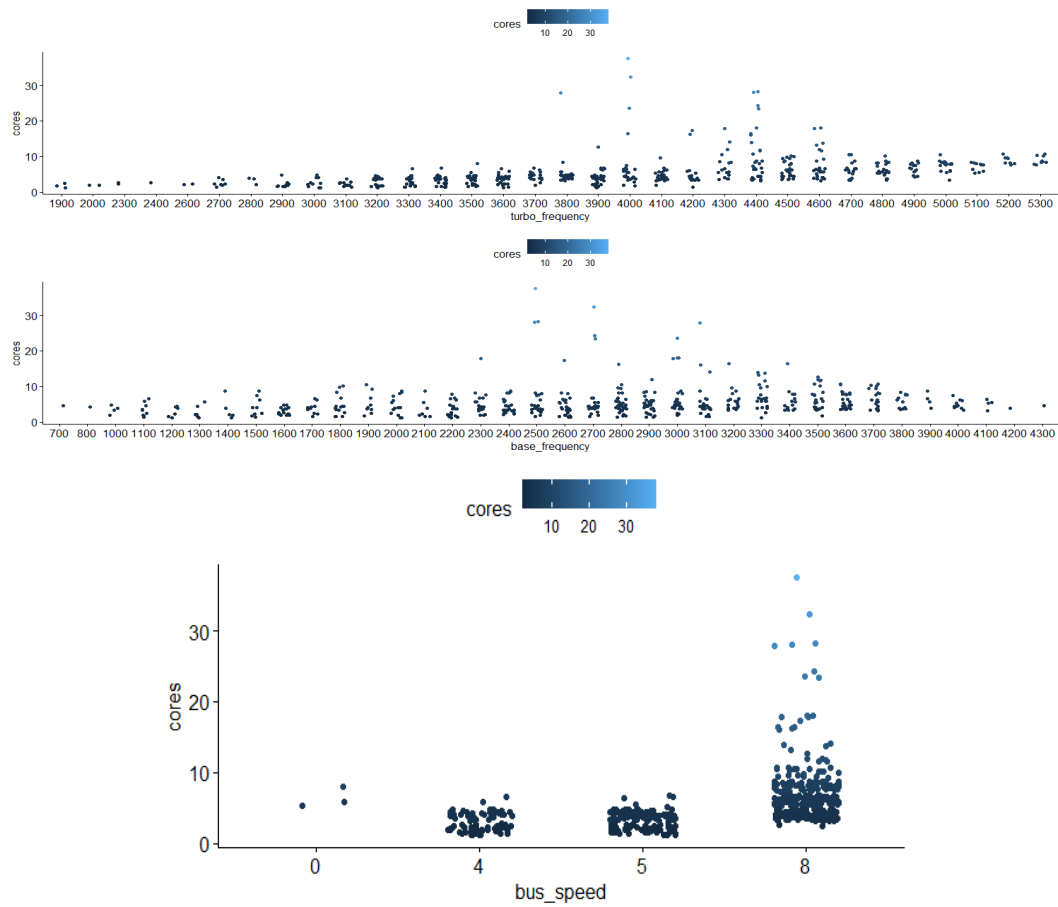
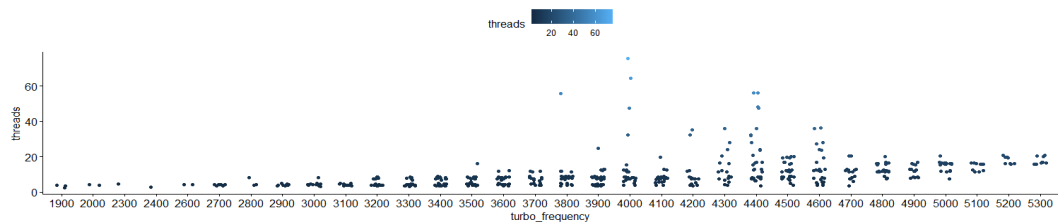


Figure 9: Box plot for cores in relation with others.

```
#strip chart for threads
> ggstripchart(newIntel, x = 'turbo_frequency', y = 'threads', color = 'threads')
> ggstripchart(newIntel, x = 'base_frequency', y = 'threads', color = 'threads')
> ggstripchart(newIntel, x = 'bus_speed', y = 'threads', color = 'threads')
```



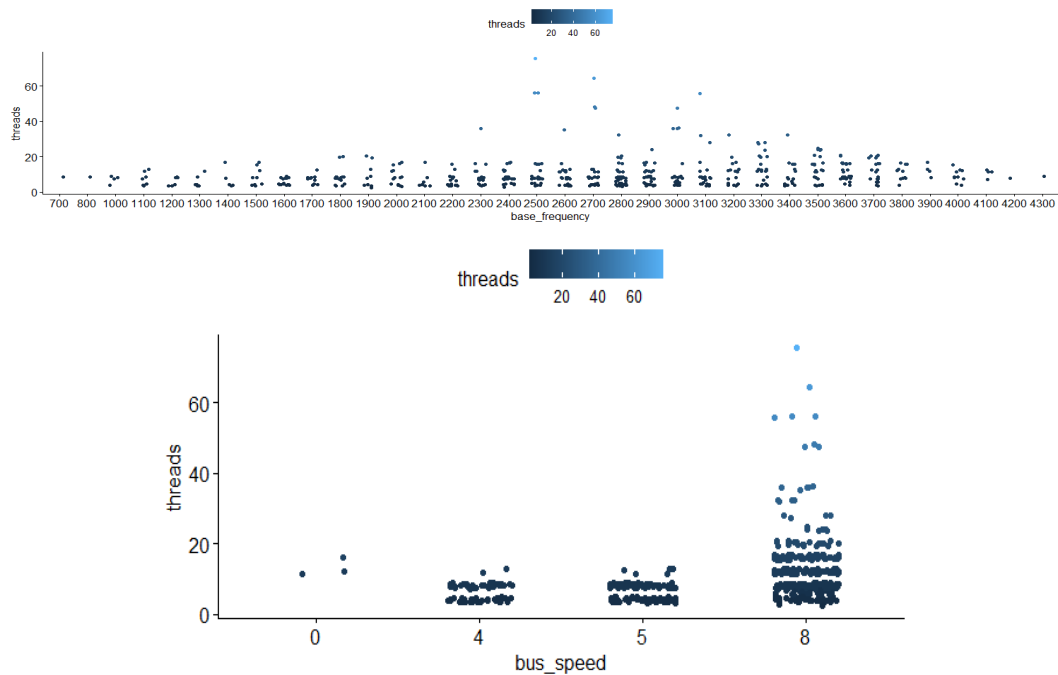


Figure 10: Box plot for thread in relation with others.

5.3.3 Box Plot for The Cache Size, Maximum Memory Size and Bus Speed

We will use **box plot** to show off distribution of `cache_size`, `max_memory_size`, `max_temp` in range $(0, 4e + 09)$ of KB, it also considers the mean value of these categories. The two packages for this work are **tidyr** and **ggplot2**.

```
#Box plot for cache_size
> gathered <- newIntel %>%
> pivot_longer(c(cache_size), values_to="KB")
> ggplot(gathered, aes(, y = KB)) + geom_boxplot()
+ labs(x="Cache_size", y = "KB" )
+ geom_boxplot(fill = 'red')
```

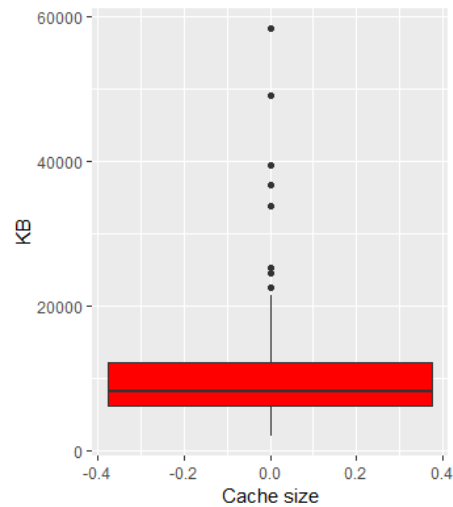


Figure 11: Box plot for cache_size.

```
#Box plot for max_memory_size
> gathered <- newIntel %>%
> pivot_longer(c(max_memory_size), values_to="KB")
> ggplot(gathered,aes(, y = KB)) + geom_boxplot()
+ labs(x="Max_memory_size", y = "KB" )
+ geom_boxplot(fill = 'red')
```

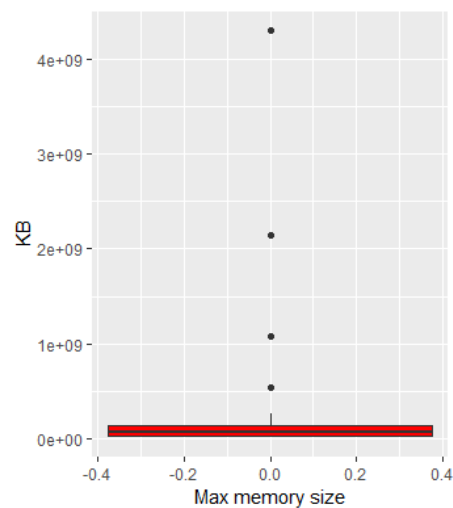


Figure 12: Box plot for max_memory_size.

```
#Box plot for bus_speed  
> gathered <- newIntel %>%  
> pivot_longer(c(bus_speed), values_to="GT_s")  
> ggplot(gathered,aes(, y = GT_s)) + geom_boxplot()  
  + labs(x="Bus speed", y = "GT/s" ) + geom_boxplot(fill = 'red')
```

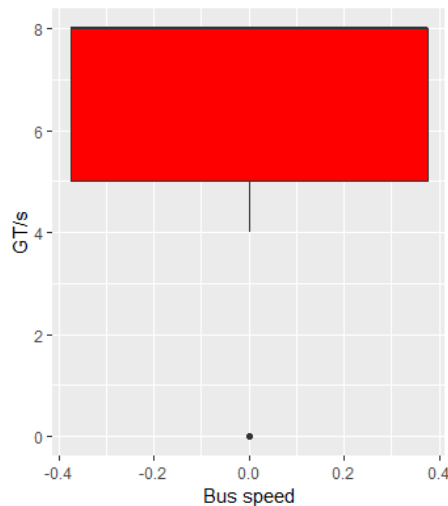


Figure 13: Box plot for bus_speed.

6 Theoretical Basis

6.1 Multi-factor ANOVA Test

6.1.1 Basic Concept of Two-way ANOVA

A two-way ANOVA is used to estimate how the mean of a quantitative variable changes according to the levels of two categorical variables. The usage of a two-way ANOVA is to know how two independent variables, in combination, affect a dependent variable.

How does the ANOVA test work ?

ANOVA tests for significance using the F-test for statistical significance. The F-test is a group-wise comparison test, which means it compares the variance in each group mean to the overall variance in the dependent variable.

If the variance within groups is smaller than the variance between groups, the F-test will find a higher F-value, and therefore a higher likelihood that the difference observed is real and not due to chance.

A two-way ANOVA with interaction tests three null hypotheses at the same time:

1. There is no difference in group means at any level of the first independent variable.
2. There is no difference in group means at any level of the second independent variable.
3. The effect of one independent variable does not depend on the effect of the other independent variable (no interaction effect).

Assumptions of the two-way ANOVA

1. Normally-distributed dependent variable: The values of the dependent variable should follow a bell curve.
2. Homogeneity of variance (or Homoscedasticity): The variation around the mean for each group being compared should be similar among all groups.
3. Independence of observation: Independent variables should not be dependent on one another (i.e. one should not cause the other). This is impossible to test with categorical variables – it can only be ensured by good experimental design. In addition, the dependent variable should represent unique observations – that is, your observations should not be grouped within locations or individuals.

6.1.2 Find the best-fit model

When doing the research, we may build up many ANOVA models to explain the data. Usually, we will want to use the best-fit model, which is the best explains the variation in the dependent variable.

The Akaike information criterion (AIC) is good test for model fit. AIC calculates the information value of each model by balancing the variaion explained against the number of parameters used.

In AIC model selection, we compare the information value of each model and choose the one with the lowest AIC value (a lower number means more information explained).

6.1.3 Levene Test for Homoscedasticity of Variance

In statistics, Levene's test is an inferential statistic used to evaluate the equality of variances for a variable determined for two or more groups. Some standard statistical procedures find that variances of the populations from which various samples are formed are equal. Levene's test assesses this assumption.

It examines the null hypothesis that the population variances are equal called homogeneity of variance or homoscedasticity. It compares the variances of k samples, where k can be more than two samples.

It's an alternative to Bartlett's test that is less sensitive to departures from normality.

Given a variable Y with sample size of N is divided into k subgroups, where N_i is the sample size of the i^{th} subgroup, the Levene Test statistic is defined as:

$$W = \frac{N - k}{k - 1} \cdot \frac{\sum_{i=1}^k N_i (\bar{Z}_i - \bar{Z})^2}{\sum_{i=1}^k \sum_{j=1}^{N_i} (Z_{ij} - \bar{Z}_i)^2}$$

where Z_{ij} can have one of the following three definitions:

1. $Z_{ij} = |Y_{ij} - \bar{Y}_i|$, where \bar{Y}_i is the **mean** of the i^{th} subgroup.
2. $Z_{ij} = |Y_{ij} - \hat{Y}_i|$, where \hat{Y}_i is the **median** of th i^{th} subgroup.
3. $Z_{ij} = |Y_{ij} - \bar{Y}_i'|$, where \bar{Y}_i' is the 10% **trimmed mean** of th i^{th} subgroup.

\bar{Z}_i are the group means of the Z_{ij} and \bar{Z} is the overall mean of the Z_{ij}

The three choices for defining Z_{ij} determine the robustness and power of Levene's test. By robustness, we mean the ability of the test to not falsely detect unequal variances when the underlying data are not normally distributed and the variables are in fact equal. By power, we mean the ability of the test to detect unequal variances when the variances are in fact unequal.

6.1.4 Tukey's Honestly Significant Difference (Tukey's HSD) post-hoc test

ANOVA will tell us if there are differences among group means, but not what the differences are. To find out which groups are statistically different from one another, we can perform a Tukey's Honestly Significant Difference (Tukey's HSD) post-hoc test for pairwise comparisons.

Tukey's test compares the means of all treatments to the mean of every other treatment and is considered the best available method in cases when confidence intervals are desired or if sample sizes are unequal.

The test statistic used in Tukey's test is denoted q and is essentially a modified t -statistic that corrects for multiple comparisons. q can be found similarly to the t -statistic:

$$q_{\alpha, k, N-k}$$

The studentized range distribution of q is defined as:

$$q_s = \frac{Y_{max} - Y_{min}}{se}$$

where, Y_{max} and Y_{min} are the largest and smallest means of the two groups being compared. se is defined as the standard error of the entire test.

6.2 Kruskal-Wallis Test when countering Assumptions' Failures in ANOVA

6.2.1 Kruskal-Wallis Test

Definition

Kruskal-Wallis test (also known as Kruskal-Wallis H test or Kruskal-Wallis ANOVA) is a non-parametric (distribution free) alternative to the one-way ANOVA.

Kruskal-Wallis test is useful when the assumptions of ANOVA are not met or there is a significant deviation from the ANOVA assumptions. If the data meets the ANOVA assumptions, it is better to use ANOVA as it is a little more powerful than non-parametric tests.

Kruskal-Wallis test used for comparing the differences between two or more groups. It is an extension to the Mann Whitney U Test, which is used for comparing two groups. It compares the mean ranks (medians) of groups.

Kruskal-Wallis test does not assume any specific distribution (such as normal distribution of samples) for calculating test statistics and p values.

The sample mean ranks or medians are compared in the Kruskal-Wallis test, which distinguishes it from the ANOVA, which compares sample means. Medians are less sensitive to outliers than means.

Kruskal-Wallis' Assumptions

The independent variable should have two or more independent groups.

The observations from the independent groups should be randomly selected from the target populations.

Observations are sampled independently from each other (no relation in observations between the groups and within the groups) i.e., each subject should have only one response.

The dependent variable should be continuous or discrete.

Kruskal-Wallis test Hypotheses

If each group distribution is not the same,

Null hypothesis: All group mean are equal. **vs** Alternative hypothesis: At least, one group mean different from other groups

In terms of medians (when each group distribution is same),

Null hypothesis: Population medians are equal. **vs** Alternative hypothesis: At least, one population mean different from other populations

Kruskal-Wallis test statistic

$$H = \left(\frac{12}{N(N+1)} \sum_{j=1}^k \frac{R_j^2}{n_j} - 3(N+1) \right)$$

where,

N is the total observation in all groups (total sample size)

k is the number of groups

n_j is sample size for the i^{th} group

R_j is the sum of ranks of j^{th} group

H is approximately chi-squared distributed with $df = k - 1$. The p -value is calculated based on the comparison between the critical value and the H value. If $H \geq$ critical value, we can reject the null hypothesis and vice versa.

6.2.2 The Epsilon-Squared Scale

For the Kruskal-Wallis test, epsilon-squared is a method of choice for effect size measurement.

An epsilon square of 0 would mean no differences (and no influence), while one of 1 would indicate a full dependency.

$0.00 < 0.01$ - Negligible

$0.01 < 0.04$ - Weak

$0.04 < 0.16$ - Moderate

$0.16 < 0.36$ - Relatively strong

$0.36 < 0.64$ - Strong

$0.64 < 1.00$ - Very strong

6.2.3 Dunn's Test for Multiple Comparisons

When the results of a Kruskal-Wallis test are statistically significant, it is appropriate to conduct Dunn's Test to determine exactly which groups are different.

Dunn's Test performs pairwise comparisons between each independent group and tells which groups are statistically significantly different at some significant level α .

Dunn's z -test statistic approximates the exact rank-sum test statistics by using the mean rankings of the outcome in each group from the preceding Kruskal-Wallis test. To compare group A and B, we calculate:

$$z_i = \frac{y_i}{\sigma_i}$$

where, i is one of the 1 to m multiple comparisons, $y_i = \overline{W}_A - \overline{W}_B$ and σ_i is the standard deviation of y_i , given by:

$$\sigma_i = \sqrt{\left[\frac{N(N+1)}{12} - \frac{\sum_{s=1}^r \tau_s^3 - \tau_s}{12(N-1)} \right] \left(\frac{1}{n_A} + \frac{1}{n_B} \right)}$$

where, N is the total number of observation across all groups, r is the number of tied ranks, and τ_s is the number of observations tied at the s^{th} specific tied value. When there are no ties, the term with the summation in the denominator equals zero, and the calculation will be simplified considerably.

Multiple-Comparison Adjustments

There are several methods for the adjustments such as the Bonferroni adjustment, Holm's stepwise adjustment, Holm-Sidak's stepwise adjustment and Benjamini-Hochberg stepwise adjustment.

We will use the Benjamini-Hochberg method since this is a really powerful tool to decrease the false discovery rate and our data set seem to be quite large, sometimes small p-values (less than 5%) happen by chance, which could lead to incorrectly reject the true null hypotheses.

6.3 Multiple Linear Regression Model

6.3.1 Basic Concept

Multiple linear regression is used to estimate the relationship between two or more independent variables and one dependent variable. Multiple linear regression can be use when we want to know:

1. How strong the relationship is between two or more independent variables and one dependent variable.
2. The value of the dependent variable at a certain value of the independent variables.

Assumptions of multiple linear regression

Linearity: the line of best fit through the data points is a straight line, rather than a curve or some sort of grouping factor. The Residuals vs Fitted and Normal Q-Q graph are used to ensure.

Normality: The data follows a normal distribution. This assumption is confirmed by the usage of Shapiro-Wilk Test and Normal Q-Q graphs as well as the Residual Histogram.

Independence of observations (Multicollinearity): In multiple linear regression, it is possible that some of the independent variables are actually correlated with one another, so it is important to make sure these before developing the regression model. If two independent variables are too highly correlated ($r^2 > 0.6$), then only one of them should be used in the regression model. We will verify this by the Correlation Matrix and Variance Inflation Factor (Vif).

Homogeneity of variance (Homoscedasticity): the size of the error in our prediction doesn't change significantly across the values of the independent variable. Or simply, standard deviation are equal for all points. The Breusch-Pagan Test, Scale-Location and Residuals vs Fitted Graph are useful when supporting the validation of this assumption.

Multiple linear regression formula

$$y = \alpha + \beta_1 X_1 + \dots + \beta_n X_n + \epsilon, \text{ where:}$$

y: the predicted value of the dependent variable.

α : the y-intercept.

$\beta_i, i = 1, 2, \dots, n$: the regression coefficient of the i^{th} variable X_i .

ϵ : model error.

6.3.2 Interpreting Linear Regression Model Output in R

Consider the following example:

```
Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min       1Q   Median       3Q      Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  42.9800     2.1750  19.761  < 2e-16 ***
speed       3.9324     0.4155   9.464  1.49e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF, p-value: 1.49e-12
```

Now, we will briefly explain each component of the model output:

Formula Call: The first item shown in the output is the formula R used to fit the data.

Residuals: The Residuals section of the model output breaks it down into 5 summary points. When assessing how well the model fit the data, we should look for a symmetrical distribution across these points on the mean value zero (0). In our example, we can see that the distribution of the residuals do not appear to be strongly symmetrical. That means that the model predicts certain points that fall far away from the actual observed points.

Coefficients:

Estimate: The coefficient Estimate contains two rows. The first one is the intercept and the second row in the Coefficients is the slope

Standard Error: The coefficient Standard Error measures the average amount that the coefficient estimates vary from the actual average value of our response variable.

t-value: The coefficient t-value is a measure of how many standard deviations our coefficient estimate is far away from 0. We want it to be far away from zero as this would indicate we could reject the null hypothesis - that is, we could declare a relationship between speed and distance exist.

Pr(>t): The Pr(>t) acronym found in the model output relates to the probability of observing any value equal or larger than t. A small p-value for the intercept and the slope indicates that we can reject the null hypothesis which allows us to conclude that there is a relationship between speed and distance. Typically, a p-value of 5% or less is a good cut-off point. The 'Signif. Codes' associated to each estimate. Three stars (or asterisks) represent a highly significant p-value.

Residual Standard Error: Residual Standard Error is measure of the quality of a linear regression fit. Theoretically, every linear model is assumed to contain an error term ϵ .

Multiple R-squared, Adjusted R-squared: The R-squared (R^2) statistic provides a measure of how well the model is fitting the actual data. It takes the form of a proportion of variance. R^2 is a measure of the linear relationship between our predictor variable (speed) and our response / target variable (dist). It always lies between 0 and 1 (i.e.: a number near 0 represents a regression that does not explain the variance in the response variable well and a number close to 1 does explain the observed variance in the response variable). In multiple regression settings, the R^2 will always increase as more variables are included in the model. That's why the adjusted R^2 is the preferred measure as it adjusts for the number of variables considered.

F-Statistic: F-statistic is a good indicator of whether there is a relationship between our predictor and the response variables. The further the F-statistic is from 1 the better it is. However, how much larger the F-statistic needs to be depends on both the number of data points and the number of predictors. Generally, when the number of data points is large, an F-statistic that is only a little bit larger than 1 is already sufficient to reject the null hypothesis (H_0 : There is no relationship between speed and distance). The reverse is true as if the number of data points is small, a large F-statistic is required to be able to ascertain that there may be a relationship between predictor and response variables.

6.3.3 Breusch-Pagan Test for Heteroscedasticity in Regression Models

A Breusch-Pagan Test is used to determine if heteroscedasticity is present in a regression analysis. Derived from the Lagrange multiplier test principle, it tests whether the variance of the errors from a regression is dependent on the values of the independent variables. In that case, heteroskedasticity is present.

The Breusch-Pagan test statistic is asymptotically distributed as χ^2_{p-1} under the null hypothesis of homoscedasticity. As a result, we can calculate the test statistic

$$\chi^2 = nR^2$$

, where n is the total number of observations, R^2 is the R-squared of the new regression model that used the squared residuals as the response values.

If the p -value correspond to this Chi-Square test statistic is less than the significance level (i.e. $\alpha = 0.05$ then reject the null hypothesis and heteroscedasticity is present. Otherwise, we fail to reject the null hypothesis. In this case, homoscedasticity is assumed to present.

6.3.4 The Usage of Correlation Matrix

In statistics, we're often interested in understanding the relationship between two variables.

One way to quantify this relationship is to use the Pearson correlation coefficient, which is a measure of the linear association between two variables. It has a value between -1 and 1 where:

-1 indicates a perfectly negative linear correlation between two variables.

0 indicates no linear correlation between two variables.

1 indicates a perfectly positive linear correlation between two variables.

The further away the correlation coefficient is from zero, the stronger the relationship between the two variables.

in some cases we want to understand the correlation between more than just one pair of variables. In these cases, we can create a correlation matrix, which is a square table that shows the the correlation coefficients between several variables.

So when to use a correlation matrix

1. A correlation matrix conveniently summarizes a dataset.
2. A correlation matrix serves as a diagnostic for regression.

One key assumption of multiple linear regression is that no independent variable in the model is highly correlated with another variable in the model.

When two independent variables are highly correlated, this results in a problem known as multicollinearity and it can make it hard to interpret the results of the regression.

One of the easiest ways to detect a potential multicollinearity problem is to look at a correlation matrix and visually check whether any of the variables are highly correlated with each other.

3. A correlation matrix can be used as an input in other analyses.

6.3.5 Multicollinearity Check with VIFs

The variance inflation factor (VIF) quantifies the extent of correlation between one predictor and the other predictors in a model. It is used for diagnosing collinearity/multicollinearity. Higher values signify that it is difficult to impossible to assess accurately the contribution of predictors to a model.

The variance inflation for a variable is then computed as:

$$VIF = \frac{1}{1 - R^2}$$

where, R^2 is the R-squared statistic of the regression where the predictor of interest is predicted by all other predictor variables.

A VIF value of 1 means that the predictor is not correlated with other variables.

The higher the value, the greater the correlation of the variable with other variables. A VIF value that exceeds 5 or 10 indicates a problematic amount of collinearity

6.4 Shapiro-Wilk Test for Normality

The Shapiro-Wilk's test or Shapiro test is a normality test in frequentist statistics. The null hypothesis of Shapiro's test is that the population is distributed normally. It is among the three tests for normality designed for detecting all kinds of departure from normality.

If the value of p is equal to or less than 0.05, then the hypothesis of normality will be rejected by the Shapiro test. On failing, the test can state that the data will not fit the distribution normally with 95% confidence. However, on passing, the test can state that there exists no significant departure from normality.

Shapiro-Wilk's Test Formula

Suppose a sample, say x_1, x_2, \dots, x_n has come from a normally distributed population. Then according to the Shapiro-Wilk's tests null hypothesis test.

$$W = \frac{(\sum_{i=1}^n a_i \cdot x_i)^2}{(\sum_{i=1}^n (x_i - \bar{X}))^2}$$

where,

x_i : the i^{th} smallest number in the given sample.

$\bar{X} = \frac{x_1 + x_2 + \dots + x_n}{n}$: the sample mean.

a_i : coefficient that can be calculated as $(a_1, a_2, \dots, a_n) = \frac{m^T V^{-1}}{C}$.

Here V is the covariance matrix, m and C are the vector norms that can be calculated as $C = \|V^{-1}m\|$ and $m = (m_1, m_2, \dots, m_n)$.

6.5 Residuals vs Leverage

A residuals vs. leverage plot is a type of diagnostic plot that allows us to identify influential observations in a regression model.

Each observation from the dataset is shown as a single point within the plot. The x-axis shows the leverage of each point and the y-axis shows the standardized residual of each point.

Leverage refers to the extent to which the coefficients in the regression model would change if a particular observation was removed from the data set. Observations with high leverage have a strong influence on the coefficients in the regression model. If we remove these observations, the coefficients of the model would change noticeably.

Standardized residuals refer to the standardized difference between a predicted value for an observation and the actual value of the observation.

Let's take a look of the following example:

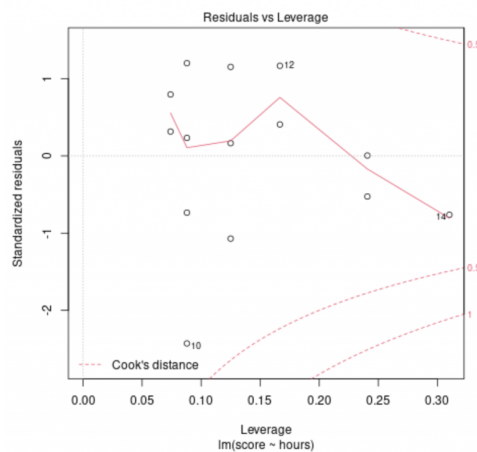


Figure 14: An example of the Residuals vs Leverage graph.

If any point in this plot falls outside of Cook's distance (the red dashed lines) then it is considered to be an influential observation. In this example, there are no points fall outside of the dashed line. This means that **this regression model does not have any influential points**

On the other hand, suppose we had the following plot:

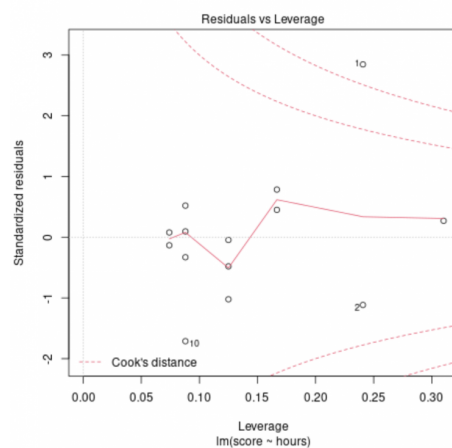


Figure 15: The existence of an influential point.

A quick glance at the graph tell us that observation number 1 in the top right corner falls outside of the red dashed lines. This indicates that it is an influential point.

6.6 Scale-Location Graph

A scale-location plot is a type of plot that displays the fitted values of a regression model along the x-axis and the the square root of the standardized residuals along the y-axis.

When looking at this plot, we check two things:

1. Verify that the red line is roughly horizontal across the plot. If it is, then the assumption of homoscedasticity is likely satisfied for a given regression model. That is, the spread of the residuals is roughly equal at all fitted values.
2. Verify that there is no clear pattern among the residuals. In other words, the residuals should be randomly scattered around the red line with roughly equal variability at all fitted values.

Let's consider the following example. We can observe two points from the Scale-Location plot for this regression model.

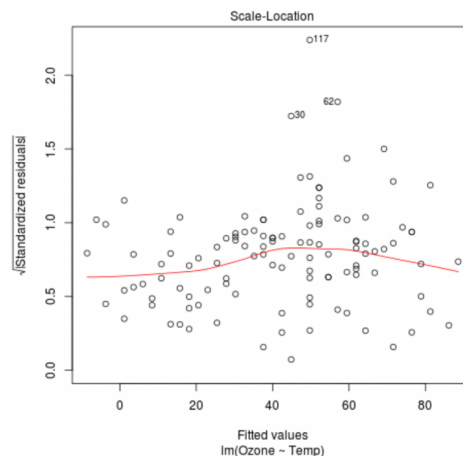


Figure 16: An example of the Scale-Location plot.

Firstly, the red line is roughly horizontal across the plot. This proves that the assumption of homoscedasticity is satisfied for a given regression model.

Secondly, the residuals should be randomly scattered around the red line with roughly equal variability at all fitted values.

6.7 Residuals vs Fitted Graph

The Residual vs Fitted plot allows us to detect several types of violations in the linear regression assumptions.

In the plot, the fitted values \hat{y} is sketched on the x-axis and the residuals $y - \hat{y}$ are represented on the y-axis. The Residuals and Fitted plot is mainly useful for investigating:

1. Whether Linearity holds. This is indicated by the mean residual value for every fitted value region being close to 0. In R this is indicated by the red line being close to the dashed line.
2. Whether Homoscedasticity holds. The spread of residuals should be approximately the same across the x-axis.
3. Whether there are outliers. This is indicated by some 'extreme' residuals that are far from the rest.

Let's take a look of the following plot for the car data set.

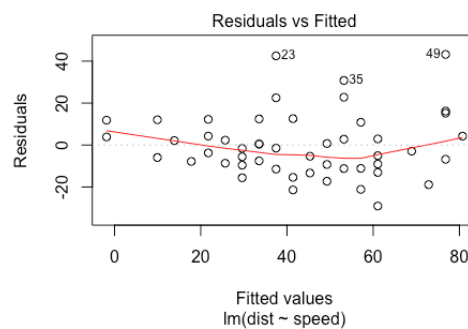


Figure 17: An example of the Residuals vs Fitted plot.

Here we see that linearity seems to hold reasonably well, as the red line is close to the dashed line. We can also note the heteroscedasticity: as we move to the right on the x-axis, the spread of the residuals seems to be increasing. Finally, the observations 23, 35, 49 may be outliers.

Considering another example:

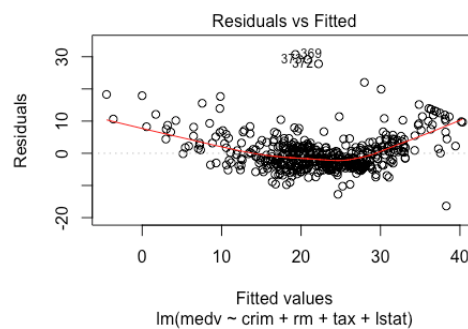


Figure 18: An example of the Residuals vs Fitted plot.

In this plot, the linearity is violated, there seems to be a quadratic relationship. Whether there is homoscedasticity or not is less obvious, we will need to investigate the others plot.

6.8 Normal Q-Q Graph

The Q-Q plot, or quantile-quantile plot, is a graphical tool to help us assess if a set of data plausibly came from some theoretical distribution such as a Normal or exponential.

For example, if we run a statistical analysis that assumes our dependent variable is Normally distributed, we can use a Normal Q-Q plot to check that assumption.

However, It's just a visual check, not an air-tight proof, so it is somewhat subjective. But it allows us to see at-a-glance if our assumption is plausible, and if not, how the assumption is violated and what data points contribute to the violation.

A Q-Q plot is a scatterplot created by plotting two sets of quantiles against one another. If both sets of quantiles came from the same distribution, we should see the points forming a line that's roughly straight. Here's an example of a Normal Q-Q plot when both sets of quantiles truly come from Normal distributions.

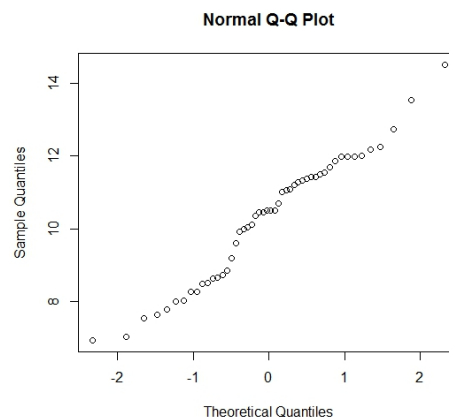


Figure 19: An example of the Normal Q-Q plot.

7 ANOVA Test for The Effect of The Numbers of Cores and Threads on other CPU Specifications

7.1 Overview

In this section, we will successively test the effects of 13 levels of cores from 2 to 38 and 15 levels of threads from 2 to 76 on other crucial factors that will extensively impact the performance of the CPU as well as the whole computer system.

In reality, characteristics such as bus speed, base frequency, turbo frequency, and maximum temperature are mathematically computed from the CPU's operating data to demonstrate the CPU's overall performance. As a result, we can only modify these factors' figures if we adjust the hardware itself, which is the cores and threads of the processors.

Cache capacity and maximum memory size (RAM limit) are also worthwhile considerations at first. Increasing the cache size and RAM limit to boost computer performance is not, however, a practical option.

To begin with, the cache size factor in our data set refers to the size of the CPU cache, which is a hardware component that works in tandem with the CPU. It aids the CPU in lowering the average cost of accessing data from the main memory (time or energy). When the volume of data exchanged between the CPU and main memory is significant, a big cache size is advantageous. However, if the data is tiny and the cache capacity is large, our CPU will spend more time looking for the cache address where the needed data is stored. In this instance, the processing speed will be slowed. We must always balance cache size with CPU speed and data size from main memory if we wish to improve performance.

Hardware, software, and cost issues all limit the amount of memory (RAM) that may be added in a computer system. The processor package or system design may limit the number of address bus bits available to the device. An operating system may only be intended to allocate a particular amount of memory due to software limitations on usable physical RAM, or it may rely on internal data structures with set addressable memory limits. There may be no financial benefit to a manufacturer in offering extra memory for mass-market personal computers. Because memory devices were relatively expensive in comparison to processors, the RAM given with the system was frequently substantially less than the hardware's address capacity.

As a result, we'll concentrate on just four variables: base frequency, maximum turbo frequency, junction temperature (or the highest temperature permitted at the processor die), and bus speed. These characteristics are practically modifiable and actually affect the computer's performance because they are plainly directly affected by cores and threads.

7.2 Effects on The Processor Base Frequency

Firstly, we are going to perform the one-way ANOVA to investigate the effect of only the number of cores or the numbers of threads on the processor base frequency.

We will use the following code segment to run the models:

```
> oneway.Cores = aov(base_frequency ~ cores, data = newIntel)
> summary(oneway.Cores)

> oneway.Threads = aov(base_frequency ~ threads, data = newIntel)
> summary(oneway.Threads)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
cores	1	12850387	12850387	23.96	1.32e-06	***
Residuals	519	278368193	536355			

Signif. codes:	0	'***'	0.001	'**'	0.01	'*' 0.05 '.' 0.1 ' ' 1

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
threads	1	9148558	9148558	16.83	4.74e-05	***
Residuals	519	282070022	543488			

Signif. codes:	0	'***'	0.001	'**'	0.01	'*' 0.05 '.' 0.1 ' ' 1

Figure 20: The results of the two models.

The results have shown that the p-value for both the cores and threads variables is extremely

low ($p < 0.0001$). It means that different numbers of cores or threads used in CPUs has a real impact on the processor base frequency.

Next, we will combine both cores and threads:

```
> twoway = aov(base_frequency ~ cores + threads, data = newIntel)
> summary(twoway)
```

```
      Df Sum Sq Mean Sq F value    Pr(>F)
cores    1 12850387 12850387   24.270 1.13e-06 ***
threads  1  4104143  4104143    7.751  0.00556 **
Residuals 518 274264050  529467
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 21: Two-way ANOVA.

Constructing a two-way ANOVA has reduced the residual variance (the residual sum of squares) and both cores and threads are statistically significant ($p\text{-value} < 0.05$).

Up to now, we currently have 3 different ANOVA models. In order to find out the best-fit one we will perform the AIC test:

```
> install.packages("raster")
> install.packages("AICcmodavg")
> library(AICcmodavg)

> model.set <- list(oneway.Cores, oneway.Threads, twoway)
> model.names <- c("oneway.Cores", "oneway.Threads", "twoway")

> aictab (model.set, modnames = model.names)
```

```
Model selection based on AICc:

      K   AICc Delta_AICc AICcwt Cum.wt    LL
twoway    4 8350.19      0.00   0.94  0.94 -4171.06
oneway.Cores 3 8355.90      5.71   0.05  1.00 -4174.92
oneway.Threads 3 8362.78     12.59   0.00  1.00 -4178.37
```

Figure 22: The result of AIC test.

The outcome shows that the 'twoway' model is the best-fit. It has the lowest AIC score, and 94% of the AIC Weight, which means that it explains 94% of the total variation in the dependent variable that can be explained by the full set of models.

After having the best-fit model, we are going to verify ANOVA's assumption by numerous tests and plots.

The following code segment will sequentially plot all necessary graphs for our research.

```
> plot(twoway)

> h <- hist(twoway[[ 'residuals']], breaks = 5, density = 40,
           col = "red", xlab="Residuals",
```

```
main="Residuals_histogram_overlayed_by_normal_distribution")

> xfit<-seq(min(twoWay[['residuals']]),max(twoWay[['residuals']]),
  length = 40)

> yfit<-dnorm(xfit,mean=mean(twoWay[['residuals']]),
  sd=sd(twoWay[['residuals']]))

> yfit<-yfit * diff(h$mids[1:2]) * length(twoWay[['residuals']])

> lines(xfit, yfit, col = "black", lwd = 2)
```

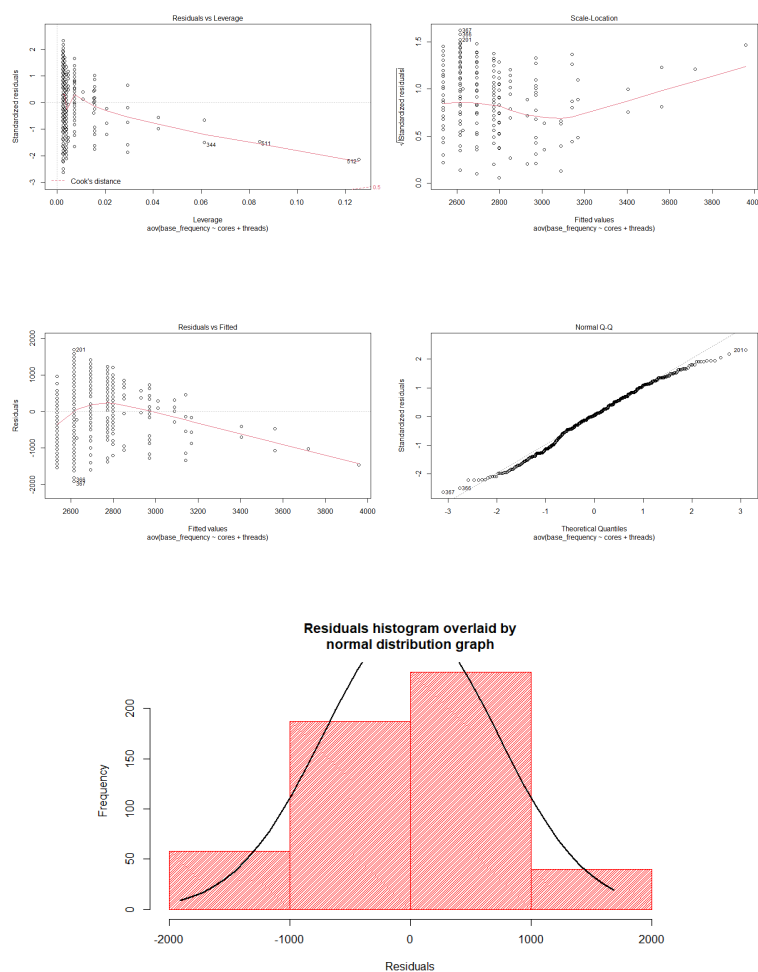


Figure 23: The result

First of all from the Residuals vs Leverage plot we can see that there is no influential point. The Normal-QQ and Residuals Histogram show that the dependent variable seems to follow normal distribution. However, the assumptions of homogeneity has been violated.

Although the spread of residuals are approximately symmetric across the x-axis of the Residuals vs Fitted plot, the red line in the Scale-Location graph does not seem to be roughly horizontal across the graph.

To confirm our quick inspect at the above plot, we need to perform the Shapiro-Wilk test for normality and Levene's Test for homoscedasticity.

```
> shapiro.test(twoway[['residuals']])
> leveneTest(base_frequency~interaction(cores, threads), data=newIntel)
```

```
Shapiro-wilk normality test
data: twoway[["residuals"]]
W = 0.98704, p-value = 0.0001391

Levene's Test for Homogeneity of Variance (center = median)
Df F value Pr(>F)
group 16 2.858 0.0001731 ***
504
---
signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 24: The result

Unfortunately, p -values obtained from Shapiro-Wilk test and Levene's Test are both significant ($p < 0.05$). We conclude that the data is not normally distributed and does not have equal variance.

Therefore, the ANOVA test is no longer suitable. In this case, Kruskal-Wallis test is more appropriate for analyzing differences among cores and threads.

```
> kruskal.test(base_frequency~interaction(cores, threads), data=newIntel)

> library(rcompanion)
> epsilonSquared(x=newIntel$base_frequency,
  g=interaction(newIntel$cores, newIntel$threads))
```

```
Kruskal-wallis rank sum test
data: base_frequency by interaction(cores, threads)
Kruskal-wallis chi-squared = 96.876, df = 16, p-value = 1.33e-13

epsilon.squared
0.186
```

Figure 25: The result

From Kruskal-Wallis test, the p -value < 0.05 indicates that there are significant differences in the base frequency among the cores and threads.

Besides, the $\epsilon^2 = 0.186$ suggests a relatively strong effect of the numbers of cores and threads on the base frequency.

To take a deeper insight on which parts of cores and threads are statistically significant different from each other, we will perform the Dunn's Test as post-hoc test for significant Kruskal-Wallis test.


```
> install.packages("FSA")
> library(FSA)
> dunnTest(base_frequency~interaction(cores,threads),
           data=newIntel,method="bh")
```

Dunn (1964) kruskal-wallis multiple comparison
p-values adjusted with the Benjamini-Hochberg method.

	comparison	P_unadj	P_adj						
1	10.20 - 12.24	-8.617398e-01	3.888307e-01	5.29190e-01	35	2.4 - 28.56	-1.056994e+00	2.905143e-01	8.063254e-01
2	10.20 - 14.28	-4.870452e-01	6.262263e-01	8.690487e-01	36	24.48 - 28.56	2.035638e-01	8.386944e-01	9.198584e-01
3	12.24 - 14.28	2.081378e-01	8.351214e-01	9.233862e-01	37	10.20 - 32.64	6.682785e-01	5.039559e-01	8.358292e-01
4	10.20 - 16.32	-1.062092e-01	9.154163e-01	9.650910e-01	38	12.24 - 32.64	1.005348e+00	3.147294e-01	8.392785e-01
5	12.24 - 16.32	5.312879e-01	5.952193e-01	8.521034e-01	39	14.28 - 32.64	8.510898e-01	3.947195e-01	8.387789e-01
6	14.28 - 16.32	2.949943e-01	7.679982e-01	9.082414e-01	40	16.32 - 32.64	6.645190e-01	8.063582e-01	8.296953e-01
7	10.20 - 18.36	1.001244e+00	3.167089e-01	8.283157e-01	41	18.36 - 32.64	1.681141e-01	8.664935e-01	9.427450e-01
8	12.24 - 18.36	1.489175e+00	1.364414e-01	8.672604e-01	42	2 - 32.64	-6.161197e-01	5.378155e-01	8.605048e-01
9	14.28 - 18.36	1.143954e+00	2.526428e-01	7.990562e-01	43	2.4 - 32.64	-5.900590e-01	5.55111e-01	8.579608e-01
10	16.32 - 18.36	8.330026e-01	4.048433e-01	8.217714e-01	44	24.48 - 32.64	1.669719e-01	8.673921e-01	9.362328e-01
11	10.20 - 2.2	1.940848e+00	5.227676e-02	3.949800e-01	45	28.56 - 32.64	2.303061e-02	9.816259e-01	9.962770e-01
12	12.24 - 2.2	2.254130e+00	2.418801e-02	2.193046e-01	46	10.20 - 38.76	9.926855e-01	3.208633e-01	8.081000e-01
13	14.28 - 2.2	1.970077e+00	4.882954e-02	3.906363e-01	47	12.24 - 38.76	1.313107e+00	1.891468e-01	7.145466e-01
14	16.32 - 2.2	1.792915e+00	8.377058e-02	5.178545e-01	48	14.28 - 38.76	1.484144e+00	2.507978e-01	8.121072e-01
15	18.36 - 2.2	1.122020e+00	2.618540e-01	8.093668e-01	49	16.32 - 38.76	9.618430e-01	3.361285e-01	8.163121e-01
16	10.20 - 2.4	5.199008e+00	2.003549e-07	6.812065e-06	50	18.36 - 38.76	4.715690e-01	6.732444e-01	8.580582e-01
17	12.24 - 2.4	3.989955e+00	6.608571e-05	1.497943e-03	51	2 - 38.76	-3.447014e-01	7.303189e-01	9.029397e-01
18	14.28 - 2.4	3.069266e+00	2.470547e-03	3.599444e-02	52	2.4 - 38.76	-2.593586e-01	7.953585e-01	9.245193e-01
19	16.32 - 2.4	2.617810e+00	8.840285e-03	9.248299e-02	53	24.48 - 38.76	4.548546e-01	6.492138e-01	8.489719e-01
20	18.36 - 2.4	1.694492e+00	9.017175e-02	5.109732e-01	54	28.56 - 38.76	3.109133e-01	7.558665e-01	9.017355e-01
21	2 - 2.4	-2.260043e-01	8.211981e-01	9.306912e-01	55	32.64 - 38.76	2.350552e-01	8.146159e-01	9.304753e-01
22	10.20 - 24.48	7.946178e-01	4.268358e-01	8.062455e-01	56	10.20 - 4.4	8.119150e-01	4.168404e-01	8.336808e-01
23	12.24 - 24.48	1.263030e+00	2.065784e-01	7.203759e-01	57	12.24 - 4.4	1.426427e+00	1.537321e-01	6.744376e-01
24	14.28 - 24.48	9.934300e-01	3.205004e-01	8.224162e-01	58	14.28 - 4.4	9.221868e-01	3.564311e-01	8.216040e-01
25	16.32 - 24.48	7.203184e-01	4.713290e-01	8.114018e-01	59	16.32 - 4.4	5.180421e-01	6.044289e-01	8.562742e-01
26	18.36 - 24.48	-1.183474e-02	9.905575e-01	9.879494e-01	60	18.36 - 4.4	-6.264579e-01	5.310146e-01	8.597379e-01
27	2 - 24.48	-1.037816e+00	2.993557e-01	8.124757e-01	61	2 - 4.4	-1.71188		

Figure 26: The result

In the 'Comparison' column the notation 10.20 – 12.24 means that "the cores number = 10

+ the threads number = 20 contrasted with the cores number = 12 + the threads number = 24".

From the table we can see that these following combinations are statistically significant difference from one another ($p - value < 0.05$).

10.20 - 2.4
12.24 - 2.4
14.28 - 2.4
2.4 - 4.4
10.20 - 4.8
2.4 - 4.8
4.4 - 4.8
2.4 - 6.12
4.8 - 6.12
2.4 - 6.6
2.4 - 8.16
4.8 - 8.16

Now we may want to visualize the effects of the numbers of cores and threads on the base frequency.

We will mark 3 labels for the significantly difference pairs found above. Let's say we use 'a' represents 2.4, 'b' represents all the intermediate pairs and 'c' represents 14.28.

Then we need to make an additional data frame so that we can add these groupwise differences to our group.

```
# Summarise the original data
> install.packages("dplyr")
> library(dplyr)
> meanBase_Freq <- newIntel %>%
+   group_by(cores, threads) %>%
+   summarise(
+     base_frequency = mean(base_frequency)
+   )
# Add the group labels
> meanBase_Freq$group <-
+   c("r", "a", "b", "b", "b", "b", "r", "b", "b", "b", "c", "r", "r", "r", "r", "r", "r")
> meanBase_Freq
```

```
# A tibble: 17 x 4
# Groups:   cores [13]
  cores threads base_frequency group
  <int>   <int>         <dbl> <chr>
1     2     2         2150    r
2     2     4         2162.    a
3     4     4         2970.    b
4     4     8         2607.    b
5     6     6         2828    b
6     6    12         3007.    b
7     8     8         2800    r
8     8    16         2960.    b
9    10    20         3080    b
10    12    24         3367.    b
11    14    28         3250    c
12    16    32         3125    r
13    18    36         2780    r
14    24    48         2800    r
15    28    56         2700    r
16    32    64         2700    r
17    38    76         2500    r
```

Figure 27: The result

As shown above, the data frame contains all available pairs of cores and threads. However, we only need those that are significantly difference. So we mark the rest as 'r' represents redundant. After that, we are going to remove the pairs which are labeled as 'r'.

```
> meanBase_Freq = subset(meanBase_Freq, meanBase_Freq$group!="r")
> meanBase_Freq
```

```
# A tibble: 9 x 4
# Groups:   cores [7]
  cores threads base_frequency group
  <int>   <int>         <dbl> <chr>
1     2     4         2162.    a
2     4     4         2970.    b
3     4     8         2607.    b
4     6     6         2828    b
5     6    12         3007.    b
6     8    16         2960.    b
7    10    20         3080    b
8    12    24         3367.    b
9    14    28         3250    c
```

Figure 28: The result

Now we are ready to start making the plot.

```
> install.packages("ggpubr")
> library(ggpubr)

# Plot the raw data
> twowayPlot <- ggplot(newIntel,
  aes(x=cores, y=base_frequency, group=threads))
  +geom_point(cex=1.5, pch=1.0,
```

```
position=position_jitter(w=0.1,h=0))

# Add means and se to the graph
> twowayPlot <- twowayPlot+
  stat_summary(fun.data = 'mean_se',geom='errorbar',width=0.2)+
  stat_summary(fun.data='mean_se', geom='pointrange')+
  geom_point(data=meanBase_Freq,aes(x=cores,y=base_frequency))

# Split up the data over the levels of threads
> twowayPlot <- twowayPlot+
  geom_text(data=meanBase_Freq, label=meanBase_Freq$group,
    vjust = -8, size = 5)+
  facet_wrap(~ threads)

# Generate title
> twowayPlot <- twowayPlot+
  theme_classic2()+
  labs(title = "Base_Frequency_response_to_the_number_of_cores_and_threads",
    x="The_number_of_cores", y="Base_Frequency")

> twowayPlot
```

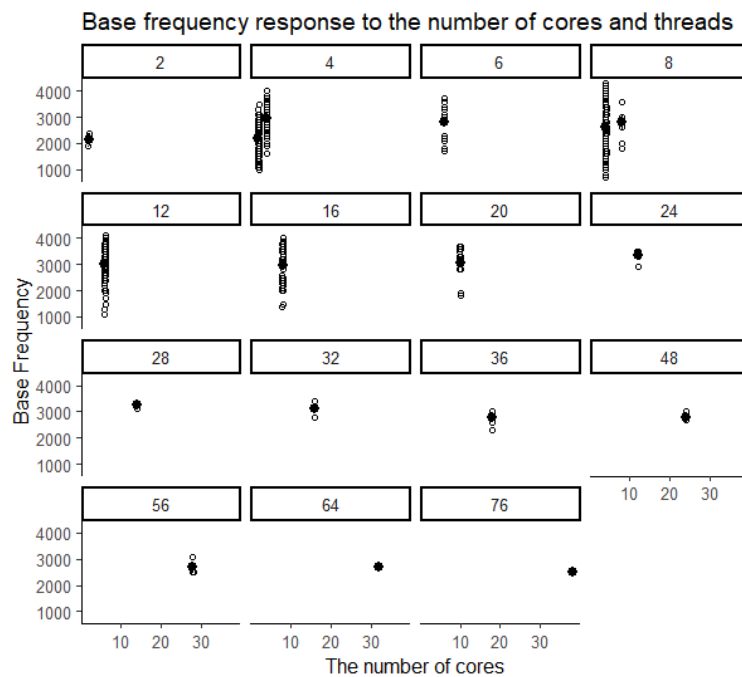


Figure 29: The result

7.3 Effects on The Turbo Frequency

Similar with the above section, the one-way ANOVA is established to investigate the effect of only the number of cores or the numbers of threads on the processor turbo frequency.

```

      Df    Sum Sq Mean Sq F value Pr(>F)
cores      1 44837105 44837105   119.9 <2e-16 ***
Residuals 519 194089113   373967
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> oneway.Threads = aov(turbo_frequency ~ threads, data = newIntel)
> summary(oneway.Threads)
      Df    Sum Sq Mean Sq F value Pr(>F)
threads      1 45099166 45099166   120.8 <2e-16 ***
Residuals 519 193827053   373463
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Figure 30: The results of the two models.

The ($p - value < 0.0001$) means that different numbers of cores or threads used in CPUs has effect on the processor turbo frequency.

Next, we will combine both cores and threads:

```

> twoway = aov(turbo_frequency ~ cores + threads, data = newIntel)
> summary(twoway)
      Df    Sum Sq Mean Sq F value Pr(>F)
cores      1 44837105 44837105 120.126 <2e-16 ***
threads      1   745334   745334   1.997 0.158
Residuals 518 193343780   373251
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Figure 31: Two-way ANOVA.

In a two-way ANOVA model, threads has ($p - value > 0.05$), which indicates that there is a difference in group mean, so this model is no longer perfect. We will not take this model in to account when calculating the AIC score.

Until now, we only have 2 different ANOVA models.

```

Model selection based on AICc:

      K    AICc Delta_AICc AICcwt Cum.wt      LL
oneway.Threads 3 8167.31      0.0   0.59   0.59 -4080.63
oneway.Cores   3 8168.01      0.7   0.41   1.00 -4080.98

```

Figure 32: The result of AIC test.

The outcome shows that the 'oneway.Threads' model is the best-fit.

After having the best-fit model, we are going to verify ANOVA's assumption by numerous tests and plots.

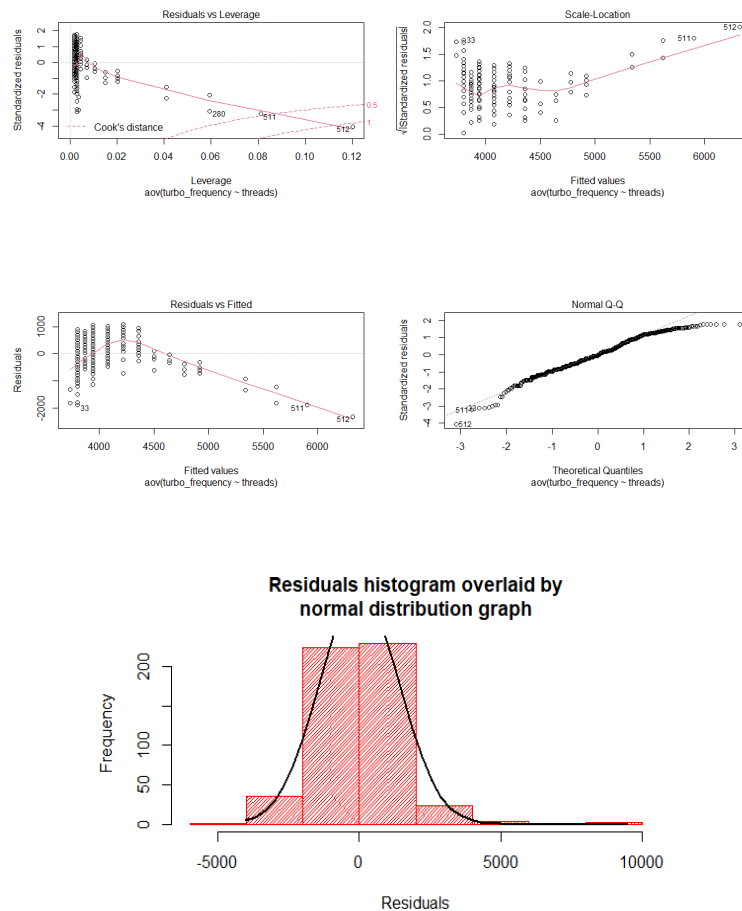


Figure 33: The result

We can see that there is no influential point. The dependent variable seem to follow normal distribution. However, the assumptions of homogeneity has been violated.

To confirm our quick inspect at the above plot, we need to perform the Shapiro-Wilk test for normality and Levene's Test for homoscedasticity.

```

> shapiro-wilk normality test
data:  oneway.Threads[["residuals"]]
W = 0.97526, p-value = 1.038e-07

> # check for homogeneity
> library(car)
Loading required package: carData
> leveneTest(turbo_frequency ~ interaction(cores,threads), data=newIntel)
Levene's Test for Homogeneity of variance (center = median)
      Df F value Pr(>F)
group 16  1.4365 0.1196
504

```

Figure 34: The result

p -values obtained from Shapiro-Wilk test is significant ($p < 0.05$). We conclude that the data is not normally distributed. But, we have sufficient evidence to say that the model satisfies homoscedasticity assumption.

Anyway, the ANOVA test is no longer suitable. Kruskal-Wallis test will then be used for analyzing differences among threads.

```

Kruskal-wallis rank sum test

data: turbo_frequency by interaction(threads)
Kruskal-wallis chi-squared = 288.6, df = 14, p-value < 2.2e-16

> library(rcompanion)
> epsilonsquared(x=newIntel$turbo_frequency, g = interaction(newIntel$threads))
epsilon.squared
0.555

```

Figure 35: The result

From Kruskal-Wallis test, the p -value < 0.05 indicates that there are significant differences in the turbo frequency among the threads.

With $\epsilon^2 = 0.555$, recommending a strong effect of the numbers threads on the turbo frequency.

The Dunn's Test as post-hoc test for significant Kruskal-Wallis test will be used for further analysis on which parts of threads are statistically significant difference from the others.

[illegible]

```

73 32 - 64 0.32935127 7.418902e-01 8.953847e-01
74 36 - 64 0.59610324 5.511063e-01 8.266594e-01
75 4 - 64 -0.90132089 3.674177e-01 6.651528e-01
76 48 - 64 0.33592942 7.369241e-01 9.103180e-01
77 56 - 64 0.21019584 8.335148e-01 9.410651e-01
78 6 - 64 0.10263339 9.182539e-01 9.452614e-01
79 12 - 76 0.69774708 4.853354e-01 8.219390e-01
80 16 - 76 1.23322732 2.174910e-01 5.190125e-01
81 2 - 76 -1.40079535 1.612753e-01 4.342027e-01
82 20 - 76 1.10482396 2.692359e-01 5.653953e-01
83 24 - 76 0.51354683 6.075689e-01 8.620910e-01
84 28 - 76 0.66093292 5.086553e-01 8.345127e-01
85 32 - 76 0.32935127 7.418902e-01 9.057962e-01
86 36 - 76 0.59610324 5.511063e-01 8.386400e-01
87 4 - 76 -0.90132089 3.674177e-01 6.768221e-01
88 48 - 76 0.33592942 7.369241e-01 9.211551e-01
89 56 - 76 0.21019584 8.335148e-01 9.512941e-01
90 6 - 76 0.10263339 9.182539e-01 9.546204e-01
91 64 - 76 0.00000000 1.000000e+00 1.000000e+00
92 12 - 8 5.84082726 5.194224e-09 9.089892e-08
93 16 - 8 8.58289041 9.251849e-18 3.238147e-16
94 2 - 8 -2.25205628 2.431872e-02 9.821020e-02
95 20 - 8 5.25501272 1.480142e-07 2.220214e-06
96 24 - 8 1.60693545 1.080685e-01 3.242056e-01
97 28 - 8 1.68397371 9.218675e-02 2.846944e-01
98 32 - 8 0.95152209 3.413394e-01 6.400114e-01
99 36 - 8 1.68780977 9.144775e-02 2.909701e-01
100 4 - 8 -7.06132623 1.649209e-12 3.463339e-11
101 48 - 8 0.86029730 3.896252e-01 6.934007e-01
102 56 - 8 0.61112668 5.411157e-01 8.355463e-01
103 6 - 8 1.01470678 3.102456e-01 6.264575e-01
104 64 - 8 0.11302460 9.100111e-01 9.651632e-01
105 76 - 8 0.11302460 9.100111e-01 9.750118e-01

```

Figure 36: The result

7.4 Effects on The Maximum Temperature

In this part, we still follow the same structure with the previous models but the "max_temp" factor will be the one to be examined:

```

> oneway.Cores = aov (max_temp~cores, data = newIntel)
> summary(oneway.Cores)
      Df Sum Sq Mean Sq F value    Pr(>F)
cores    1   4352     4352   33.59 1.18e-08 ***
Residuals 519  67236      130
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> oneway.Threads = aov (max_temp~threads, data = newIntel)
> summary(oneway.Threads)
      Df Sum Sq Mean Sq F value    Pr(>F)
threads    1   3180     3180   24.13 1.21e-06 ***
Residuals 519  68408      132
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Figure 37: The result of two models.

The results have indicated that different numbers of cores or threads used in CPUs has a real influence on the processor base frequency.

Next, we will combine both cores and threads:

```

Df Sum Sq Mean Sq F value Pr(>F)
cores      1    4352      4352    34.12 9.18e-09 ***
threads    1    1162      1162     9.11 0.00267 **
Residuals 518    66074        128
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Figure 38: Two-way ANOVA on max_temp.

Two-way ANOVA model shows that both cores and threads are statistically significant ($p - value < 0.05$).

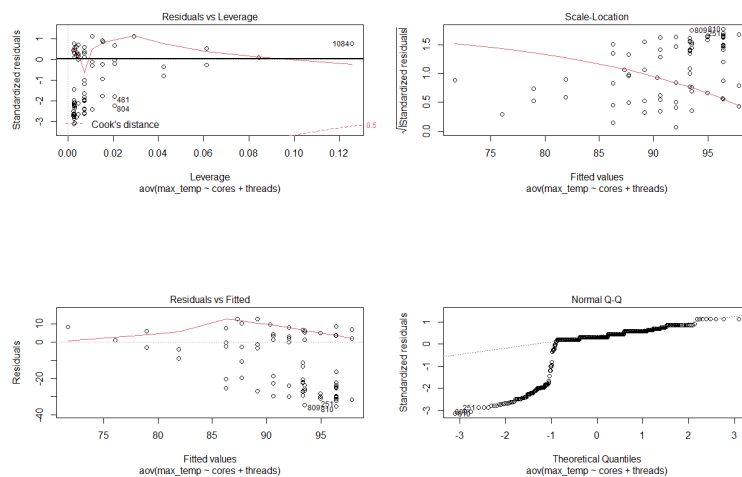
Now, we will find out which one is the best-fit:

Model selection based on AICc:

	K	AICc	Delta_AICc	AICcwt	Cum.Wt	LL
twoway	4	4009.70	0.00	0.97	0.97	-2000.81
oneway.Cores	3	4016.75	7.05	0.03	1.00	-2005.35
oneway.Threads	3	4025.75	16.05	0.00	1.00	-2009.85

Figure 39: The result of AIC test.

It appears that the 'twoway' model is the best-fit. ANOVA's assumptions are going to be confirmed by numerous tests and plots.



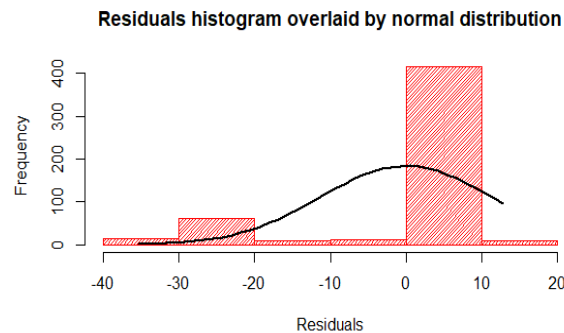


Figure 40: The result

Quickly glance at those diagrams above, it is obvious that the model does not follow normal distribution as well as fail the homogeneity's assumption.

To confirm our brief conclusion, we need to perform the Shapiro-Wilk test for normality and Levene's Test for homoscedasticity.

```
> shapiro.test(twoway[['residuals']])
Shapiro-Wilk normality test
data:  twoway[["residuals"]]
W = 0.67094, p-value < 2.2e-16

> library(car)
> leveneTest(max_temp~interaction(cores,threads),data=newIntel)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value    Pr(>F)
group 16  5.587 4.008e-11 ***
      504
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 41: The result

As the result, Kruskal-Wallis test is more appropriate for analyzing differences among cores and threads.

```
> kruskal.test(max_temp~interaction(cores,threads),data=newIntel)
Kruskal-wallis rank sum test
data:  max_temp by interaction(cores, threads)
Kruskal-Wallis chi-squared = 146.51, df = 16, p-value < 2.2e-16

> epsilonSquared(x=newIntel$max_temp,
+               g=interaction(newIntel$cores,newIntel$threads))
epsilon.squared
0.282
```

Figure 42: Kruskal-Wallis result.

The Kruskal-Wallis test helps us to conclude that there are significant differences in the maximum temperature among the cores and threads. And the $\epsilon^2 = 0.282$ which between (0.16;0.36) suggests a relatively strong effect of the numbers of cores and threads on the base frequency.

We continue performing the Dunn's Test as post-hoc test for significant Kruskal-Wallis test.

	Comparison	Z	P.unadj	P.adj
1	10.20 - 12.24	3.130070016	1.747646e-03	7.922664e-03
2	10.20 - 14.28	0.969333895	3.323786e-01	4.808882e-01
3	12.24 - 14.28	-1.434618504	1.513958e-01	2.606308e-01
4	10.20 - 16.32	2.477807390	1.321925e-02	3.908300e-02
5	12.24 - 16.32	-0.154636298	8.771081e-01	1.000000e+00
6	14.28 - 16.32	1.168458545	2.426219e-01	3.707480e-01
7	10.20 - 18.36	3.03802992	2.439043e-03	1.005181e-02
8	12.24 - 18.36	0.096499244	9.231241e-01	1.000000e+00
9	14.28 - 18.36	1.467569362	1.422212e-01	2.511959e-01
10	16.32 - 18.36	0.235905910	8.135057e-01	9.878283e-01
11	10.20 - 2.2	-0.810803766	4.174784e-01	5.853305e-01
12	12.24 - 2.2	-2.520863548	1.170672e-02	3.538032e-02
13	14.28 - 2.2	-1.307391447	1.910798e-01	3.130946e-01
14	16.32 - 2.2	-2.261433854	2.373240e-02	6.724181e-02
15	18.36 - 2.2	-2.529951946	1.140781e-02	3.526052e-02
16	10.20 - 2.4	-3.234826794	1.217165e-03	6.621380e-03
17	12.24 - 2.4	-5.351745070	8.711006e-08	3.948989e-06
18	14.28 - 2.4	-2.598497222	9.363280e-03	2.961410e-02
19	16.32 - 2.4	-4.217560876	2.469592e-05	3.053313e-04
20	18.36 - 2.4	-5.036963704	4.729744e-07	1.072075e-05
21	2.2 - 2.4	-0.271271688	7.861821e-01	9.720069e-01
22	10.20 - 24.48	2.173168059	2.976767e-02	8.262047e-02
23	12.24 - 24.48	-0.157649169	8.747333e-01	1.000000e+00
24	14.28 - 24.48	1.066519163	2.861890e-01	4.230620e-01
25	16.32 - 24.48	-0.015263244	9.878222e-01	1.000000e+00
26	18.36 - 24.48	-0.232656097	8.160285e-01	9.821228e-01
27	2.2 - 24.48	2.132614381	3.295637e-02	8.788366e-02
28	2.4 - 24.48	3.651032465	2.611882e-04	1.973422e-03
29	10.20 - 28.56	2.133157383	3.291182e-02	8.952014e-02
30	12.24 - 28.56	-0.192682317	8.472078e-01	1.000000e+00
31	14.28 - 28.56	1.034084770	3.010965e-01	4.403132e-01
32	16.32 - 28.56	-0.047697637	9.619572e-01	1.000000e+00
33	18.36 - 28.56	-0.266576798	7.897950e-01	9.676768e-01
34	2.2 - 28.56	2.105477821	3.524974e-02	9.045215e-02
35	2.4 - 28.56	3.608780898	3.076393e-04	1.992331e-03
36	24.48 - 28.56	-0.030339597	9.757962e-01	1.000000e+00
37	10.20 - 32.64	1.311645142	1.896399e-01	3.145247e-01
38	12.24 - 32.64	-0.104554700	9.167292e-01	1.000000e+00
39	14.28 - 32.64	0.727267943	4.670618e-01	6.416203e-01
40	16.32 - 32.64	-0.011730128	9.906409e-01	1.000000e+00
41	18.36 - 32.64	-0.156434288	8.756907e-01	1.000000e+00
42	2.2 - 32.64	1.588367121	1.122033e-01	2.119397e-01
43	2.4 - 32.64	2.128095226	3.332919e-02	8.716865e-02
44	24.48 - 32.64	-0.001261961	9.989931e-01	1.000000e+00
45	28.56 - 32.64	0.020191374	9.838907e-01	1.000000e+00
46	10.20 - 38.76	1.264716872	2.059728e-01	3.334798e-01
47	12.24 - 38.76	-0.149074766	8.814946e-01	9.907708e-01
48	14.28 - 38.76	0.684257473	4.938126e-01	6.649358e-01
49	16.32 - 38.76	-0.054740598	9.563451e-01	1.000000e+00
50	18.36 - 38.76	-0.200331665	8.412212e-01	1.000000e+00
51	2.2 - 38.76	1.549104113	1.213567e-01	2.206016e-01
52	2.4 - 38.76	2.080256573	3.750201e-02	9.273223e-02
53	24.48 - 38.76	-0.042906669	9.657759e-01	1.000000e+00
54	28.56 - 38.76	-0.021453335	9.828840e-01	1.000000e+00
55	32.64 - 38.76	-0.034002762	9.728749e-01	1.000000e+00
56	10.20 - 4.4	2.365322212	1.801439e-02	5.212675e-02
57	12.24 - 4.4	-1.980804288	4.761323e-02	1.136035e-01
58	14.28 - 4.4	0.152179258	8.790456e-01	9.962516e-01
59	16.32 - 4.4	-1.448618791	1.474441e-01	2.570820e-01
60	18.36 - 4.4	-1.947479642	5.147726e-02	1.207053e-01
61	2.2 - 4.4	1.684901469	9.200758e-02	1.895914e-01
62	2.4 - 4.4	8.578249608	9.632774e-18	1.310057e-13
63	24.48 - 4.4	-1.244591324	2.132820e-01	3.412512e-01
64	28.56 - 4.4	-1.202702349	2.290915e-01	3.622834e-01
65	32.64 - 4.4	-0.728617521	4.662357e-01	6.470209e-01
66	38.76 - 4.4	-0.680919730	4.959223e-01	6.612297e-01
67	10.20 - 4.8	-1.499621720	1.337124e-01	2.392749e-01
68	12.24 - 4.8	-4.358041957	1.312312e-05	1.983050e-04
69	14.28 - 4.8	-1.751996360	7.977443e-02	1.695207e-01
70	16.32 - 4.8	-3.383528352	7.156082e-04	4.423760e-03
71	18.36 - 4.8	-4.119345544	3.799500e-05	4.306100e-04
72	2.2 - 4.8	0.344275130	7.306394e-01	9.200644e-01
73	2.4 - 4.8	3.378638995	7.284560e-04	4.307392e-03
74	24.48 - 4.8	-2.919480969	3.506148e-03	1.402459e-02
75	28.56 - 4.8	-2.876983594	4.014965e-03	1.560101e-02
76	32.64 - 4.8	-1.694881095	9.009798e-02	1.885127e-01
77	38.76 - 4.8	-1.646948302	9.956867e-02	2.021095e-01
78	4.4 - 4.8	-6.389685257	1.662275e-10	1.130347e-08
79	10.20 - 6.12	-1.095114431	2.734665e-01	4.086972e-01
80	12.24 - 6.12	-4.083697359	4.432475e-05	4.305833e-04
81	14.28 - 6.12	-1.571680505	1.160247e-01	2.161555e-01
82	16.32 - 6.12	-3.181752860	1.463867e-03	7.110210e-03
83	18.36 - 6.12	-3.877627123	1.054802e-04	8.965818e-04
84	2.2 - 6.12	0.454602593	6.493952e-01	8.331863e-01
85	2.4 - 6.12	3.371066682	7.487773e-04	4.243071e-03
86	24.48 - 6.12	-2.753286607	5.900022e-03	2.006007e-02
87	28.56 - 6.12	-2.711213156	6.703752e-03	2.170739e-02
88	32.64 - 6.12	-1.608943303	1.076287e-01	2.091073e-01
89	38.76 - 6.12	-1.561173546	1.184828e-01	2.177522e-01
90	4.4 - 6.12	-5.133375184	2.845918e-07	9.676121e-06
91	4.8 - 6.12	0.573767812	5.661249e-01	7.403172e-01
92	10.20 - 6.6	-2.004360554	4.503145e-02	1.093621e-01
93	12.24 - 6.6	-4.527604590	5.965608e-06	1.014153e-04
94	14.28 - 6.6	-2.102506138	3.550896e-02	8.942998e-02
95	16.32 - 6.6	-3.636767373	2.760811e-04	1.976159e-03
96	18.36 - 6.6	-4.320715505	1.555241e-05	2.115128e-04
97	2.2 - 6.6	0.000000000	1.000000e+00	1.000000e+00
98	2.4 - 6.6	0.863137189	3.880620e-01	5.555415e-01
99	24.48 - 6.6	-3.186202152	1.441538e-03	7.261082e-03
100	28.56 - 6.6	-3.145659160	1.657130e-03	7.771368e-03
101	32.64 - 6.6	-1.907567210	5.644718e-02	1.301155e-01
102	38.76 - 6.6	-1.860413863	6.282699e-02	1.424079e-01
103	4.4 - 6.6	-5.098595572	3.421828e-07	9.307371e-06
104	4.8 - 6.6	-1.136774260	2.556327e-01	3.862894e-01
105	6.12 - 6.6	-1.410367453	1.584312e-01	2.693331e-01
106	10.20 - 8.16	-1.180893343	2.376451e-01	3.672697e-01
107	12.24 - 8.16	-4.101958969	4.096669e-05	4.285746e-04
108	14.28 - 8.16	-1.621570255	1.048954e-01	2.067503e-01
109	16.32 - 8.16	-3.214984935	1.304514e-03	6.823610e-03
110	18.36 - 8.16	-3.901603585	9.555755e-05	8.663885e-04
111	2.2 - 8.16	0.404553407	6.858058e-01	8.716784e-01
112	2.4 - 8.16	2.835406268	4.576744e-03	1.728992e-02
113	24.48 - 8.16	-2.789364990	5.281151e-03	1.890096e-02
114	28.56 - 8.16	-2.747623425	6.002891e-03	1.991203e-02
115	32.64 - 8.16	-1.638548831	1.013073e-01	2.026145e-01
116	38.76 - 8.16	-1.590908996	1.116301e-01	2.138266e-01
117	4.4 - 8.16	-4.895987754	9.781316e-07	1.900370e-05
118	4.8 - 8.16	0.291633535	7.705668e-01	9.614412e-01
119	6.12 - 8.16	-0.191145644	8.484115e-01	9.946893e-01
120	6.6 - 8.16	1.201060579	2.297277e-01	3.591146e-01
121	10.20 - 8.8	-1.369238927	1.709246e-01	2.869845e-01
122	12.24 - 8.8	-3.699616246	2.159257e-04	1.727406e-03
123	14.28 - 8.8	-1.806420879	7.085262e-02	1.554187e-01
124	16.32 - 8.8	-3.124619899	1.780349e-03	7.810566e-03
125	18.36 - 8.8	-3.614969949	3.003825e-04	2.042601e-03
126	2.2 - 8.8	0.000000000	1.000000e+00	1.000000e+00
127	2.4 - 8.8	0.495031594	6.205778e-01	8.037960e-01
128	24.48 - 8.8	-2.821183648	4.784680e-03	1.758693e-02
129	28.56 - 8.8	-2.785285353	5.348064e-03	1.864966e-02
130	32.64 - 8.8	-1.819703141	6.880422e-02	1.533996e-01
131	38.76 - 8.8	-1.774721715	7.594385e-02	1.639423e-01
132	4.4 - 8.8	-3.034061289	2.412854e-03	1.025463e-02
133	4.8 - 8.8	-0.634062388	5.260401e-01	6.945772e-01
134	6.12 - 8.8	-0.824146401	4.098564e-01	5.806299e-01
135	6.6 - 8.8	0.000000000	1.000000e+00	1.000000e+00
136	8.16 - 8.8	-0.724628823	4.686798e-01	6.374045e-01

Figure 43: The result

7.5 Effects on The Bus Speed

We now perform the one-way ANOVA to investigate the effect of only the number of cores or the number of threads on the bus speed.

```
> onewayBS.Cores = aov(bus_speed ~ cores, data = newIntel)
> summary(onewayBS.Cores)
      Df Sum Sq Mean Sq F value Pr(>F)
cores    1  326.2   326.2   143.5 <2e-16 ***
Residuals 516 1173.3     2.3
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> onewayBS.Threads = aov(bus_speed ~ threads, data = newIntel)
> summary(onewayBS.Threads)
      Df Sum Sq Mean Sq F value Pr(>F)
threads    1  234.8   234.78   95.79 <2e-16 ***
Residuals 516 1264.7     2.45
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 44: The results of the two models.

The p-value for both the cores and threads variables is extremely low ($p < 0.0001$), so that different numbers of cores or threads used in CPUs does has influence on the processor's bus speed.

Next, we will combine both cores and threads:

```
> twowayBS = aov(bus_speed ~ cores + threads, data = newIntel)
> summary(twowayBS)
      Df Sum Sq Mean Sq F value    Pr(>F)
cores    1  326.2   326.2   156.07 < 2e-16 ***
threads    1   96.9    96.9   46.34 2.77e-11 ***
Residuals 515 1076.4     2.1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 45: Two-way ANOVA.

Constructing a two-way ANOVA indicates that both cores and threads are statistically significant ($p - value < 0.05$).

Then, we will use the AIC test to select the best-fit model:

Model selection based on AICc:

	K	AICc	Delta_AICc	AICcwt	Cum.Wt	LL
twowayBS	4	1856.96	0.00	1	1	-924.44
onewayBS.Cores	3	1899.56	42.60	0	1	-946.76
onewayBS.Threads	3	1938.43	81.47	0	1	-966.19

Figure 46: The result of AIC test.

The outcome shows that the 'twoway' model is the best-fit. It has the lowest AIC score, and 100% of the AIC Weight.

After having the best-fit model, we are going to make sure that ANOVA's assumptions is still satisfied.

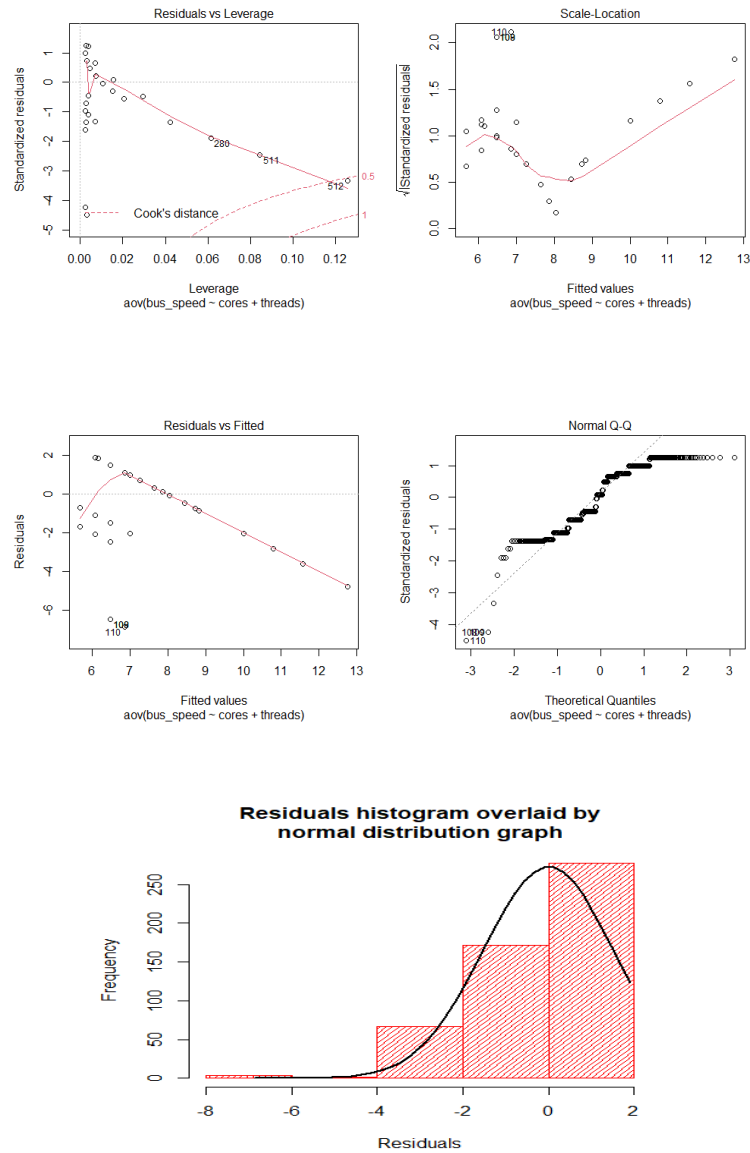


Figure 47: The result

We now perform the Shapiro-Wilk test for normality and Levene's Test for homoscedasticity.

```
> shapiro.test(twoway[['residuals']])

Shapiro-Wilk normality test

data: twoway[["residuals"]]
W = 0.8972, p-value < 2.2e-16
> leveneTest(bus_speed ~ interaction(cores, threads), data=newIntel)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value    Pr(>F)
group 16   8.025 < 2.2e-16 ***
      504
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 48: The result

The p -values obtained from Shapiro-Wilk test and Levene' Test are both significant ($p < 0.05$). Combining with the plots, We have enough proof to conclude that the data is not normally distributed and does not have equal variance.

So, we try Kruskal-Wallis test on this case.

```
> kruskal.test(bus_speed ~ interaction(cores, threads), data=newIntel)

Kruskal-Wallis rank sum test

data: bus_speed by interaction(cores, threads)
Kruskal-Wallis chi-squared = 265.5, df = 16, p-value < 2.2e-16

> epsilonSquared(x=newIntel$bus_speed, g = interaction(newIntel$cores,newIntel$threads))
epsilon.squared
0.511
```

Figure 49: The result

The p -value < 0.05 indicates that there are significant differences in the bus speed among the cores and threads. $\epsilon^2 = 0.511$ advise a strong effect of the numbers of cores and threads on the base frequency.

The Dunn's Test as post-hoc test for significant Kruskal-Wallis test then will be included to study which parts are the source of the differences.



Dunn (1964) Kruskal-Wallis multiple comparison
p-values adjusted with the Benjamini-Hochberg method.

	Comparison	Z	P.unadj	P.adj
1	10.20 - 12.24	0.00000000	1.000000e+00	1.000000e+00
2	10.20 - 14.28	0.00000000	1.000000e+00	1.000000e+00
3	12.24 - 14.28	0.00000000	1.000000e+00	1.000000e+00
4	10.20 - 16.32	0.00000000	1.000000e+00	1.000000e+00
5	12.24 - 16.32	0.00000000	1.000000e+00	1.000000e+00
6	14.28 - 16.32	0.00000000	1.000000e+00	1.000000e+00
7	10.20 - 18.36	0.00000000	1.000000e+00	1.000000e+00
8	12.24 - 18.36	0.00000000	1.000000e+00	1.000000e+00
9	14.28 - 18.36	0.00000000	1.000000e+00	1.000000e+00
10	16.32 - 18.36	0.00000000	1.000000e+00	1.000000e+00
11	10.20 - 2.2	0.00000000	1.000000e+00	1.000000e+00
12	12.24 - 2.2	0.00000000	1.000000e+00	1.000000e+00
13	14.28 - 2.2	0.00000000	1.000000e+00	1.000000e+00
14	16.32 - 2.2	0.00000000	1.000000e+00	1.000000e+00
15	18.36 - 2.2	0.00000000	1.000000e+00	1.000000e+00
16	10.20 - 2.4	8.11879572	4.708323e-16	1.280664e-14
17	12.24 - 2.4	4.74221402	2.113951e-06	2.395811e-05
18	14.28 - 2.4	3.91052986	9.209387e-05	7.827979e-04
19	16.32 - 2.4	3.91052986	9.209387e-05	8.349845e-04
20	18.36 - 2.4	4.35040737	1.358848e-05	1.421565e-04
21	2.2 - 2.4	2.79323561	5.218367e-03	3.548489e-02
22	10.20 - 24.48	0.00000000	1.000000e+00	1.000000e+00
23	12.24 - 24.48	0.00000000	1.000000e+00	1.000000e+00
24	14.28 - 24.48	0.00000000	1.000000e+00	1.000000e+00
25	16.32 - 24.48	0.00000000	1.000000e+00	1.000000e+00
26	18.36 - 24.48	0.00000000	1.000000e+00	1.000000e+00
27	2.2 - 24.48	0.00000000	1.000000e+00	1.000000e+00
28	2.4 - 24.48	-3.40367936	6.648475e-04	5.023292e-03
29	10.20 - 28.56	0.00000000	1.000000e+00	1.000000e+00
30	12.24 - 28.56	0.00000000	1.000000e+00	1.000000e+00
31	14.28 - 28.56	0.00000000	1.000000e+00	1.000000e+00
32	16.32 - 28.56	0.00000000	1.000000e+00	1.000000e+00
33	18.36 - 28.56	0.00000000	1.000000e+00	1.000000e+00
34	2.2 - 28.56	0.00000000	1.000000e+00	1.000000e+00
35	2.4 - 28.56	-3.40367936	6.648475e-04	5.318780e-03
36	24.48 - 28.56	0.00000000	1.000000e+00	1.000000e+00
37	10.20 - 32.64	0.00000000	1.000000e+00	1.000000e+00
38	12.24 - 32.64	0.00000000	1.000000e+00	1.000000e+00
39	14.28 - 32.64	0.00000000	1.000000e+00	1.000000e+00
40	16.32 - 32.64	0.00000000	1.000000e+00	1.000000e+00
41	18.36 - 32.64	0.00000000	1.000000e+00	1.000000e+00
42	2.2 - 32.64	0.00000000	1.000000e+00	1.000000e+00
43	2.4 - 32.64	-1.98527075	4.711436e-02	2.288412e-01
44	24.48 - 32.64	0.00000000	1.000000e+00	1.000000e+00
45	28.56 - 32.64	0.00000000	1.000000e+00	1.000000e+00
46	10.20 - 38.76	0.00000000	1.000000e+00	1.000000e+00
47	12.24 - 38.76	0.00000000	1.000000e+00	1.000000e+00
48	14.28 - 38.76	0.00000000	1.000000e+00	1.000000e+00
49	16.32 - 38.76	0.00000000	1.000000e+00	1.000000e+00
50	18.36 - 38.76	0.00000000	1.000000e+00	1.000000e+00
51	2.2 - 38.76	0.00000000	1.000000e+00	1.000000e+00
52	2.4 - 38.76	-1.98527075	4.711436e-02	2.373168e-01
53	24.48 - 38.76	0.00000000	1.000000e+00	1.000000e+00
54	28.56 - 38.76	0.00000000	1.000000e+00	1.000000e+00
55	32.64 - 38.76	0.00000000	1.000000e+00	1.000000e+00
56	10.20 - 4.4	1.97018335	4.881736e-02	2.289366e-01
57	12.24 - 4.4	1.18651267	2.354199e-01	9.147745e-01
58	14.28 - 4.4	0.98357498	3.253245e-01	1.000000e+00
59	16.32 - 4.4	0.98357498	3.253245e-01	1.000000e+00
60	18.36 - 4.4	1.09130763	2.751375e-01	9.847028e-01
61	2.2 - 4.4	0.70644585	4.799109e-01	1.000000e+00
62	2.4 - 4.4	-9.08725461	1.015713e-19	4.604567e-18
63	24.48 - 4.4	0.85842982	3.906552e-01	1.000000e+00
64	28.56 - 4.4	0.85842982	3.906552e-01	1.000000e+00
65	32.64 - 4.4	0.50354503	6.145811e-01	1.000000e+00
66	38.76 - 4.4	0.50354503	6.145811e-01	1.000000e+00
67	10.20 - 4.8	4.82813274	1.378192e-06	1.703947e-05
68	12.24 - 4.8	2.75705808	5.832399e-03	3.777173e-02
69	14.28 - 4.8	2.26524229	2.349782e-02	1.278281e-01
70	16.32 - 4.8	2.26524229	2.349782e-02	1.331543e-01
71	18.36 - 4.8	2.52469104	1.158000e-02	7.158548e-02
72	2.2 - 4.8	1.61193825	1.069754e-01	4.546454e-01
73	2.4 - 4.8	-6.53260578	6.463511e-11	1.255768e-09
74	24.48 - 4.8	1.96795567	4.907314e-02	2.152886e-01
75	28.56 - 4.8	1.96795567	4.907314e-02	2.224649e-01
76	32.64 - 4.8	1.14345987	2.528477e-01	9.293862e-01
77	38.76 - 4.8	1.14345987	2.528477e-01	9.552024e-01
78	4.4 - 4.8	4.23092683	2.327304e-05	2.260809e-04
79	10.20 - 6.12	0.98137164	3.264095e-01	1.000000e+00
80	12.24 - 6.12	0.58212121	5.604850e-01	1.000000e+00
81	14.28 - 6.12	0.48127883	6.303183e-01	1.000000e+00
82	16.32 - 6.12	0.48127883	6.303183e-01	1.000000e+00
83	18.36 - 6.12	0.53471248	5.928487e-01	1.000000e+00
84	2.2 - 6.12	0.34470687	7.303147e-01	1.000000e+00
85	2.4 - 6.12	-11.34648515	7.720419e-30	5.249885e-28
86	24.48 - 6.12	0.41946298	6.748778e-01	1.000000e+00
87	28.56 - 6.12	0.41946298	6.748778e-01	1.000000e+00
88	32.64 - 6.12	0.24534291	8.061909e-01	1.000000e+00
89	38.76 - 6.12	0.24534291	8.061909e-01	1.000000e+00
90	4.4 - 6.12	-1.51195219	1.305460e-01	5.380079e-01
91	4.8 - 6.12	-6.39965870	1.557246e-10	2.647319e-09
92	10.20 - 6.6	0.00000000	1.000000e+00	1.000000e+00
93	12.24 - 6.6	0.00000000	1.000000e+00	1.000000e+00
94	14.28 - 6.6	0.00000000	1.000000e+00	1.000000e+00
95	16.32 - 6.6	0.00000000	1.000000e+00	1.000000e+00
96	18.36 - 6.6	0.00000000	1.000000e+00	1.000000e+00
97	2.2 - 6.6	0.00000000	1.000000e+00	1.000000e+00
98	2.4 - 6.6	-8.88756786	6.246107e-19	2.123676e-17
99	24.48 - 6.6	0.00000000	1.000000e+00	1.000000e+00
100	28.56 - 6.6	0.00000000	1.000000e+00	1.000000e+00
101	32.64 - 6.6	0.00000000	1.000000e+00	1.000000e+00
102	38.76 - 6.6	0.00000000	1.000000e+00	1.000000e+00
103	4.4 - 6.6	-2.13774025	3.253783e-02	1.701979e-01
104	4.8 - 6.6	-5.32251605	1.023417e-07	1.546498e-06
105	6.12 - 6.6	-1.06942495	2.848782e-01	9.685860e-01
106	10.20 - 8.16	0.19837054	8.427552e-01	1.000000e+00
107	12.24 - 8.16	0.12085740	9.038040e-01	1.000000e+00
108	14.28 - 8.16	0.10039595	9.200300e-01	1.000000e+00
109	16.32 - 8.16	0.10039595	9.200300e-01	1.000000e+00
110	18.36 - 8.16	0.11127424	9.113989e-01	1.000000e+00
111	2.2 - 8.16	0.07226987	9.423871e-01	1.000000e+00
112	2.4 - 8.16	-11.35718763	6.830837e-30	9.289938e-28
113	24.48 - 8.16	0.08771830	9.301006e-01	1.000000e+00
114	28.56 - 8.16	0.08771830	9.301006e-01	1.000000e+00
115	32.64 - 8.16	0.05157352	9.588685e-01	1.000000e+00
116	38.76 - 8.16	0.05157352	9.588685e-01	1.000000e+00
117	4.4 - 8.16	-2.42623086	1.525656e-02	9.021271e-02
118	4.8 - 8.16	-6.88208201	5.898404e-12	1.336972e-10
119	6.12 - 8.16	-1.08619945	2.773908e-01	9.673114e-01
120	6.6 - 8.16	0.21455879	8.301113e-01	1.000000e+00
121	10.20 - 8.8	0.00000000	1.000000e+00	1.000000e+00
122	12.24 - 8.8	0.00000000	1.000000e+00	1.000000e+00
123	14.28 - 8.8	0.00000000	1.000000e+00	1.000000e+00
124	16.32 - 8.8	0.00000000	1.000000e+00	1.000000e+00
125	18.36 - 8.8	0.00000000	1.000000e+00	1.000000e+00
126	2.2 - 8.8	0.00000000	1.000000e+00	1.000000e+00
127	2.4 - 8.8	-5.09725098	3.446213e-07	4.686850e-06
128	24.48 - 8.8	0.00000000	1.000000e+00	1.000000e+00
129	28.56 - 8.8	0.00000000	1.000000e+00	1.000000e+00
130	32.64 - 8.8	0.00000000	1.000000e+00	1.000000e+00
131	38.76 - 8.8	0.00000000	1.000000e+00	1.000000e+00
132	4.4 - 8.8	-1.27212187	2.033298e-01	8.133193e-01
133	4.8 - 8.8	-2.96875762	2.990063e-03	2.140256e-02
134	6.12 - 8.8	-0.62491709	5.320255e-01	1.000000e+00
135	6.6 - 8.8	0.00000000	1.000000e+00	1.000000e+00
136	8.16 - 8.8	-0.12944850	8.970028e-01	1.000000e+00

Figure 50: The result

From the table we can see that these following combinations are statistically significant difference from one another ($p - value < 0.05$).

10.20 - 2.4

12.24 - 2.4
14.28 - 2.4
16.32 - 2.4
18.36 - 2.4
2.2 - 2.4
2.4 - 24.48
2.4 - 28.56
2.4 - 4.4
10.20 - 4.8
12.24 - 4.8
18.36 - 4.8
2.4 - 4.8
4.4 - 4.8
2.4 - 6.12
4.8 - 6.12
2.4 - 6.6
4.8 - 6.6
2.4 - 8.16
4.4 - 8.16
4.8 - 8.16
2.4 - 8.8
4.8 - 8.8

7.6 Summary

Overall, we can observe that in fact, there is a real impact of the number of cores and threads on other variables. However, when trying to examine their effect with the ANOVA models, one or both core assumptions for ANOVA to work well are failed, which are normality and homogeneity.

Consequently, we have to try another approach with the Kruskal-Wallis Test to study whether there are statistically significant differences among the cores and threads or not. If indeed there are, the Dunn's test, in addition served as the post-hoc test for the Kruskal-Wallis Test, will show us which parts of cores and threads cause the difference. Besides, the epsilon-squared also contribute to the analysis by giving the level of effect.

Throughout the researching process with ANOVA, all of the models' variation around the mean for each group being compared is not equal among all groups. The Dunn's test has proved its efficiency on providing information about which parts has caused the statistically significant difference.

8 Fitting Linear Regression Models

8.1 Overview

In this section, we are intending to build a linear regression model for researching the dependency of each variables on the others.

During the whole process, we will need to keep in mind the following criteria:

Verify the fitness of linear regression's assumptions:

Linearity: Predictors in the model have a straight-line relationship with the dependent variable.

Normality: The residuals of the model should follow the normal distribution.

Multi-Collinearity: Predictors are not correlated to each other.

Homogeneity: Standard deviations are equal for all observations.

While building a regression model, we will try the backward elimination strategy, which starts with all predictors in the model, iteratively removes the least contributed predictors, and stops when you have a model where all predictors are statistically significant.

8.2 Diagnostic for Regression with Correlation Matrix

The following code segment will help us generate the correlation matrix to take a look inside the relationship between independent variables as well as between independent and dependent variables:

```
> test_cor_matrix = newIntel
> round(cor(test_cor_matrix),4)
```

	cores	threads	bus_speed	base_frequency	turbo_frequency	cache_size	max_memory_size	max_temp
cores	1.0000	0.9734	0.4408	0.2101	0.4332	0.9539	0.6641	-0.2466
threads	0.9734	1.0000	0.3710	0.1772	0.4345	0.9483	0.6569	-0.2108
bus_speed	0.4408	0.3710	1.0000	0.4907	0.5240	0.4735	0.2212	0.0078
base_frequency	0.2101	0.1772	0.4907	1.0000	0.4899	0.2669	0.1460	-0.2730
turbo_frequency	0.4332	0.4345	0.5240	0.4899	1.0000	0.5718	0.1206	0.0795
cache_size	0.9539	0.9483	0.4735	0.2669	0.5718	1.0000	0.6096	-0.2094
max_memory_size	0.6641	0.6569	0.2212	0.1460	0.1206	0.6096	1.0000	-0.3478
max_temp	-0.2466	-0.2108	0.0078	-0.2730	0.0795	-0.2094	-0.3478	1.0000

Figure 51: The correlation matrix

A quick glance at the matrix tell us that the correlation between cores and threads is 0.9734, which indicates that they are strongly positively correlated. Hence, it is essential to not fit cores and threads in the same model.

For the relations like bus speed and maximum temperature, whose correlated value is close to 0 (0.0078 in this case), simply because they are not correlated at all.

The next noticeable point from the matrix is that cache size is the dependent variable that is most related to our two statistically significant variables, cores and threads (0.9539 and 0.9483, respectively).

On the other hand, maximum temperature's correlated values with others seems to be negative or close to zero so there is no reason to examine the dependency of this factor.

8.3 Fitting Cache Size in Linear Regression Models

8.3.1 With Cores and The Others

Firstly, we will build the linear regression model without threads in the predictors list and run the summary on our model.

```
> library(lmtest)
> library(car)
> library(lindia)
> Y = newIntel$cache_size
> model = lm(Y~cores+bus_speed+base_frequency+
  turbo_frequency+max_memory_size+max_temp,data=newIntel)
> summary(model)
```

Call:
lm(formula = Y ~ cores + bus_speed + base_frequency + turbo_frequency +
max_memory_size + max_temp, data = newIntel)

Residuals:

	Min	1Q	Median	3Q	Max
	-3641.9	-974.2	-61.2	723.3	9745.8

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-4.394e+03	7.503e+02	-5.856	8.45e-09 ***
cores	1.371e+03	2.753e+01	49.799	< 2e-16 ***
bus_speed	-3.606e+01	5.188e+01	-0.695	0.487
base_frequency	-1.764e-01	1.245e-01	-1.416	0.157
turbo_frequency	2.074e+00	1.423e-01	14.581	< 2e-16 ***
max_memory_size	2.196e-07	2.105e-07	1.043	0.297
max_temp	-8.781e+00	6.997e+00	-1.255	0.210

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1584 on 514 degrees of freedom
Multiple R-squared: 0.9415, Adjusted R-squared: 0.9408
F-statistic: 1378 on 6 and 514 DF, p-value: < 2.2e-16

Figure 52: The model's summary

From the table, The adjusted R-squared is really high (0.9408), which indicates that the model has fit well. But, we need to drop bus speed, base frequency, max memory size and max temperature out of the model since their p -value > 0.05 and run the summary again.

Before doing that, we have to make sure our model still meet the assumptions of linear regression models.

The following code segment will successively verify each assumption:

```
> plot(model)
> h<-hist(model[['residuals']],breaks=5,density=40,
  col="red", xlab="Residuals"
  ,main = "Residuals histogram overlaid by normal distribution graph")
> xfit<-seq(min(model[['residuals']]),max(model[['residuals']])
  ,length=40)
> yfit<-dnorm(xfit,mean=mean(model[['residuals']])
  ,sd=sd(model[['residuals']]))
> yfit<-yfit*diff(h$mids[1:2])*length(model[['residuals']])
> lines(xfit,yfit,col="black",lwd = 2)
> vif(model) # Check for Multi-Collinearity
> shapiro.test(model[['residuals']])# Check for Normality
> bptest(model)# Check for homogeneity
```

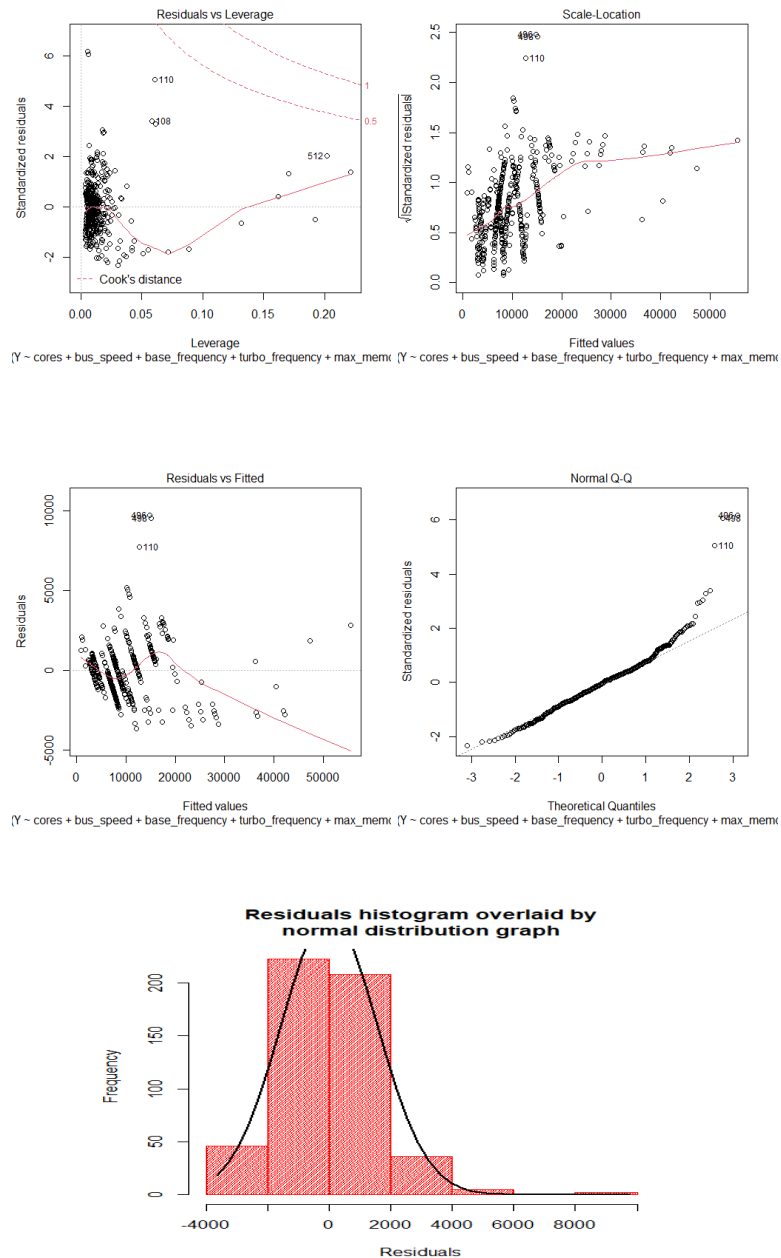


Figure 53: The plots for our model

The Residuals vs Leverage graph show that our model don not have any influential points. But from the rest, we can see that item with row index 110, 496, 498 are the out-liners. In case, our model passes all assumption, we will need to delete these row before running a new summary.

From the Normal-QQ graph, the predictors tend to have a linear relation with the dependent variable. However, the red line in Residual vs Fitted plot does not close to 0, in fact, it has a decreasing trend, which means that linearity is violated. The residual histogram seems to fit the normal distribution. Since the spread of residuals in Residuals vs Fitted plot is not symmetric about the x-axis and the red line in Scale-Location graph is not approximately horizontal, the model does not meet homoscedasticity assumption.

There are no VIF values that exceed 5 or 10, which indicate all variables in this model are not correlated to each other.

cores	bus_speed	base_frequency	turbo_frequency	max_memory_size	max_temp
2.585193	1.747621	1.799559	1.926943	2.005763	1.396495

Figure 54: The VIF result

```

shapiro-wilk normality test

data: model[["residuals"]]
W = 0.92815, p-value = 4.253e-15

studentized Breusch-Pagan test

data: model
BP = 49.922, df = 6, p-value = 4.873e-09

```

Figure 55: The result of Shapiro-Wilk and Breusch-Pagan tests

Unluckily, the p -value in both tests is less than 0.05 so the model disobeys the normality and homogeneity assumptions. Therefore, we will stop this path of fitting the cache size with cores and the others.

8.3.2 With Threads and The Others

Similarly, we now build the linear regression model without cores.

```

Call:
lm(formula = Y ~ threads + bus_speed + base_frequency + turbo_frequency +
    max_memory_size + max_temp, data = newIntel)

Residuals:
    Min       1Q   Median       3Q      Max
-5058.1  -909.9   -67.1    817.3   9324.7

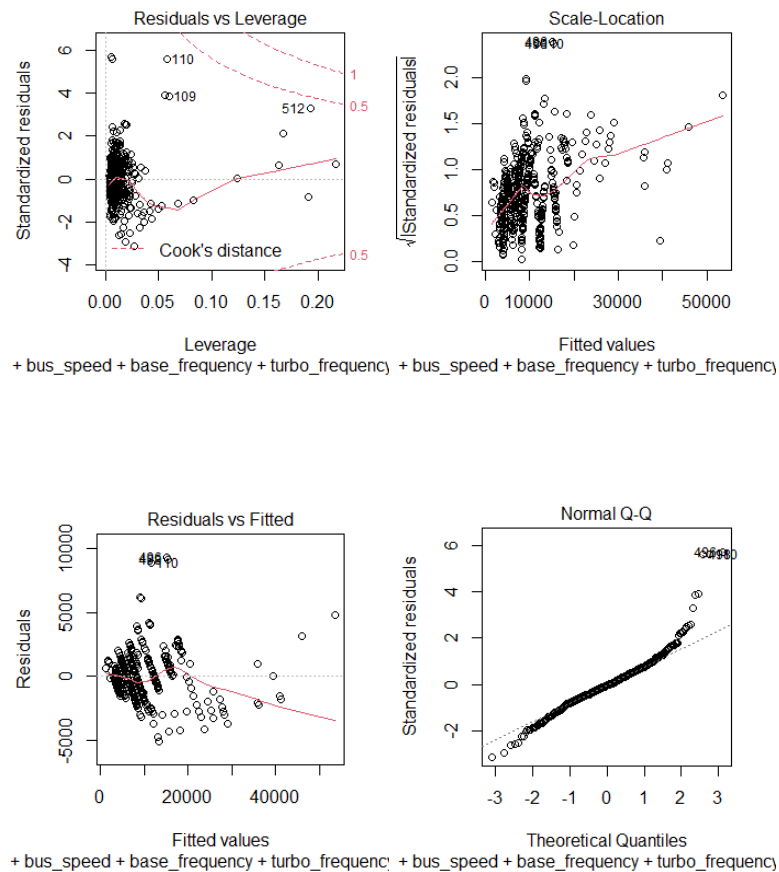
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.234e+03  7.792e+02  -2.868   0.0043 **
threads       6.338e+02  1.330e+01  47.650 < 2e-16 ***
bus_speed     2.981e+02  5.228e+01   5.701 2.01e-08 ***
base_frequency -2.462e-01  1.289e-01  -1.911   0.0566 .
turbo_frequency  1.823e+00  1.500e-01  12.158 < 2e-16 ***
max_memory_size  2.240e-07  2.189e-07   1.023   0.3066
max_temp      -3.033e+01  7.186e+00  -4.220 2.89e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1643 on 514 degrees of freedom
Multiple R-squared:  0.9371, Adjusted R-squared:  0.9363
F-statistic: 1275 on 6 and 514 DF, p-value: < 2.2e-16

```

Figure 56: The model's summary

The R-squared value is high (0.9363) that means the model has fit well. We have to get rid of base frequency and max memory size ($p\text{-value} > 0.05$) and then run the summary again. But we have to check if our model is satisfied with the assumptions.



Is histogram overlaid by normal distri

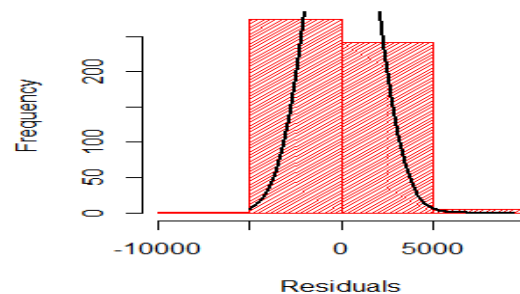


Figure 57: The plots for our model

The Residuals vs Leverage graph show that our model don not have any influential points. But from the rest, we can see that item with row index 110, 496, 498 again are the outliners. In case, our model passes all assumption, we will need to delete these row before running a new summary

Similary with the cores model, this model also does not meet almost all of the assumptions.

There are no VIF values that exceed 5 or 10 so a problematic amount of collinearity does not exist.

threads	bus_speed	base_frequency	turbo_frequency	max_memory_size	max_temp
2.410836	1.650642	1.791783	1.990997	2.017122	1.369895

Figure 58: The VIF result

```
studentized Breusch-Pagan test

data: model
BP = 50.446, df = 6, p-value = 3.827e-09

Shapiro-Wilk normality test

data: model[["residuals"]]
W = 0.9285, p-value = 4.661e-15
```

Figure 59: The result of Shapiro-Wilk and Breusch-Pagan tests

The $p - value$ in both tests is less than 0.05 so the model breach the normality and homogeneity assumptions. Therefore, the cache size can not fit in a linear regression model.

8.4 Fitting Maximum Memory Size in Linear Regression Models

8.4.1 With Cores and The Others

Firstly, we will build the linear regression model with cores in the predictors list and run the summary on our model.

```
Call:
lm(formula = Y ~ cores + bus_speed + base_frequency + turbo_frequency +
    cache_size + max_temp, data = newIntel)

Residuals:
    Min       1Q   Median       3Q      Max
-1.550e+09 -9.273e+07  4.821e+06  8.204e+07  3.382e+09

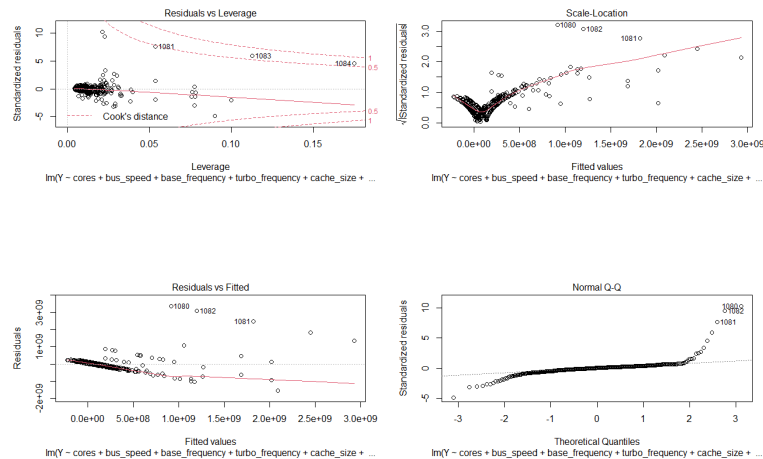
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  752939446  158775067   4.742  2.74e-06 ***
cores        68465122   13576769   5.043  6.37e-07 ***
bus_speed   -4690665    10862328  -0.432  0.666047
base_frequency  42376      26051   1.627  0.104433
turbo_frequency -156210     34731  -4.498  8.50e-06 ***
cache_size     9621       9223   1.043  0.297359
max_temp    -5441317    1447103  -3.760  0.000189 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 331600000 on 514 degrees of freedom
Multiple R-squared:  0.5025,    Adjusted R-squared:  0.4967
F-statistic: 86.52 on 6 and 514 DF,  p-value: < 2.2e-16
```

Figure 60: The model's summary

From the table, The adjusted R-squared not high (0.4967), which indicates that the model may not appropriate in this circumstance.

Despite that, we still try to take a further inspect on whether the model meet linear regression assumptions or not.



Residuals histogram overlaid by normal distribution graph

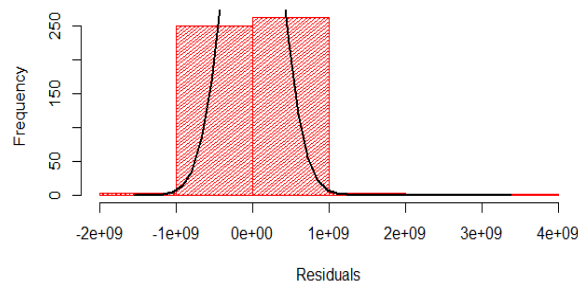


Figure 61: The plots for our model

From all the plots above, our model has some out-liners. The normal distribution seems to be applied in this case. Nevertheless, homogeneity and linearity tends to be disrupted.

The cores and cache size with $vif \approx 14.35$ and 17.05 , respectively, are the two that have a strong relation with each other as we have diagnosed before.

```
> vif(model)
      cores      bus_speed base_frequency turbo_frequency      cache_size      max_temp
14.348143    1.748630    1.797329    2.620869    17.045943    1.363274
```

Figure 62: The VIF result

```
> shapiro.test(model[["residuals"]])

Shapiro-wilk normality test

data: model[["residuals"]]
W = 0.5875, p-value < 2.2e-16

> # Check for homogeneity
> bptest(model)

studentized Breusch-Pagan test

data: model
BP = 107.41, df = 6, p-value < 2.2e-16
```

Figure 63: The result of Shapiro-Wilk and Breusch-Pagan tests

The Shapiro-Wilk and Breusch-Pagan Tests has once again proved our conclusion above. Hence, the model is aborted.

8.4.2 With Threads And The Others

The summary for our linear regression model are shown below:

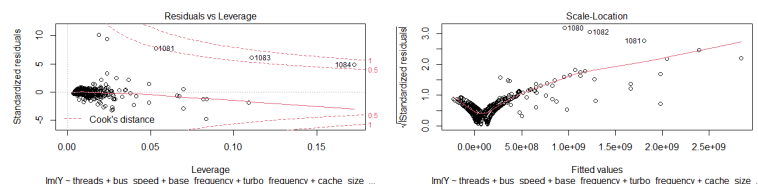
```
Call:
lm(formula = Y ~ threads + bus_speed + base_frequency + turbo_frequency +
    cache_size + max_temp, data = newIntel)

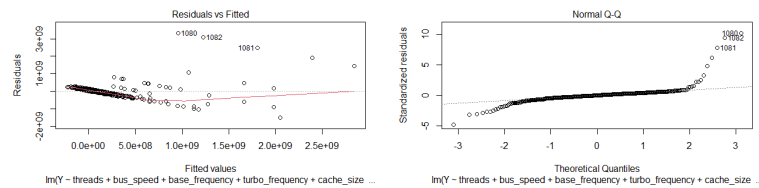
Residuals:
    Min       1Q   Median       3Q      Max
-1.513e+09 -9.752e+07  1.729e+06  9.302e+07  3.335e+09

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  856720196  153517011  5.581 3.88e-08 ***
threads      32352211   6066345   5.333 1.45e-07 ***
bus_speed    11723141   10839599   1.082  0.280
base_frequency  39779    25972    1.532  0.126
turbo_frequency -169742    33426  -5.078 5.34e-07 ***
cache_size     9078     8870    1.023  0.307
max_temp     -6456788   1443605  -4.473 9.52e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 330700000 on 514 degrees of freedom
Multiple R-squared:  0.5053,    Adjusted R-squared:  0.4995
F-statistic: 87.49 on 6 and 514 DF, p-value: < 2.2e-16
```

Figure 64: The model's summary





Residuals histogram overlaid by normal distribution graph

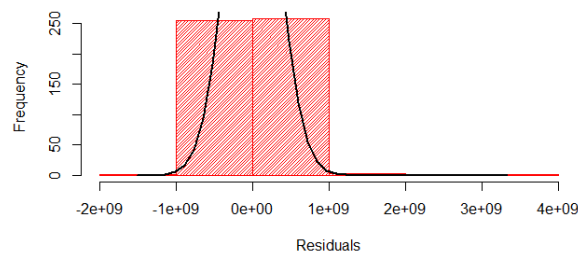


Figure 65: The plots for our model

```
> vif(model)
threads      bus_speed base_frequency turbo_frequency  cache_size  max_temp
12.375411    1.751041    1.796309      2.441118      15.854868    1.364265
```

Figure 66: The VIF result

```
> shapiro.test(model[["residuals"]])
Shapiro-Wilk normality test
data:  model[["residuals"]]
W = 0.59595, p-value < 2.2e-16

> # Check for homogeneity
> bptest(model)
studentized Breusch-Pagan test
data:  model
BP = 112.6, df = 6, p-value < 2.2e-16
```

Figure 67: The result of Shapiro-Wilk and Breusch-Pagan tests

The result is similar with previous cores model where a low R-squared value give us enough evidence to finalize that the maximum memory size is not able to match in a linear regression model.

8.5 Fitting Bus Speed In A Linear Regression Model

8.5.1 With Cores And The Others

The linear regression model with cores will be implemented as following:


```
Call:
lm(formula = YBS ~ cores + cache_size + base_frequency + turbo_frequency +
    max_memory_size + max_temp, data = newIntel)

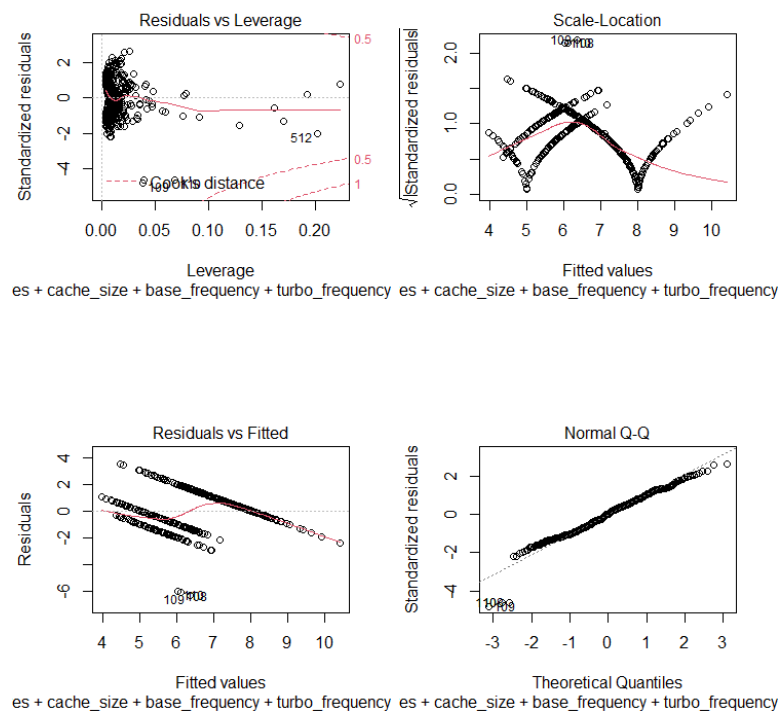
Residuals:
    Min       1Q   Median       3Q      Max
-6.3546 -0.9678  0.0345  0.9338  3.5400

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.156e+00  6.566e-01  -1.761 0.078839 .
cores        1.845e-01  5.588e-02   3.301 0.001029 **
cache_size   -2.604e-05  3.747e-05  -0.695 0.487300
base_frequency 9.008e-04  9.831e-05   9.163 < 2e-16 ***
turbo_frequency 5.164e-04  1.419e-04   3.638 0.000302 ***
max_memory_size -7.732e-11 1.790e-10  -0.432 0.666047
max_temp      2.613e-02  5.843e-03   4.472 9.53e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.346 on 514 degrees of freedom
Multiple R-squared:  0.4283,    Adjusted R-squared:  0.4217
F-statistic: 64.19 on 6 and 514 DF,  p-value: < 2.2e-16
```

Figure 68: The model's summary

The adjusted R-squared is not really high (0.4842), which indicates that the model may not proper. However, as we did before, we will try to investigate the linear regression's assumption.



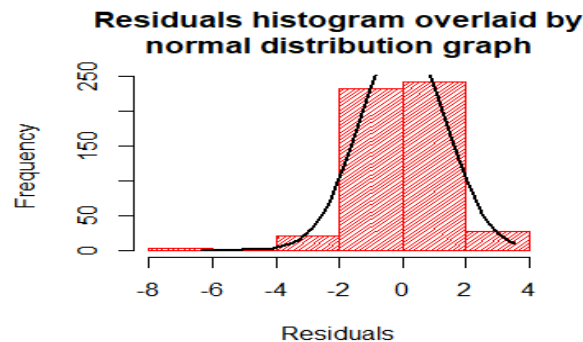


Figure 69: The plots for our model

Normality seems to be fitted well. But, other plots show really strange behaviour which means that other assumptions have been failed.

The VIF values of cores and cache size exceed 10, which indicates these variables are correlated to each other.

```
> vif(modelBS) # not use when only 1 attribute
      cores      cache_size base_frequency turbo_frequency max_memory_size      max_temp
14.745326    17.065989      1.552911      2.655634      2.009281      1.348305
```

Figure 70: The VIF result

```
Shapiro-wilk normality test
data:  modelBS[["residuals"]]
W = 0.97496, p-value = 8.859e-08

studentized Breusch-Pagan test
data:  modelBS
BP = 75.334, df = 6, p-value = 3.277e-14
```

Figure 71: The result of Shapiro-Wilk and Breusch-Pagan tests

The result from the above tests strengthen our conclusion that the linear regression model is not suitable in this case.

8.5.2 With Threads and The Others

The summary of this model is included below. But, we strongly believe that the result will be the same and linear regression still not suitable for our observation.

```
Call:
lm(formula = YBS ~ threads + cache_size + base_frequency + turbo_frequency +
    max_memory_size + max_temp, data = newIntel)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-3.1852	-0.8511	0.0377	0.8809	3.0994

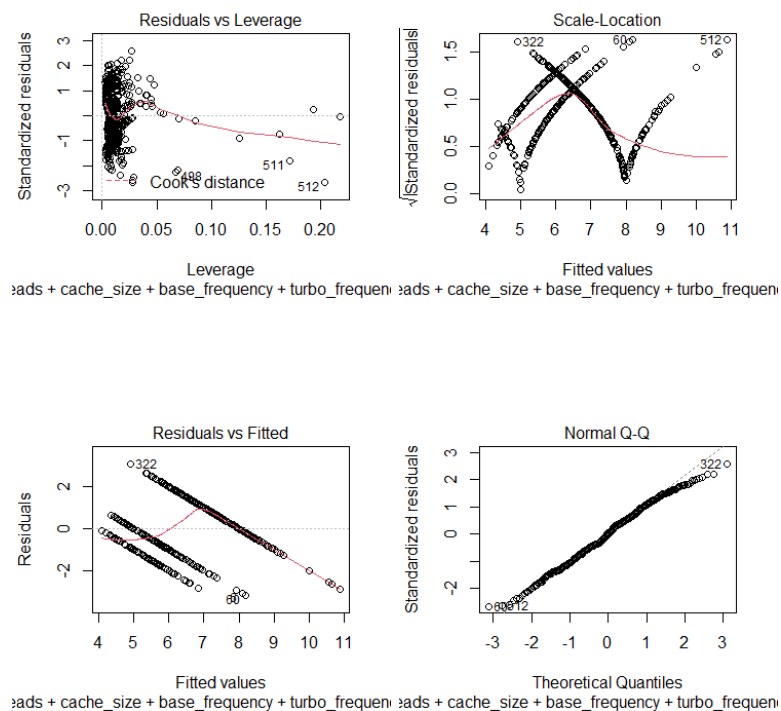
Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	9.271e-01	5.918e-01	1.567	0.117848
threads	-1.364e-01	2.296e-02	-5.941	5.26e-09 ***
cache_size	2.877e-04	3.256e-05	8.836	< 2e-16 ***
base_frequency	9.175e-04	8.877e-05	10.336	< 2e-16 ***
turbo_frequency	-3.818e-05	1.274e-04	-0.300	0.764468
max_memory_size	-4.801e-11	1.628e-10	-0.295	0.768123
max_temp	1.990e-02	5.320e-03	3.740	0.000205 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.21 on 511 degrees of freedom
Multiple R-squared: 0.5007, Adjusted R-squared: 0.4948
F-statistic: 85.41 on 6 and 511 DF, p-value: < 2.2e-16

Figure 72: The model's summary



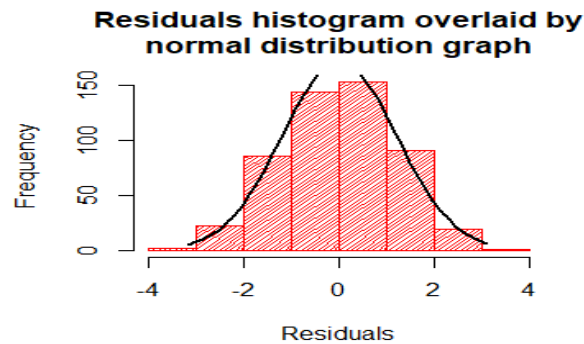


Figure 73: The plots for our model

```
> vif(modelBS) # not use when only 1 attribute
      threads      cache_size base_frequency turbo_frequency max_memory_size      max_temp
13.212129    15.816527      1.560566      2.639133      2.053689      1.337540
```

Figure 74: The VIF result

```
Shapiro-wilk normality test
data:  modelBS[["residuals"]]
W = 0.9926, p-value = 0.0115

studentized Breusch-Pagan test
data:  modelBS
BP = 51.35, df = 6, p-value = 2.519e-09
```

Figure 75: The result of Shapiro-Wilk and Breusch-Pagan tests

Same with our prediction, the results has proved that the bus speed is not capable of fitting a linear regression model.

8.6 Fitting Base Frequency in Linear Regression Models

8.6.1 With Cores and The Others

Let's build the model without threads and run the summary.

```
Call:
lm(formula = YBF ~ cores + bus_speed + cache_size + turbo_frequency +
    max_memory_size + max_temp, data = newIntel)

Residuals:
    Min       1Q   Median       3Q      Max
-1580.20  -373.36   63.64   444.00  1486.68

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.045e+03  2.587e+02   7.906 1.64e-14 ***
cores        -2.081e+01  2.347e+01  -0.887   0.376
bus_speed     1.559e+02  1.701e+01   9.163 < 2e-16 ***
cache_size    -2.204e-02  1.556e-02  -1.416   0.157
turbo_frequency 5.237e-01  5.515e-02   9.494 < 2e-16 ***
max_memory_size 1.208e-07  7.430e-08   1.627   0.104
max_temp      -2.266e+01  2.267e+00  -9.998 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 560 on 514 degrees of freedom
Multiple R-squared:  0.4465,    Adjusted R-squared:  0.44
F-statistic: 69.1 on 6 and 514 DF,  p-value: < 2.2e-16
```

Figure 76: The model's summary

Since the adjusted R-squared is not high (0.44), once again, the linear regression is not an applicable approach.

```
Shapiro-Wilk normality test

data:  modelBF[["residuals"]]
W = 0.98444, p-value = 2.289e-05

studentized Breusch-Pagan test

data:  modelBF
BP = 91.038, df = 6, p-value < 2.2e-16
```

Figure 77: The results of Shapiro-Wilk and Breusch-Pagan tests

It is clear that the model is also not satisfied with the normality and homogeneity assumptions when both p - values are all less than 0.05.

8.6.2 With Threads and The Others

Now, let's try with threads model.

```
Call:
lm(formula = YBF ~ threads + bus_speed + cache_size + turbo_frequency +
    max_memory_size + max_temp, data = newIntel)

Residuals:
    Min       1Q   Median       3Q      Max
-1569.69  -376.92    62.99   448.27  1505.71

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.002e+03  2.529e+02   7.916 1.52e-14 ***
threads      -4.670e+00  1.056e+01  -0.442  0.6585
bus_speed     1.524e+02  1.711e+01   8.907 < 2e-16 ***
cache_size    -2.864e-02  1.499e-02  -1.911  0.0566 .
turbo_frequency  5.362e-01  5.301e-02  10.115 < 2e-16 ***
max_memory_size  1.142e-07  7.457e-08   1.532  0.1262
max_temp      -2.251e+01  2.287e+00  -9.840 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 560.3 on 514 degrees of freedom
Multiple R-squared:  0.4458,    Adjusted R-squared:  0.4394
F-statistic: 68.92 on 6 and 514 DF, p-value: < 2.2e-16
```

Figure 78: The results

```
Shapiro-Wilk normality test

data:  modelBF[["residuals"]]
W = 0.98495, p-value = 3.235e-05

studentized Breusch-Pagan test

data:  modelBF
BP = 93.486, df = 6, p-value < 2.2e-16
```

Figure 79: The results of Shapiro-Wilk and Breusch-Pagan tests

Similarly, this model is also not fit with linear regression's assumptions.

8.7 Fitting Turbo Frequency in Linear Regression Models

8.7.1 With Cores and The Others

Here is the model's summary with cores in the predictors list:

```
Call:
lm(formula = Y ~ cores + bus_speed + base_frequency + cache_size +
    max_memory_size + max_temp, data = newIntel)

Residuals:
    Min       1Q   Median       3Q      Max
-1398.94 -259.13    5.81   250.17  1232.70

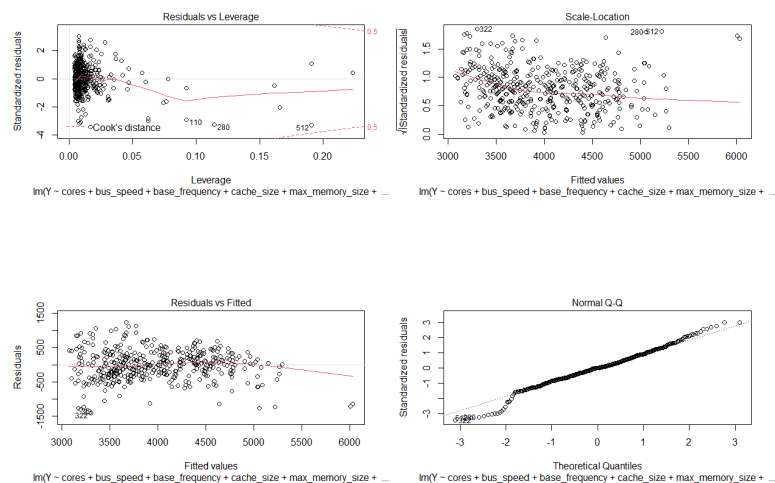
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.293e+03  1.939e+02   6.669 6.68e-11 ***
cores        -1.377e+02  1.623e+01  -8.483 2.35e-16 ***
bus_speed     4.862e+01  1.336e+01   3.638 0.000302 ***
base_frequency 2.849e-01  3.001e-02   9.494 < 2e-16 ***
cache_size    1.411e-01  9.673e-03  14.581 < 2e-16 ***
max_memory_size -2.424e-07  5.390e-08  -4.498 8.50e-06 ***
max_temp      1.079e+01  1.764e+00   6.113 1.93e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 413.1 on 514 degrees of freedom
Multiple R-squared:  0.6329,    Adjusted R-squared:  0.6286
F-statistic: 147.7 on 6 and 514 DF,  p-value: < 2.2e-16
```

Figure 80: The model's summary

From the table, the adjusted R-squared is 0.6286, which is above the average. Besides, all the predictors have a strong relationship with the dependent variable. Hence, we hope that there might be a chance of fitting a linear regression model in this case.

But, we still have to make sure our model meet the assumptions of linear regression models.



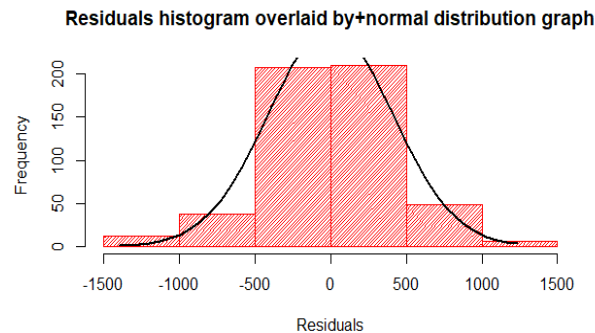


Figure 81: The plots for our model

The histogram and Normal Q-Q graph show that the model may follow the normal distribution. But, the other plots contain unusual pattern meaning that linearity and homogeneity assumptions are likely to be violated.

There exists a problematic amount of collinearity since VIF values of cores and cache size are far larger than 10.

cores	bus_speed	base_frequency	cache_size	max_memory_size
13.208797	1.705350	1.537018	12.083660	1.933898
max_temp				
1.305827				

Figure 82: The VIF result

```
> # Check for Normality
> shapiro.test(model[["residuals"]])

Shapiro-wilk normality test

data:  model[["residuals"]]
W = 0.98475, p-value = 2.832e-05

> # Check for homogeneity
> bptest(model)

studentized Breusch-Pagan test

data:  model
BP = 69.626, df = 6, p-value = 4.878e-13
```

Figure 83: The result of Shapiro-Wilk and Breusch-Pagan tests

As consequences, we will have to end this path of fitting the turbo frequency in a linear regression model.

8.7.2 With Threads and The Others

Similarly, we now establish the linear regression model with threads.


```
Call:
lm(formula = Y ~ threads + bus_speed + base_frequency + cache_size +
    max_memory_size + max_temp, data = newIntel)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-1569.67	-255.99	-9.62	250.29	1243.84

Coefficients:

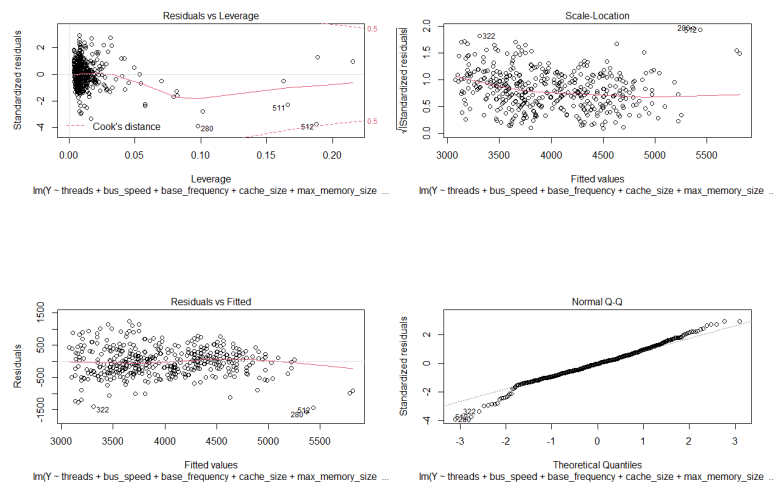
	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	1.091e+03	1.978e+02	5.516	5.49e-08 ***
threads	-4.741e+01	7.747e+00	-6.120	1.86e-09 ***
bus_speed	2.241e+01	1.394e+01	1.608	0.109
base_frequency	3.096e-01	3.061e-02	10.115	< 2e-16 ***
cache_size	1.225e-01	1.008e-02	12.158	< 2e-16 ***
max_memory_size	-2.815e-07	5.542e-08	-5.078	5.34e-07 ***
max_temp	1.313e+01	1.804e+00	7.275	1.30e-12 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 425.8 on 514 degrees of freedom
Multiple R-squared: 0.6099, Adjusted R-squared: 0.6054
F-statistic: 133.9 on 6 and 514 DF, p-value: < 2.2e-16

Figure 84: The model's summary

The result is the same with the previous model. It raises an opportunity to establish a linear regression model here. If our model pass all assumption, we will have to eliminate bus speed out of the model because of its exceeding $p - value$.



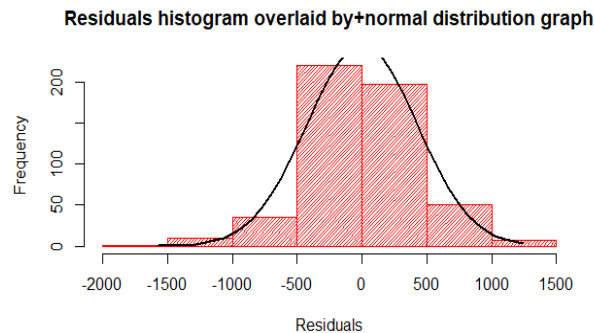


Figure 85: The plots for our model

threads	bus_speed	base_frequency	cache_size	max_memory_size
12.173240	1.746246	1.504925	12.338669	1.924668
max_temp				
1.285031				

Figure 86: The VIF result

Sadly, there is no differences on the outcome from the plots and VIF. Except for normality, all others assumption are breached.

```
> # Check for Normality
> shapiro.test(model[["residuals"]])

Shapiro-Wilk normality test

data:  model[["residuals"]]
W = 0.98733, p-value = 0.0001714

> # Check for homogeneity
> bptest(model)

studentized Breusch-Pagan test

data:  model
BP = 74.088, df = 6, p-value = 5.914e-14
```

Figure 87: The result of Shapiro-Wilk and Breusch-Pagan tests

The p -value in both tests is less than 0.05 has put an end to this model. We still can not explore the dependency of turbo frequency on the others predictors with linear regression model.

8.8 Summary

In conclusion, all of the variables, which have been selected to fit in a linear regression model, have failed some or all of the assumptions. Beside, in some circumstances, the R-squared value for the model is even below the average. In general, the linear regression model are not appropriate to predicts their dependency of variables in our data set.

However, there are so interesting points from the research that we need to mention. First of all, although the linear regression model is not a suitable approach, we still observe a strong

relationship between some dependent variables and some predictors such as the cache size, cores, threads... Secondly, the model summary for each dependent variables always indicate that there are some predictors that have a strong impact while the others does not. In other words, there exist a relationship between the dependent variable and predictors yet it is not a linear relationship because of its assumption failures.

Base on all analyses have been made, we have come to a prediction that may be a non-linear regression model will be a better choice. However, the time, knowledge and experience's limitations have prevented us from doing a further study using non-linear regression model with our data set. In the future, when the chance comes, we will try our best to construct research with the non-linear regression model.

9 References

1. Jay L.Devore. Probability and Statistics for Engineering and The Sciences, 9th Edition. California Polytechnic State University, San Luis Obispo.
2. Harold R.Lindman. Analysis of Variance in Experimental Design. Springer-Verlag, New York (1992).
3. Erjavec N. Tests for Homogeneity of Variance. In: Lovric M. (eds) International Encyclopedia of Statistical Science. Springer, Berlin, Heidelberg (2011).
4. H. Levene. Robust Tests for Equality of Variances. In: I. Olkin, et al., Eds., Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling. Stanford University Press, Palo Alto (1960); pp 278-292.
5. Jason T.Newsom. Post Hoc Tests. In: Univariate Quantitative Methods. Portland State University (2020).
6. Alexis Dinno. Nonparametric Pairwise Multiple Comparisons in Independent Groups Using Dunn's Test. The Stata Journal (2015); 15(1):292-300.
7. Susan Pedersen. Research Methods: Effect Sizes and "What If" Analyses as Supplements to Statistical Significance Tests. Journal of Early Intervention (2003); 25(4):310-319.
8. T.Van Hecke. Power Study of ANOVA Versus Kruskal-Wallis Test. Journal of Statistics and Management Systems (2012); 15(2-3):241-7.
9. J.D.Jobson. Multiple Linear Regression. In: Applied Multivariate Data Analysis. Springer Texts in Statistic. Springer, New York (1991).
10. Jamal I.Daoud. Multicollinearity and Regression Analysis. Journal of Physics: Conference Series (2017); 949.
11. Edward R.Mansfield and Billy P.Helms. Detecting Multicollinearity. The American Statistician (1982); 36(3a):158-160.
12. Aylin Alin. Multicollinearity. WIREs Computational Statistics (2010); 2(3):370-374.
13. Kelly H.Zou, Kemal Tuncali and Stuart G.Silverman. Correlation and Simple Linear Regression. Radiology (2003); 227(3).



14. P.J.Twomey and M.H.Kroll. How to use Linear Regression and Correlation in Quantitative Method Comparison Studies. *International Journal of Clinical Practice* (2008); 62(4):529-538.
15. Zofia Hanusz, Joanna Tarasinska and Wojciech Zielinski. Shapiro-Wilk Test with Known Mean. *REVSTAT - Statistic Journal* (2016); 14(1):89-100.
16. T.S. Breusch and A.R.Pagan. A Simple Test for Heteroscedasticity and Random Coefficient Variation. *The Econometric Society* (1979); 47(5):1287-1294.
17. Peter J.Rousseeuw. A Diagnostic Plot for Regression Outliers and Leverage Points. In: *Computational Statistics and Data Analysis* (1991); 11(1):127-129.
18. Joseph W.McKean, Simon J.Sheather and Thomas P.Hettmansperger. The Use and Interpretation of Residuals Based on Robust Estimation. *Journal of the American Statistical Association* (1993); 88(424):1254-1263.
19. C.P.Quesenberry and Chatles Quesenberry Jr. On the Distribution of Residuals form Fitted Parametric Models. *Journal of Statistical Computation and Simulation* (1982). 15(2-3):129-140.
20. T.Johnsson. A Procedure for Stepwise Regression Analysis. *Statistical Papers* 33, 21-29 (1992).
21. C.Agostinelli. Robust Stepwise Regression. *Journal of Applied Statistics* (2002); 29(6):825-840.