VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF APPLIED SCIENCE

PROBABILITY AND STATISTICS (MT2013)

Probability and Statistics Assignment - Semester 222

The impact of CPU's characteristics on its Thermal Design Power

---

Advisor:	Dr. Nguyen Tien Dung	
Student:	Le Hoang Duy	2152040
	Le Tran Nguyen Khoa	2152674
	Dinh Viet Thanh	2152966
	Dang Bao Tin	2152313
	Le Viet Tung	2153083

HO CHI MINH CITY, MAY 2023

Member list & Workload

No.	Full name	Student ID	Task	Contribution
1	Le Hoang Duy	2152040	Search for topics, theoratical background, code R	100%
2	Le Tran Nguyen Khoa	2152674	Data preprocessing cleaning, data clarification, ANOVA and writing report	100%
3	Dinh Viet Thanh	2152966	R code, testing regression model and R markdown code	100%
4	Dang Bao Tin	2152313	Managing progress, collecting tasks, R code and researching special models for regression analysis	100%
5	Le Viet Tung	2153083	Data preprocessing cleaning, data clarification, ANOVA and checking report	100%

Contents

1 Abstract

The development trend of CPU (Central Processing Unit) has caught great interest among the technician community in recent years. To improve the performance and tackle the heat-tradeoffs, many solutions and modifications in CPU design has been taken. In this project, we will analyze the data of CPUs manufactured by Intel as a case study. This data contains almost every information of a complete, usable CPU, from technical specifications, to recommend prices and intended end-users. Specifically, Thermal Design Power (TDP) would be our main focus, since it represents the heat barrier and roughly the power consumption of a CPU generation.

We made use of three regression models: Multiple Linear Regression model, Random Forrest Regression model and Logistic Regression model to examine the relationships between other attributes on Thermal Design Power. Along the way, Analysis of Variance (ANOVA) and non-parametric tests such as Kruskal-Wallis H-Test and Dunn's test were also use whenever normality of residuals and homoscedasticity were not satisfied.

The results of experimenting multiple models demonstrated that Multiple Linear Regression model was not as good as Random Forrest Regression model in predicting the TDP with respect to other attributes. We also found that, the Thermal Design Power in recent years was tremendously different from the past, indicating a big innovation is going on. In conclusion, by using statistical methods learnt in class, we got a deeper insight into how to analyze a data from scratch, and make conclusions about the trends and details that is not apparent.

2 Background

2.1 Regression Model

2.1.1 Linear Regression Model

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression is to model the linear relationship between the explanatory (independent) variables and response (dependent) variables. The formula of multiple linear regression:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

where:

- y_i is the dependent variable.
- x_{ip} is the explanatory variables.
- β_0 is the y-intercept (constant term).
- β_p is the slope coefficients for each explanatory variable.
- ϵ is the model's error term (also known as the residuals).

When using linear regression, there are several assumptions that are typically made.

Assumption 1: Linearity, the relationship between the dependent variable and the independent variable is linear.

Assumption 2: Independence, the observations are independent of each other

$$Cov(\varepsilon_i, \varepsilon_j) = 0, i \neq j$$

where $Cov(\varepsilon_i, \varepsilon_j)$ is the covariance between the errors for observations i and j.

Assumption 3: Homoscedasticity, the variance of the errors is constant across all levels of the independent variable(s)

$$Var(\varepsilon_i) = \sigma^2, \forall i$$

where $Var(\varepsilon_i)$ is the variance of the error for observations i and σ^2 is a constant.

Assumption 4: Normality, the errors are normally distributed

$$\varepsilon \sim N(0, \sigma^2)$$

where ε is the error term and $N(0, \sigma^2)$ denotes a normal distribution with mean 0 and variance σ^2.

2.1.2 Random Forest regression

Random Forest regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.

Random forest is an ensemble of decision trees. This is to say that many trees, constructed in a certain "random" way form a Random Forest, whether:

- Each tree is created from a different sample of rows and at each node, a different sample of features is selected for splitting.
- Each of the trees makes its own individual prediction.
- These predictions are then averaged to produce a single result.
- The averaging makes a Random Forest better than a single Decision Tree hence improves its accuracy and reduces overfitting. A prediction from the Random Forest Regressor is an average of the predictions produced by the trees in the forest.

When using random forest regression, there are several assumptions that are typically made.

Assumption 1: Independence of observations, this assumption states that the observations in the dataset used for building the Random Forest regression model should be independent.

Assumption 2: Input feature representation, this assumption emphasizes the importance of appropriate representation of input features (independent variables).

Assumption 3: Decision tree assumptions, Random Forest regression is an ensemble of decision trees, and the assumptions of individual decision trees within the Random Forest ensemble apply.

Assumption 4: Appropriate hyperparameter tuning, Random Forest regression has several hyperparameters, such as the number of trees, the maximum depth of trees, the minimum number of samples required to split a node, among others. However, there is no specific mathematical formula for hyperparameter tuning, as it depends on the specific dataset and problem.

2.1.3 Logistic regression

Logistic regression is one of the most commonly used forms of nonlinear regression. It is used to estimate the probability of an event based on one or more independent variables. Logistic regression identifies the relationships between the enumerated variables and independent variables using the probability theory.

A variable is said to be enumerated if it can possess only one value from a given set of values. Logistic Regression Models are generally used in cases when the rate of growth does not remain constant over a period of time. For example -when a new technology is introduced in the market, firstly its demand increases at a faster rate but then gradually slows down.

When using Logistic regression, there are several assumptions that are typically made.

Assumption 1: Binary outcome, Logistic regression models the probability (P) of an event occurring, which is typically represented by a binary outcome variable (e.g., success/failure, yes/no, 1/0). The binary outcome variable can be denoted as Y, where Y = 1 represents the occurrence of the event of interest and Y = 0 represents the absence of the event.

Assumption 2: Independence of observations, Observations should be independent of each other, meaning that the value of the dependent variable for one observation should not be influenced by the value of the dependent variable for another observation.

Assumption 3: Linearity of independent variables and log-odds, Logistic regression models the log-odds (or logit) of the binary outcome, denoted as $ln(p/(1-p))$, where p is the probability of the event occurring. the relationship between the predictor variables (denoted as X) and the log-odds of the binary outcome is assumed to be linear. Mathematically, this can be represented as:

$$ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

where: β_0, β_1, β_2, ..., β_k are the model parameters (coefficients) to be estimated, and x_1, x_2, ..., x_k are the predictor variables.

Assumption 4: Large sample size, Logistic regression assumes a sufficiently large sample size to ensure reliable estimates of the model parameters. While there is no strict cutoff for sample size, a rule of thumb is to have at least 10-20 observations per predictor variable to ensure stable parameter estimates and reliable statistical inference.

2.2 Statistics measurements

2.2.1 The Q-Q plot

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 residuals are normally distributed, we can use a Normal Q-Q plot to check that assumption. 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.

2.2.2 R-Squared (R^2)

R^2 is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable in a regression model.

Whereas correlation explains the strength of the relationship between an independent and a dependent variable, R-squared explains the extent to which the variance of one variable explains the variance of the second variable. So, if the R^2 of a model is 0.50, then approximately half of the observed variation can be explained by the model's inputs.

The formula for R-squared:

$$R^2 = 1 - \frac{\text{Unexplained Variation}}{\text{Total Variation}}$$

The calculation of R-squared requires several steps. This includes taking the data points (observations) of dependent and independent variables and finding the line of best fit, often from a regression model. From there, you would calculate predicted values, subtract actual values, and square the results. This yields a list of errors squared, which is then summed and equals the unexplained variance.

The **adjusted coefficient of determination** is the multiple coefficient of determination R^2 modified to account for the number of variables and the sample size. It is calculated by

$$\text{Adjusted } R^2 = 1 - \frac{n-1}{n-(k+1)} \times (1 - R^2)$$

2.2.3 P-Value

In statistics, the p-value is a measure of the evidence against a null hypothesis. It is the probability of observing a test statistic as extreme as, or more extreme than, the one calculated from the data, assuming that the null hypothesis is true.

In other words, the p-value is the probability of obtaining the observed results or more extreme results, assuming that the null hypothesis is true. If the p-value is low (usually less than 0.05), it suggests that the observed results are unlikely to be due to chance and provides evidence against the null hypothesis. Conversely, if the p-value is high, it suggests that the observed results are likely to be due to chance, and there is insufficient evidence to reject the null hypothesis.

The p-value is an important concept in hypothesis testing, which is a common statistical method used to make decisions based on data. It helps researchers determine whether the observed data supports or contradicts a particular hypothesis.

2.3 Analysis of Variance ANOVA

Analysis of variance (ANOVA) is a statistical method used to test for differences among two or more population means by analyzing the variances of samples taken from the populations.

One-way ANOVA is a statistical method to compare the variances of multiple levels of a single factor.

For each observation under the treatment i under the j observation called y_{ij} we have the linear combination:

$$y_{ij} = \mu + \tau_i + \epsilon_{ij} \begin{cases} i = 1, 2, ..., a. \\ j = 1, 2, ..., n. \end{cases}$$

where,

- μ is the overall mean.
- τ_i is the effect of the ith treatment effect.
- ϵ_{ij} is a random component error.

We could rewritten the model as.

$$y_{ij} = \mu_i + \epsilon_{ij} \begin{cases} i = 1, 2, ..., a. \\ j = 1, 2, ..., n. \end{cases}$$

where,

- $\mu_i = \mu + \tau_i$
- τ_i is the effect of the ith treatment effect.
- ϵ_{ij} is a random component error.

To perform ANOVA, the following assumption is made: ϵ_{ij} is normally and independently distributed : $\epsilon_{ij} \approx N(0, \sigma^2)$, and each treatment is a sample that follows $N(0, \sigma^2)$.

1. Normality: The populations have distributions that are approximately normal.
2. Homoscedasticity : The populations have the same variance
3. Independent: the data is random and independent.

However the Normality and Homogeneity of variance are only loose requirement as the method still well despite failing these assumption[?] However we will also use the Kruskal - Wallis test for anything that do not sastify the assumption. We want to test the Null hypothesis:

$$\begin{cases} H_0 : \mu_1 = \mu_2 = ... = \mu_n \\ H_1 : \text{two mean are different} \end{cases}$$

Total sum of squares:

$$SS_T = \sum_{i=1}^{a} \sum_{j=1}^{n} (y_{ij} - \bar{y})^2$$

$$SS_T = n\sum_{i=1}^{a}(\bar{y}_i - \bar{y})^2 + \sum_{i=1}^{a}\sum_{j=1}^{n}(y_{ij} - \bar{y}_i)^2$$

or

$$SS_T = SS_{Treatment} + SS_{Error}$$

where degree of freedom is:

$$df(SS_T) = N - 1 \quad df(SS_{Treatment}) = a - 1 \quad df(SS_{Error}) = N - a$$

Mean square for treatments:

$$MS_{Treatment} = SS_{Treatment}/df(SS_{Treatment})$$

$$MS_{Treatment} = SS_{Treatment}/(a-1)$$

Mean square for error:

$$MS_{Error} = SS_{Treatment}/df(SS_{Error})$$

$$MS_{Error} = SS_{Treatment}/(N-a)$$

F test statistic:

$$F_0 = \frac{MS_{Treatment}}{MS_{Error}}$$

If

$$F_0 > F_{\alpha,a-1,a(n-1)}$$

2.4 Kruskal - Wallis H-Test

Kruskal - Wallis test which uses ranks of data from three or more independent simple random samples to test the null hypothesis that the samples come from populations with the same median. The Kruskal-Wallis test for equal medians does not require normal distributions, so it is a distribution-free or non-parametric test. In applying the Kruskal - Wallis test we need to compute the test statistic H.

$$H = \frac{12}{N(N+1) * \sum_{i=1}^{k}\frac{R_i^2}{n_i} - 3(N+1)}$$

where:

- N is the number of values from all combined samples.
- R_i is the sum of ranks from a particular sample, and n_i is the number of values from the corresponding rank sum.

- n_i is the number of values from the corresponding rank sum.

$$\begin{cases} H_0 : \text{The samples come from populations with the same median.} \\ H_1 : \text{The samples come from populations with medians that are not all equal.} \end{cases}$$

2.5 Levene test

Levene's test is used to test if k samples have equal variance. In this assignment, we will use it as the primary tool for testing the Homogeneity of variance.

Given a variable Y with sample of size N divided into k subgroups where N_i is the sample size of the ith subgroup, the Levene test is defined as:

$$\begin{cases} H_0 : \sigma_1^2 = \sigma_2^2 = ... = \sigma_k^2 \\ H_1 : \text{there are at least one pair with unequal variance.} \end{cases}$$

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

where Z_{ij} cahn have one of these following definitions:

- $Z_{ij} = Y_{iJ} - \bar{Y}_i$ where $\bar{Y}_i$ is the mean of the ith subgroup.
- $Z_{ij} = Y_{iJ} - \tilde{Y}_i$ where $\tilde{Y}_i$ is the median of the ith subgroup.
- $Z_{ij} = Y_{iJ} - \bar{Y}_i^{'}$ where $\bar{Y}_i^{'}$ is the trimmed mean of the ith subgroup.

The three choice for detemining Z_{ij} determine the robustness and power of Levene's test. We will choose choice where $\tilde{Y}_i$ is the median as it is the default choice of LeveneTest in R

2.6 Shapiro-Wilk test

The Shapiro-Wilk test, calculates a W statistic that tests whether a random sample, x_1, x_2, ..., x_n come from a normal distribution. The W statistic is calculated as:

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

where:

- $x_{(i)}$ are the ordered sample values
- a_i are the constant generated from the means, variance and covariance of the order of a sample of size n from a normal distribution.
- $\bar{x}$ is the sample mean

We would like to use this test to test the Null hypothesis:

$$\begin{cases} H_0 : \text{the population is normally distributed} \\ H_1 : \text{the population is not normally distributed} \end{cases}$$

if the p-value is less than α then we can reject the null hypothesis this test and consider our data to not be Normally distributed.

2.7 Post-hoc comparison tests

Post-hoc comparison tests are usually used to identify the differences between multiple groups after the study has been concluded. Specifically, it is used to see the pairwise differences between different groups of the dataset after an ANOVA is done and the conclusion has been drawn that there is statistically significant result.[3]

2.7.1 Tukey HSD test

Tukey HSD's test compares the means of every treatment to the means of every other treatment; that is, it applies simultaneously to the set of all pairwise comparisons

Tukey HSD test perform a pairwise comparison between the means of the treatments by seeing whether their difference is statistically significant as compared to the expected standard error. It makes use of studentized range statistic:

$$Q = \frac{\bar{y}_{\max} - \bar{y}_{\min}}{SE} \tag{1}$$

where, $\bar{y}_{\max}$ and $\bar{y}_{\min}$ are the largest and smallest sample means, respectively.

This test indicates two means are different if $Q > g(\alpha, f) \times S$[2], where, S is the standard error of this statistic, and $g(\alpha, f)$ is studentized range distribution of significant level α and even degree of freedom f.

2.7.2 Dunn's z-test

Dunn's z-test statistic approximates exact rank-sum test statistics by using the mean rankings of the outcome in each group from the preceding Kruskal-Wallis test and basing inference on the differences in mean ranks in each group.[1]

The statistic to compare the difference in mean between group A and group B.

$$z = \frac{\bar{W}_A - \bar{W}_B}{S} \tag{2}$$

where S is the standard error.

$$S = \sqrt{(\frac{N(N+1)}{12} - \frac{\sum_{r}^{s=1} \tau_s^3 - \tau_s}{12(N-1)})(\frac{1}{n_A} + \frac{1}{n_B})} \tag{3}$$

where, N is the total number of observations 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.

In our project, we use The Bonferroni's p-value adjustment, which basically multiply the p-value with a constant m.

3 Data description

3.1 Overview of the dataset

This dataset contains detailed specifications, release dates, and release prices of Intel CPUs. These specifications include some important attributes of a CPU, and describes the performance (through *Base frequency*), power consumption and heat (through *Thermal design power*), and the technology trend (through *Number of cores* and *Lithography*).

CPU (Central Processing Unit) is the most fundamental of a computer. CPU basically loads the instruction and execute it clock by clock, and produces return values. Modern computers divides itself into many small processing units (called cores). During execution, it generates a lot of heat.

The dataset is retrieved from Computer Parts (CPUs and GPUs) Dataset (Kaggle) by author Ilissek. Some notable attributes of this dataset are:

- Vertical Segment (or Market segment) : describes which market a CPU is intended for. There are for markets segments: Mobile, Desktop, Embedded and Server.
- Status : the current condition of the supply of CPU. They can be: Announce (incoming sales), Launched (currently active, fully supported), End of life (stop manufacturing, still supported) and End of interactive support (out of the support cycle).
- Launch date : the quarter-year date of which the CPU was available on the market.
- Lithography : the chip printing technique that was used to manufacture the CPU. Roughly speaking, as technology advances forward, this technique is getting smaller and smaller.
- Recommended Customer Price : the price recommended by Intel for retailers selling the CPU.
- Number of Cores : number of processing units on one CPU. More core does not mean better CPU. However, it helps utilizing parallel computing (multiple programs running at the same time).
- Base frequency : expected operating clock rate of a CPU. Larger frequency mean faster clock rate and therefore better performance.
- Thermal design power : theoretical heat and power consumption ceiling, the amount of heat needed to be cooled for normal operation of the CPU.
- Temperature (T) : the maximum temperature allowed on the CPU, before it could be damaged.

3.2 Importing and Preprocessing

```r
pacman::p_load(
    rio,     # for dealing with basic import export
    ggplot2, # for dealing with plot formats
    zoo      # for dealing with year quarter formats
    car,     # for levent and shapiro
    FSA,     # for Dunn test
)

data <- import("./cpu-raw.csv") # rio::import

data <- data[, c("Vertical_Segment","Status","Launch_Date",
                 "Lithography","Recommended_Customer_Price",
                 "nb_of_Cores","Processor_Base_Frequency",
                 "TDP","T")]
```

The primary packages used in this process are:

- `rio` : for intuitive I/O code. With this package, import and export dataset is easier and safer. It could also handle multiple file formats, so that we do not have to change the command each time we change the file format.
- `zoo` : for year-quarter format. In our data, the `Launch date` is in non-standard format, and difficult to be operated on. This package helps to transform into standard year-quarter format, and provides useful operations, such as plotting and taking difference on these formats.
- `ggplot2` : is a famous plotting package for R language.
- `car` : for the levent and shapiro for the assumption.
- `FSA` : for the post hoc test.

The rest of the code is just choosing the attributes that are useful for our purpose.

The original labels are very long and descriptive, we might not want that such level of details during coding. Therefore, the labels are suppressed into small, compact abbreviations. Then, we export the code to `cpu-clean.csv`.

```r
names(data) <- c("market", "status", "ldate", "litho",
    "rprice", "ncore", "bfreq", "tdp",
    "temp")
```

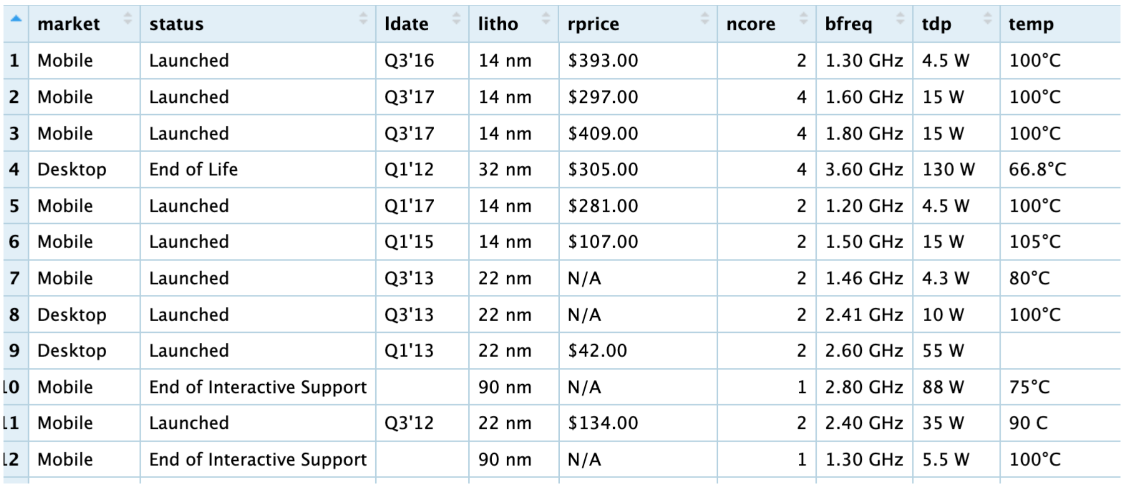
```

export("./cpu-clean.csv")
```

3.3 Data cleaning

After choosing the approriate attributes, we now have the subset of the original raw dataset. However, since the values vary in types (such as string, non-standard year-quarter format and numeric-string), we might want transform them into reproducible types, so that the analysis later on is easier, homogeneous and accurate.

Note that this cleaning process **does not** remove the `NA` values, unless necessary. The reason is that, in one instance, there might be important values that should not be eliminated. Under different scopes of study, we can not treat instances with `NA` as an invalid datum for all scopes. In later sections, when we focus on a specific pattern of the data, only by then that the data will have a tailored `NA` cleaning, and we will not, by chance, loose any important instance.

This process took `/rcode/cpu-short.csv` from the importing procedure above as an input, and produce `/rcode/cpu-clean.csv` as an output.



	market	status	ldate	litho	rprice	ncore	bfreq	tdp	temp
1	Mobile	Launched	Q3'16	14 nm	$393.00	2	1.30 GHz	4.5 W	100°C
2	Mobile	Launched	Q3'17	14 nm	$297.00	4	1.60 GHz	15 W	100°C
3	Mobile	Launched	Q3'17	14 nm	$409.00	4	1.80 GHz	15 W	100°C
4	Desktop	End of Life	Q1'12	32 nm	$305.00	4	3.60 GHz	130 W	66.8°C
5	Mobile	Launched	Q1'17	14 nm	$281.00	2	1.20 GHz	4.5 W	100°C
6	Mobile	Launched	Q1'15	14 nm	$107.00	2	1.50 GHz	15 W	105°C
7	Mobile	Launched	Q3'13	22 nm	N/A	2	1.46 GHz	4.3 W	80°C
8	Desktop	Launched	Q3'13	22 nm	N/A	2	2.41 GHz	10 W	100°C
9	Desktop	Launched	Q1'13	22 nm	$42.00	2	2.60 GHz	55 W	
10	Mobile	End of Interactive Support		90 nm	N/A	1	2.80 GHz	88 W	75°C
11	Mobile	Launched	Q3'12	22 nm	$134.00	2	2.40 GHz	35 W	90 C
12	Mobile	End of Interactive Support		90 nm	N/A	1	1.30 GHz	5.5 W	100°C

Figure 1: *Data before cleaning.*

`market` and `status` are left unchanged, since the values are straightforward. The remaining attributes (columns) are processed as followed:

Launch dates (ldate)

```
data[,"ldate"] <- ( as.yearqtr(data[,"ldate"], format = "Q%q'%y"))
```

Our goal is to transform raw, non-standard year-quarter into `zoo`'s standards. The function `as.yearqtr` takes a column and a format string as parameters. The format string is represented as: "Q%q'%y", in which two flags "%q", "%y" stands for quarter and year, respectively. The format string hints the function to know the positions of quarter and year in our raw string.

Lithography (litho)

```r
data[,"litho"] <- as.numeric(gsub(" nm", "", data[,"litho"]))
```

Our goal is to cut out "nm", since every entry is recorded in nanometers anyway. In this code, gsub substitutes the pattern " nm" to "". Notice that the pattern are regular expressions, and would be used intensively during this cleaning process.

Recommended Customer Price (rprice)

```r
data[,"rprice"] <- gsub("(^\\$(\\d)+.(\\d)+ - )", "", data[,"rprice"])
data$rprice <- ifelse(data$rprice == "N/A", NA, data$rprice)
data$rprice <- as.numeric(gsub('\\$|,', '', data$rprice))
```

Some rprice values have ranges instead of sole numbers. We want to cut out uneccesary characters and only keep the largest price. After that, we eliminate $ symbol from the string, as well as cast the string to numeric type.

Base frequency (bfreq)

```r
data[,"bfreq"] <- as.numeric(gsub("( GHz)|( MHz)", "", data[,"bfreq"]))
data<- data[!is.na(data$bfreq), ]
data$bfreq[data$bfreq > 10] <- data$bfreq[data$bfreq > 10]*0.001
```

Our goal is to cut out "GHz" and "MHz" from the string, and convert all "MHz" values into "GHz". Heuristically, we observed that any value greater than 10 must be MHz, so we can transform every value like that will be multiply by 0.001 to get the according MHz value.

Thermal design power (TDP)

Same as for the litho our goal is to simply also cut out W from the observation and change it to numeric type.

```r
data[,"tdp"] <- as.numeric(gsub(" W", "", data[,"tdp"]))
```

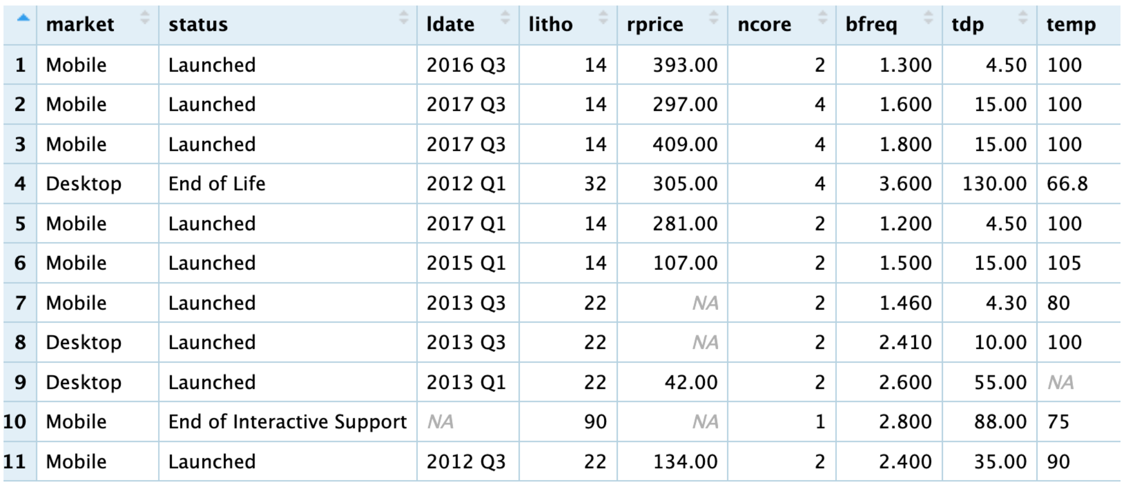
Temperature (temp)

```r
data[,"temp"] <- (gsub("[^0-9.\\-]+", ",", data[,"temp"]))
for (i in seq_along(data[["temp"]])) {
    temp_values <- strsplit(data[i, "temp"], ",")
    temp_values <- unlist(lapply(temp_values, as.numeric))
    max_value <- max(temp_values, na.rm = TRUE)
    if (is.infinite(max_value)) {
        max_value <- NA
    }
    data[i, "temp"] <- max_value
}

export(data, "cpu-clean.csv")
```

Our goal is to only match the numeric values, then, take the maximum among those. The approach to processing the complicated strings in `temp` is described as follows:

- First, we attempt to match every decimal numbers possible, including the irrelevant number. The rest are replaced with commas `","`. The result of this process will create a string of numbers separated by commas. By doing this, the numbers are well isolated for our purpose.
- Second, we split these numbers and form a vector of them. This can be done through `strsplit` function. Notice that our numbers are still in string format.
- Third, we cast all these strings to numeric and push them into a vector of values using `unlist` and `lapply`
- Fourth, we find the maximum among all these values. Invalid numbers will automatically become $-\infty$, and will be further treated as `NA`.
- Notice that, we must loop through each row of the list to accomplish the above procedure.

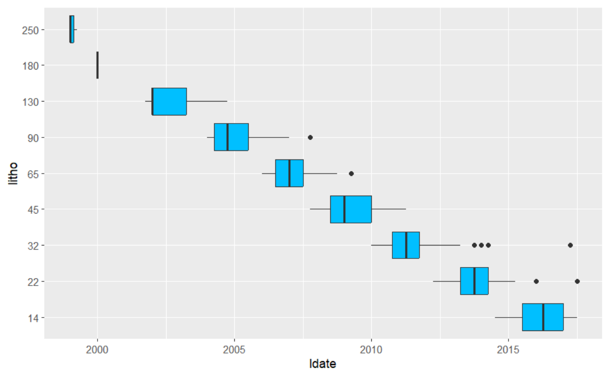
Finally, the program produces `cpu-clean.csv` as a cleaned data, ready for further exploitation in the later sections.



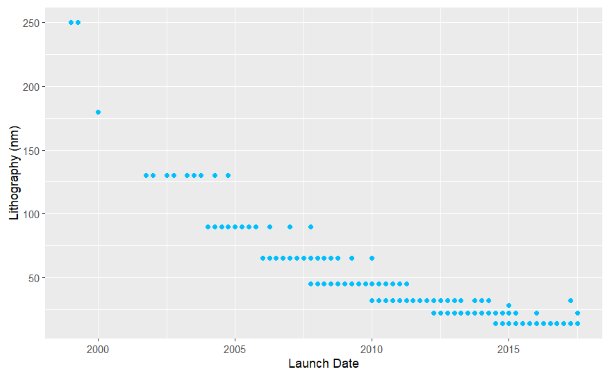
	market	status	ldate	litho	rprice	ncore	bfreq	tdp	temp
1	Mobile	Launched	2016 Q3	14	393.00	2	1.300	4.50	100
2	Mobile	Launched	2017 Q3	14	297.00	4	1.600	15.00	100
3	Mobile	Launched	2017 Q3	14	409.00	4	1.800	15.00	100
4	Desktop	End of Life	2012 Q1	32	305.00	4	3.600	130.00	66.8
5	Mobile	Launched	2017 Q1	14	281.00	2	1.200	4.50	100
6	Mobile	Launched	2015 Q1	14	107.00	2	1.500	15.00	105
7	Mobile	Launched	2013 Q3	22	NA	2	1.460	4.30	80
8	Desktop	Launched	2013 Q3	22	NA	2	2.410	10.00	100
9	Desktop	Launched	2013 Q1	22	42.00	2	2.600	55.00	NA
10	Mobile	End of Interactive Support	NA	90	NA	1	2.800	88.00	75
11	Mobile	Launched	2012 Q3	22	134.00	2	2.400	35.00	90

Figure 2: *Data after cleaning.*

4 Data clarification



(a) *Box plots and Summary of Lithography*

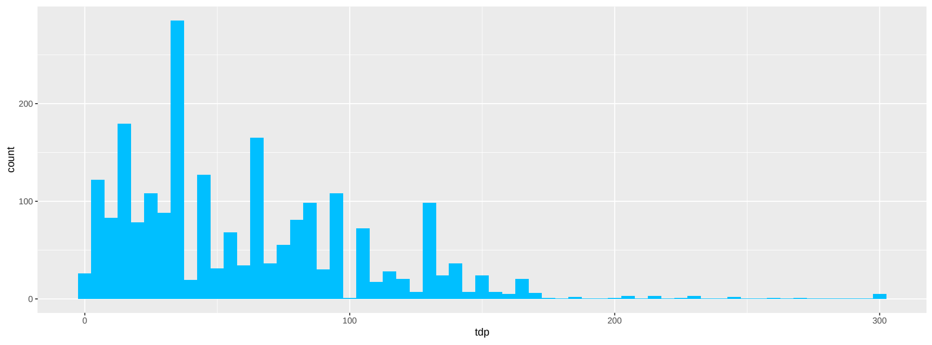


(b) *Plot of lithography over time.*

Figure 3: *Lithography plots*

```r
ggplot(data, aes(x = ldate, y = litho)) +
  geom_boxplot(fill="deepskyblue")

ggplot(data, aes(x = ldate, y = litho)) +
    geom_point(shape = 1,color = "blue") +
    labs(x = "Launch Date", y = "Lithography (nm)")
```

The scatter plot of *Lithography* [**Figure 3**] shows that it is getting smaller over time, and is categorized into specific time intervals. *Lithography* spans the distribution over an interval of time which make it more powerful than *Launch date*. In our models, we always use *Lithography* instead of *Launch date*. Which we will be proving later in 5.2.2.

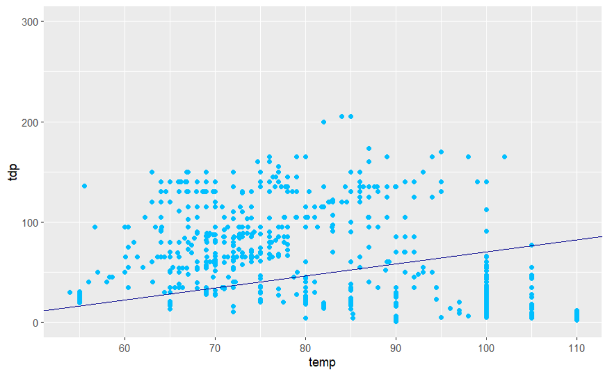


```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  0.025  25.000  47.000  60.242  85.000 300.000      49 
```

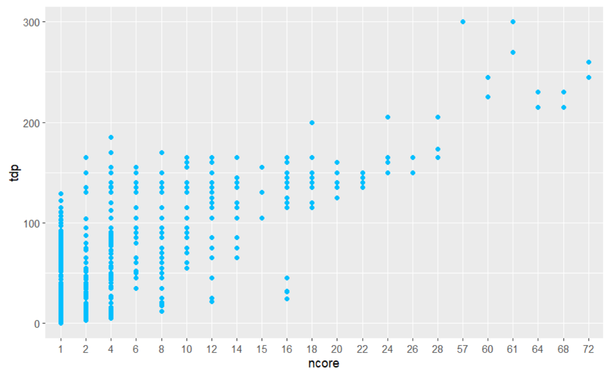
Figure 4: *Histogram and Summary of Thermal Design Power*

```r
ggplot(data, aes(x = tdp)) +
  geom_histogram(binwidth = 5, fill = "deepskyblue")

summary(data$tdp)
```

[**Figure 4**] The occurrences of values $\geq$ 150W are very rare, so they could be treated as outliers in further analysis.

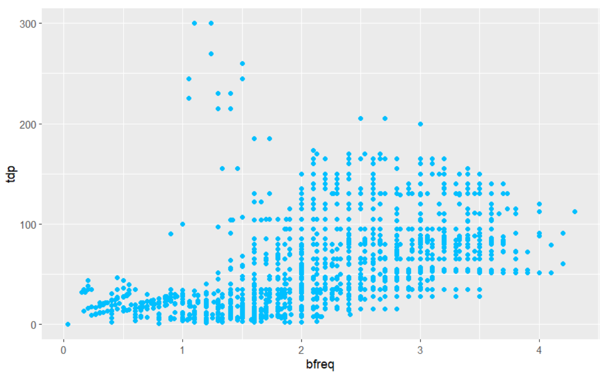


(a) *Different trends of TDP on different ranges of temperature*

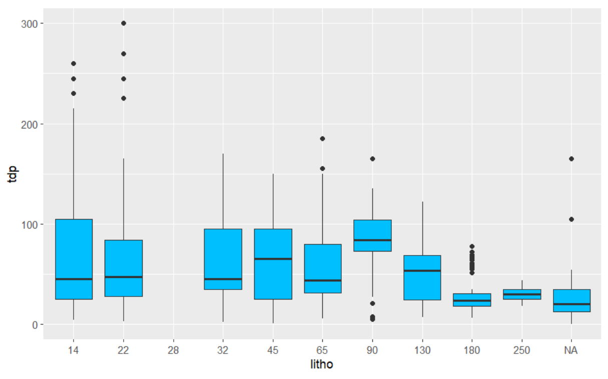


(b) *Increasing trend of no. cores and TDP*

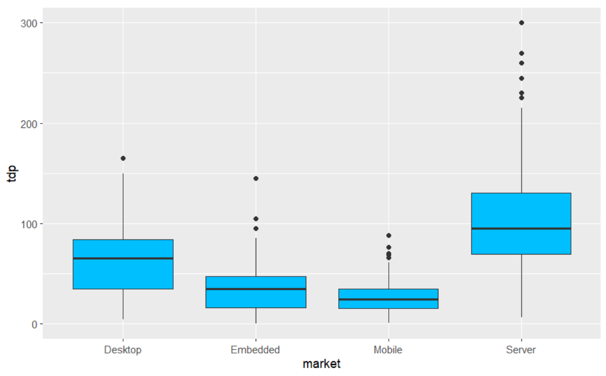
```r
ggplot(data, aes(x = tdp)) +
  ggplot(data, aes(x = temp, y = tdp)) +
  geom_point(color = "deepskyblue", ) +
  geom_abline(mapping = aes(intercept = -50, slope = 1.2), color =
  "darkblue")

ggplot(data, aes(x = ncore, y = tdp)) +
  geom_point(color = "deepskyblue")

ggplot(data, aes(x = bfreq, y = tdp)) +
  geom_point(color = "deepskyblue")

ggplot(data, aes(x = litho, y = tdp)) +
  geom_boxplot(fill = "deepskyblue")

ggplot(data, aes(x = market, y = tdp)) +
  geom_boxplot(fill = "deepskyblue")
```



(a) *Increasing trend of Base frequency and TDP*



(b) *TDP within different Lithography*



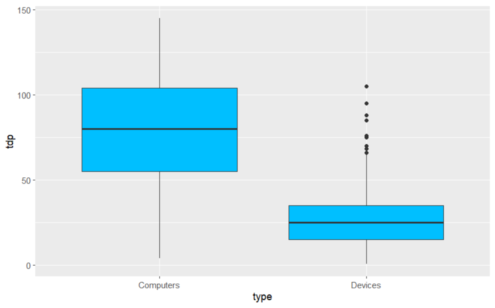
(c) *Differences of TDP in different Markets*

Figure 6: *The relationship visualizations between TDP and other factors*

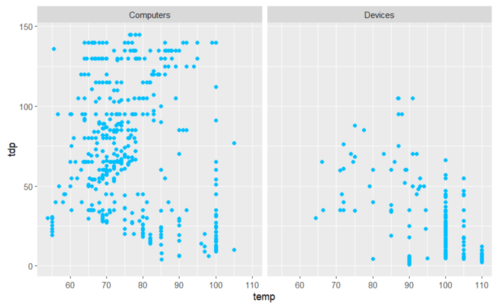
From the above visualizations, we could make a few comments from these relationships, which motivates us to use certain Regression models:

- **[Figure 5b]** `tdp ~ ncore` : As `ncore` increases, `TDP` also increases. This motivates us to use a Linear regression model. However, we also observe that, there are distinctive "clusters" on different ranges. Maybe a decision-based model, like Random Forrest, is better.
- **[Figure 6a]** `tdp ~ bfreq` : The data for this relationship is quite variant. However, the dominant trend is still linearly increasing, using Linear regression can be reliable.
- **[Figure 6b]** `tdp ~ litho` : There is no clear upward and downward trend. Instead, `TDP` is converging and gets more and more stable over time.
- **[Figure 5a, 6c]** `tdp ~ temp` and `tdp ~ market` : There are two different trends happening: above the reference line is increasing trend, while below is decreasing trend. In fact, we have a feeling that these two trends come from different market segments `type=(Desktop + Server)` `(Mobile + Embedded)`, so we would plot them out to verify:

```r
data$type <- ifelse(data$market == 'Server' | data$market == 'Desktop',
"Computers", "Devices")
data$type <- as.factor(data$type)

ggplot(data, aes(x = type, y = tdp)) +
    geom_boxplot(fill = "deepskyblue")

ggplot(data, aes(x = temp, y = tdp)) +
    geom_point(color = "deepskyblue", ) +
facet_wrap(~data$type)
```



(a) *TDP between Types.*



(b) *TDP in different Market segmentation*

Turns out, different trends belongs to different Market segments (`type`). This could be helpful for us to classify the `TDP` into different markets using Logistic regression.

5 Data analysis

As stated in our Report's title, the main focus of the analysis will be *Thermal Design Power* and the affect of other attributes (variables) on this dependent variable.

Thermal Design Power `TDP` is an interesting characteristic that comes with the CPU. Theoretically, this is the maximum amount of heat that can be released by its cooling system, which should never be exceeded. This characteristic somehow represents the heat barrier of a processor, and if its cooling system is designed to deal with almost all the heat it release, `TDP` roughly equates to its power consumption. The higher the `TDP`, the better the CPU's cooling system, but also demonstrates the amount of energy consumption required to dissipate the heat.

The reason why we did not use *Temperature* but instead *Thermal Design Power* is that we want to find the amount of energy needed to keep the CPU operational properly, not the temperature point that the CPU can work but be potentially damaged.

We would try to utilize some Regression models to predict its `TDP` via other configurations (or determinants) such as *Thermal design power*, *Number of cores*, *Base frequency*, *Temperature*, *Lithography*, *Market* and its *Status*. Also, we will see if with the determined `TDP`, we can classify the CPU, whether it belongs to mass computing devices (Server) or personal usage (Desktop and Mobile).

5.1 Data preparation

First, we load the cleaned data set from the cleaning process above.

```r
pacman::p_load(
    rio,     # for imports & exports
    ggplot2, # for plots
    zoo      # for year-quarter formats
)

data <- import("cpu-clean.csv") # rio::import
```

Refer to [**Figure 4**] and our statement previously, the occurrences of values ≥ 150 is rare, we decided to cut them out from our data set. At the same time, we also remove the `NAs` rows from the data set, note that only the `NAs` associated with specific columns are removed, the reason not to remove all is described in **Section 3.3**.

```r
data <- data[data$tdp < 150, ]
data <- data[!is.na(data$tdp), ]
data <- data[!is.na(data$bfreq), ]
data <- data[!is.na(data$litho), ]
data <- data[!is.na(data$ncore), ]
data <- data[!is.na(data$temp), ]
```

Because we would make use of Regression models to capture the relationships, a Test Set and a Training Set must be present to perform cross-validation to test the fitness of the model, besides visualization method by drawing graphs and checking other coefficients. The original data set is splitted into two smaller sets, training set and validated set (or test set). In detail, 80% data is used for training set while 20% is used for test set.

```r
set.seed(123)

train_indices <- sample(1:nrow(data), nrow(data) * 0.8)
train <- data[train_indices, ]
test <- data[-train_indices, ]
```

- To make a random generation homogeneous among all our tests, we specified a seed (123). Each time we run the cell, we would get consistent results of the split.
- `sample()` helps us to take a sample from all the elements of our data set using without replacement. The return value of this function are the indices of 80% the data set, randomly chosen.
- After that, we use a simple indexing technique to assign to a training set and test set, accordingly.

5.2 The relationships between TDP and other factors

5.2.1 Hypothesis testing

To explain further the statements we made above, we will assess the following assumptions by using several techniques such as hypothesis testing, ANOVA and computing the covariances.

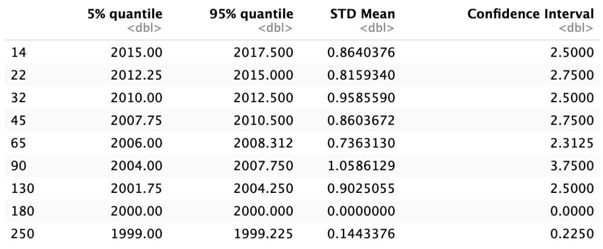
The assumptions we are aiming to:

1. Lithography as a CPU era.
2. Thermal Design Power with respect to lithography.

5.2.2 Lithography as a CPU era.

In this small section, we will demonstrate why **Lithography as a better representative than Launch date**. To do that, we start by looking at the confidence interval and the visualizations of Lithography over the years.

```r
data$litho <- as.factor(data$litho)

retval <- data.frame(NA, NA, NA, NA)
names(retval)<-c("5% quantile","95% quantile", "STD Mean", "Confidence Interval")

for (lit in levels(data$litho))
{
      quants <- quantile(
        data[data\$litho == lit, ]\$ldate,
        na.rm = T,
        probs = c(0.05,0.95)
      )

      dates <- data[data\$litho == lit, ]\$ldate

      new_row <- data.frame(quants[1], quants[2], mean(sd(dates, na.rm=TRUE), na.rm=TRUE),quants[2]-quants[1])
      names(new_row)<-c("5% quantile","95% quantile", "STD Mean", "Confidence Interval")

      retval <- rbind(retval, new_row)
      rm(dates)
}
rownames(retval) <- c("NULL", levels(data$litho))
retval <- retval[-1,]

print(retval)
```



	5% quantile <dbl>	95% quantile <dbl>	STD Mean <dbl>	Confidence Interval <dbl>
14	2015.00	2017.500	0.8640376	2.5000
22	2012.25	2015.000	0.8159340	2.7500
32	2010.00	2012.500	0.9585590	2.5000
45	2007.75	2010.500	0.8603672	2.7500
65	2006.00	2008.312	0.7363130	2.3125
90	2004.00	2007.750	1.0586129	3.7500
130	2001.75	2004.250	0.9025055	2.5000
180	2000.00	2000.000	0.0000000	0.0000
250	1999.00	1999.225	0.1443376	0.2250

Figure 8: *Summary of Confidence Interval of Lithography over the years*

Looking at the Mean of Standard Deviation (`STD Mean`), these *means are pretty stable*, and the

Confidence Interval column tells us that most of the era of CPU design *spans for about two and a half years*, and these era are approximately mutually exclusive. This is its big advantage over using Launch date only, because we can now consider a range of values and group different launch dates together wich possibly share the same characteristics. So, everytime we wants to refer to a period of CPU, we always use Lithography.

5.2.3 Thermal Design Power with respect to lithography.

One more thing we want to emphasize is the **stability of TDP in recent eras**, and the fact that it is converging. We will do the ANOVA to test and see if there is a significant difference in the tdp between the lithography era.

We will have the null hypothesis: H_0: the mean of the tdp in each type of lithography is the same. H_1: there exists a pair of lithography type so that their mean is different.

From the Data clarification, we can see that there are particularly few CPUs with lithography of 28 and 250, so we will remove them as well as all the row that has NA value

```r
# remove data with few count group ldate and remove NA
data <- data[data$litho !=28,]
data <- data[data$litho !=250,]

data <- data[!is.na(data$tdp), ]
data <- data[!is.na(data$litho), ]
```

We will create an ANOVA model.

```r
litho_anova_model<- aov(tdp ~ litho ,data = data)
```

Getting the result of the model.

```r
summary(litho_anova_model)
```

```
##               Df  Sum Sq Mean Sq F value              Pr(>F)
## litho          7  153527   21932   15.62 <0.0000000000000002 ***
## Residuals   1860 2612315    1404
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can see that as the p-value < 0.05, we can reject the null hypothesis H_0 and accept the alternative hypothesis H_1 that there exists a pair of litho type so that their TDP mean is difference.

To satisfy the requirements of One-way ANOVA, we should check its assumptions on Normality and Homoscedasticity (homogeneous variance).

```r
qqPlot(residuals(litho_anova_model))
shapiro.test(residuals(litho_anova_model))
leveneTest(tdp ~ litho ,data = data)
```

- We make a Q-Q Plot to explore its residuals.
- We perform a Shapiro-Wilk test of Normality:

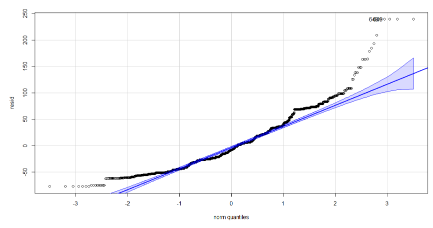
  Null hypothesis H_0 : The residuals of the ANOVA model are normally distributed.

  Alternative hypothesis H_1 : The residuals of the ANOVA model is not normally distributed.

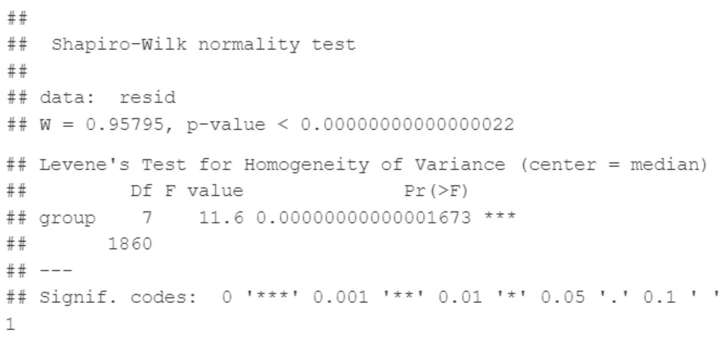
- We perform Levene's test to test for the equality of variances.

  Null hypothesis H_0 : The variance of `TDP` between categorical `litho` is equal.

  Alternative hypothesis H_1 : The variance between them is not equal.



(a) *Q-Q Plot of the ANOVA model*



```
##
##  Shapiro-Wilk normality test
##
## data:  resid
## W = 0.95795, p-value < 0.00000000000000022
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value                Pr(>F)
## group    7    11.6 0.00000000000001673 ***
##       1860
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
1
```

(b) *Results of Shapiro-Wilk and Levene's Test*

Figure 9: *Summary of the tests.*

We can see that the normality and variance test both fail as their p-value is both less than 0.05 and the qqplot has a lot of point that deviate from the line, however, the ANOVA test are also resilient against the violation of the two assumptions [2]. But to make sure we will also be including the Kruskal - Wallis test as a non-parametric alternative to the ANOVA test.

- Kruskal-Wallis H-Test

  Null hypothesis H_0 : The `TDP` come from the same populations with same median.

  Alternative hypothesis H_1 : These `TDP` come from populations with median that are not equal.

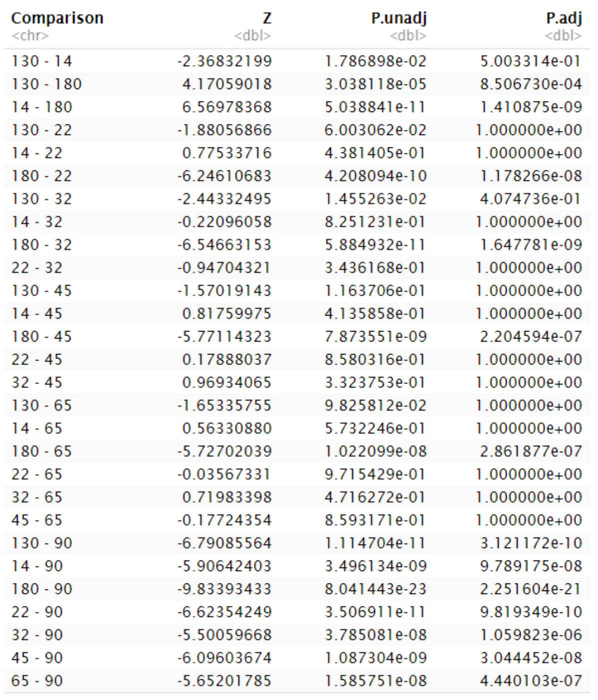
```
##  Kruskal-Wallis rank sum test
##
## data:  tdp by litho
## Kruskal-Wallis chi-squared = 106.78, df = 7, p-value <
## 0.00000000000000022
```

```
kruskal.test(tdp ~ litho, data = data)
```

The p-value of the Kruskal - Wallis test is also less than 0.05, which shows that there also exist at least a pair of `litho` type so that their `tdp` median is different.

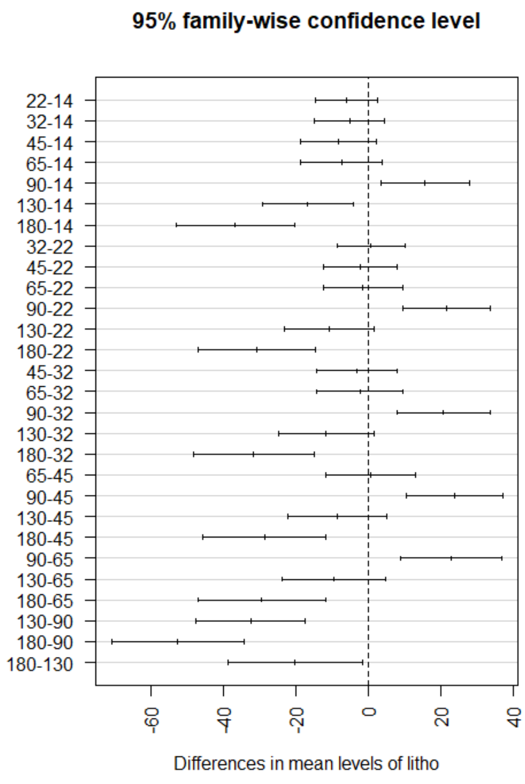
Finally we will analyse the result with a post hoc test to see which mean are different from each other. For this, we will use the TUKEY HSD test for ANOVA and Dunn test for Kruskal Wallis.

```
Tukey <- TukeyHSD(litho_anova_model)
plot(Tukey,las = 2)
dunnTest(tdp ~ litho, data = data,method = "bonferroni")
```



Comparison <chr>	Z <dbl>	P.unadj <dbl>	P.adj <dbl>
130 - 14	-2.36832199	1.786898e-02	5.003314e-01
130 - 180	4.17059018	3.038118e-05	8.506730e-04
14 - 180	6.56978368	5.038841e-11	1.410875e-09
130 - 22	-1.88056866	6.003062e-02	1.000000e+00
14 - 22	0.77533716	4.381405e-01	1.000000e+00
180 - 22	-6.24610683	4.208094e-10	1.178266e-08
130 - 32	-2.44332495	1.455263e-02	4.074736e-01
14 - 32	-0.22096058	8.251231e-01	1.000000e+00
180 - 32	-6.54663153	5.884932e-11	1.647781e-09
22 - 32	-0.94704321	3.436168e-01	1.000000e+00
130 - 45	-1.57019143	1.163706e-01	1.000000e+00
14 - 45	0.81759975	4.135858e-01	1.000000e+00
180 - 45	-5.77114323	7.873551e-09	2.204594e-07
22 - 45	0.17888037	8.580316e-01	1.000000e+00
32 - 45	0.96934065	3.323753e-01	1.000000e+00
130 - 65	-1.65335755	9.825812e-02	1.000000e+00
14 - 65	0.56330880	5.732246e-01	1.000000e+00
180 - 65	-5.72702039	1.022099e-08	2.861877e-07
22 - 65	-0.03567331	9.715429e-01	1.000000e+00
32 - 65	0.71983398	4.716272e-01	1.000000e+00
45 - 65	-0.17724354	8.593171e-01	1.000000e+00
130 - 90	-6.79085564	1.114704e-11	3.121172e-10
14 - 90	-5.90642403	3.496134e-09	9.789175e-08
180 - 90	-9.83393433	8.041443e-23	2.251604e-21
22 - 90	-6.62354249	3.506911e-11	9.819349e-10
32 - 90	-5.50059668	3.785081e-08	1.059823e-06
45 - 90	-6.09603674	1.087304e-09	3.044452e-08
65 - 90	-5.65201785	1.585751e-08	4.440103e-07

(a) *dunn test result*



(b) *TUKEY result plot*

Figure 10: *Summary of the post hoc tests.*

We can interpret the result of our data as follow:

1. There are a significant difference in `tdp` 's mean and median between the CPU of bigger `lithography`(180,130,90) and CPU of smaller `lithography`(14,22,32,45,65)
2. The CPU's `lithography` from 14,22,32,45,65 have their mean being relatively the same.
3. As `lithography` also reflect `release date`, as proven in 5.2.2 it also reflect the change in `tdp` of CPU 's design over time.

Overall the result of the ANOVA test and Kruskal Wallis test show the same result but as ANOVA has failed a couple of assumptions, the result might not be as reliable.

5.3 Regression analysis

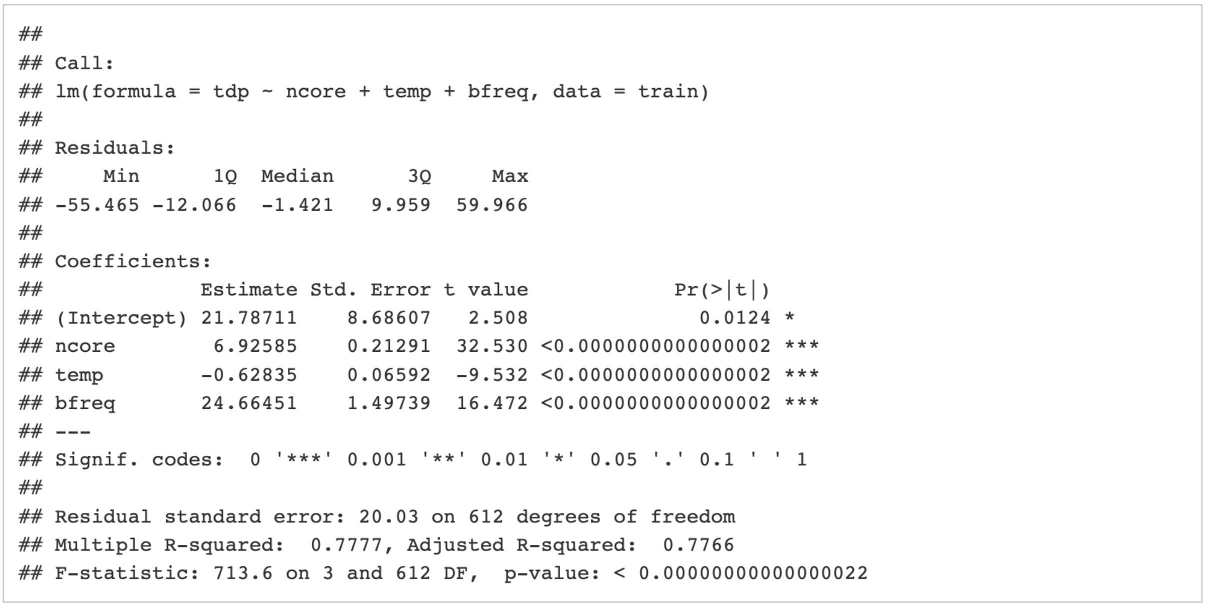
Base on checking the linearity of each column related to each other, in this section, we will conduct an investigation on the relationship between thermal design power with other features. There are many types of regression models, including simple linear regression, multiple linear regression, polynomial regression, logistic regression, and more. In this report, we will use certain model to test if there is any significant relationship between these figures.

5.3.1 Predicting Multi-Linear Model

Since the scatter plot of the feature **thermal design power** (tdp) with other variable maybe related to Linear regression. Now, first of all, we will build a linear regression model using `lm()`.

```r
# Build the model
model.lr <- lm(tdp ~ ncore + bfreq + temp, data = train)

# Summary of the model
summary(model.lr)
```

The `summary()` command gives us an overview over some statistics of our model.



```
##
## Call:
## lm(formula = tdp ~ ncore + temp + bfreq, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.465 -12.066  -1.421   9.959  59.966
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 21.78711    8.68607   2.508              0.0124 *
## ncore        6.92585    0.21291  32.530 <0.0000000000000002 ***
## temp        -0.62835    0.06592  -9.532 <0.0000000000000002 ***
## bfreq       24.66451    1.49739  16.472 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.03 on 612 degrees of freedom
## Multiple R-squared:  0.7777, Adjusted R-squared:  0.7766
## F-statistic: 713.6 on 3 and 612 DF,  p-value: < 0.00000000000000022
```

Figure 11: *Summary of linear model*

We can observe that all of the variables have the p-value less than 0.05, therefore; all predictors that we have chosen is involved in the building process.

Now we must check for Multiple-Linear model assumption:

- Residual Errors have a Mean Value of Zero.
- Residual Errors have Constant Variance.
- The errors are normally distributed.

Testing for Residual Errors have a Mean Value of Zero: We can check this assumption using the histogram of the residual:

```r
ggplot(model.lr, aes(x = resid(model.lr))) +
  geom_histogram(binwidth = 2, fill = "deepskyblue") # histogram of residuals
```

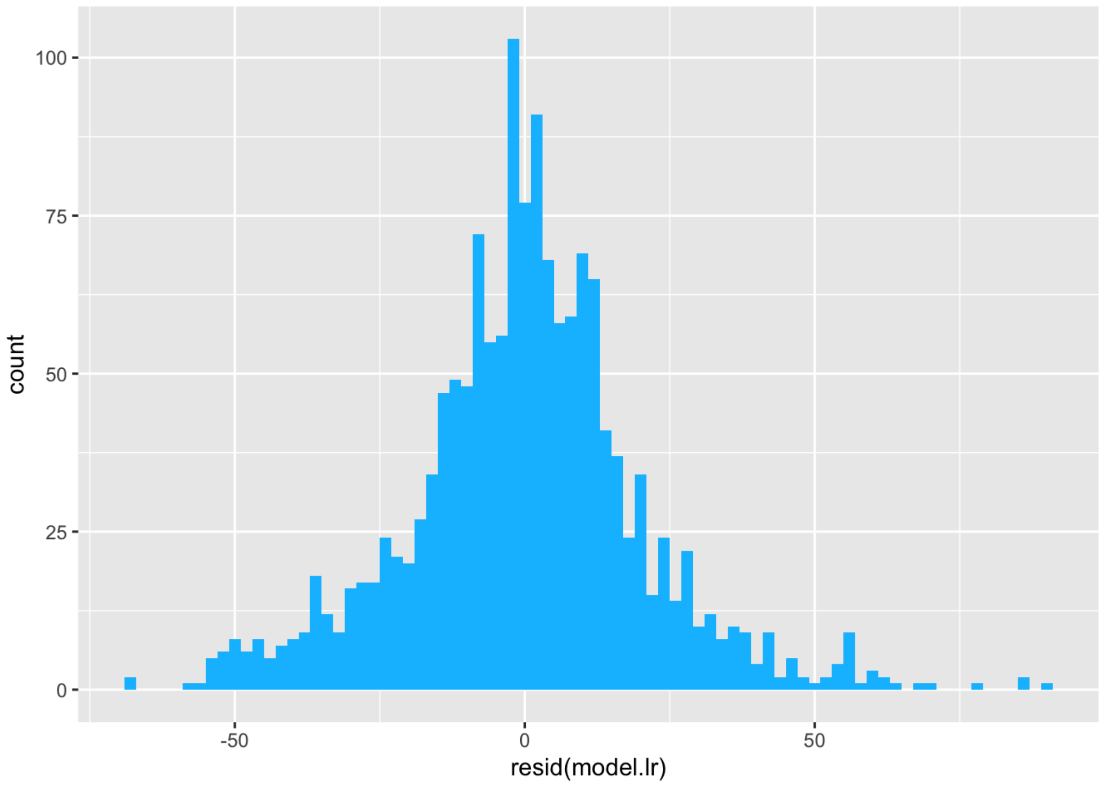


Figure 12: *Histogram of residuals*

We can observe that Residuals distribute mainly around the value 0, which is highly indicates that the residuals is scatter mostly around 0 and disperse equally two side of the graph, leading to the fact that the mean is approximately 0. Therefore, this assumption is met.

Testing for Residual Errors have Constant Variance: We can check this assumption using the Scale-Location plot. In this plot we can see the fitted values vs the square root of the standardized residuals. Ideally, we would want to see the residual points equally spread around the red line, which would indicate constant variance.

```
plot(model.lr, which = 3)
```

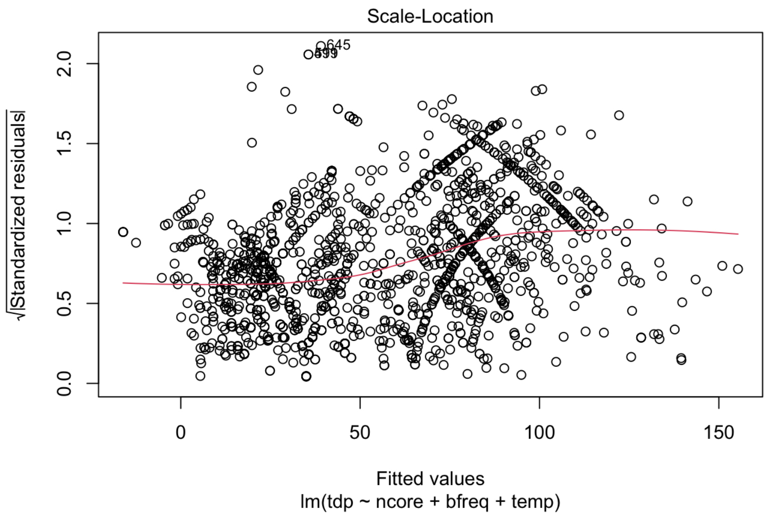


Figure 13: *Scale-Location plot*

In the above plot, we can see that the residual points are equally spread out in a weird pattern, or in other words the residuals scatter is not following any formal distribution and is random. Thus, this assumption is met.

Testing for normality of the the errors: To check this, we have to use Q-Q plot for normality consideration. The output we expect that the residuals will mostly scatter close to the straight line to get normality hypothesis accepted.

```r
plot(model.lr, which = 2)
```

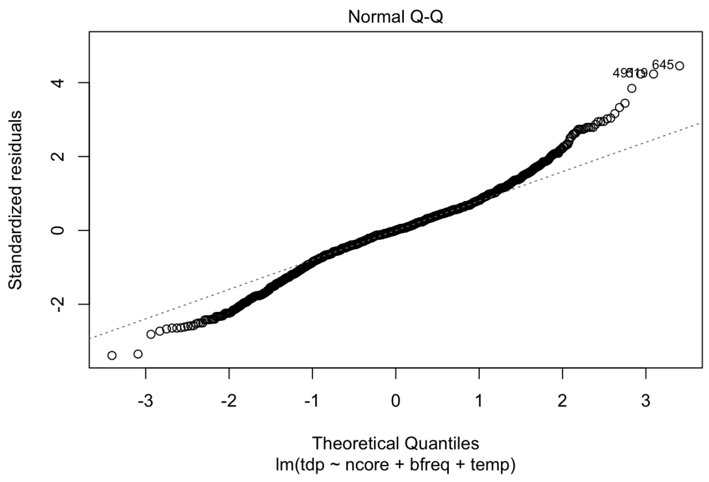


Figure 14: *Q-Q plot for residuals*

In the Q-Q plot, it can be seen that those points are lying near the line, while only a few tail points are not lying near the line. We can conclude that this model can be normally accurate, not 100% accurate. Perhaps another model will be more fitted for this data.

When we look at the summary table, the value of multiple R^2 is approximately 0.7777, it means that 77.77% of thermal design power variable is explained by the model and the rest 22.23% is defined by other factors, which is still accountable but not high. This model so far performing very well.

We will do the scatter plotting for the predicted value of test set compared with real values of the dataset using `plot()` function for more clear vision of this trend.

```r
# Create data frame for real tdp value and predicted tdp value (for Testing
the test set)
comtab.lr <- test['tdp']
comtab.lr['tdp_predicted'] <- as.data.frame(predict(model.lr, newdata =
test))

# Plotting
# The majority of points lie near the line, so its ok.
ggplot(comtab.lr, aes(x = tdp, y = tdp_predicted)) +
  geom_point(shape=1, color="blue") +
  geom_abline(mapping=aes(intercept= 0, slope = 1), color="darkblue") +
  labs(x = "TDP", y = "TDP Predicted")
```

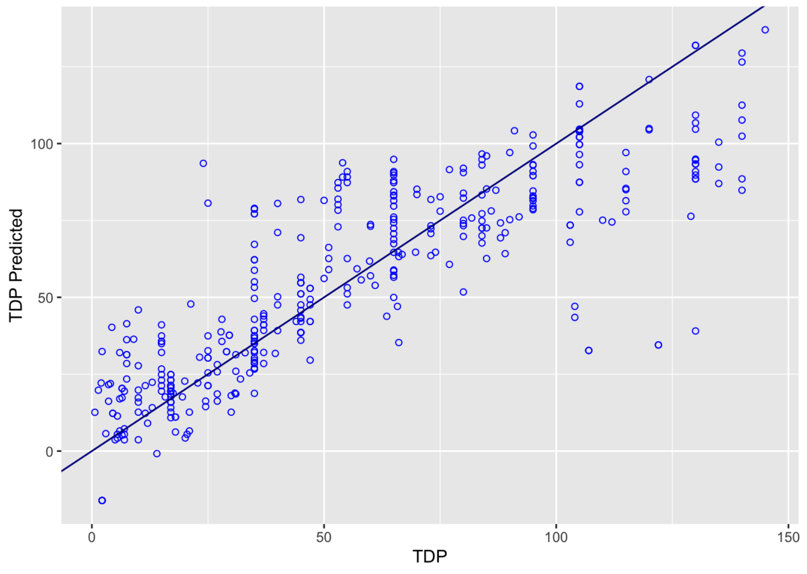


Figure 15: *Predict comparing*

At the end of this model, we will do the scatter plotting for the predicted value compared with the real value in test set. The plotted line in graph is $(d) : y = x$. The more concentration on this line the more correct the model does.

In conclusion, this model is an acceptable evaluation with the data set, which is shown by the above figures. Nevertheless, the initial assumption about data set's linearity is our subjective point of view, and that is the reason why we continue with the next model.

5.3.2 Random Forrest Regression Model

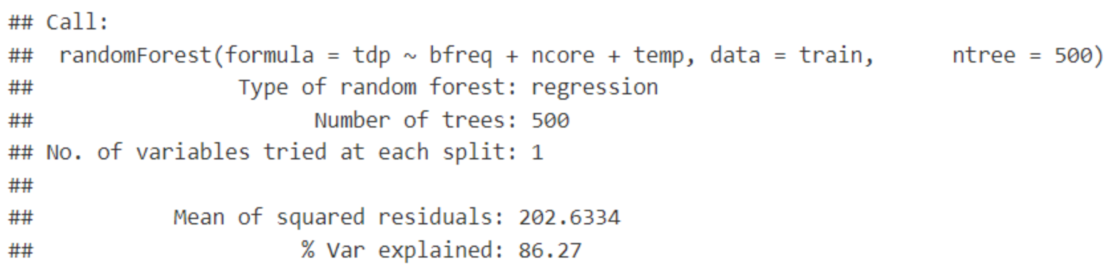
In this section, we will introduce *Random Forest Regression model.* In fact, Random Forest Regression models are often used for predicting system performance metrics such as CPU usage, response time, and throughput. The algorithm can handle both continuous and categorical data, making it a suitable choice for modeling CPU attribute data that may contain a mix of numerical and categorical variables.

With the second point of view, when we consider the correlation between TDP and 3 other ones: number of core, base frequency and temperature, there are many vertical lines created in the scatter-plot graphs. It can be understood as an independence between these variables, simultaneously, the non-linear relationships are also be observed clearly in these graphs.

First, we will build the Random Forest regression model:

```r
# Build model
model.rfr <- randomForest(formula = tdp ~ bfreq + ncore + temp, data = train,
ntree = 500)

# Print model's information
print(model.rfr)
```

The console the return the summary of the model



```
## Call:
##  randomForest(formula = tdp ~ bfreq + ncore + temp, data = train,      ntree = 500)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 202.6334
##                     % Var explained: 86.27
```

Figure 16: *Random Forest Regression*

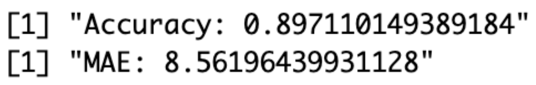
In the output above, the "% Var explained" (percentage of variance explained) is a measure of the amount of variation in the target variable that is explained by the random forest regression model. Specifically, it represents the percentage of the total variance in the target variable that is accounted for by the model, and *a higher percentage of variance explained is considered better*, as it indicates that the model is able to explain a larger proportion of the variation in the target variable.

An % variance explained of 89.55% is generally considered to be a good result for a random forest regression model, as it suggests that the model is able to explain a significant amount of the variation in the target variable.

Another method to test the fitness of this model is checking the **Mean Absolute Error** (MAE) of this model. The lower the MAE, the better this model validates our hypothesis. To check MAE, using the following commands:

```r
# Create data frame for real tdp value and predicted tdp value
comtab.rfr <- test['tdp']
comtab.rfr['tdp_predicted'] <- as.data.frame(predict(model.rfr, newdata = test), row.names = NULL)

# Evaluate model performance
accuracy <- sum(1-abs(comtab.rfr$tdp_predicted - comtab.rfr$tdp) / comtab.rfr$tdp) / nrow(comtab.rfr)
MAE <- sum(abs(comtab.rfr$tdp_predicted - comtab.rfr$tdp)) / nrow(comtab.rfr)
print(paste("Accuracy:", accuracy))
print(paste("MAE:", MAE))
```

This yields the output in the console



```
[1] "Accuracy: 0.897110149389184"
[1] "MAE: 8.56196439931128"
```

Figure 17: *Mean Absolute Error*

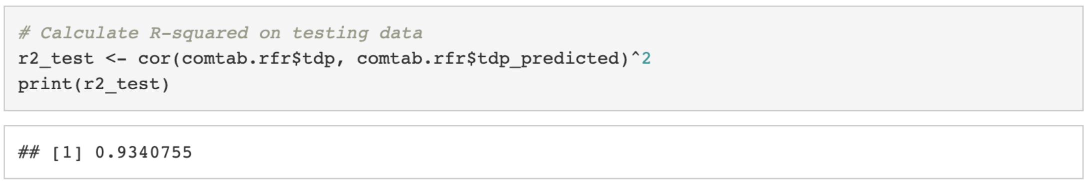
The MAE represents the average absolute difference between the predicted and actual values of the target variable, across all observations in the data set. It is calculated by taking the mean of the absolute values of the differences between the predicted and actual values taken from the **test set** we divided before.

The acceptable level of MAE depends on the specific problem and the context in which the model is being used. In general, a lower MAE is considered better, as it indicates that the model is making more accurate predictions. In this case, the error MAE value is 8.56196 which is relatively small compare to the range of the data, moreover, the accuracy of this model is approximately

89.711%, which is very high, indicating that this model is reliable. Similarly with the R-squared validation, to compute the R-squared value, we use the following commands:

```r
# calculate R-squared on testing data
r2_test <- cor(comtab.rfr$tdp, comtab.rfr$tdp_predicted)^2
print(r2_test)
```

The output in the console is



```r
# Calculate R-squared on testing data
r2_test <- cor(comtab.rfr$tdp, comtab.rfr$tdp_predicted)^2
print(r2_test)
```

```
## [1] 0.9340755
```

Figure 18: *R-squared Value*

Similarly in the previous model, the closer the value R-squared to 1, the more fitted the model is, which in this case is 0.9340755 very close to 1 so this strengthen the alternative hypothesis that there is a significant relationship between these features.

Because the assumption of this Random Forest model requires that the residual must follow normal distribution, we will use Q-Q plot to test if the residuals is normally distributed:

```r
# Calculate the residuals by subtracting the actual values from the predicted values
residuals <- comtab.rfr$tdp - comtab.rfr$tdp_predicted

# Create a normal probability plot of the residuals
qqnorm(residuals)
qqline(residuals)
```

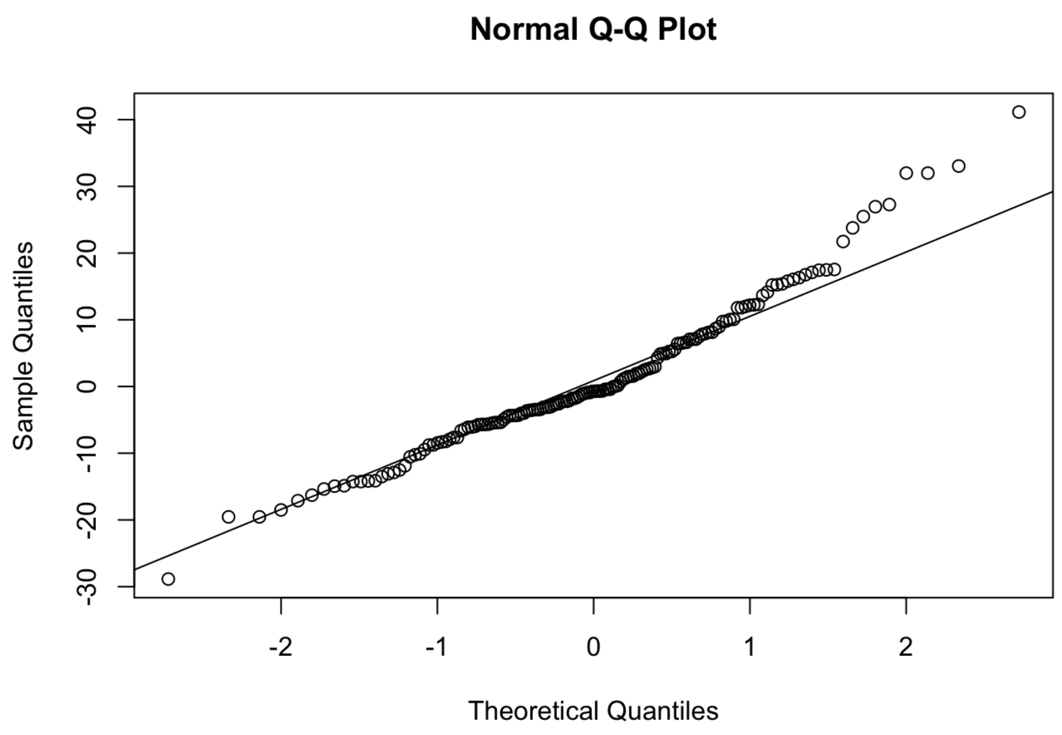


Figure 19: *Theoretical quantiles Plot*

In the Q-Q plot, it can be seen that those points are not fully lying near the line. We can conclude that this model may not be normally accurate.

We will do the scatter plotting for the predicted value of test set compared with real values of the **test** data set using `plot()` function:

```
# Plot the predicted - actual
ggplot(comtab.rfr, aes(x = tdp, y = tdp_predicted)) +
  geom_point(shape = 1, color = "blue") +
  geom_abline(mapping = aes(intercept = 0, slope = 1), color = "darkblue") +
  labs(x = "TDP", y = "TDP Predicted")
```

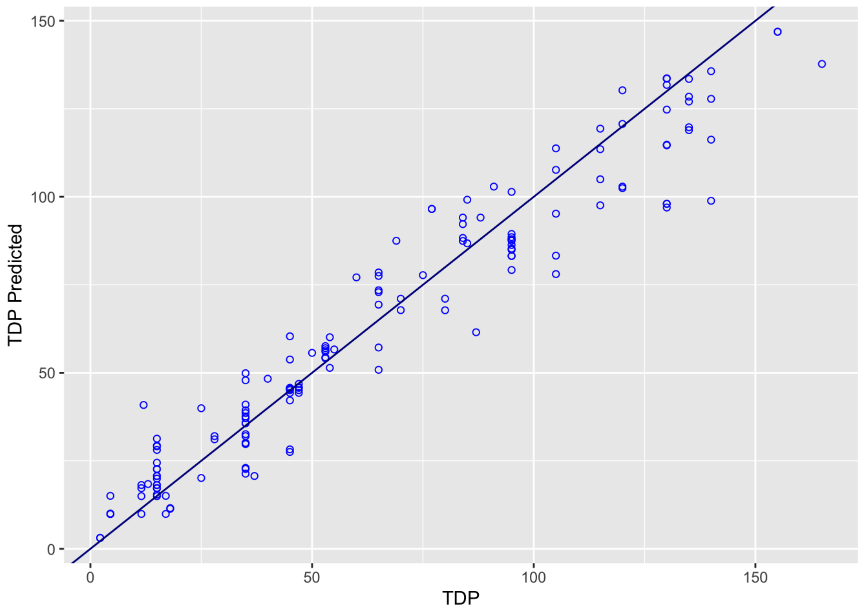


Figure 20: *Random Forest model prediction*

As we can see, the closer the points to the line, the more accurate the prediction, and we clearly see, similarly to the previous model, this is somewhat like 80% normally distributed in residuals, which is explained most of the variables involved.

In conclusion, compared to Linear regression model, the current model - Random forest give us the more optimistic results which are proven in the above figures such as error, accuracy and R^2 value. As a result, we conclude this model is fitter to our data and get the quite high results, not 100% accuracy. Perhaps, more data should be obtained or some indexes such as number of tree be modified to reach the higher accuracy.

5.4 Classification in Market of CPU using Logistic Regression Model

Logistic Regression is a type of statistical analysis that is used to model and predict the probability of a binary outcome based on one or more predictor variables. It is a type of supervised learning algorithm that is commonly used in machine learning and data analysis.

The logistic regression model estimates the probability of the dependent variable taking a certain value based on the values of the independent variables. The output of the logistic regression model is a logistic function, which maps a continuous range of inputs to a range of probabilities between 0 and 1.

Logistic regression is commonly used for classification tasks, where the goal is to predict the probability of an event occurring. In our project, we want to study that if the CPU chips manufactured for market **Desktop** or **Server** is different from that of the rest of the market or not. Now, we will label each with corresponding type 1 or 0, 1 for market **Desktop** or **Server** and 0 for the rest:

```r
# Add Type column
# Server or Desktop will be in type 1, otherwise type 0
train$type <- ifelse(train$market == 'Server' | train$market == 'Desktop', 1,
0)
test$type <- ifelse(test$market == 'Server' | test$market == 'Desktop', 1, 0)
```

Then, we will build the logistic model for train dataset and make prediction using test dataset:

```r
# Model
model.l <- glm(type ~ tdp, data = train, family = 'binomial')

# Make predictions on the test data
predicted_probs <- predict(model.l, newdata = test, type = "response")
predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)
```

Now, different from other two previous models, evaluate the performance of a binary classification model, such as a logistic regression model, we use confusion matrix. The confusion matrix provides a summary of the number of correct and incorrect predictions made by the model, and is constructed by comparing the predicted values of the model with the actual values of the dependent variable.

```r
# Evaluate model performance
# Convert the predicted classes and actual classes to factors with the same
levels and labels
predicted_classes_factor <- factor(predicted_classes, levels = c(0, 1),
labels = c("Negative", "Positive"))
actual_classes_factor <- factor(test$type, levels = c(0, 1), labels =
c("Negative", "Positive"))

# Create a confusion matrix
conf_matrix <- confusionMatrix(predicted_classes_factor,
actual_classes_factor)
```

The confusion matrix consists of four entries:

- True positives (tp): The number of cases where the model predicted a positive outcome and the actual outcome was positive.

- False positives (fp): The number of cases where the model predicted a positive outcome but the actual outcome was negative.
- False negatives (fn): The number of cases where the model predicted a negative outcome but the actual outcome was positive.
- True negatives (tn): The number of cases where the model predicted a negative outcome and the actual outcome was negative.

To use a confusion matrix to evaluate the performance of a logistic regression model, the first step is to use the model to make predictions on a test dataset that was not used to train the model. Then, the predicted values of the model are compared to the actual values of the dependent variable, and a confusion matrix is constructed.

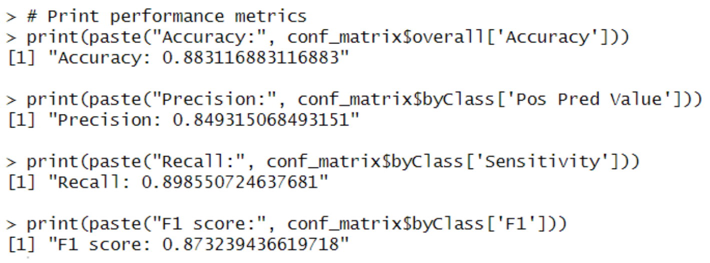
```r
# Print the confusion matrix
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Negative Positive
##   Negative       68        2
##   Positive        1       83
##
##                Accuracy : 0.9805
##                  95% CI : (0.9441, 0.996)
##     No Information Rate : 0.5519
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.9607
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9855
##             Specificity : 0.9765
##          Pos Pred Value : 0.9714
##          Neg Pred Value : 0.9881
##              Prevalence : 0.4481
##          Detection Rate : 0.4416
##    Detection Prevalence : 0.4545
##       Balanced Accuracy : 0.9810
##
##        'Positive' Class : Negative
##
```

Figure 21: *Confusion Matrix*

Once the confusion matrix is constructed, several performance metrics can be calculated to evaluate the model, including:

- Accuracy: The proportion of cases that were correctly classified by the model, calculated as $\dfrac{tp+tn}{tp+fp+fn+tn}$.
- Precision: The proportion of cases that were predicted to be positive by the model that were actually positive, calculated as $\dfrac{tp}{tp+fp}$.
- Recall: The proportion of actual positive cases that were correctly identified by the model, calculated as $\dfrac{tp}{tp+fn}$.
- F1 score: A weighted average of precision and recall, calculated as $2 \times \dfrac{precision \times recall}{precision + recall}$.



```
> # Print performance metrics
> print(paste("Accuracy:", conf_matrix$overall['Accuracy']))
[1] "Accuracy: 0.883116883116883"

> print(paste("Precision:", conf_matrix$byClass['Pos Pred Value']))
[1] "Precision: 0.849315068493151"

> print(paste("Recall:", conf_matrix$byClass['Sensitivity']))
[1] "Recall: 0.898550724637681"

> print(paste("F1 score:", conf_matrix$byClass['F1']))
[1] "F1 score: 0.873239436619718"
```

Figure 22: *Confusion Matrix attributes*

It is clear from the provided performance metrics that the logistic regression model is exhibiting exceptional performance.

With an accuracy of 0.8831, the model is demonstrating an impressive ability to correctly predict the outcome in 88.31% of the cases. Additionally, the precision of the model is 0.8493, indicating that out of all the cases predicted as positive, approximately 84.93% of them were actually positive. Moreover, the recall of the model is 0.8986, which means that out of all the actual positive cases, the model correctly identified 89.86% of them as positive. Finally, the F1 score of the model is 0.8732, which is a harmonic mean of precision and recall.

Overall, these impressive performance metrics suggest that the logistic regression model concludes the differences between market Desktop or Server and the rest of market. Therefore, it can be inferred that the market is taking a good account of **thermal design power**, as the model's high accuracy, precision, recall, and F1 score indicate its ability to make reliable and accurate predictions.

6 Conclusion

In general, we have conducted several test models and received many positive outcomes that provide strong proof to reject the null hypothesis that there is no significant relationship between these categories, therefore confirming our prediction that the thermal power design of each CPU strongly depends on the number of cores and base frequency, as well as the temperature that can occur when entering hard mode of task of the devices it is attached on, leaving the market chip producer a clearer view to balance out between the power consumption and the performance of the chips to better their benefits in the technology field.

With the topic **Researching on the Thermal Design Power of CPU** uses the R programming language to process statistical data and realize the model linear regression model, our team had a more intuitive view of how to extract data, process and analyze raw data, turning them into valuable data sources long-term, or better yet, being able to generalize the general situation and make predictions about the data set.

During the process of implementing the thesis will not be able to avoid some minor errors, as well as some of the predictions might not be exactly as reality, our group hopes that this thesis report will provide another aspect for the viewers about the issue of dealing with the Thermal Design Power problem in the real world.

References

[1] Alexis Dinno. Nonparametric pairwise multiple comparisons in independent groups using dunn's test. *The Stata Journal*, (1):292–300, 2015.

[2] George C. Runger Douglas C. Montgomery. *Applied Statistics and Probability for Engineers*. John Wiley, Sons, Inc, 2003.

[3] Foster et al. *An Introduction to Psychological Statistics*. LibreTexts, 2022.

[4] Mario F.Triola. *Elementary Statistics*. 2018.

[5] Mark Lapedus. Breaking the 2nm barrier, 2021. URL https://semiengineering.com/breaking-the-2nm-barrier/. Last accessed 24 April 2023.

[6] Michael Lesk and Brian Kernighan. Computer typesetting of technical journals on UNIX. In *Proceedings of American Federation of Information Processing Societies: 1977 National Computer Conference*, pages 879–888, Dallas, Texas, 1977.

[7] Mary Natrella. *NIST/SEMATECH e-Handbook of Statistical Methods*. NIST, 2012. URL http://www.itl.nist.gov/div898/handbook/.