VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

PROBABILITY AND STATISTICS

THE EVOLUTION OF COMPUTER PROCESSORS: A STATISTICS OF COMMON PROPERTIES

Instructor: Dr. Nguyen Thi Mong Ngoc
Students: Chau Dang Minh - 2013748

Ho Chi Minh City, May 2024

Evaluation

N.O.	Student	ID	Works	Completed
1	Chau Dang Minh	2212287	Dataset overview Preprocessing	100%
2	Ha Khoi Nguyen	2212287	Descriptive statistics	100%
3	Nguyen Thi Mai Anh	2210103	Theories Slides and Presentation	
4	Võ Ninh Giang	2210834	Inferential statistics	
5	Trinh Viet Cuong	2210447	Inferential statistics	

Introduction and Acknowledgement

Phenomena that are meaningful to humans appear not to be completely stochastic. In the same sense, datasets produced by humans, or nature in time circulations have insights to be analyzed, which is accounted by Statistics. Thanks to Dr. Nguyen Thi Mong Ngoc's supervision in Probability and Statistics course, we have a chance to study basic statistics within an assignment with a tiny dataset. We organized our report in the following structure

1. Overview of the dataset. In this chapter, we carefully describe in details as much as possible the dataset, specifically the properties of each instance. We also notice which features to be used for later statistical tasks.
2. Preprocessing. We process data cleaning and some computations.
3. Descriptive statistics. We calculate some qualitative features of the dataset.
4. Inferential statistics. Our problems are explicitly stated and solved.

Contents

Chapter I

Dataset Overview

As Computer Science students, we are assigned to analyze a dataset about computer processors, namely CPUs and GPUs. Our dataset is credited to Intel, Game-Debate, and the companies involved in producing the part. Information of CPUs and GPUs are collected separately into two files, namely `Intel_CPUs.csv` and `All_GPUs.csv`. Let us get familiar to some technical features, given in the metadata or acquired over the internet.

Table I.1: Some technical features of CPUs and GPUs

N.O.	**Feature name**	**Relevant to**	**Details**
1	Lithography	CPU, GPU	The semiconductor technology used to manufacture an integrated circuit, and is reported in nanometer
2	Number of Cores	CPU	A hardware term that describes the number of independent central processing units
3	Number of Threads	CPU	A Thread, or thread of execution, is a software term for the basic ordered sequence of instructions that can be passed
4	Base Frequency	CPU	Describes the rate at which the processor's transistors open and close.
5	Cache	CPU	An area of fast memory located on the processor.
6	Thermal Design Power	CPU	Represents the average power, in watts, the processor dissipates when operating at Base Frequency with all cores active under an Intel-defined, high-complexity workload.
7	Embedded Availability	CPU	In essence, an embedded processor is a CPU chip used in a system which is not a general-purpose workstation, laptop or desktop computer.
8	Embedded Availability	CPU	In essence, an embedded processor is a CPU chip used in a system which is not a general-purpose workstation, laptop or desktop computer.
9	Memory Types	CPU	Single Channel, Dual Channel, Triple Channel, and Flex Mode.

Continued on next page

Table I.1: Some technical features of CPUs and GPUs (Continued)

10	Instruction Set	CPU	
11	Maximal Temperature	CPU	
12	Architecture	GPU	
13	Dedicated and Integrated	GPU	Whether the GPU is solely used or shares memory with a CPU
14	(Front-side) Bus Speed	CPU, GPU	The speed at which data is transferred between the processors and other components such as the memory, chipset, and peripherals.
15	No-Execute Bit	CPU	Hardware-based security feature that can reduce exposure to viruses and malicious-code attacks.
16	Thermal Monitoring Technologies	CPU	Protects the processor package and the system from thermal failure through several thermal management features.

Chapter II

Foundations

A statistical Hypothesis is a statement or assumption about the parameters of one or more populations, which may be true or false. Each such truth value can be expressed in the form of a hypothesis, usually denoted as H_0, called the null hypothesis, and an alternative hypothesis, usually denoted as H_1 or Halt, which is its complement.

1 Theory of Sampling

1.1 Terminology

1. A population is a set $\{x_i\}_{i=1}^N$ of elements belonging to the object under study, which need to be observed, collected, and analyzed based on certain characteristics. A parameter is a numerical description of a population.
2. A sample is a subset $\{x_{i_j}\}_{j=1}^n$ of units selected from the population according to a certain sampling method. A statistic is a numerical description of a sample.
3. Let the population belong to a distribution represented by a random variable X. A random sample is a collection $\{X_i\}_{i=1}^n$ such that $X, X_1, \ldots, X_n$ are independent and identically distributed.

1.2 Statistical characteristics

Statistical characteristics are important properties on a population, a sample or a random sample directly related to the content of the study and survey, which need to be collected from the units of the population, generally divided into two groups

1. Central tendency includes mean, median, mode, giving us an idea of the trend of the data points clustering around certain values.
2. Dispersion includes variance, standard deviation, which describe the spread or variability of the data points around the central tendency.

In details, these quantities are formulated as followings.

1. Population mean
$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{2.1.1}$$
2. Sample mean
$$\bar{x} = \frac{1}{n}\sum_{j=1}^{n} x_{i_j}. \tag{2.1.2}$$

3. Random sample mean
$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i. \tag{2.1.3}$$

4. Population variance
$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})^2. \tag{2.1.4}$$

5. Sample variance
$$s^2 = \frac{1}{n-1}\sum_{j=1}^{n}(x_{i_j} - \bar{x})^2. \tag{2.1.5}$$

6. Random sample variance
$$S^2 = \frac{1}{N-1}\sum_{n=1}^{n}(X_i - \bar{X})^2. \tag{2.1.6}$$

7. Population, sample and random sample derivations are the square roots σ, s and S, respectively.

8. Sample ratio
$$f = \frac{n}{N} \tag{2.1.7}$$

9. Coefficient of variation
$$\mathrm{CV} = \left(\frac{s}{\bar{x}}\right) \cdot 100\% \tag{2.1.8}$$

10. Suppose that the sample $\{x_{i_j}\}_{j=1}^{n}$ is in increasing order. The sample median is calculated as
$$\mathrm{med} = \begin{cases} x_{i_{k+1}}, & \text{if } n = 2k+1 \\ \dfrac{1}{2}(x_{i_k} + x_{i_{k+1}}), & \text{if } n = 2k. \end{cases} \tag{2.1.9}$$

11. Quartiles include three values Q_1, Q_2 and Q_3. Where Q_2 is the median, Q_1 and Q_3 is the median of the set of samples less than and more than Q_2, respectively. The quantity $\mathrm{IQR} = Q_3 - Q_1$ is called the interquartile range.

12. An outlier is a point outsize $(Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR)$.

2 Statistical Hypothesis Testing

The process of testing a statistical hypothesis is a standardized procedure for making decisions to reject or not reject a hypothesis based on sample data. This process is called hypothesis testing and typically involves four steps:

1. **Setting Hypotheses** involves stating which hypothesis is null and which is alternative. The hypotheses are formulated in a mutually exclusive manner, meaning if one is true, the other must be false.

2. **Constructing an Analysis Plan** outlines how sample data will be used to evaluate the null hypothesis. Evaluation typically centers around a single test statistic.

3. **Analyzing Sample Data** determines the values of the sample statistic (mean, proportion, t-statistic, z-score, etc.) as described in the analysis plan.

4. **Interpreting Results** applies decision rules outlined in the analysis plan. If the observed result is inconsistent with the null hypothesis, reject it.

Similar to estimation, statistical hypothesis testing does not provide results with 100% certainty but rather with a certain level of confidence, and errors can occur. Errors can be classified into two types:

- **Type I Error:** Rejecting the null hypothesis H_0 and accepting the alternative hypothesis H_1 whereas H_0 is true.
- **Type II Error:** Failing to reject the null hypothesis H_0 and not accepting the alternative hypothesis H_1 whereas H_1 is true.

Both types of errors can have adverse consequences. Depending on the situation, it's assessed which type of error leads to more serious consequences and should be minimized.

To make decisions about rejecting the null hypothesis, statisticians rely on specific rules. These rules are listed in the analysis plan. Traditionally, statisticians describe these decision rules in two ways: referencing a p-value or referencing a region of acceptance.

- **p-value:** The strength of evidence in favor of a null hypothesis is measured by the p-value. Suppose the test statistic is denoted by S. The p-value is the probability of observing a test statistic as extreme as S, assuming the null hypothesis is true. If the p-value is smaller than the significance level, we reject the null hypothesis.
- **Region of Acceptance:** The region of acceptance is a range of values. If the sample statistic falls within the region of acceptance, the null hypothesis is not rejected. The region of acceptance is set so that the probability of committing a Type I error is equivalent to the significance level.

Values outside the region of acceptance are called the region of rejection. If the sample statistic falls within the region of rejection, the null hypothesis is rejected. In such cases, it's said that the null hypothesis has been rejected at the significance level.

3 Linear Regression

Linear regression constructs a linear model between independent variables $\mathbf{x} = (0, x_1, \dots, x_d) \in \mathbb{R}^{d+1}$ and a dependent variable $y \in \mathbb{R}$. Traditionally, the relation is expressed as

$$\begin{aligned} y &= w_0 + \sum_{i=1}^{n} w_i x_i \\ &= \mathbf{x}^\top \mathbf{w}, \end{aligned} \tag{2.3.1}$$

where $\{w_i\}_{i=0}^d$ are parameters to learn. Let $\{(\mathbf{x}_n, \hat{y}_n)\}_{n=1}^N$ be collected observations. Our objective is to minimize the error

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= \sum_{n=1}^{N} (y - \hat{y})^2 \\ &= \sum_{n=1}^{N} (\mathbf{w}^\top \mathbf{x}_n - \hat{y}_n)^2 \\ &= \|\mathbf{X}\mathbf{w} - \hat{\mathbf{y}}\|^2, \end{aligned} \tag{2.3.2}$$

where $\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_n)^\top \in \mathbb{R}^n$ and $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n) \in \mathbb{R}^{n \times (d+1)}$. Taking the derivative of $\mathcal{L}$, we have

$$\frac{\mathrm{d}\mathcal{L}(\mathbf{w})}{\mathrm{d}\mathbf{w}} = 2\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \hat{\mathbf{y}}).$$

Equivalently, $\mathbf{X}^\top\mathbf{X}\mathbf{w} = \mathbf{X}^\top\hat{\mathbf{y}}$. If $\mathbf{X}^\top\mathbf{X}$ is invertible i.e. the independent variables are linearly independent, we have the unique solution

$$\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\hat{\mathbf{y}}. \tag{2.3.3}$$

Along with the independence of independent variables, some requirements need to be satisfies for an accurate and reliable solution.

1. Linearity: The independent and dependent variables correlate. This means that the change in one or more independent variables induces the change in the dependent variable.
2. Independence: The observed data in the dataset are independent of each other. This means that the value of the dependent variable for one observed data point does not depend on the value of the dependent variable for another observed data point.
3. Across all levels of the independent variables, the variance of the errors remains constant. This indicates that the number of independent variables does not affect the variability of the errors.
4. Normality: The errors in the model follow a normal distribution.

4 Analysis of Variance

Analysis of Variance (ANOVA) is a statistical analysis tool used to partition the total observed variability found in a dataset into two parts: Systematic factors and Random factors. Systematic factors statistically influence a certain dataset, while random factors do not. We use ANOVA to determine the influence of independent variables on the dependent variable in regression studies.

The T-Test and Z-Test methods, developed in the 20th century, were used for statistical analysis until 1918 when Ronald Fisher introduced the Analysis of Variance method. Both serve as hypothesis tests to assess whether there is any significant difference between means. The T-Test is used when the population variance is unknown or the sample size is small ($n < 30$). Meanwhile, the Z-Test is applied when the population variance is known and the sample size is large ($n > 30$). The T-Test uses the Student's t-distribution, while the Z-Test uses the standard normal distribution. As the sample size increases, the T-distribution converges to the standard normal distribution. The T-Test can be understood as a statistical test used to compare and analyze whether the mean values of two populations are different when the standard deviation is unknown. Conversely, the Z-Test is a parametric test applied when the standard deviation is known to determine whether the means of two datasets are different. ANOVA is also known as Fisher's analysis of variance and is an extension of both T-Tests and Z-Tests.

4.1 ANOVA Usecases

ANOVA test is the initial step in analyzing factors influencing a specific dataset. Upon completing a test, additional checks on measurable methodological factors contributing to dataset inconsistency are conducted. The results, using F-distribution, can be utilized to generate supplementary data suitable for proposed regression models.

ANOVA test enables comparison of more than two groups simultaneously to determine whether a relationship exists among them. The outcome of ANOVA analysis, the F-statistic (also known as the F-ratio), allows for analysis across multiple datasets to ascertain variation between samples and within samples.

If no significant differences exist among the tested groups (known as the null hypothesis), the F-statistic result of ANOVA will be close to 1. The distribution of all possible values of the F-statistic follows the F-distribution. This is a family of probability distributions characterized by two parameters, known as the degrees of freedom of the numerator and the degrees of freedom of the denominator.

4.2 Types of ANOVA

There are two main types of ANOVA: one-way ANOVA and two-way ANOVA. Additionally, there are variations of ANOVA such as Multivariate ANOVA (MANOVA), which differs from ANOVA in that the former tests multiple dependent variables simultaneously while the latter evaluates only one dependent variable at a time. One-way or two-way refers to the number of independent variables in the analysis of variance. Furthermore, ANOVA relies on assumptions. ANOVA tests assume that the data are normally distributed, the variance is nearly equal across groups, and all observations are independent. If these assumptions are not met, ANOVA may not be useful for group comparisons.

4.2.1 One-way ANOVA

One-way ANOVA is used to assess the effect of a single factor on a single response variable. It determines whether all samples are similar i.e. determine whether there is a statistically significant difference between the means of three or more independent (unrelated) groups. One-way ANOVA is formulated as

$$Y_{i,j} = \mu_j + \varepsilon_{i,j}, \tag{2.4.1}$$

where $i = 1, \ldots, n_j$ are indices of observations within the same group, $j = 1, \ldots, k$ are indices of influenced groups, μ_j s the mean for each affected group and $\varepsilon_{i,j} \sim N(0, \sigma^2)$ is normal-distributed errors, independent for all i and j. Mean and variance of $Y_{i,j}$ are $E(Y_{i,j}) = \mu_j$ and $Var(Y_{i,j}) = \sigma^2$, respectively.

4.2.2 Two-way ANOVA

Two-way ANOVA is an extension of one-way ANOVA, where there are two independent factors. It is used to observe the interaction between two factors and test the effects of two factors simultaneously. It is formulated as

$$Y_{i,j} = \mu_j + \gamma_j + (\mu\gamma)i, j + \varepsilon i, j, k. \tag{2.4.2}$$

Assumptions about random variables are similar to one-way ANOVA.

Chapter III

Data Preprocessing

1 Data Cleaning

With `RStudio`, the working directory is automatically determined. Otherwise, it can be indicated by `here` library.

Listing 3.1.1: Required libraries and working directory setup

```
# Libraries and options
library(dplyr)
library(here)
library(knitr)
library(kableExtra)

# Self-defined functions
source("utils.R")

# Working directory
setwd(here())
```

Now our working directory have been explicated, we can use relative paths to read the data. With `RMarkdown`, we can prettify the rendering.

Listing 3.1.2: RStudio data object initialization

```
  # Read the CSV file into a data frame
  cpu_data <- read.csv("dataset/Intel_CPUs.csv")
  gpu_data <- read.csv("dataset/All_GPUs.csv")

  # Inspect the CPU data
  kable(head(cpu_data), format = "html") %>%
    kable_styling()
```

Invalid cells may contain `NA`, an empty string, or other values showing us that this cell's data was not collecting correctly. At the very first step, we want to select only columns such that the percentage of valid cells exceeds our predefined value. Then we filter out all instances with invalid features. Note that careful column selection possibly remains more instances for later tasks. For CPU data, we concern about the price, so let us choose a quota for which the column of prices is maintained.

Listing 3.1.3: Cleaning functions

```
# Check if a cell has a valid value
is_valid <- function(value) {
  return(!is.na(value)
```

Product_Collection	Vertical_Segment	Processor_Number	Status	Launch_Date	Lithography	Recommended_Customer_Price	nb_of_Cores	nb_of_Threads	Pr
7th Generation Intel® Core™ i7 Processors	Mobile	i7-7Y75	Launched	Q3'16	14 nm	$393.00	2	4	1.3
8th Generation Intel® Core™ i5 Processors	Mobile	i5-8250U	Launched	Q3'17	14 nm	$297.00	4	8	1.6
8th Generation Intel® Core™ i7 Processors	Mobile	i7-8550U	Launched	Q3'17	14 nm	$409.00	4	8	1.8
Intel® Core™ X-series Processors	Desktop	i7-3820	End of Life	Q1'12	32 nm	$305.00	4	8	3.6
7th Generation Intel® Core™ i5 Processors	Mobile	i5-7Y57	Launched	Q1'17	14 nm	$281.00	2	4	1.2
Intel® Celeron® Processor 3000 Series	Mobile	3205U	Launched	Q1'15	14 nm	$107.00	2	2	1.5

Figure 3.1.1: First instances of CPUs data

```
          & !is.null(value)
          & !value == ""
          & !value == "N/A"
          & !trimws(value) == "-"
          & !value == "missing"
          & !value == "unknown")
    # Add your criteria
  }


  # Select columns with enough valid cells
  filtered_data <- function(data, valid_percentage=0.5) {
  selected_columns <- character(0)

  for (col in colnames(data)) {
    valid_count <- sum(is_valid(data[[col]]))
    total_instances <- length(data[[col]])

    if ((valid_count / total_instances) >= fill) {
      selected_columns <- c(selected_columns, col)
    }
  }

  return(data[selected_columns])
}
```

Listing 3.1.4: Cleaned CPU data

```
filtered_cpu_data <- filtered_data(cpu_data, valid_percentage=0.5)

processed_cpu_data <-
  filtered_cpu_data[
    apply(filtered_cpu_data, 1, function(row) all(sapply(row, is_valid))), ]

selected_cpu_data <- processed_cpu_data[] # Adjust selected columns for your later needs
selected_cpu_data <- unique(selected_cpu_data)
kable(head(selected_cpu_data), format = "html") %>%
  kable_styling()
```

Listing 3.1.5: Cleaned GPU data

```
filtered_gpu_data <- filtered_data(gpu_data, valid_percentage=0.5)

processed_gpu_data <-
  filtered_gpu_data[
    apply(filtered_gpu_data, 1, function(row) all(sapply(row, is_valid))), ]

selected_gpu_data <- processed_gpu_data[] # Adjust selected columns for your later needs
selected_gpu_data <- unique(selected_gpu_data)
kable(head(selected_gpu_data), format = "html") %>%
  kable_styling()
```

	Product_Collection	Vertical_Segment	Processor_Number	Status	Launch_Date	Lithography	Recommended_Customer_Price	nb_of_Cores	nb_of_Threads
1	7th Generation Intel® Core™ i7 Processors	Mobile	i7-7Y75	Launched	Q3'16	14 nm	$393.00	2	4
2	8th Generation Intel® Core™ i5 Processors	Mobile	i5-8250U	Launched	Q3'17	14 nm	$297.00	4	8
3	8th Generation Intel® Core™ i7 Processors	Mobile	i7-8550U	Launched	Q3'17	14 nm	$409.00	4	8
5	7th Generation Intel® Core™ i5 Processors	Mobile	i5-7Y57	Launched	Q1'17	14 nm	$281.00	2	4
11	Intel® Pentium® Processor 2000 Series	Mobile	2020M	Launched	Q3'12	22 nm	$134.00	2	2
14	Intel® Pentium® Processor 4000 Series	Mobile	4405U	Launched	Q3'15	14 nm	$161.00	2	4

Figure 3.1.2: First instances of selected CPUs data

It's worth finding the key of the data to know which feature uniquely determine an instance. We prioritize categorical features, specifically pure names.

Listing 3.1.6: Keys of the data

```
nrow(selected_cpu_data) - nrow(selected_cpu_data[, c("Product_Collection",
                                                     "Processor_Number")])
nrow(selected_gpu_data) - nrow(selected_gpu_data[, c("Name")])
# Outputs: 0
```

2 Data Pre-computation

Some features in our data have values that need to be reformatted for easily later sorting and analyses. Therefore, we need to gain a good understand on the features.

Listing 3.2.1: A processing for selected features

```
cpu_columns <- colnames(cpu_data)
gpu_columns <- colnames(gpu_data)
intersect(cpu_columns, gpu_columns)
# Output: character(0)
```

Since the data files have no common features, we took a look at them independently and decided to pre-compute some features.

	Architecture	Best_Resolution	Core_Speed	DVI_Connection	Dedicated	Direct_X	HDMI_Connection	Integrated	L2_Cache	Manufacturer	Max_Power
2	R600 XT	1366 x 768	-	2	Yes	DX 10	0	No	0KB	AMD	215 Watts
3	R600 PRO	1366 x 768	-	2	Yes	DX 10	0	No	0KB	AMD	200 Watts
5	RV630	1024 x 768	-	2	Yes	DX 10	0	No	0KB	AMD	45 Watts
6	RV630	1024 x 768	-	2	Yes	DX 10	0	No	0KB	AMD	50 Watts
7	R700 RV790 XT	1920 x 1080	870 MHz	1	Yes	DX 10.1	1	No	0KB	AMD	190 Watts
8	R600 GT	1024 x 768	-	2	Yes	DX 10	0	No	0KB	AMD	150 Watts

Figure 3.1.3: First instances of selected GPUs data

1. Use the common column name `Release_Date` for both data. Extract `Release_Year` and `Release_Quarter` for each instance.
2. If `Recommended_Customer_Price` is a range, compute the average.

Listing 3.2.2: Extract Year and Quarter from Dates

```
month_to_quarter <- function(month) {
quarter <- switch(month,
                  "Jan" = "1",
                  "Feb" = "1",
                  "Mar" = "1",
                  "Apr" = "2",
                  "May" = "2",
                  "Jun" = "2",
                  "Jul" = "3",
                  "Aug" = "3",
                  "Sep" = "3",
                  "Oct" = "4",
                  "Nov" = "4",
                  "Dec" = "4",
                  "Unknown")
  return(quarter)
}

names(selected_cpu_data)[names(selected_cpu_data) == "Launch_Date"] <- "Release_Date"
selected_cpu_data$Release_Year <- as.integer(sub("Q[1-4]'(\\d+)", "\\1",
                                        gsub("\\s+", "", selected_cpu_data$Release_Date))) +
                                            2000
selected_cpu_data$Release_Quarter <- as.integer(sub("Q([1-4])'.*", "\\1",
                                          gsub("\\s+", "", selected_cpu_data$Release_Date)))

selected_gpu_data$Release_Year <-
  as.integer(sub(".*-(\\d{4}) .*", "\\1", selected_gpu_data$Release_Date))
selected_gpu_data$Release_Quarter <-
```

```
  as.character(sub(".*-(\\w+)-\\d{4}.*", "\\1", selected_gpu_data$Release_Date))
selected_gpu_data$Release_Quarter <- sapply(selected_gpu_data$Release_Quarter, month_to_quarter)
```

Chapter IV

Descriptive Statistics

1 CPU Data

```r
litho_year <- extracted_selected_cpu_data %>%
  group_by(Release_Year) %>%
  summarize(mean_litho = mean(Lithography),
            median_litho = median(Lithography),
    .groups = "drop"
  )

ggplot(litho_year, aes(x = Release_Year)) +
  geom_line(aes(y = mean_litho, color = "Mean")) +
  geom_line(aes(y = median_litho, color = "Median")) +
  scale_color_manual(values = c("Mean" = "blue", "Median" = "red")) +
  labs(x = "Year", y = "Lithography", title = "Mean and Median Lithography by Year") +
  scale_x_continuous(breaks = seq(min(litho_year$Release_Year), max(litho_year$Release_Year),
  by = 1)) +
  theme_minimal()
```

```r
options(repr.plot.width = 15, repr.plot.height =8)
ggplot(data = Intel_clean, aes(y = Status, x = Launch_Year, fill = Status)) +
  geom_boxplot() +
  labs(x = "Status", y = "Launch Date",title = "Boxplot of Status over Year") +
  scale_x_continuous(breaks = seq(min(Litho_year$Launch_Year), max(Litho_year$Launch_Year), by
      = 1))
```

```r
options(repr.plot.width = 15, repr.plot.height =8)
ggplot(data = Intel_clean, aes(y = Vertical_Segment, x = Launch_Year, fill = Vertical_Segment))
    +
  geom_boxplot() +
  labs(x = "Year", y = "Vertical Segment",title = "Boxplot of Vertical Segment over Year") +
  scale_x_continuous(breaks = seq(min(Litho_year$Launch_Year), max(Litho_year$Launch_Year), by
      = 1))
```

```r
options(repr.plot.width = 15, repr.plot.height =8)
ggplot(data = Intel_clean, aes(y = Instruction_Set, x = Launch_Year, fill = Instruction_Set)) +
  geom_violin() +
  labs(x = "Launch Date", y = "Instruction set",title = "Violinplot of Instruction set over
      Year") +
  scale_x_continuous(breaks = seq(min(Litho_year$Launch_Year), max(Litho_year$Launch_Year), by
      = 1))
```

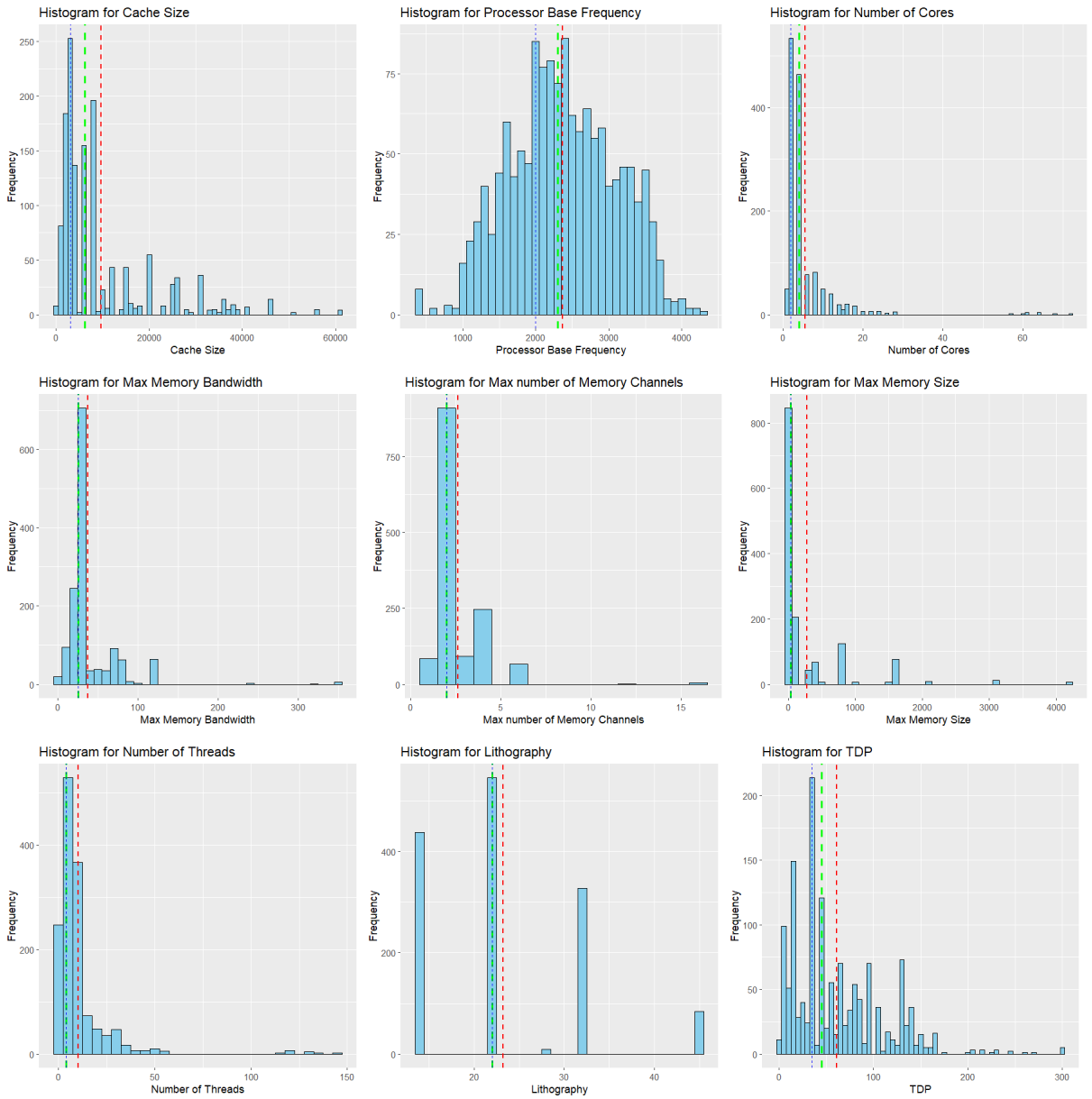

Figure 4.1.1: Histograms of several features in CPU data

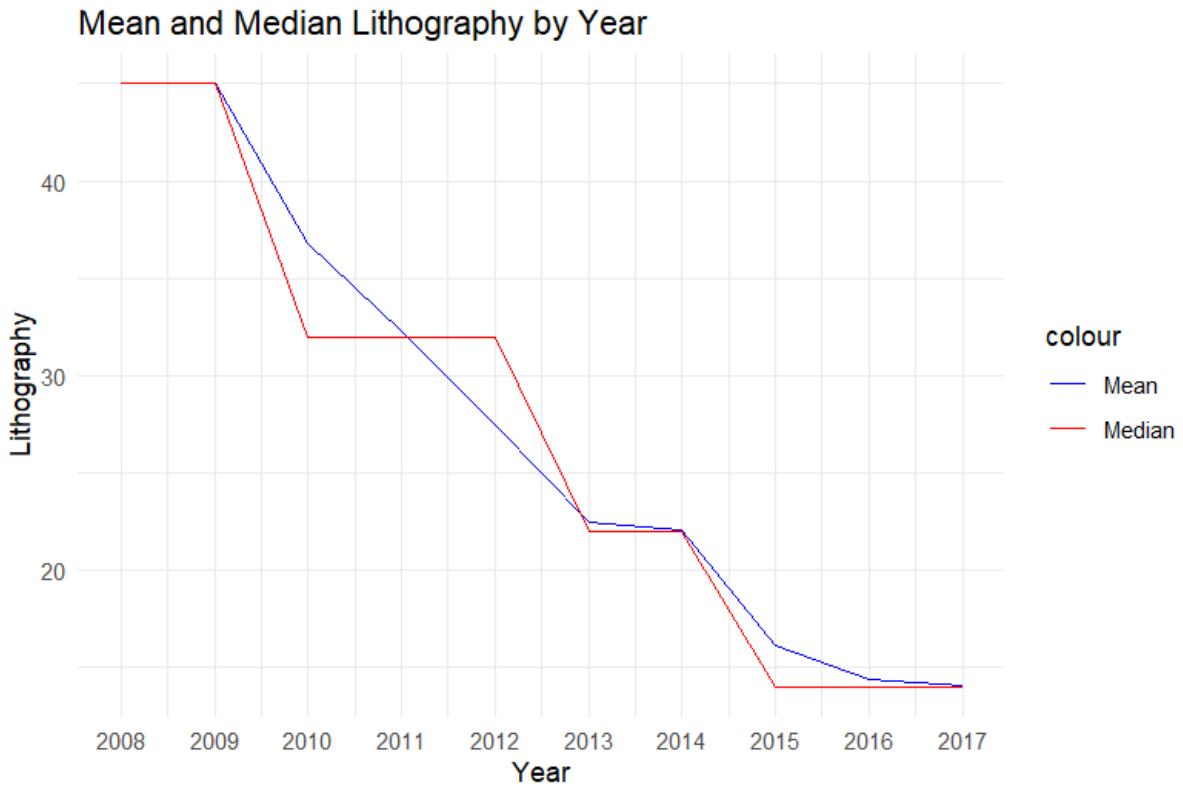

Figure 4.1.2: Mean and median of lithography over years

```
options(repr.plot.width = 15, repr.plot.height =8)
ggplot(data = Intel_clean, aes(y = Instruction_Set, x = Launch_Year, fill = Instruction_Set)) +
  geom_violin() +
  labs(x = "Launch Date", y = "Instruction set",title = "Violinplot of Instruction set over
      Year") +
  scale_x_continuous(breaks = seq(min(Litho_year$Launch_Year), max(Litho_year$Launch_Year), by
      = 1))
```

2 GPU Data

```
freq <- table(clean_GPUs$Manufacturer, clean_GPUs$Release_Year)
total_count <- colSums(freq)
percentage <- prop.table(freq, margin = 2) * 100

barplot(freq,
        legend.text = TRUE,
        main = "Counts of Each Year",
        xlab = "Year",
        ylab = "Count",
        col = c("skyblue", "salmon", "lightgreen", "yellow"),
        border = "black")
```

```
barplot(percentage,
        legend.text = TRUE,
        main = "Market Share Percentage by Year",
        xlab = "Year",
        ylab = "Percentage",
        col = c("skyblue", "salmon", "lightgreen", "yellow"),
```

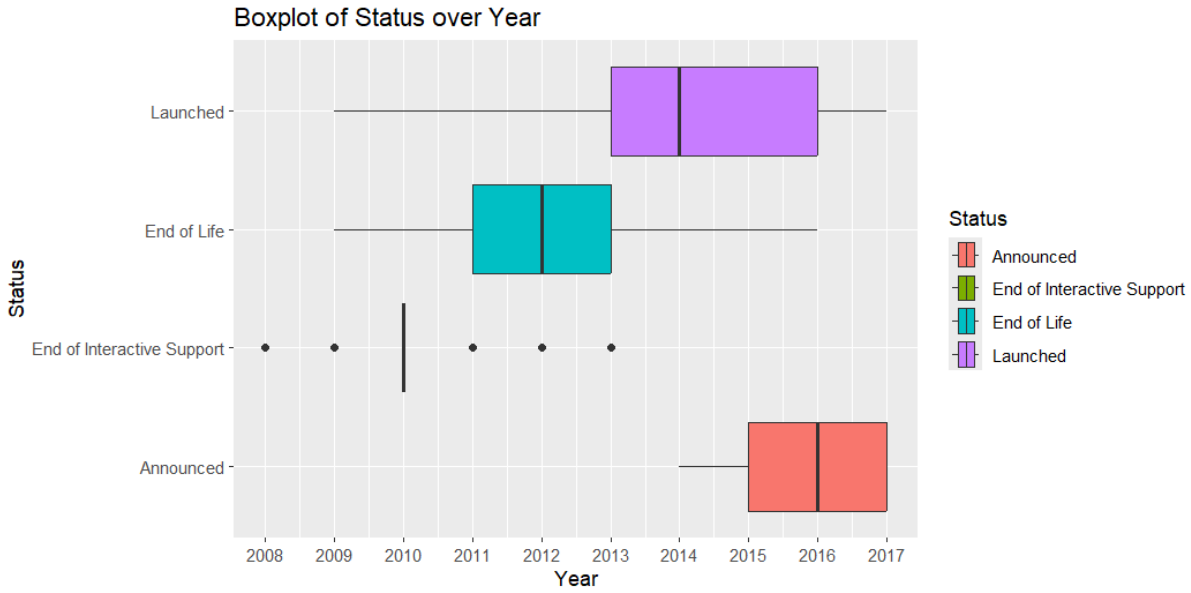

Figure 4.1.3: Status over years

```
       border = "black")
```

```
scatter_plot <-
  ggplot(clean_GPUs, aes(x = Release_Year + Release_Month/12, y = Memory, color =
      Manufacturer)) +
  geom_point() +
  scale_color_manual(values = c("skyblue", "salmon", "lightgreen", "yellow")) +
  labs(x = "Year", y = "GPU Memory", title = "Scatter Plot of GPU Memory over Years") +
  theme_minimal()
print(scatter_plot)
```

```
memory_summary <- clean_GPUs %>%
  group_by(Release_Year) %>%
  summarise(mean_memory = mean(Memory),
            median_memory = median(Memory))

line_plot <- ggplot(memory_summary, aes(x = Release_Year)) +
  geom_line(aes(y = mean_memory, color = "Mean")) +
  geom_line(aes(y = median_memory, color = "Median")) +
  scale_color_manual(values = c("Mean" = "blue", "Median" = "red")) +
  labs(x = "Year", y = "Memory", title = "Mean and Median Memory by Year") +
  theme_minimal()
print(line_plot)
```

```
moore_law <- data.frame(
  Release_Year = seq(min(memory_summary$Release_Year), max(memory_summary$Release_Year), 1),
  Memory = 2^(0.5 * (seq(min(memory_summary$Release_Year), max(memory_summary$Release_Year), 1)
      - min(memory_summary$Release_Year - 8)))
)

memory_summary$log_mean_memory <- log(memory_summary$mean_memory)
memory_summary$log_median_memory <- log(memory_summary$median_memory)

line_plot <- ggplot(memory_summary, aes(x = Release_Year)) +
  geom_line(aes(y = log_mean_memory, color = "Logarithm of Mean"), size = 1) +
  geom_line(aes(y = log_median_memory, color = "Logarithm of Median"), size = 1) +
```

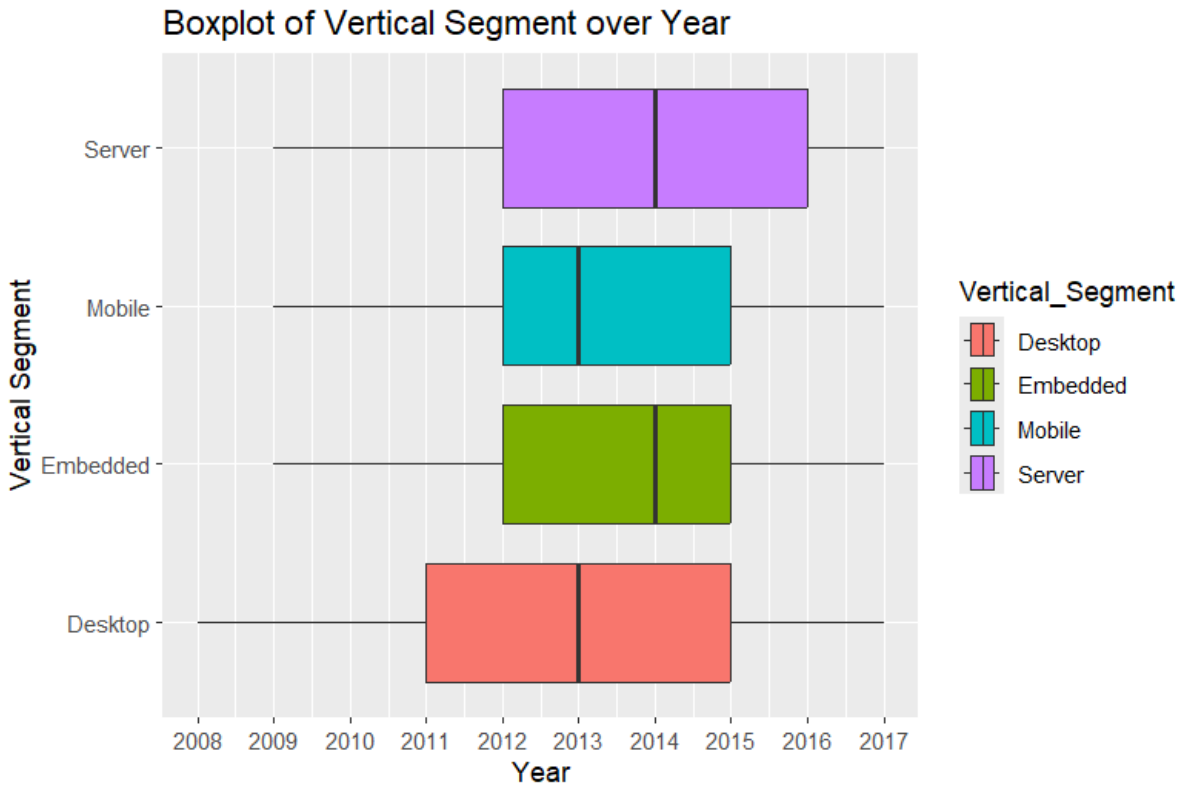

Figure 4.1.4: Segment over years

```
  geom_line(data = moore_law, aes(y = log(Memory), color = "Moore's Law"), size = 1, linetype =
      "dashed") +
  scale_color_manual(values = c("Logarithm of Mean" = "blue", "Logarithm of Median" = "red",
      "Moore's Law" = "green4")) +
  labs(x = "Year", y = "Logarithm of Memory", title = "Logarithm of Mean and Median Memory by
      Year") +
  theme_minimal()

print(line_plot)
```

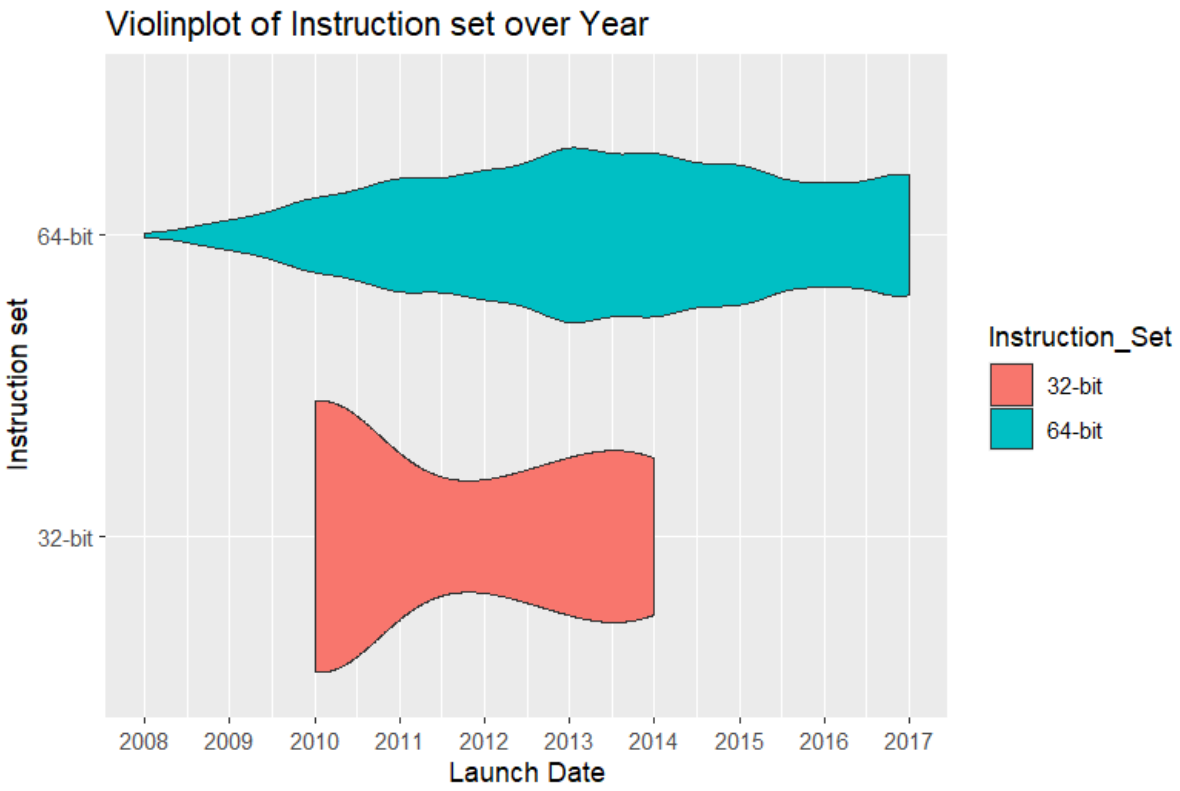

Figure 4.1.5: Instruction set over years

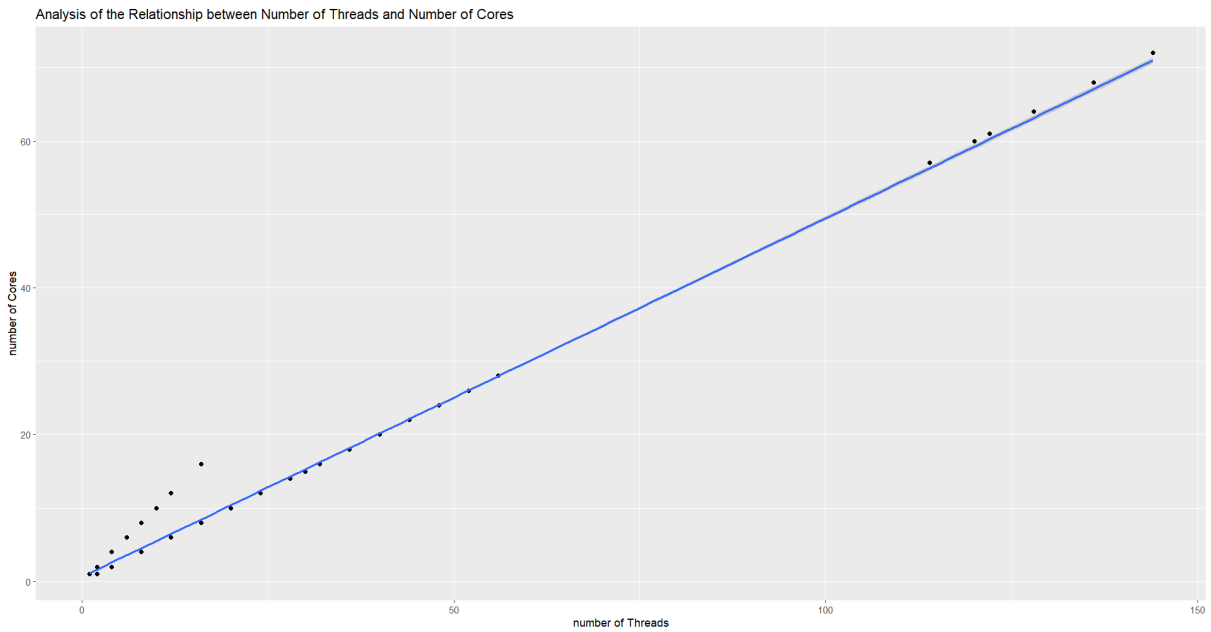

Figure 4.1.6: Relation between Number of Threads and Number of Cores

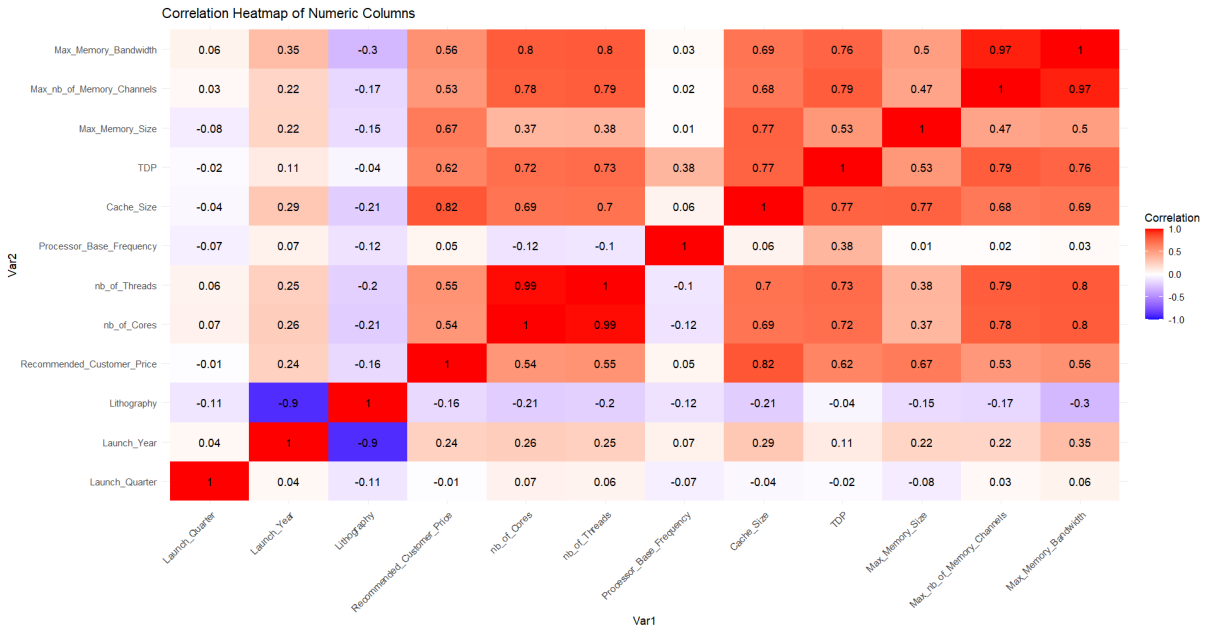

Figure 4.1.7: Overall Correlation

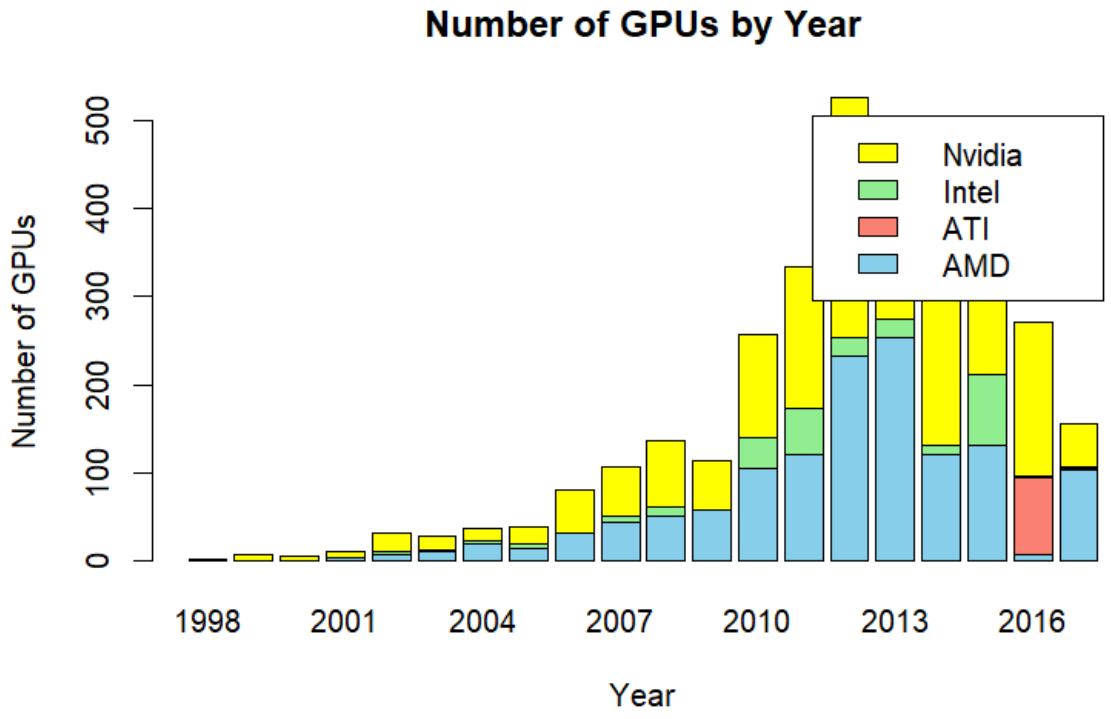

Figure 4.2.1: Sales of GPU manufacturer over years

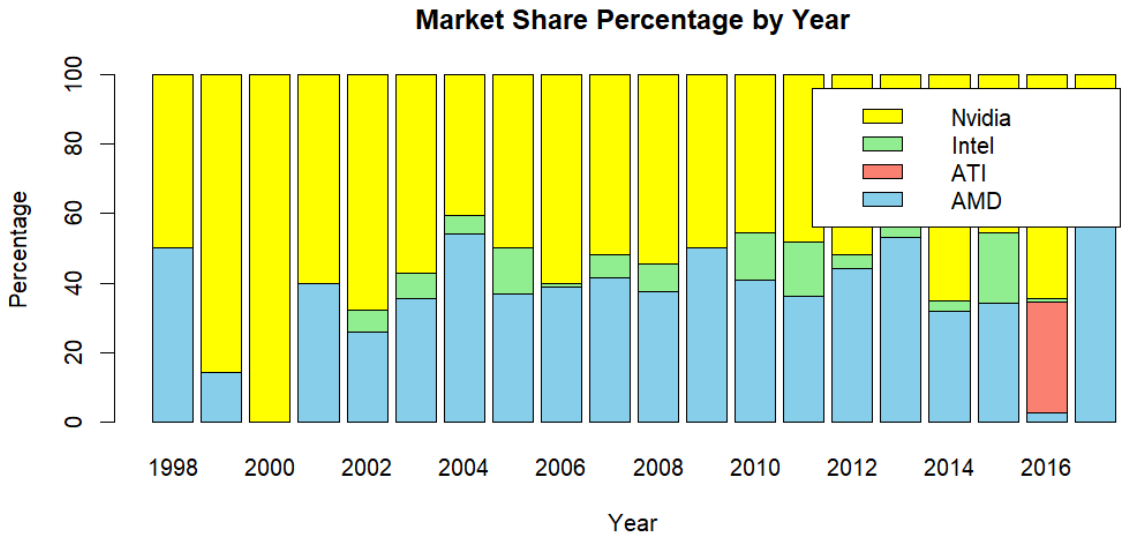

Figure 4.2.2: Market Share of GPU manufacturer over years

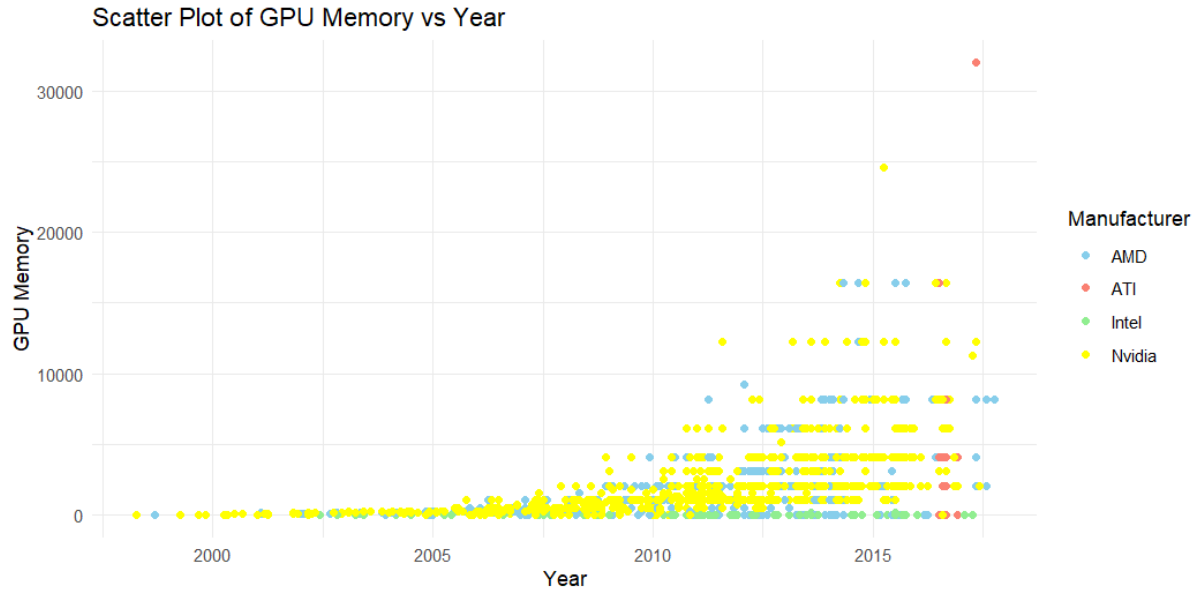

Figure 4.2.3: GPU Memory over Years

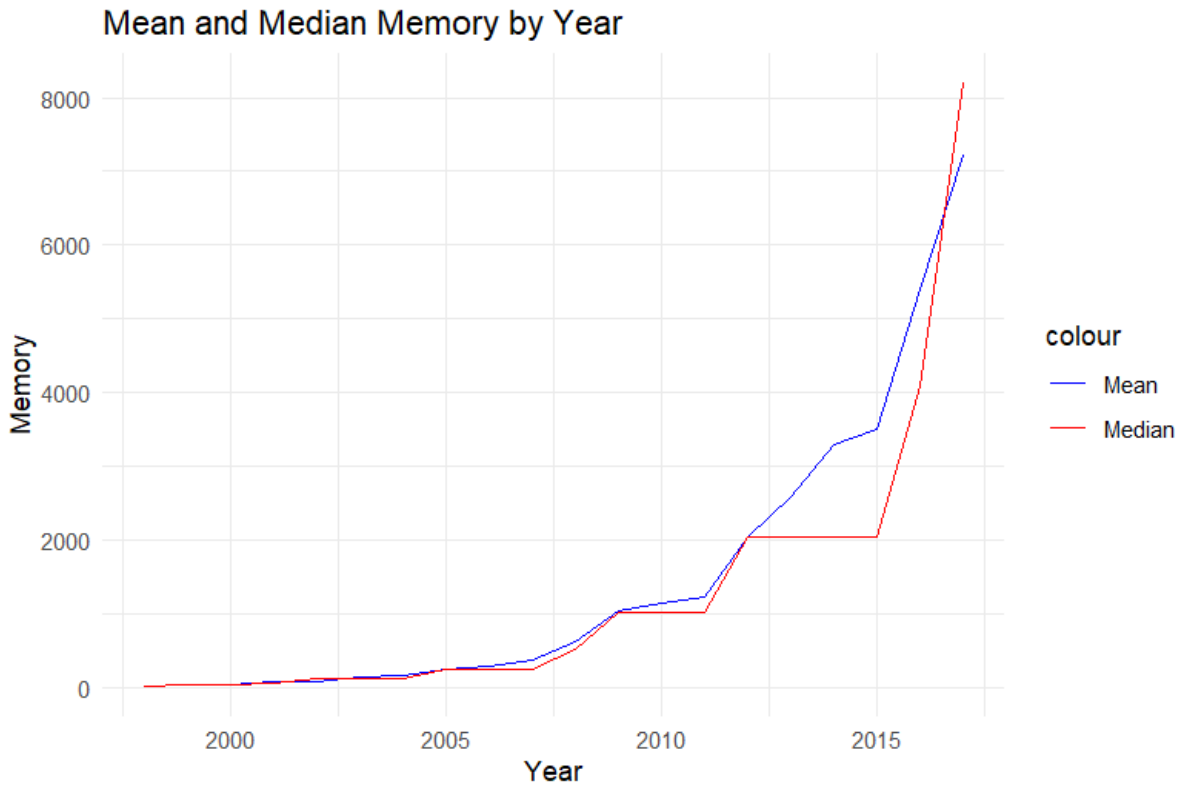

Figure 4.2.4: Mean and median of memory over years

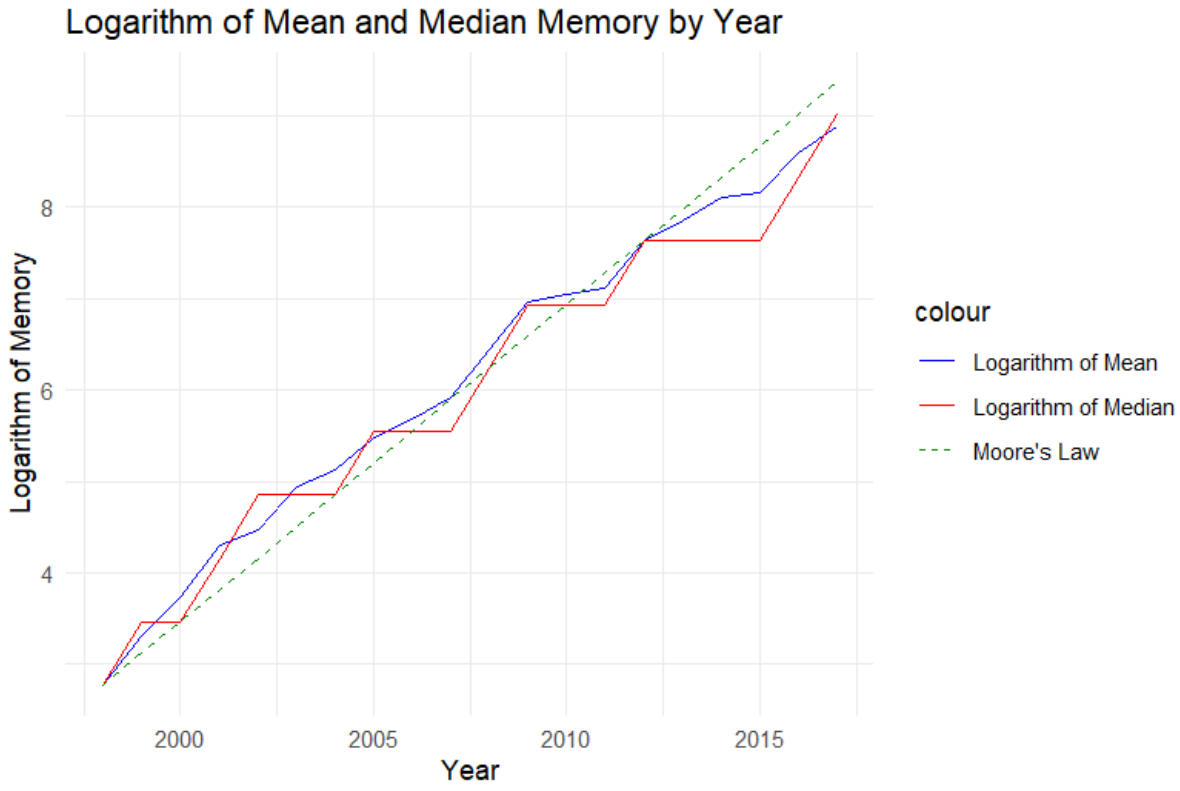

Figure 4.2.5: Logarithm of Mean and Median Memory by Year

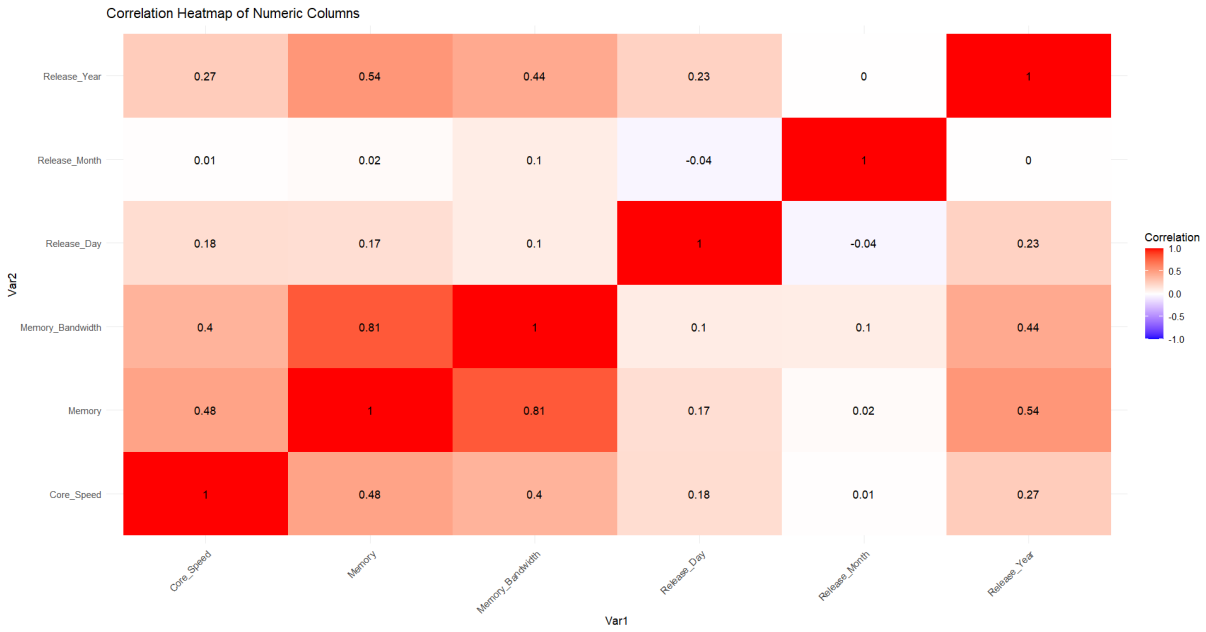

Figure 4.2.6: Enter Caption

Chapter V

Inferential Statistics

1 Two-way ANOVA

1.1 Problem Statement

Many customers believe that the product lines in each applied segment will have different numbers of cores, so perform ANOVA testing to check if there is a relationship between `Product_Collection` and `Vertical_Segment` to `nb_of_Cores`.

```
CPUs_ANOVA_input <-
get_valid(
  CPUs_selected %>%
    filter(Product_Collection %in% c("Core", "Legacy") &
         Vertical_Segment %in% c("Desktop", "Embedded", "Mobile")) %>%
    select(Product_Collection, Vertical_Segment, nb_of_Cores)
)
```

```
> table(Intel_clean$Product_Collection, Intel_clean$Vertical_Segment)

          Desktop Embedded Mobile Server
  Atom          3       22     48     26
  Celeron      27        8     44      0
  Core        118       32    165      0
  Legacy      119       41    168     56
  Pentium      42        3     27      6
  Quark         0        8      0      0
  Xeon          0       47      7    379
```

Figure 5.1.1: Data Partition based on `Product_Collection` and `Vertical_Segment`

1.2 Tests of Conditional Hypotheses

Before ANOVA test, let us test whether initial conditions are satisfied.

1. Testing whether `nb_of_Cores` is normal-distributed (H_0) or not (H_1). Since $p_{\text{value}} < 0.05$, we reject H_0 i.e. `nb_of_Cores` is not normal-distributed.

```
library(nortest)
shapiro.test(CPUs_ANOVA_input$nb_of_Cores)

# Output:  Shapiro-Wilk normality test

# data: CPUs_ANOVA_input$nb_of_Cores
```

```
# W = 0.62213, p-value < 2.2e-16
```

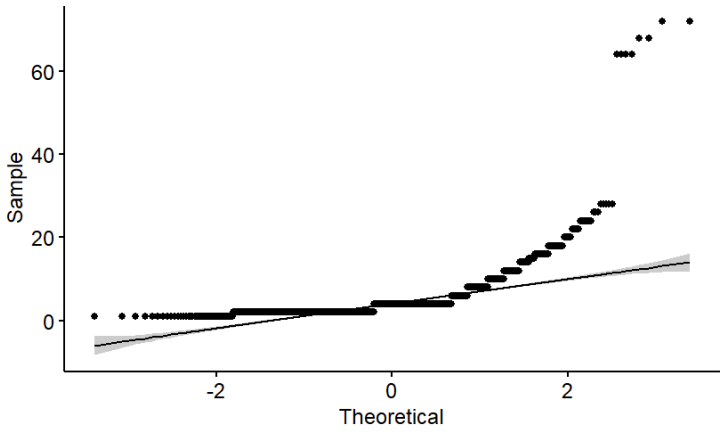

Figure 5.1.2: Q-Q plot returned by `ggqqplot(nb_of_Cores)`. Observed values are far away from normal expectation.

2. Testing whether the variances are equal (H_0) or there exist two groups of different variances (H_1). Since $p_{\text{value}} = 1.339e-07 < 0.05$, we reject H_0.

```
library(car)
leveneTest(CPUs_ANOVA_input$nb_of_Cores ~
           CPUs_ANOVA_input$Product_Collection *
           CPUs_ANOVA_input$Vertical_Segment)

# Output
# Levene's Test for Homogeneity of Variance (center = median)
# Df F value   Pr(>F)
# group  5  8.2835 1.339e-07 ***
#    645
```

Thus, `nb_of_Cores` is not normally distributed, and the variances of the groups are not equal. However, the sample size is large (more than 30). Therefore, we can disregard these violations.

1.3 ANOVA Test

Our objective is to examine the impact of the two factors `Product_Collection` and `Vertical_Segment` on `nb_of_Cores`. Since this is a large sample, even though the sample violates the assumptions of normal distribution and homogeneity of variances, the two-factor ANOVA method can still be applied. However, the results are only for reference. For more accurate results, other methods can be used.

```
av <- aov(nb_of_Cores ~
          Product_Collection *
          Vertical_Segment,
        data = CPUs_ANOVA_input)
summary(av)
# Output
#                                 Df Sum Sq Mean Sq F value Pr(>F)
#Product_Collection                1   94.6   94.64  46.895 1.75e-11 ***
#Vertical_Segment                  2  198.2   99.08  49.093 < 2e-16 ***
#Product_Collection:Vertical_Segment 2 27.4   13.71   6.795  0.0012 **
#Residuals                       645 1301.7   2.02
```

From the mean squares, we see that the effects of `Product_Collection` and `Vertical_Segment` appear to be similar, and both effects are statistically significant, as the p-values for both factors are very low. Specifically, for `Product_Collection`, the hypotheses are the average number of CPU cores in the two

product types, Core and Legacy, are the same (H_0) or different (H_1). For `Vertical_Segment`, they are the average number of CPU cores in the application segments Desktop, Embedded, and Mobile are the same (H_0) or at least two segments have different average numbers of CPU cores (H_1). Both H_0 hypotheses are rejected. We conclude that the average number of CPU cores is different for different product lines (`Product_Collection`) and application segments (`Vertical_Segment`), so the average number of CPU cores depends on the product line and application segment.

Chapter VI

Discussion