Intro To Statistical Computing

Final Project
MATH 208

Authors: *Ling Fei Zhang, Brandon Ma*

Instructor:	*Russell J. Steele*
Department:	Statistics
Date:	November 30, 2022

Contents

1 Set up

```
library(readxl)
library(here)
library(matlib)
library(formatR)
library(gridExtra)
library(tidyverse)
library(tidyr)
knitr::opts_chunk$set(tidy.opts=list(width.cutoff=60),
                      tidy=TRUE,
                      echo=TRUE,
                      # comment=NA,
                      message=FALSE,
                      warning=FALSE)
cpu_gpu_data<-read_csv("chip_dataset.csv")
```

```
names(cpu_gpu_data)
```

```
##  [1] "ID"                      "Product"                 "Type"
##  [4] "Release Date"            "Process Size (nm)"       "TDP (W)"
##  [7] "Die Size (mm^2)"         "Transistors (million)" "Freq (MHz)"
## [10] "Foundry"                 "Vendor"                  "FP16 GFLOPS"
## [13] "FP32 GFLOPS"             "FP64 GFLOPS"
```

```
head(cpu_gpu_data)
```

```
## # A tibble: 6 x 14
##      ID Product    Type  Relea~1 Proce~2 TDP (~3 Die S~4 Trans~5 Freq ~6 Foundry
##   <dbl> <chr>      <chr> <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <chr>
## 1     0 AMD Athlo~ CPU   2007-0~      65      45      77     122    2200 Unknown
## 2     1 AMD Athlo~ CPU   2018-0~      14      35     192    4800    3200 Unknown
## 3     2 Intel Cor~ CPU   2020-0~      10      28      NA      NA    2600 Intel
## 4     3 Intel Xeo~ CPU   2013-0~      22      80     160    1400    1800 Intel
## 5     4 AMD Pheno~ CPU   2011-0~      45     125     258     758    3700 Unknown
## 6     5 Intel Xeo~ CPU   2013-0~      22      95     160    1400    2400 Intel
## # ... with 4 more variables: Vendor <chr>, `FP16 GFLOPS` <dbl>,
## #   `FP32 GFLOPS` <dbl>, `FP64 GFLOPS` <dbl>, and abbreviated variable names
## #   1: `Release Date`, 2: `Process Size (nm)`, 3: `TDP (W)`,
## #   4: `Die Size (mm^2)`, 5: `Transistors (million)`, 6: `Freq (MHz)`
```

2 Task 1

2.1 part a)

```r
# Numerical summary in tables
summary <- cpu_gpu_data %>%
    select(Type, "Process Size (nm)", "TDP (W)", "Die Size (mm^2)",
        "Transistors (million)", "Freq (MHz)") %>%
    pivot_longer(cols = all_of(c("Process Size (nm)", "TDP (W)",
        "Die Size (mm^2)", "Transistors (million)", "Freq (MHz)")),
        names_to = "Characteristics") %>%
    group_by(Type, Characteristics) %>%
    summarise_all(list(Min = min, Max = max, Med = median, IQR = IQR),
        na.rm = TRUE)
summary
```

```
## # A tibble: 10 x 6
## # Groups:   Type [2]
##    Type  Characteristics         Min   Max   Med    IQR
##    <chr> <chr>                 <dbl> <dbl> <dbl>  <dbl>
##  1 CPU   Die Size (mm^2)           1   684   149  108
##  2 CPU   Freq (MHz)              600  4700  2400 1000
##  3 CPU   Process Size (nm)         7   180    32   76
##  4 CPU   TDP (W)                   1   400    65   60
##  5 CPU   Transistors (million)    37 19200   410 1086
##  6 GPU   Die Size (mm^2)           6   826   148  155
##  7 GPU   Freq (MHz)              100  2321   600  438
##  8 GPU   Process Size (nm)         0   250    40   62
##  9 GPU   TDP (W)                   2   900    50   91.5
## 10 GPU   Transistors (million)     8 54200   716 2590
```

```r
# Graphical summary
p1 <- cpu_gpu_data %>%
    ggplot(aes(x = Type, y = `Process Size (nm)`, fill = Type)) +
    stat_boxplot(geom = "errorbar", width = 0.25) + geom_boxplot(na.rm = TRUE) +
    ylab("Process Size (nm)") + xlab("Type") + ggtitle("Box plot of Process Size (nm) by Type")
p2 <- cpu_gpu_data %>%
    ggplot(aes(x = `Process Size (nm)`, col = Type)) + geom_density(size = 1,
    na.rm = TRUE) + xlab("Process Size (nm)") + ylab("Density") +
    ggtitle("Density plot of Process Size (nm) by Type")

grid.arrange(p1, p2)
```

```r
p1 <- cpu_gpu_data %>%
    ggplot(aes(x = Type, y = `TDP (W)`, fill = Type)) + stat_boxplot(geom = "errorbar",
    width = 0.25) + geom_boxplot(na.rm = TRUE) + ylab("TDP (W)") +
    xlab("Type") + ggtitle("Box plot of TDP (W) by Type")
p2 <- cpu_gpu_data %>%
    ggplot(aes(x = `TDP (W)`, col = Type)) + geom_density(size = 1,
    na.rm = TRUE) + xlab("TDP (W)") + ylab("Density") + ggtitle("Density plot of TDP (W) by Type")

grid.arrange(p1, p2)
```

```r
p1 <- cpu_gpu_data %>%
    ggplot(aes(x = Type, y = `Die Size (mm^2)`, fill = Type)) +
```

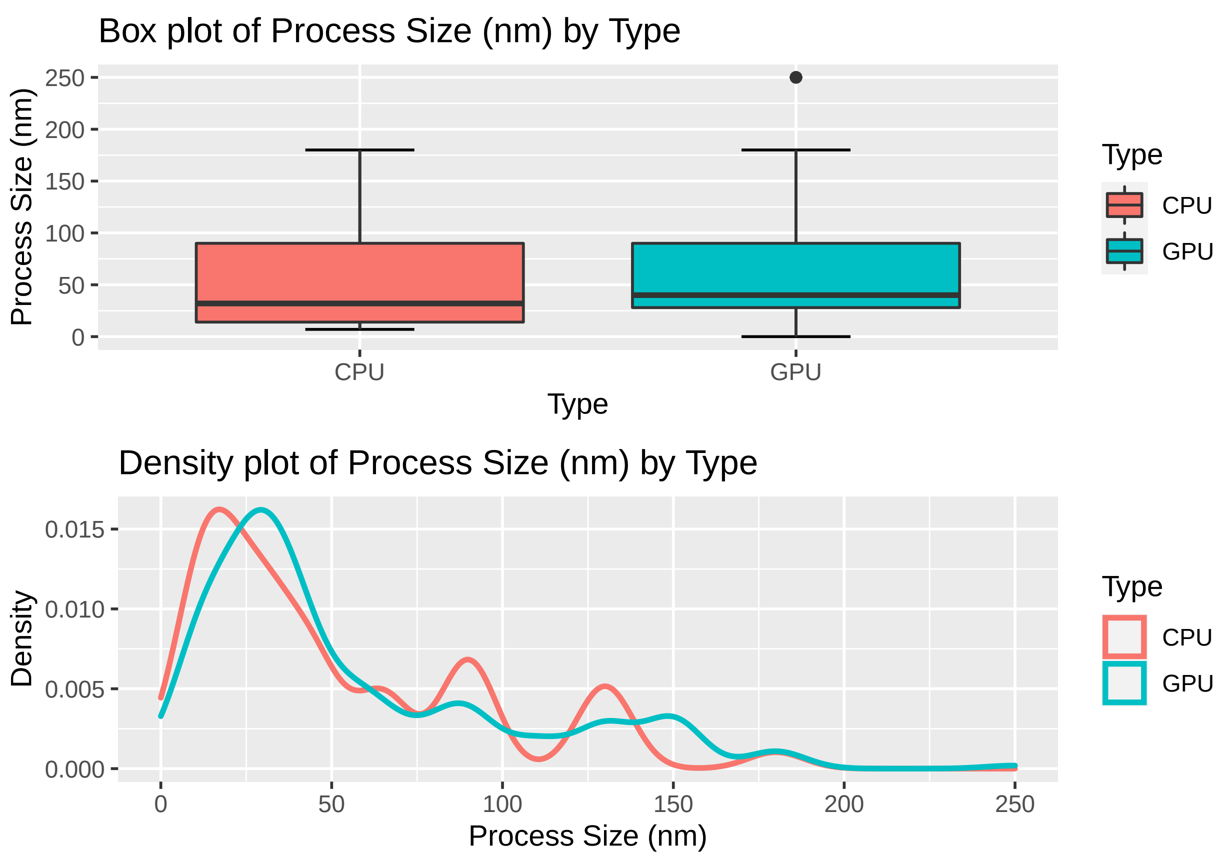


Figure 1: Process Size Data

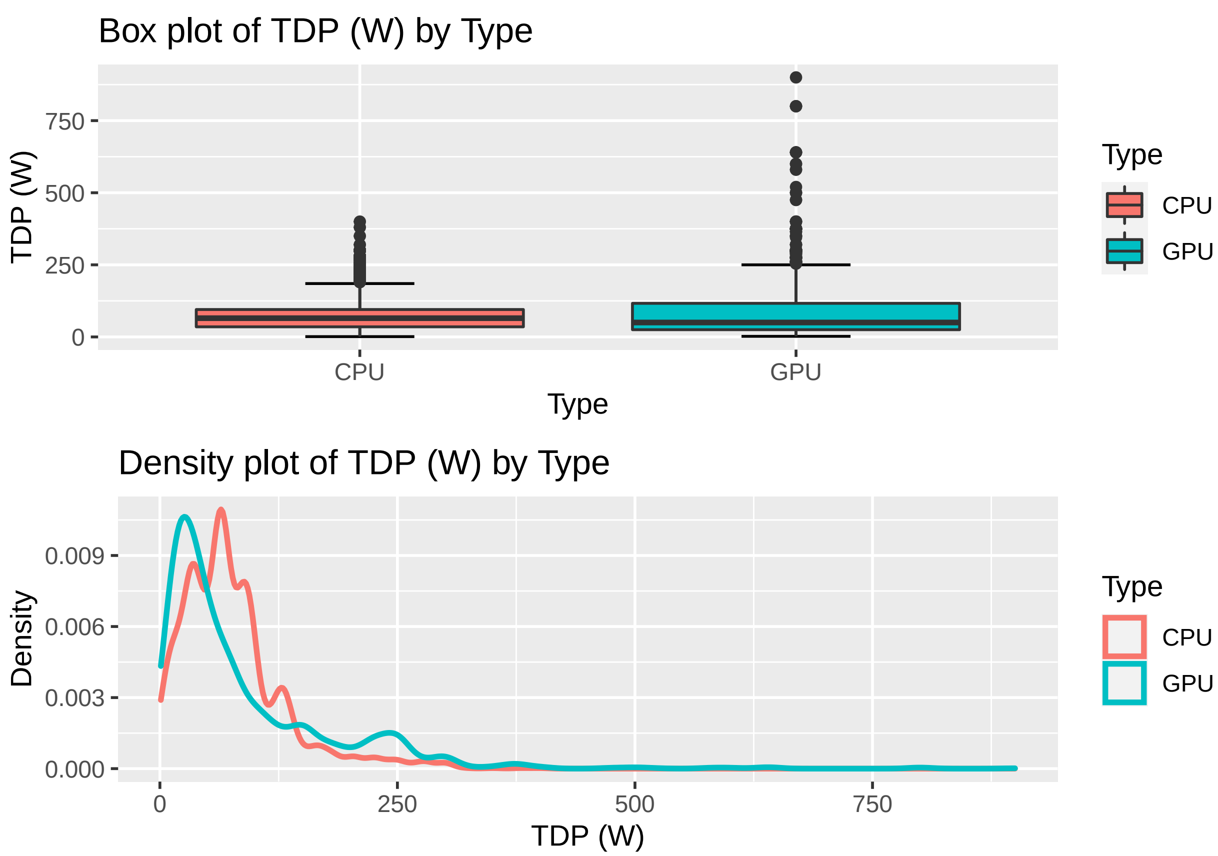


Figure 2: TDP Data

```
    stat_boxplot(geom = "errorbar", width = 0.25) + geom_boxplot(na.rm = TRUE) +
    ylab("Die Size (mm^2)") + xlab("Type") + ggtitle("Box plot of Die Size by Type")
p2 <- cpu_gpu_data %>%
    ggplot(aes(x = `Die Size (mm^2)`, col = Type)) + geom_density(size = 1,
    na.rm = TRUE) + xlab("Die Size (mm^2)") + ylab("Density") +
    ggtitle("Density plot of Die Size (mm^2) by Type")

grid.arrange(p1, p2)
```

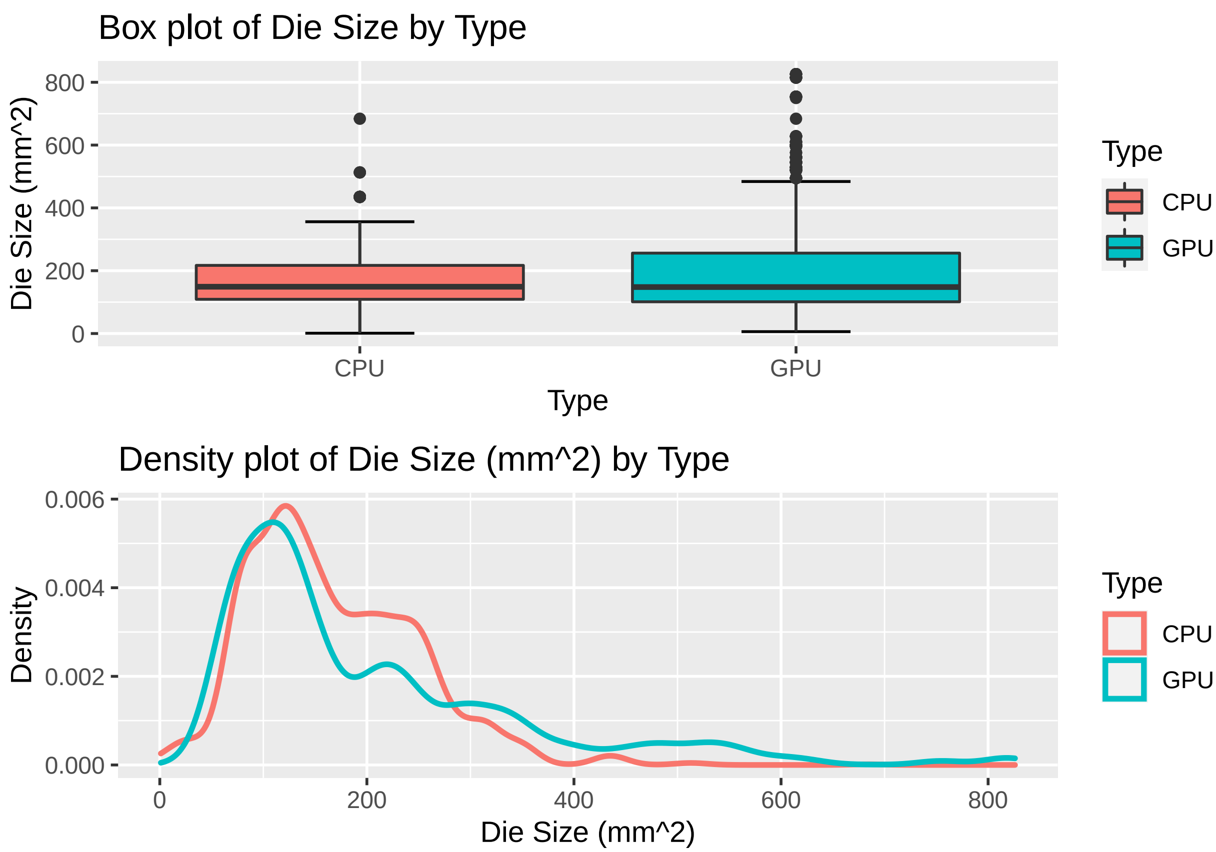


Figure 3: Die Size Data

```
p1 <- cpu_gpu_data %>%
    ggplot(aes(x = Type, y = `Transistors (million)`, fill = Type)) +
    stat_boxplot(geom = "errorbar", width = 0.25) + geom_boxplot(na.rm = TRUE) +
    ylab("Transistors (million)") + xlab("Type") + ggtitle("Box plot of Transistors (million) by Type")
p2 <- cpu_gpu_data %>%
    ggplot(aes(x = `Transistors (million)`, col = Type)) + geom_density(size = 1,
    na.rm = TRUE) + xlab("Transistors (million)") + ylab("Density") +
    ggtitle("Density plot of Transistors (million) by Type")

grid.arrange(p1, p2)
```

```
p1 <- cpu_gpu_data %>%
    ggplot(aes(x = Type, y = `Freq (MHz)`, fill = Type)) + stat_boxplot(geom = "errorbar",
    width = 0.25) + geom_boxplot(na.rm = TRUE) + ylab("Freq (MHz)") +
```

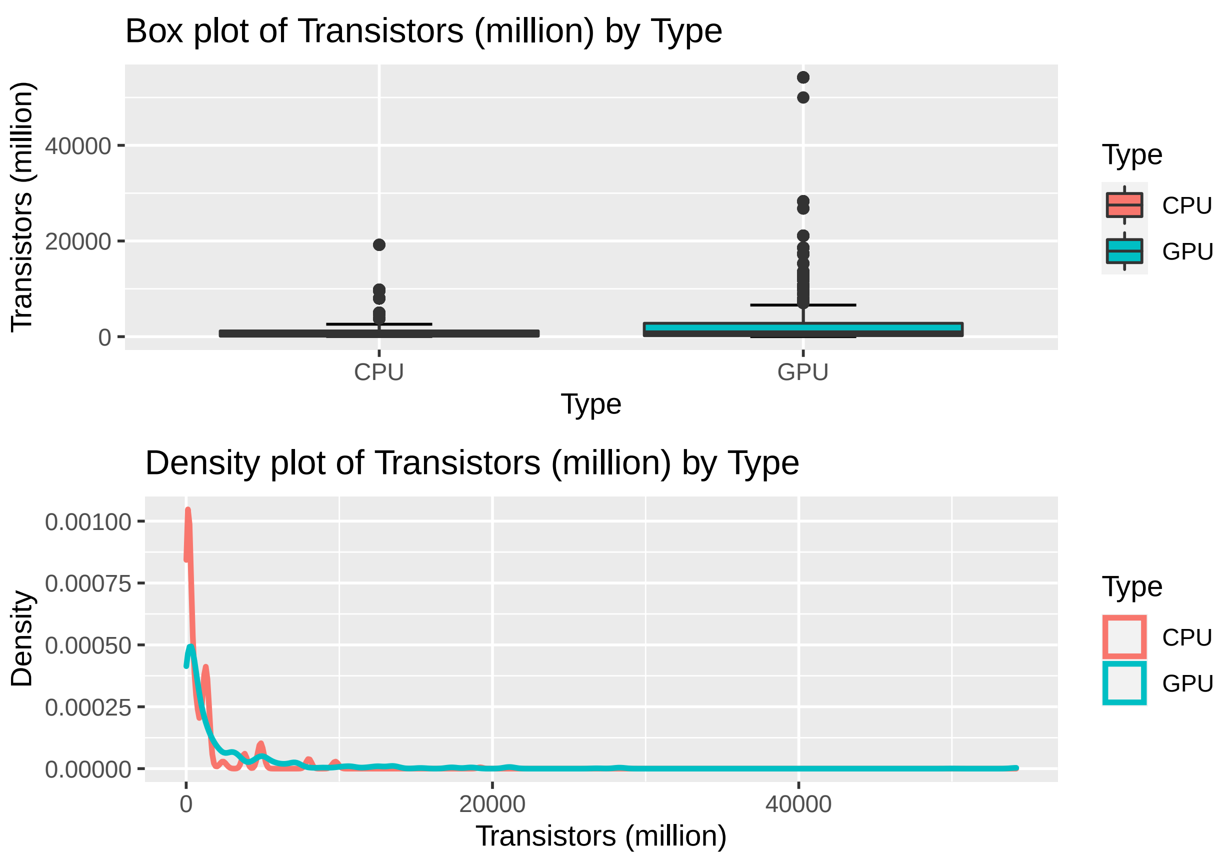


Figure 4: Transistors Data

```
    xlab("Type") + ggtitle("Box plot of Freq (MHz) by Type")
p2 <- cpu_gpu_data %>%
    ggplot(aes(x = `Freq (MHz)`, col = Type)) + geom_density(size = 1,
    na.rm = TRUE) + xlab("Freq (MHz)") + ylab("Density") + ggtitle("Density plot of Freq (MHz) by Type")

grid.arrange(p1, p2)
```

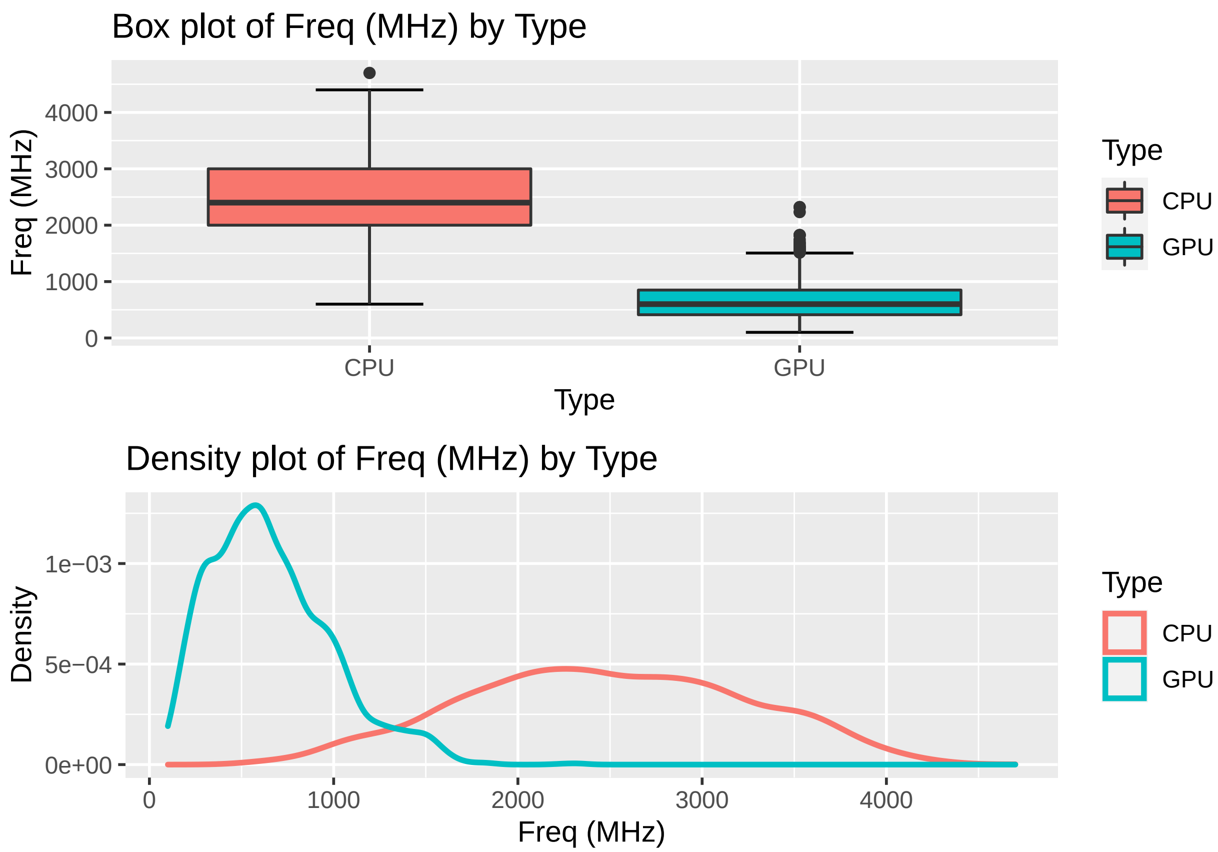


Figure 5: Frequency Data

2.1.1 Process Size

From the numerical summary, we can see that CPUs has a minimum of 7, maximum of 180, median of 32 and IQR of 76 and that GPUs has a minimum of 0, maximum of 250, median of 40 and IQR of 62. The characteristics of the distribution can also be observed from Figure 1. From the boxplot, we can see that the central location and spread of both types are very similar. However, the GPUs have an outlier at the point 250, where the max comes from. The density plot tells us that the skewness of both types are similar.

2.1.2 TDP

From the numerical summary, we can see that CPUs has a minimum of 1, maximum of 400, median of 65 and IQR of 60 and that GPUs has a minimum of 2, maximum of 900, median of 50 and IQR of 91.5. The characteristics of the distribution can also be observed from Figure 2. From the boxplot, we can see that the central locations are very similar, whereas the spread of the distribution is quite different as GPUs reach up to 900. The density plot tells us that the skewness of the two types are quite similar.

2.1.3 Die Size

From the numerical summary, we can see that CPUs has a minimum of 1, maximum of 684, median of 149 and IQR of 108 and that GPUs has a minimum of 6, maximum of 826, median of 148 and IQR of 155. The characteristics of the distribution can also be observed from Figure 3. From the boxplot, we can see that the central locations are very similar, with a slightly greater spread with the GPUs. The density plot tells us that the skewness of the two types are very similar.

2.1.4 Transistors

From the numerical summary, we can see that CPUs has a minimum of 37, maximum of 19200, median of 410 and IQR of 1086 and that GPUs has a minimum of 8, maximum of 54200, median of 716 and IQR of 2590. The characteristics of the distribution can also be observed from Figure 4. From the boxplot, we can see that the central locations are somewhat similar. However, the spread of the GPUs is much greater than the spread of the CPUs. This can be easily observed, as the maximum number of transistors in a CPU is 19200 in contrast to 54200 in GPU. We can also observe a few outliers in the GPUs from the boxplot. From the density plot, we can see that the skewness of the distributions are also quite different. We can see that proportional to the size of the distribution, CPUs tend to have a smaller number of transistors compared to GPUs, as the skewness is much higher when the number of transistors is small.

2.1.5 Frequency

From the numerical summary, we can see that CPUs has a minimum of 600, maximum of 4700, median of 2400 and IQR of 1000 and that GPUs has a minimum of 100, maximum of 2321, median of 600 and IQR of 438. The characteristics of the distribution can also be observed from Figure 5. From the boxplot, we can see that the central locations are very different. This is clear as the medians are 2400 and 600 for CPUs and GPUs respectively. The density plot tells us that both the skewness and the spread of the two types are very different. CPUs have a bigger spread, but a smaller skewness. On the other hand, GPUs have a smaller spread, but a greater skewness.

2.2 part b)

```
# Numerical Summary
table(cpu_gpu_data["Foundry"])
```

```
## Foundry
##      GF     IBM   Intel     NEC Renesas Samsung    Sony    TSMC     UMC Unknown
##     265       3    1390       2       1      60      10    2178      79     866
```

```
vendor_vs_foundry <- cpu_gpu_data %>%
    group_by(Vendor, Type) %>%
    summarise(`unique foundry` = n_distinct(Foundry))
vendor_vs_foundry
```

```
## # A tibble: 7 x 3
## # Groups:   Vendor [5]
##   Vendor Type  `unique foundry`
##   <chr>  <chr>            <int>
## 1 AMD    CPU                  3
## 2 AMD    GPU                  4
## 3 ATI    GPU                  5
## 4 Intel  CPU                  1
## 5 Intel  GPU                  2
## 6 NVIDIA GPU                  5
## 7 Other  GPU                  4
```

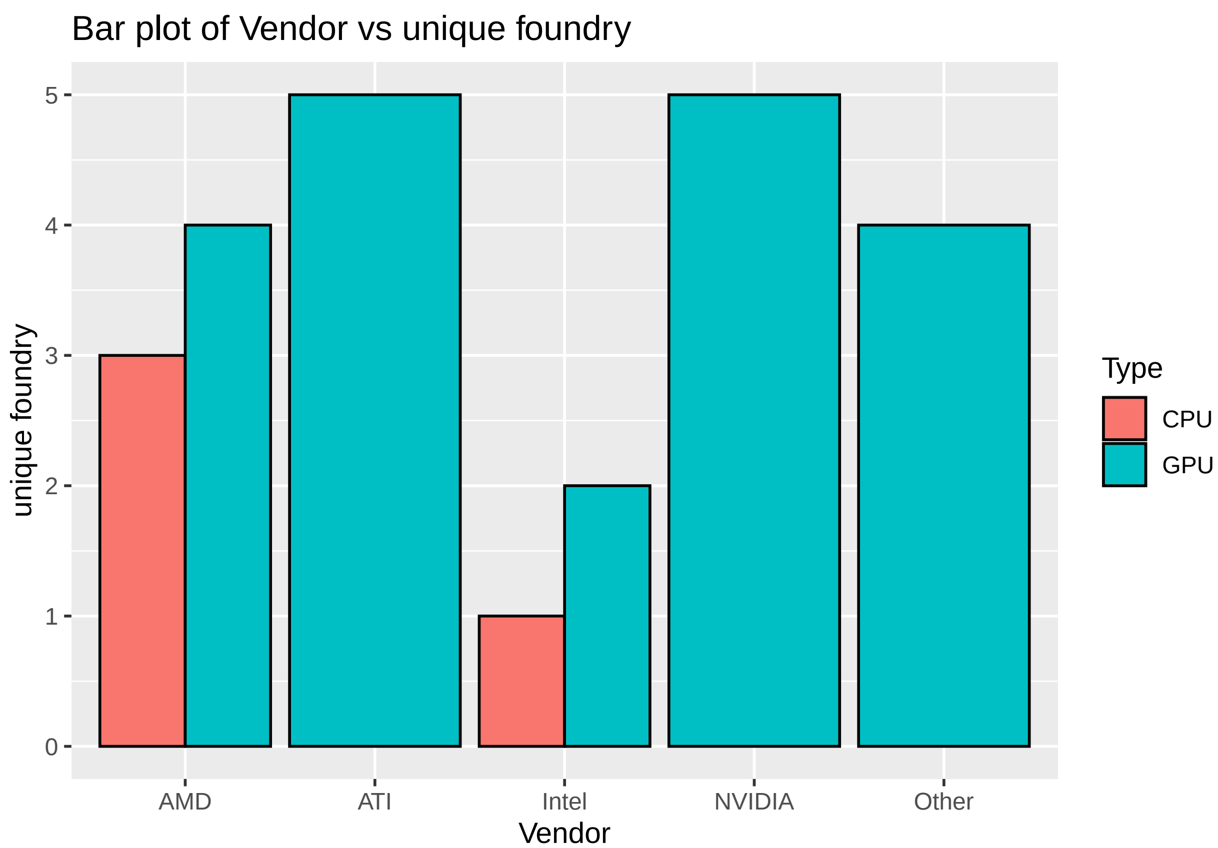
We can see that the only vendor that has a unique foundry is Intel, but that is only specific to the CPU processors. For GPUs, even Intel has 2 distinct foundries. All other vendors have more than 1 distinct foundry, regardless of the type of the processor.

```
foundry_vs_vendor <- cpu_gpu_data %>%
    group_by(Foundry, Type) %>%
    summarise(`unique vendor` = n_distinct(Vendor))
foundry_vs_vendor
```

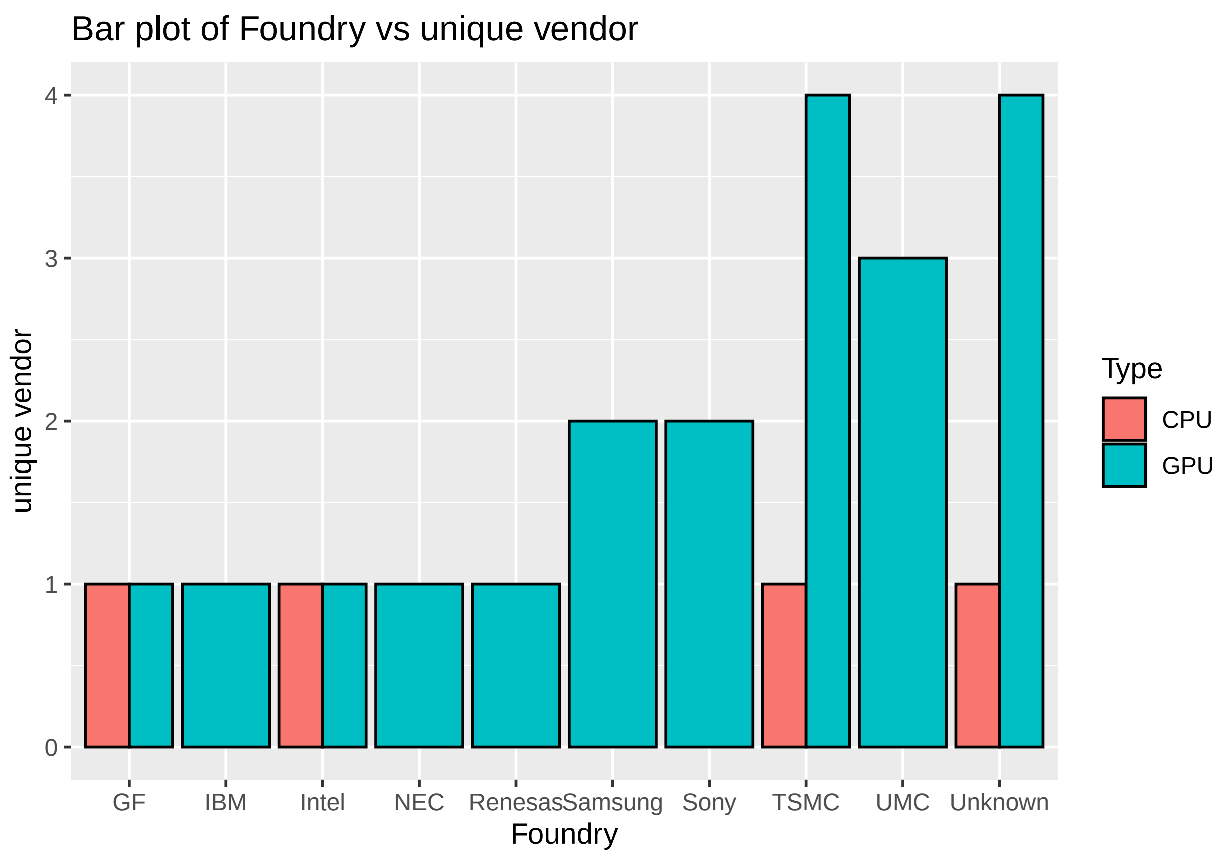
```
## # A tibble: 14 x 3
## # Groups:   Foundry [10]
##    Foundry Type  `unique vendor`
##    <chr>   <chr>           <int>
##  1 GF      CPU                 1
##  2 GF      GPU                 1
##  3 IBM     GPU                 1
##  4 Intel   CPU                 1
##  5 Intel   GPU                 1
##  6 NEC     GPU                 1
##  7 Renesas GPU                 1
##  8 Samsung GPU                 2
##  9 Sony    GPU                 2
## 10 TSMC    CPU                 1
## 11 TSMC    GPU                 4
## 12 UMC     GPU                 3
## 13 Unknown CPU                 1
## 14 Unknown GPU                 4
```

Here, we can see from the second table that many of the foundries have a unique vendor. We should keep in mind that IMB, NEC, Renesas, Samsung, Sony and UMC are all foundries that have a small count. We observe that in general, each foundry like to stay with a small number of unique vendors. Additionally, we observe that **all CPU foundries have a unique vendor**. This information can also be observed from the graphs below, perhaps even better.

```
# Graphical Summary
vendor_vs_foundry %>%
    ggplot(aes(x = Vendor, y = `unique foundry`, group = Type,
        fill = Type)) + geom_bar(col = "black", stat = "identity",
    position = "dodge") + ggtitle("Bar plot of Vendor vs unique foundry")
```

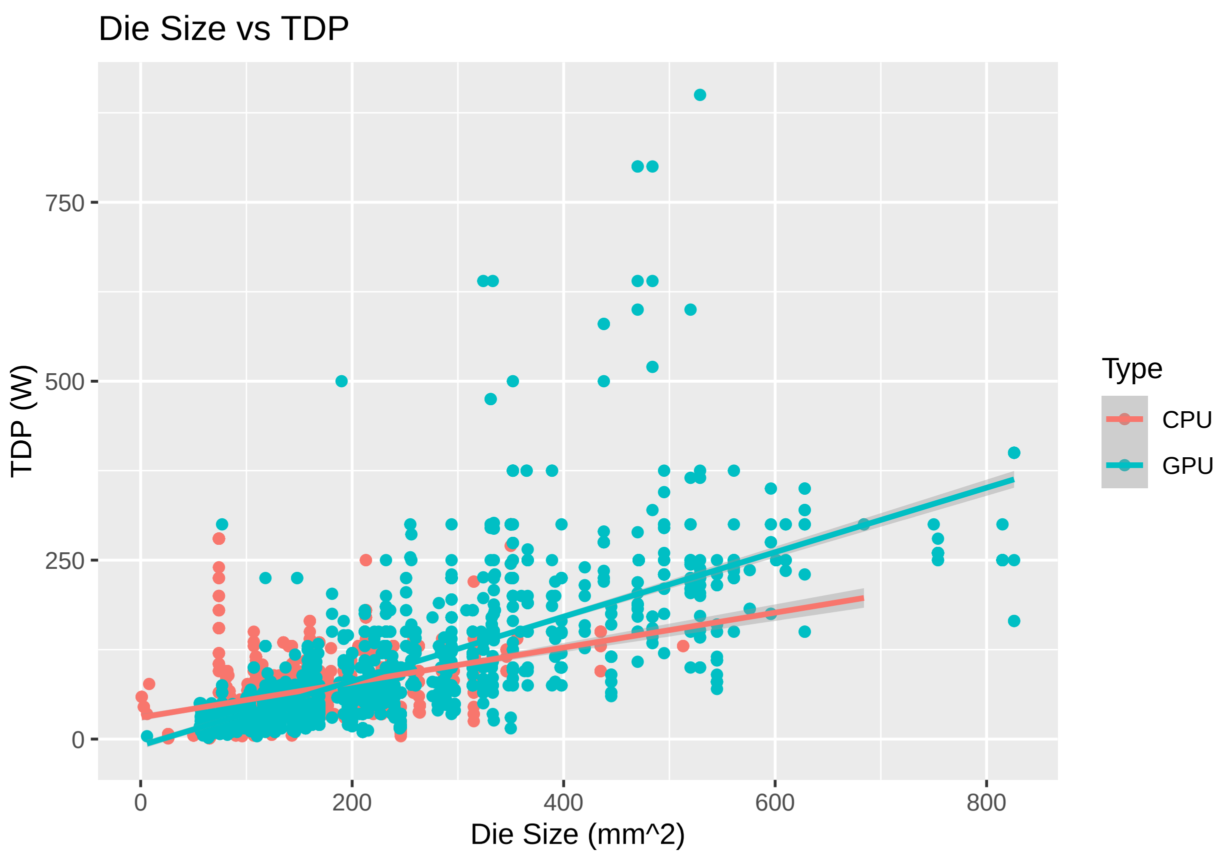



```
foundry_vs_vendor %>%
    ggplot(aes(x = Foundry, y = `unique vendor`, group = Type,
        fill = Type)) + geom_bar(col = "black", stat = "identity",
    position = "dodge") + ggtitle("Bar plot of Foundry vs unique vendor")
```



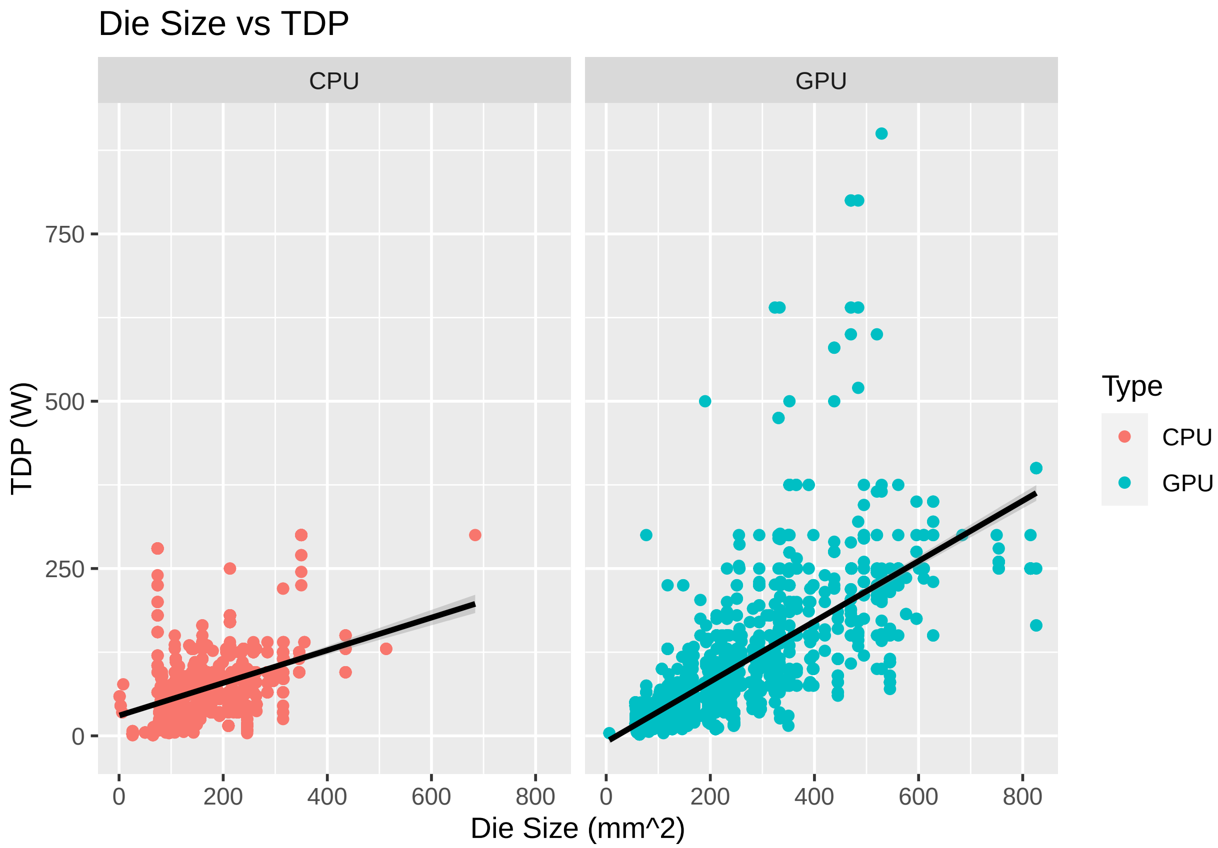
2.3 part c)

```
# Graph
cpu_gpu_data %>%
    ggplot(aes(x = `Die Size (mm^2)`, y = `TDP (W)`, col = Type)) +
    geom_point() + xlab("Die Size (mm^2)") + ylab("TDP (W)") +
    ggtitle("Die Size vs TDP") + geom_smooth(method = "lm")
```



We can also observe the graphs separately, as follows:

```
cpu_gpu_data %>%
    ggplot(aes(x = `Die Size (mm^2)`, y = `TDP (W)`, col = Type)) +
    geom_point() + facet_wrap(~Type) + xlab("Die Size (mm^2)") +
    ylab("TDP (W)") + ggtitle("Die Size vs TDP") + geom_smooth(method = "lm",
    col = "black")
```



In both graphs, we can see that the association between Die Size and TDP does not really depend on Type. It's worth noting that the two features in GPUs have a greater spread, whereas the association in CPUs are much more concentrated. However, we can observe that both types of processors follow a similar regression line. We can also observe the association numerically.

```
# Numerical summary in tables
summary[c(1, 4, 6, 9), ]
```

```
## # A tibble: 4 x 6
## # Groups:   Type [2]
##   Type  Characteristics   Min   Max   Med   IQR
##   <chr> <chr>           <dbl> <dbl> <dbl> <dbl>
## 1 CPU   Die Size (mm^2)     1   684   149 108
## 2 CPU   TDP (W)             1   400    65  60
## 3 GPU   Die Size (mm^2)     6   826   148 155
## 4 GPU   TDP (W)             2   900    50  91.5
```

From the numerical table above, we can also observe that the medians of both characteristics are similar, regardless of the processor type (CPU or GPU).

3 Task 2

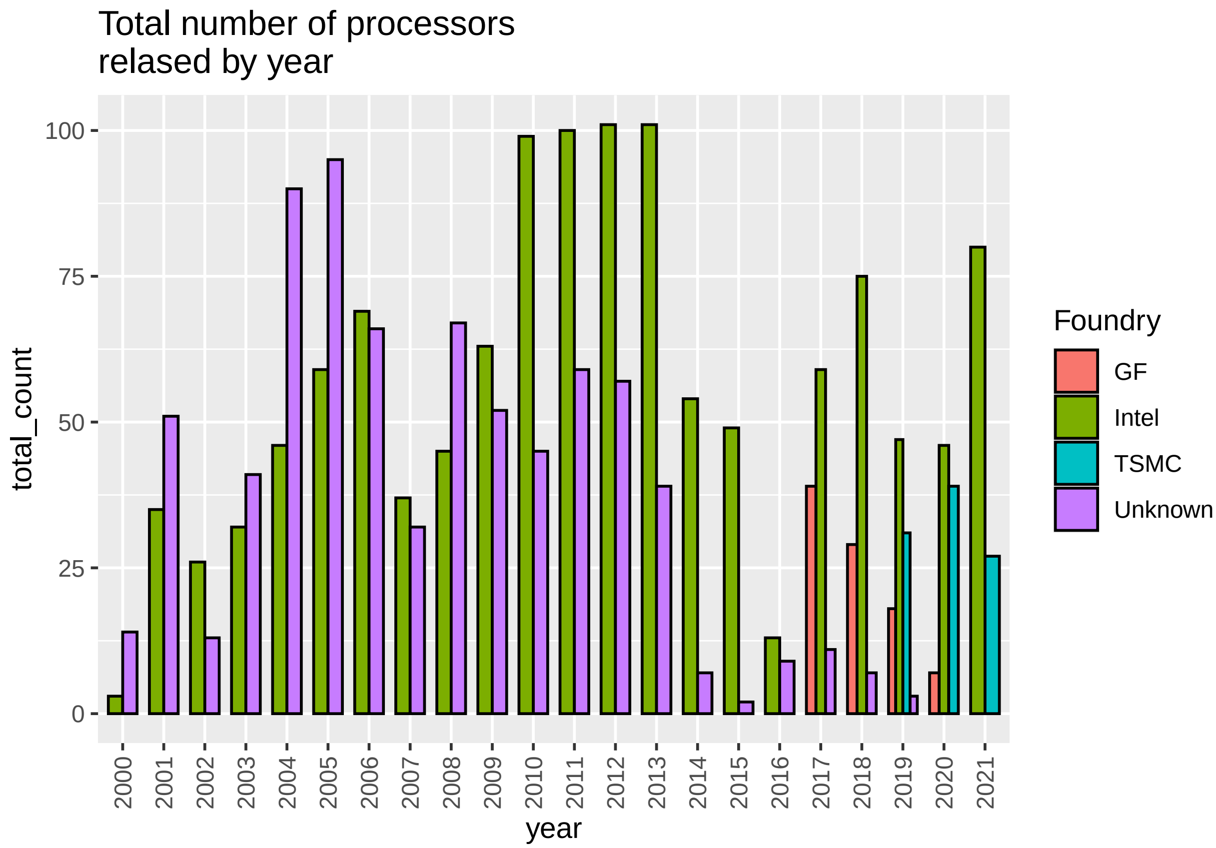
3.1 Part a)

3.1.1 CPU

```
foundry_vs_year <- cpu_gpu_data[cpu_gpu_data$Type == "CPU", ] %>%
    group_by(Foundry, year = format(as.Date(na.omit(`Release Date`),
        format = "%Y-%m-%d"), "%Y")) %>%
    summarise(total_count = n())

foundry_vs_year <- na.omit(foundry_vs_year)

foundry_vs_year
```

```
## # A tibble: 49 x 3
## # Groups:   Foundry [4]
##    Foundry year  total_count
##    <chr>   <chr>       <int>
##  1 GF      2017           39
##  2 GF      2018           29
##  3 GF      2019           18
##  4 GF      2020            7
##  5 Intel   2000            3
##  6 Intel   2001           35
##  7 Intel   2002           26
##  8 Intel   2003           32
##  9 Intel   2004           46
## 10 Intel   2005           59
## # ... with 39 more rows
```

```
foundry_vs_year %>%
    ggplot(aes(x = year, y = total_count, group = Foundry, fill = Foundry,
        width = 0.7)) + geom_bar(col = "black", stat = "identity",
    position = "dodge", na.rm = TRUE) + ggtitle("Total number of processors
relased by year") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5,
        hjust = 1))
```

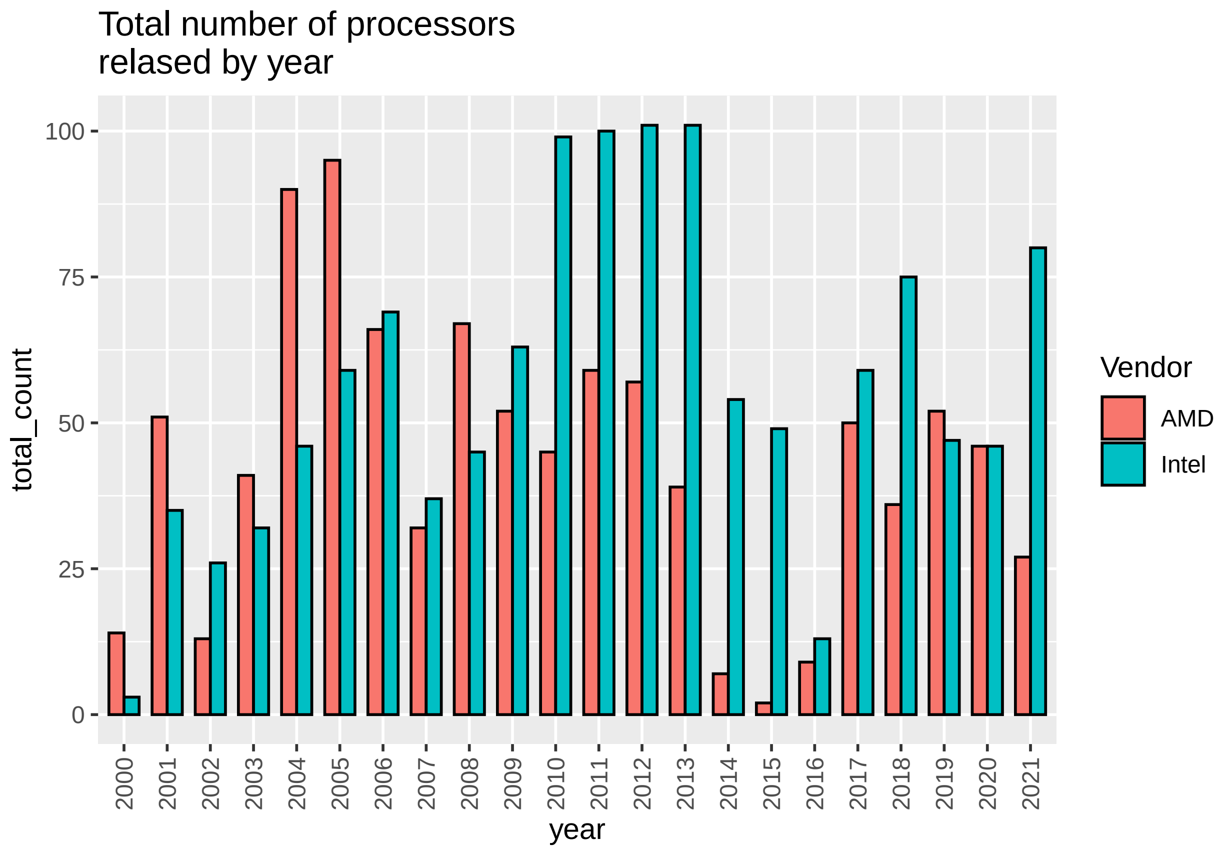



```r
vendor_vs_year <- cpu_gpu_data[cpu_gpu_data$Type == "CPU", ] %>%
    group_by(Vendor, year = format(as.Date(na.omit(`Release Date`),
        format = "%Y-%m-%d"), "%Y")) %>%
    summarise(total_count = n())

vendor_vs_year <- na.omit(vendor_vs_year)

vendor_vs_year
```

```
## # A tibble: 44 x 3
## # Groups:   Vendor [2]
##    Vendor year  total_count
##    <chr>  <chr>       <int>
##  1 AMD    2000           14
##  2 AMD    2001           51
##  3 AMD    2002           13
##  4 AMD    2003           41
##  5 AMD    2004           90
##  6 AMD    2005           95
##  7 AMD    2006           66
##  8 AMD    2007           32
##  9 AMD    2008           67
## 10 AMD    2009           52
## # ... with 34 more rows
```

```r
vendor_vs_year %>%
    ggplot(aes(x = year, y = total_count, group = Vendor, fill = Vendor,
```

```
        width = 0.7)) + geom_bar(col = "black", stat = "identity",
    position = "dodge", na.rm = TRUE) + ggtitle("Total number of processors
relased by year") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5,
        hjust = 1))
```



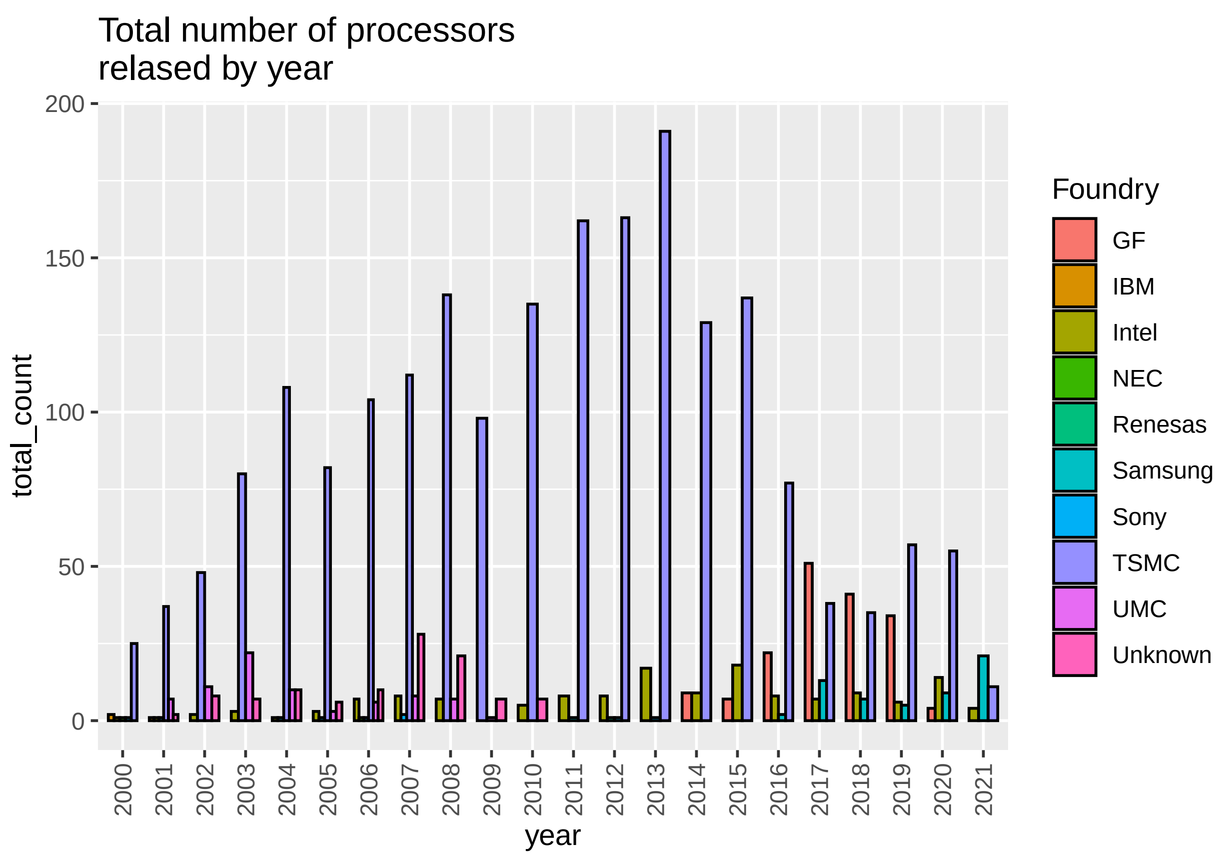
3.1.2 GPU

```
foundry_vs_year <- cpu_gpu_data[cpu_gpu_data$Type == "GPU", ] %>%
    group_by(Foundry, year = format(as.Date(na.omit(`Release Date`),
        format = "%Y-%m-%d"), "%Y")) %>%
    summarise(total_count = n())

foundry_vs_year <- na.omit(foundry_vs_year)

foundry_vs_year
```

```
## # A tibble: 89 x 3
## # Groups:   Foundry [10]
##    Foundry year  total_count
##    <chr>   <chr>       <int>
##  1 GF      2014            9
##  2 GF      2015            7
##  3 GF      2016           22
##  4 GF      2017           51
##  5 GF      2018           41
```

```
##   6 GF       2019            34
##   7 GF       2020             4
##   8 IBM      2000             2
##   9 IBM      2001             1
## 10 Intel    2000             1
## # ... with 79 more rows
```

```
foundry_vs_year %>%
    ggplot(aes(x = year, y = total_count, group = Foundry, fill = Foundry,
        width = 0.7)) + geom_bar(col = "black", stat = "identity",
    position = "dodge", na.rm = TRUE) + ggtitle("Total number of processors
relased by year") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5,
        hjust = 1))
```

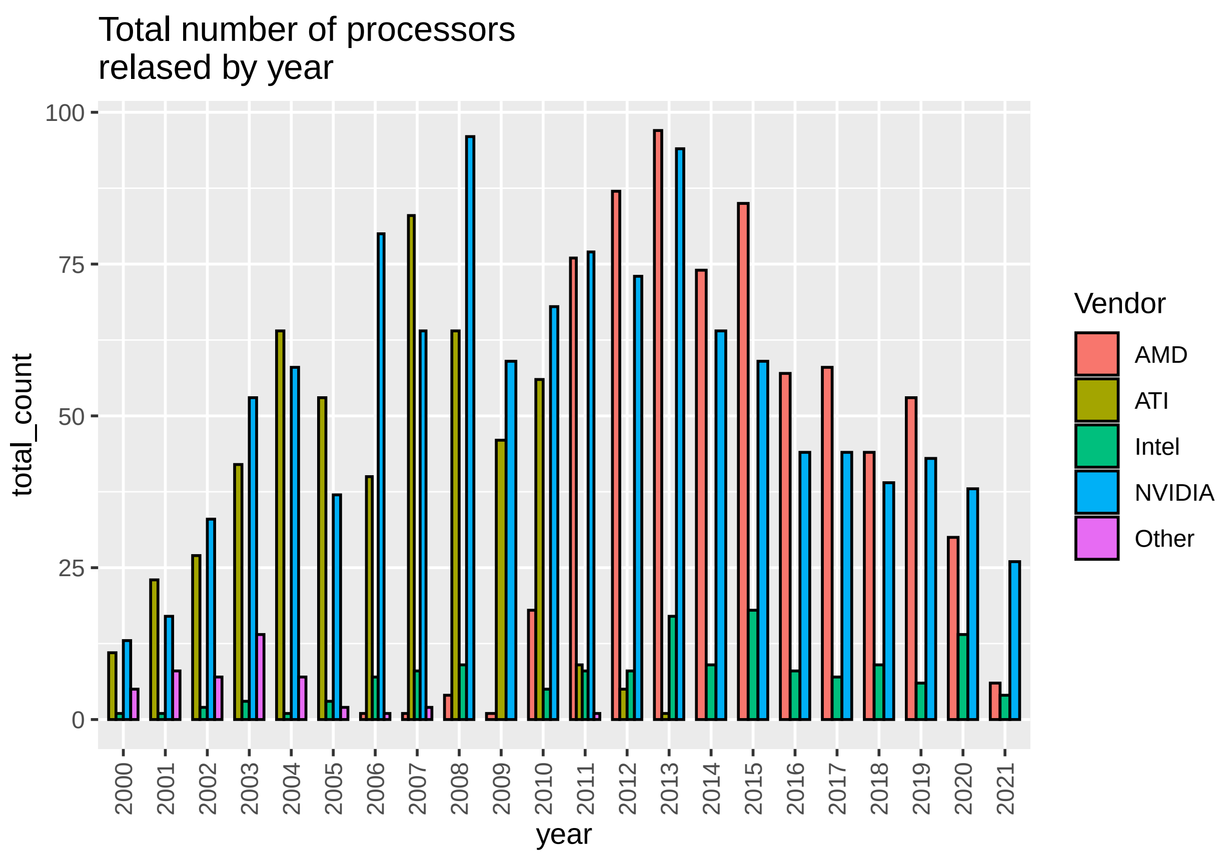



```
vendor_vs_year <- cpu_gpu_data[cpu_gpu_data$Type == "GPU", ] %>%
    group_by(Vendor, year = format(as.Date(na.omit(`Release Date`),
        format = "%Y-%m-%d"), "%Y")) %>%
    summarise(total_count = n())

vendor_vs_year <- na.omit(vendor_vs_year)

vendor_vs_year
```

```
## # A tibble: 82 x 3
## # Groups:   Vendor [5]
##    Vendor year  total_count
```

```
##     <chr>  <chr>       <int>
##   1 AMD    2006            1
##   2 AMD    2007            1
##   3 AMD    2008            4
##   4 AMD    2009            1
##   5 AMD    2010           18
##   6 AMD    2011           76
##   7 AMD    2012           87
##   8 AMD    2013           97
##   9 AMD    2014           74
##  10 AMD    2015           85
## # ... with 72 more rows
```

```r
vendor_vs_year %>%
    ggplot(aes(x = year, y = total_count, group = Vendor, fill = Vendor,
        width = 0.7)) + geom_bar(col = "black", stat = "identity",
    position = "dodge", na.rm = TRUE) + ggtitle("Total number of processors
relased by year") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5,
        hjust = 1))
```



Over the years, we have seen a general increase in release of processors from 2000 to around 2012, then a general decrease from 2012 to 2021. For CPUs, Intel was the leading foundry and vendor for most of the years. For GPUs, TSMC was the top foundry. ATI was the leading GPU vendor 2010, then AMD and NVIDIA took over.

3.2 Part b)

3.2.1 Expected Number of transistors per microchip

```
cpu_data <- cpu_gpu_data[cpu_gpu_data$Type == "CPU", ] %>%
    group_by(`Transistors (million)`, year = format(as.Date(na.omit(`Release Date`),
        format = "%Y-%m-%d"), "%Y"))
curr <- mean(cpu_data[cpu_data$year == 2000, ]$`Transistors (million)`,
    na.rm = TRUE)
CPU_exp_nb_trans <- c(curr)
years <- seq(2000, 2021, by = 1)
for (i in 1:21) {
    curr <- curr * sqrt(2)
    CPU_exp_nb_trans <- append(CPU_exp_nb_trans, curr)
}
```

3.2.1.1 CPU

```
gpu_data <- cpu_gpu_data[cpu_gpu_data$Type == "GPU", ] %>%
    group_by(`Transistors (million)`, year = format(as.Date(na.omit(`Release Date`),
        format = "%Y-%m-%d"), "%Y"))
curr <- mean(gpu_data[gpu_data$year == 2000, ]$`Transistors (million)`,
    na.rm = TRUE)
GPU_exp_nb_trans <- c(curr)
for (i in 1:21) {
    curr <- curr * sqrt(2)
    GPU_exp_nb_trans <- append(GPU_exp_nb_trans, curr)
}
```

3.2.1.2 GPU

3.2.2 Observed number of transistors per microchip

```
cpu_data <- cpu_gpu_data[cpu_gpu_data$Type == "CPU", ] %>%
    group_by(`Transistors (million)`, year = format(as.Date(na.omit(`Release Date`),
        format = "%Y-%m-%d"), "%Y"))
CPU_obs_nb_trans <- c()

for (i in 0:21) {
    curr <- mean(cpu_data[cpu_data$year == 2000 + i, ]$`Transistors (million)`,
        na.rm = TRUE)
    CPU_obs_nb_trans <- append(CPU_obs_nb_trans, curr)
}
```

3.2.2.1 CPU

```
gpu_data <- cpu_gpu_data[cpu_gpu_data$Type == "GPU", ] %>%
    group_by(`Transistors (million)`, year = format(as.Date(na.omit(`Release Date`),
        format = "%Y-%m-%d"), "%Y"))
GPU_obs_nb_trans <- c()

for (i in 0:21) {
```

```r
    curr <- mean(gpu_data[gpu_data$year == 2000 + i, ]$`Transistors (million)`,
        na.rm = TRUE)
    GPU_obs_nb_trans <- append(GPU_obs_nb_trans, curr)
}
```

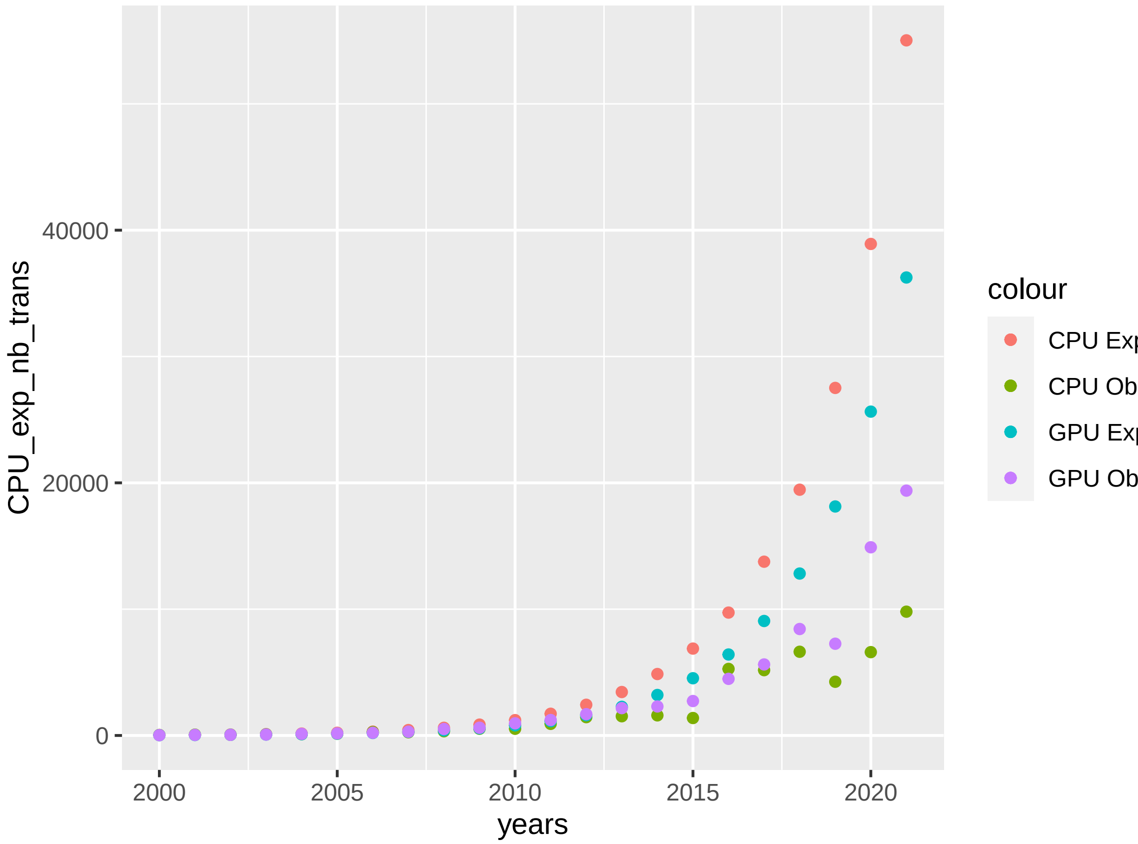
3.2.2.2 GPU

```r
result <- data.frame(years, CPU_exp_nb_trans, CPU_obs_nb_trans,
    GPU_exp_nb_trans, GPU_obs_nb_trans)
result
```

3.2.2.3 All together

```
##    years CPU_exp_nb_trans CPU_obs_nb_trans GPU_exp_nb_trans GPU_obs_nb_trans
## 1   2000         38.00000         38.00000         25.03704         25.03704
## 2   2001         53.74012         65.19767         35.40772         42.53191
## 3   2002         76.00000         51.84615         50.07407         49.77612
## 4   2003        107.48023         89.31507         70.81543         73.44444
## 5   2004        152.00000        104.21324        100.14815        124.62609
## 6   2005        214.96046        137.46575        141.63087        164.60227
## 7   2006        304.00000        269.11864        200.29630        215.09244
## 8   2007        429.92092        252.11111        283.26174        295.70470
## 9   2008        608.00000        335.61290        400.59259        509.34337
## 10  2009        859.84185        533.86087        566.52348        613.83019
## 11  2010       1216.00000        530.23776        801.18519        959.77931
## 12  2011       1719.68369        919.41333       1133.04695       1224.07059
## 13  2012       2432.00000       1450.51852       1602.37037       1685.98266
## 14  2013       3439.36738       1522.49587       2266.09391       2165.05612
## 15  2014       4864.00000       1591.91667       3204.74074       2298.16541
## 16  2015       6878.73477       1382.41667       4532.18782       2727.20588
## 17  2016       9728.00000       5269.85714       6409.48148       4482.54839
## 18  2017      13757.46953       5170.67857       9064.37564       5622.32323
## 19  2018      19456.00000       6626.77273      12818.96296       8429.37500
## 20  2019      27514.93907       4252.48077      18128.75128       7262.18750
## 21  2020      38912.00000       6597.50000      25637.92593      14899.23077
## 22  2021      55029.87814       9800.00000      36257.50256      19382.00000
```

```r
ggplot(data = result, aes(years)) + geom_point(aes(y = CPU_exp_nb_trans,
    color = "CPU Expected")) + geom_point(aes(y = CPU_obs_nb_trans,
    color = "CPU Observed")) + geom_point(aes(y = GPU_exp_nb_trans,
    color = "GPU Expected")) + geom_point(aes(y = GPU_obs_nb_trans,
    color = "GPU Observed"))
```



3.2.2.4 Graphically

Hence, from the table and plot above, we can say that Moore's Law is no longer valid.