1. Part A

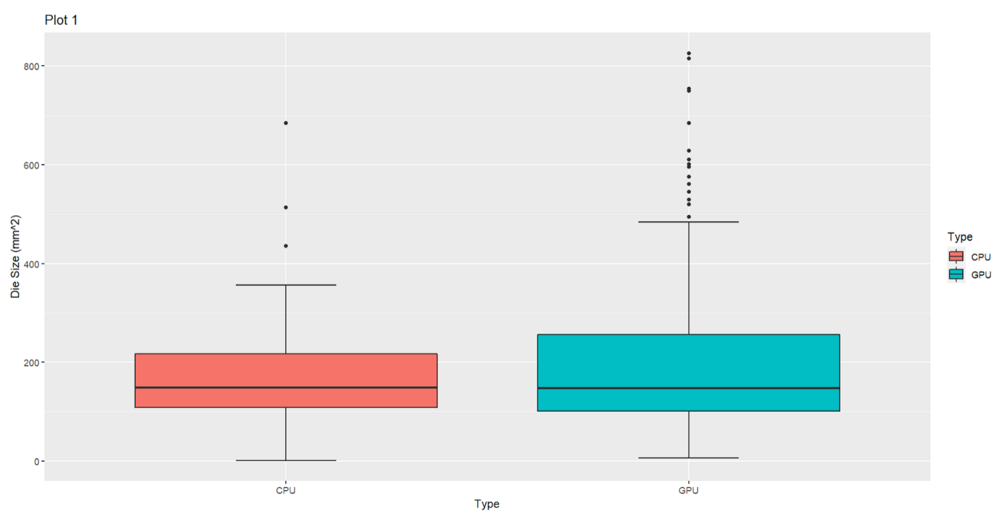
Die size:

- The distributions for GPUs and CPUs are similar in that they are both skewed right and have similar central locations. Both distributions have outliers, but GPUs have more outliers. The data for GPUs are more spread out. Both groups have missing observations.

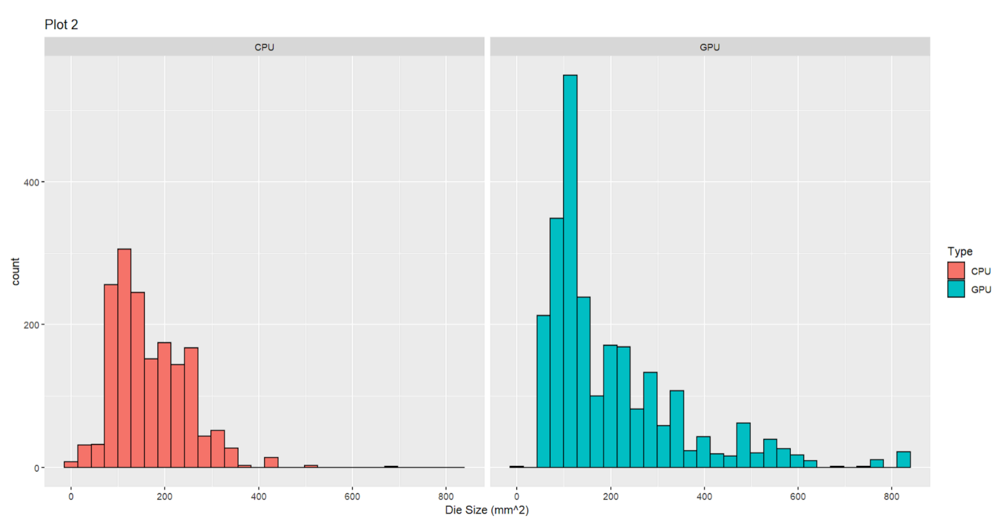
```
cpu_gpu_data %>% group_by(Type) %>% select(Type, `Die Size (mm^2)`) %>% summarise_all(list(Avg=mean,
Med=median, Q25=~quantile(.,probs=c(0.25), na.rm = TRUE), Q75=~quantile(.,probs=c(0.75), na.rm = TRUE),
StD=sd, IQR=IQR), na.rm = TRUE) %>% pivot_longer(!Type, names_to="Die_Size") %>%
pivot_wider(id_cols=Die_Size, names_from=Type)
```

```
# A tibble: 6 x 3
  Die_Size   CPU   GPU
  <chr>    <dbl> <dbl>
1 Avg      167.   203.
2 Med      149    148
3 Q25      109    101
4 Q75      217    256
5 StD       79.7  148.
6 IQR      108    155
```

```
ggplot(cpu_gpu_data, aes(x=Type, y=`Die Size (mm^2)`, fill=Type)) + stat_boxplot(geom="errorbar",
width=0.25) + geom_boxplot() + labs(y="Die Size (mm^2)", title="Plot 1")
```




```
ggplot(cpu_gpu_data, aes(x=`Die Size (mm^2)`, group=Type, fill=Type)) + geom_histogram(col="black") +
labs(title="Plot 2") + facet_wrap(~Type)
```



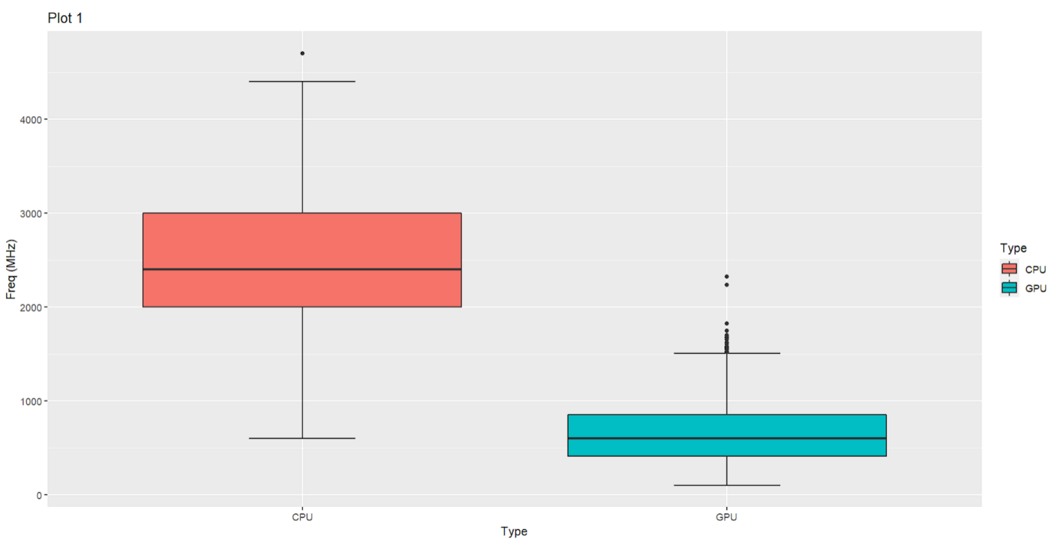
Frequency:

- The distribution for CPUs is approximately symmetric, while that of GPUs is slightly skewed right. The central locations for CPUs are also larger, and the data is more spread out than that of GPUs. Both distributions have outliers, but GPUs have more outliers. There are no missing observations.

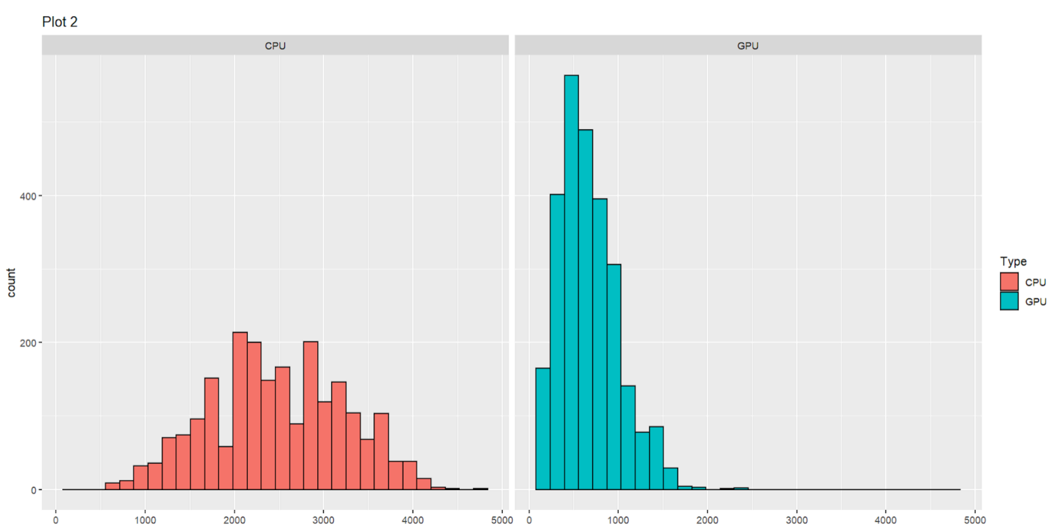
```
cpu_gpu_data %>% group_by(Type) %>% select(Type, `Freq (MHz)`)%>% summarise_all(list(Avg=mean,
Med=median, Q25=~quantile(.,probs=c(0.25), na.rm = TRUE), Q75=~quantile(.,probs=c(0.75), na.rm = TRUE),
StD=sd, IQR=IQR), na.rm = TRUE) %>% pivot_longer(!Type, names_to="Freq") %>% pivot_wider(id_cols=Freq,
names_from=Type)
```

```
# A tibble: 6 x 3
  Freq    CPU   GPU
  <chr> <dbl> <dbl>
1 Avg   2482.  663.
2 Med   2400   600
3 Q25   2000   412
4 Q75   3000   850
5 StD    755.  331.
6 IQR   1000   438
```

```
ggplot(cpu_gpu_data, aes(x=Type, y=`Freq (MHz)`, fill=Type)) + stat_boxplot(geom="errorbar",
width=0.25) + geom_boxplot() + labs(y="Freq (MHz)", title="Plot 1")
```




```
ggplot(cpu_gpu_data, aes(x=`Freq (MHz)`, group=Type, fill=Type)) + geom_histogram(col="black") +
labs(title="Plot 2") + facet_wrap(~Type)
```



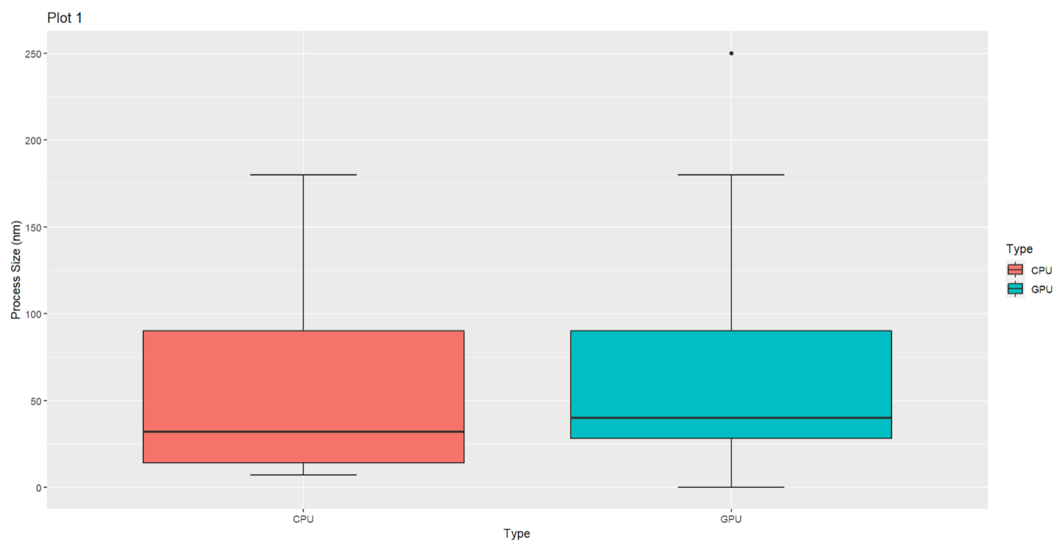
Process size:

- The distributions for GPUs and CPUs are similar in that they are both slightly skewed right and have similar central locations and spread. GPUs have one outlier while CPUs have none. There are no missing observations.

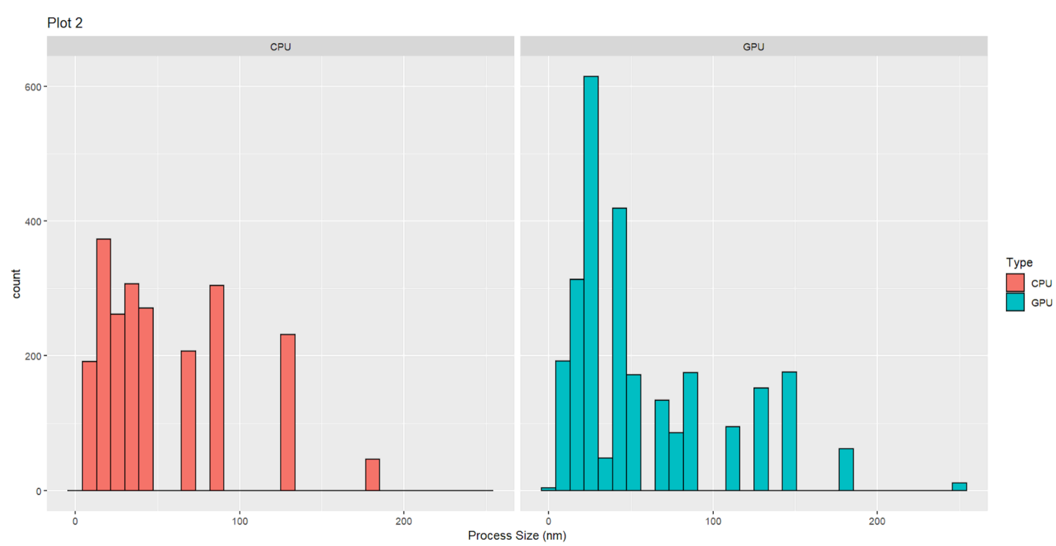
```
cpu_gpu_data %>% group_by(Type) %>% select(Type, `Process Size (nm)`)%>% summarise_all(list(Avg=mean,
Med=median, Q25=~quantile(.,probs=c(0.25), na.rm = TRUE), Q75=~quantile(.,probs=c(0.75), na.rm = TRUE),
StD=sd, IQR=IQR), na.rm = TRUE) %>% pivot_longer(!Type, names_to="Process_Size") %>%
pivot_wider(id_cols=Process_Size, names_from=Type)
```

```
# A tibble: 6 x 3
  Process_Size   CPU   GPU
  <chr>        <dbl> <dbl>
1 Avg           52.0  57.7
2 Med           32    40
3 Q25           14    28
4 Q75           90    90
5 StD           42.1  47.1
6 IQR           76    62
```

```
ggplot(cpu_gpu_data, aes(x=Type, y=`Process Size (nm)`, fill=Type)) + stat_boxplot(geom="errorbar",
width=0.25) + geom_boxplot() + labs(y="Process Size (nm)", title="Plot 1")
```




```
ggplot(cpu_gpu_data, aes(x=`Process Size (nm)`, group=Type, fill=Type)) + geom_histogram(col="black") +
labs(title="Plot 2") + facet_wrap(~Type)
```



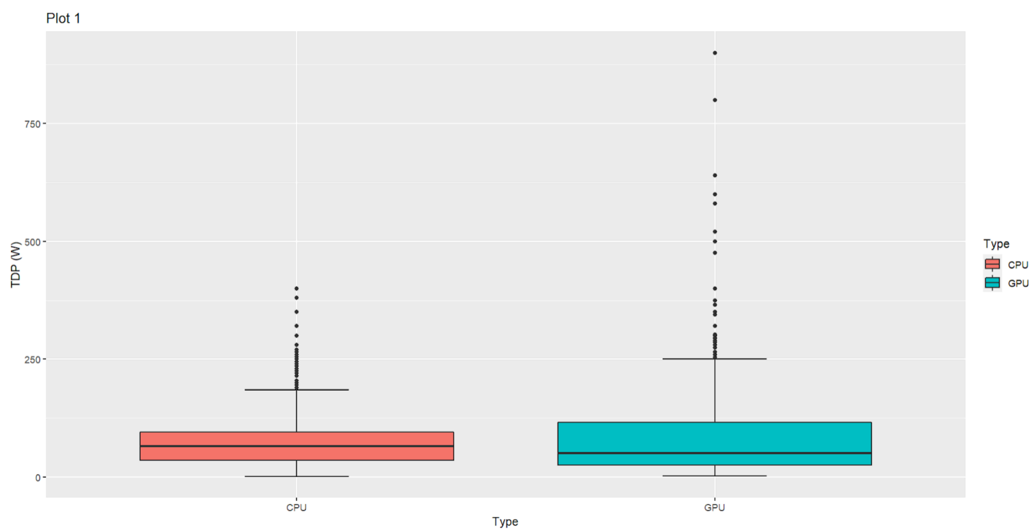
Thermal design power:

- The distributions for GPUs and CPUs are similar in that they are both skewed right and have similar central locations. Both distributions have outliers, but the outliers for GPUs are more spread out. As a result, the data for GPUs is more spread out. There are missing observations for GPUs.

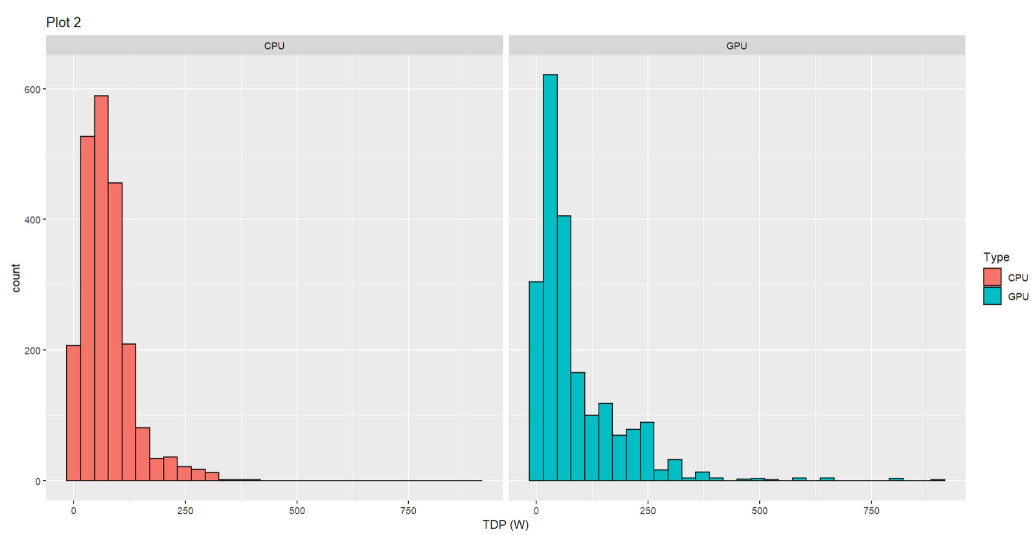
```
cpu_gpu_data %>% group_by(Type) %>% select(Type, `TDP (W)`)%>% summarise_all(list(Avg=mean, Med=median,
Q25=~quantile(.,probs=c(0.25), na.rm = TRUE), Q75=~quantile(.,probs=c(0.75), na.rm = TRUE), StD=sd,
IQR=IQR), na.rm = TRUE) %>% pivot_longer(!Type, names_to="TDP") %>% pivot_wider(id_cols=TDP,
names_from=Type)
```

```
# A tibble: 6 x 3
  TDP     CPU   GPU
  <chr> <dbl> <dbl>
1 Avg    75.4  87.8
2 Med    65    50
3 Q25    35    25
4 Q75    95   116.
5 StD    54.4  94.8
6 IQR    60    91.5
```

```
ggplot(cpu_gpu_data, aes(x=Type, y=`TDP (W)`, fill=Type)) + stat_boxplot(geom="errorbar", width=0.25) +
geom_boxplot() + labs(y="TDP (W)", title="Plot 1")
```




```
ggplot(cpu_gpu_data, aes(x=`TDP (W)`, group=Type, fill=Type)) + geom_histogram(col="black") +
labs(title="Plot 2") + facet_wrap(~Type)
```



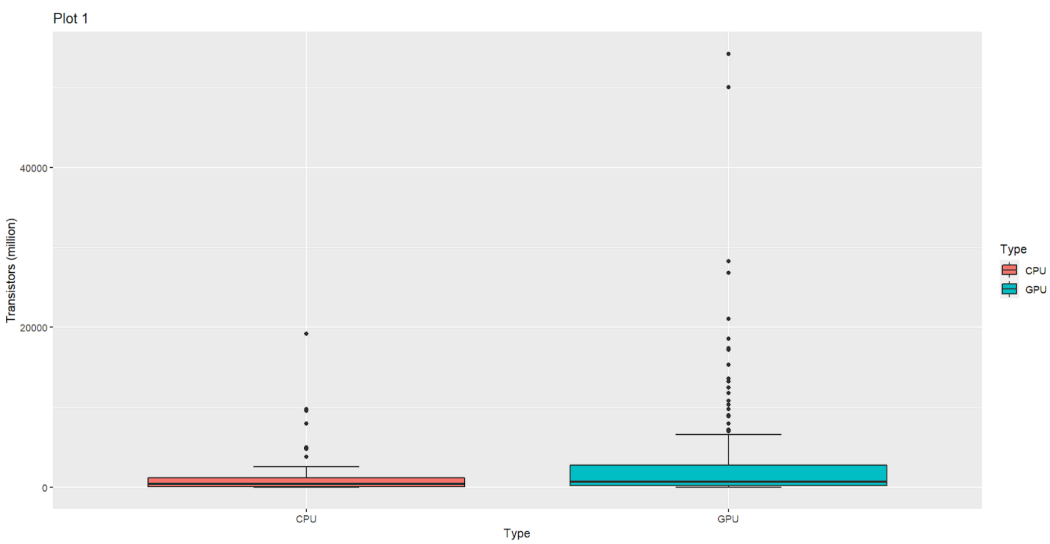
Transistors:

- The distributions for GPUs and CPUs are similar because they are both skewed right and have outliers. The central location for GPUs are larger, and the data is more spread out than that of CPUs. Both groups have missing observations.

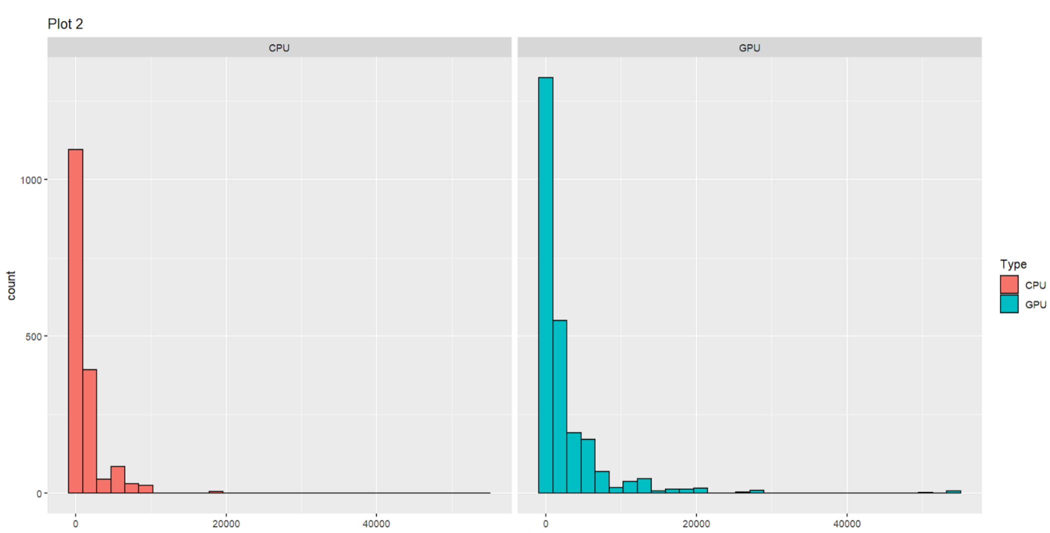
```
cpu_gpu_data %>% group_by(Type) %>% select(Type, `Transistors (million)`)%>%
summarise_all(list(Avg=mean, Med=median, Q25=~quantile(.,probs=c(0.25), na.rm = TRUE),
Q75=~quantile(.,probs=c(0.75), na.rm = TRUE), StD=sd, IQR=IQR), na.rm = TRUE) %>% pivot_longer(!Type,
names_to="Transistors") %>% pivot_wider(id_cols=Transistors, names_from=Type)
```

```
# A tibble: 6 x 3
  Transistors   CPU   GPU
  <chr>       <dbl> <dbl>
1 Avg         1156. 2455.
2 Med          410   716
3 Q25          114   210
4 Q75         1200  2800
5 StD         2037. 4896.
6 IQR         1086  2590
```

```
ggplot(cpu_gpu_data, aes(x=Type, y=`Transistors (million)`, fill=Type)) +
stat_boxplot(geom="errorbar", width=0.25) + geom_boxplot() + labs(y="Transistors (million)",
title="Plot 1")
```




```
ggplot(cpu_gpu_data, aes(x=`Transistors (million)`, group=Type, fill=Type)) +
geom_histogram(col="black") + labs(title="Plot 2") + facet_wrap(~Type)
```



1. Part B

There are some strong associations between the number of processors released by the vendors and foundries. The GF foundry exclusively releases semiconductors to the AMD vendor, as shown by the 1 in the numerical summary (symbolizing a full proportion) and the solid bar in Plot 1. The Intel foundry exclusively releases semiconductors to their Intel vendor. The Samsung foundry releases a large proportion to the NVIDIA vendor, while the other foundries are more mixed. On the other hand, the Intel vendor releases semiconductors almost exclusively from their Intel foundry. The ATI and NVIDIA vendor releases a large proportion from the TSMC foundry, while the other vendors are more mixed.

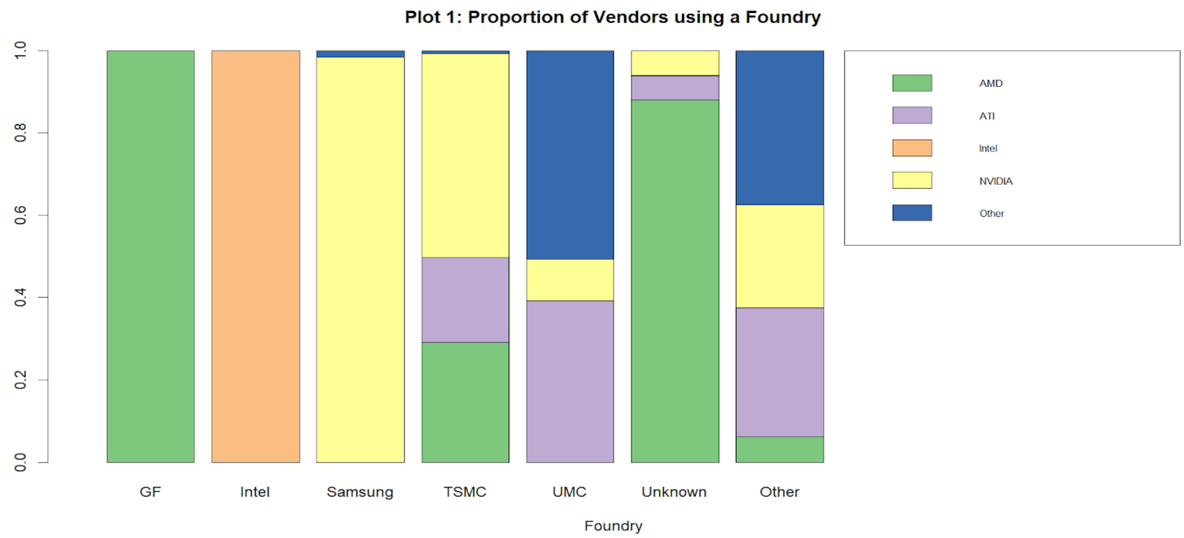
- Numerical summaries:

```
cpu_gpu_data2 <- cpu_gpu_data %>%  mutate(Foundry_Lump=fct_lump(Foundry, 6))
cpu_gpu_array <- xtabs(~Vendor+Foundry_Lump+Type, data=cpu_gpu_data2)

column_props <- apply(cpu_gpu_array, c("Vendor", "Foundry_Lump"), sum) %>% prop.table(., c(2))
column_props

        Foundry_Lump
Vendor    GF Intel    Samsung        TSMC       UMC     Unknown  Other
  AMD      1     0 0.00000000 0.291092746 0.0000000 0.879907621 0.0625
  ATI      0     0 0.00000000 0.206152433 0.3924051 0.057736721 0.3125
  Intel    0     1 0.00000000 0.000000000 0.0000000 0.002309469 0.0000
  NVIDIA   0     0 0.98333333 0.494949495 0.1012658 0.060046189 0.2500
  Other    0     0 0.01666667 0.007805326 0.5063291 0.000000000 0.3750
```

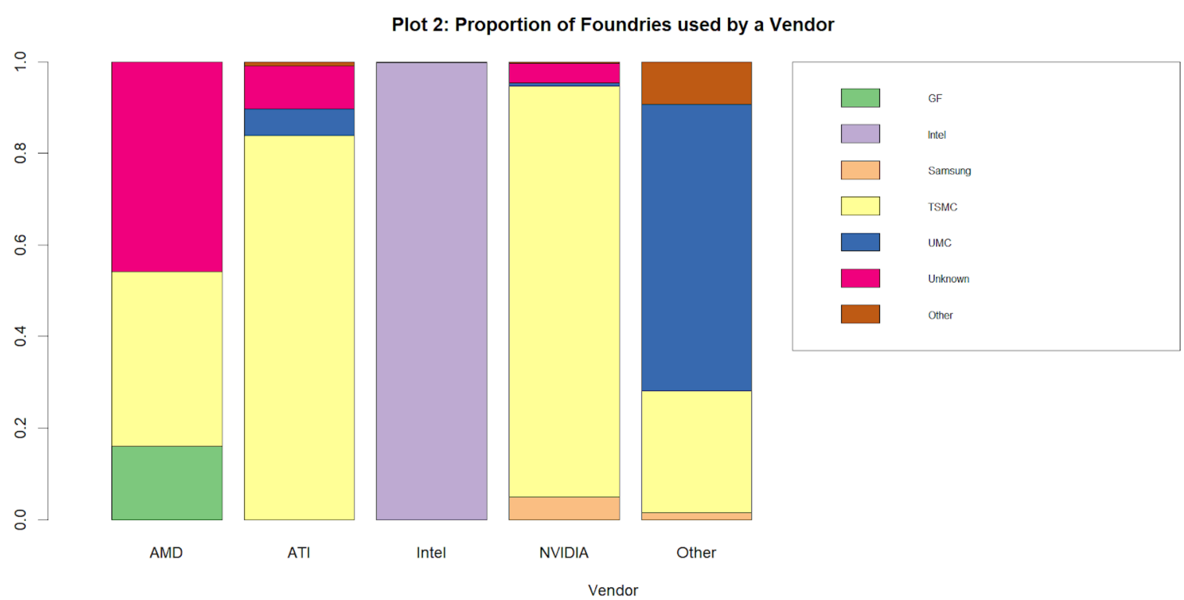
```
cpu_gpu_array2 <- xtabs(~Foundry_Lump+Vendor+Type, data=cpu_gpu_data2)

column_props <- apply(cpu_gpu_array2, c("Foundry_Lump", "Vendor"), sum) %>% prop.table(., c(2))
column_props

            Vendor
Foundry_Lump          AMD         ATI       Intel      NVIDIA    Other
     GF      0.1594464501 0.000000000 0.000000000 0.000000000 0.000000
     Intel   0.0000000000 0.000000000 0.998563218 0.000000000 0.000000
     Samsung 0.0000000000 0.000000000 0.000000000 0.049125729 0.015625
     TSMC    0.3814681107 0.839252336 0.000000000 0.897585346 0.265625
     UMC     0.0000000000 0.057943925 0.000000000 0.006661116 0.625000
     Unknown 0.4584837545 0.093457944 0.001436782 0.043297252 0.000000
     Other   0.0006016847 0.009345794 0.000000000 0.003330558 0.093750
```

- Graphical summaries:

```
colors <- c(brewer.pal(n=5, name="Accent"))
myplot <- barplot(column_props, col=colors, xlim=c(0, 12), main="Plot 1: Proportion of Vendors using a Foundry", xlab="Foundry")
legend("topright", legend = c("AMD", "ATI", "Intel", "NVIDIA", "Other"), fill=colors, cex=0.65)
```




```
colors <- c(brewer.pal(n=7, name="Accent"))
myplot <- barplot(column_props, col=colors, xlim=c(0, 9.5), main="Plot 2: Proportion of Foundries used by a Vendor", xlab="Vendor")
legend("topright", legend = c("GF", "Intel", "Samsung", "TSMC", "UMC", "Unknown", "Other"), fill=colors, cex=0.65)
```



The association does not seem to depend on whether they are CPUs or GPUs. For both groups, the GF foundry exclusively releases semiconductors to the AMD vendor, and the Intel foundry exclusively releases to their Intel vendor. The Samsung foundry does not apply to CPUs (nor do the UMC or Other foundries), but it releases a large proportion to the NVIDIA vendor for GPUs. On the other hand, the Intel vendor releases semiconductors almost exclusively from their Intel foundry for both CPUs and GPUs. The ATI and NVIDIA vendor does not apply to CPUs, but they release a large proportion from the TSMC foundry for GPUs.

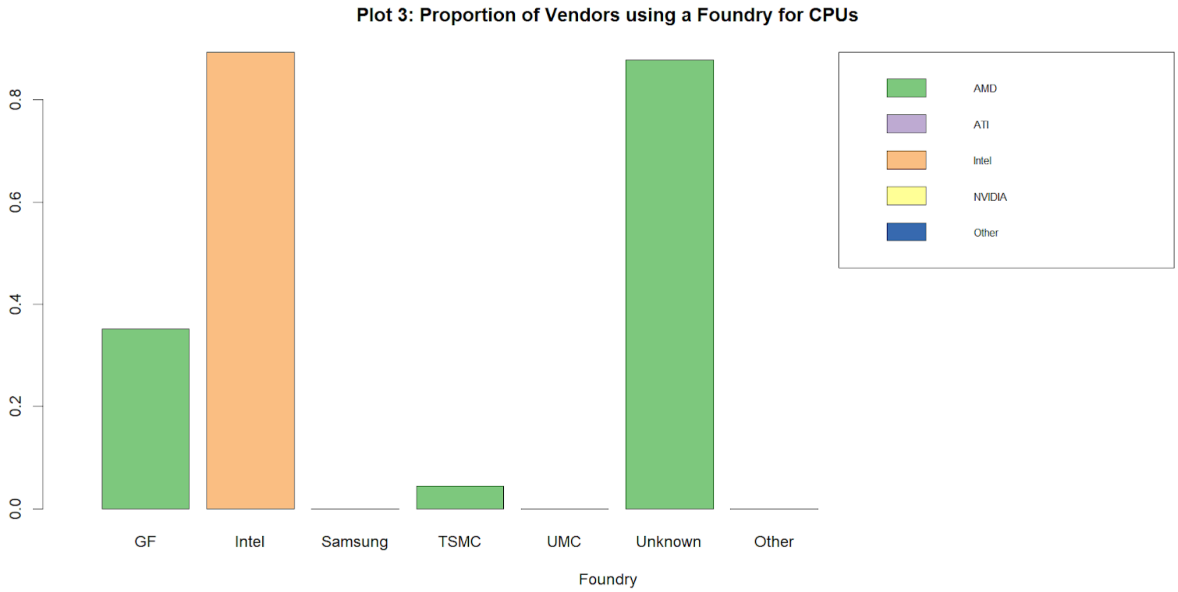
- Numerical summaries: CPU vs. GPU

```
column_props <- apply(cpu_gpu_array, c("Vendor", "Foundry_Lump", "Type"), sum) %>% prop.table(., c(2))
column_props

, , Type = CPU

        Foundry_Lump
Vendor          GF     Intel Samsung       TSMC UMC   Unknown Other
  AMD    0.3509434 0.0000000       0 0.04453627   0 0.8775982     0
  ATI    0.0000000 0.0000000       0 0.00000000   0 0.0000000     0
  Intel  0.0000000 0.8935252       0 0.00000000   0 0.0000000     0
  NVIDIA 0.0000000 0.0000000       0 0.00000000   0 0.0000000     0
  Other  0.0000000 0.0000000       0 0.00000000   0 0.0000000     0

, , Type = GPU

        Foundry_Lump
Vendor          GF     Intel    Samsung        TSMC       UMC     Unknown  Other
  AMD    0.6490566 0.0000000 0.00000000 0.246556474 0.0000000 0.002309469 0.0625
  ATI    0.0000000 0.0000000 0.00000000 0.206152433 0.3924051 0.057736721 0.3125
  Intel  0.0000000 0.1064748 0.00000000 0.000000000 0.0000000 0.002309469 0.0000
  NVIDIA 0.0000000 0.0000000 0.98333333 0.494949495 0.1012658 0.060046189 0.2500
  Other  0.0000000 0.0000000 0.01666667 0.007805326 0.5063291 0.000000000 0.3750
```

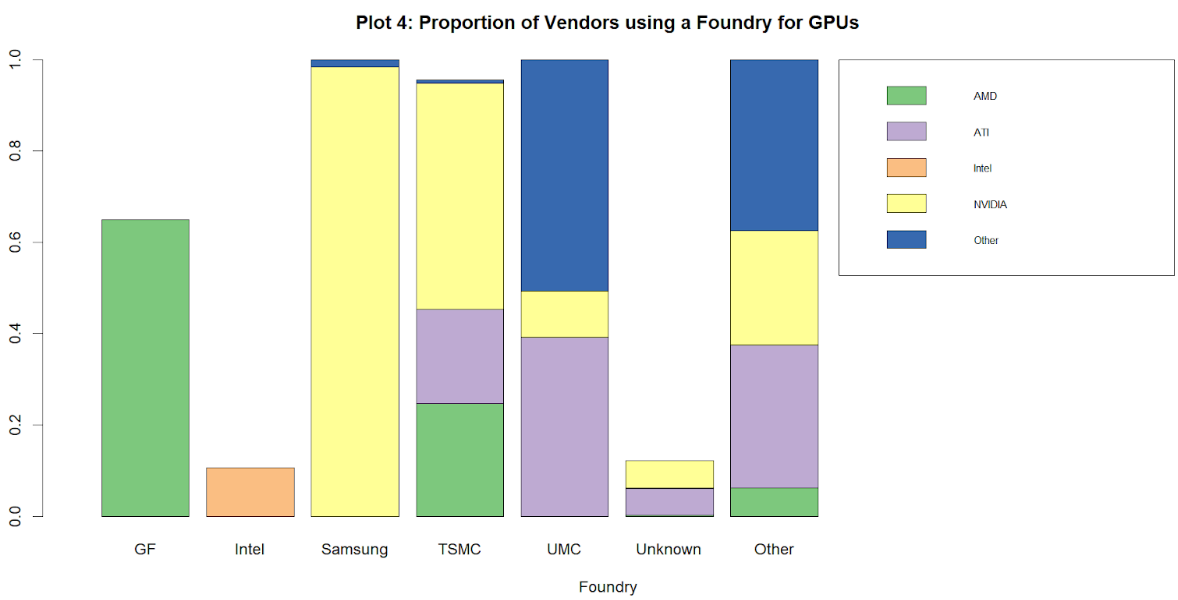
```
column_props <- apply(cpu_gpu_array2, c("Foundry_Lump", "Vendor", "Type"), sum) %>% prop.table(., c(2))
column_props

, , Type = CPU

            Vendor
Foundry_Lump        AMD ATI     Intel NVIDIA Other
     GF      0.05595668   0 0.0000000      0     0
     Intel   0.00000000   0 0.8922414      0     0
     Samsung 0.00000000   0 0.0000000      0     0
     TSMC    0.05836342   0 0.0000000      0     0
     UMC     0.00000000   0 0.0000000      0     0
     Unknown 0.45728039   0 0.0000000      0     0
     Other   0.00000000   0 0.0000000      0     0

, , Type = GPU

            Vendor
Foundry_Lump          AMD         ATI       Intel      NVIDIA    Other
     GF      0.1034897714 0.000000000 0.000000000 0.000000000 0.000000
     Intel   0.0000000000 0.000000000 0.106321839 0.000000000 0.000000
     Samsung 0.0000000000 0.000000000 0.000000000 0.049125729 0.015625
     TSMC    0.3231046931 0.839252336 0.000000000 0.897585346 0.265625
     UMC     0.0000000000 0.057943925 0.000000000 0.006661116 0.625000
     Unknown 0.0012033694 0.093457944 0.001436782 0.043297252 0.000000
     Other   0.0006016847 0.009345794 0.000000000 0.003330558 0.093750
```

- Graphical summaries: CPU vs. GPU

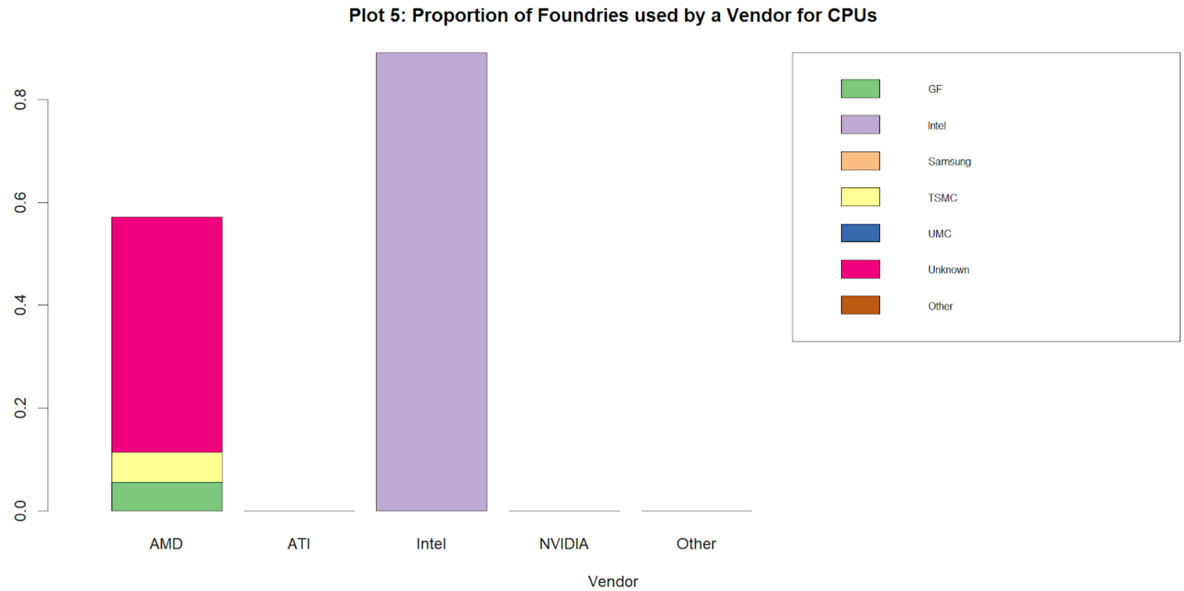
```
colors <- c(brewer.pal(n=5, name="Accent"))
myplot1 <- barplot(column_props[,,1], col=colors, xlim=c(0, 12), main="Plot 3: Proportion of Vendors using a Foundry for CPUs", xlab="Foundry")
legend("topright", legend = c("AMD", "ATI", "Intel", "NVIDIA", "Other"), fill=colors, cex=0.65)
```



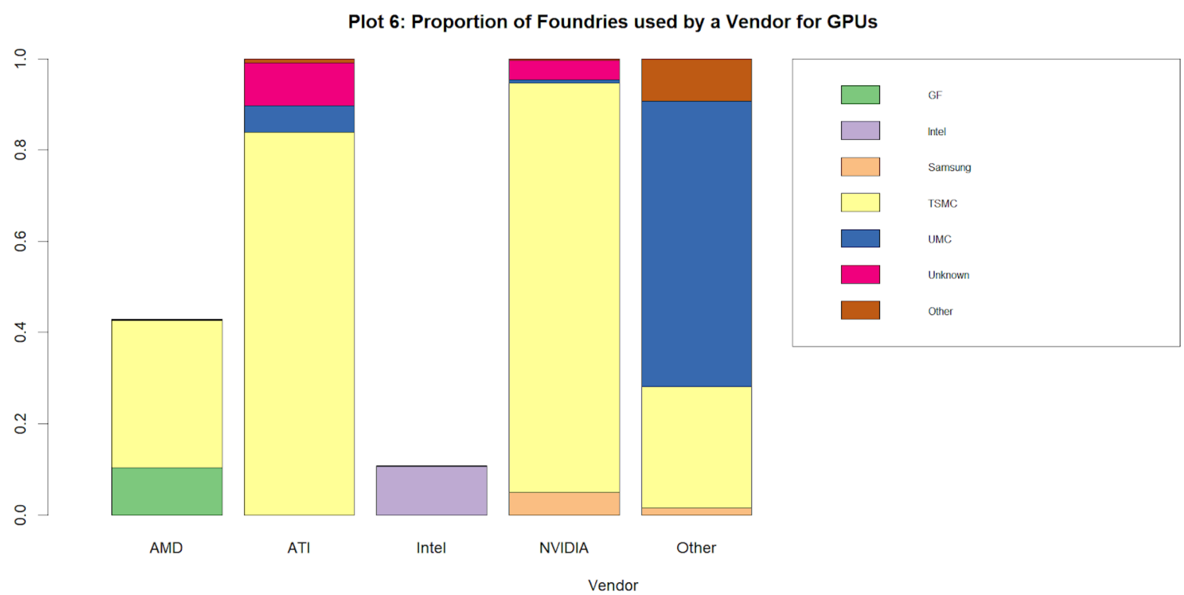

```
myplot2 <- barplot(column_props[,,2], col=colors, xlim=c(0, 12), main="Plot 4: Proportion of Vendors using a Foundry for GPUs", xlab="Foundry")
legend("topright", legend = c("AMD", "ATI", "Intel", "NVIDIA", "Other"), fill=colors, cex=0.65)
```



```
colors <- c(brewer.pal(n=7, name="Accent"))
myplot1 <- barplot(column_props[,,1], col=colors, xlim=c(0, 9.5), main="Plot 5: Proportion of Foundries
used by a Vendor for CPUs", xlab="Vendor")
legend("topright", legend = c("GF", "Intel", "Samsung", "TSMC", "UMC", "Unknown", "Other"), fill=colors,
cex=0.65)
```




```
myplot2 <- barplot(column_props[,,2], col=colors, xlim=c(0, 9.5), main="Plot 6: Proportion of Foundries
used by a Vendor for GPUs", xlab="Vendor")
legend("topright", legend = c("GF", "Intel", "Samsung", "TSMC", "UMC", "Unknown", "Other"), fill=colors,
cex=0.65)
```

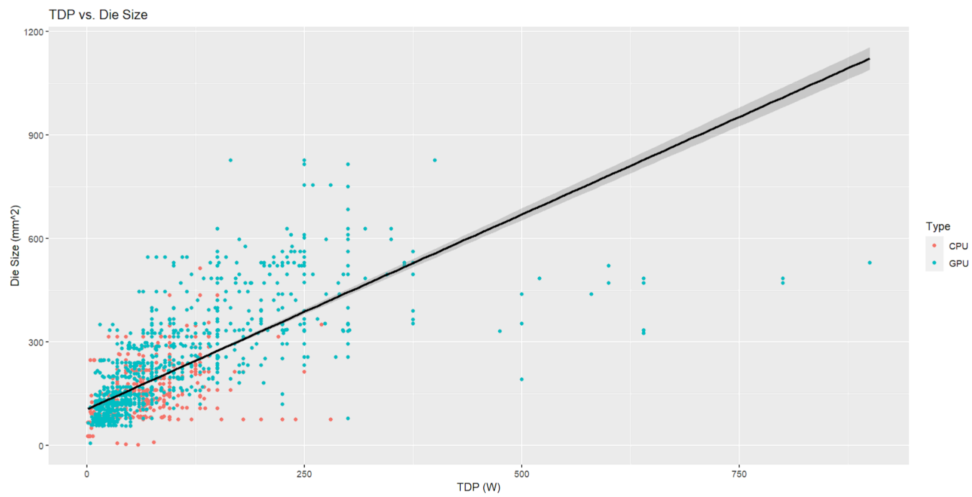


1. Part C

The association between Die Size and Thermal Design Power depends on Type. The correlation coefficient for CPUs is 0.411, so it represents a positive and moderate relationship. The correlation coefficient for GPUs is 0.731, so the graph has a steeper trajectory. Without Type, the correlation coefficient comes at an in-between number.

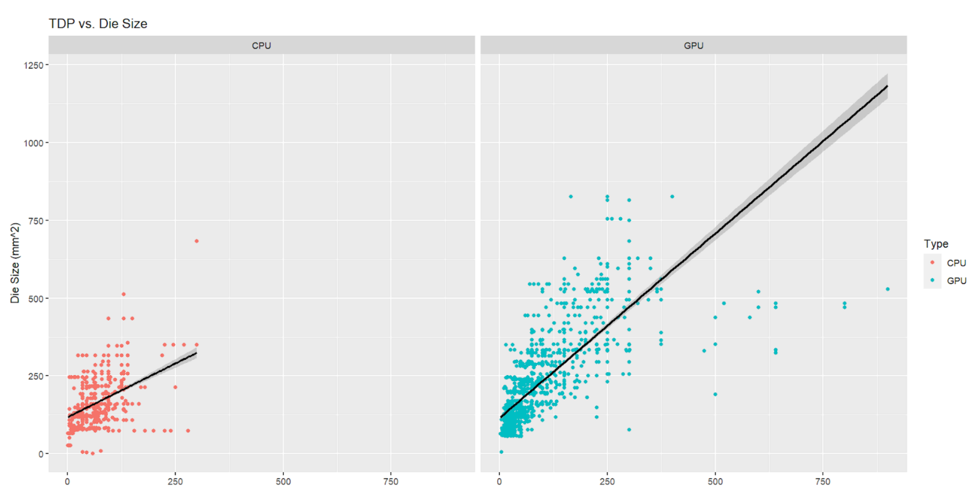
- Correlation without Type:

```
cpu_gpu_data %>% drop_na(`TDP (W)`, `Die Size (mm^2)`) %>% summarise(Correlation=cor(`TDP (W)`, `Die
Size (mm^2)`))

# A tibble: 1 x 1
  Correlation
        <dbl>
1       0.681
```



- Correlation with Type:

```
cpu_gpu_data %>% drop_na(`TDP (W)`, `Die Size (mm^2)`) %>% group_by(Type) %>%
summarise(Correlation=cor(`TDP (W)`, `Die Size (mm^2)`))

# A tibble: 2 x 2
  Type  Correlation
  <chr>       <dbl>
1 CPU         0.411
2 GPU         0.731

ggplot(cpu_gpu_data, aes(x=`TDP (W)`, y=`Die Size (mm^2)`, col=Type)) + geom_point() + facet_wrap(~Type)
+ labs(x="TDP (W)", y="Die Size (mm^2)", title="TDP vs. Die Size") + geom_smooth(method="lm",
col="black")
```



2. Part A

Intel and TSMC consistently produced processors over the years 2000-2021, with both increasing to produce the most processors in the year 2013 and decreasing thereafter. Other foundries produced processors at different years. UMC stopped producing after 2009, while Samsung and GF only started after 2011 and 2014 respectively–seeming to take the place of UMC and other foundries belonging in the Other or Unknown categories.

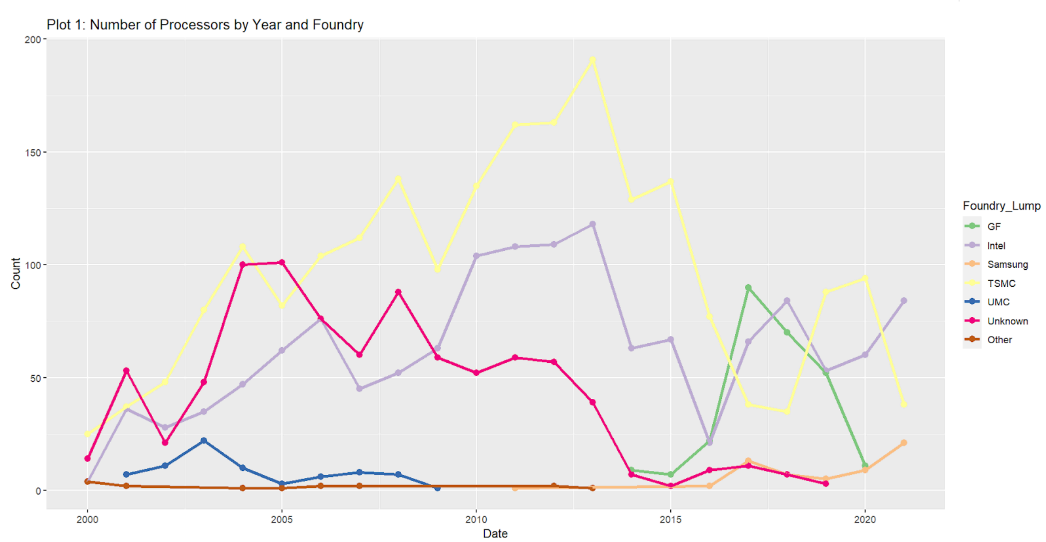
- Number of processors by year and foundry: Numerical summary

```
cpu_gpu_data2 <- cpu_gpu_data %>%  mutate(Foundry_Lump=fct_lump(Foundry, 6)) %>% filter(`Release
Date`!="NaT")
dates <- as.Date(cpu_gpu_data2$`Release Date`, "%m/%d/%Y")
cpu_gpu_data2 <- cpu_gpu_data2 %>% mutate(FirstofYear=floor_date(dates, unit="year"))
foundrybyYear <- cpu_gpu_data2 %>% group_by(FirstofYear, Foundry_Lump) %>% summarise(count=n())

foundrybyYear2 <- foundrybyYear %>% pivot_wider(., id_cols="FirstofYear", names_from="Foundry_Lump",
values_from="count")
foundrybyYear2[is.na(foundrybyYear2)] <- 0
foundrybyYear2 %>% print(n=22)
```

```
# A tibble: 22 x 8
# Groups:   FirstofYear [22]
   FirstofYear Intel  TSMC Unknown Other   UMC Samsung    GF
   <date>      <int> <int>   <int> <int> <int>   <int> <int>
 1 2000-01-01      4    25      14     4     0       0     0
 2 2001-01-01     36    37      53     2     7       0     0
 3 2002-01-01     28    48      21     0    11       0     0
 4 2003-01-01     35    80      48     0    22       0     0
 5 2004-01-01     47   108     100     1    10       0     0
 6 2005-01-01     62    82     101     1     3       0     0
 7 2006-01-01     76   104      76     2     6       0     0
 8 2007-01-01     45   112      60     2     8       0     0
 9 2008-01-01     52   138      88     0     7       0     0
10 2009-01-01     63    98      59     0     1       0     0
11 2010-01-01    104   135      52     0     0       0     0
12 2011-01-01    108   162      59     0     0       1     0
13 2012-01-01    109   163      57     2     0       0     0
14 2013-01-01    118   191      39     1     0       0     0
15 2014-01-01     63   129       7     0     0       0     9
16 2015-01-01     67   137       2     0     0       0     7
17 2016-01-01     21    77       9     0     0       2    22
18 2017-01-01     66    38      11     0     0      13    90
19 2018-01-01     84    35       7     0     0       7    70
20 2019-01-01     53    88       3     0     0       5    52
21 2020-01-01     60    94       0     0     0       9    11
22 2021-01-01     84    38       0     0     0      21     0
```

- Number of processors by year and foundry: Graphical summary

```
ggplot(foundrybyYear, aes(x=FirstofYear, y=count, col=Foundry_Lump)) + geom_point(size=2.5) +
geom_line(size=1.25) + labs(x="Date", y="Count", title="Plot 1: Number of Processors by Year and
Foundry") + scale_color_brewer(palette="Accent")
```



AMD, Intel, and NVIDIA consistently produced processors over the years 2000-2021, with all having certain years of increased production and certain years of low. AMD most notably peaked in 2012, while Intel peaked in 2013 and NVIDIA less dramatically peaked in 2008 and 2013. Other vendors produced processors at different years. ATI stopped producing after 2013 and vendors in the Other category stopped producing after 2011, perhaps due to the increased production of the aforementioned three vendors.

- Number of processors by year and vendor: Numerical summary

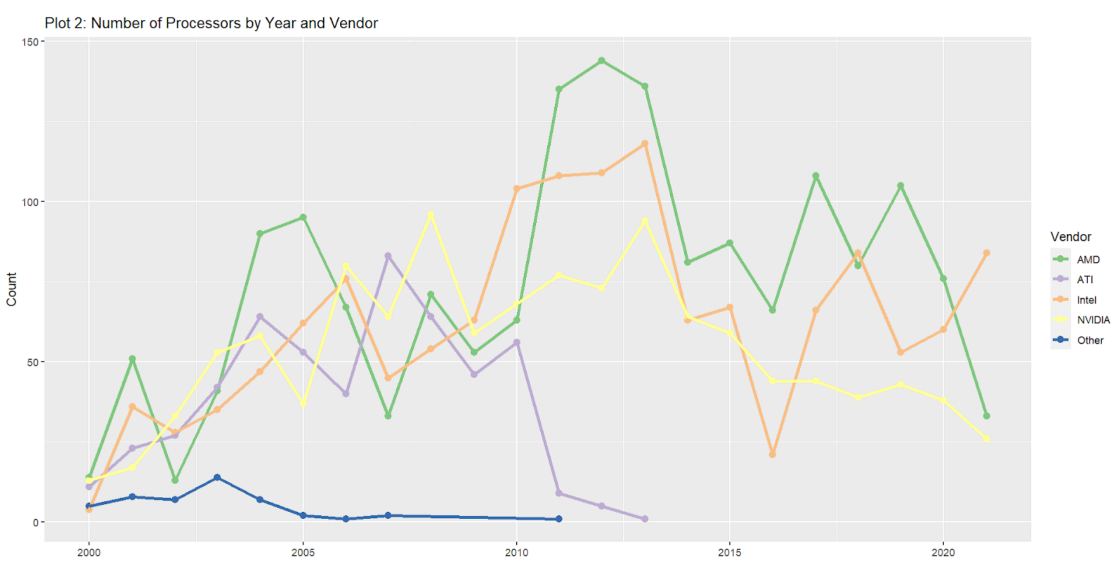
```
cpu_gpu_data2 <- cpu_gpu_data %>% filter(`Release Date`!="NaT")
dates <- as.Date(cpu_gpu_data2$`Release Date`, "%m/%d/%Y")
cpu_gpu_data2 <- cpu_gpu_data2 %>% mutate(FirstofYear=floor_date(dates, unit="year"))
vendorbyYear <- cpu_gpu_data2 %>% group_by(FirstofYear, Vendor) %>% summarise(count=n())
```

```
vendorbyYear2 <- vendorbyYear %>% pivot_wider(., id_cols="FirstofYear", names_from="Vendor",
values_from="count")
vendorbyYear2[is.na(vendorbyYear2)] <- 0
vendorbyYear2 %>% print(n=22)
```

```
# A tibble: 22 x 6
# Groups:   FirstofYear [22]
   FirstofYear   AMD   ATI Intel NVIDIA Other
   <date>      <int> <int> <int>  <int> <int>
 1 2000-01-01     14    11     4     13     5
 2 2001-01-01     51    23    36     17     8
 3 2002-01-01     13    27    28     33     7
 4 2003-01-01     41    42    35     53    14
 5 2004-01-01     90    64    47     58     7
 6 2005-01-01     95    53    62     37     2
 7 2006-01-01     67    40    76     80     1
 8 2007-01-01     33    83    45     64     2
 9 2008-01-01     71    64    54     96     0
10 2009-01-01     53    46    63     59     0
11 2010-01-01     63    56   104     68     0
12 2011-01-01    135     9   108     77     1
13 2012-01-01    144     5   109     73     0
14 2013-01-01    136     1   118     94     0
15 2014-01-01     81     0    63     64     0
16 2015-01-01     87     0    67     59     0
17 2016-01-01     66     0    21     44     0
18 2017-01-01    108     0    66     44     0
19 2018-01-01     80     0    84     39     0
20 2019-01-01    105     0    53     43     0
21 2020-01-01     76     0    60     38     0
22 2021-01-01     33     0    84     26     0
```

- Number of processors by year and vendor: Graphical summary

```
ggplot(vendorbyYear, aes(x=FirstofYear, y=count, col=Vendor)) + geom_point(size=2.5) +
geom_line(size=1.25) + labs(x="Date", y="Count", title="Plot 2: Number of Processors by Year and
Vendor") + scale_color_brewer(palette="Accent")
```



2. Part B

Moore's Law holds true. If I test the correlation between my expected transistor calculations and the actual transistor numbers, the correlation coefficient rounds to 0.92 and 1.00 for CPUs and GPUs respectively. This means the strength of the relationship is very strong, and we can see both numerically and graphically that these numbers and distribution are very similar.

```
> cor(MytransistorsbyYear2$CPU, transistorsbyYear$CPU)
[1] 0.9178498
> cor(MytransistorsbyYear2$GPU, transistorsbyYear$GPU)
[1] 0.9967188
```

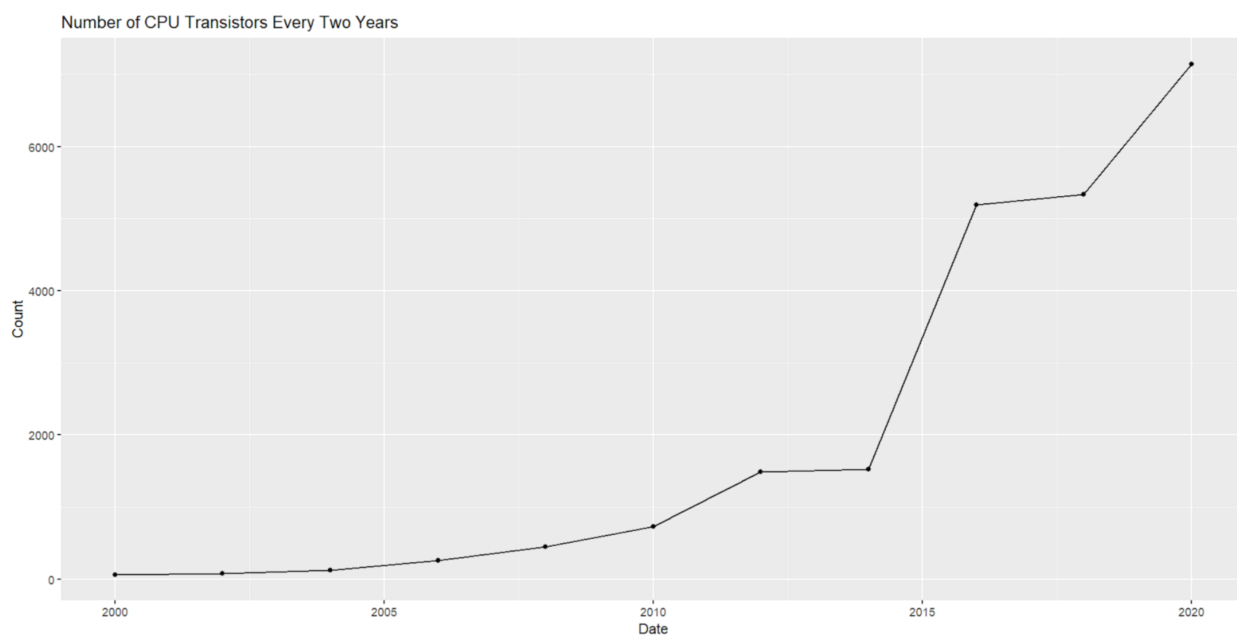
- What I observed in the data numerically:

```
dates <- as.Date(cpu_gpu_data$`Release Date`, "%m/%d/%Y")
cpu_gpu_data2 <- cpu_gpu_data %>% mutate(Firstof2ndYear=floor_date(dates, unit="2 years"))

transistorsbyYear <- cpu_gpu_data2 %>% group_by(Firstof2ndYear, Type) %>%
summarise(Average=mean(`Transistors (million)`, na.rm=TRUE)) %>% drop_na() %>% pivot_wider(.,
id_cols="Firstof2ndYear", names_from="Type", values_from="Average") %>% print()
```

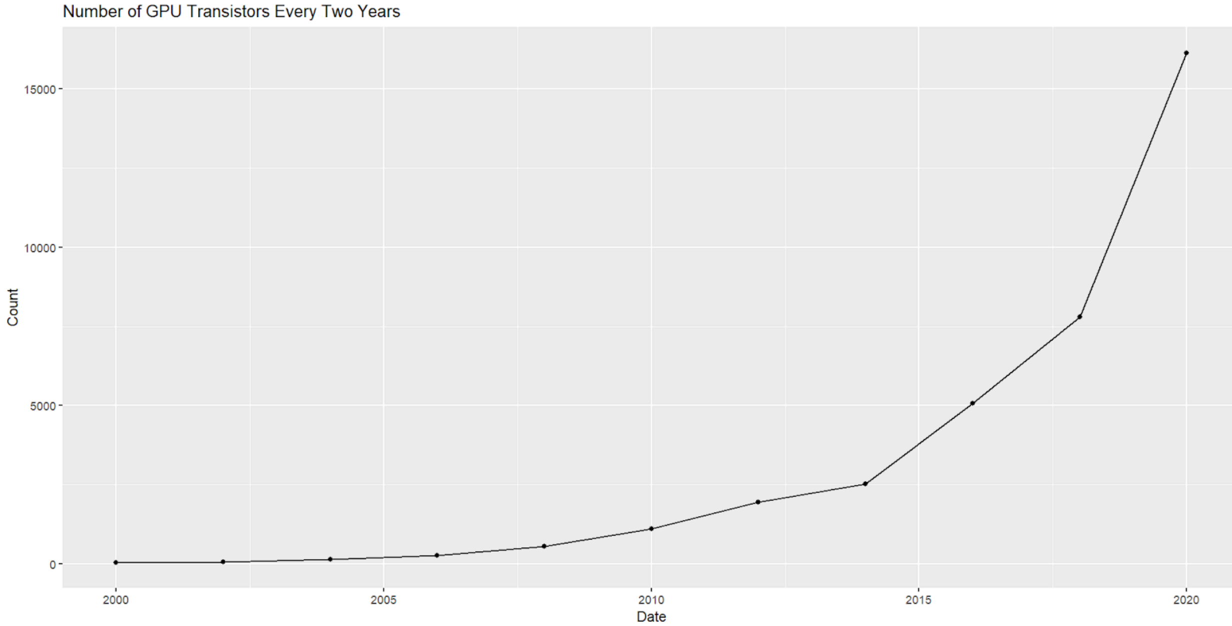
```
# A tibble: 11 x 3
# Groups:   Firstof2ndYear [11]
   Firstof2ndYear    CPU     GPU
   <date>          <dbl>   <dbl>
 1 2000-01-01       60.7    36.1
 2 2002-01-01       76.3    64.4
 3 2004-01-01      121.    142.
 4 2006-01-01      263.    260.
 5 2008-01-01      445.    550.
 6 2010-01-01      729.   1102.
 7 2012-01-01     1489.   1940.
 8 2014-01-01     1522.   2515.
 9 2016-01-01     5191.   5070.
10 2018-01-01     5341.   7793.
11 2020-01-01     7150.  16144.
```

- What I observed in the data graphically:

```
ggplot(transistorsbyYear, aes(x=Firstof2ndYear, y=CPU)) + geom_point() + geom_line() + labs(x="Date",
y="Count", title="Number of CPU Transistors Every Two Years")
```



```
ggplot(transistorsbyYear, aes(x=Firstof2ndYear, y=GPU)) + geom_point() + geom_line() + labs(x="Date",
y="Count", title="Number of GPU Transistors Every Two Years")
```

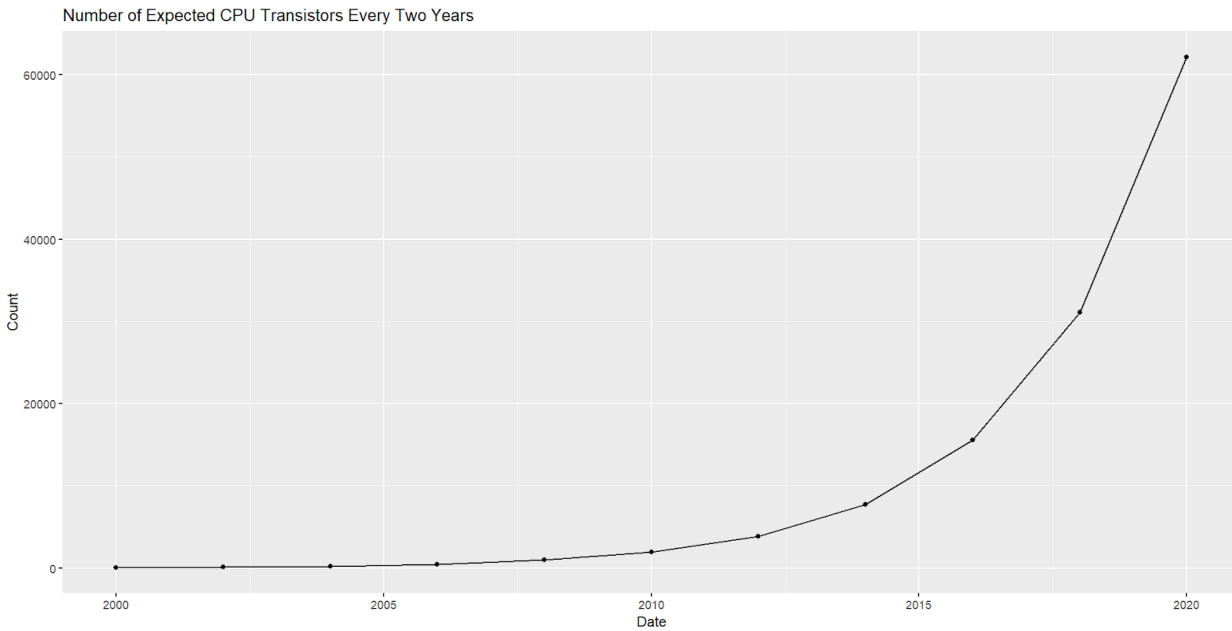


- What I expected to see if Moore's law held numerically:

```
doubleComputing <- function(tibble, column, replaceAmount) {
  ComputedTransistors <- tibble
  for(i in 0:replaceAmount) {
    ComputedTransistors[[column]][i+1] <- transistorsbyYear[[column]][1]*(2^i)
  }
  ComputedTransistors
}

MytransistorsbyYear <- doubleComputing(transistorsbyYear, 2, 10)
MytransistorsbyYear2 <- doubleComputing(MytransistorsbyYear, 3, 10)
MytransistorsbyYear2

# A tibble: 11 x 3
# Groups:   Firstof2ndYear [11]
   Firstof2ndYear     CPU     GPU
   <date>           <dbl>   <dbl>
 1 2000-01-01        60.7    36.1
 2 2002-01-01       121.     72.3
 3 2004-01-01       243.    145.
 4 2006-01-01       486.    289.
 5 2008-01-01       971.    578.
 6 2010-01-01      1943.   1157.
 7 2012-01-01      3885.   2314.
 8 2014-01-01      7771.   4627.
 9 2016-01-01     15541.   9254.
10 2018-01-01     31083.  18508.
11 2020-01-01     62166.  37016.
```

- What I expected to see if Moore's law held graphically:

```
ggplot(MytransistorsbyYear2, aes(x=Firstof2ndYear, y=CPU)) + geom_point() + geom_line() + labs(x="Date", y="Count", title="Number of Expected CPU Transistors Every Two Years")
```




```
ggplot(MytransistorsbyYear2, aes(x=Firstof2ndYear, y=GPU)) + geom_point() + geom_line() + labs(x="Date", y="Count", title="Number of Expected GPU Transistors Every Two Years")
```