Project on CPU, GPU, and TPU

Lixian Chen, Zhanhao Zhang, Jingbin Cao

4/22/2021

Preparing Required Packages

Read Data

```
data <- read.csv("../data/Runtime.csv")
head(data)
```

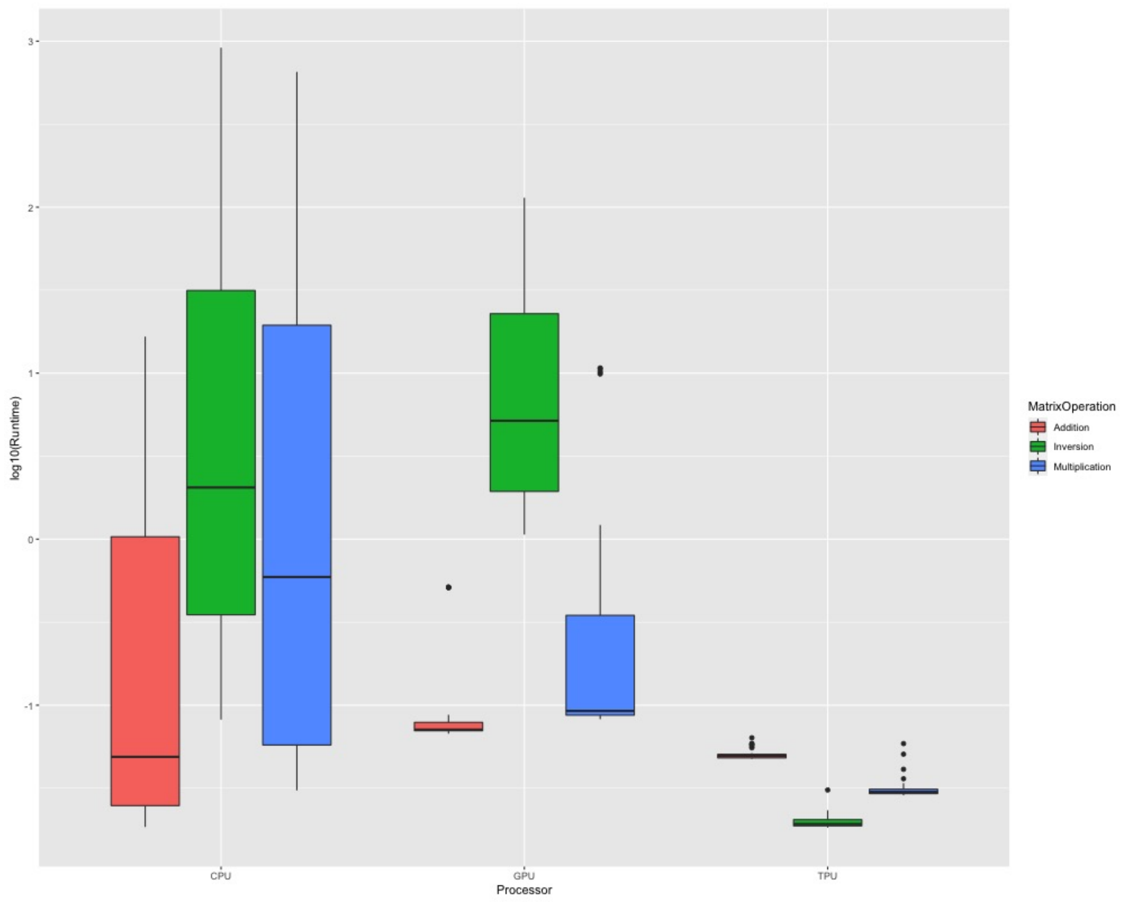
```
##      Runtime Processor MatrixSize MatrixOperation Trial
## 1 0.02059865       CPU         10        Addition     1
## 2 0.01871443       CPU         10        Addition     2
## 3 0.01863551       CPU         10        Addition     3
## 4 0.01847029       CPU         10        Addition     4
## 5 0.01866961       CPU         10        Addition     5
## 6 0.02328420       CPU         20        Addition     1
```

We tested three types of processors CPU, GPU, and TPU for three kinds of matrix operation, addition, multiplication, and inversion, with the matrix from size 10 to size 2160. We repeat each test for five times. **We measured log10(run-time) for each trial, and we use that as the evaluation of the performances.**

Simple Plots

Here is the general visualization for the performances of each processor under three matrix operations:

```
jpeg(filename = "../figs/overview.jpeg", width = 1000, height = 800,quality = 10000)
ggplot(data = data, aes(x = Processor, y = log10(Runtime))) +
geom_boxplot(aes(fill = MatrixOperation))
while (!is.null(dev.list()))  dev.off()
```



Jingbin Cao Part I

Research different Matrix Sizes One Way Anova for different matrix sized for each pair of processor and matrix operation: $\mu_1 = Matrix_{Size} = 160$ $\mu_2 = Matrix_{Size} = 320$ $\mu_3 = Matrix_{Size} = 640$ $\mu_4 = Matrix_{Size} = 1280$

Getting Data

```
cpu_add <- data[data$Processor == "CPU" & data$MatrixOperation=="Addition" & data$MatrixSize >= 160,]
cpu_mult <- data[data$Processor == "CPU" & data$MatrixOperation=="Multiplication" & data$MatrixSize >=
cpu_inv <- data[data$Processor == "CPU" & data$MatrixOperation=="Inversion" & data$MatrixSize >= 160,]
gpu_add <- data[data$Processor == "GPU" & data$MatrixOperation=="Addition" & data$MatrixSize >= 160,]
gpu_mult <- data[data$Processor == "GPU" & data$MatrixOperation=="Multiplication" & data$MatrixSize >=
gpu_inv <- data[data$Processor == "GPU" & data$MatrixOperation=="Inversion" & data$MatrixSize >= 160,]
tpu_add <- data[data$Processor == "TPU" & data$MatrixOperation=="Addition" & data$MatrixSize >= 160,]
tpu_mult <- data[data$Processor == "TPU" & data$MatrixOperation=="Multiplication" & data$MatrixSize >=
tpu_inv <- data[data$Processor == "TPU" & data$MatrixOperation=="Inversion" & data$MatrixSize >= 160,]
```

Anovas

```
summary(mod_cpu_add <- aov(log10(Runtime) ~ as.factor(MatrixSize), data=cpu_add))
```

```
##                       Df Sum Sq Mean Sq F value Pr(>F)    
## as.factor(MatrixSize)  3 15.478   5.159   76040 <2e-16 ***
## Residuals             16  0.001   0.000                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(mod_cpu_mult <- aov(log10(Runtime) ~ as.factor(MatrixSize), data=cpu_mult))
```

```
##                       Df Sum Sq Mean Sq F value Pr(>F)    
## as.factor(MatrixSize)  3  19.53   6.511  393855 <2e-16 ***
## Residuals             16   0.00   0.000                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(mod_cpu_inv <- aov(log10(Runtime) ~ as.factor(MatrixSize), data=cpu_inv))
```

```
##                       Df Sum Sq Mean Sq F value Pr(>F)    
## as.factor(MatrixSize)  3  15.67   5.223  653718 <2e-16 ***
## Residuals             16   0.00   0.000                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(mod_gpu_add <- aov(log10(Runtime) ~ as.factor(MatrixSize), data=gpu_add))
```

```
##                       Df Sum Sq Mean Sq F value Pr(>F)    
## as.factor(MatrixSize)  3 2.6297  0.8766   10205 <2e-16 ***
## Residuals             16 0.0014  0.0001                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(mod_gpu_mult <- aov(log10(Runtime) ~ as.factor(MatrixSize), data=gpu_mult))
```

```
##                       Df Sum Sq Mean Sq F value Pr(>F)    
## as.factor(MatrixSize)  3 12.328   4.109   34853 <2e-16 ***
## Residuals             16  0.002   0.000                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(mod_gpu_inv <- aov(log10(Runtime) ~ as.factor(MatrixSize), data=gpu_inv))
```

```
##                       Df Sum Sq Mean Sq F value Pr(>F)    
## as.factor(MatrixSize)  3  4.083   1.361 4518902 <2e-16 ***
## Residuals             16  0.000   0.000                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(mod_tpu_add <- aov(log10(Runtime) ~ as.factor(MatrixSize), data=tpu_add))
```

```
##                       Df   Sum Sq  Mean Sq F value Pr(>F)
## as.factor(MatrixSize)  3 0.003245 0.001081   0.849  0.487
## Residuals             16 0.020376 0.001273
```

```
summary(mod_tpu_mult <- aov(log10(Runtime) ~ as.factor(MatrixSize), data=tpu_mult))
```

```
##                       Df  Sum Sq  Mean Sq F value Pr(>F)
## as.factor(MatrixSize)  3 0.00678 0.002259   0.707  0.562
## Residuals             16 0.05116 0.003197
```

```
summary(mod_tpu_inv <- aov(log10(Runtime) ~ as.factor(MatrixSize), data=tpu_inv))
```

```
##                       Df  Sum Sq  Mean Sq F value Pr(>F)
## as.factor(MatrixSize)  3 0.00667 0.002224   0.936  0.446
## Residuals             16 0.03801 0.002376
```

```
# From the tables, we can see that only TPU & Multiplication and TPU & Inversion do not have Matrix Siz
# plot(aov(Runtime ~ as.factor(MatrixSize), data=tpu_mult))
# plot(aov(Runtime ~ as.factor(MatrixSize), data=tpu_inv))
```

Confidence Interval for LMs

```
round(digits=4,confint(mod_cpu_add))
```

```
##                              2.5 %  97.5 %
## (Intercept)                -1.1994 -1.1838
## as.factor(MatrixSize)320    1.0339  1.0560
## as.factor(MatrixSize)640    1.6346  1.6567
## as.factor(MatrixSize)1280   2.3952  2.4173
```

```
round(digits=4,confint(mod_cpu_mult))
```

```
##                             2.5 % 97.5 %
## (Intercept)               0.1445 0.1522
## as.factor(MatrixSize)320  0.9245 0.9354
## as.factor(MatrixSize)640  1.7674 1.7783
## as.factor(MatrixSize)1280 2.6596 2.6705
```

```
round(digits=4,confint(mod_cpu_inv))
```

```
##                             2.5 % 97.5 %
## (Intercept)               0.5928 0.5982
## as.factor(MatrixSize)320  0.6918 0.6994
## as.factor(MatrixSize)640  1.5130 1.5206
## as.factor(MatrixSize)1280 2.3589 2.3664
```

```r
round(confint(mod_gpu_add),4)
```

```
##                               2.5 %  97.5 %
## (Intercept)                 -1.1665 -1.1489
## as.factor(MatrixSize)320   -0.0066  0.0183
## as.factor(MatrixSize)640    0.0865  0.1114
## as.factor(MatrixSize)1280   0.8550  0.8798
```

```r
round(digits=4,confint(mod_gpu_mult))
```

```
##                               2.5 %  97.5 %
## (Intercept)                 -1.0610 -1.0404
## as.factor(MatrixSize)320    0.3907  0.4198
## as.factor(MatrixSize)640    1.1183  1.1474
## as.factor(MatrixSize)1280   2.0502  2.0793
```

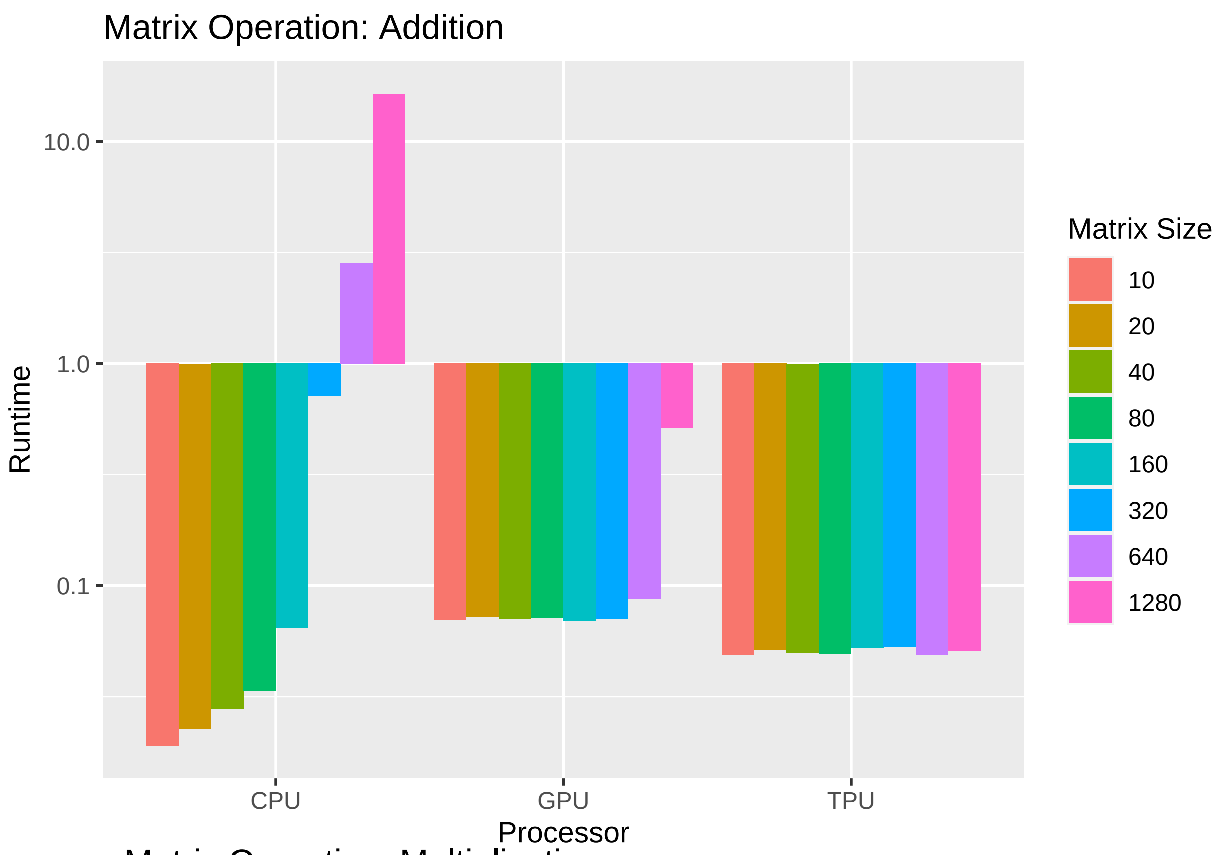
```r
round(digits=4,confint(mod_gpu_inv))
```

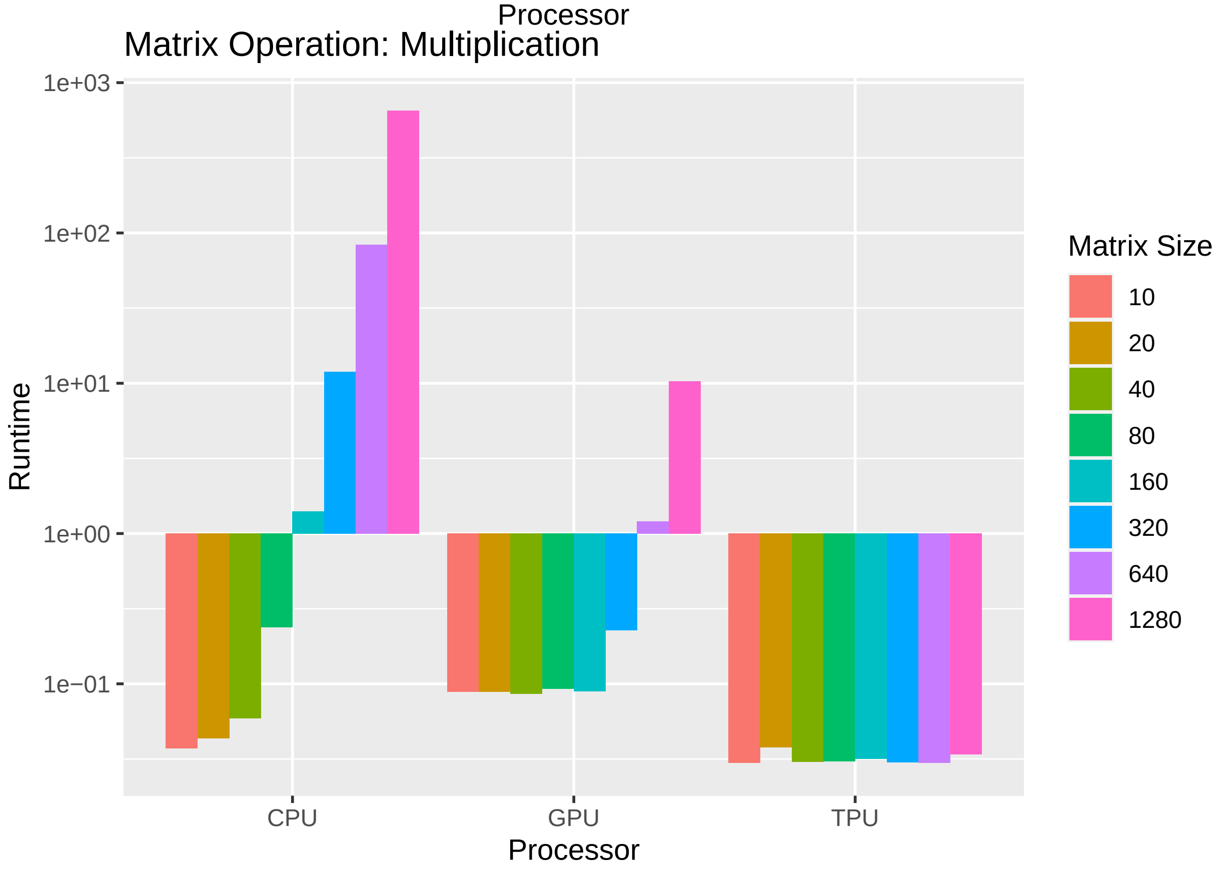
```
##                              2.5 % 97.5 %
## (Intercept)                 0.8389 0.8400
## as.factor(MatrixSize)320   0.4203 0.4218
## as.factor(MatrixSize)640   0.8101 0.8116
## as.factor(MatrixSize)1280  1.2162 1.2177
```

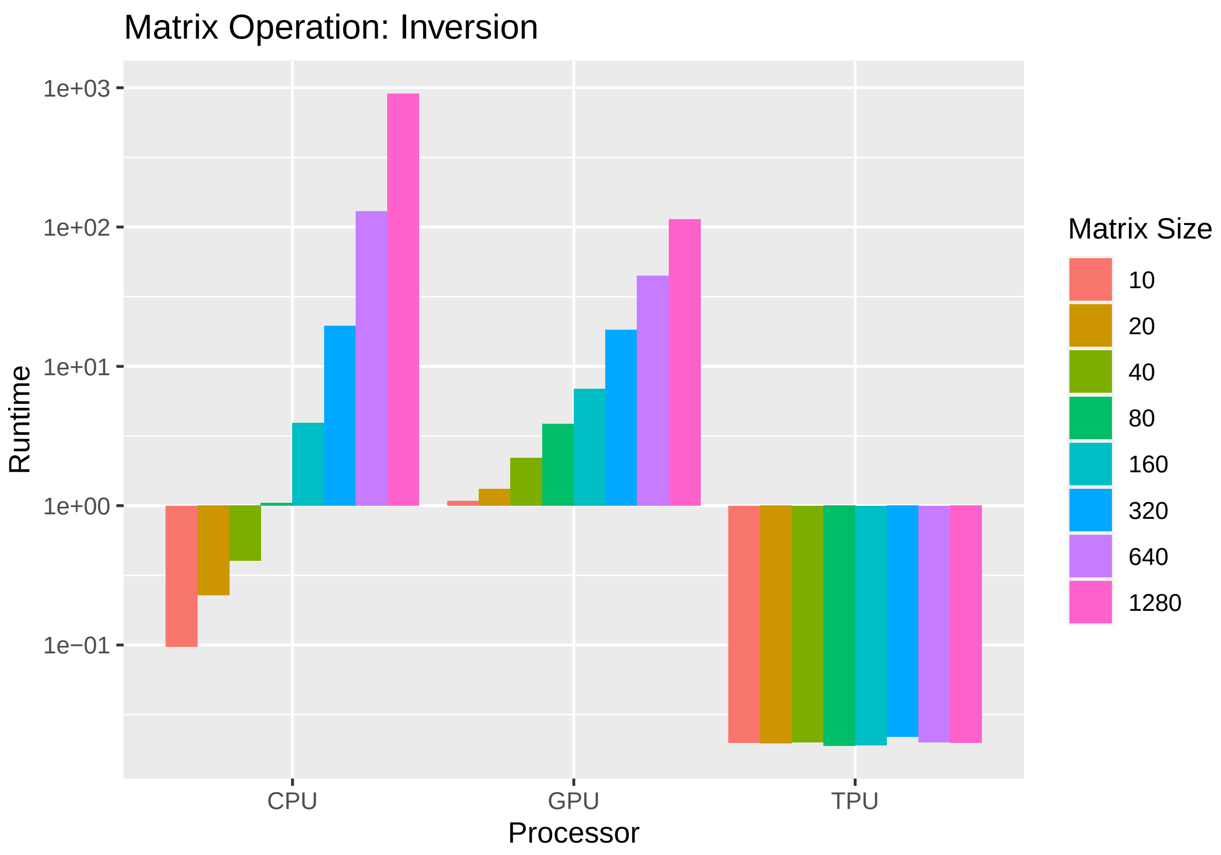
Zhanhao Zhang Part I

Interaction Plots

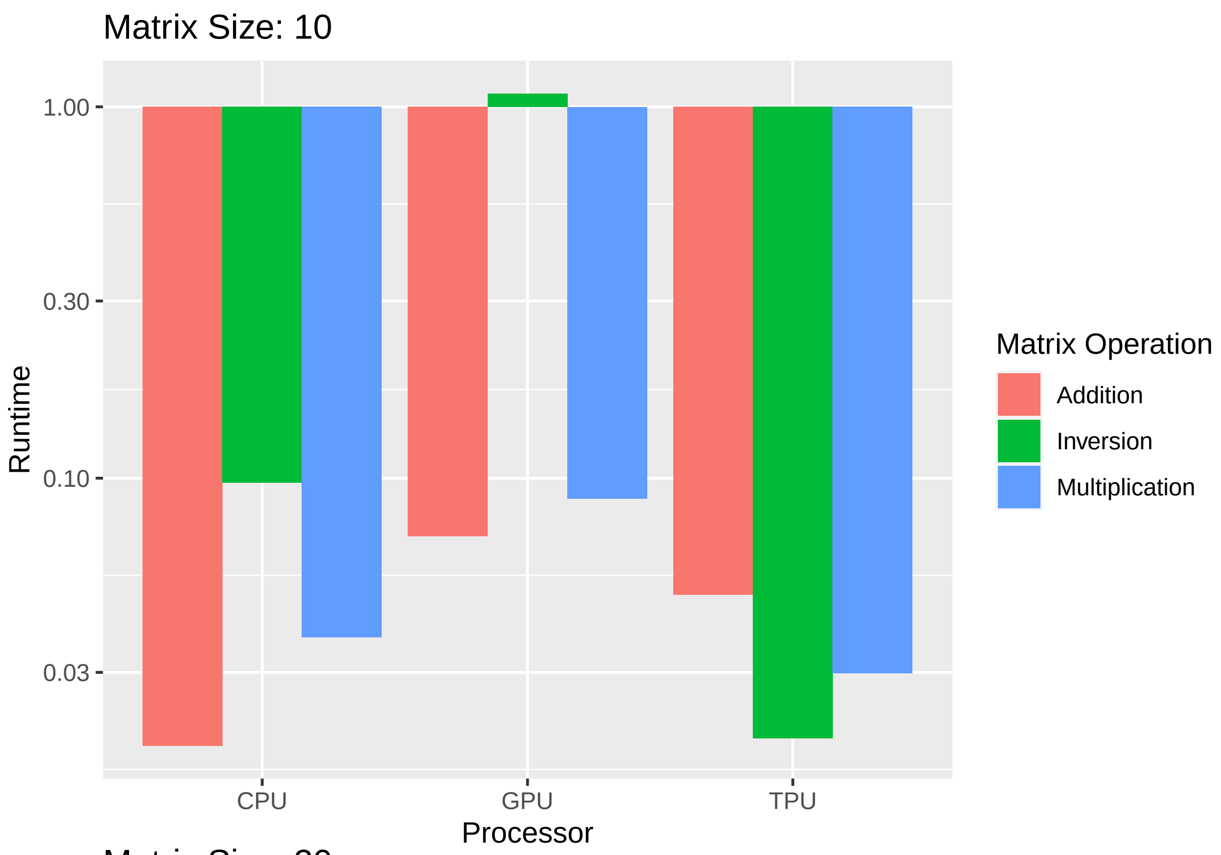
```r
#ggplot(data = data, aes(x = Processor, y = log10(Runtime))) +
 # geom_boxplot(aes(fill = MatrixOperation))
for(operation in unique(data$MatrixOperation)){
  #png(paste0("../figs/", operation, ".png"), width = 500, height = 500)
  p <- data %>%
    dplyr::filter(MatrixOperation == operation) %>%
    group_by(MatrixSize, MatrixOperation, Processor) %>%
    summarize(Runtime = mean(Runtime)) %>%
    ggplot(aes(x = Processor, y = Runtime)) +
    #geom_boxplot(aes(fill = as.factor(MatrixSize))) +
    geom_bar(aes(fill = as.factor(MatrixSize)), stat = "identity",
             position = "dodge") +
    scale_y_log10() +
    ggtitle(paste0("Matrix Operation: ", operation)) +
    #facet_wrap( ~ MatrixOperation, scales = "free", nrow = 1) +
    guides(fill=guide_legend(title = "Matrix Size"))
  print(p)
  #dev.off()
}
```

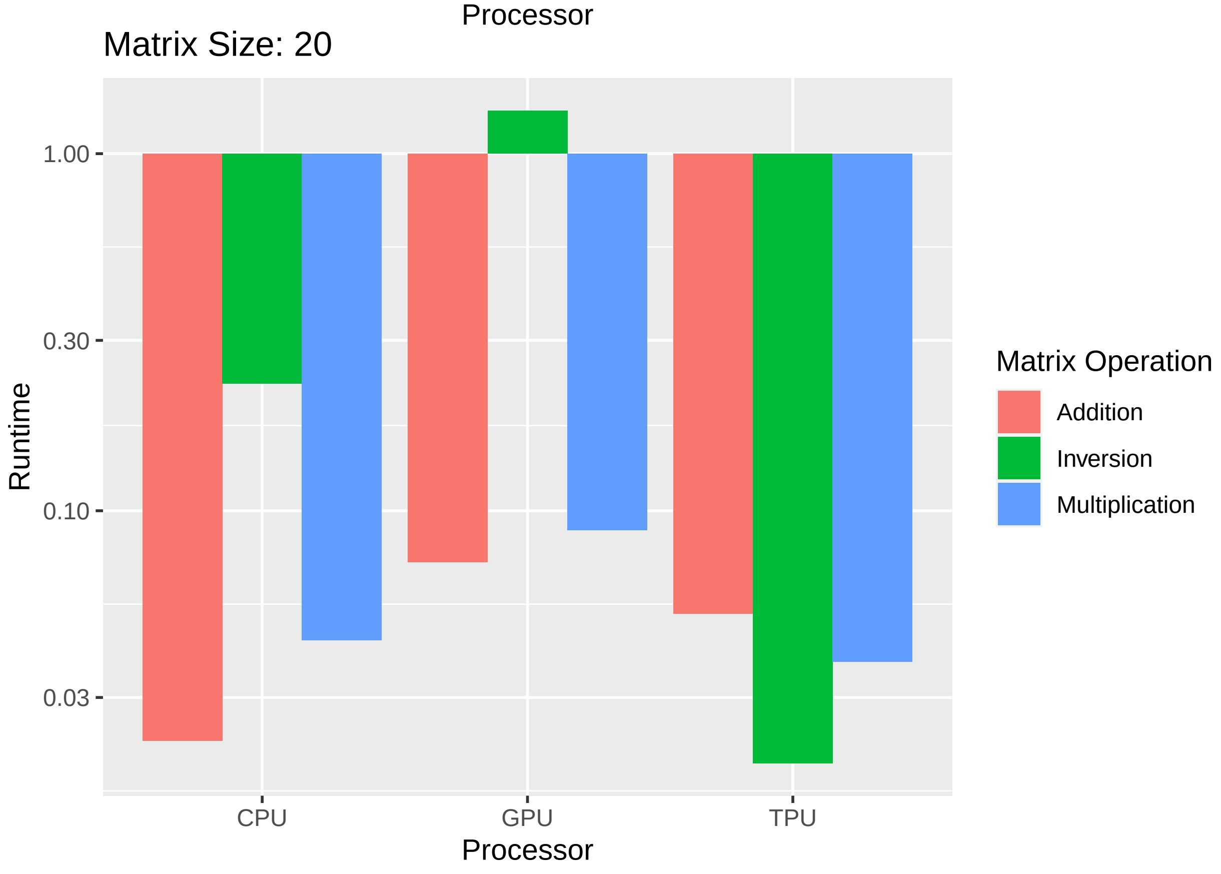

Matrix Operation: Addition
Runtime
10.0
1.0
0.1
CPU
GPU
TPU
Processor
Matrix Size
10
20
40
80
160
320
640
1280

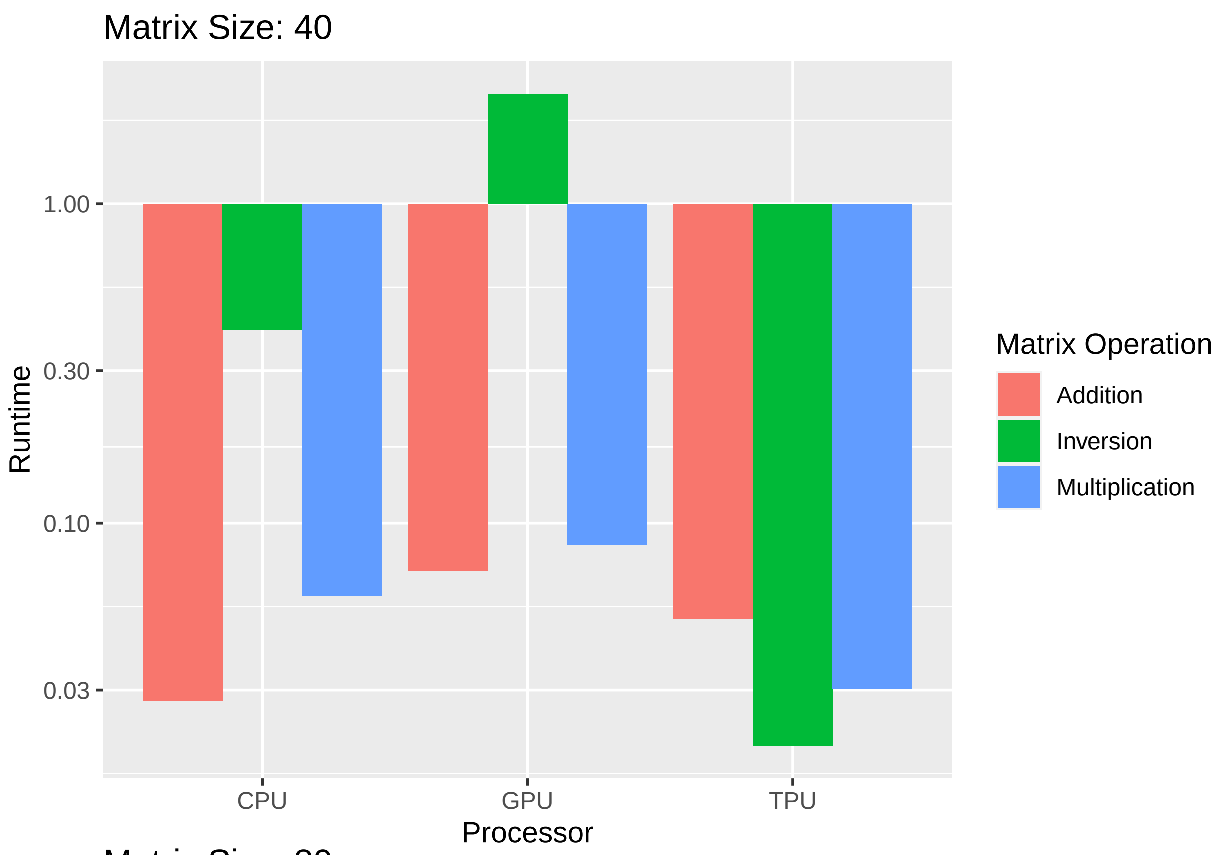

Matrix Operation: Multiplication
Runtime
1e+03
1e+02
1e+01
1e+00
1e−01
CPU
GPU
TPU
Processor
Matrix Size
10
20
40
80
160
320
640
1280

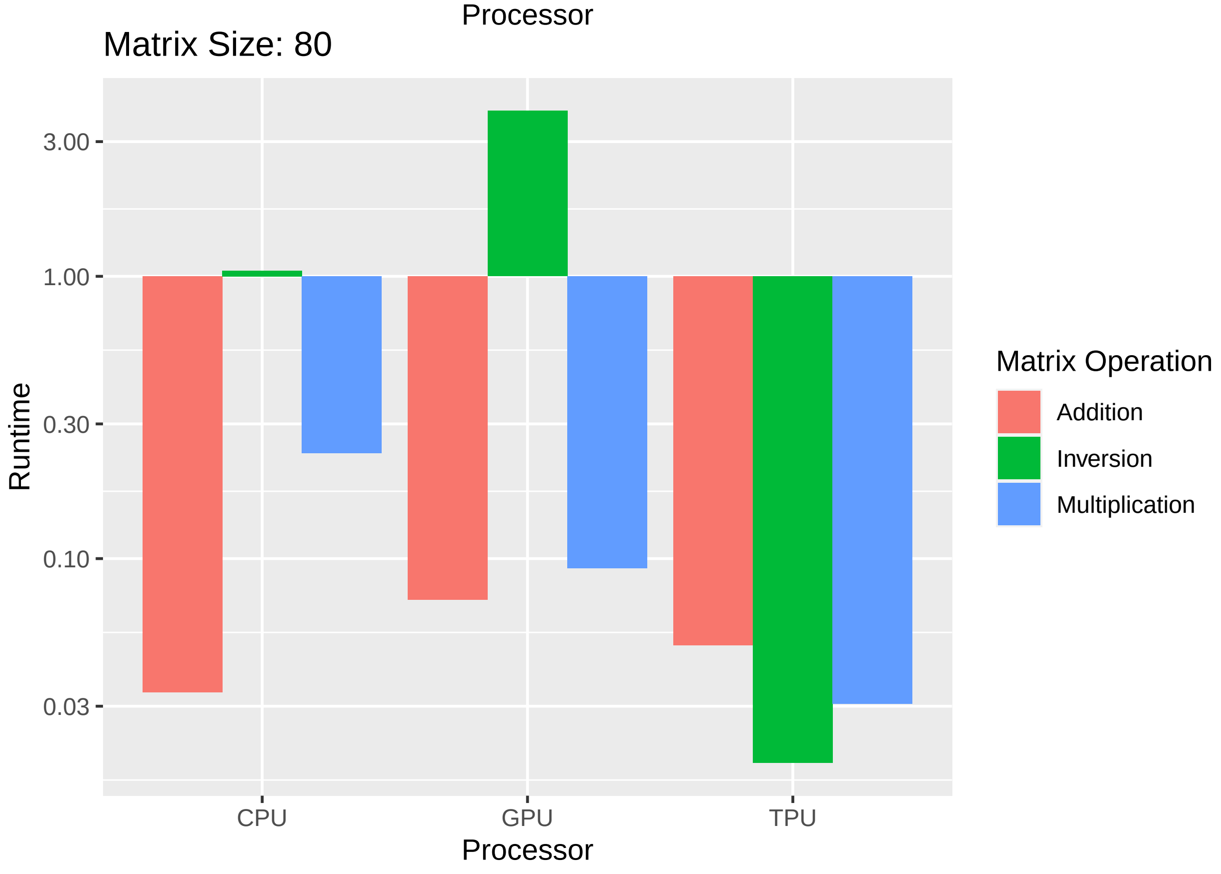


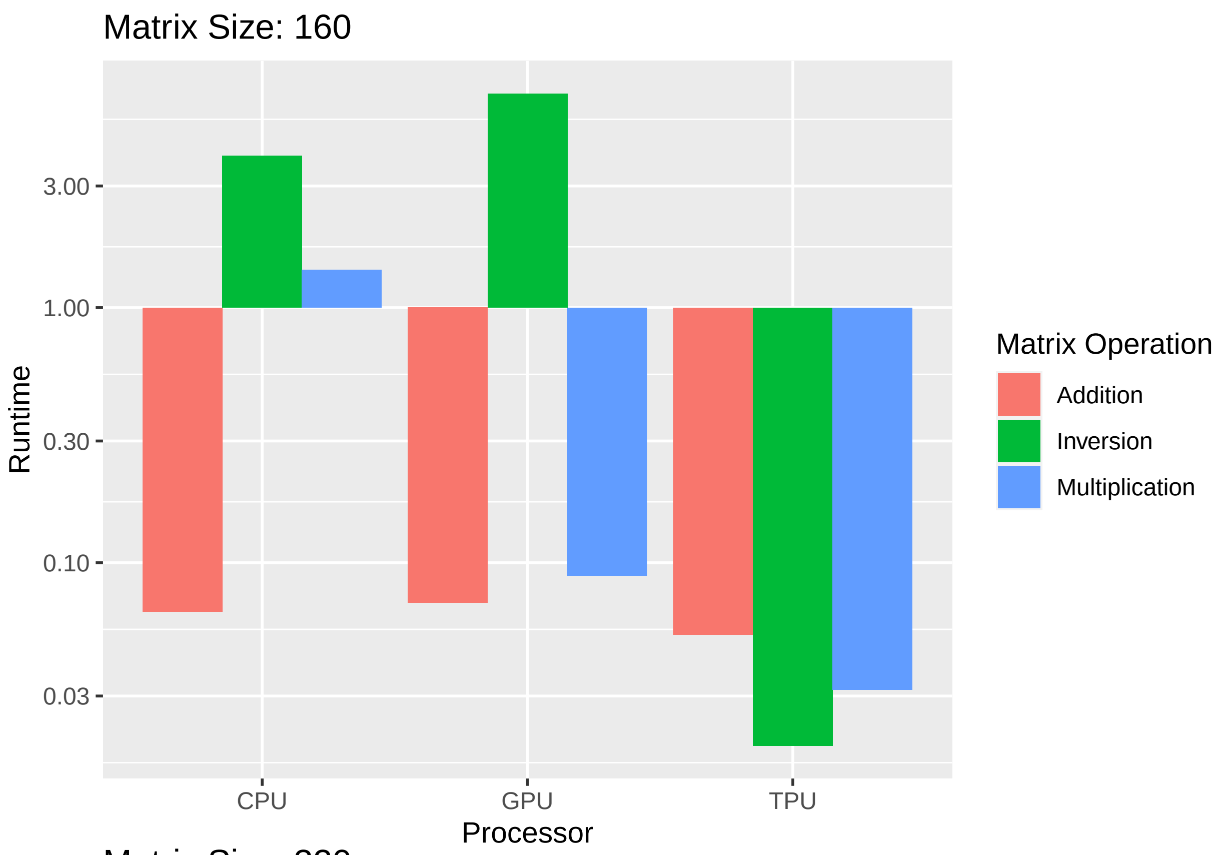
```
for(size in unique(data$MatrixSize)){
  p <- data %>%
    dplyr::filter(MatrixSize == size) %>%
    group_by(MatrixSize, MatrixOperation, Processor) %>%
    summarize(Runtime = mean(Runtime)) %>%
    ggplot(aes(x = Processor, y = Runtime)) +
    #geom_boxplot(aes(fill = as.factor(MatrixOperation))) +
    geom_bar(aes(fill = as.factor(MatrixOperation)), position = "dodge",
             stat = "identity") +
    scale_y_log10() +
    ggtitle(paste0("Matrix Size: ", size)) +
    #facet_wrap( ~ MatrixOperation, scales = "free", nrow = 1) +
    guides(fill=guide_legend(title = "Matrix Operation"))
  print(p)
}
```

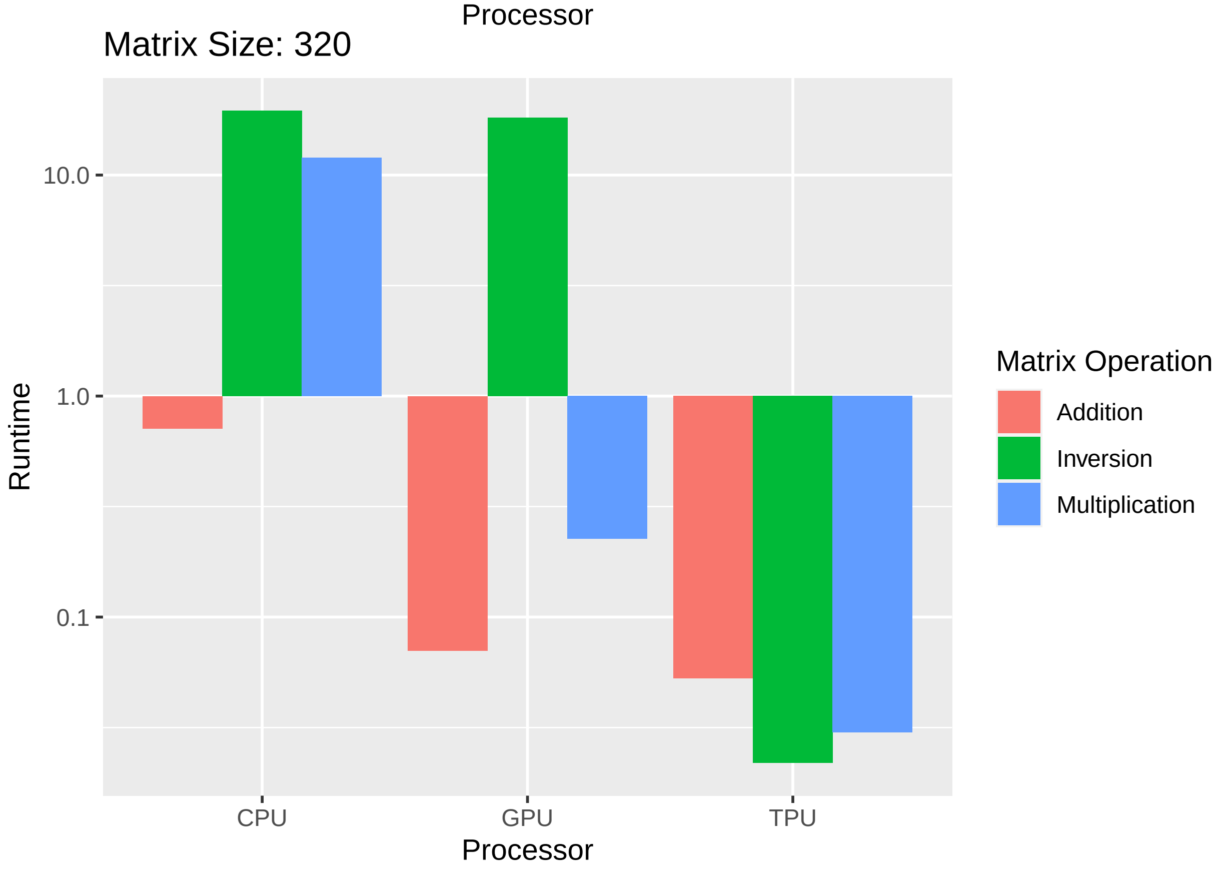

Matrix Size: 10
Runtime
1.00
0.30
0.10
0.03
CPU
GPU
TPU
Processor
Matrix Operation
Addition
Inversion
Multiplication

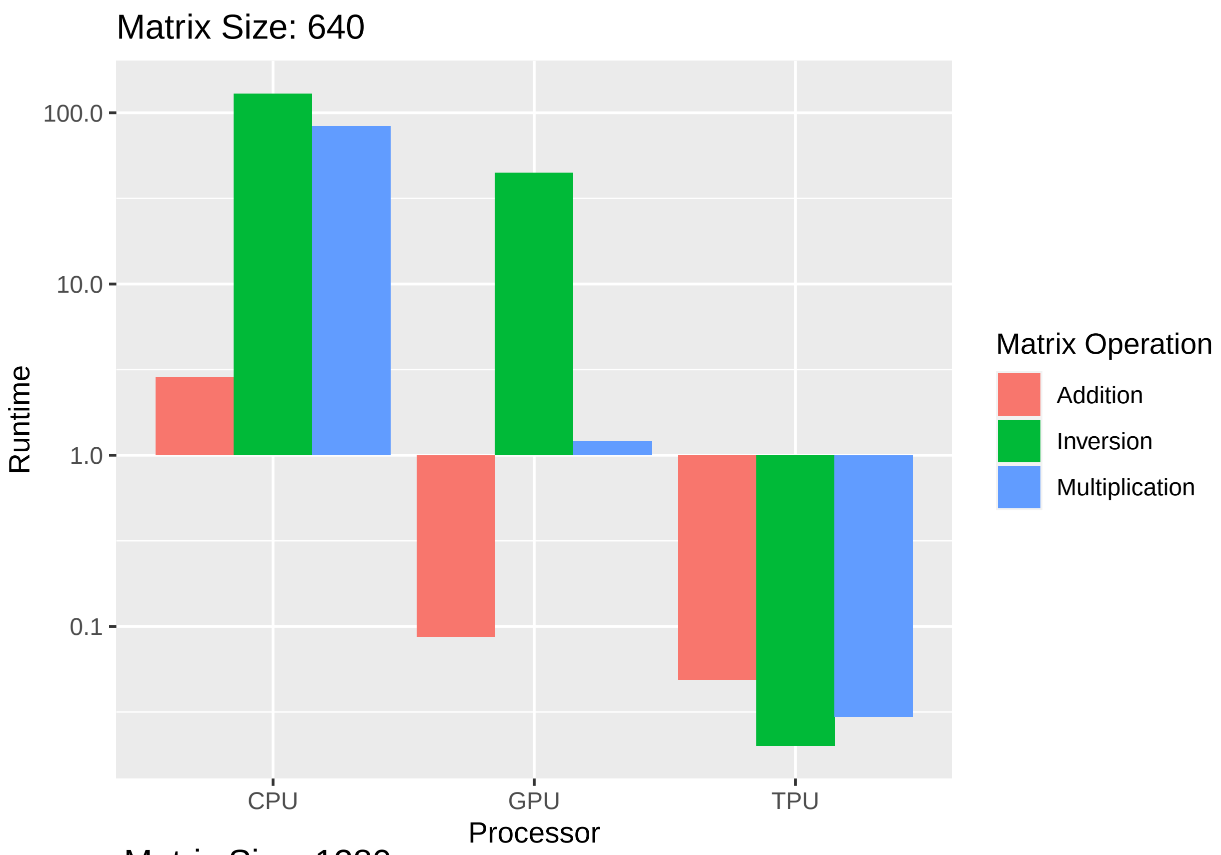

Matrix Size: 20
Runtime
1.00
0.30
0.10
0.03
CPU
GPU
TPU
Processor
Matrix Operation
Addition
Inversion
Multiplication

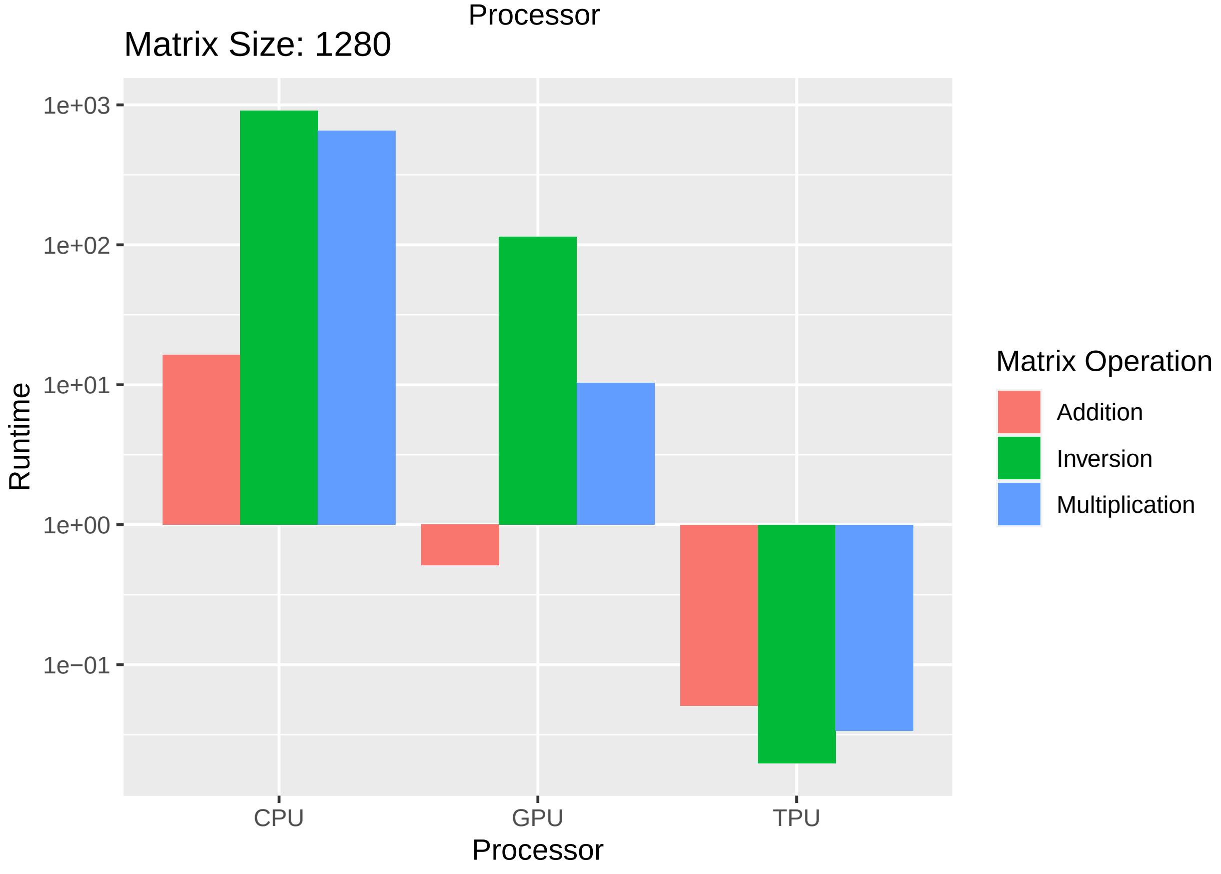

Matrix Size: 40
Runtime
1.00
0.30
0.10
0.03
CPU
GPU
TPU
Processor
Matrix Operation
Addition
Inversion
Multiplication


Matrix Size: 80
Runtime
3.00
1.00
0.30
0.10
0.03
CPU
GPU
TPU
Processor
Matrix Operation
Addition
Inversion
Multiplication

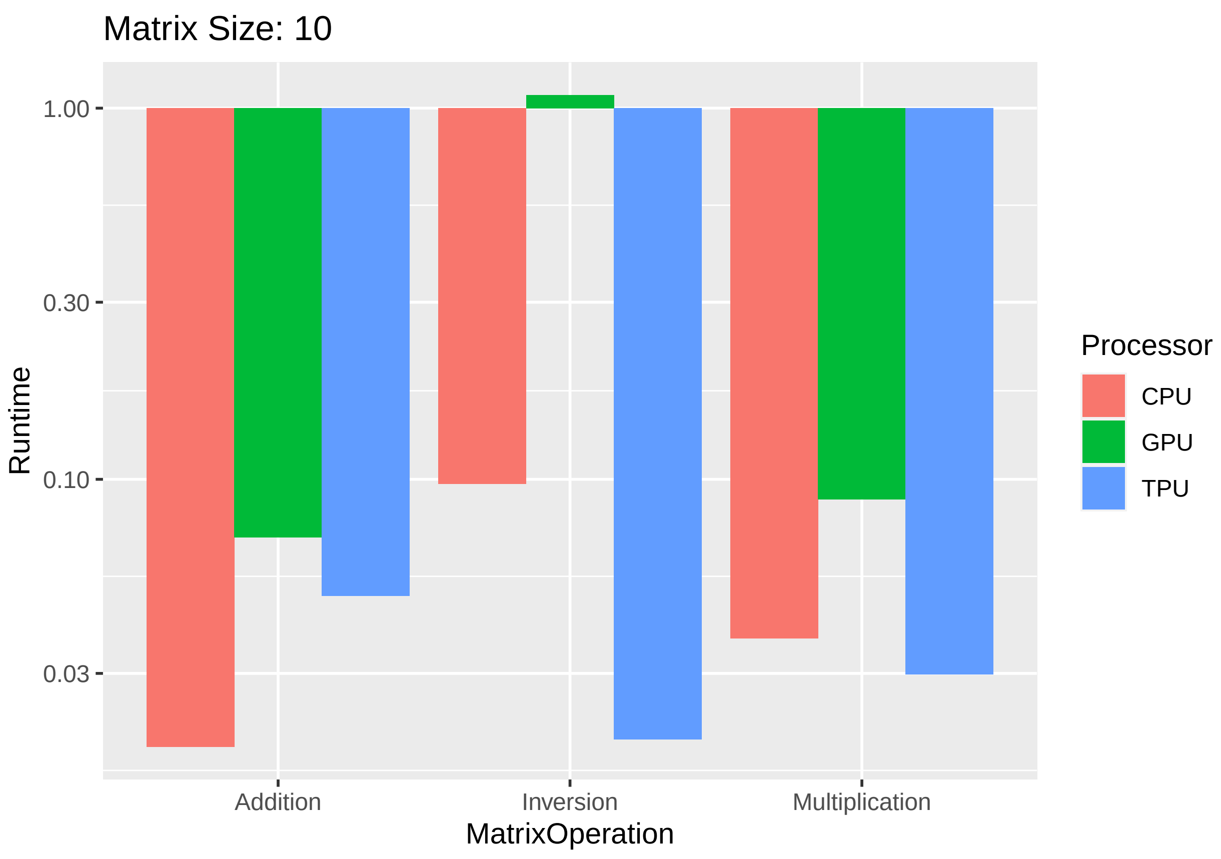

Matrix Size: 160
3.00
1.00
0.30
0.10
0.03
Runtime
CPU
GPU
TPU
Processor
Matrix Operation
Addition
Inversion
Multiplication

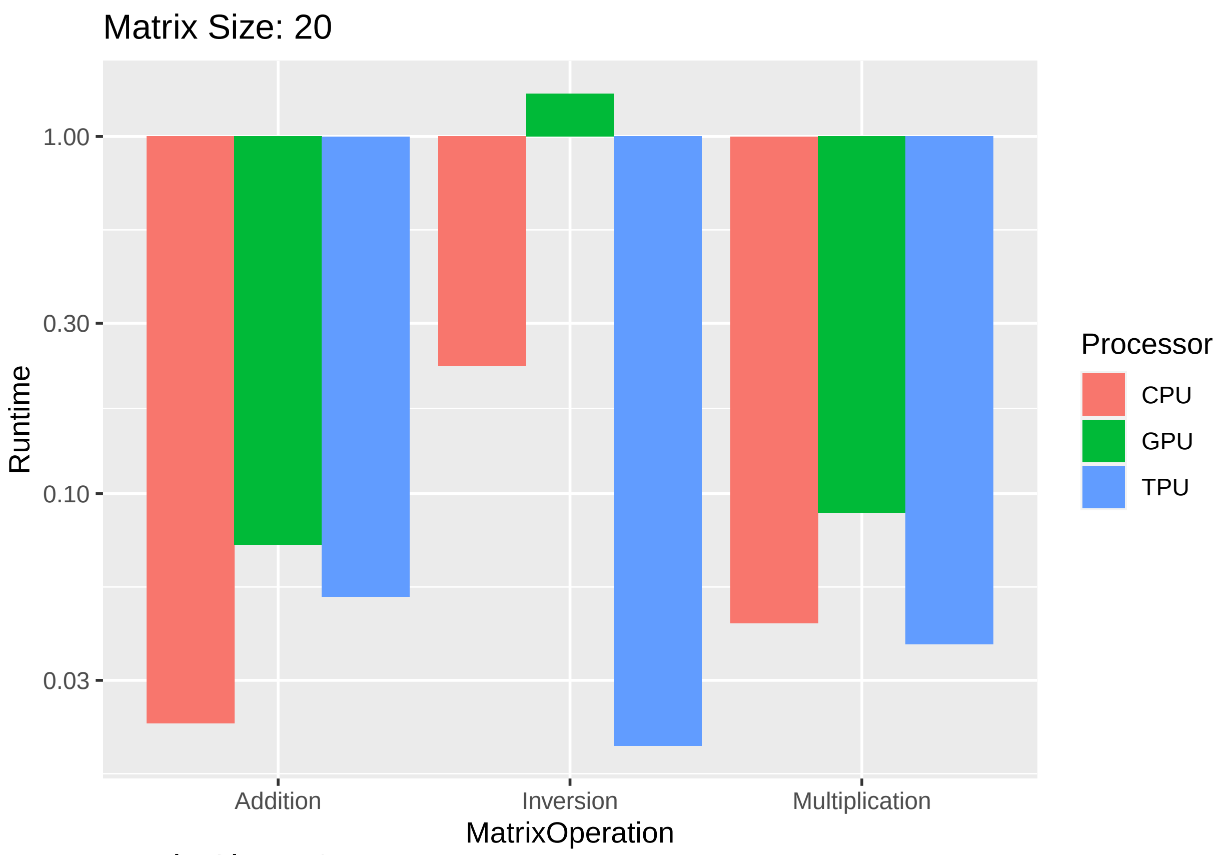

Matrix Size: 320
10.0
1.0
0.1
Runtime
CPU
GPU
TPU
Processor
Matrix Operation
Addition
Inversion
Multiplication


Matrix Size: 640
100.0
10.0
1.0
0.1
Runtime
CPU
GPU
TPU
Processor
Matrix Operation
Addition
Inversion
Multiplication

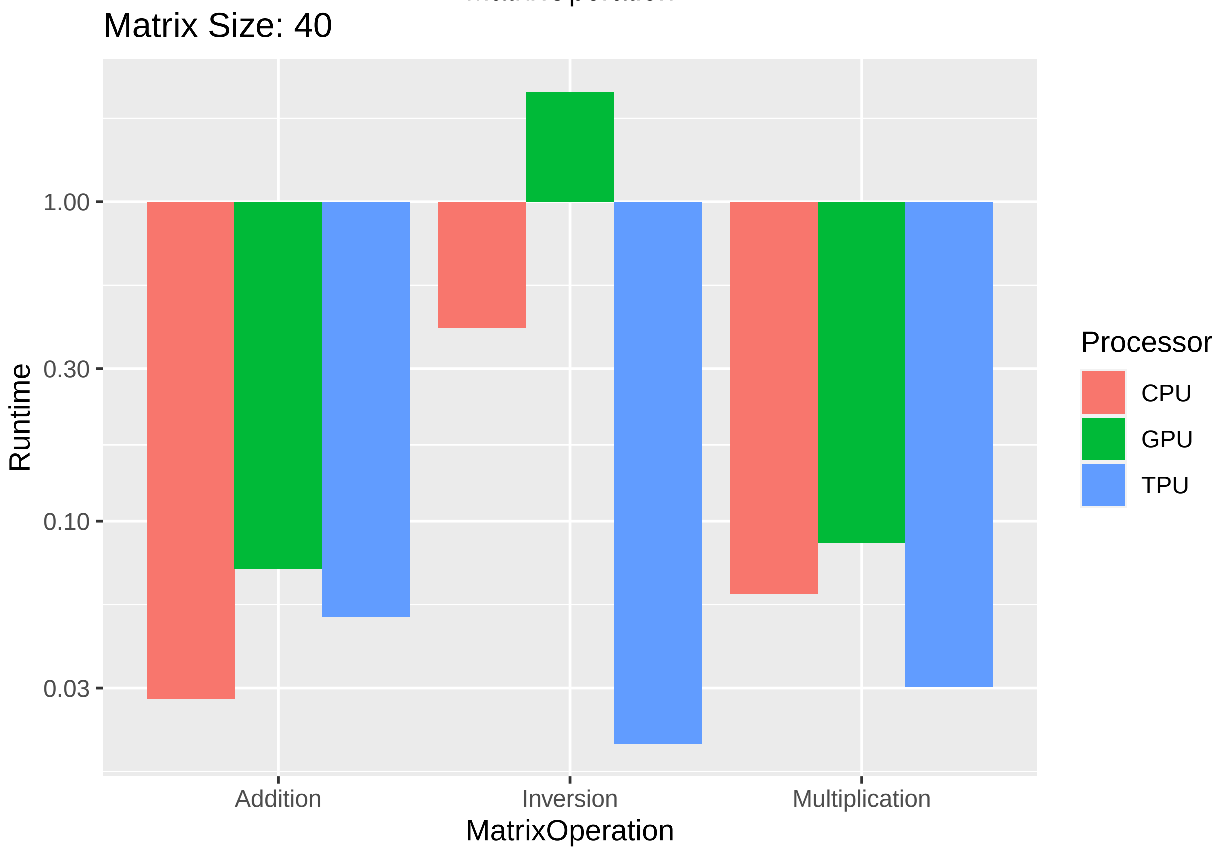

Matrix Size: 1280
1e+03
1e+02
1e+01
1e+00
1e−01
Runtime
CPU
GPU
TPU
Processor
Matrix Operation
Addition
Inversion
Multiplication

```r
for(size in unique(data$MatrixSize)){
  p <- data %>%
    dplyr::filter(MatrixSize == size) %>%
    group_by(MatrixSize, MatrixOperation, Processor) %>%
    summarize(Runtime = mean(Runtime)) %>%
    ggplot(aes(x = MatrixOperation, y = Runtime)) +
    #geom_boxplot(aes(fill = as.factor(Processor))) +
    geom_bar(aes(fill = as.factor(Processor)), position = "dodge",
             stat = "identity") +
    scale_y_log10() +
    ggtitle(paste0("Matrix Size: ", size)) +
    #facet_wrap( ~ MatrixOperation, scales = "free", nrow = 1) +
    guides(fill=guide_legend(title = "Processor"))
  print(p)
}
```



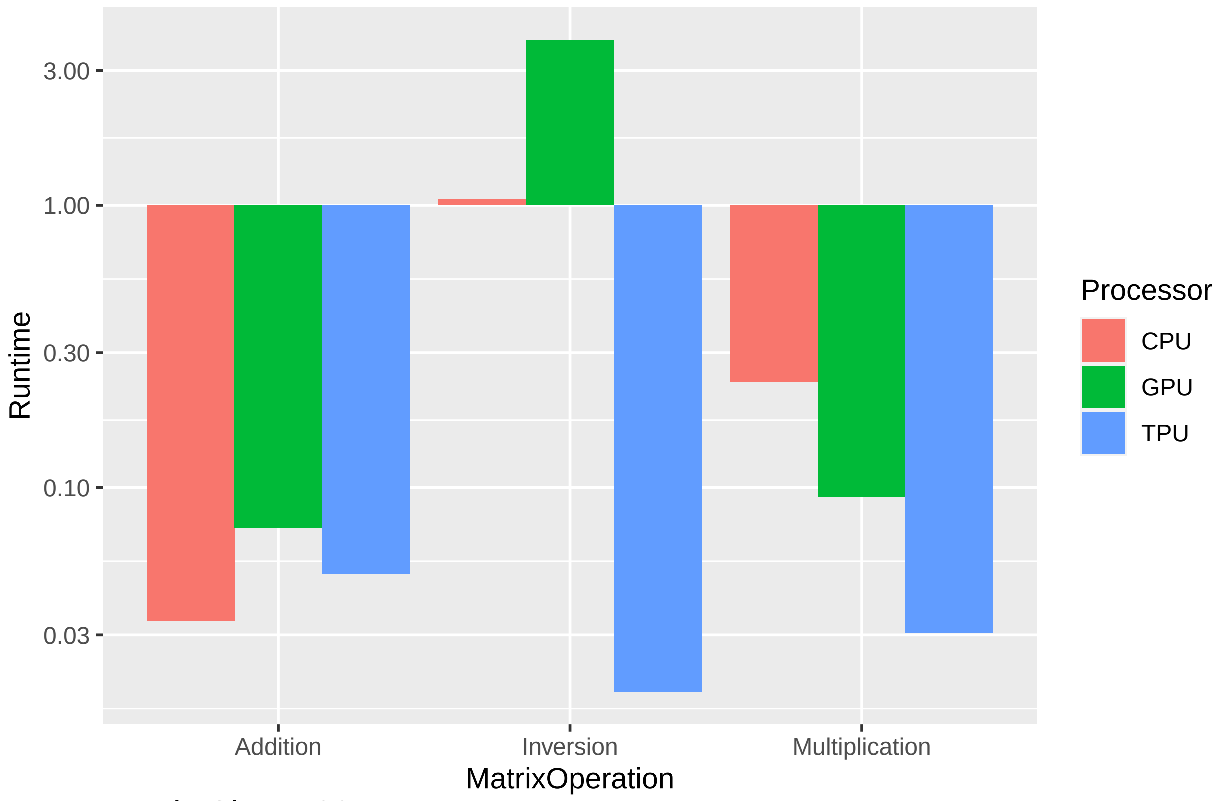


Matrix Size: 20
Runtime
1.00
0.30
0.10
0.03
Addition
Inversion
Multiplication
MatrixOperation
Processor
CPU
GPU
TPU

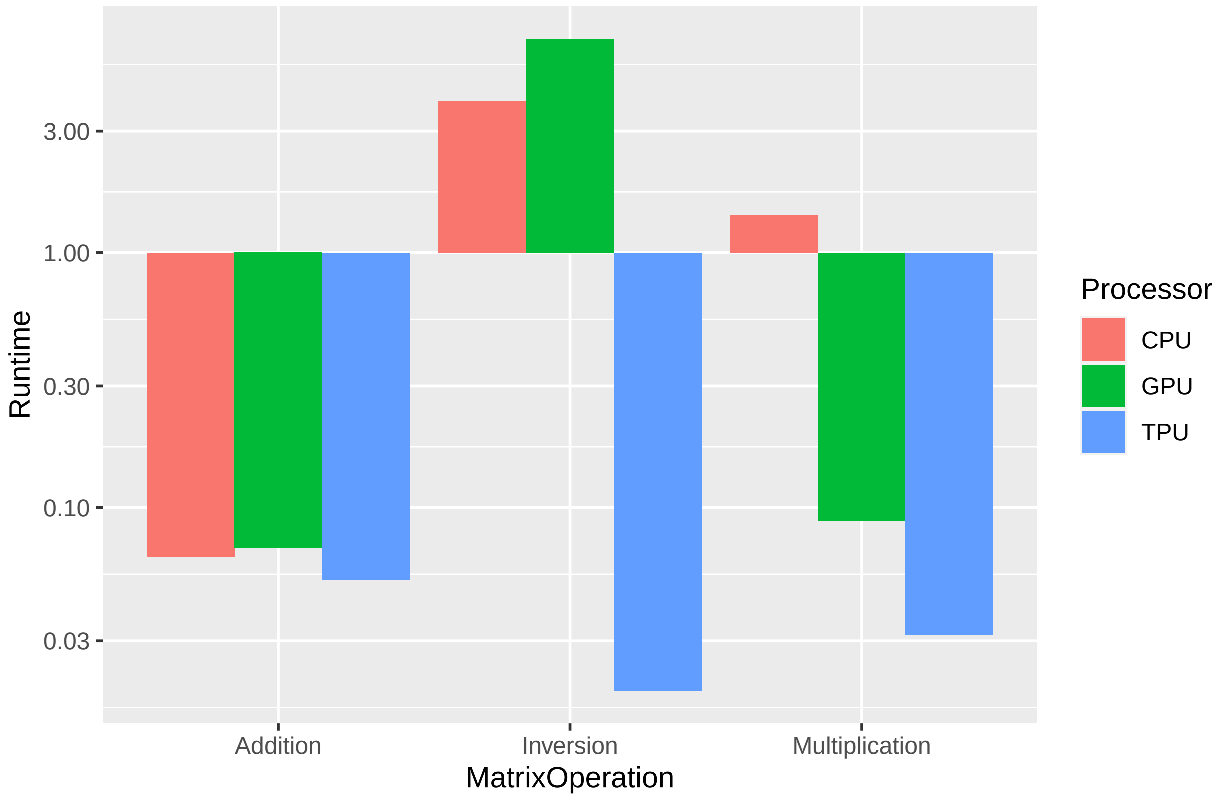


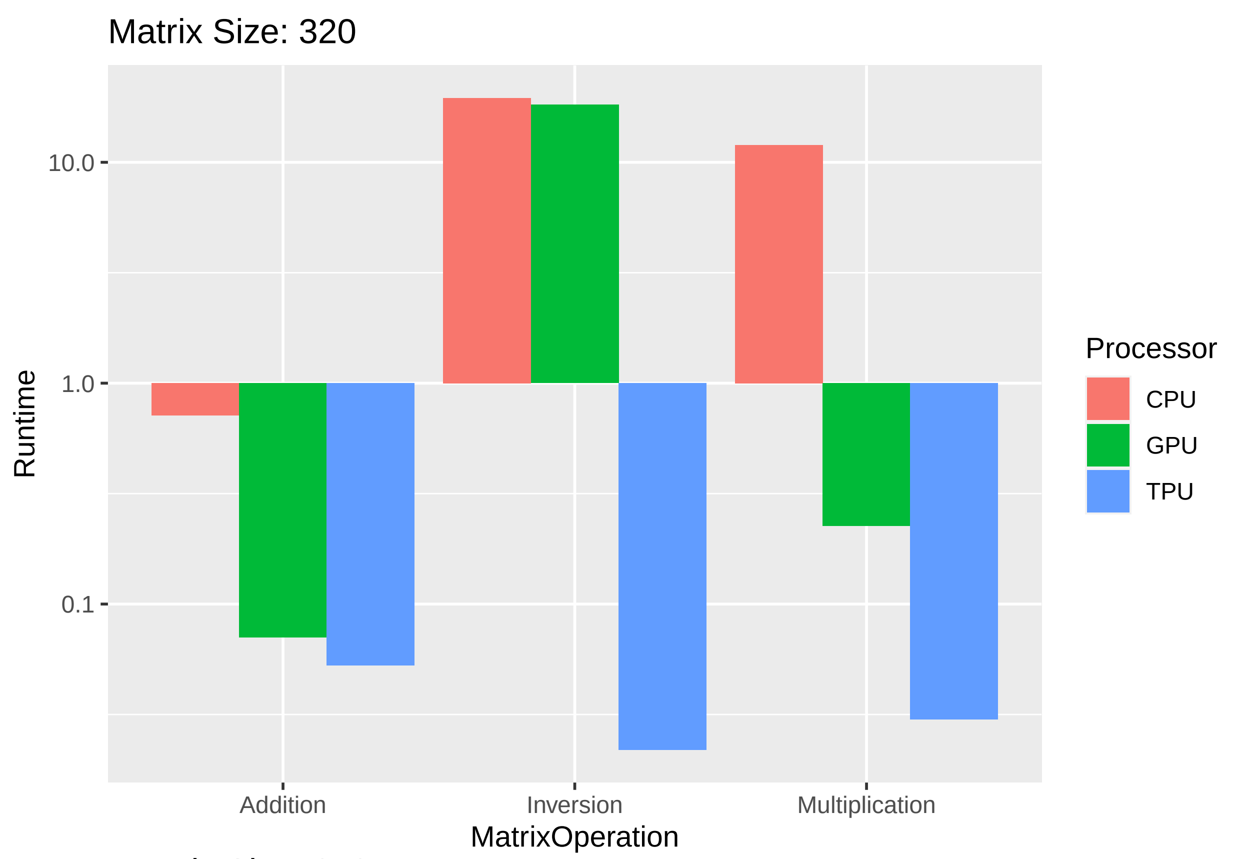
Matrix Size: 40
Runtime
1.00
0.30
0.10
0.03
Addition
Inversion
Multiplication
MatrixOperation
Processor
CPU
GPU
TPU

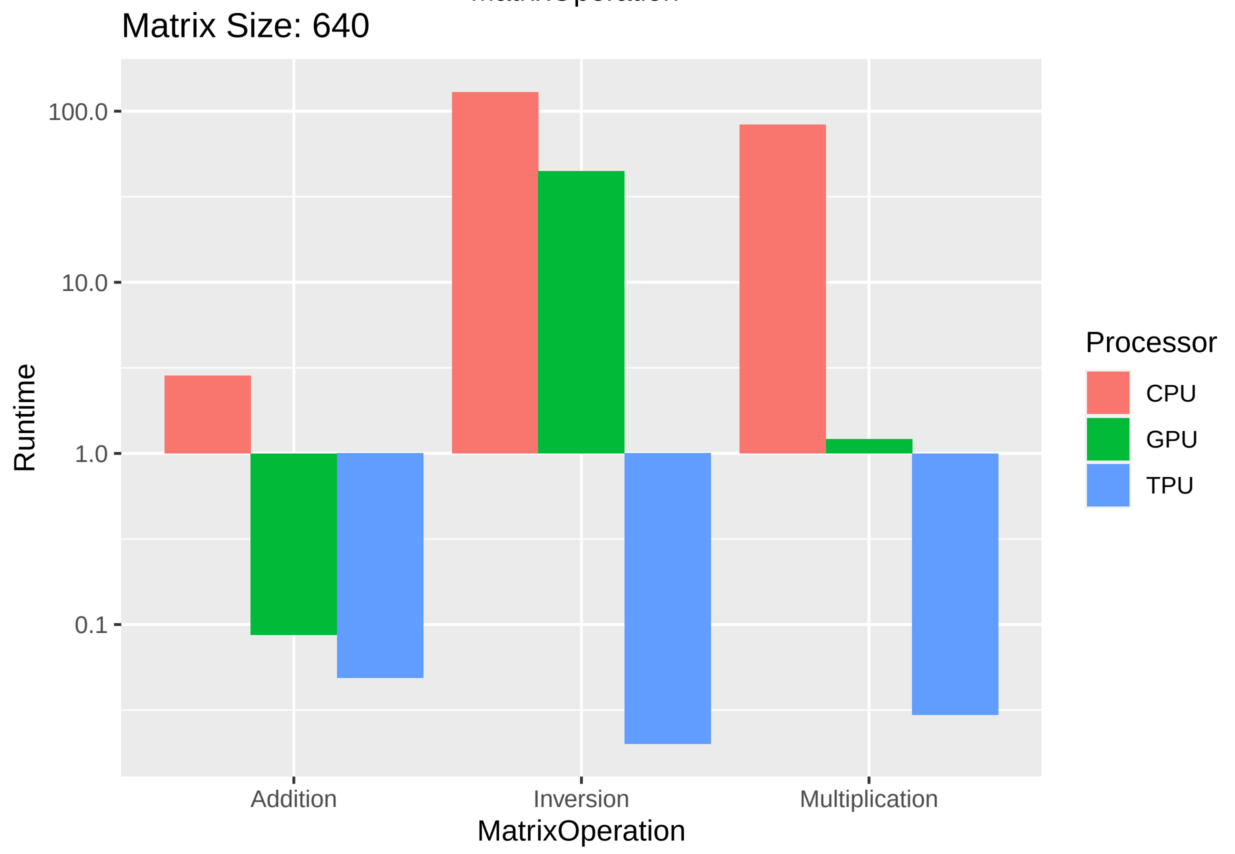
Matrix Size: 80

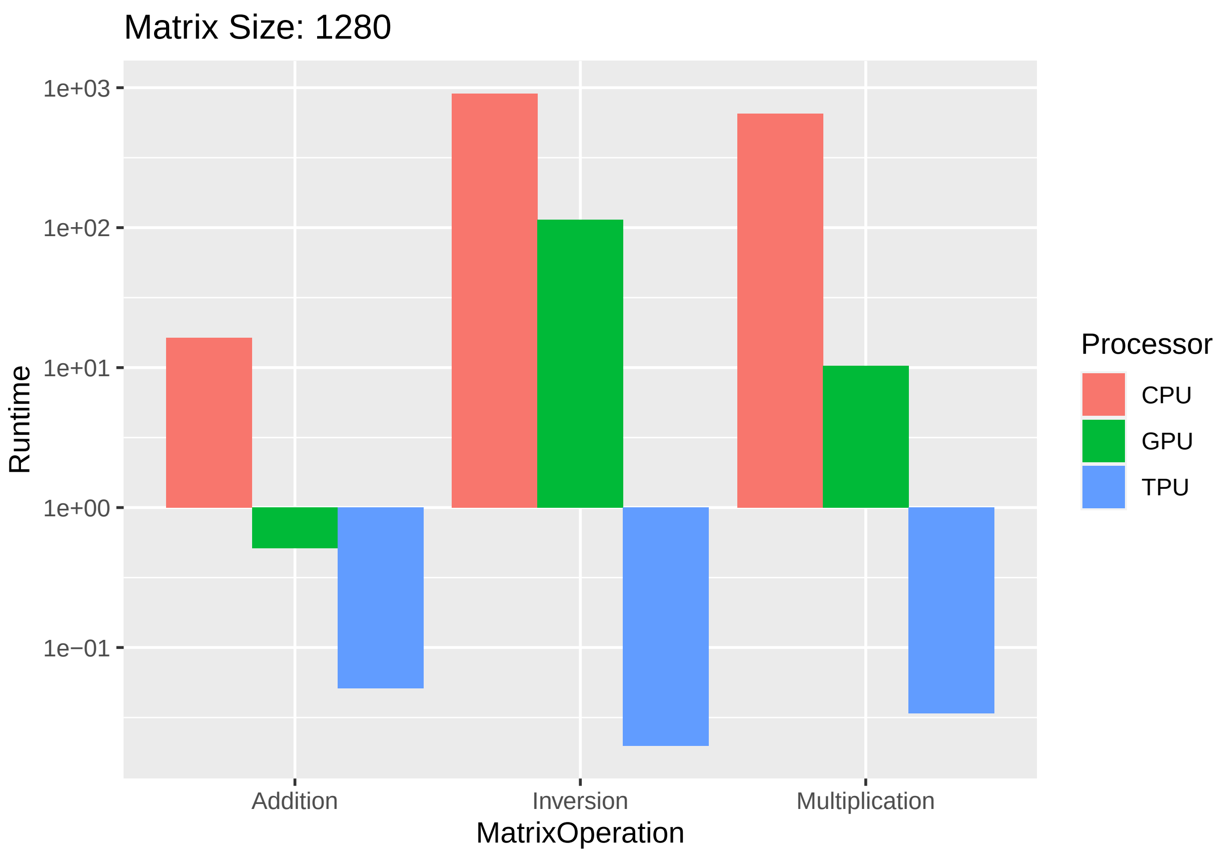


Matrix Size: 160

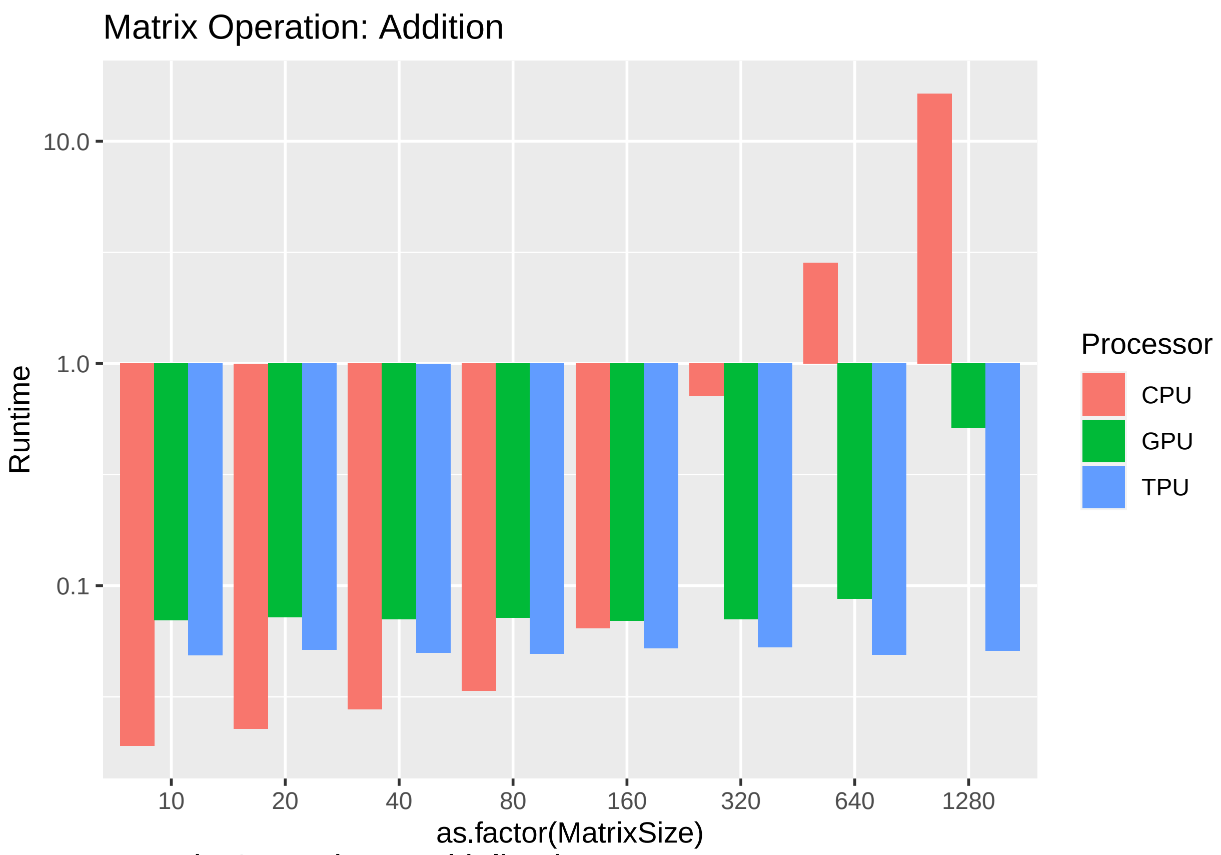


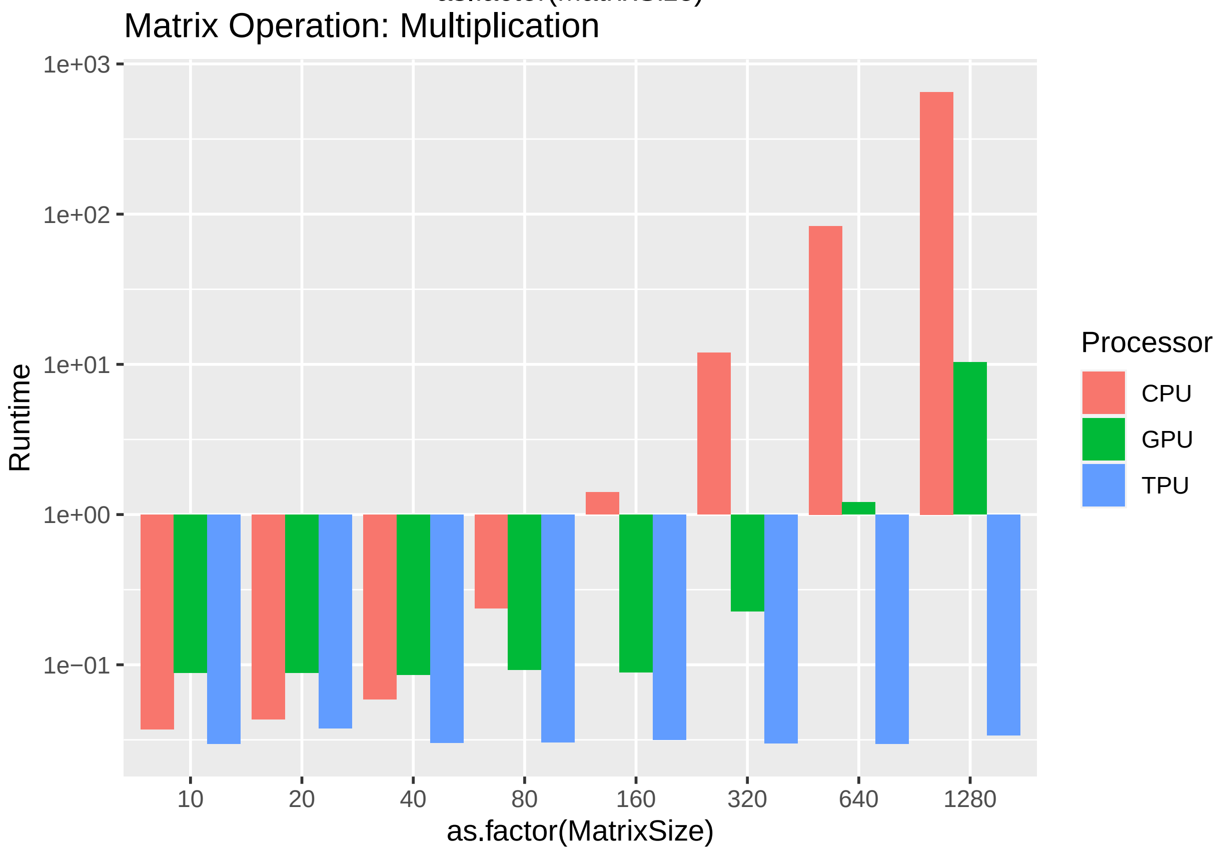

Matrix Size: 320
Runtime
10.0
1.0
0.1
Addition
Inversion
Multiplication
MatrixOperation
Processor
CPU
GPU
TPU

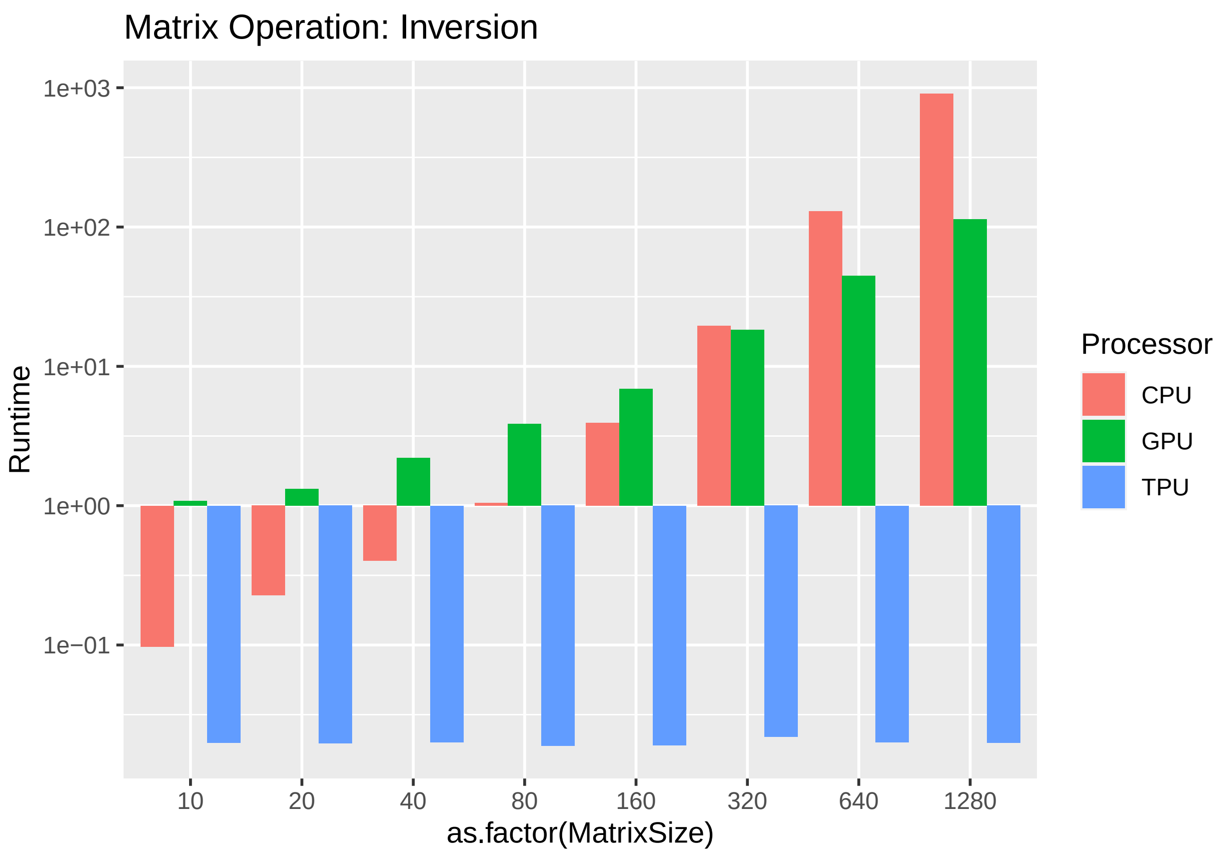

Matrix Size: 640
Runtime
100.0
10.0
1.0
0.1
Addition
Inversion
Multiplication
MatrixOperation
Processor
CPU
GPU
TPU



```r
for(operation in unique(data$MatrixOperation)){
  p <- data %>%
    dplyr::filter(MatrixOperation == operation) %>%
    group_by(MatrixSize, MatrixOperation, Processor) %>%
    summarize(Runtime = mean(Runtime)) %>%
    ggplot(aes(x = as.factor(MatrixSize), y = Runtime)) +
    #geom_boxplot(aes(fill = as.factor(Processor))) +
    geom_bar(aes(fill = as.factor(Processor)), stat = "identity",
             alpha = 1, position = "dodge") +
    scale_y_log10() +
    ggtitle(paste0("Matrix Operation: ", operation)) +
    #facet_wrap( ~ MatrixOperation, scales = "free", nrow = 1) +
    guides(fill=guide_legend(title = "Processor"))
  print(p)
}
```


Matrix Operation: Addition
Runtime
10.0
1.0
0.1
10
20
40
80
160
320
640
1280
as.factor(MatrixSize)
Processor
CPU
GPU
TPU


Matrix Operation: Multiplication
Runtime
1e+03
1e+02
1e+01
1e+00
1e−01
10
20
40
80
160
320
640
1280
as.factor(MatrixSize)
Processor
CPU
GPU
TPU



Pros & Cons of Each Processor

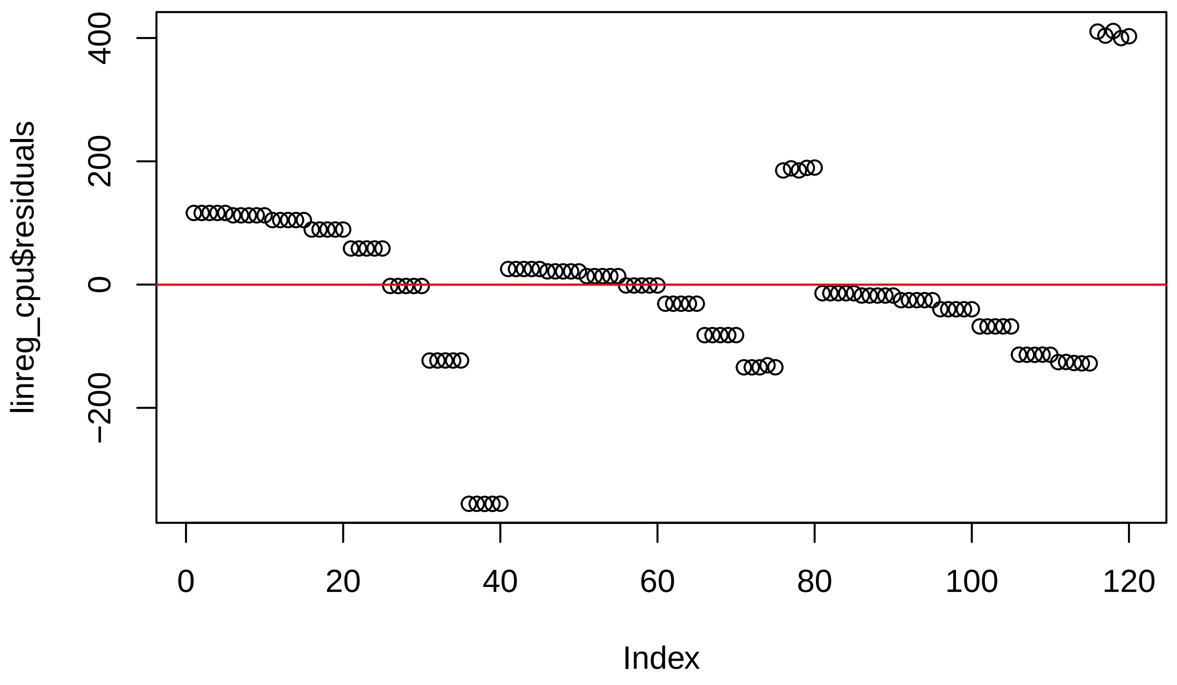
CPU:

```
df_cpu <- data[data$Processor == "CPU",]
linreg_cpu <- lm(Runtime ~ MatrixSize + as.factor(MatrixOperation),
                 data = df_cpu)
summary(linreg_cpu) %>% print()
```

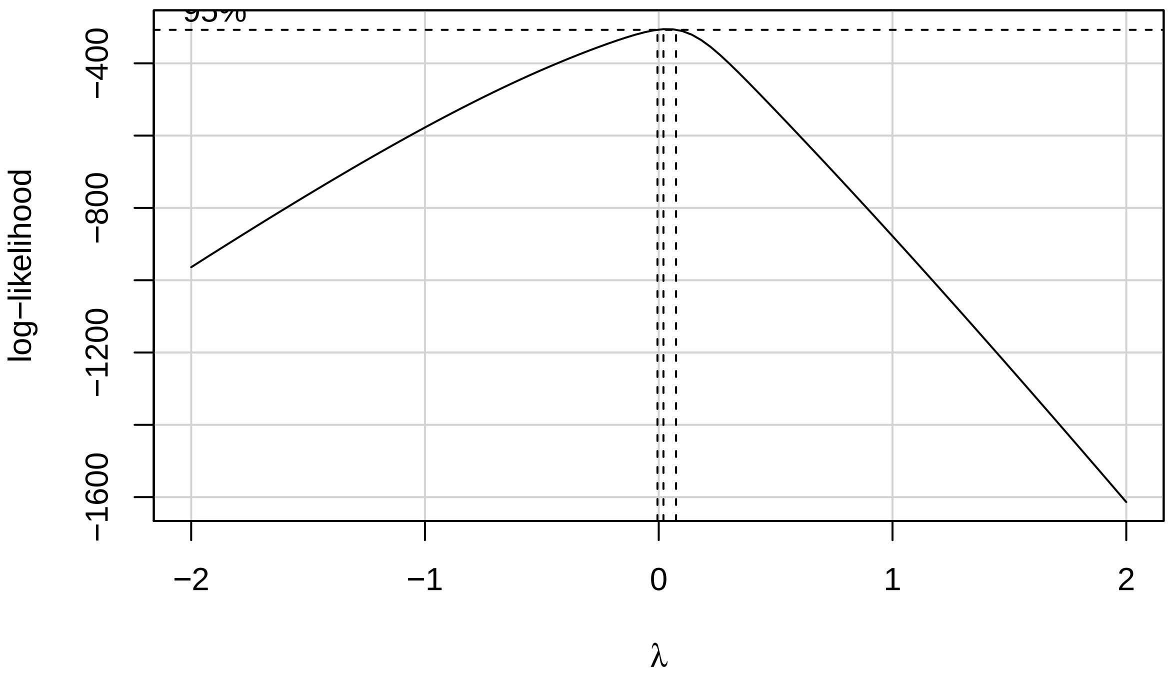
```
##
## Call:
## lm(formula = Runtime ~ MatrixSize + as.factor(MatrixOperation),
##     data = df_cpu)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -355.80  -71.41   -8.16   66.26  411.40
##
## Coefficients:
##                                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                             -120.02352   24.18254  -4.963 2.40e-06
## MatrixSize                                 0.38443    0.03081  12.477  < 2e-16
## as.factor(MatrixOperation)Inversion      130.34056   31.25182   4.171 5.89e-05
## as.factor(MatrixOperation)Multiplication  90.96003   31.25182   2.911  0.00433
##
```

```
## (Intercept)                               ***
## MatrixSize                                ***
## as.factor(MatrixOperation)Inversion      ***
## as.factor(MatrixOperation)Multiplication **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 139.8 on 116 degrees of freedom
## Multiple R-squared:    0.6,  Adjusted R-squared:  0.5896
## F-statistic: 57.99 on 3 and 116 DF,  p-value: < 2.2e-16
```

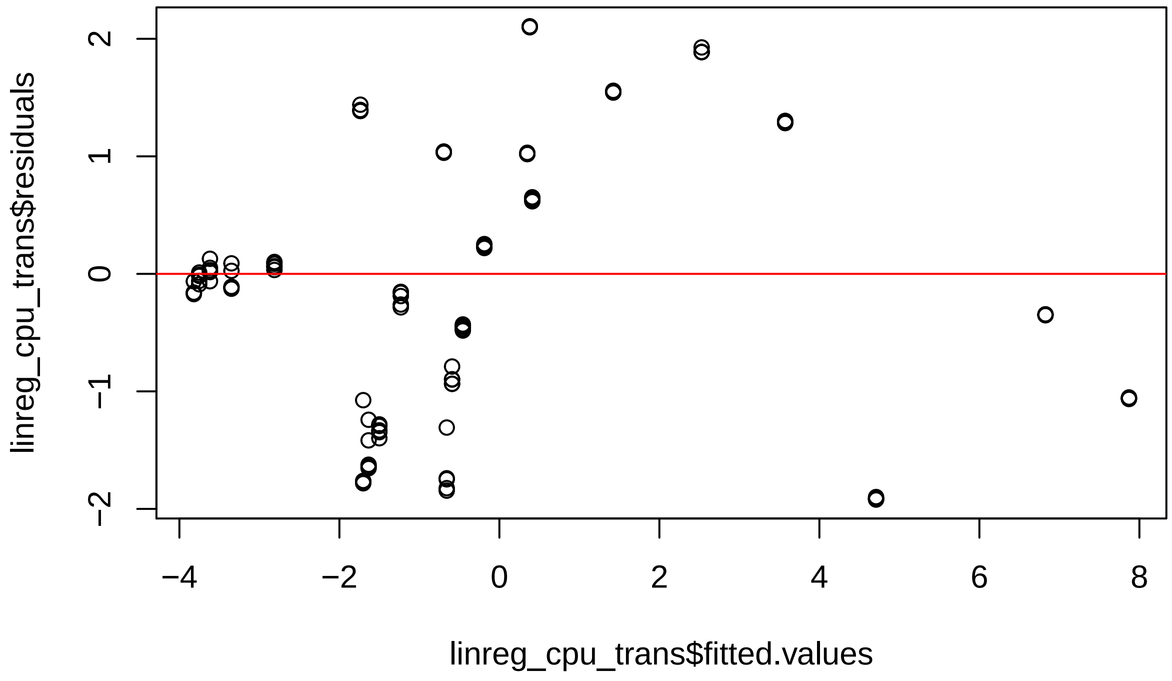
```r
plot(linreg_cpu$residuals)
abline(h=0, col="red")
```



```r
boxCox(linreg_cpu)
```



```
linreg_cpu_trans <- lm(log(Runtime) ~ MatrixSize + as.factor(MatrixOperation),
                       data = df_cpu)
plot(linreg_cpu_trans$fitted.values, linreg_cpu_trans$residuals)
abline(h=0, col="red")
```

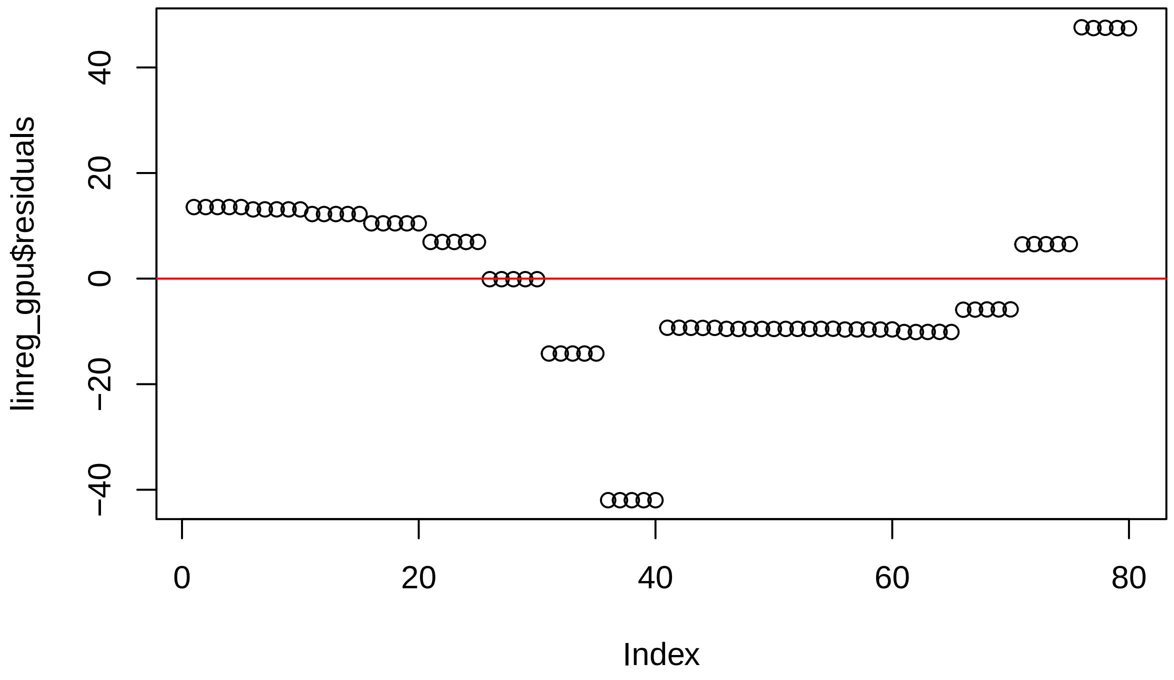


GPU:

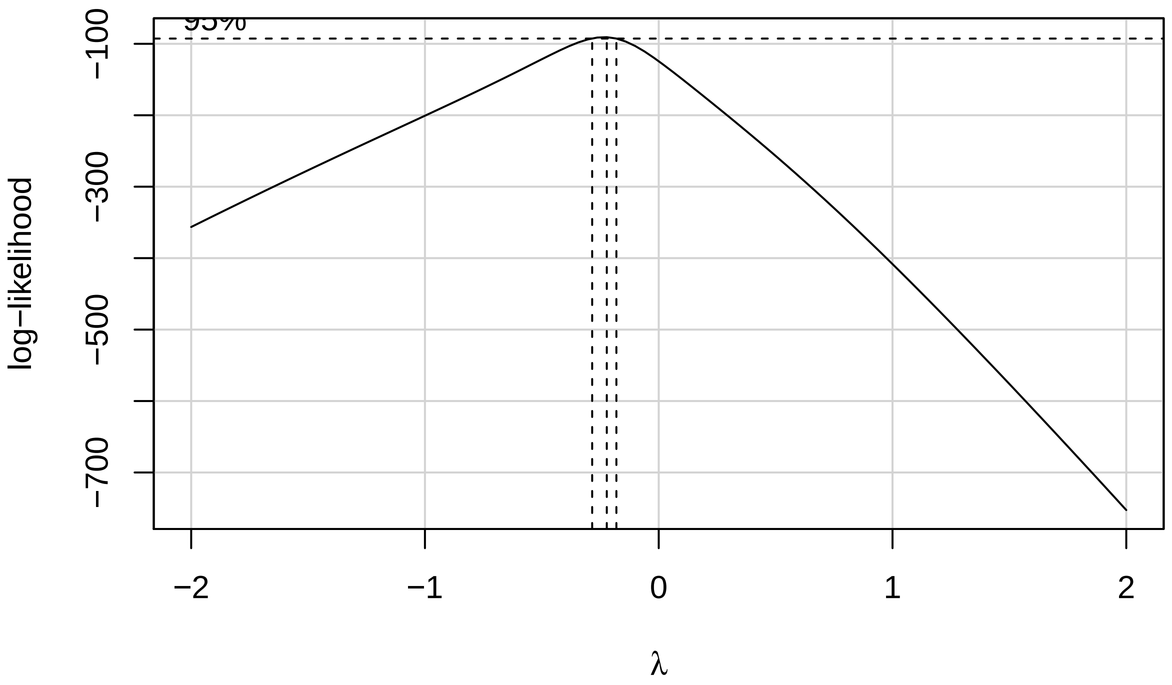
```
df_gpu <- data[(data$Processor == "GPU") & (data$MatrixOperation != "Multiplication"),]
linreg_gpu <- lm(Runtime ~ MatrixSize + as.factor(MatrixOperation),
                       data = df_gpu)
summary(linreg_gpu) %>% print()
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixSize + as.factor(MatrixOperation),
##     data = df_gpu)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -41.981  -9.566  -2.974  10.907  47.604
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        -13.920265   3.377960  -4.121 9.43e-05 ***
## MatrixSize                           0.044073   0.005066   8.699 4.57e-13 ***
## as.factor(MatrixOperation)Inversion  23.893954   4.195848   5.695 2.15e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.76 on 77 degrees of freedom
## Multiple R-squared:  0.584,  Adjusted R-squared:  0.5732
## F-statistic: 54.05 on 2 and 77 DF,  p-value: 2.161e-15
```

```
plot(linreg_gpu$residuals)
abline(h=0, col="red")
```



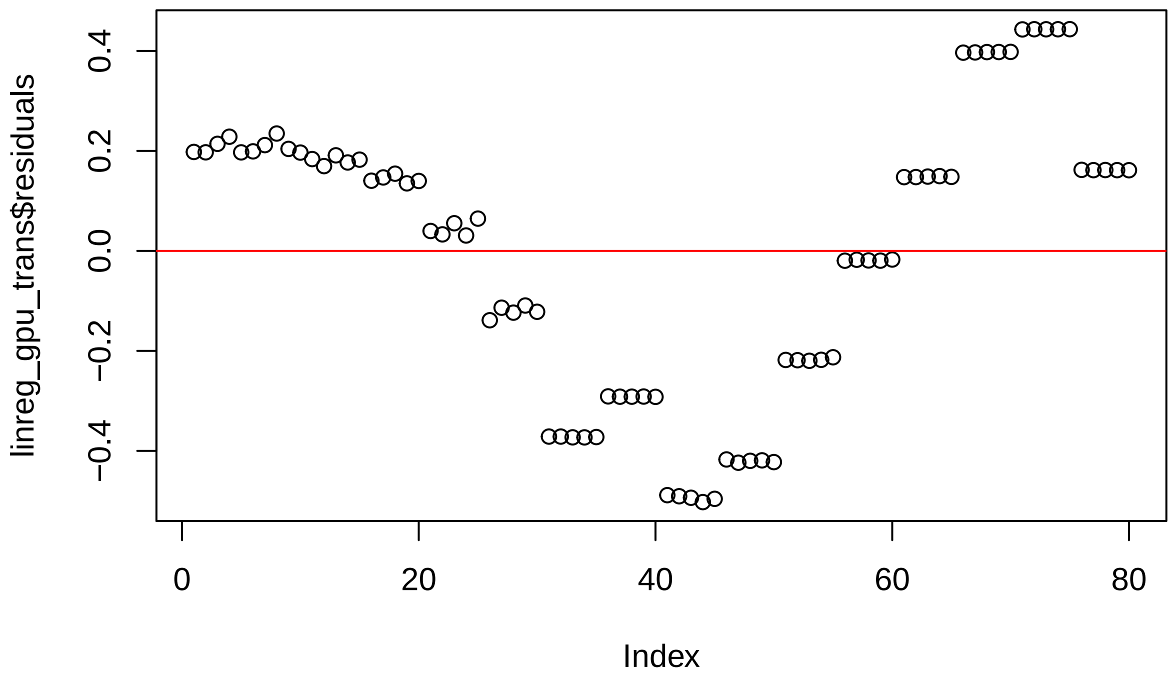
```
boxCox(linreg_gpu)
```



```
linreg_gpu_trans <- lm(log10(Runtime) ~ MatrixSize +
                         as.factor(MatrixOperation), data = df_gpu)
summary(linreg_gpu_trans)
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ MatrixSize + as.factor(MatrixOperation),
##     data = df_gpu)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.50239 -0.23761  0.09982  0.19255  0.44349
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        -1.374e+00  5.133e-02  -26.77   <2e-16 ***
## MatrixSize                          1.075e-03  7.699e-05   13.96   <2e-16 ***
## as.factor(MatrixOperation)Inversion  1.894e+00  6.376e-02   29.70   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2851 on 77 degrees of freedom
## Multiple R-squared:  0.9333, Adjusted R-squared:  0.9315
## F-statistic: 538.4 on 2 and 77 DF,  p-value: < 2.2e-16
```

```
plot(linreg_gpu_trans$residuals)
abline(h=0, col="red")
```

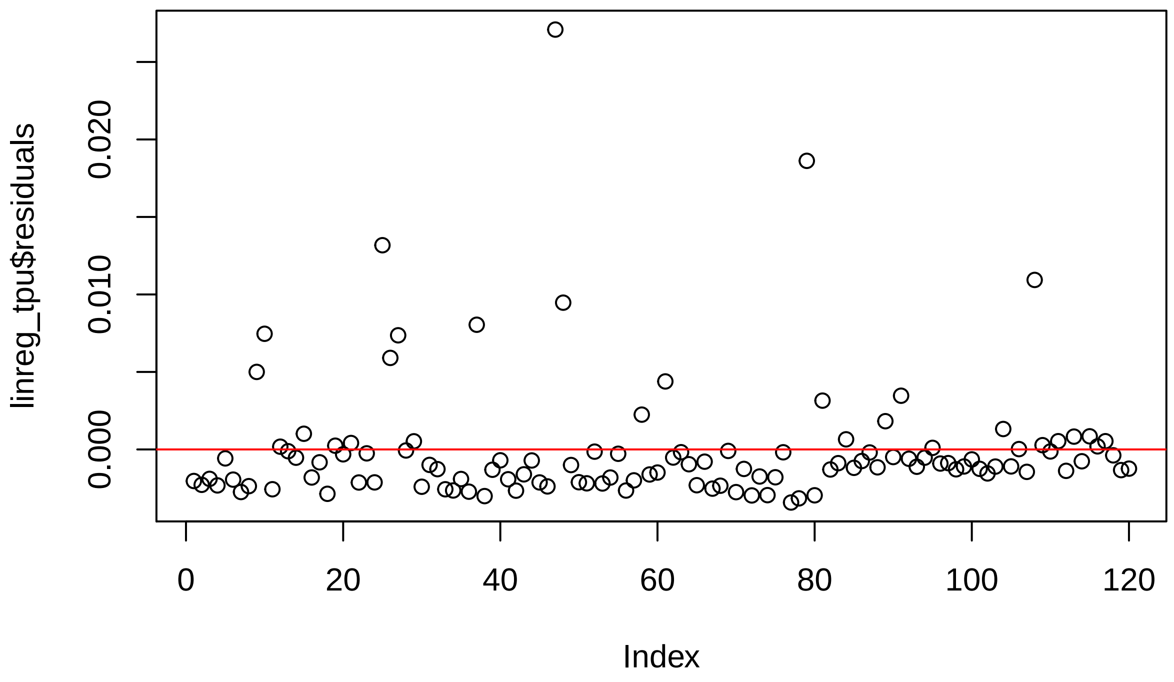


TPU:

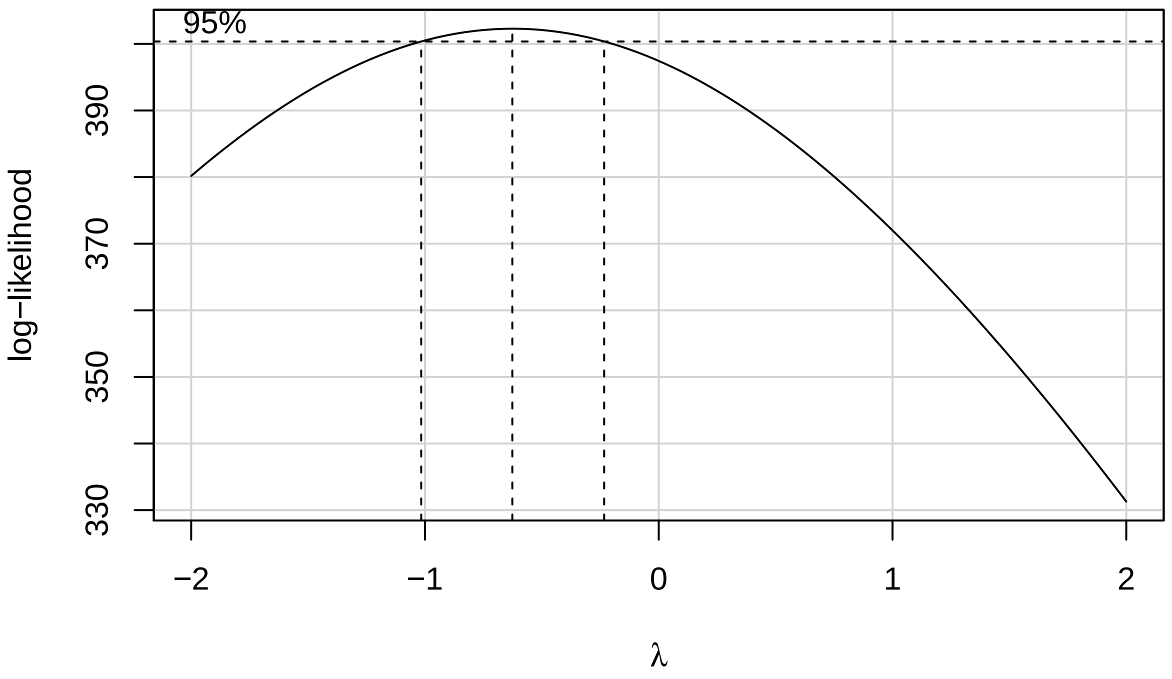
```r
df_tpu <- data[data$Processor == "TPU",]
linreg_tpu <- lm(Runtime ~ MatrixSize + as.factor(MatrixOperation),
                 data = df_tpu)
summary(linreg_tpu) %>% print()
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixSize + as.factor(MatrixOperation),
##     data = df_tpu)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0034224 -0.0020056 -0.0010012 -0.0000435  0.0270875
##
## Coefficients:
##                                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                                5.039e-02  7.237e-04   69.62   <2e-16
## MatrixSize                                 3.595e-07  9.221e-07    0.39    0.697
## as.factor(MatrixOperation)Inversion      -3.058e-02  9.353e-04  -32.70   <2e-16
## as.factor(MatrixOperation)Multiplication -1.877e-02  9.353e-04  -20.07   <2e-16
##
## (Intercept)                              ***
## MatrixSize
## as.factor(MatrixOperation)Inversion      ***
## as.factor(MatrixOperation)Multiplication ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.004183 on 116 degrees of freedom
## Multiple R-squared:  0.9036, Adjusted R-squared:  0.9012
## F-statistic: 362.6 on 3 and 116 DF,  p-value: < 2.2e-16
```

```
plot(linreg_tpu$residuals)
abline(h=0, col="red")
```



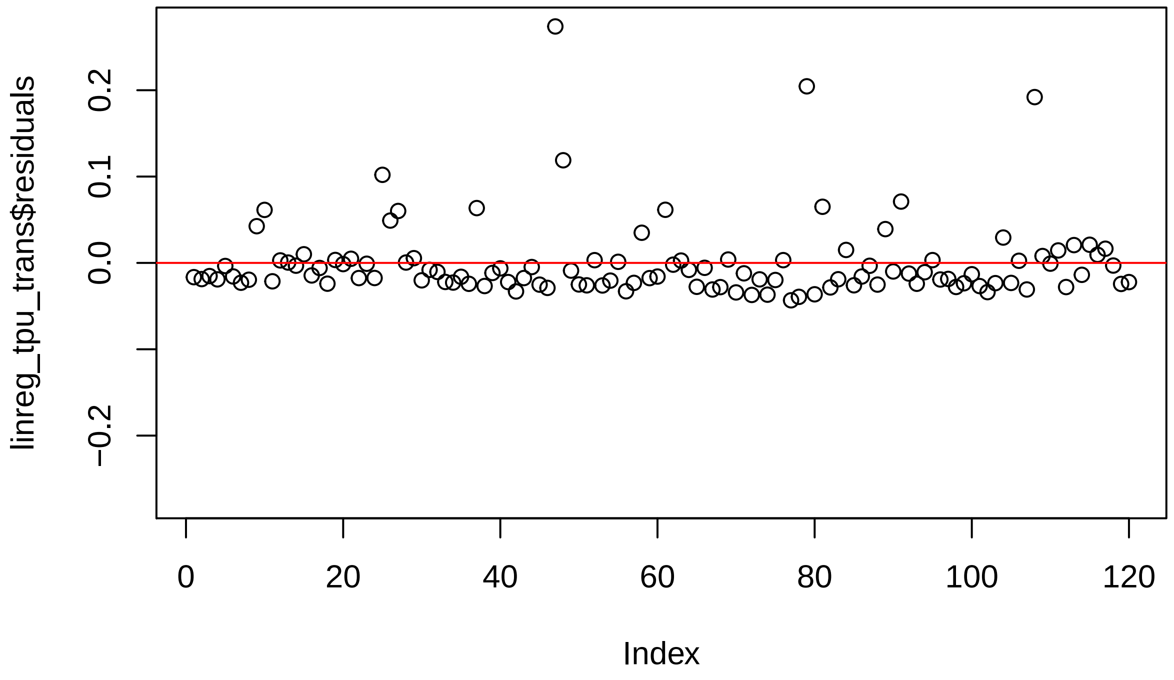
```
boxCox(linreg_tpu)
```



```
linreg_tpu_trans <- lm(log10(Runtime) ~ MatrixSize +
                        as.factor(MatrixOperation), data = df_tpu)
summary(linreg_tpu_trans) %>% print()
```

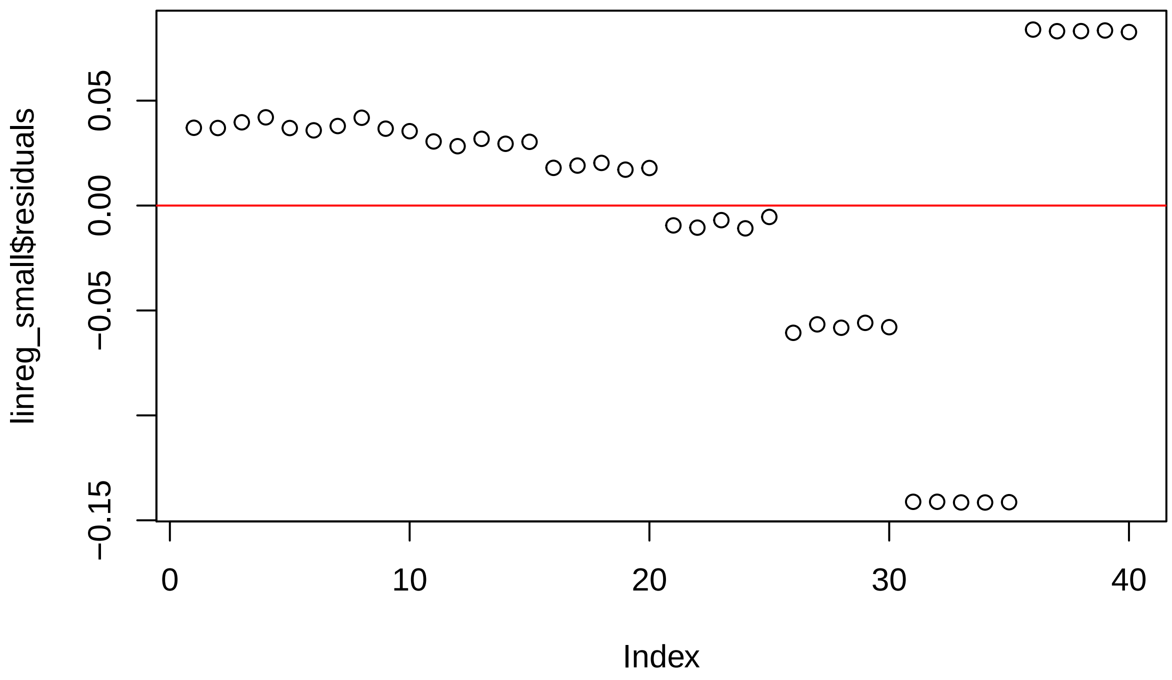
```
##
## Call:
## lm(formula = log10(Runtime) ~ MatrixSize + as.factor(MatrixOperation),
##     data = df_tpu)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.043314 -0.023467 -0.014163  0.003178  0.273842
##
## Coefficients:
##                                           Estimate Std. Error  t value
## (Intercept)                             -1.299e+00  8.067e-03 -161.041
## MatrixSize                               4.439e-06  1.028e-05    0.432
## as.factor(MatrixOperation)Inversion     -4.050e-01  1.043e-02  -38.852
## as.factor(MatrixOperation)Multiplication -2.061e-01  1.043e-02  -19.770
##                                         Pr(>|t|)
## (Intercept)                               <2e-16 ***
## MatrixSize                                 0.667
## as.factor(MatrixOperation)Inversion       <2e-16 ***
## as.factor(MatrixOperation)Multiplication   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04662 on 116 degrees of freedom
## Multiple R-squared:  0.9287, Adjusted R-squared:  0.9268
## F-statistic: 503.3 on 3 and 116 DF,  p-value: < 2.2e-16
```

```r
m <- max(abs(linreg_tpu_trans$residuals))
plot(linreg_tpu_trans$residuals, ylim=c(-m, m))
abline(h=0, col="red")
```

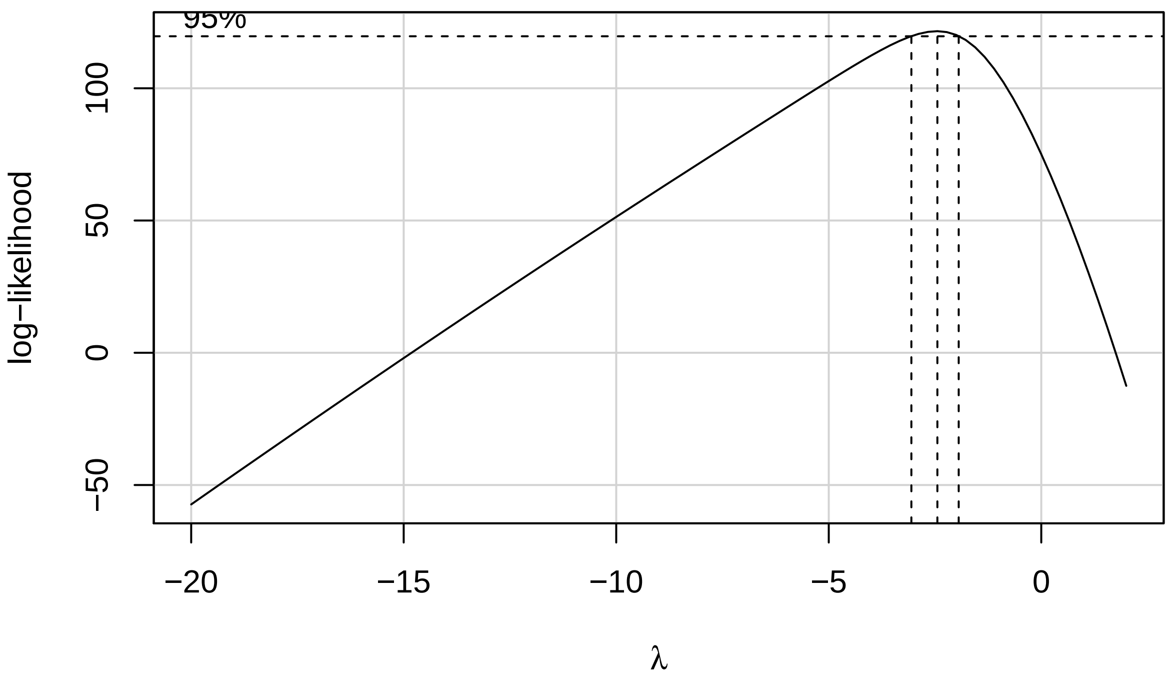


GPU Addition

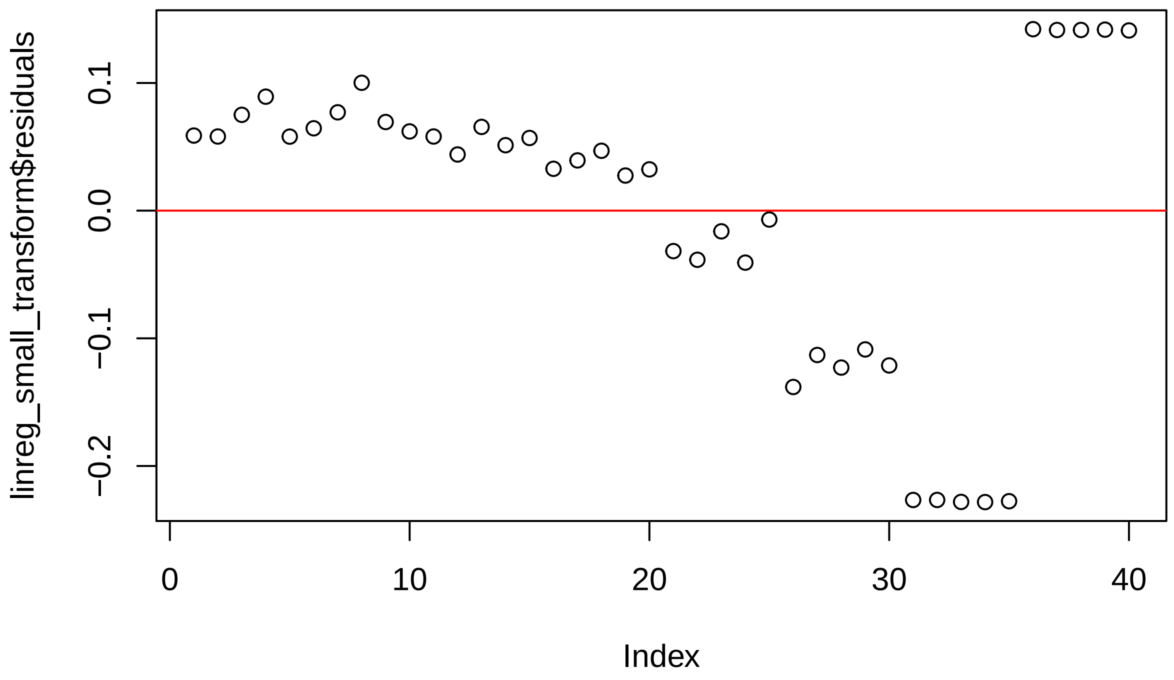
```r
df_small <- data[(data$Processor == "GPU") & (data$MatrixOperation == "Addition"),]
linreg_small <- lm(Runtime ~ MatrixSize, data = df_small)
plot(linreg_small$residuals)
abline(h=0, col="red")
```



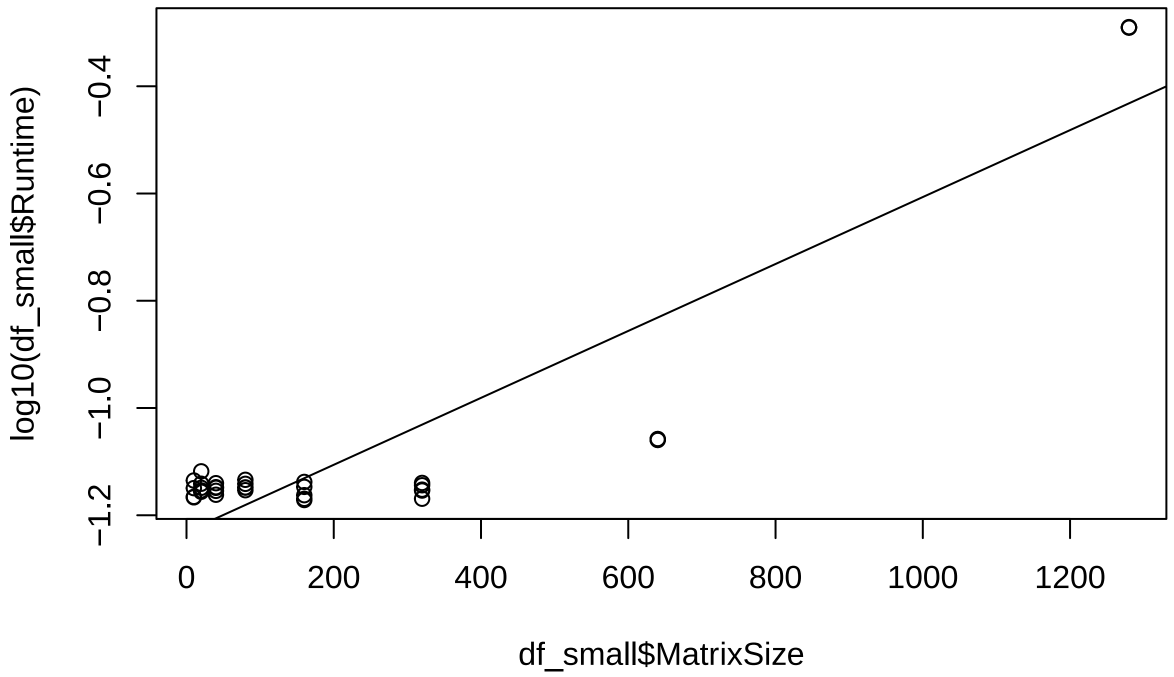
```r
boxCox(linreg_small, lambda = seq(-20, 2, 0.1))
```



```r
linreg_small_transform <- lm(log10(Runtime) ~ MatrixSize, data = df_small)
plot(linreg_small_transform$residuals)
abline(h=0, col="red")
```

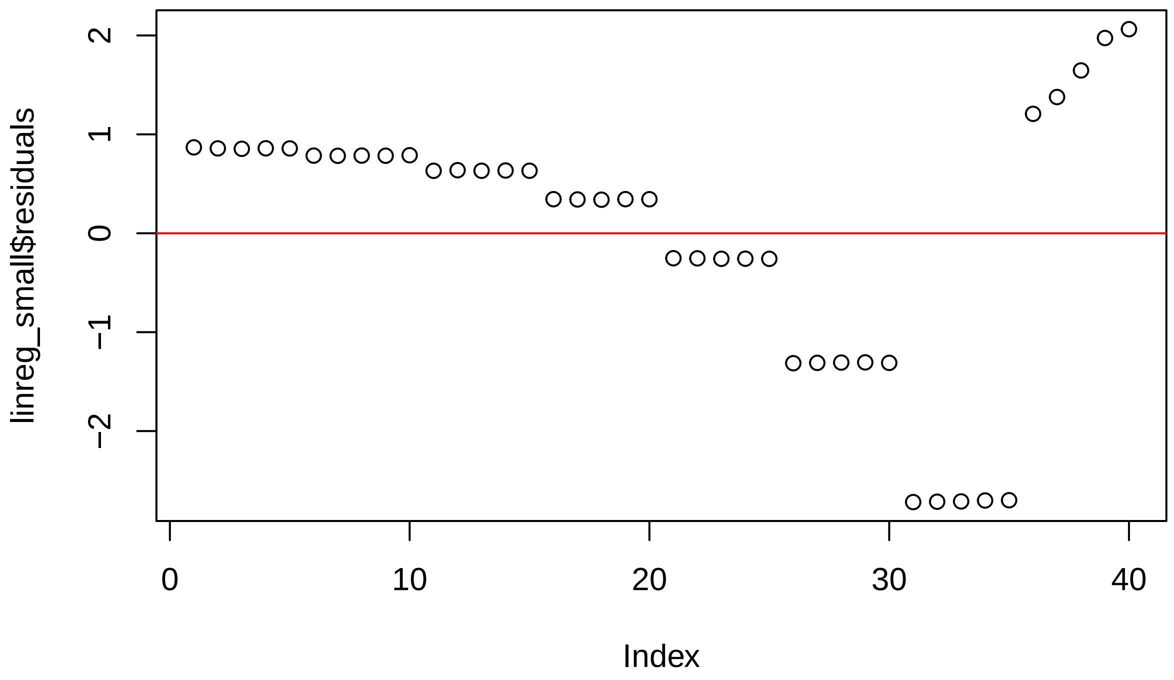


```
plot(df_small$MatrixSize, log10(df_small$Runtime))
abline(a=linreg_small_transform$coefficients[1],
       b=linreg_small_transform$coefficients[2])
```

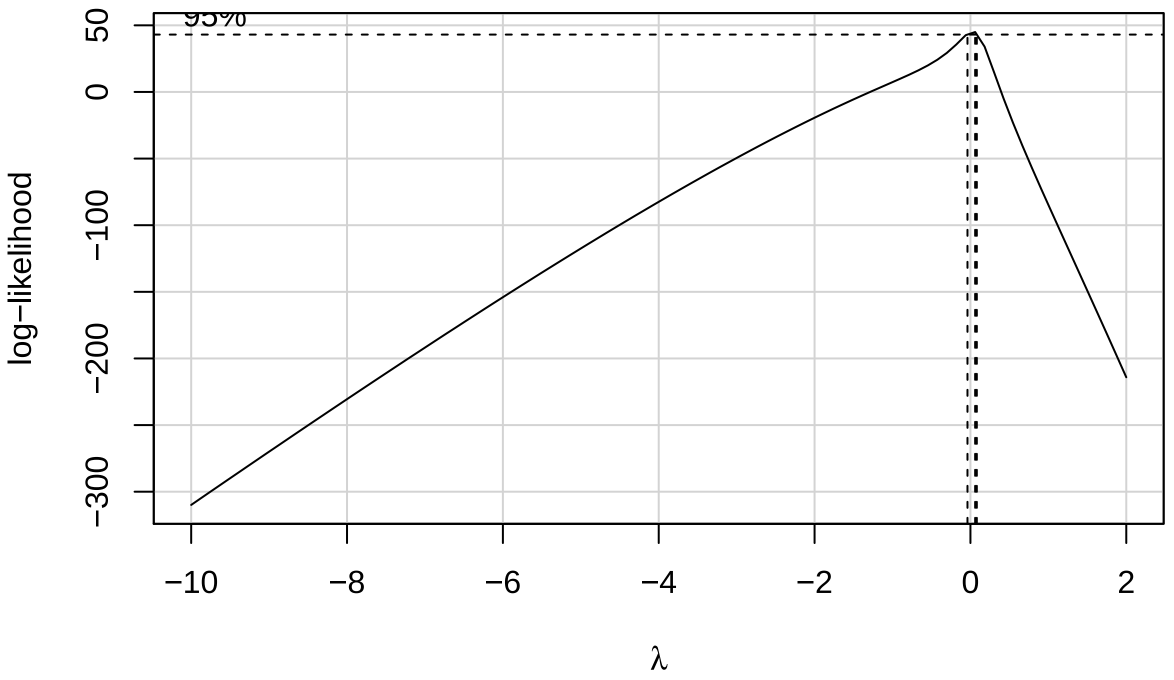


GPU Multiplication

```
df_small <- data[(data$Processor == "GPU") & (data$MatrixOperation == "Multiplication"),]
linreg_small <- lm(Runtime ~ MatrixSize, data = df_small)
plot(linreg_small$residuals)
abline(h=0, col="red")
```



```
boxCox(linreg_small, lambda = seq(-10, 2, 0.1))
```

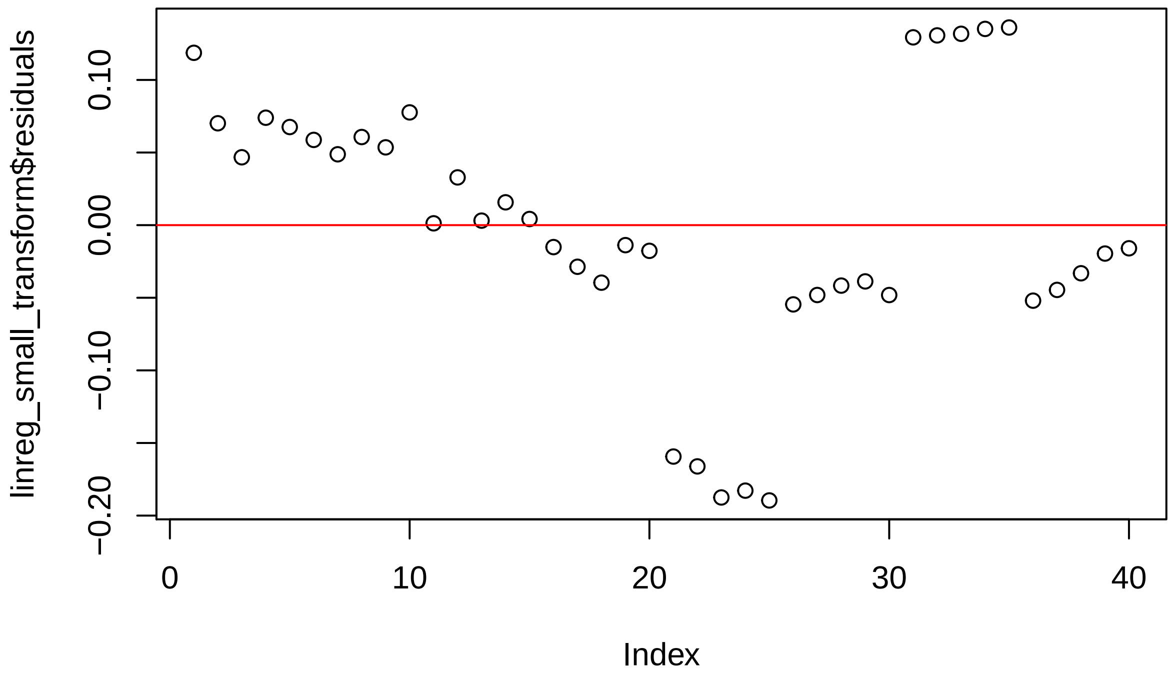


```
linreg_small_transform <- lm(log10(Runtime) ~ MatrixSize, data = df_small)
summary(linreg_small_transform)
```

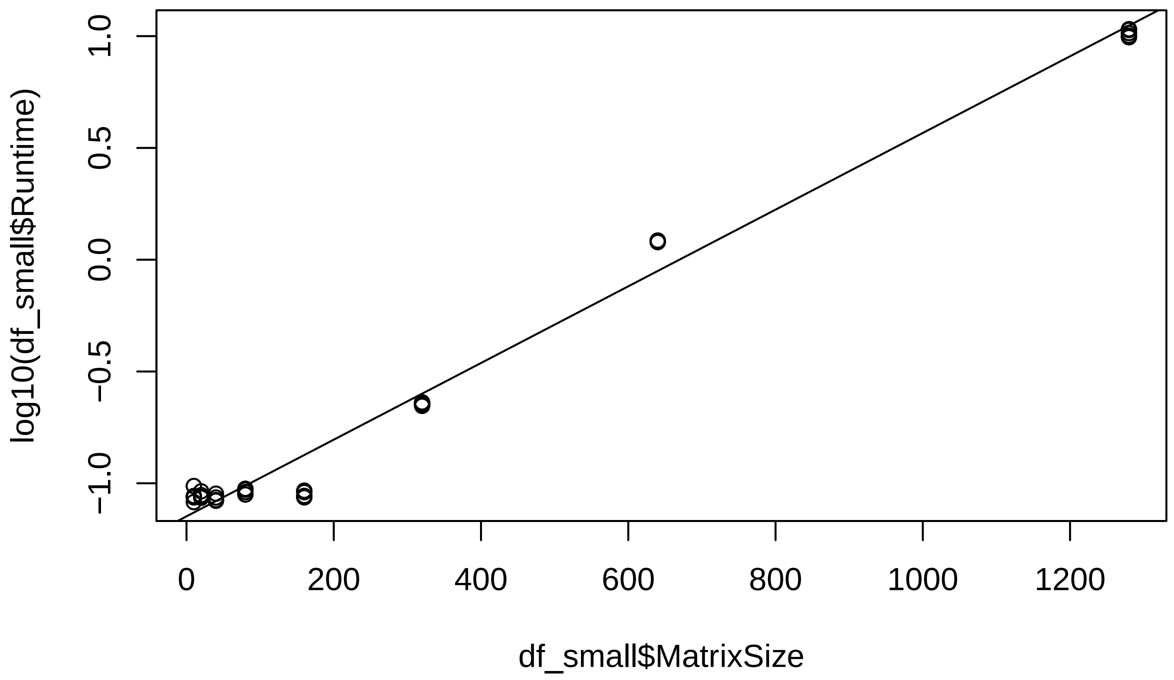
```
##
## Call:
## lm(formula = log10(Runtime) ~ MatrixSize, data = df_small)
##
```

```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.189553 -0.042371 -0.006256  0.062402  0.136090
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.148e+00  1.822e-02  -63.01   <2e-16 ***
## MatrixSize   1.715e-03  3.487e-05   49.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09131 on 38 degrees of freedom
## Multiple R-squared:  0.9845, Adjusted R-squared:  0.9841
## F-statistic:  2419 on 1 and 38 DF,  p-value: < 2.2e-16
```

```r
plot(linreg_small_transform$residuals)
abline(h=0, col="red")
```

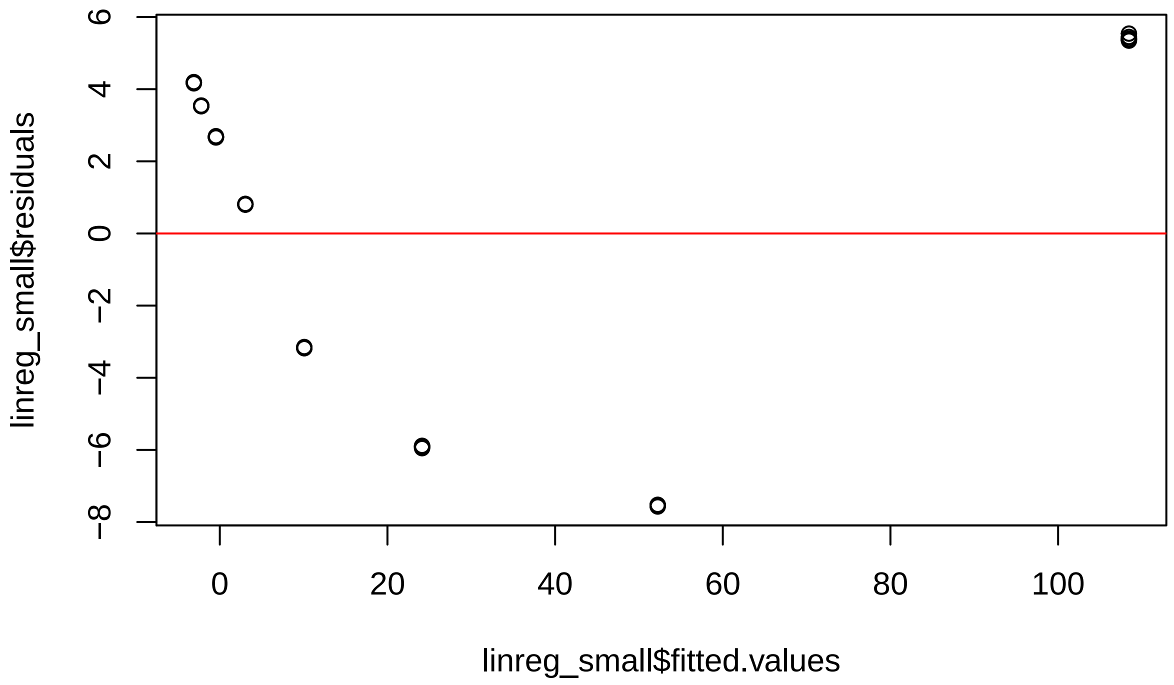


```r
plot(df_small$MatrixSize, log10(df_small$Runtime))
abline(a=linreg_small_transform$coefficients[1],
       b=linreg_small_transform$coefficients[2])
```

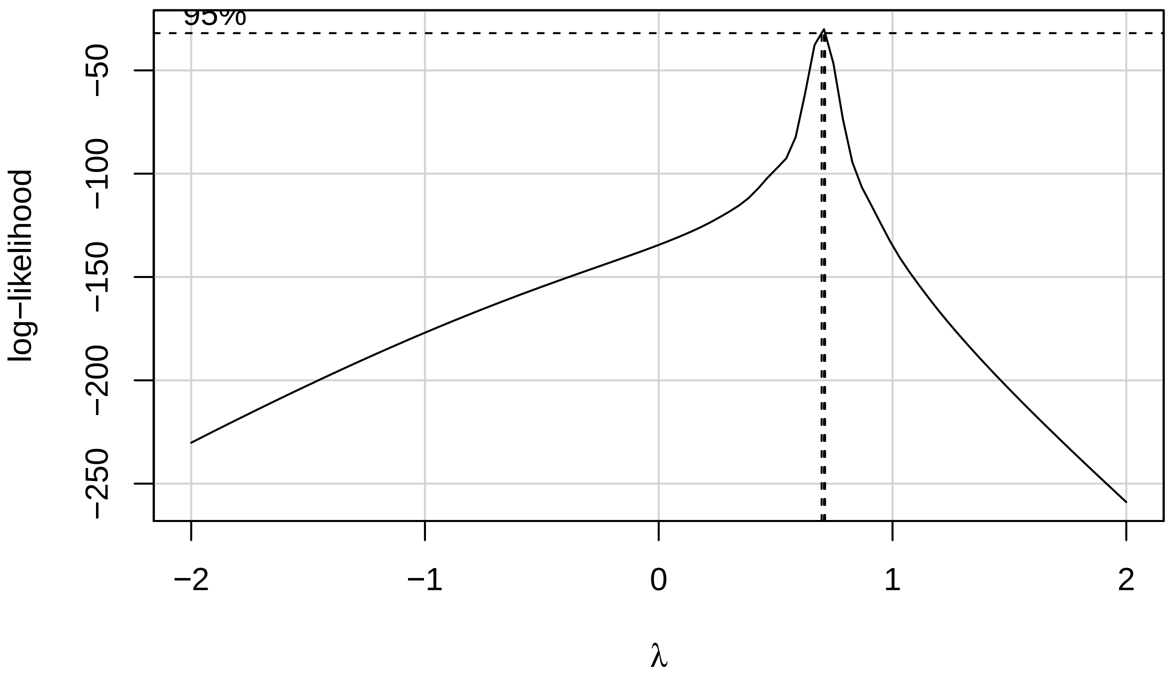


GPU Inversion

```
df_small <- data[(data$Processor == "GPU") & (data$MatrixOperation == "Inversion"),]
linreg_small <- lm(Runtime ~ MatrixSize, data = df_small)
plot(linreg_small$fitted.values, linreg_small$residuals)
abline(h=0, col="red")
```



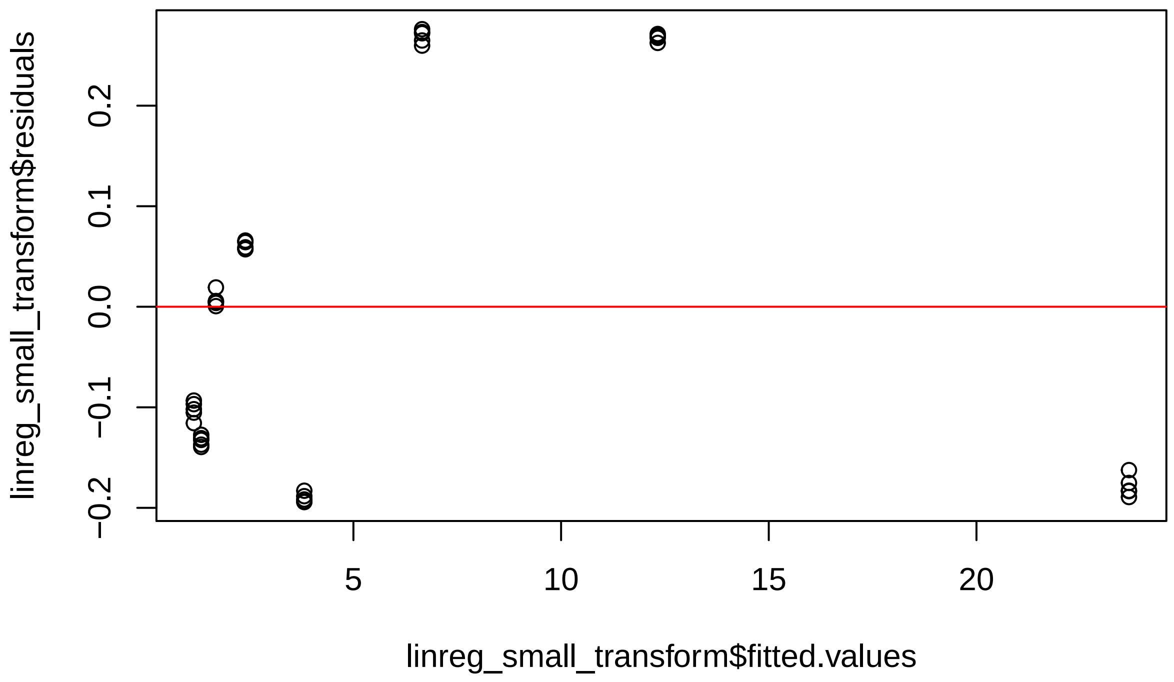
```
boxCox(linreg_small, lambda = seq(-2, 2, 0.1))
```



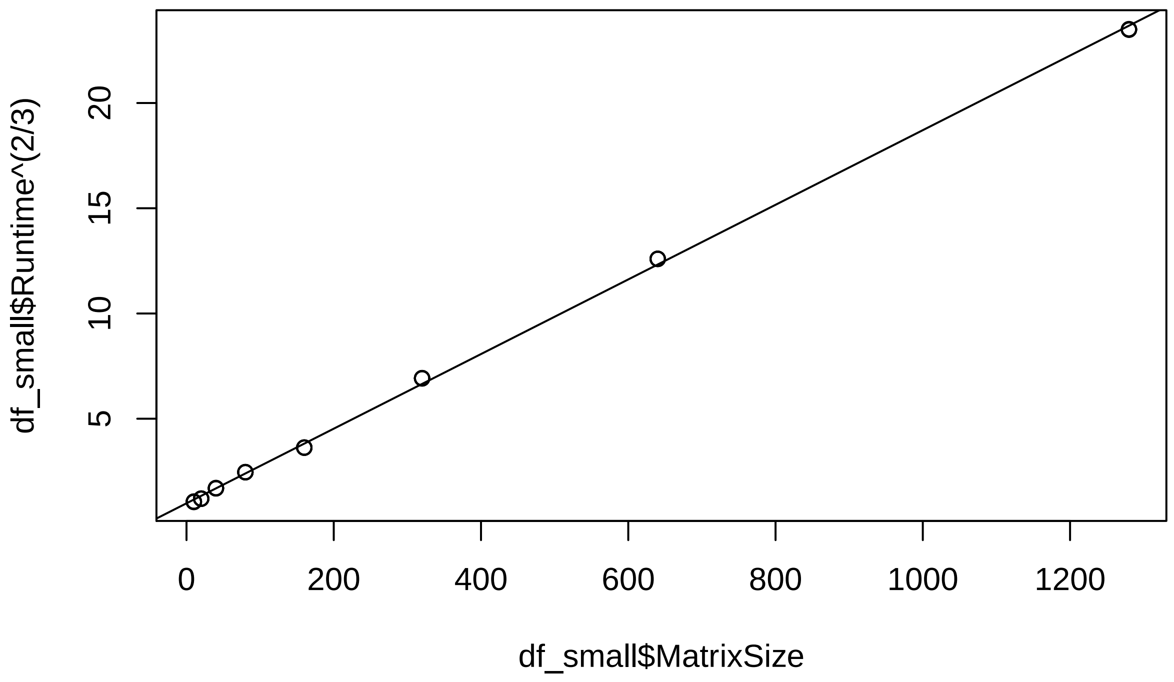
```
linreg_small_transform <- lm((Runtime^(2/3)) ~ MatrixSize, data = df_small)
summary(linreg_small_transform)
```

```
##
## Call:
## lm(formula = (Runtime^(2/3)) ~ MatrixSize, data = df_small)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19426 -0.14522 -0.04634  0.11432  0.27609
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 9.817e-01  3.581e-02   27.42   <2e-16 ***
## MatrixSize  1.773e-02  6.852e-05  258.71   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1794 on 38 degrees of freedom
## Multiple R-squared:  0.9994, Adjusted R-squared:  0.9994
## F-statistic: 6.693e+04 on 1 and 38 DF,  p-value: < 2.2e-16
```

```
#plot(linreg_small_transform)
plot(linreg_small_transform$fitted.values, linreg_small_transform$residuals)
abline(h=0, col="red")
```

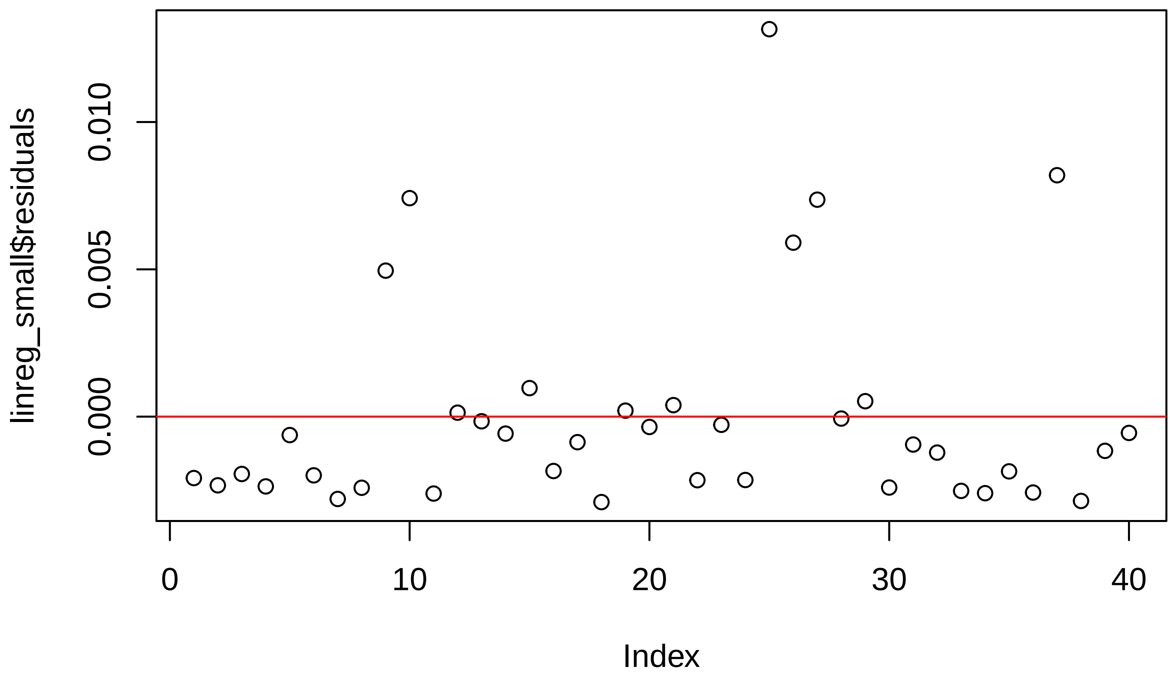


```r
plot(df_small$MatrixSize, df_small$Runtime^(2/3))
abline(a=linreg_small_transform$coefficients[1],
       b=linreg_small_transform$coefficients[2])
```

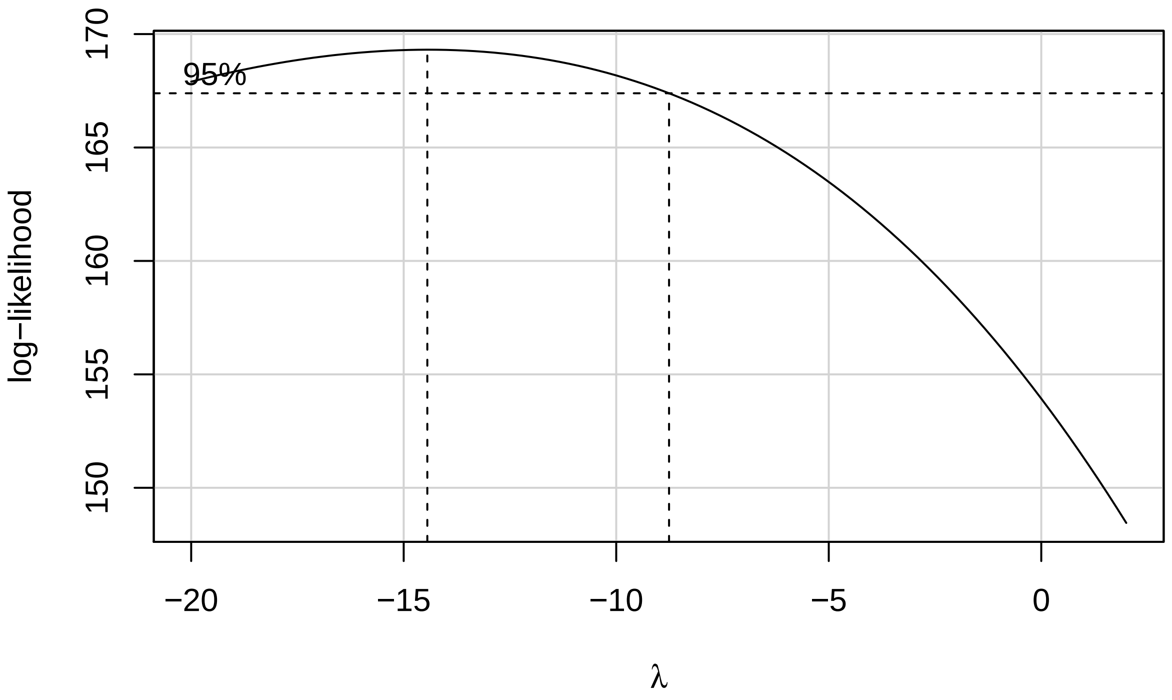


TPU Addition

```r
df_small <- data[(data$Processor == "TPU") & (data$MatrixOperation == "Addition"),]
linreg_small <- lm(Runtime ~ MatrixSize, data = df_small)
plot(linreg_small$residuals)
abline(h=0, col="red")
```



```
boxCox(linreg_small, lambda = seq(-20, 2, 0.1))
```

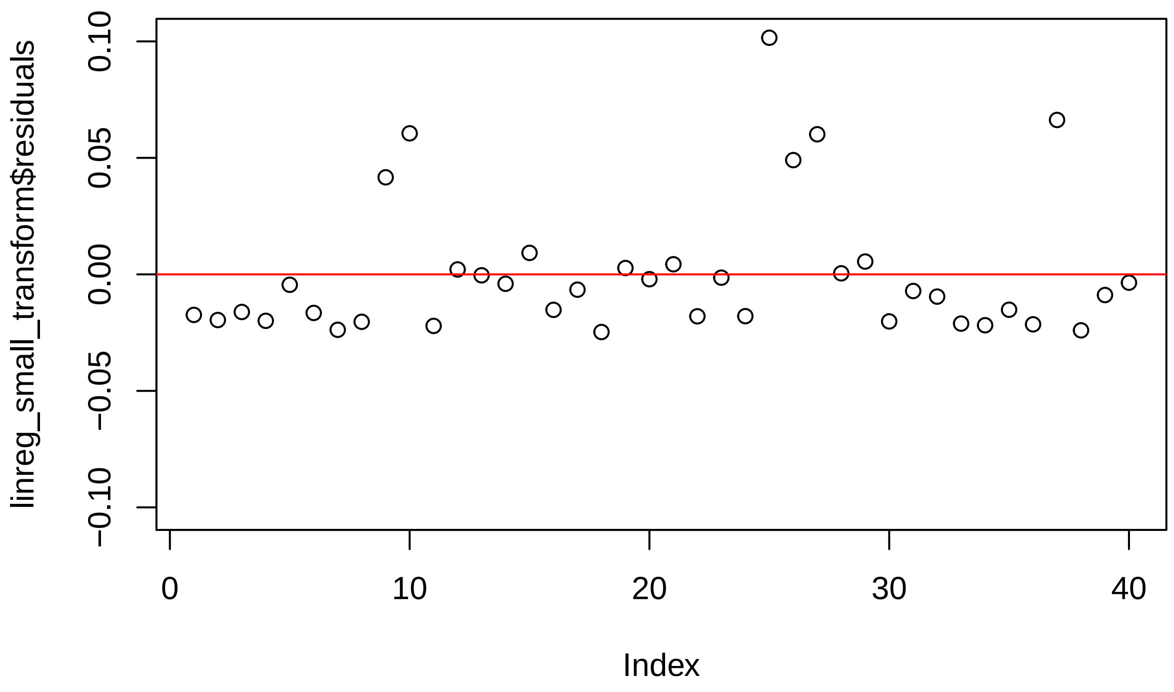


```
linreg_small_transform <- lm(log10(Runtime) ~ MatrixSize, data = df_small)
summary(linreg_small_transform)
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ MatrixSize, data = df_small)
##
```

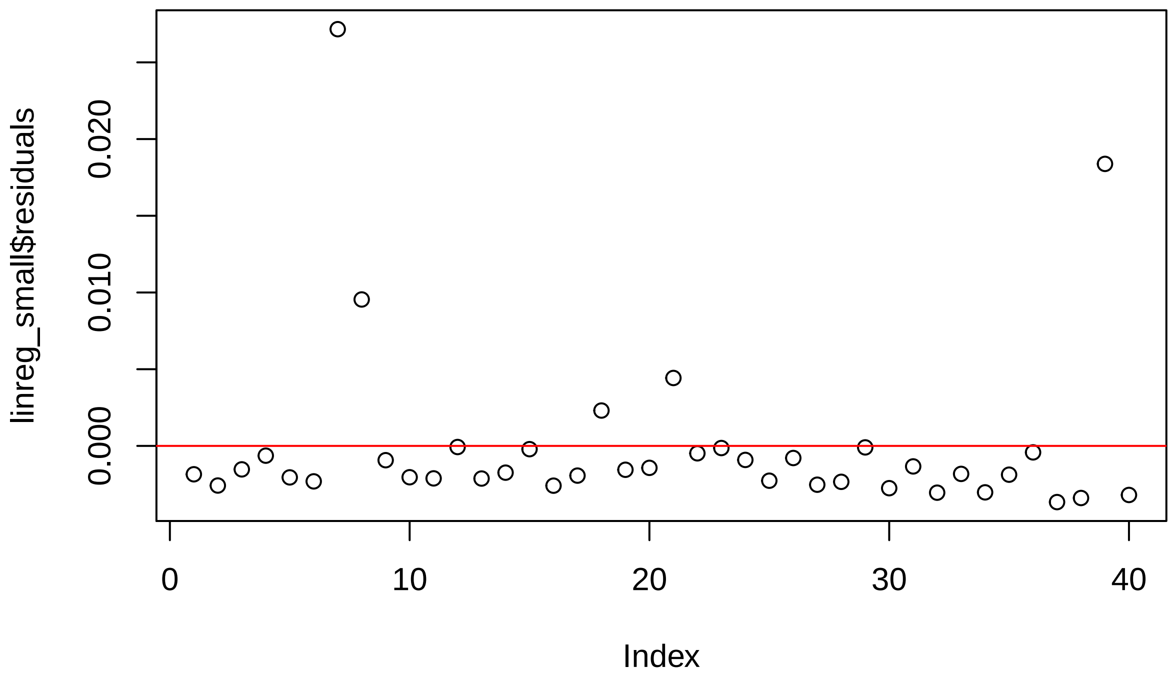
```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.024742 -0.019679 -0.007998  0.002266  0.101552
##
## Coefficients:
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -1.298e+00  5.948e-03 -218.276   <2e-16 ***
## MatrixSize   1.560e-06  1.138e-05    0.137    0.892
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02981 on 38 degrees of freedom
## Multiple R-squared:  0.0004943,  Adjusted R-squared:  -0.02581
## F-statistic: 0.01879 on 1 and 38 DF,  p-value: 0.8917
```

```r
m <- max(abs(linreg_small_transform$residuals))
plot(linreg_small_transform$residuals, ylim=c(-m, m))
abline(h=0, col="red")
```

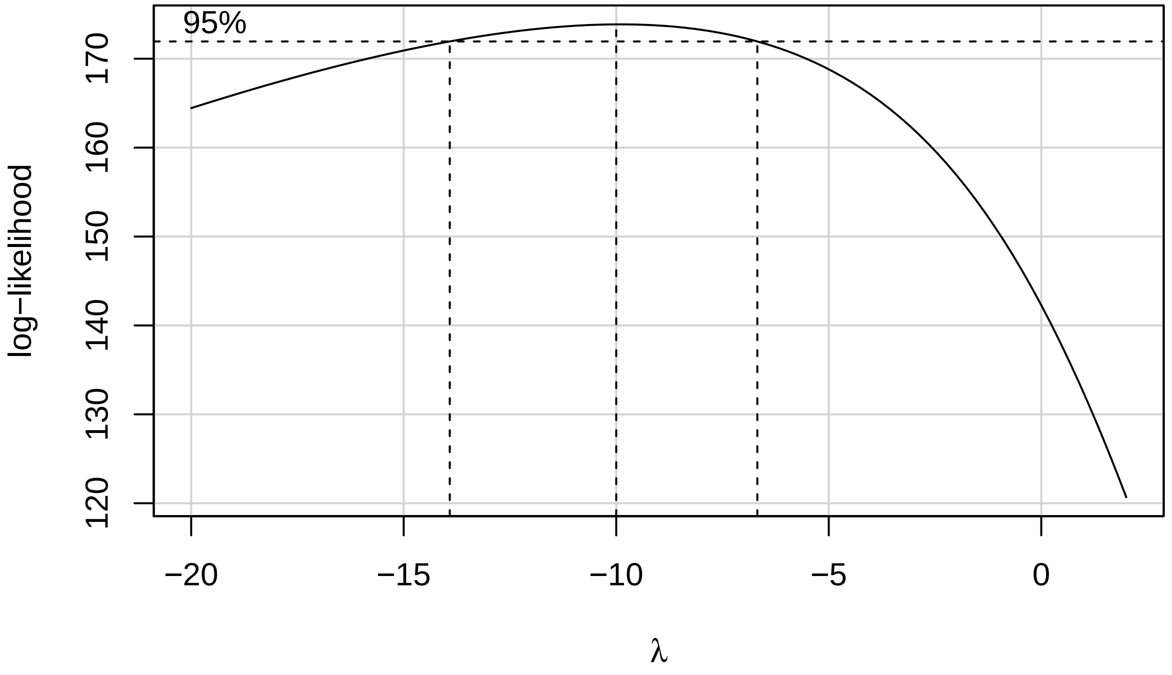


TPU Multiplication

```r
df_small <- data[(data$Processor == "TPU") & (data$MatrixOperation == "Multiplication"),]
linreg_small <- lm(Runtime ~ MatrixSize, data = df_small)
plot(linreg_small$residuals)
abline(h=0, col="red")
```



```
boxCox(linreg_small, lambda = seq(-20, 2, 0.1))
```

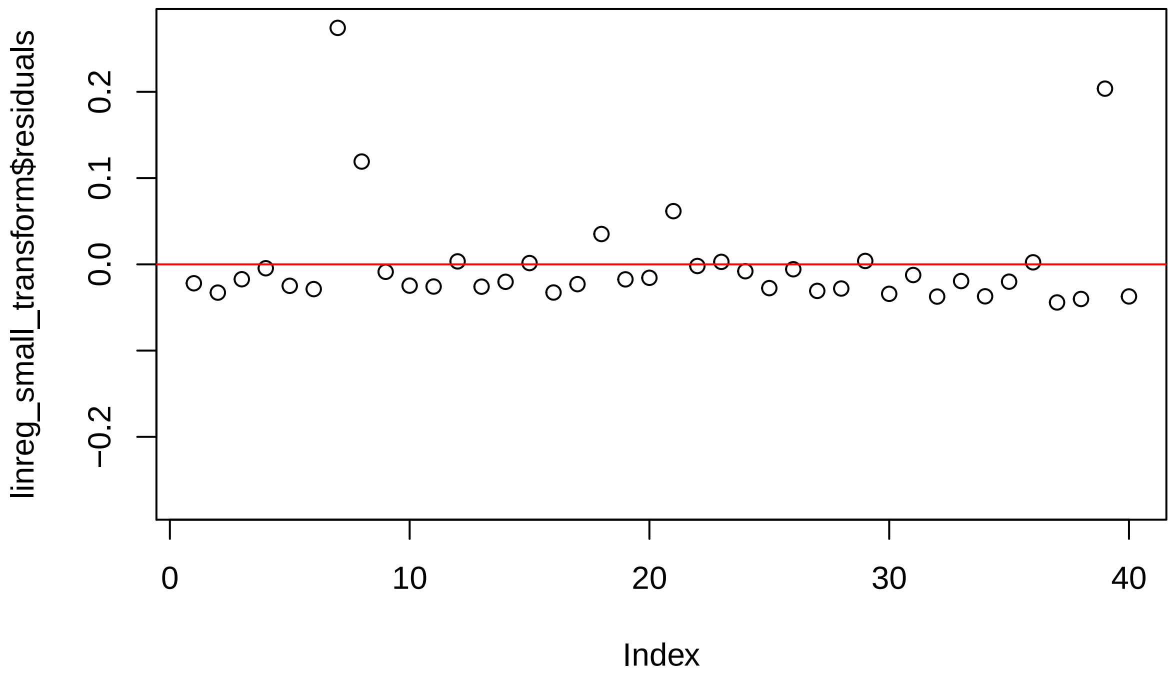


```
linreg_small_transform <- lm(log10(Runtime) ~ MatrixSize, data = df_small)
summary(linreg_small_transform)
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ MatrixSize, data = df_small)
##
```

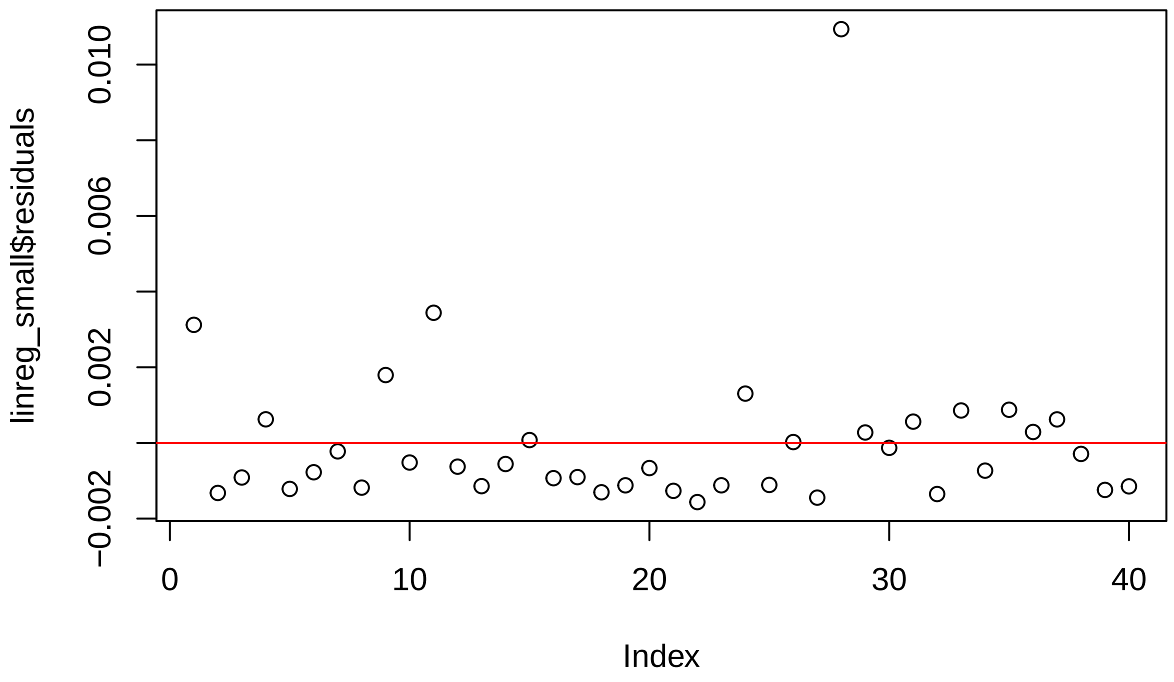
```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.044167 -0.028197 -0.019721 -0.001056  0.274107
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -1.506e+00  1.279e-02 -117.694   <2e-16 ***
## MatrixSize   5.326e-06  2.448e-05    0.218    0.829
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06411 on 38 degrees of freedom
## Multiple R-squared:  0.001244,   Adjusted R-squared:  -0.02504
## F-statistic: 0.04734 on 1 and 38 DF,  p-value: 0.8289
```

```
m <- max(abs(linreg_small_transform$residuals))
plot(linreg_small_transform$residuals, ylim=c(-m, m))
abline(h=0, col="red")
```

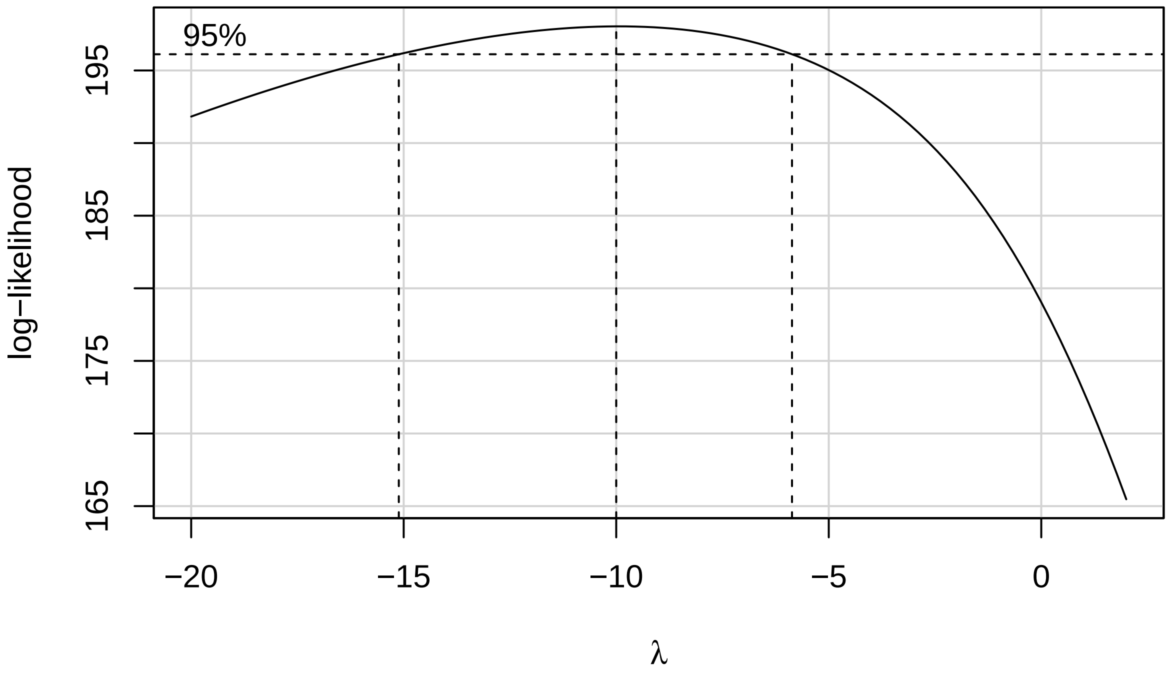


TPU Inversion

```
df_small <- data[(data$Processor == "TPU") & (data$MatrixOperation == "Inversion"),]
linreg_small <- lm(Runtime ~ MatrixSize, data = df_small)
plot(linreg_small$residuals)
abline(h=0, col="red")
```



```
boxCox(linreg_small, lambda = seq(-20, 2, 0.1))
```

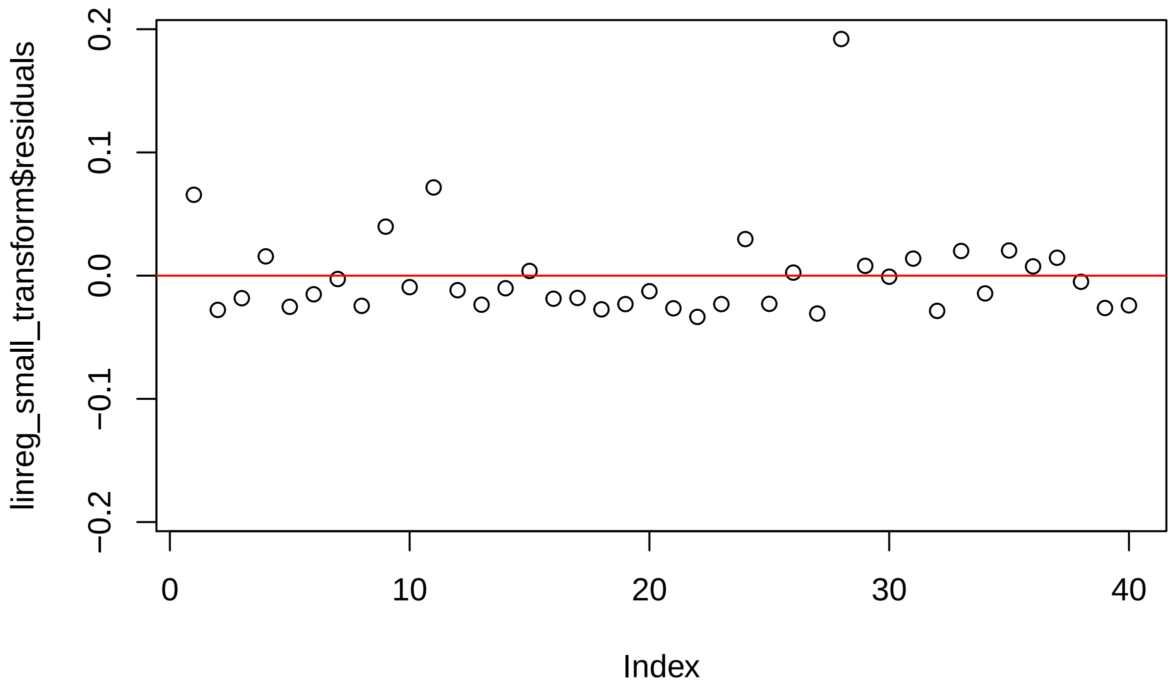


```
linreg_small_transform <- lm(log10(Runtime) ~ MatrixSize, data = df_small)
summary(linreg_small_transform)
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ MatrixSize, data = df_small)
##
```

```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.033506 -0.023723 -0.012231  0.009422  0.192046
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -1.705e+00  8.068e-03 -211.303   <2e-16 ***
## MatrixSize   6.430e-06  1.544e-05    0.416    0.679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04043 on 38 degrees of freedom
## Multiple R-squared:  0.004544,   Adjusted R-squared:  -0.02165
## F-statistic: 0.1734 on 1 and 38 DF,  p-value: 0.6794
```

```
#plot(linreg_small_transform)
m <- max(abs(linreg_small_transform$residuals))
plot(linreg_small_transform$residuals, ylim=c(-m, m))
abline(h=0, col="red")
```

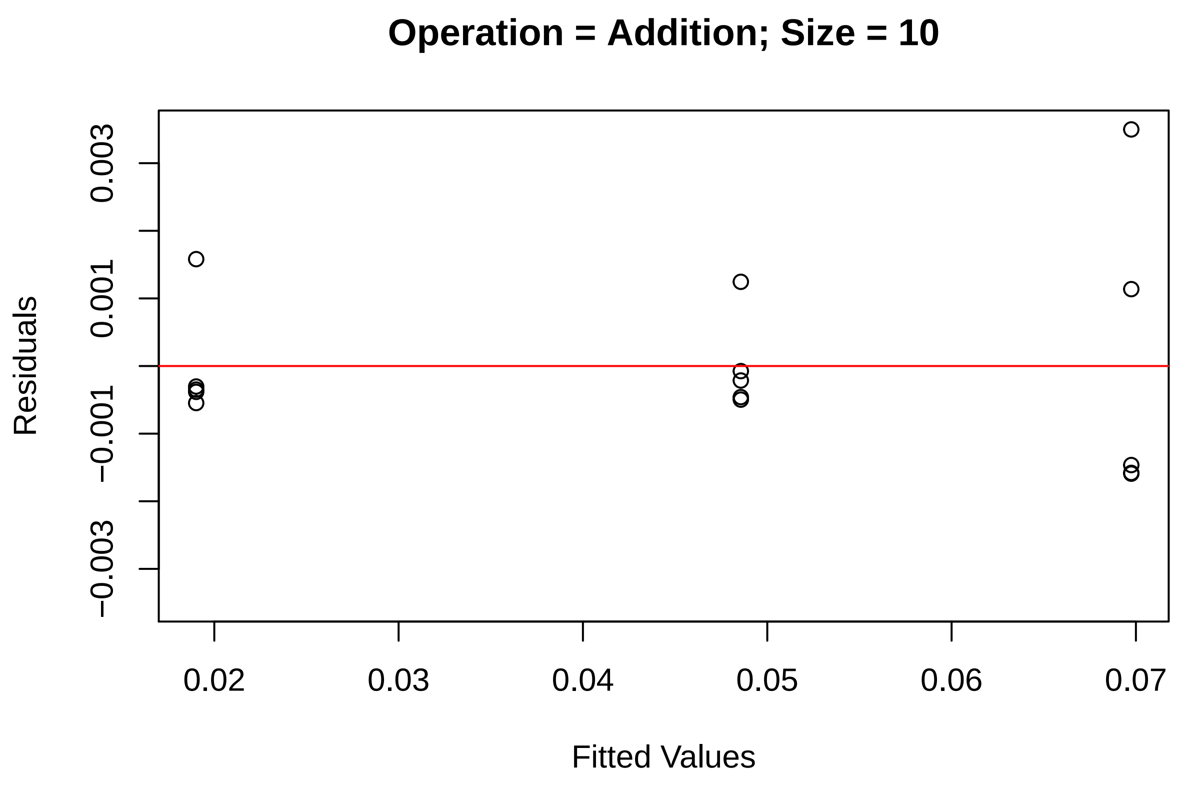


ANOVA tests fixing matrix operation and matrix size

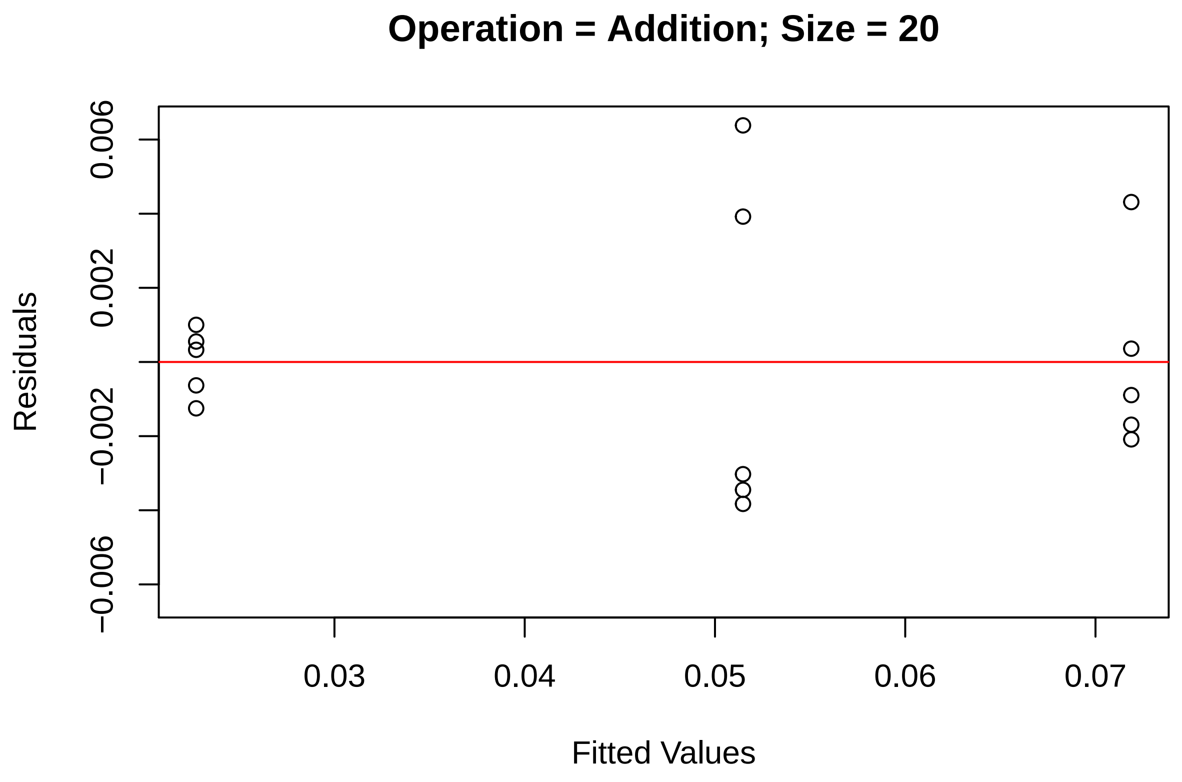
```
operation_arr <- unique(data$MatrixOperation)
size_arr <- unique(data$MatrixSize)

p_val_table <- matrix(0, nrow = length(size_arr), ncol = 2 * length(operation_arr))
rownames(p_val_table) <- size_arr
colnames(p_val_table) <- c(paste0("GPU_", operation_arr),
                           paste0("TPU_", operation_arr))
est_table <- matrix(0, nrow = length(size_arr), ncol = 2 * length(operation_arr))
ci_table <- matrix(0, nrow = length(size_arr), ncol = 2 * length(operation_arr))
```

```r
for(i in 1:length(operation_arr)){
  for(j in 1:length(size_arr)){
    operation <- operation_arr[i]
    size <- size_arr[j]

    headline <- paste0("Operation = ", operation, "; Size = ", size)
    df_small <- data[(data$MatrixOperation == operation) & (data$MatrixSize == size),]
    linreg_small <- lm(Runtime ~ Processor, data = df_small)
    sm <- summary(linreg_small)
    print(paste0(headline, ": ", round(sm$adj.r.squared, 3)))
    p_val_table[j, i] <- as.numeric(sm$coefficients[2,4])
    p_val_table[j, length(operation_arr)+i] <- as.numeric(sm$coefficients[3,4])

    est_table[j, i] <- paste0(round(sm$coefficients[2,1], 3), " (",
                              round(sm$coefficients[2,2], 3), ")")
    est_table[j, length(operation_arr)+i] <- paste0(round(sm$coefficients[3,1], 3),
                                                    " (",
                                                    round(sm$coefficients[3,2], 3),
                                                    ")")
    ci_gpu <- round(sm$coefficients[2,1] +
                      c(-1,1)*qnorm(0.975)*sm$coefficients[2,2], 3)
    ci_tpu <- round(sm$coefficients[3,1] +
                      c(-1,1)*qnorm(0.975)*sm$coefficients[3,2], 3)
    ci_table[j, i] <- paste0("(", ci_gpu[1], " to ", ci_gpu[2], ")")
    ci_table[j, length(operation_arr)+i] <- paste0("(", ci_tpu[1], " to ",
                                                   ci_tpu[2], ")")
    #print(headline)
    #print(summary(linreg_small))
    m <- max(abs(linreg_small$residuals))
    plot(linreg_small$fitted.values, linreg_small$residuals, main = headline,
         xlab = "Fitted Values", ylab = "Residuals", ylim=c(-m, m))
    abline(h=0, col="red")
  }
}
```

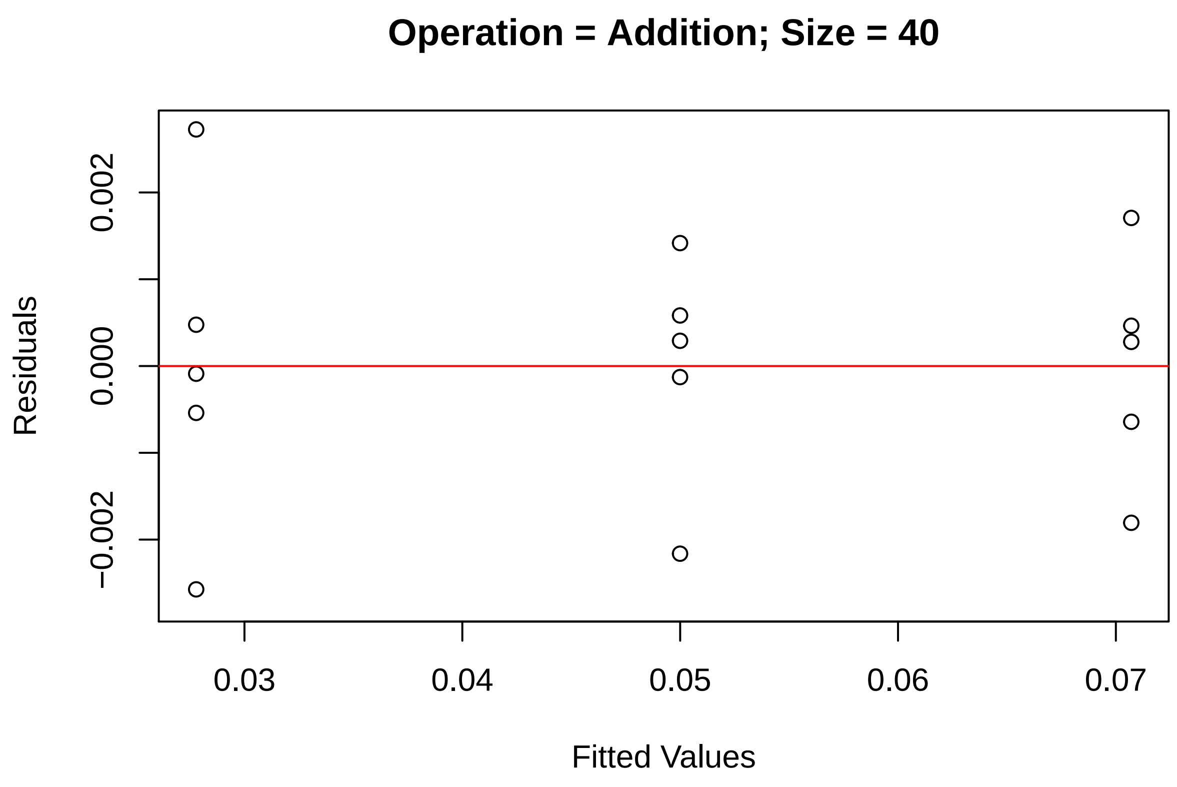
```
## [1] "Operation = Addition; Size = 10: 0.995"
```



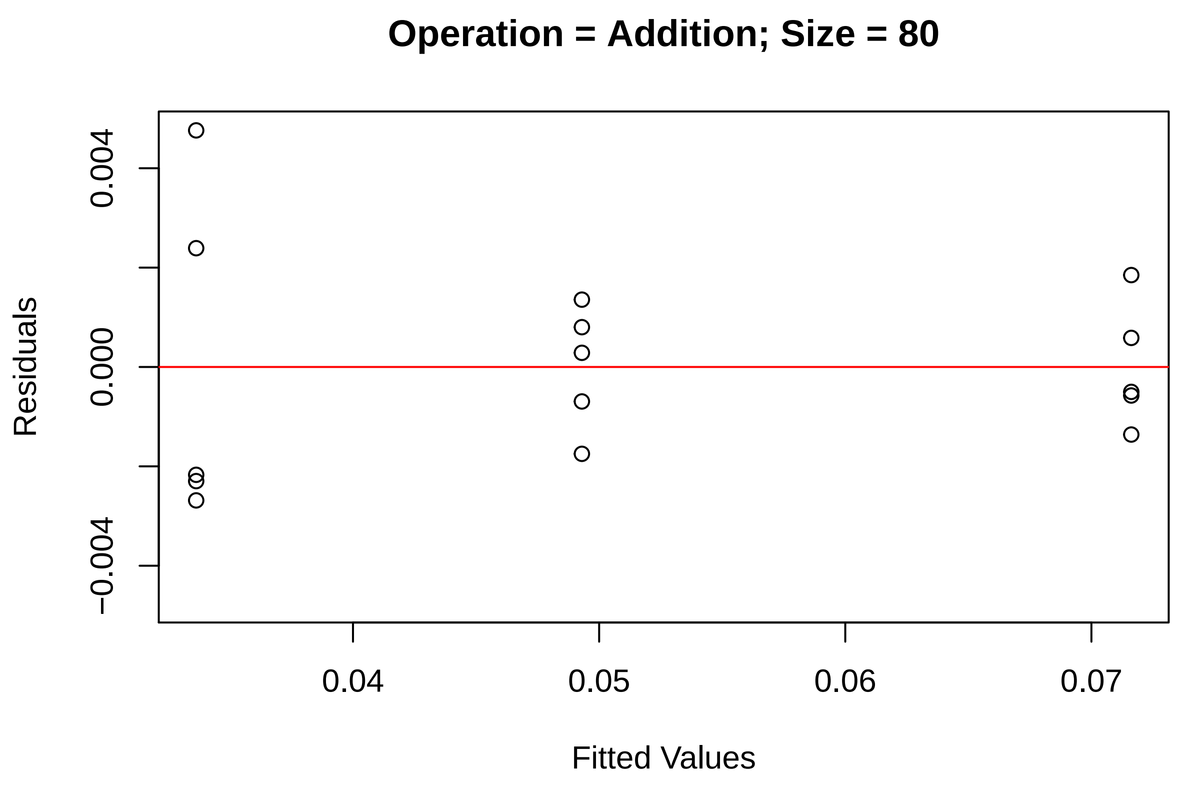
```
## [1] "Operation = Addition; Size = 20: 0.977"
```



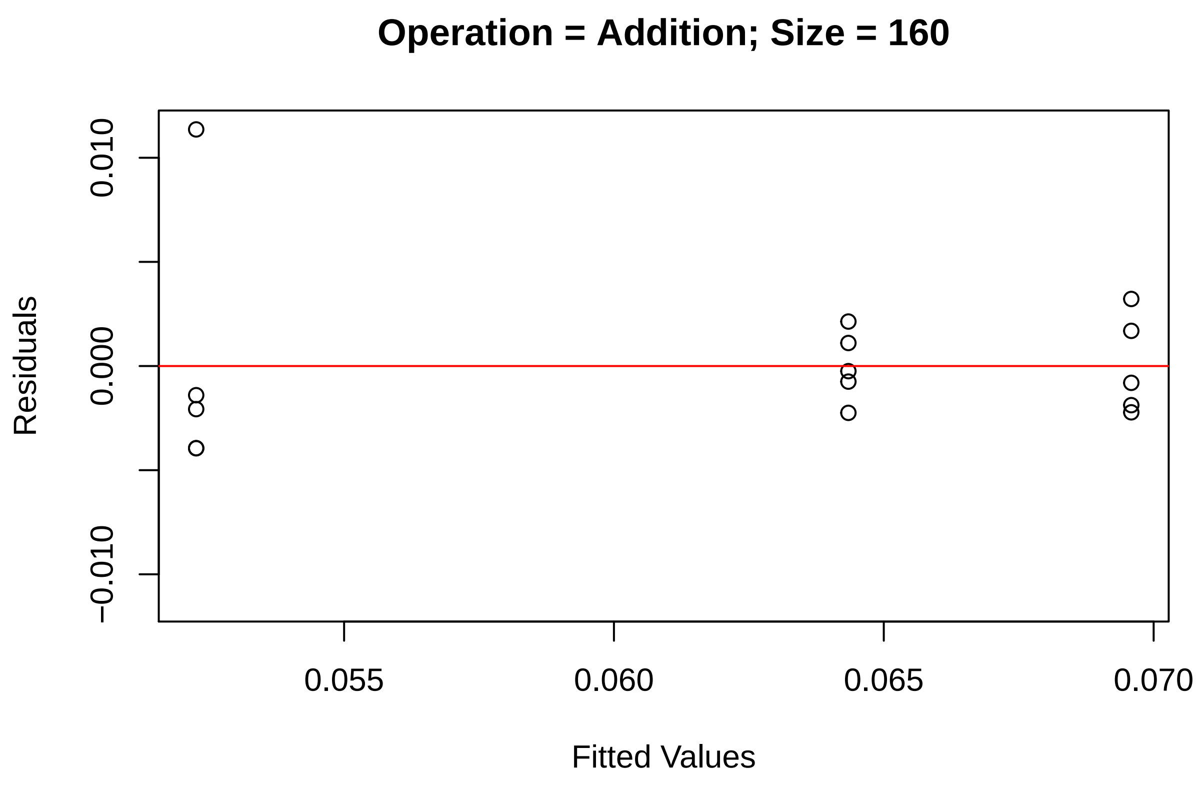
```
## [1] "Operation = Addition; Size = 40: 0.993"
```



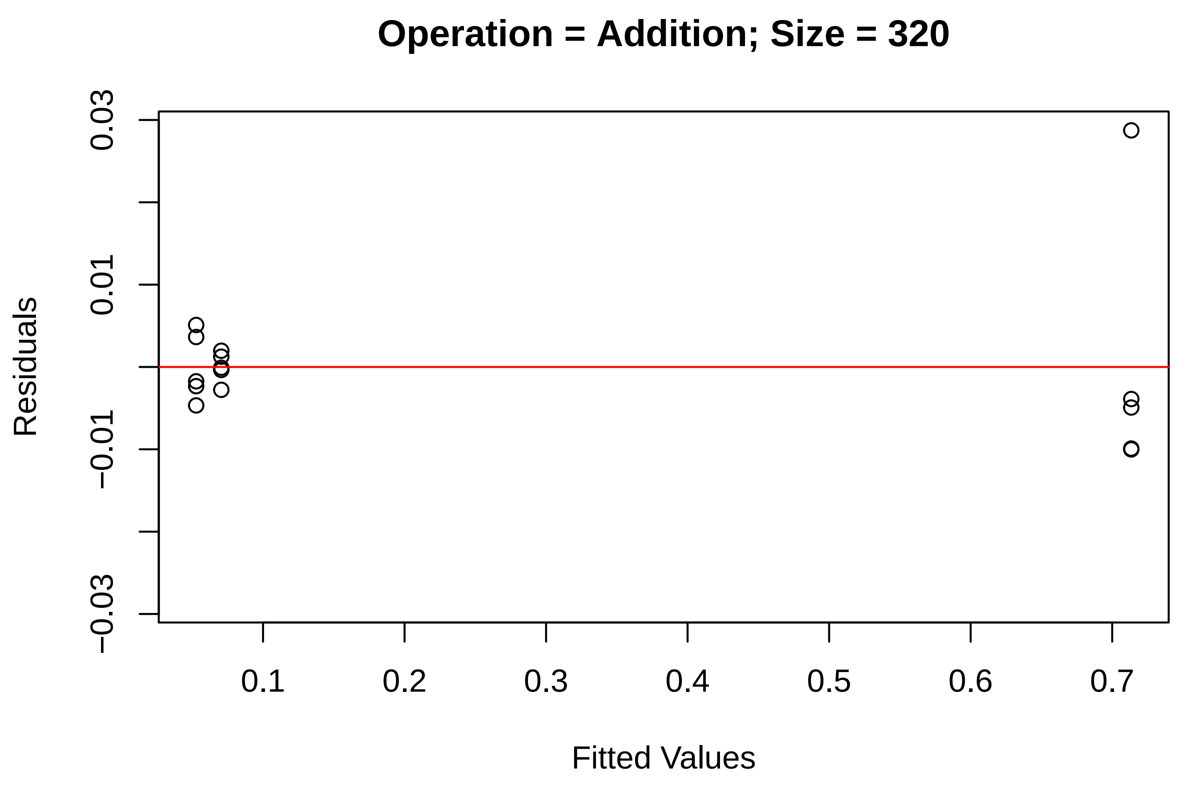
```
## [1] "Operation = Addition; Size = 80: 0.982"
```



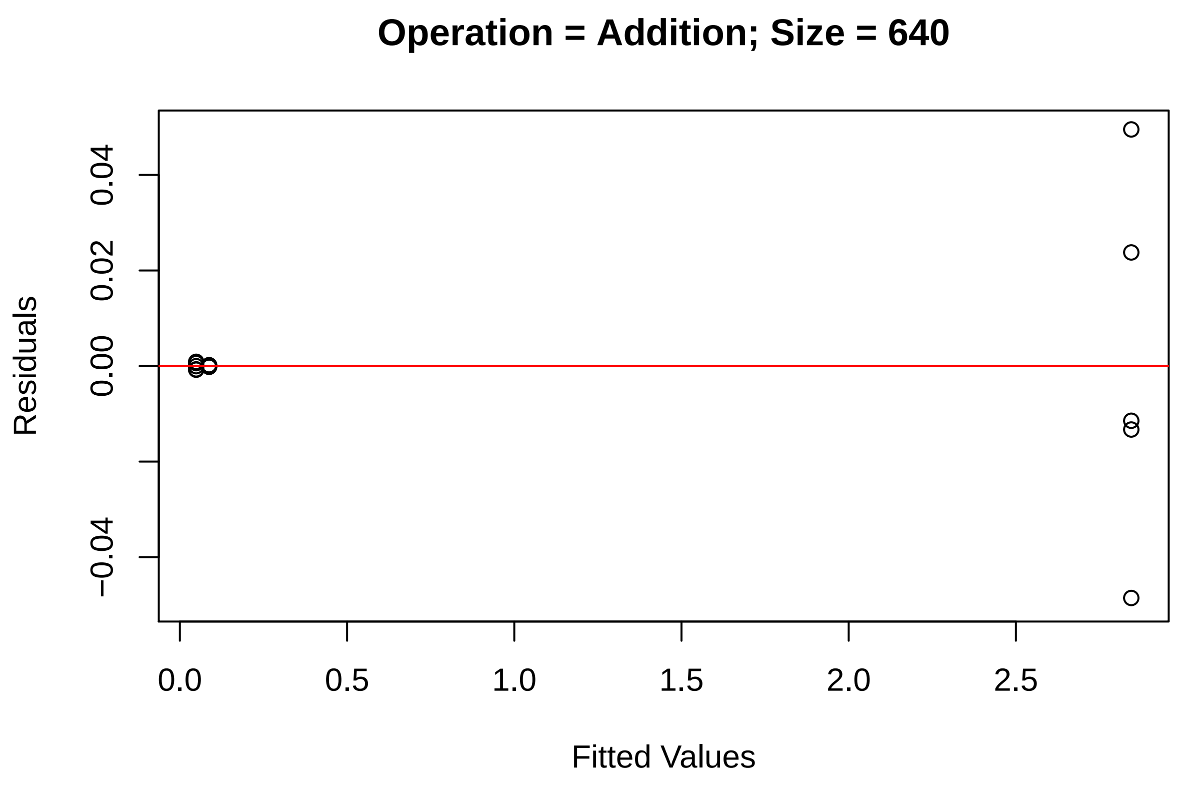
```
## [1] "Operation = Addition; Size = 160: 0.764"
```



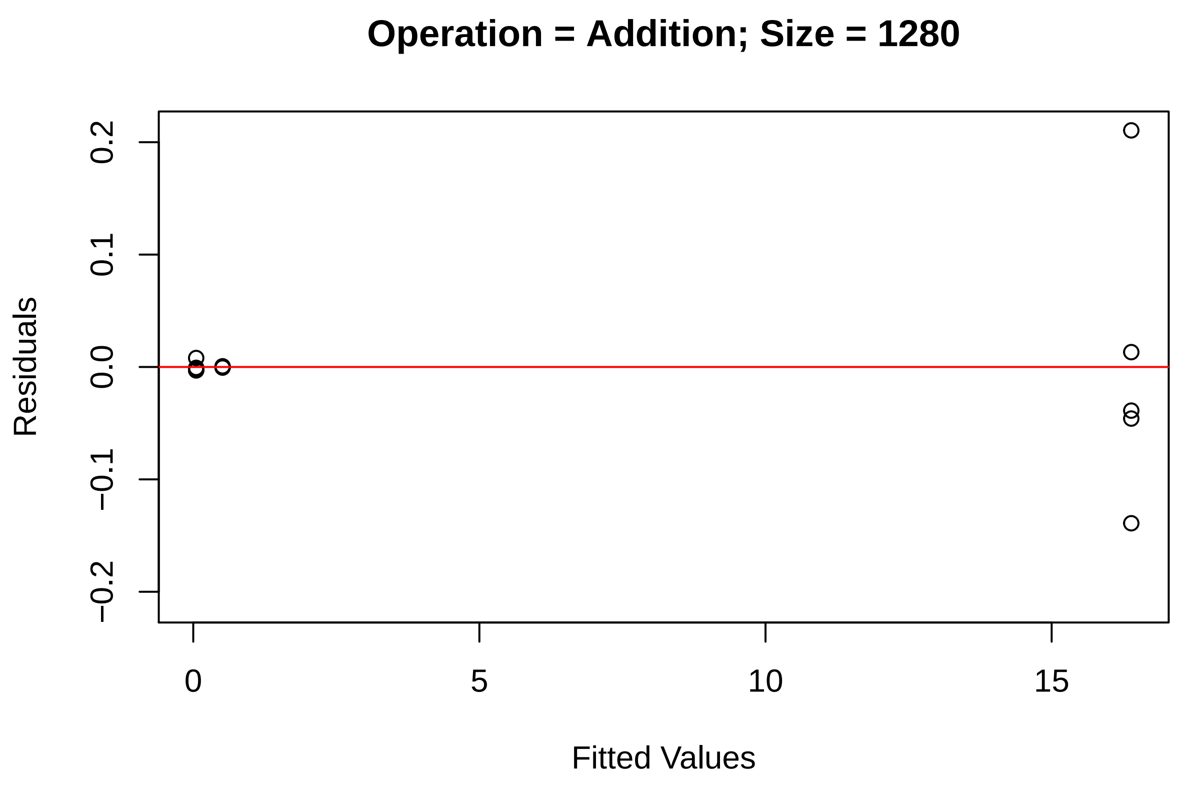
```
## [1] "Operation = Addition; Size = 320: 0.999"
```



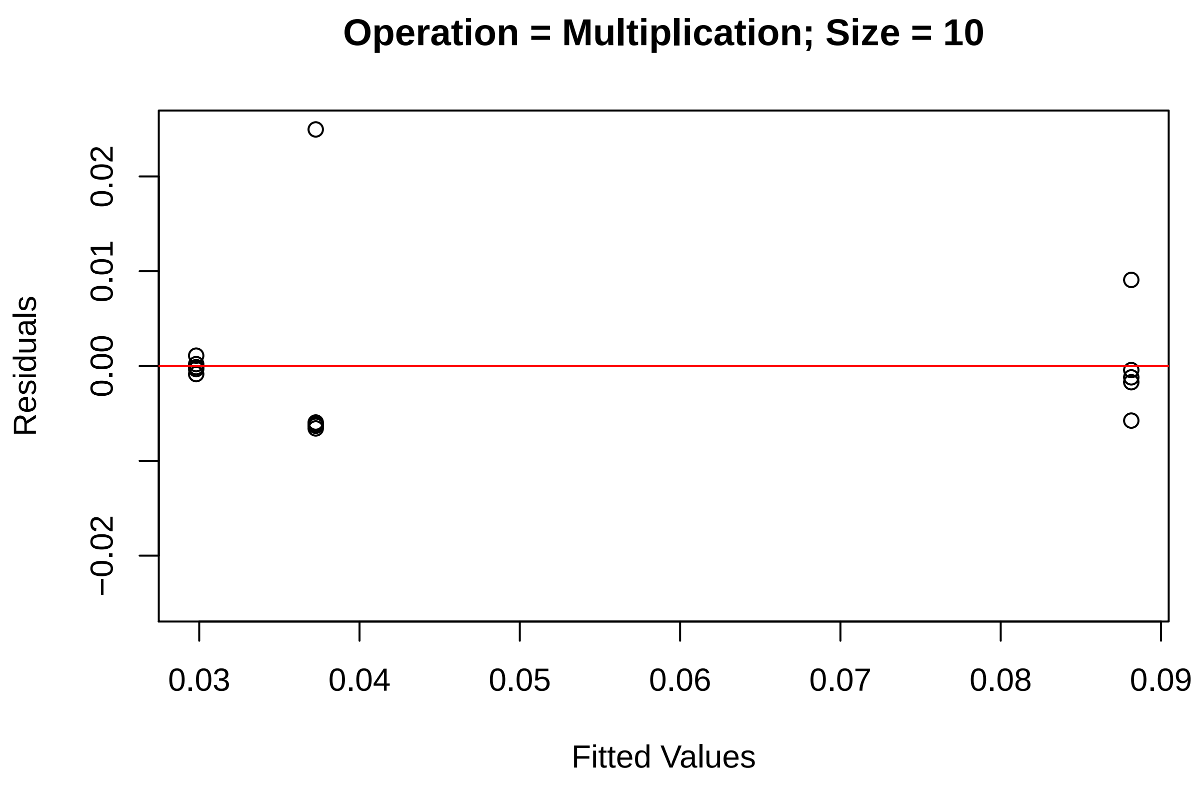
```
## [1] "Operation = Addition; Size = 640: 1"
```



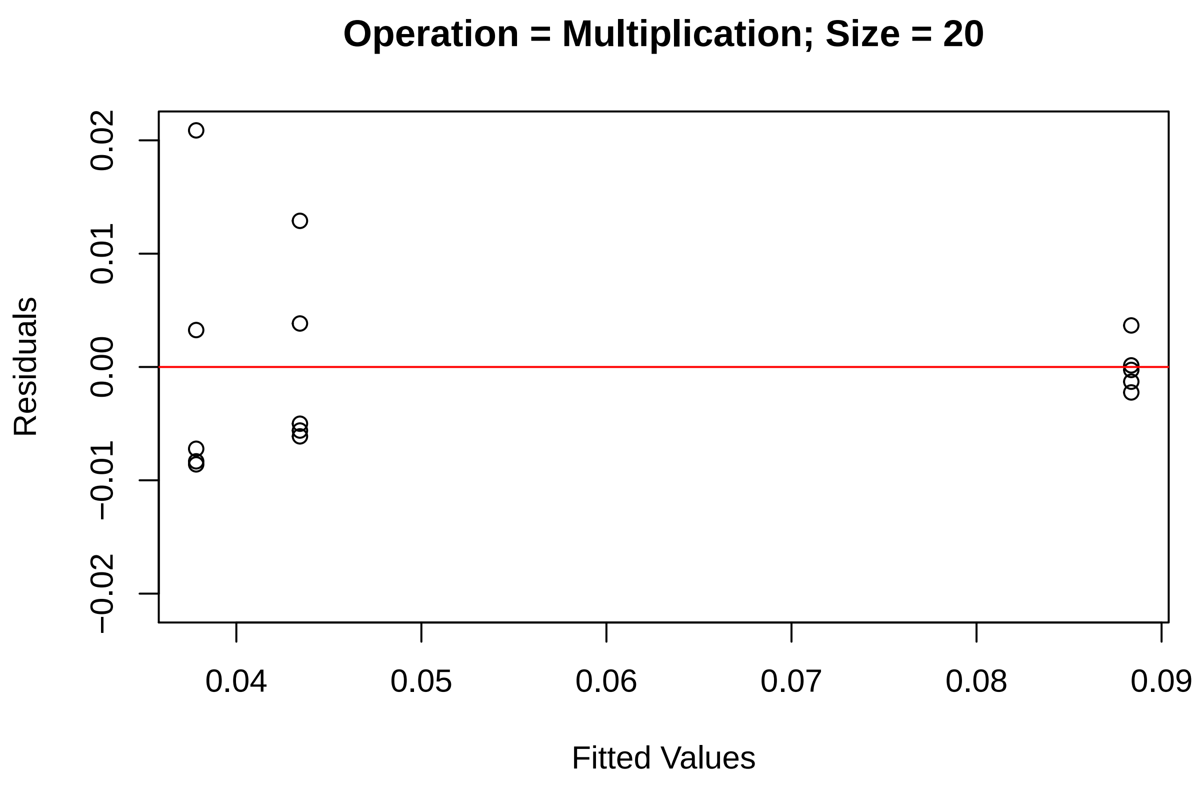
```
## [1] "Operation = Addition; Size = 1280: 1"
```



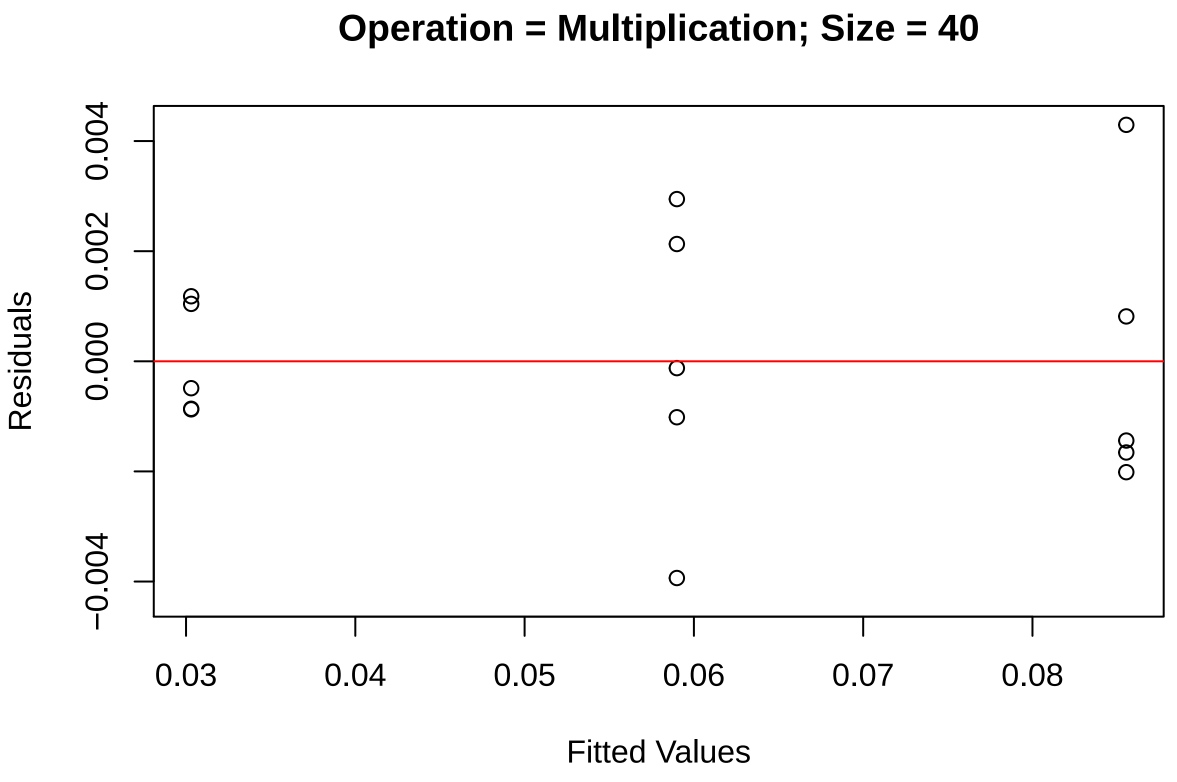
```
## [1] "Operation = Multiplication; Size = 10: 0.904"
```



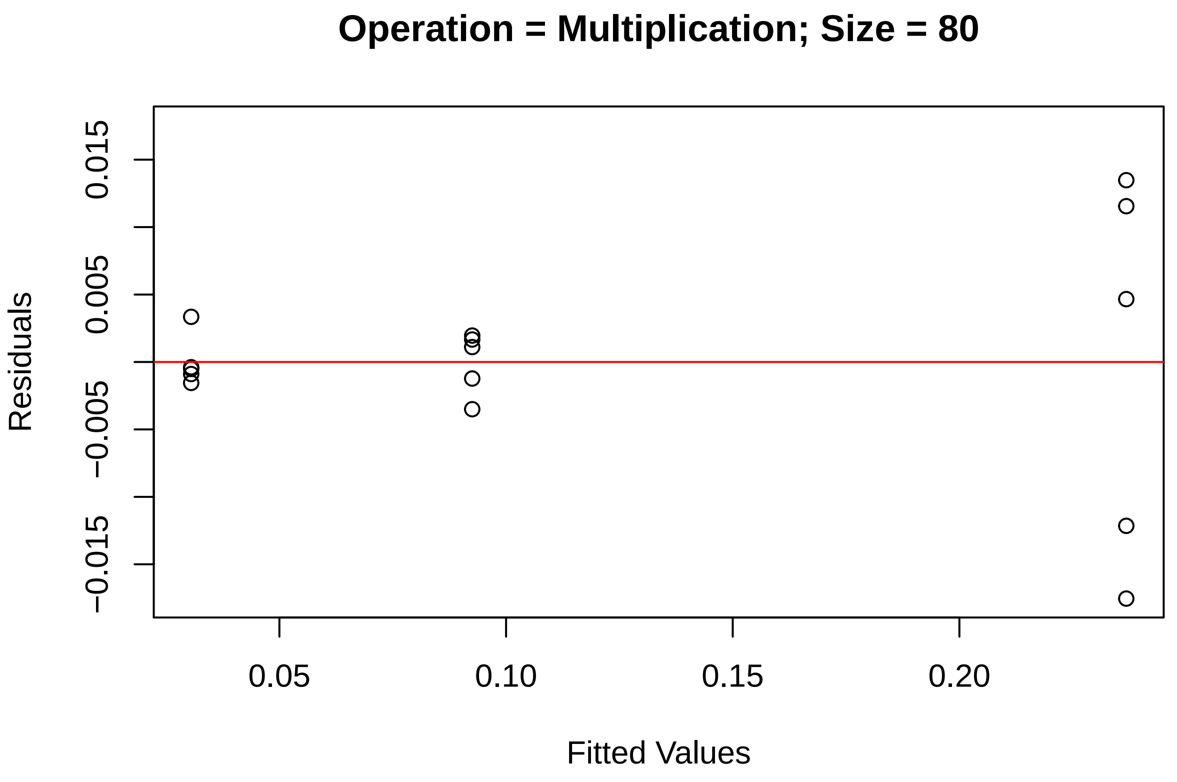
```
## [1] "Operation = Multiplication; Size = 20: 0.873"
```



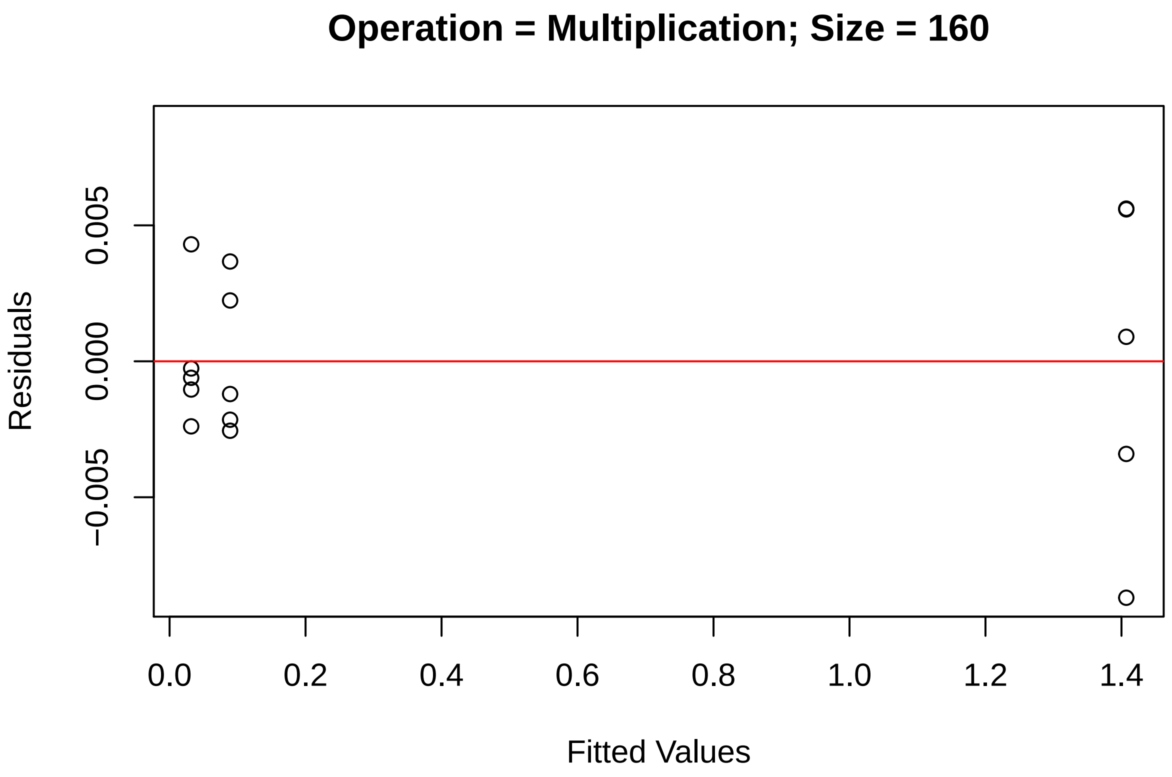
```
## [1] "Operation = Multiplication; Size = 40: 0.991"
```



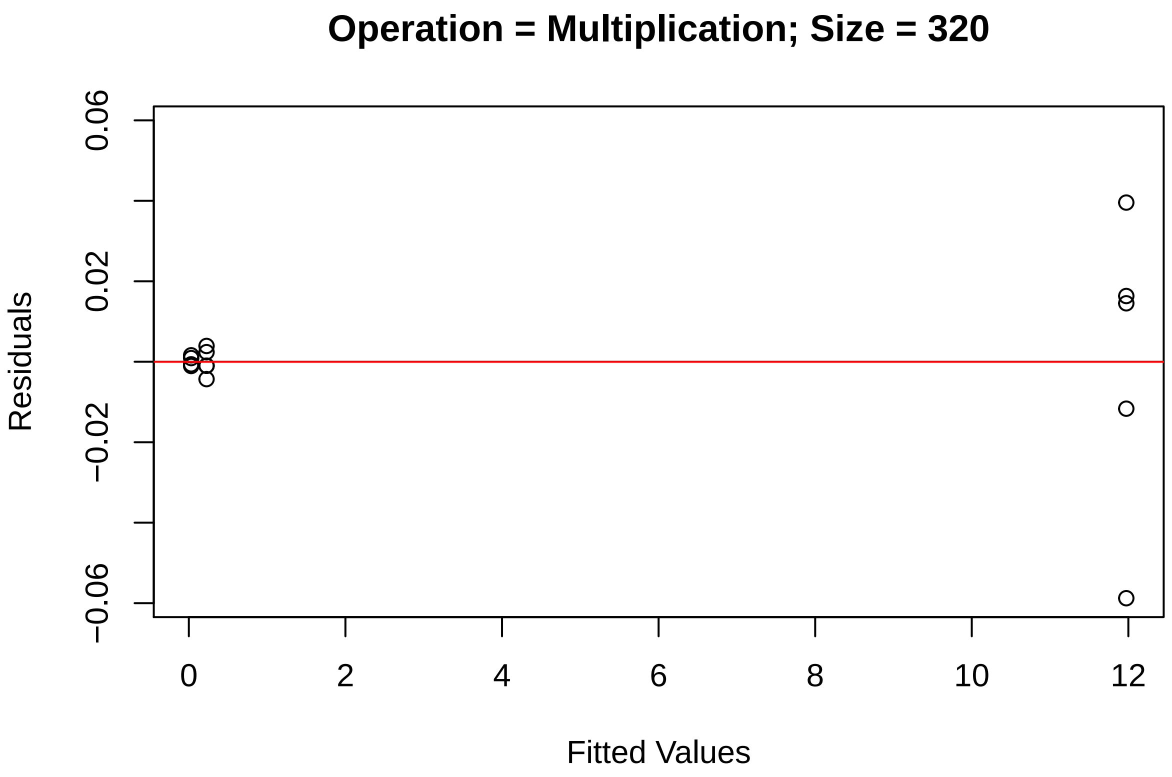
```
## [1] "Operation = Multiplication; Size = 80: 0.991"
```



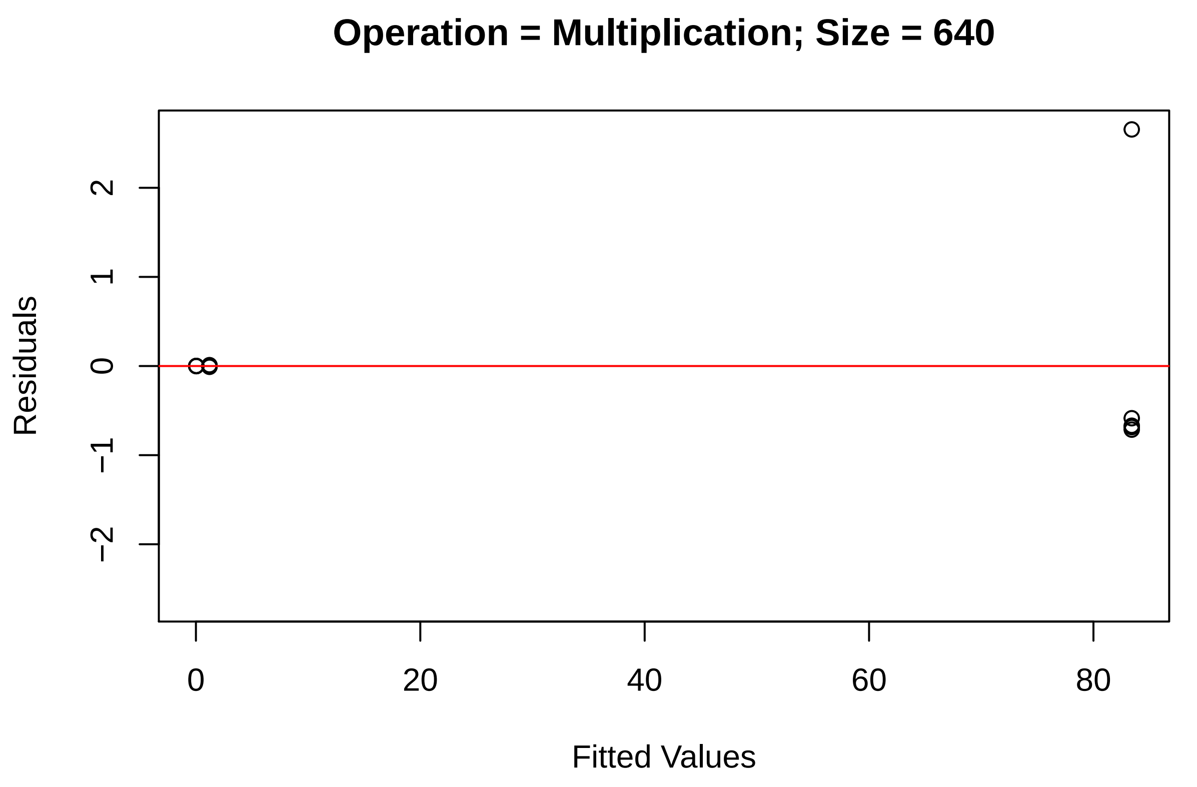
```
## [1] "Operation = Multiplication; Size = 160: 1"
```



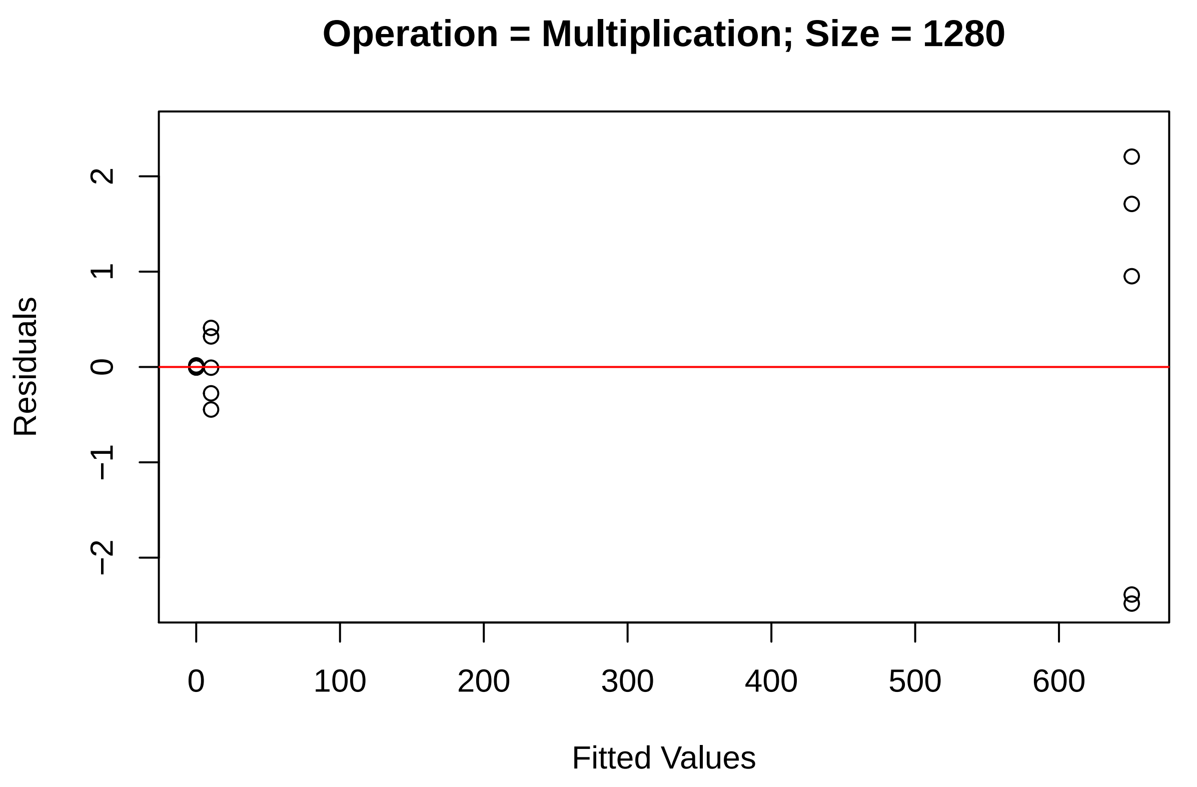
```
## [1] "Operation = Multiplication; Size = 320: 1"
```



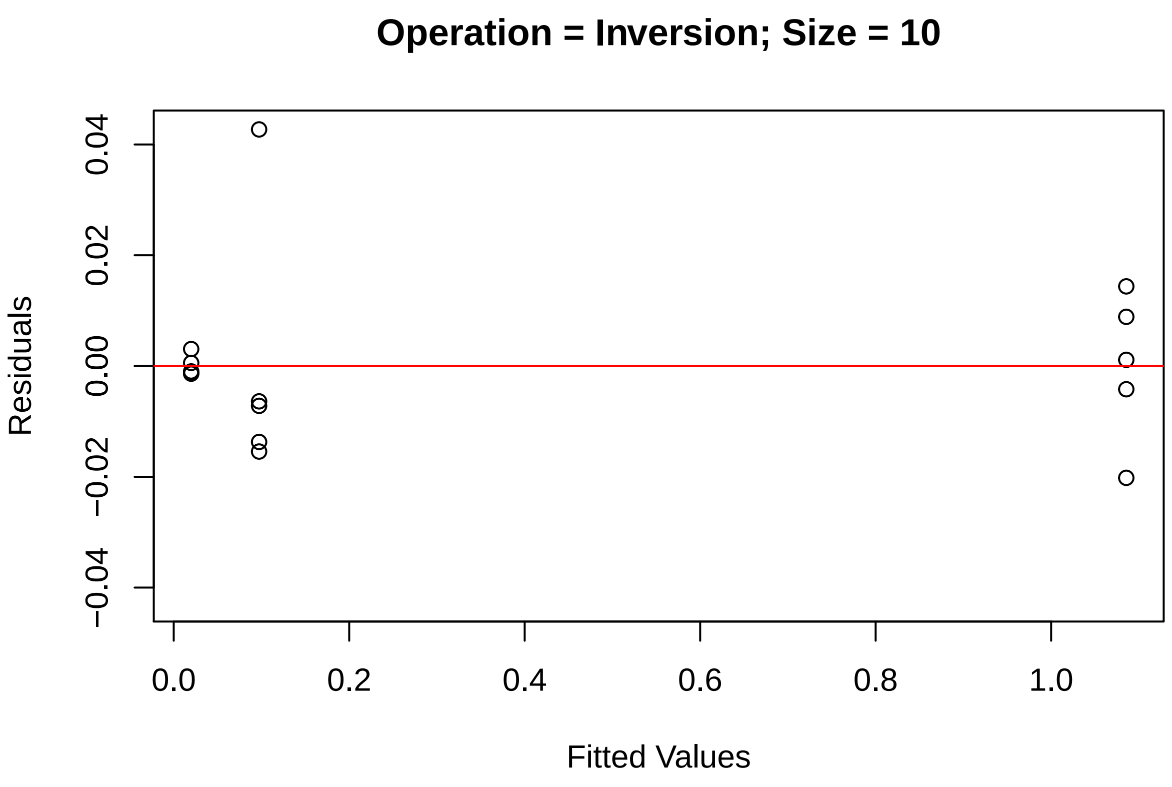
```
## [1] "Operation = Multiplication; Size = 640: 1"
```



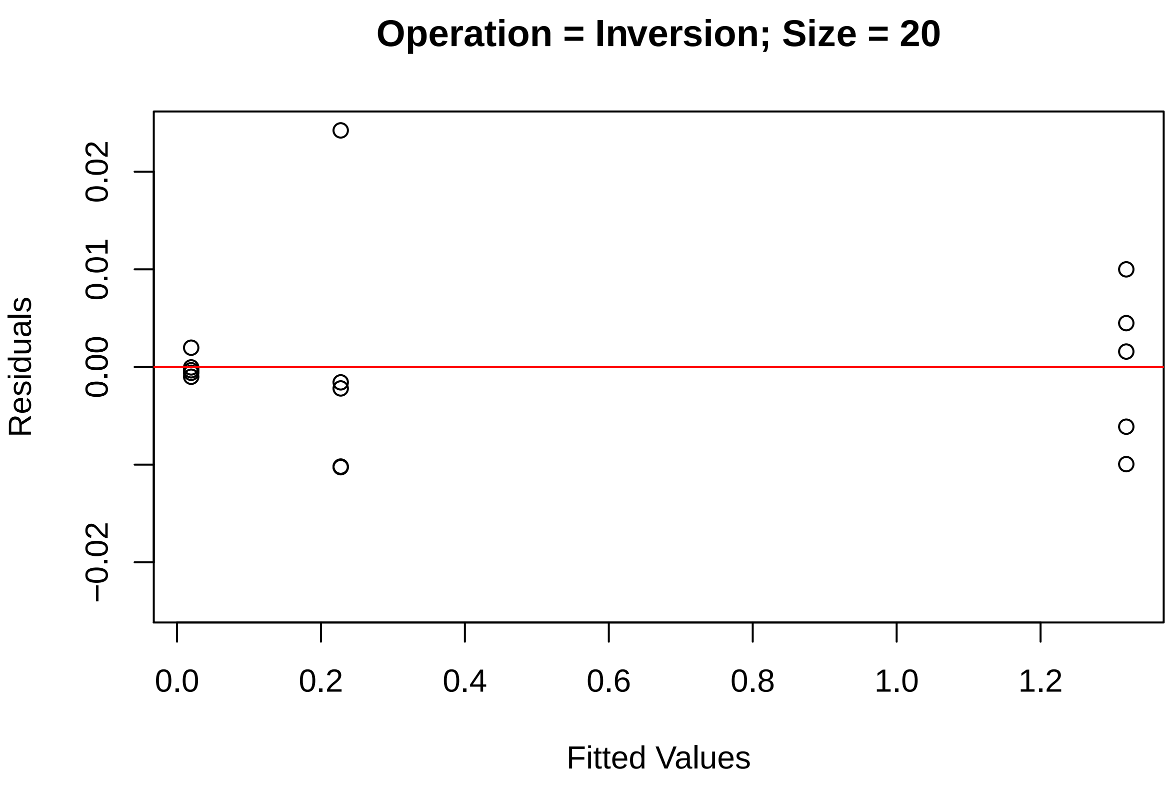
```
## [1] "Operation = Multiplication; Size = 1280: 1"
```



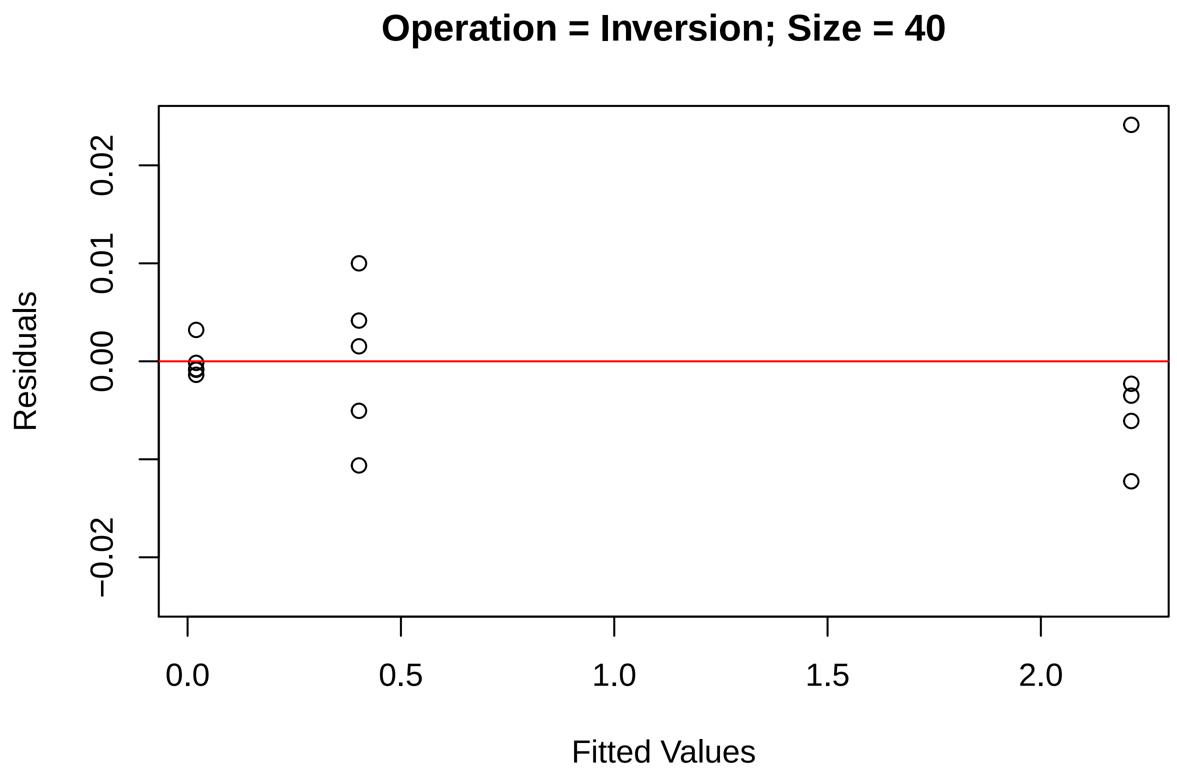
```
## [1] "Operation = Inversion; Size = 10: 0.999"
```



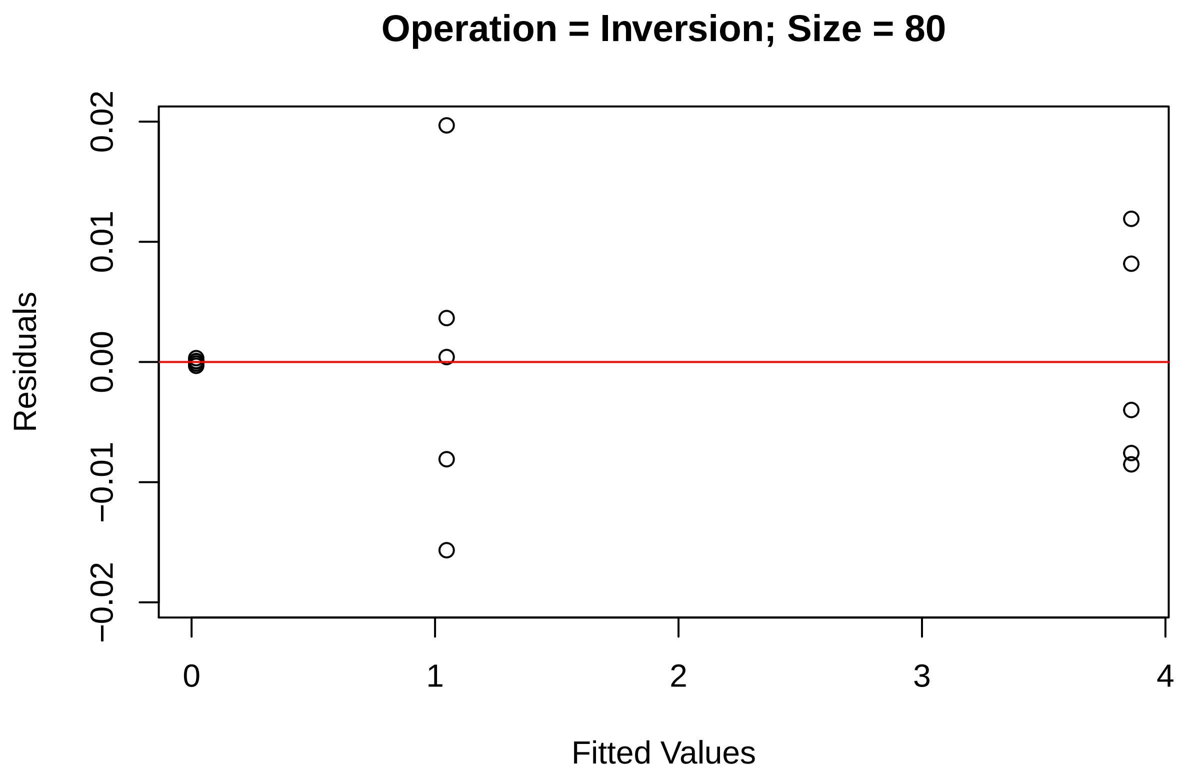
```
## [1] "Operation = Inversion; Size = 20: 1"
```



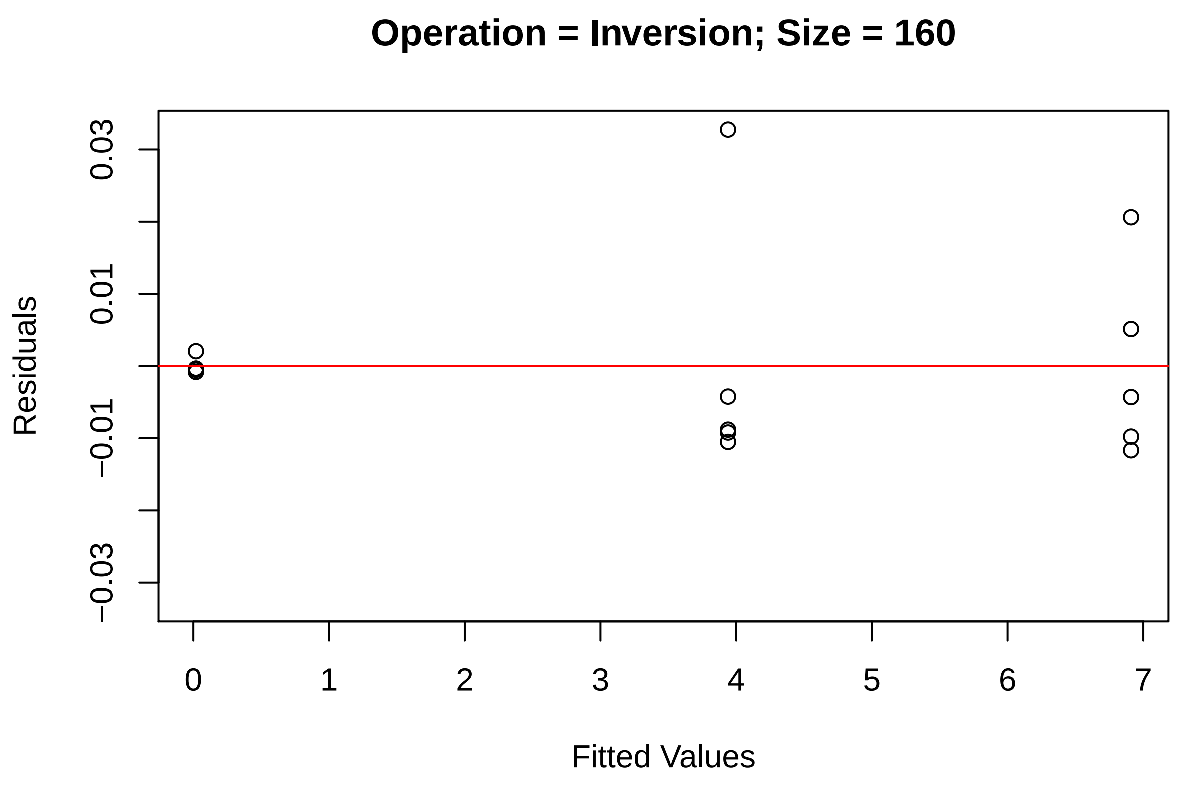
```
## [1] "Operation = Inversion; Size = 40: 1"
```



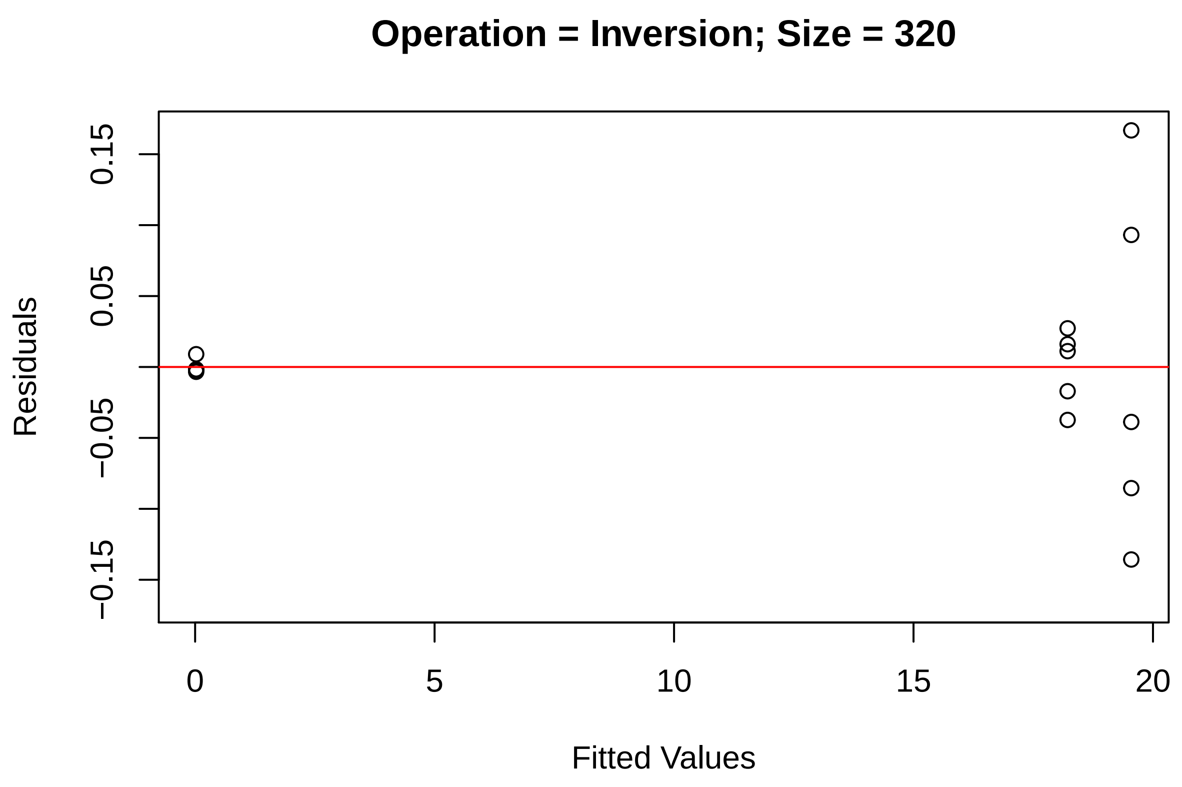
```
## [1] "Operation = Inversion; Size = 80: 1"
```



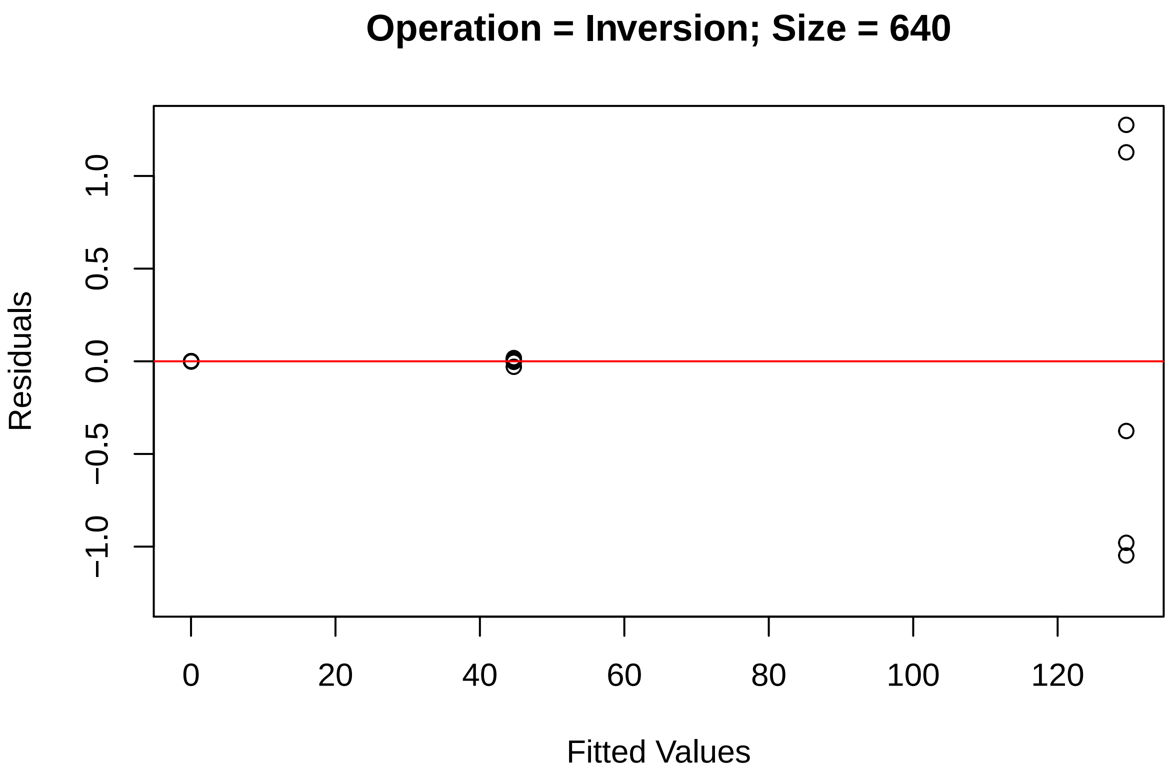
```
## [1] "Operation = Inversion; Size = 160: 1"
```



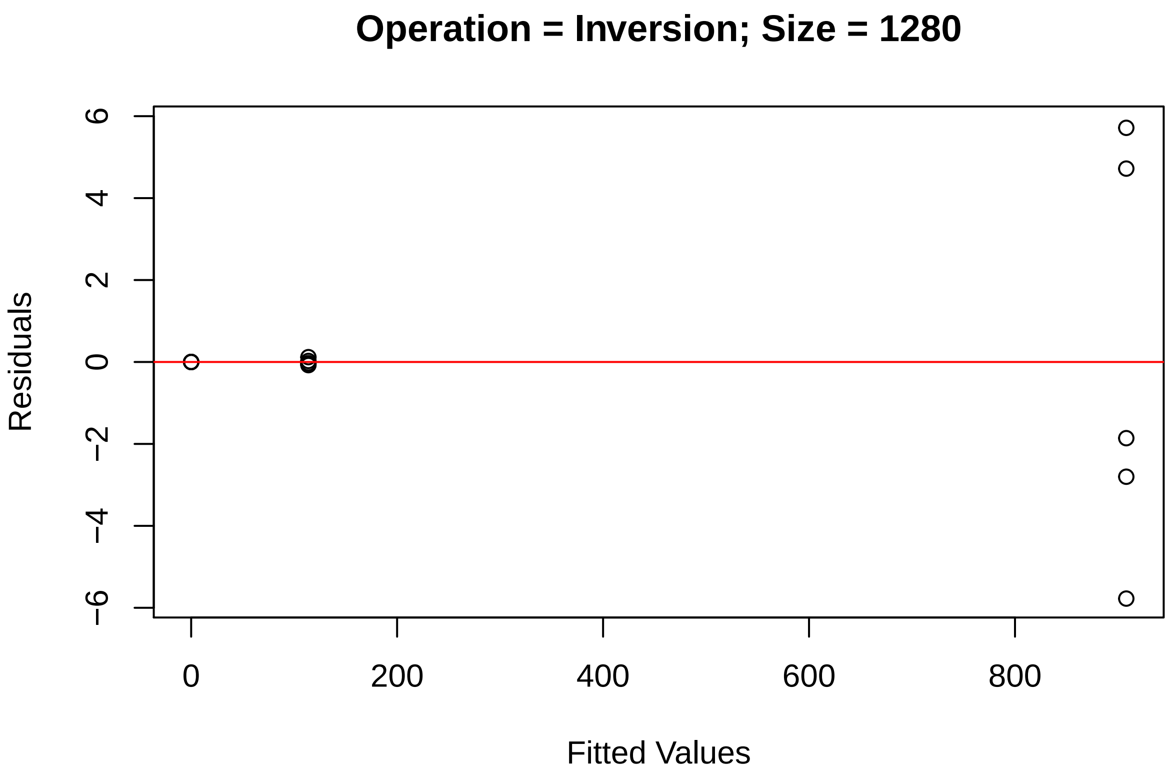
```
## [1] "Operation = Inversion; Size = 320: 1"
```



```
## [1] "Operation = Inversion; Size = 640: 1"
```



```
## [1] "Operation = Inversion; Size = 1280: 1"
```



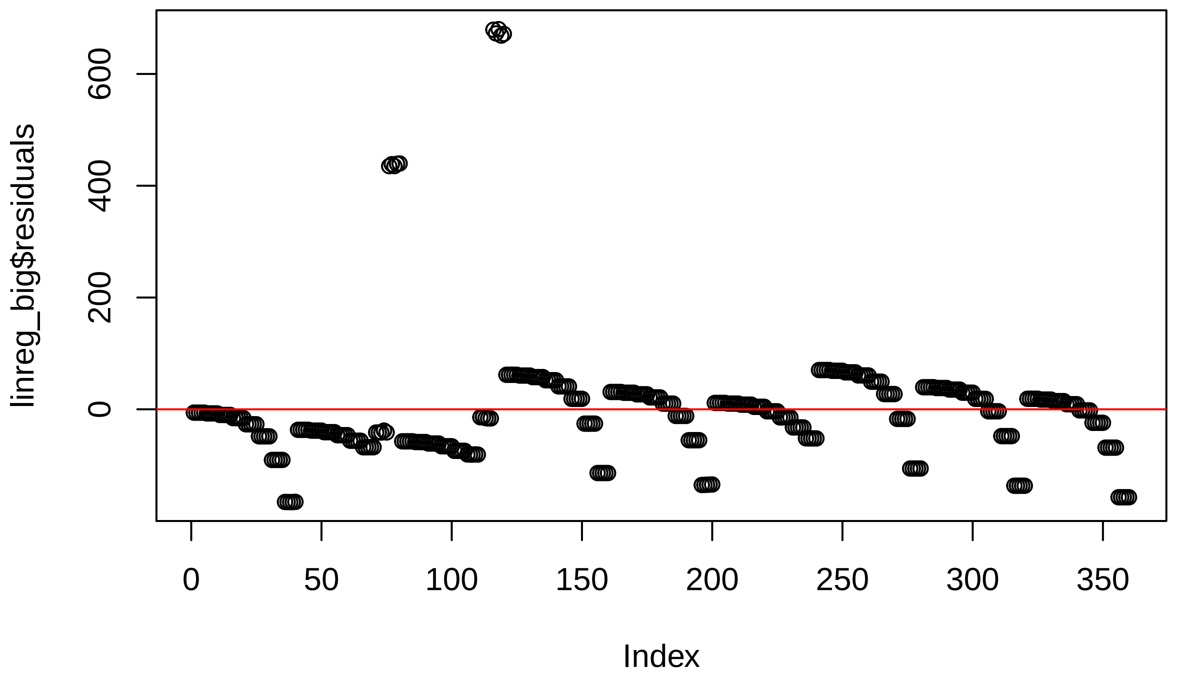
```r
#print(p_val_table)

parse_number <- function(x){
  ret <- paste0("$", str_replace_all(formatC(x, digits=4), "e", " \times 10^{"))
  if(str_detect(ret, "\times")){
    ret <- paste0(ret, "}")
  }
  ret <- paste0(ret, "$")
  ret
}
p_val_display <- p_val_table
for(i in 1:nrow(p_val_display)){
  for(j in 1:ncol(p_val_display)){
    p_val_display[i, j] <- parse_number(p_val_table[i, j])
  }
}
#p_val_display %>% print()

for(i in 1:nrow(p_val_display)){
  #paste0(p_val_display[i,], collapse = " & ") %>% print()
  #paste0(ci_table[i,], collapse = " & ") %>% print()
}
```

```r
df_small <- data[(data$MatrixOperation == "Inversion") & (data$MatrixSize == 1280),]
linreg_small <- lm(Runtime ~ Processor, data = df_small)
summary(linreg_small)
```

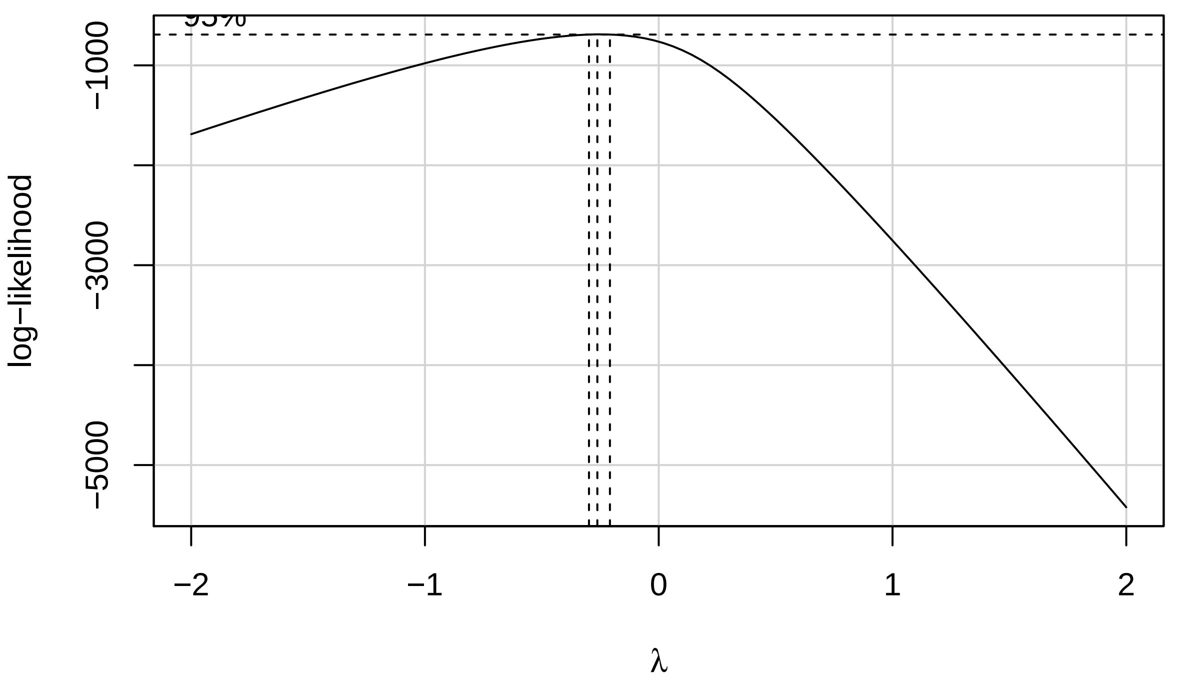
```
##
## Call:
## lm(formula = Runtime ~ Processor, data = df_small)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7751 -0.0557 -0.0008  0.0129  5.7155
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   908.077      1.289   704.7   <2e-16 ***
## ProcessorGPU -794.204      1.822  -435.8   <2e-16 ***
## ProcessorTPU -908.058      1.822  -498.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.881 on 12 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.474e+05 on 2 and 12 DF,  p-value: < 2.2e-16
```

Big model

```r
linreg_big <- lm(Runtime ~ MatrixOperation + Processor + MatrixSize, data = data)
plot(linreg_big$residuals)
abline(h=0, col="red")
```



```r
boxCox(linreg_big)
```

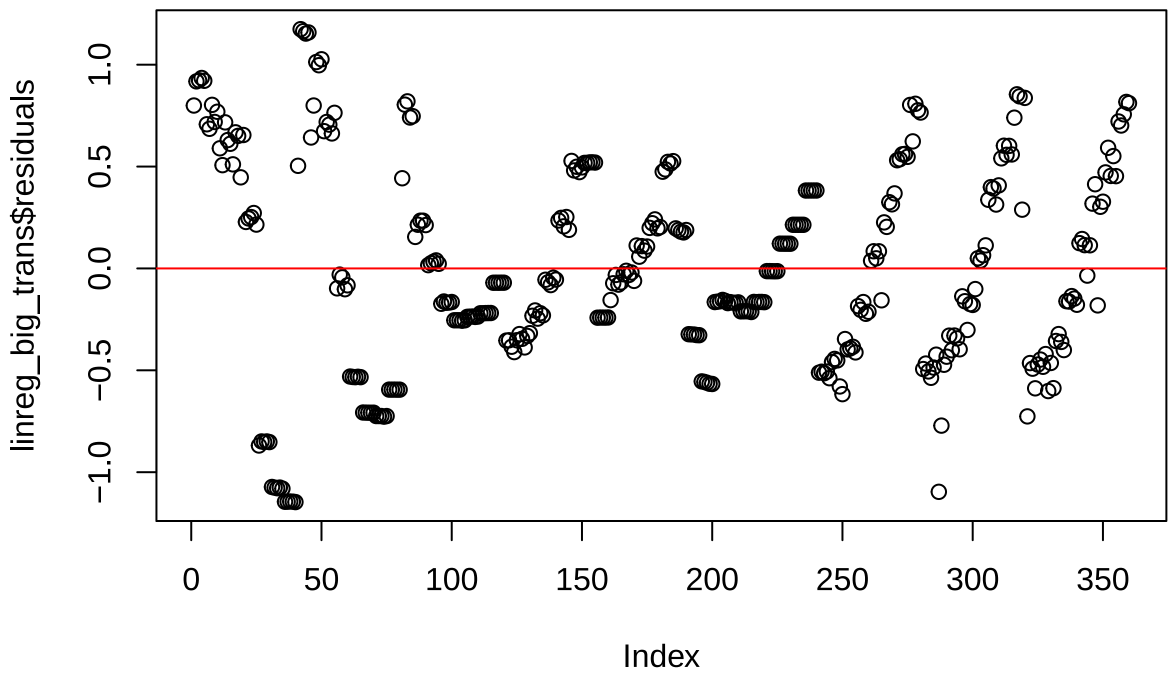


```r
linreg_big_trans <- lm(Runtime^(-1/3) ~ MatrixOperation * Processor +
                          poly(MatrixSize, 4),
                        data = data)
summary(linreg_big_trans)
```

```
##
## Call:
```

```
## lm(formula = Runtime^(-1/3) ~ MatrixOperation * Processor + poly(MatrixSize,
##     4), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.14670 -0.33016 -0.06913  0.38243  1.17423
##
## Coefficients:
##                                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                                2.30016    0.07962  28.888  < 2e-16
## MatrixOperationInversion                  -1.36533    0.11260 -12.125  < 2e-16
## MatrixOperationMultiplication             -0.82850    0.11260  -7.358 1.37e-12
## ProcessorGPU                              -0.04793    0.11260  -0.426 0.670607
## ProcessorTPU                               0.40803    0.11260   3.624 0.000334
## poly(MatrixSize, 4)1                      -7.84492    0.50358 -15.578  < 2e-16
## poly(MatrixSize, 4)2                       3.46485    0.50358   6.880 2.80e-11
## poly(MatrixSize, 4)3                      -1.73253    0.50358  -3.440 0.000652
## poly(MatrixSize, 4)4                       0.33158    0.50358   0.658 0.510688
## MatrixOperationInversion:ProcessorGPU     -0.30152    0.15925  -1.893 0.059131
## MatrixOperationMultiplication:ProcessorGPU  0.35805    0.15925   2.248 0.025177
## MatrixOperationInversion:ProcessorTPU      2.35357    0.15925  14.779  < 2e-16
## MatrixOperationMultiplication:ProcessorTPU  1.29535    0.15925   8.134 7.46e-15
##
## (Intercept)                               ***
## MatrixOperationInversion                  ***
## MatrixOperationMultiplication             ***
## ProcessorGPU
## ProcessorTPU                              ***
## poly(MatrixSize, 4)1                      ***
## poly(MatrixSize, 4)2                      ***
## poly(MatrixSize, 4)3                      ***
## poly(MatrixSize, 4)4
## MatrixOperationInversion:ProcessorGPU     .
## MatrixOperationMultiplication:ProcessorGPU *
## MatrixOperationInversion:ProcessorTPU     ***
## MatrixOperationMultiplication:ProcessorTPU ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5036 on 347 degrees of freedom
## Multiple R-squared:  0.8226, Adjusted R-squared:  0.8165
## F-statistic: 134.1 on 12 and 347 DF,  p-value: < 2.2e-16
```

```r
plot(linreg_big_trans$residuals)
abline(h=0, col="red")
```



Lixian Chen Part I

When the processor is the same, is there any operation and matrix size effect? We want to answer the following question: in each scenario, which processor should we use? When the matrix size is the same, is there any processor or operation type effect, or interaction?

Getting Data Ready

```
# Read Data
df <- fread("../data/Runtime.csv", header=TRUE)
attach(df)

MatrixOperation<-factor(df$MatrixOperation)
Processor<-factor(df$Processor)

summary(Runtime)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0183   0.0313   0.0712  28.2915   1.2407 913.7929
```

```
summary(Processor)
```

```
## CPU GPU TPU
## 120 120 120
```

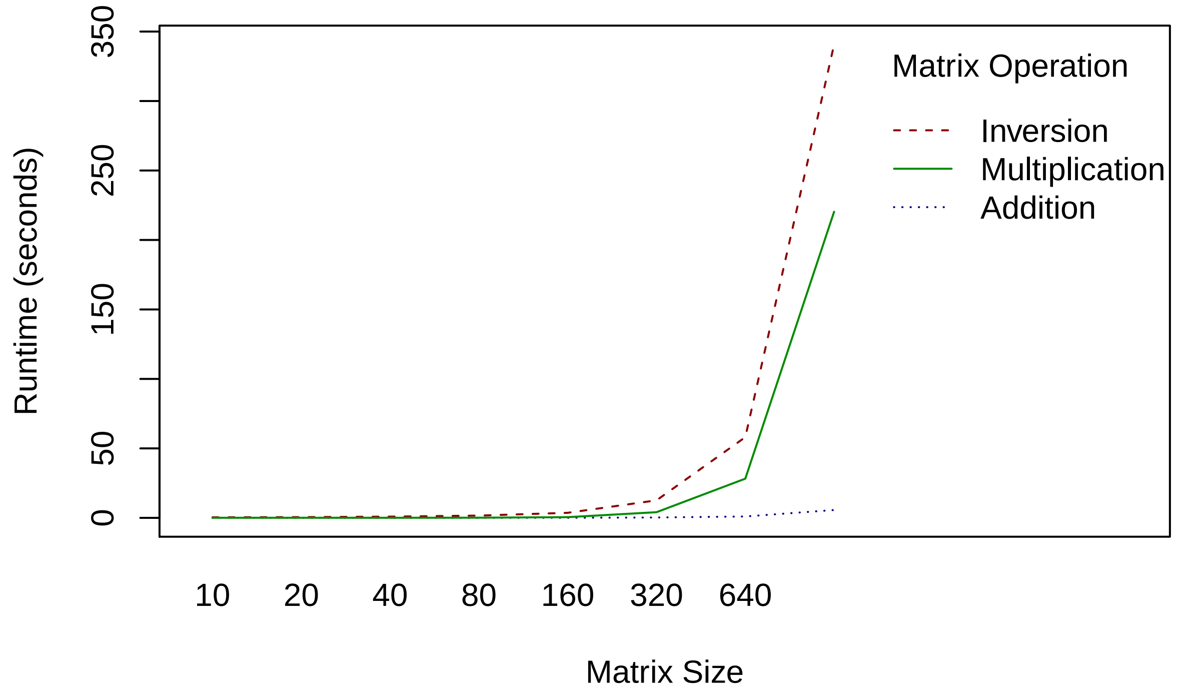
```
summary(MatrixOperation)
```

```
##       Addition      Inversion Multiplication
##            120            120            120
```

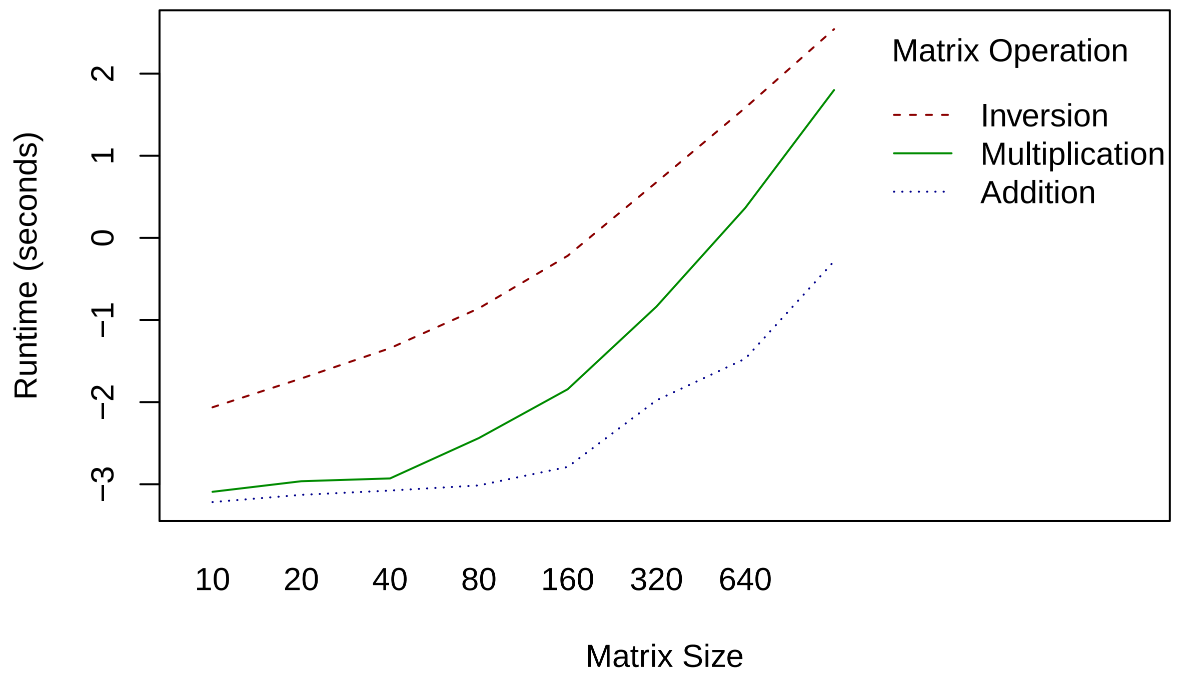
```
summary(MatrixSize)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    10.0    35.0   120.0   318.8   400.0  1280.0
```

```
op=par(mfrow=c(1,1))
#op=par(mfrow=c(2,1))
interaction.plot(df$MatrixSize, df$MatrixOperation, df$Runtime, ylab = "Runtime (seconds)",
                 xlab = "Matrix Size", trace.label = "Matrix Operation", col = c("blue4", "red4", "greer
```



```
interaction.plot(df$MatrixSize, df$MatrixOperation, log(df$Runtime), ylab = "Runtime (seconds)",
                 xlab = "Matrix Size", trace.label = "Matrix Operation", col = c("blue4", "red4", "greer
```



```r
par(op)


CPUdata <- df %>% dplyr::filter(Processor=="CPU")
GPUdata <- df %>% dplyr::filter(Processor=="GPU")
TPUdata <- df %>% dplyr::filter(Processor=="TPU")

summary(CPUdata$Runtime)
```

```
##     Min.   1st Qu.   Median     Mean   3rd Qu.     Max.
##   0.0185   0.0576   0.5576  76.2816  13.0731 913.7929
```

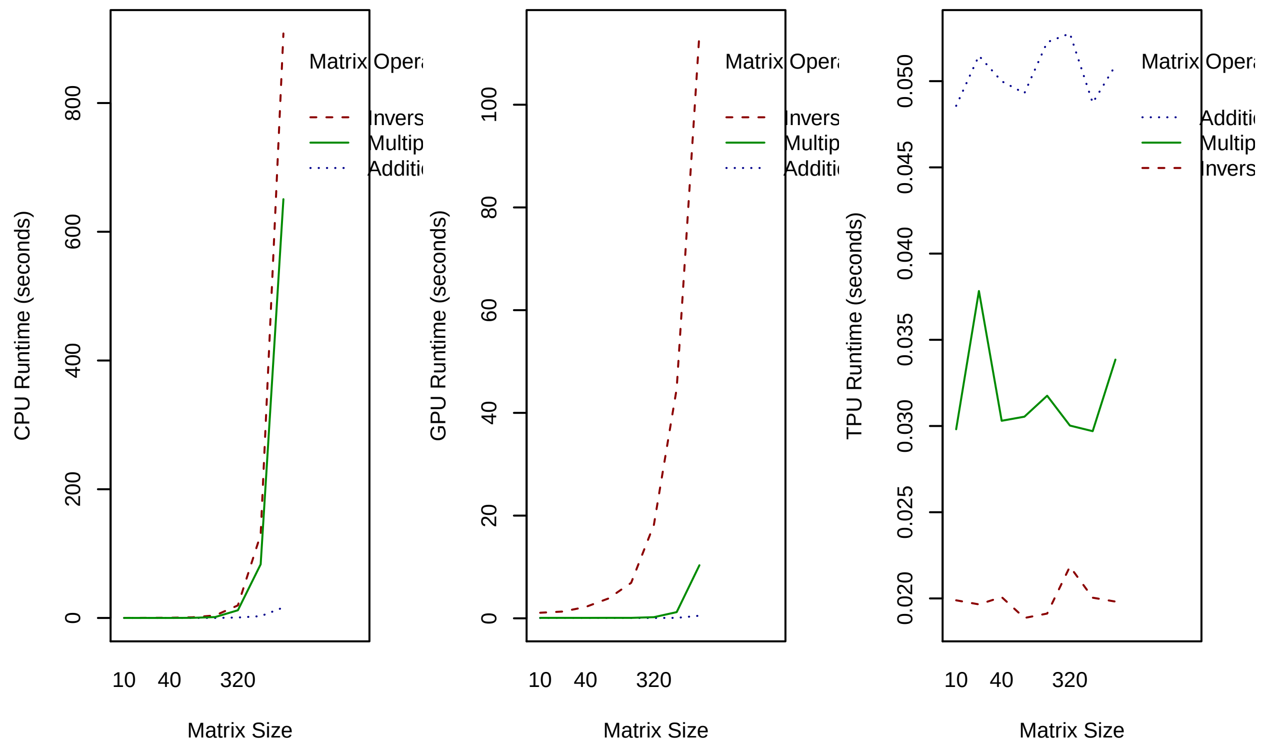
```r
summary(GPUdata$Runtime)
```

```
##      Min.    1st Qu.    Median      Mean    3rd Qu.      Max.
##   0.06736   0.08084   0.15957   8.55880   2.63984 113.99131
```

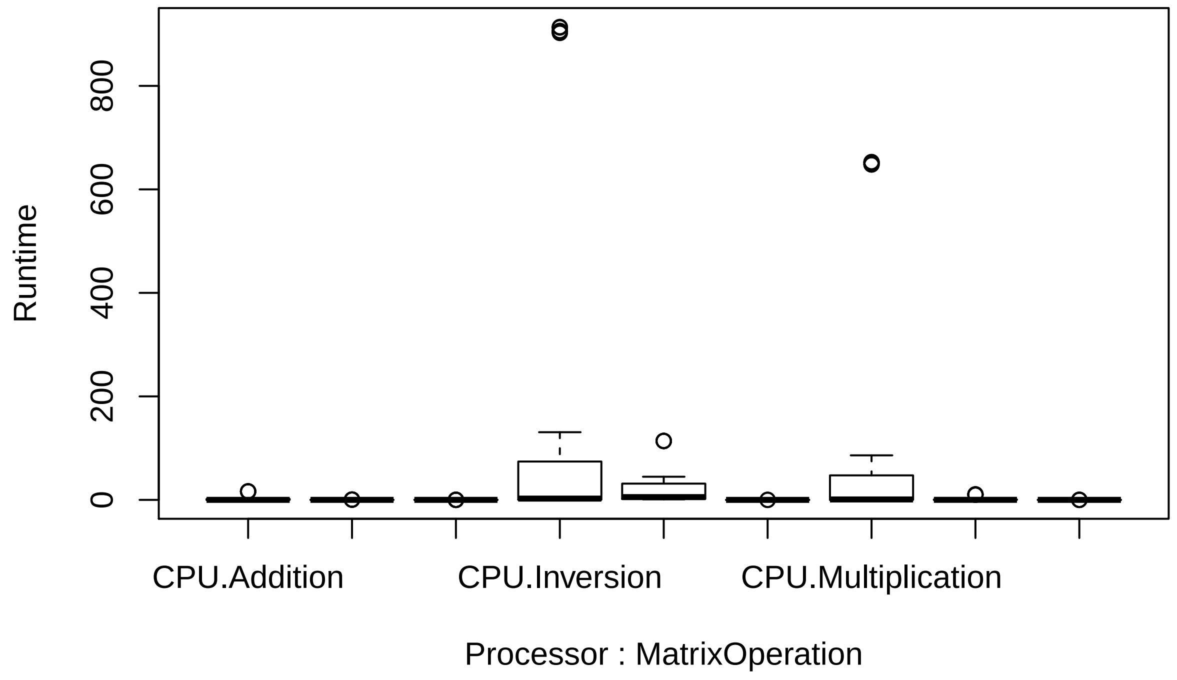
```r
summary(TPUdata$Runtime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01831 0.02046 0.03004 0.03405 0.04831 0.06362
```

```r
op=par(mfrow=c(1,3))
interaction.plot(CPUdata$MatrixSize, CPUdata$MatrixOperation, CPUdata$Runtime, ylab = "CPU Runtime (seco
                 xlab = "Matrix Size", trace.label = "Matrix Operation", col = c("blue4", "red4", "gree
interaction.plot(GPUdata$MatrixSize, GPUdata$MatrixOperation, GPUdata$Runtime, ylab = "GPU Runtime (seco
                 xlab = "Matrix Size", trace.label = "Matrix Operation", col = c("blue4", "red4", "gree
interaction.plot(TPUdata$MatrixSize, TPUdata$MatrixOperation, TPUdata$Runtime, ylab = "TPU Runtime (seco
                 xlab = "Matrix Size", trace.label = "Matrix Operation", col = c("blue4", "red4", "gree
```



```r
par(op)

#boxplot(Runtime~Processor*MatrixOperation)
#tapply(Runtime,list(Processor, MatrixOperation),mean)
#tapply(Runtime,MatrixOperation,mean)

boxplot(Runtime~Processor*MatrixOperation)
```



```r
round(tapply(Runtime,list(Processor, MatrixOperation),mean),digits=4)
```

```
##     Addition Inversion Multiplication
## CPU   2.5147  132.8553        93.4747
## GPU   0.1280   24.0220         1.5264
## TPU   0.0505    0.0199         0.0317
```

```r
#tapply(Runtime,MatrixOperation,mean)

# group_by(df, Processor) %>%
#   summarise(
#     count = n(),
#     mean = mean(Runtime, na.rm = TRUE),
#     sd = sd(Runtime, na.rm = TRUE)
#   )
#
# group_by(df, MatrixOperation) %>%
#   summarise(
#     count = n(),
#     mean = mean(Runtime, na.rm = TRUE),
#     sd = sd(Runtime, na.rm = TRUE)
#   )

detach(df)

size320data <- df %>% dplyr::filter(MatrixSize==320)
size640data <- df %>% dplyr::filter(MatrixSize==640)
size1280data <- df %>% dplyr::filter(MatrixSize==1280)
```

Mean of Runtime and Boxplot

```r
tapply(size320data$Runtime,list(size320data$Processor, size320data$MatrixOperation),mean)
```

```
##       Addition   Inversion Multiplication
## CPU 0.71350303 19.54677558    11.97371540
## GPU 0.07051702 18.21737680     0.22623463
## TPU 0.05276523  0.02184839     0.03002601
```

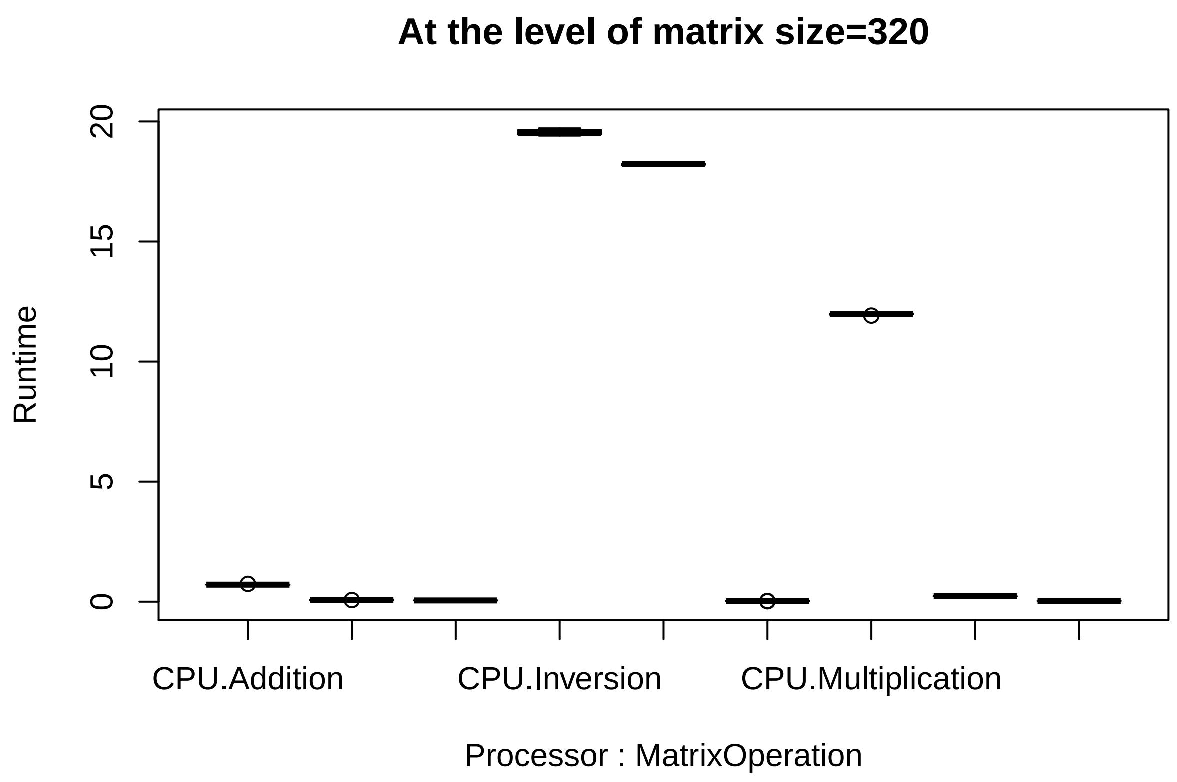
```r
tapply(size640data$Runtime,list(size640data$Processor, size640data$MatrixOperation),mean)
```

```
##       Addition    Inversion Multiplication
## CPU 2.84506497 129.50422368    83.41594014
## GPU 0.08734841  44.69854727     1.20817008
## TPU 0.04873762   0.02004447     0.02969995
```

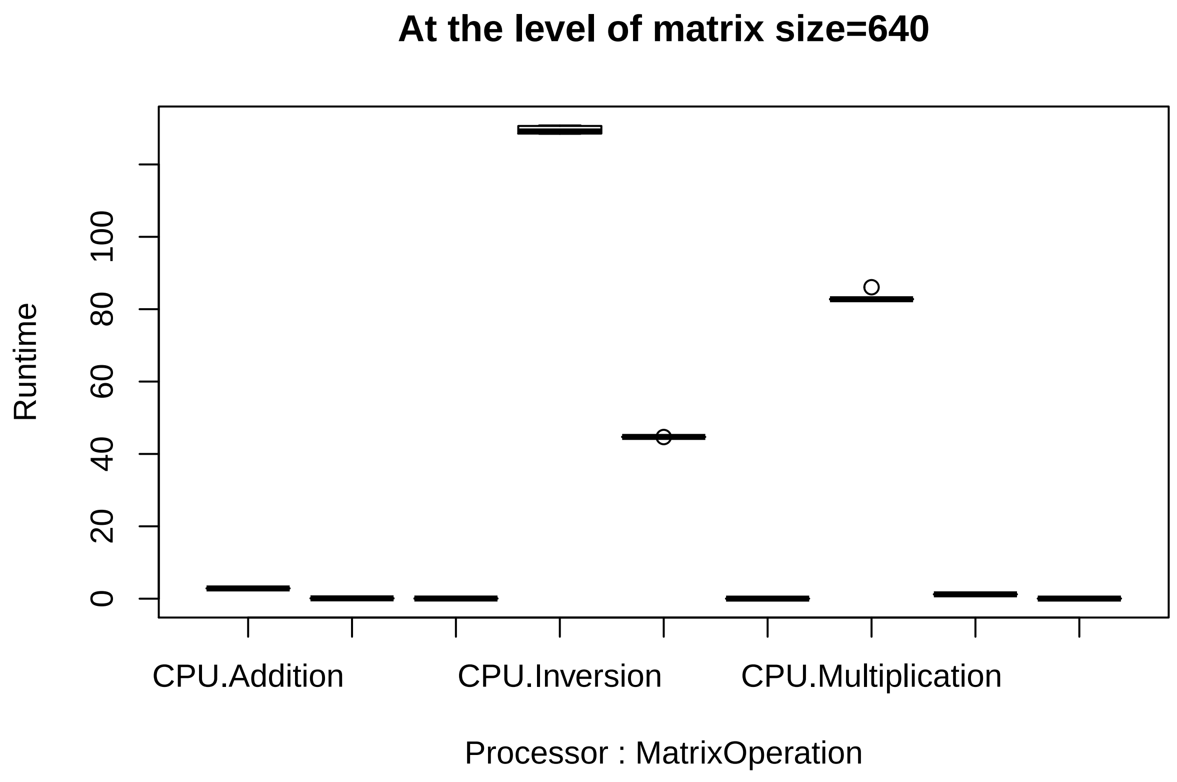
```r
tapply(size1280data$Runtime,list(size1280data$Processor, size1280data$MatrixOperation),mean)
```

```
##        Addition    Inversion Multiplication
## CPU 16.39168239 908.07733464   650.62470641
## GPU  0.51256037 113.87343698    10.33359056
## TPU  0.05090575   0.01981788     0.03385234
```

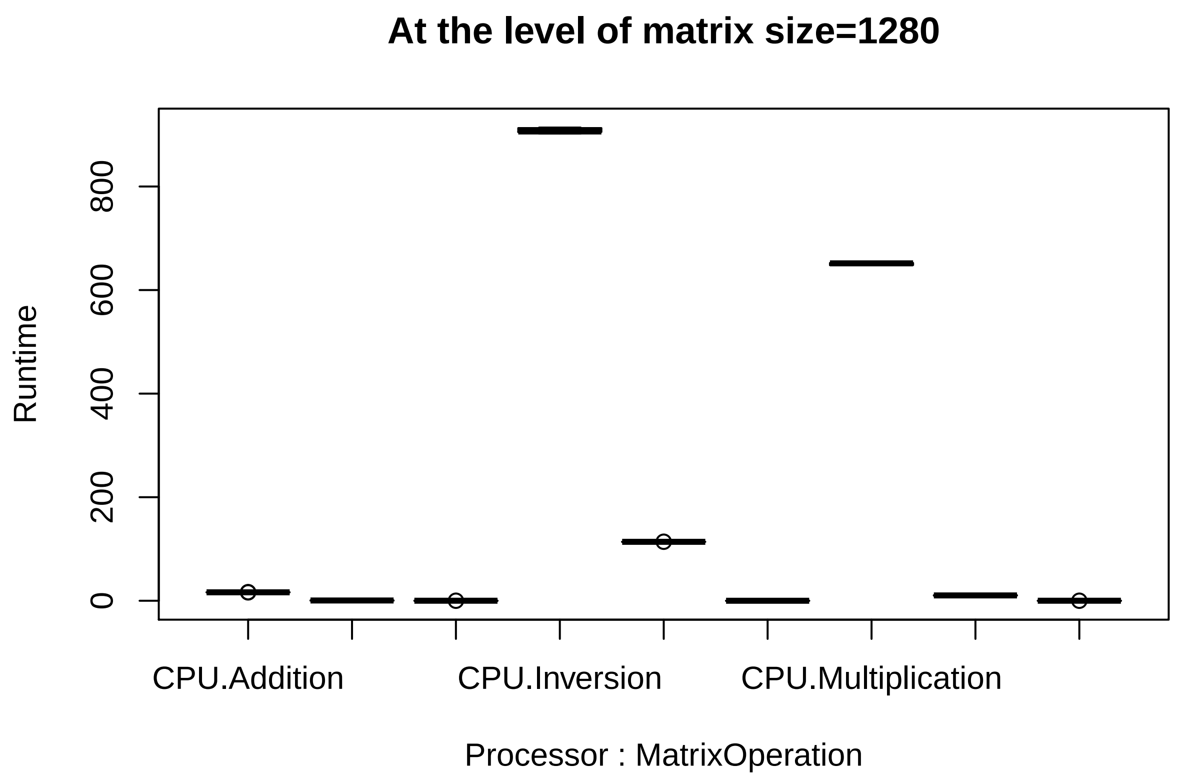
```r
boxplot(Runtime~Processor*MatrixOperation, data = size320data, main="At the level of matrix size=320")
```



```r
boxplot(Runtime~Processor*MatrixOperation, data = size640data, main="At the level of matrix size=640")
```



```r
boxplot(Runtime~Processor*MatrixOperation, data = size1280data, main="At the level of matrix size=1280"
```



Analysis

Anova

```r
############ 320
fit2<-lm(log10(Runtime)~Processor+MatrixOperation, data = size320data)
summary(fit2)
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ Processor + MatrixOperation, data = size320data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.84717 -0.41443 -0.05434  0.39193  0.83638
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.1912     0.1870   1.022   0.3128
## ProcessorGPU                   -0.9198     0.2048  -4.491 5.90e-05 ***
## ProcessorTPU                   -2.2309     0.2048 -10.892 1.55e-13 ***
## MatrixOperationInversion        1.1534     0.2048   5.631 1.56e-06 ***
## MatrixOperationMultiplication   0.4957     0.2048   2.420   0.0201 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5609 on 40 degrees of freedom
## Multiple R-squared:  0.7914, Adjusted R-squared:  0.7706
## F-statistic: 37.94 on 4 and 40 DF,  p-value: 4.091e-13
```

```
fit1<-lm(log10(Runtime)~Processor*MatrixOperation, data = size320data)
summary(fit1)
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ Processor * MatrixOperation, data = size320data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.064926 -0.008334 -0.000891  0.002071  0.157877
##
## Coefficients:
##                                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                                -0.14669    0.01471  -9.971 6.71e-12
## ProcessorGPU                               -1.00513    0.02081 -48.310  < 2e-16
## ProcessorTPU                               -1.13204    0.02081 -54.409  < 2e-16
## MatrixOperationInversion                    1.43776    0.02081  69.104  < 2e-16
## MatrixOperationMultiplication               1.22492    0.02081  58.874  < 2e-16
## ProcessorGPU:MatrixOperationInversion       0.97455    0.02942  33.121  < 2e-16
## ProcessorTPU:MatrixOperationInversion      -1.82762    0.02942 -62.113  < 2e-16
## ProcessorGPU:MatrixOperationMultiplication -0.71857    0.02942 -24.421  < 2e-16
## ProcessorTPU:MatrixOperationMultiplication -1.46896    0.02942 -49.924  < 2e-16
##
## (Intercept)                                ***
## ProcessorGPU                               ***
## ProcessorTPU                               ***
## MatrixOperationInversion                   ***
## MatrixOperationMultiplication              ***
## ProcessorGPU:MatrixOperationInversion      ***
## ProcessorTPU:MatrixOperationInversion      ***
## ProcessorGPU:MatrixOperationMultiplication ***
## ProcessorTPU:MatrixOperationMultiplication ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0329 on 36 degrees of freedom
## Multiple R-squared:  0.9994, Adjusted R-squared:  0.9992
## F-statistic:  6965 on 8 and 36 DF,  p-value: < 2.2e-16
```

```
anova(fit2, fit1)
```

```
## Analysis of Variance Table
##
## Model 1: log10(Runtime) ~ Processor + MatrixOperation
## Model 2: log10(Runtime) ~ Processor * MatrixOperation
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     40 12.586
## 2     36  0.039  4    12.547 2898.4 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod320<-aov(log10(Runtime)~Processor*MatrixOperation, data = size320data)
#anova(mod320)
Anova(mod320,type="III")
```

```
## Anova Table (Type III tests)
##
## Response: log10(Runtime)
##                          Sum Sq Df  F value    Pr(>F)
## (Intercept)              0.1076  1   99.421 6.713e-12 ***
## Processor                3.8465  2 1777.140 < 2.2e-16 ***
## MatrixOperation          6.0215  2 2782.019 < 2.2e-16 ***
## Processor:MatrixOperation 12.5469  4 2898.441 < 2.2e-16 ***
## Residuals                0.0390 36
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modF320<-lm(log10(Runtime)~Processor+MatrixOperation, data = size320data)
modA320<-lm(log10(Runtime)~Processor, data = size320data)
modB320<-lm(log10(Runtime)~MatrixOperation, data = size320data)

anova(modA320, modF320)
```

```
## Analysis of Variance Table
##
## Model 1: log10(Runtime) ~ Processor
## Model 2: log10(Runtime) ~ Processor + MatrixOperation
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     42 22.629
## 2     40 12.586  2    10.043 15.959 8.024e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(modB320, modF320)
```

```
## Analysis of Variance Table
##
## Model 1: log10(Runtime) ~ MatrixOperation
## Model 2: log10(Runtime) ~ Processor + MatrixOperation
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     42 50.295
## 2     40 12.586  2     37.71 59.923 9.271e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
################# 640
fit2_640<-lm(Runtime~Processor+MatrixOperation, data = size640data)
summary(fit2)
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ Processor + MatrixOperation, data = size320data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.84717 -0.41443 -0.05434  0.39193  0.83638
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.1912     0.1870   1.022   0.3128
## ProcessorGPU                   -0.9198     0.2048  -4.491 5.90e-05 ***
## ProcessorTPU                   -2.2309     0.2048 -10.892 1.55e-13 ***
## MatrixOperationInversion        1.1534     0.2048   5.631 1.56e-06 ***
## MatrixOperationMultiplication   0.4957     0.2048   2.420   0.0201 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5609 on 40 degrees of freedom
## Multiple R-squared:  0.7914, Adjusted R-squared:  0.7706
## F-statistic: 37.94 on 4 and 40 DF,  p-value: 4.091e-13
```

```
fit1_640<-lm(Runtime~Processor*MatrixOperation, data = size640data)
summary(fit1_640)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor * MatrixOperation, data = size640data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04780 -0.00908 -0.00014  0.00088  2.65489
##
## Coefficients:
##                                             Estimate Std. Error  t value
## (Intercept)                                   2.8451     0.2781   10.229
## ProcessorGPU                                 -2.7577     0.3933   -7.011
## ProcessorTPU                                 -2.7963     0.3933   -7.109
## MatrixOperationInversion                    126.6592     0.3933  322.008
## MatrixOperationMultiplication                80.5709     0.3933  204.837
## ProcessorGPU:MatrixOperationInversion       -82.0480     0.5563 -147.497
## ProcessorTPU:MatrixOperationInversion      -126.6879     0.5563 -227.746
## ProcessorGPU:MatrixOperationMultiplication  -79.4501     0.5563 -142.827
## ProcessorTPU:MatrixOperationMultiplication  -80.5899     0.5563 -144.876
##                                            Pr(>|t|)
## (Intercept)                                3.38e-12 ***
## ProcessorGPU                               3.18e-08 ***
## ProcessorTPU                               2.36e-08 ***
## MatrixOperationInversion                    < 2e-16 ***
## MatrixOperationMultiplication               < 2e-16 ***
## ProcessorGPU:MatrixOperationInversion       < 2e-16 ***
## ProcessorTPU:MatrixOperationInversion       < 2e-16 ***
## ProcessorGPU:MatrixOperationMultiplication  < 2e-16 ***
## ProcessorTPU:MatrixOperationMultiplication  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6219 on 36 degrees of freedom
## Multiple R-squared:  0.9998, Adjusted R-squared:  0.9998
## F-statistic: 2.928e+04 on 8 and 36 DF,  p-value: < 2.2e-16
```

```
anova(fit2_640, fit1_640)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ Processor + MatrixOperation
## Model 2: Runtime ~ Processor * MatrixOperation
##   Res.Df     RSS Df Sum of Sq     F    Pr(>F)
## 1     40 23128.8
## 2     36    13.9  4     23115 14940 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod640<-aov(Runtime~Processor*MatrixOperation, data = size640data)
Anova(mod640,type="III")
```

```
## Anova Table (Type III tests)
##
## Response: Runtime
##                           Sum Sq Df   F value    Pr(>F)
## (Intercept)                   40  1   104.635 3.380e-12 ***
## Processor                     26  2    33.235 6.649e-09 ***
## MatrixOperation            41097  2 53125.511 < 2.2e-16 ***
## Processor:MatrixOperation  23115  4 14940.049 < 2.2e-16 ***
## Residuals                     14 36
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modF640<-lm(Runtime~Processor+MatrixOperation, data = size640data)
modA640<-lm(Runtime~Processor, data = size640data)
modB640<-lm(Runtime~MatrixOperation, data = size640data)

anova(modA640, modF640)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ Processor
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1     42 47583
## 2     40 23129  2     24454 21.146 5.421e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(modB640, modF640)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ MatrixOperation
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1     42 66151
## 2     40 23129  2     43023 37.203 7.452e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##################### 1280
fit2_1280<-lm(Runtime~Processor+MatrixOperation, data = size1280data)
summary(fit2)
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ Processor + MatrixOperation, data = size320data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.84717 -0.41443 -0.05434  0.39193  0.83638
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.1912     0.1870   1.022   0.3128
## ProcessorGPU                   -0.9198     0.2048  -4.491 5.90e-05 ***
## ProcessorTPU                   -2.2309     0.2048 -10.892 1.55e-13 ***
## MatrixOperationInversion        1.1534     0.2048   5.631 1.56e-06 ***
## MatrixOperationMultiplication   0.4957     0.2048   2.420   0.0201 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5609 on 40 degrees of freedom
## Multiple R-squared:  0.7914, Adjusted R-squared:  0.7706
## F-statistic: 37.94 on 4 and 40 DF,  p-value: 4.091e-13
```

```
fit1_1280<-lm(Runtime~Processor*MatrixOperation, data = size1280data)
summary(fit1_1280)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor * MatrixOperation, data = size1280data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7751 -0.0345 -0.0008  0.0132  5.7155
##
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                                  16.3917     0.8192   20.01
## ProcessorGPU                                -15.8791     1.1585  -13.71
## ProcessorTPU                                -16.3408     1.1585  -14.11
## MatrixOperationInversion                    891.6857     1.1585  769.69
## MatrixOperationMultiplication               634.2330     1.1585  547.46
## ProcessorGPU:MatrixOperationInversion      -778.3248     1.6384 -475.06
## ProcessorTPU:MatrixOperationInversion      -891.7167     1.6384 -544.27
## ProcessorGPU:MatrixOperationMultiplication -624.4120     1.6384 -381.12
## ProcessorTPU:MatrixOperationMultiplication -634.2501     1.6384 -387.12
##                                            Pr(>|t|)
## (Intercept)                                 < 2e-16 ***
## ProcessorGPU                               7.42e-16 ***
## ProcessorTPU                               3.09e-16 ***
## MatrixOperationInversion                    < 2e-16 ***
## MatrixOperationMultiplication               < 2e-16 ***
## ProcessorGPU:MatrixOperationInversion       < 2e-16 ***
## ProcessorTPU:MatrixOperationInversion       < 2e-16 ***
## ProcessorGPU:MatrixOperationMultiplication  < 2e-16 ***
## ProcessorTPU:MatrixOperationMultiplication  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.832 on 36 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.751e+05 on 8 and 36 DF,  p-value: < 2.2e-16
```

```
anova(fit2_1280, fit1_1280)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ Processor + MatrixOperation
## Model 2: Runtime ~ Processor * MatrixOperation
##   Res.Df     RSS Df Sum of Sq     F    Pr(>F)
## 1     40 1281658
## 2     36     121  4   1281538 95485 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modF1280<-lm(Runtime~Processor+MatrixOperation, data = size1280data)
modA1280<-lm(Runtime~Processor, data = size1280data)
modB1280<-lm(Runtime~MatrixOperation, data = size1280data)

anova(modA1280, modF1280)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ Processor
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1     42 2145628
## 2     40 1281658  2    863970 13.482 3.345e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(modB1280, modF1280)
```

```
## Analysis of Variance Table
##
## Model 1: Runtime ~ MatrixOperation
## Model 2: Runtime ~ Processor + MatrixOperation
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1     42 3837050
## 2     40 1281658  2   2555392 39.876 2.989e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod1280<-aov(Runtime~Processor*MatrixOperation, data = size1280data)
Anova(mod1280,type="III")
```

```
## Anova Table (Type III tests)
##
## Response: Runtime
##                            Sum Sq Df   F value    Pr(>F)
## (Intercept)                  1343  1    400.39 < 2.2e-16 ***
## Processor                     866  2    128.99 < 2.2e-16 ***
## MatrixOperation           2106061  2 313838.85 < 2.2e-16 ***
## Processor:MatrixOperation 1281538  4  95485.43 < 2.2e-16 ***
## Residuals                     121 36
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#also fix processor, 1 way ANOVA
size320CPUdata <- data %>% dplyr::filter(MatrixSize==320 & Processor=="CPU")
size640CPUdata <- data %>% dplyr::filter(MatrixSize==640 & Processor=="CPU")
size1280CPUdata <- data %>% dplyr::filter(MatrixSize==1280 & Processor=="CPU")

size320GPUdata <- data %>% dplyr::filter(MatrixSize==320 & Processor=="GPU")
size640GPUdata <- data %>% dplyr::filter(MatrixSize==640 & Processor=="GPU")
size1280GPUdata <- data %>% filter(MatrixSize==1280 & Processor=="GPU")

size320TPUdata <- data %>% dplyr::filter(MatrixSize==320 & Processor=="TPU")
size640TPUdata <- data %>% dplyr::filter(MatrixSize==640 & Processor=="TPU")
size1280TPUdata <- data %>% dplyr::filter(MatrixSize==1280 & Processor=="TPU")

mod320CPU<-lm(Runtime~MatrixOperation, data = size320CPUdata)
summary(mod320CPU)
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixOperation, data = size320CPUdata)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.135726 -0.025213 -0.004924  0.022535  0.166785
##
## Coefficients:
```

```
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.71350    0.03425   20.83 8.67e-11 ***
## MatrixOperationInversion       18.83327    0.04844  388.78  < 2e-16 ***
## MatrixOperationMultiplication 11.26021    0.04844  232.45  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07659 on 12 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 7.654e+04 on 2 and 12 DF,  p-value: < 2.2e-16
```

```
mod320GPU<-lm(Runtime~MatrixOperation, data = size320GPUdata)
summary(mod320GPU)
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixOperation, data = size320GPUdata)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.037341 -0.001891 -0.000090  0.003157  0.027248
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.070517   0.006911   10.20 2.88e-07 ***
## MatrixOperationInversion       18.146860   0.009774 1856.69  < 2e-16 ***
## MatrixOperationMultiplication  0.155718   0.009774   15.93 1.95e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01545 on 12 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 2.279e+06 on 2 and 12 DF,  p-value: < 2.2e-16
```

```
mod320TPU<-lm(Runtime~MatrixOperation, data = size320TPUdata)
summary(mod320TPU)
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixOperation, data = size320TPUdata)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.004670 -0.001985 -0.001053  0.001259  0.009004
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.052765   0.001724  30.613 9.26e-13 ***
## MatrixOperationInversion       -0.030917   0.002438 -12.684 2.60e-08 ***
## MatrixOperationMultiplication -0.022739   0.002438  -9.329 7.55e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.003854 on 12 degrees of freedom
## Multiple R-squared:  0.9351, Adjusted R-squared:  0.9242
## F-statistic: 86.39 on 2 and 12 DF,  p-value: 7.504e-08
```

```
mod640CPU<-lm(Runtime~MatrixOperation, data = size640CPUdata)
summary(mod640CPU)
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixOperation, data = size640CPUdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04780 -0.67767 -0.04855  0.03665  2.65489
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     2.8451     0.4817   5.906 7.18e-05 ***
## MatrixOperationInversion      126.6592     0.6812 185.922  < 2e-16 ***
## MatrixOperationMultiplication  80.5709     0.6812 118.269  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.077 on 12 degrees of freedom
## Multiple R-squared:  0.9997, Adjusted R-squared:  0.9996
## F-statistic: 1.771e+04 on 2 and 12 DF,  p-value: < 2.2e-16
```

```
mod640GPU<-lm(Runtime~MatrixOperation, data = size640GPUdata)
summary(mod640GPU)
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixOperation, data = size640GPUdata)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0296608 -0.0026586 -0.0000269  0.0062409  0.0174795
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.087348   0.005132   17.02 9.09e-10 ***
## MatrixOperationInversion      44.611199   0.007258 6146.63  < 2e-16 ***
## MatrixOperationMultiplication  1.120822   0.007258  154.43  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01148 on 12 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 2.457e+07 on 2 and 12 DF,  p-value: < 2.2e-16
```

```
mod640TPU<-lm(Runtime~MatrixOperation, data = size640TPUdata)
summary(mod640TPU)
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixOperation, data = size640TPUdata)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0013937 -0.0007722  0.0003468  0.0007116  0.0008858
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      0.0487376  0.0003826  127.39  < 2e-16 ***
## MatrixOperationInversion      -0.0286932  0.0005411  -53.03 1.33e-15 ***
## MatrixOperationMultiplication -0.0190377  0.0005411  -35.19 1.77e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0008555 on 12 degrees of freedom
## Multiple R-squared:  0.9959, Adjusted R-squared:  0.9952
## F-statistic:  1456 on 2 and 12 DF,  p-value: 4.774e-15
```

```r
mod1280CPU<-lm(Runtime~MatrixOperation, data = size1280CPUdata)
summary(mod1280CPU)
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixOperation, data = size1280CPUdata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7751 -2.1233 -0.0388  1.3311  5.7155
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     16.392      1.416   11.58 7.19e-08 ***
## MatrixOperationInversion       891.686      2.002  445.43  < 2e-16 ***
## MatrixOperationMultiplication  634.233      2.002  316.82  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.165 on 12 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 1.051e+05 on 2 and 12 DF,  p-value: < 2.2e-16
```

```r
mod1280GPU<-lm(Runtime~MatrixOperation, data = size1280GPUdata)
summary(mod1280GPU)
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixOperation, data = size1280GPUdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.44634 -0.03290 -0.00015  0.01275  0.40972
```

```
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.51256    0.09736   5.265    2e-04 ***
## MatrixOperationInversion      113.36088    0.13768 823.339   <2e-16 ***
## MatrixOperationMultiplication   9.82103    0.13768  71.330   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2177 on 12 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 4.162e+05 on 2 and 12 DF,  p-value: < 2.2e-16
```

```
mod1280TPU<-lm(Runtime~MatrixOperation, data = size1280TPUdata)
summary(mod1280TPU)
```

```
##
## Call:
## lm(formula = Runtime ~ MatrixOperation, data = size1280TPUdata)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0052021 -0.0029268 -0.0008882  0.0003531  0.0168402
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.050906   0.002731  18.640 3.17e-10 ***
## MatrixOperationInversion      -0.031088   0.003862  -8.049 3.53e-06 ***
## MatrixOperationMultiplication -0.017053   0.003862  -4.415 0.000842 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.006107 on 12 degrees of freedom
## Multiple R-squared:  0.8441, Adjusted R-squared:  0.8182
## F-statistic:  32.5 on 2 and 12 DF,  p-value: 1.434e-05
```

```
#grand overall model
unique(data$MatrixSize)
```

```
## [1]   10   20   40   80  160  320  640 1280
```

```
mod<-aov(Runtime~Processor+MatrixOperation+as.factor(MatrixSize), data = df)
summary(mod)
```

```
##                        Df  Sum Sq Mean Sq F value   Pr(>F)
## Processor               2  418909  209455  17.292 6.92e-08 ***
## MatrixOperation         2  160590   80295   6.629  0.00149 **
## as.factor(MatrixSize)   7 1356897  193842  16.003  < 2e-16 ***
## Residuals             348 4215313   12113
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
mod2<-lm(Runtime~Processor+MatrixOperation+as.factor(MatrixSize), data = df)
summary(mod2)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor + MatrixOperation + as.factor(MatrixSize),
##     data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -193.22  -48.01    2.87   24.60  652.92
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      20.76253   20.09392   1.033 0.302193
## ProcessorGPU                    -67.72278   14.20855  -4.766 2.76e-06 ***
## ProcessorTPU                    -76.24754   14.20855  -5.366 1.47e-07 ***
## MatrixOperationInversion         51.40131   14.20855   3.618 0.000341 ***
## MatrixOperationMultiplication    30.77990   14.20855   2.166 0.030967 *
## as.factor(MatrixSize)20           0.04295   23.20246   0.002 0.998524
## as.factor(MatrixSize)40           0.16240   23.20246   0.007 0.994420
## as.factor(MatrixSize)80           0.43837   23.20246   0.019 0.984937
## as.factor(MatrixSize)160          1.23193   23.20246   0.053 0.957687
## as.factor(MatrixSize)320          5.48415   23.20246   0.236 0.813292
## as.factor(MatrixSize)640         28.92915   23.20246   1.247 0.213304
## as.factor(MatrixSize)1280       188.71361   23.20246   8.133 7.45e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 110.1 on 348 degrees of freedom
## Multiple R-squared:  0.3148, Adjusted R-squared:  0.2931
## F-statistic: 14.53 on 11 and 348 DF,  p-value: < 2.2e-16
```

```r
#res <- resid(mod2)
#plot(fitted(mod2), res)
#abline(0,0)
mod3<-lm(Runtime~Processor*MatrixOperation+as.factor(MatrixSize), data = df)
summary(mod3)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor * MatrixOperation + as.factor(MatrixSize),
##     data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -160.59  -60.48   21.96   27.09  620.35
##
## Coefficients:
##                                              Estimate Std. Error t value
## (Intercept)                                 -25.61060   22.74453  -1.126
## ProcessorGPU                                 -2.38672   24.12422  -0.099
## ProcessorTPU                                 -2.46422   24.12422  -0.102
## MatrixOperationInversion                    130.34056   24.12422   5.403
## MatrixOperationMultiplication                90.96003   24.12422   3.770
## as.factor(MatrixSize)20                       0.04295   22.74453   0.002
## as.factor(MatrixSize)40                       0.16240   22.74453   0.007
## as.factor(MatrixSize)80                       0.43837   22.74453   0.019
## as.factor(MatrixSize)160                      1.23193   22.74453   0.054
## as.factor(MatrixSize)320                      5.48415   22.74453   0.241
## as.factor(MatrixSize)640                     28.92915   22.74453   1.272
## as.factor(MatrixSize)1280                   188.71361   22.74453   8.297
## ProcessorGPU:MatrixOperationInversion      -106.44661   34.11680  -3.120
## ProcessorTPU:MatrixOperationInversion      -130.37115   34.11680  -3.821
## ProcessorGPU:MatrixOperationMultiplication  -89.56157   34.11680  -2.625
## ProcessorTPU:MatrixOperationMultiplication  -90.97880   34.11680  -2.667
##                                            Pr(>|t|)
## (Intercept)                                0.260946
## ProcessorGPU                               0.921248
## ProcessorTPU                               0.918699
## MatrixOperationInversion                   1.23e-07 ***
## MatrixOperationMultiplication              0.000192 ***
## as.factor(MatrixSize)20                    0.998495
## as.factor(MatrixSize)40                    0.994307
## as.factor(MatrixSize)80                    0.984634
## as.factor(MatrixSize)160                   0.956836
## as.factor(MatrixSize)320                   0.809606
## as.factor(MatrixSize)640                   0.204262
## as.factor(MatrixSize)1280                  2.45e-15 ***
## ProcessorGPU:MatrixOperationInversion      0.001961 **
## ProcessorTPU:MatrixOperationInversion      0.000158 ***
## ProcessorGPU:MatrixOperationMultiplication 0.009048 **
## ProcessorTPU:MatrixOperationMultiplication 0.008022 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 107.9 on 344 degrees of freedom
## Multiple R-squared:  0.3491, Adjusted R-squared:  0.3207
## F-statistic:  12.3 on 15 and 344 DF,  p-value: < 2.2e-16
```

```
modN<-aov(Runtime~Processor+MatrixOperation+MatrixSize, data = df)
summary(modN)
```

```
##                  Df  Sum Sq Mean Sq F value   Pr(>F)
## Processor         2  418909  209455  16.915 9.64e-08 ***
## MatrixOperation   2  160590   80295   6.484  0.00172 **
## MatrixSize        1 1188587 1188587  95.984  < 2e-16 ***
## Residuals       354 4383623   12383
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod2N<-lm(Runtime~Processor+MatrixOperation+MatrixSize, data = df)
summary(mod2N)
```

```
##
```

```
## Call:
## lm(formula = Runtime ~ Processor + MatrixOperation + MatrixSize,
##     data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -166.02  -49.26   -6.72   27.83  680.12
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     4.65633   13.86977   0.336 0.737283
## ProcessorGPU                  -67.72278   14.36612  -4.714 3.50e-06 ***
## ProcessorTPU                  -76.24754   14.36612  -5.307 1.97e-07 ***
## MatrixOperationInversion       51.40131   14.36612   3.578 0.000395 ***
## MatrixOperationMultiplication  30.77990   14.36612   2.143 0.032832 *
## MatrixSize                      0.13877    0.01416   9.797  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 111.3 on 354 degrees of freedom
## Multiple R-squared:  0.2874, Adjusted R-squared:  0.2773
## F-statistic: 28.56 on 5 and 354 DF,  p-value: < 2.2e-16
```

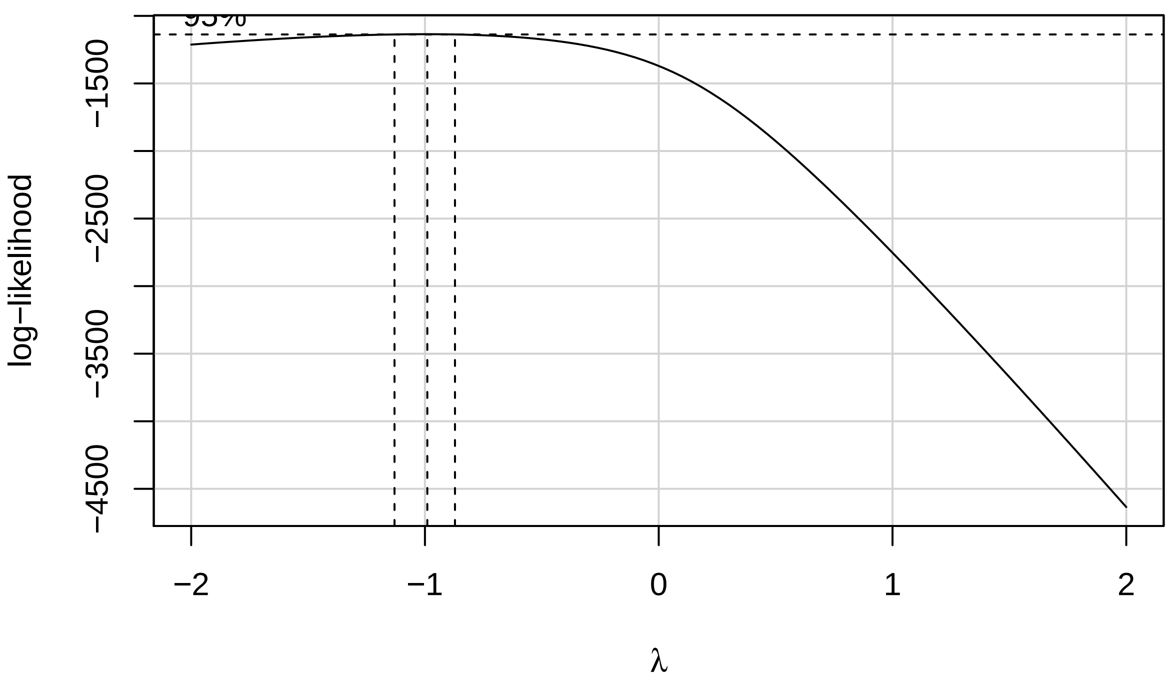
```
#res <- resid(mod2N)
#plot(fitted(mod2N), res)
#abline(0,0)

modNI<-aov(Runtime~Processor*MatrixOperation+MatrixSize, data = df)
summary(modNI)
```

```
##                            Df  Sum Sq Mean Sq F value   Pr(>F)
## Processor                   2  418909  209455  17.570 5.35e-08 ***
## MatrixOperation             2  160590   80295   6.736  0.00135 **
## MatrixSize                  1 1188587 1188587  99.706  < 2e-16 ***
## Processor:MatrixOperation   4  211306   52826   4.431  0.00166 **
## Residuals                 350 4172318   11921
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod2NI<-lm(Runtime~Processor*MatrixOperation+MatrixSize, data = df)
summary(mod2NI)
```

```
##
## Call:
## lm(formula = Runtime ~ Processor * MatrixOperation + MatrixSize,
##     data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -133.39  -50.94    8.95   37.59  647.55
##
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                              -41.7168    17.8226  -2.341
## ProcessorGPU                              -2.3867    24.4140  -0.098
## ProcessorTPU                              -2.4642    24.4140  -0.101
## MatrixOperationInversion                 130.3406    24.4140   5.339
## MatrixOperationMultiplication             90.9600    24.4140   3.726
## MatrixSize                                 0.1388     0.0139   9.985
## ProcessorGPU:MatrixOperationInversion   -106.4466    34.5267  -3.083
## ProcessorTPU:MatrixOperationInversion   -130.3712    34.5267  -3.776
## ProcessorGPU:MatrixOperationMultiplication  -89.5616    34.5267  -2.594
## ProcessorTPU:MatrixOperationMultiplication  -90.9788    34.5267  -2.635
##                                            Pr(>|t|)
## (Intercept)                                0.019810 *
## ProcessorGPU                               0.922179
## ProcessorTPU                               0.919660
## MatrixOperationInversion                   1.69e-07 ***
## MatrixOperationMultiplication              0.000227 ***
## MatrixSize                                  < 2e-16 ***
## ProcessorGPU:MatrixOperationInversion      0.002212 **
## ProcessorTPU:MatrixOperationInversion      0.000187 ***
## ProcessorGPU:MatrixOperationMultiplication 0.009886 **
## ProcessorTPU:MatrixOperationMultiplication 0.008787 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 109.2 on 350 degrees of freedom
## Multiple R-squared:  0.3218, Adjusted R-squared:  0.3043
## F-statistic: 18.45 on 9 and 350 DF,  p-value: < 2.2e-16
```

```
#res <- resid(mod2NI)
#plot(fitted(mod2NI), res)
#abline(0,0)

#boxcox transform
boxCox(mod2N, family="yjPower", plotit = TRUE)
```



```
lambda=-1
depvar.transformed <- yjPower(data$Runtime, lambda)
Processor <- data$Processor
MatrixOperation <- data$MatrixOperation
MatrixSize <- data$MatrixSize
mod2NT<-lm(depvar.transformed~Processor+MatrixOperation+MatrixSize)
summary(mod2NT)
```

```
##
## Call:
## lm(formula = depvar.transformed ~ Processor + MatrixOperation +
##     MatrixSize)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.56272 -0.17233 -0.05672  0.21517  0.50449
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   1.859e-01  2.873e-02   6.472 3.22e-10 ***
## ProcessorGPU                 -6.921e-02  2.975e-02  -2.326 0.020587 *
## ProcessorTPU                 -4.210e-01  2.975e-02 -14.148  < 2e-16 ***
## MatrixOperationInversion      3.280e-01  2.975e-02  11.023  < 2e-16 ***
## MatrixOperationMultiplication 1.107e-01  2.975e-02   3.719 0.000233 ***
## MatrixSize                    3.815e-04  2.934e-05  13.006  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2305 on 354 degrees of freedom
## Multiple R-squared:  0.5974, Adjusted R-squared:  0.5917
## F-statistic:   105 on 5 and 354 DF,  p-value: < 2.2e-16
```

```r
#res <- resid(mod2NT)
#plot(fitted(mod2NT), res)
#abline(0,0)
```

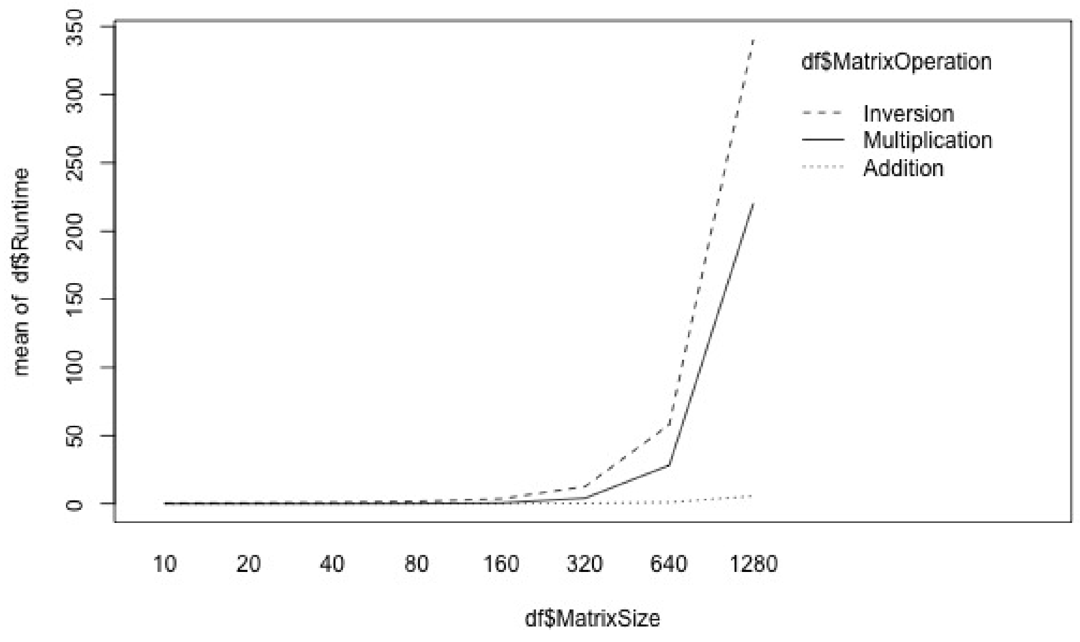
Zhanhao Zhang Part II

General Visualization

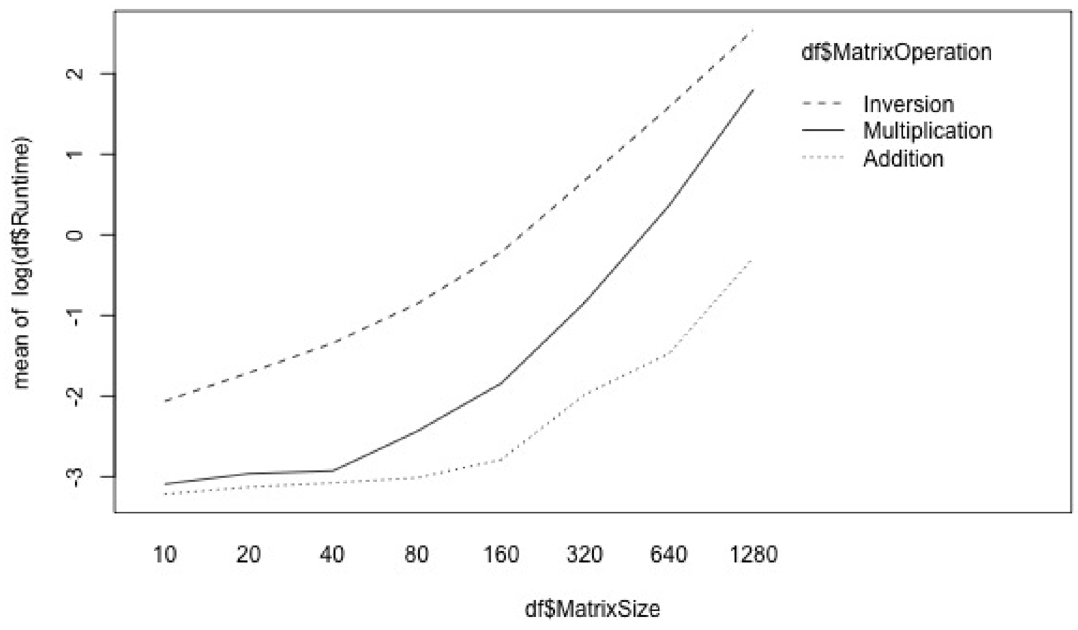
Lixian Chen Part II: Plots

Interaction Plots

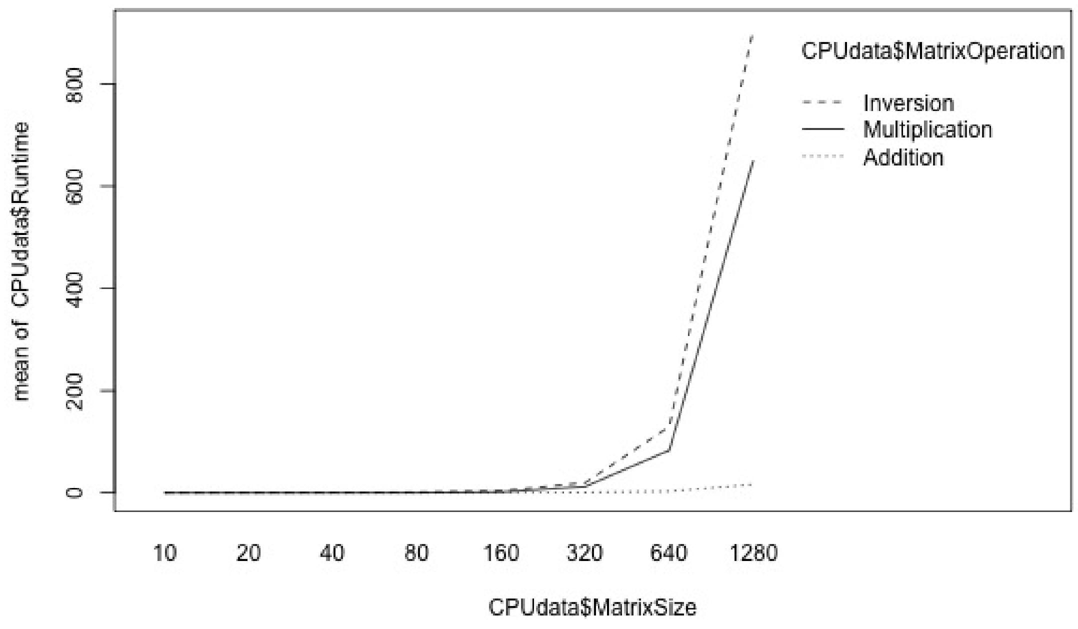
```r
jpeg(filename = "../figs/interaction_size_time.jpeg", width = 600, height = 400,quality = 10000)
interaction.plot(df$MatrixSize, df$MatrixOperation, df$Runtime)
while (!is.null(dev.list()))  dev.off()
```



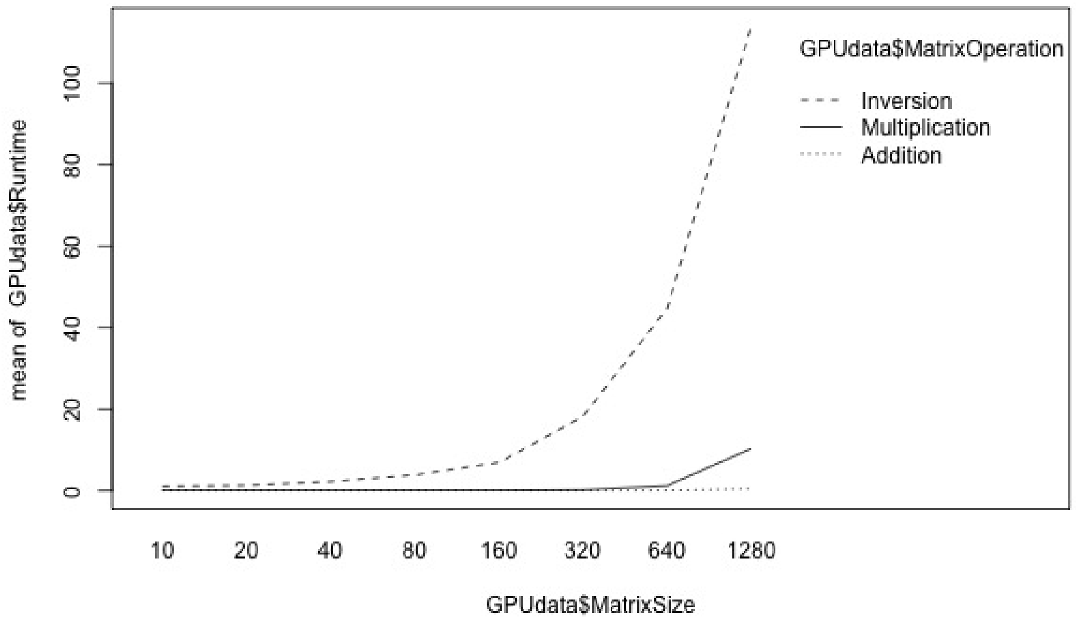
```r
jpeg(filename = "../figs/interaction_size_log_time.jpeg", width = 600, height = 400,quality = 10000)
interaction.plot(df$MatrixSize, df$MatrixOperation, log(df$Runtime))
while (!is.null(dev.list()))  dev.off()
```



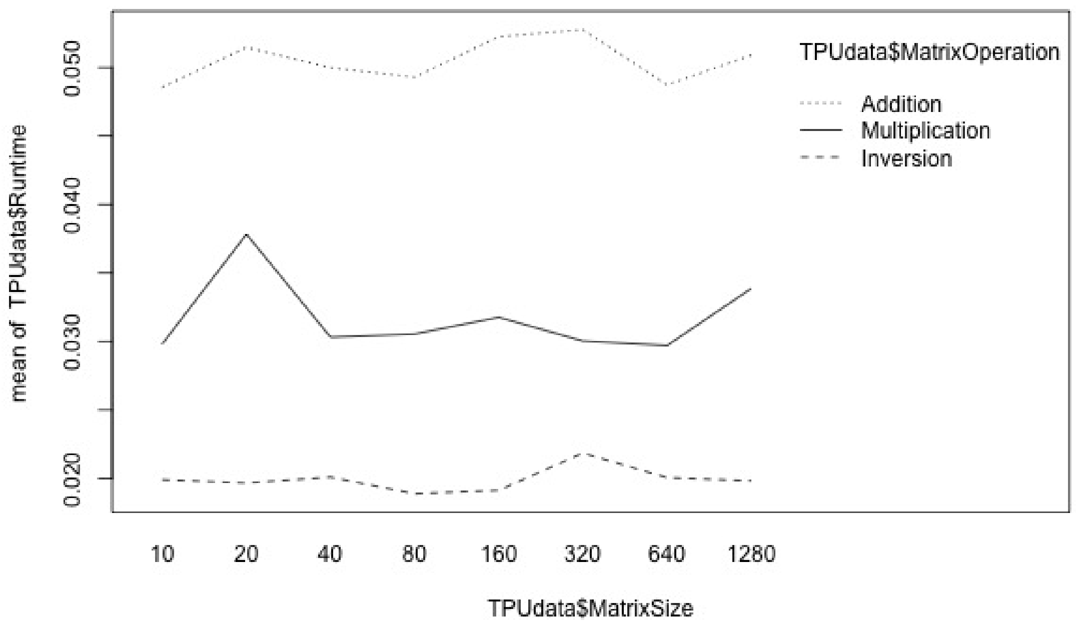
```r
jpeg(filename = "../figs/interaction_CPU_size_time.jpeg", width = 600, height = 400,quality = 10000)
interaction.plot(CPUdata$MatrixSize, CPUdata$MatrixOperation, CPUdata$Runtime)
while (!is.null(dev.list()))  dev.off()
```



```
jpeg(filename = "../figs/interaction_GPU_size_time.jpeg", width = 600, height = 400,quality = 10000)
interaction.plot(GPUdata$MatrixSize, GPUdata$MatrixOperation, GPUdata$Runtime)
while (!is.null(dev.list()))  dev.off()
```

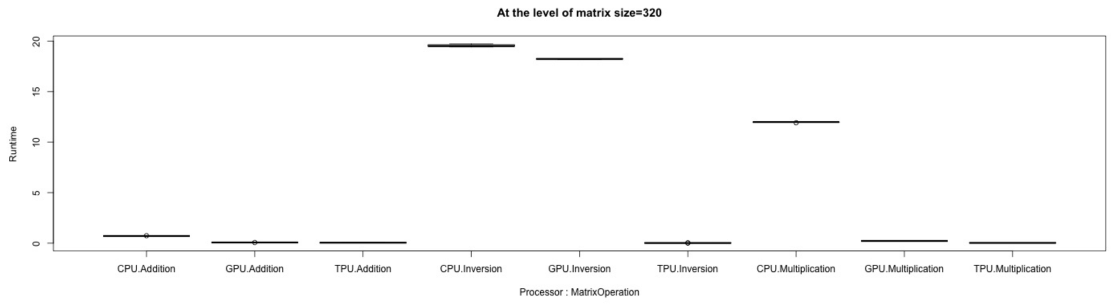


```
jpeg(filename = "../figs/interaction_TPU_size_time.jpeg", width = 600, height = 400,quality = 10000)
interaction.plot(TPUdata$MatrixSize, TPUdata$MatrixOperation, TPUdata$Runtime)
while (!is.null(dev.list()))  dev.off()
```

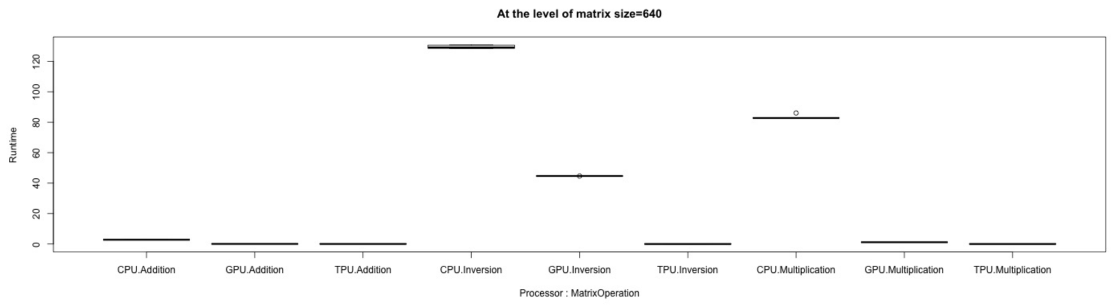


Boxplots

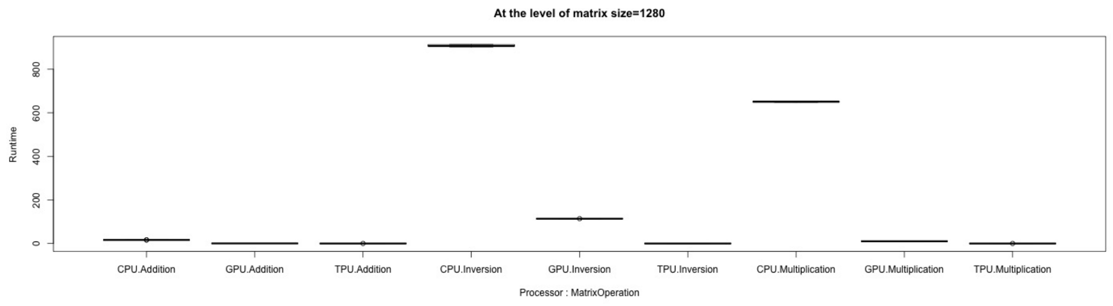
```
jpeg(filename = "../figs/Operation_vs_runtime_size320.jpeg", width = 1400, height = 400,quality = 10000
boxplot(Runtime~Processor*MatrixOperation, data = size320data, main="At the level of matrix size=320")
while (!is.null(dev.list()))  dev.off()
```



```
jpeg(filename = "../figs/Operation_vs_runtime_size640.jpeg", width = 1400, height = 400,quality = 10000
boxplot(Runtime~Processor*MatrixOperation, data = size640data, main="At the level of matrix size=640")
while (!is.null(dev.list()))  dev.off()
```



```r
jpeg(filename = "../figs/Operation_vs_runtime_size1280.jpeg", width = 1400, height = 400,quality = 1000
boxplot(Runtime~Processor*MatrixOperation, data = size1280data, main="At the level of matrix size=1280"
while (!is.null(dev.list()))  dev.off()
```

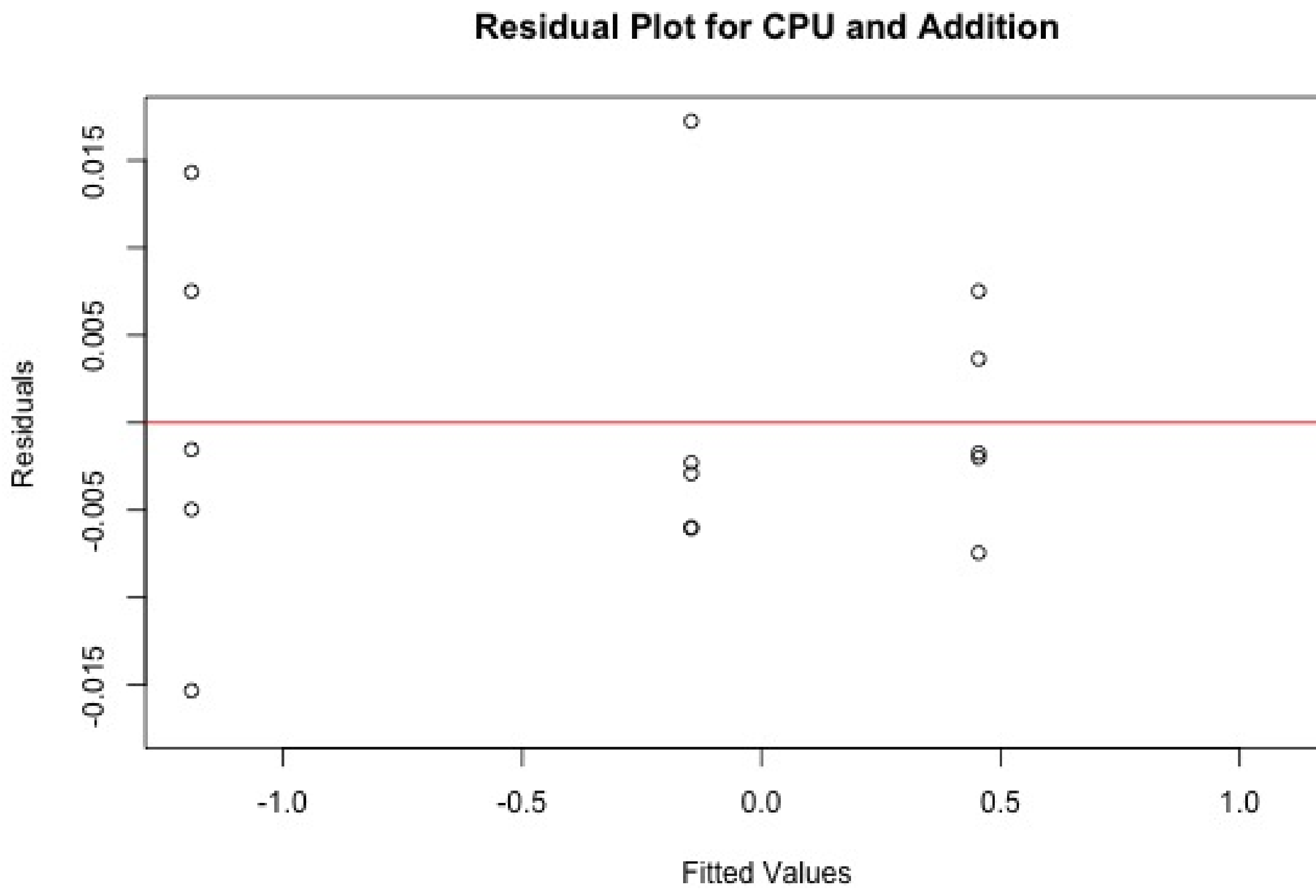


Jingbin Cao Part II

Plot Anova Residuals

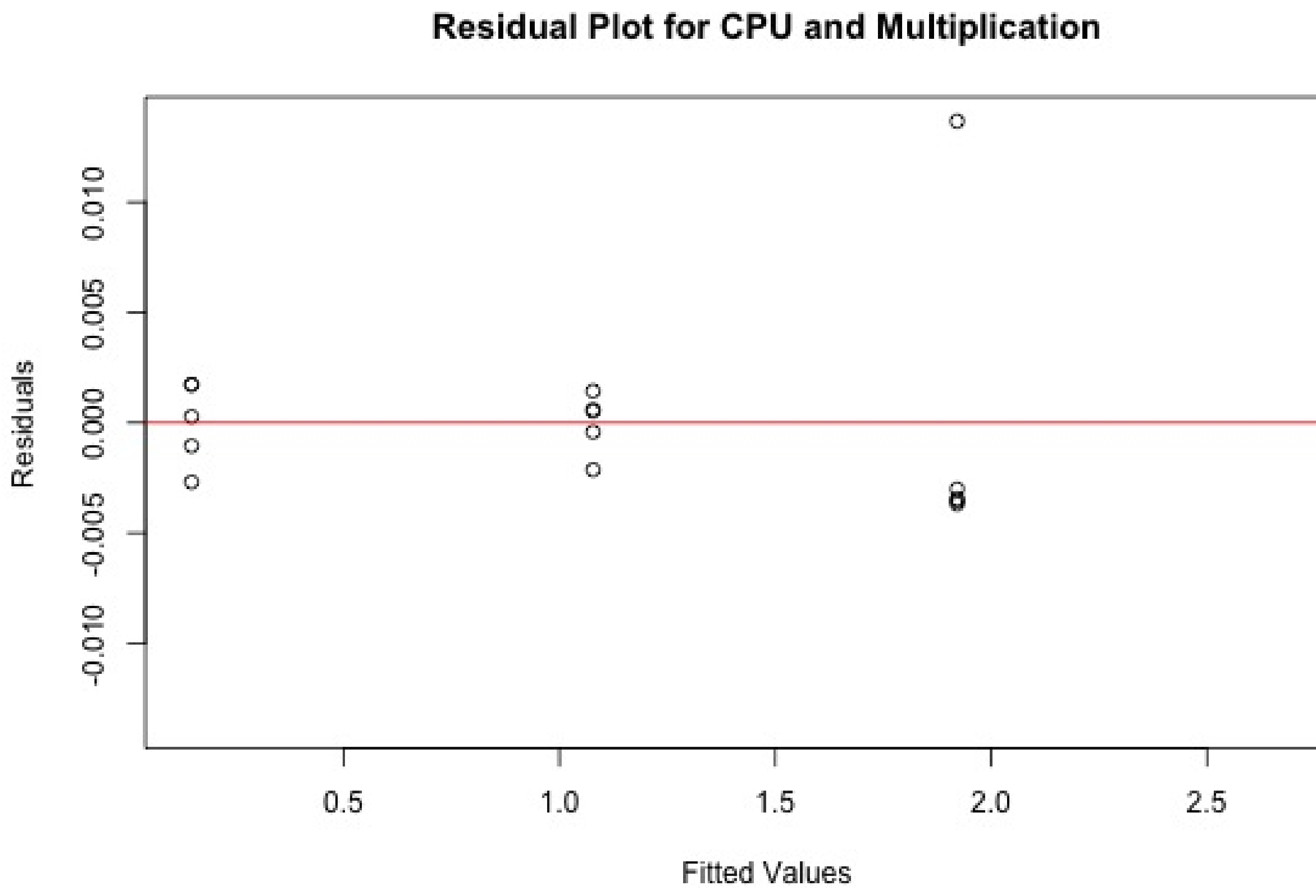
CPU Add

```r
jpeg(filename = "../figs/res_cpu_add.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(mod_cpu_add$residuals))
plot(mod_cpu_add$fitted.values,mod_cpu_add$residuals, ylim=c(-m, m), main = "Residual Plot for CPU and 
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_cpu_add.jpeg")
```



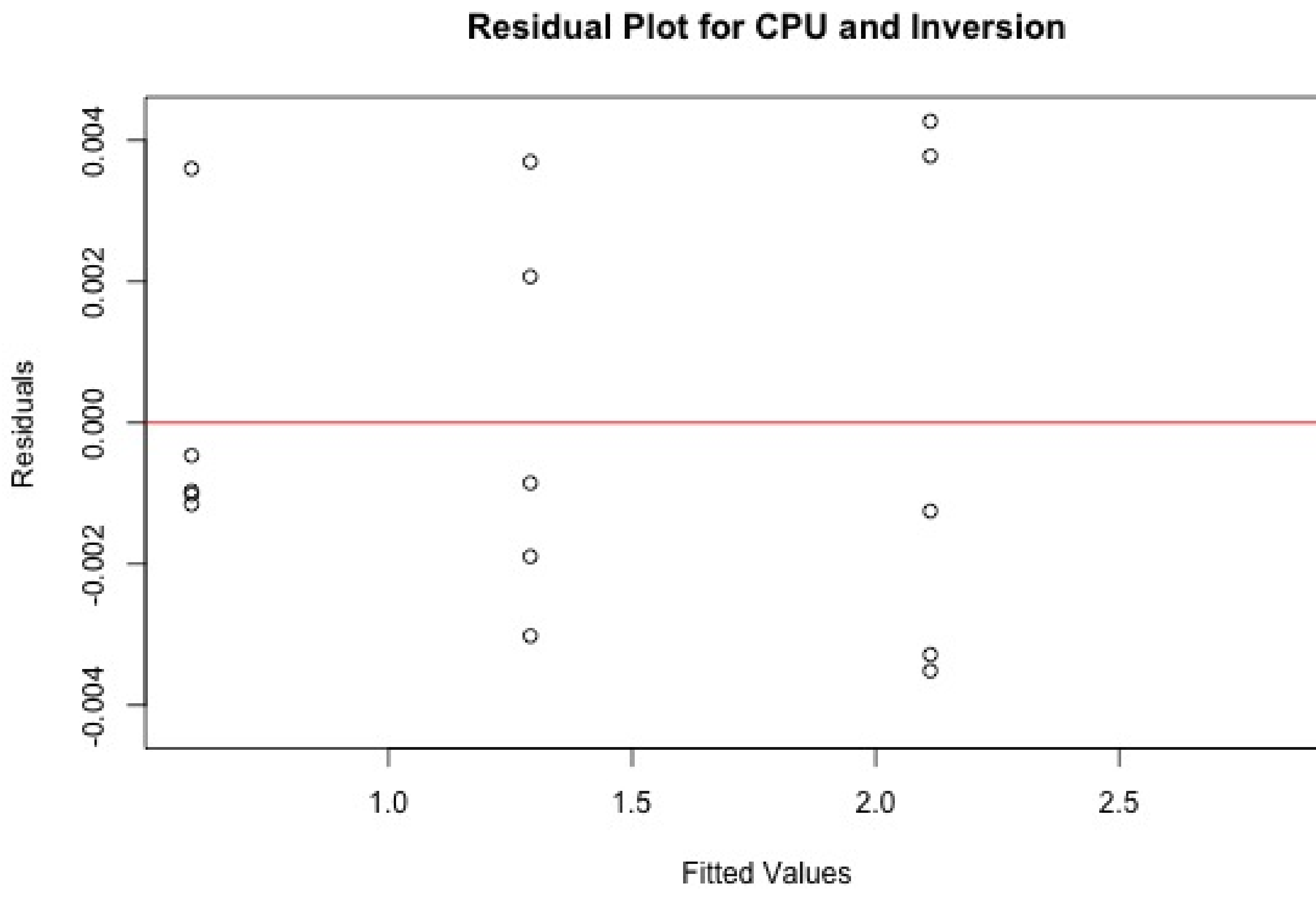
### CPU Multiplication

```
jpeg(filename = "../figs/res_cpu_mult.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(mod_cpu_mult$residuals))
plot(mod_cpu_mult$fitted.values ,mod_cpu_mult$residuals, ylim=c(-m, m), main = "Residual Plot for CPU a
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_cpu_mult.jpeg")
```



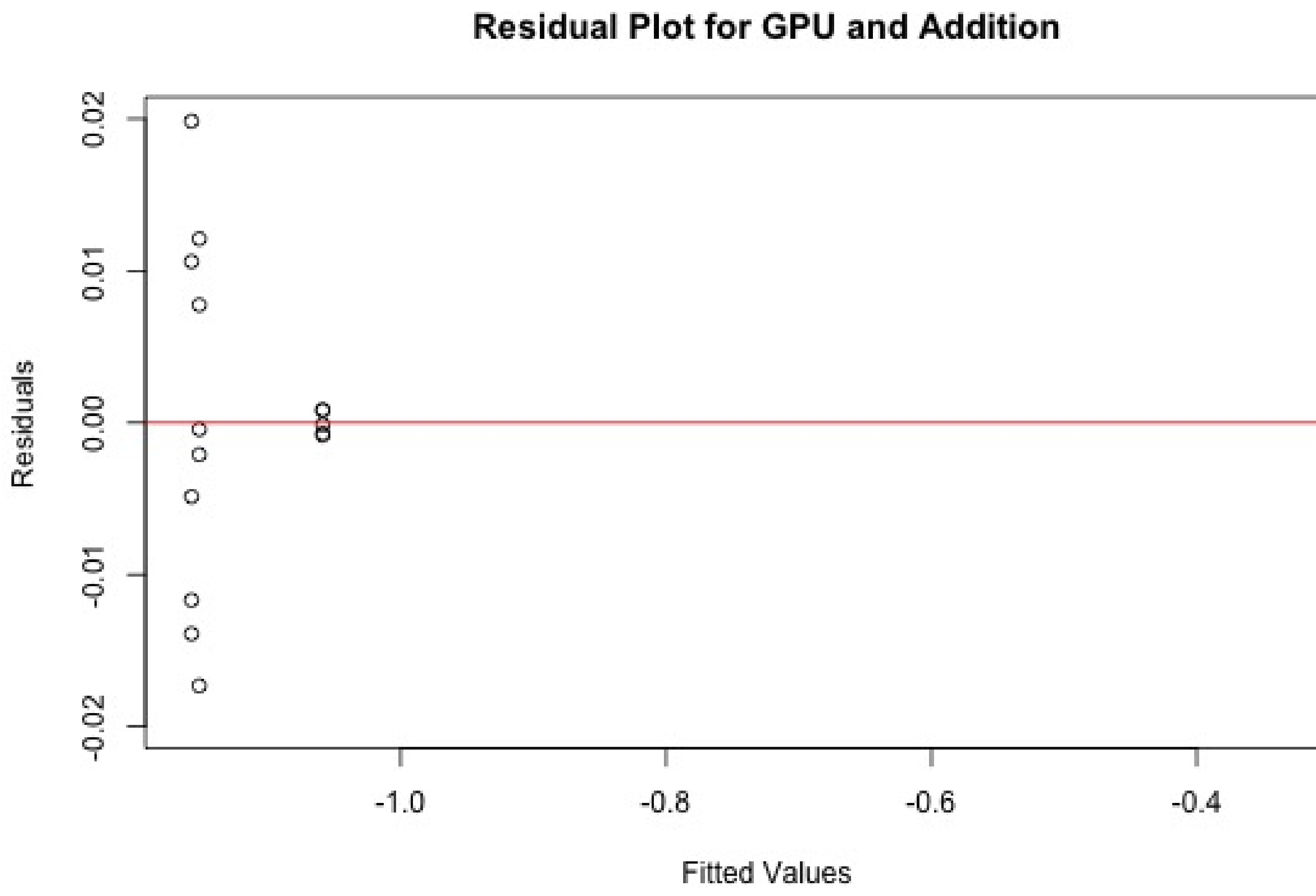
### CPU Inversion

```
jpeg(filename = "../figs/res_cpu_inv.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(mod_cpu_inv$residuals))
plot(mod_cpu_inv$fitted.values,mod_cpu_inv$residuals, ylim=c(-m, m), main = "Residual Plot for CPU and 
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_cpu_inv.jpeg")
```



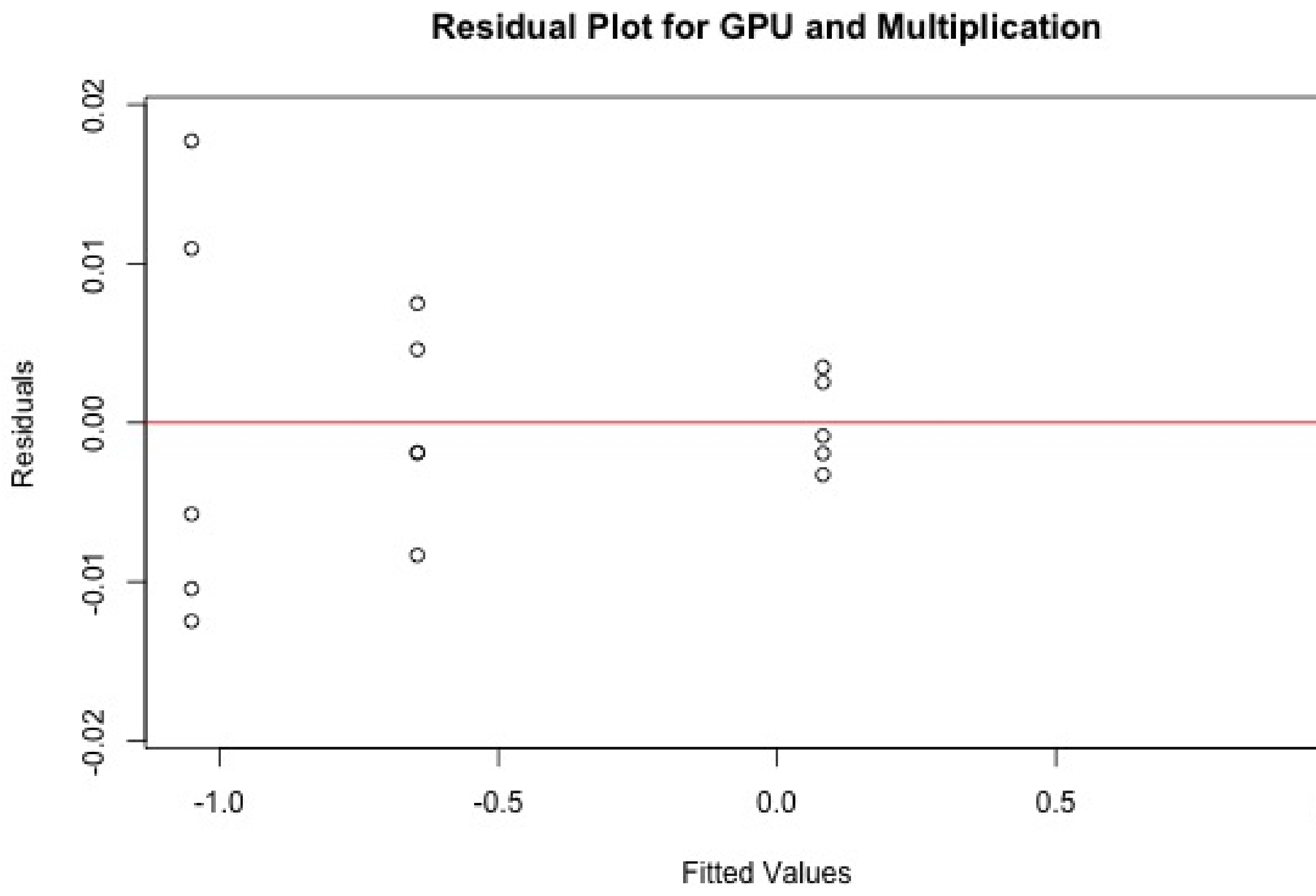
GPU Add

```
jpeg(filename = "../figs/res_gpu_add.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(mod_gpu_add$residuals))
plot(mod_gpu_add$fitted.values,mod_gpu_add$residuals, ylim=c(-m, m), main = "Residual Plot for GPU and 
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_gpu_add.jpeg")
```



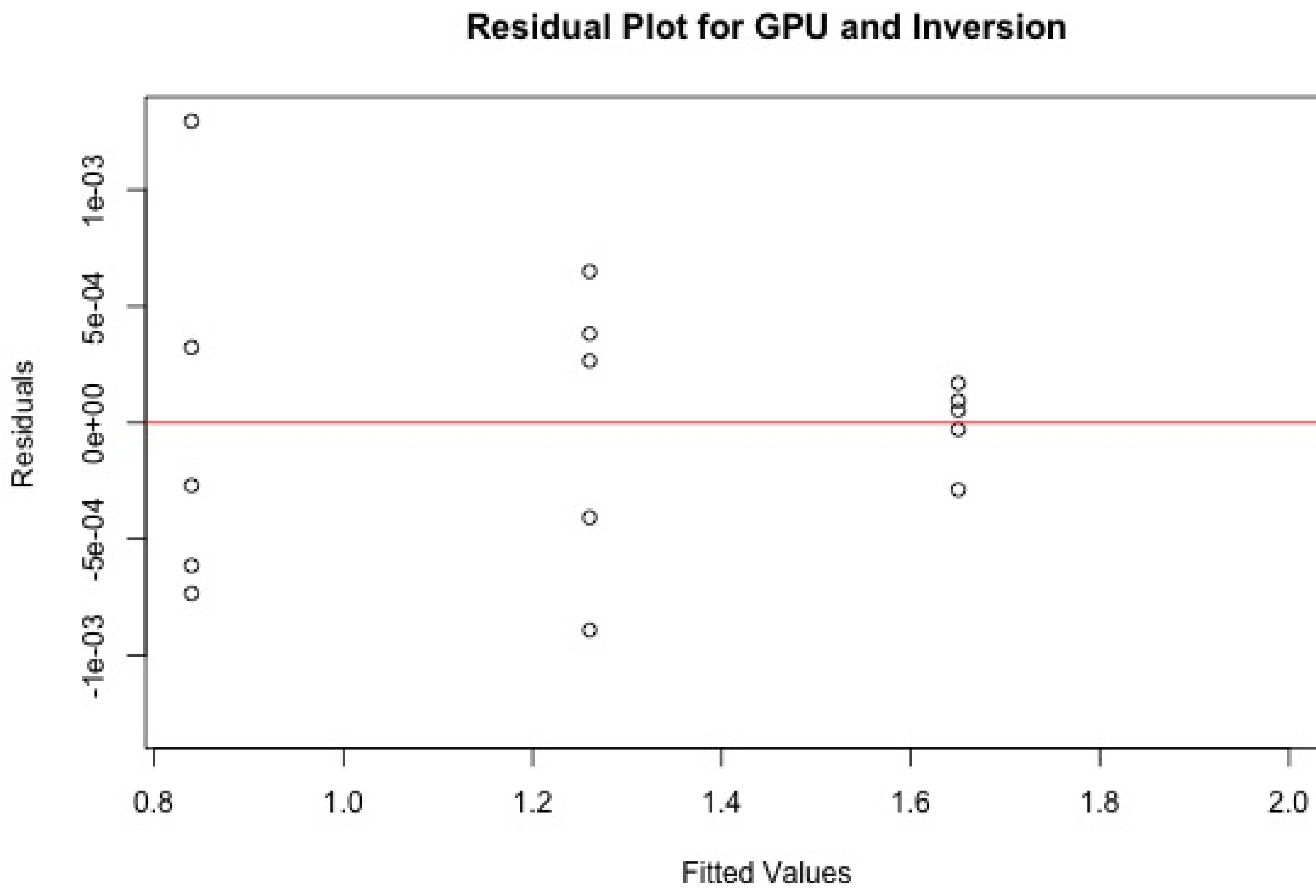
### GPU Multiplication

```
jpeg(filename = "../figs/res_gpu_mult.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(mod_gpu_mult$residuals))
plot(mod_gpu_mult$fitted.values ,mod_gpu_mult$residuals, ylim=c(-m, m), main = "Residual Plot for GPU a
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_gpu_mult.jpeg")
```



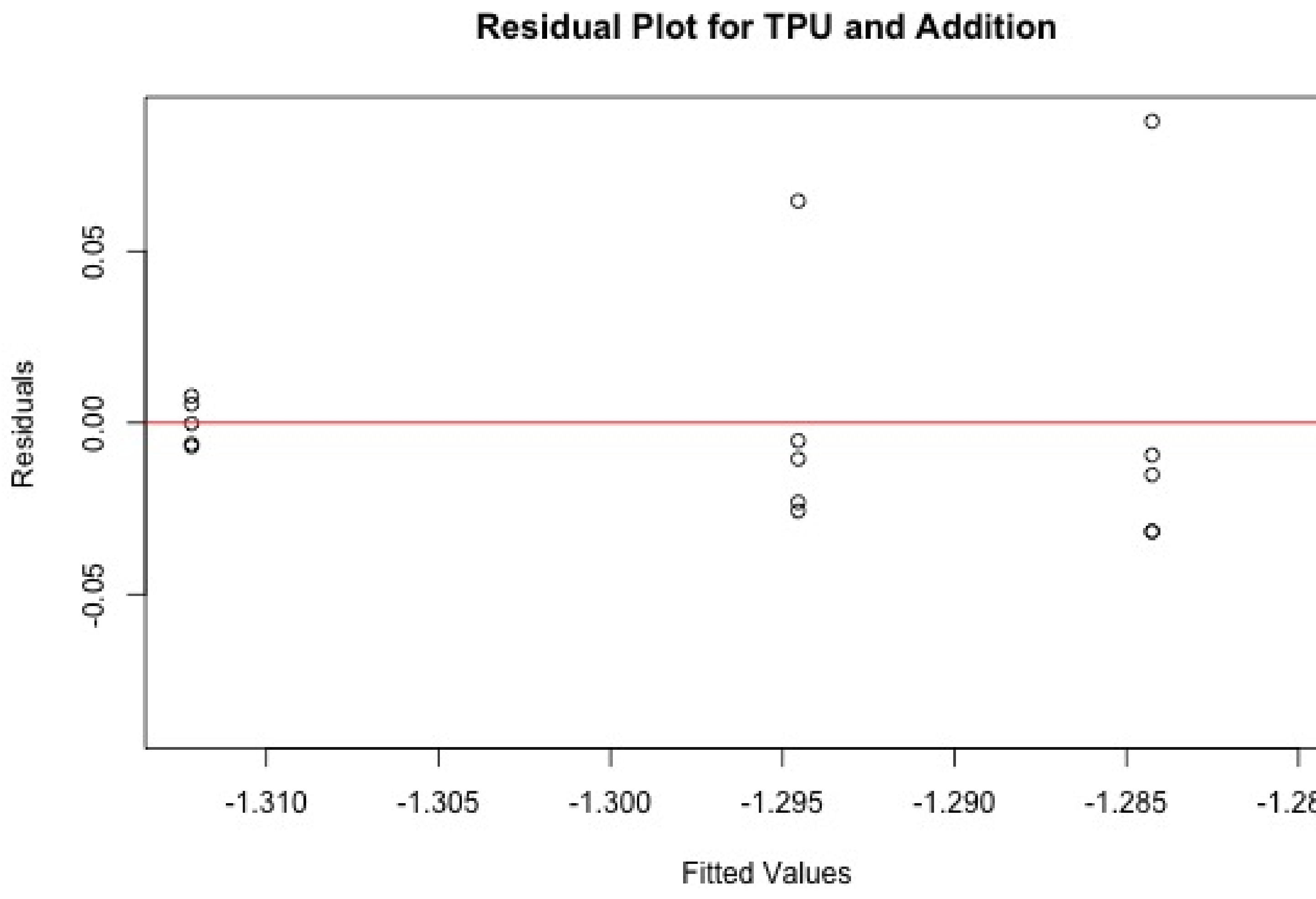
### GPU Inversion

```r
jpeg(filename = "../figs/res_gpu_inv.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(mod_gpu_inv$residuals))
plot(mod_gpu_inv$fitted.values, mod_gpu_inv$residuals, ylim=c(-m, m), main = "Residual Plot for GPU and
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_gpu_inv.jpeg")
```



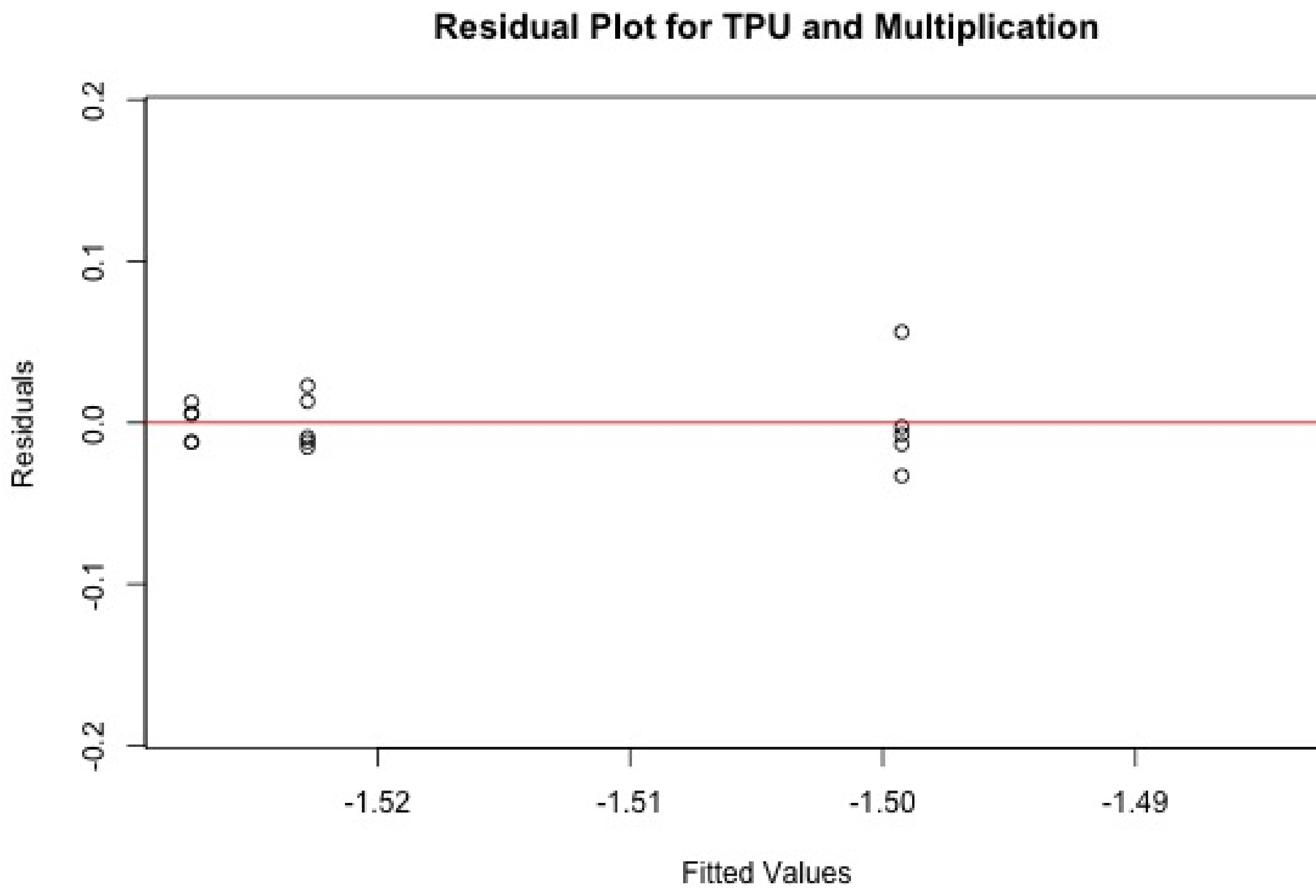
TPU Add

```
jpeg(filename = "../figs/res_tpu_add.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(mod_tpu_add$residuals))
plot(mod_tpu_add$fitted.values, mod_tpu_add$residuals, ylim=c(-m, m), main = "Residual Plot for TPU and
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_tpu_add.jpeg")
```



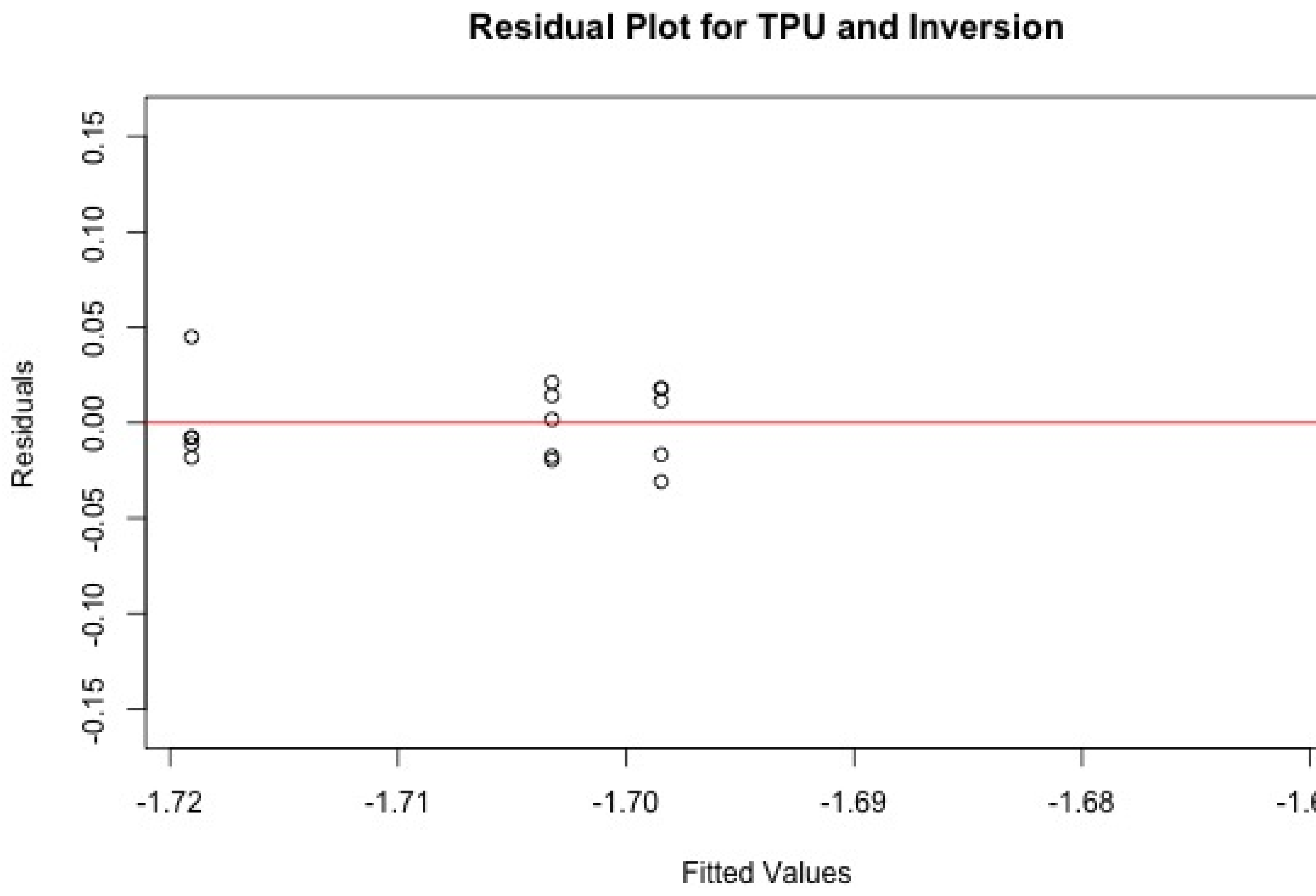
### TPU Multiplication

```
jpeg(filename = "../figs/res_tpu_mult.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(mod_tpu_mult$residuals))
plot(mod_tpu_mult$fitted.values, mod_tpu_mult$residuals, ylim=c(-m, m), main = "Residual Plot for TPU a
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_tpu_mult.jpeg")
```



### TPU Inversion

```
jpeg(filename = "../figs/res_tpu_inv.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(mod_tpu_inv$residuals))
plot(mod_tpu_inv$fitted.values, mod_tpu_inv$residuals, ylim=c(-m, m), main = "Residual Plot for TPU and
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_tpu_inv.jpeg")
```



## Linear Regression for Matrix Size as continuous We just need to run linear regression on seven pairs that do have Matrix Size Effects

```
summary(lm_cpu_add <- lm(log10(Runtime)~as.numeric(MatrixSize),data=data[data$Processor == "CPU" & data
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ as.numeric(MatrixSize), data = data[data$Processor ==
##     "CPU" & data$MatrixOperation == "Addition", ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.35107 -0.18405 -0.13000  0.08052  0.62584
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -1.5277230  0.0646279  -23.64   <2e-16 ***
## as.numeric(MatrixSize)  0.0024138  0.0001237   19.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.3239 on 38 degrees of freedom
## Multiple R-squared:  0.9093, Adjusted R-squared:  0.9069
## F-statistic: 380.9 on 1 and 38 DF,  p-value: < 2.2e-16
```

```
summary(lm_cpu_mult <- lm(log10(Runtime)~as.numeric(MatrixSize),data[data$Processor == "CPU" & data$Mat
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ as.numeric(MatrixSize), data = data[data$Processor ==
##     "CPU" & data$MatrixOperation == "Multiplication", ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6282 -0.5727 -0.1663  0.5620  0.9143
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -0.9187389  0.1195854  -7.683 2.99e-09 ***
## as.numeric(MatrixSize)  0.0033880  0.0002288  14.805  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5993 on 38 degrees of freedom
## Multiple R-squared:  0.8522, Adjusted R-squared:  0.8484
## F-statistic: 219.2 on 1 and 38 DF,  p-value: < 2.2e-16
```

```
summary(lm_cpu_inv <- lm(log10(Runtime)~as.numeric(MatrixSize),data[data$Processor == "CPU" & data$Matr
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ as.numeric(MatrixSize), data = data[data$Processor ==
##     "CPU" & data$MatrixOperation == "Inversion", ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79157 -0.41955 -0.03787  0.47617  0.67656
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -0.324820   0.100347  -3.237  0.00251 **
## as.numeric(MatrixSize)  0.002947   0.000192  15.346  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5029 on 38 degrees of freedom
## Multiple R-squared:  0.8611, Adjusted R-squared:  0.8574
## F-statistic: 235.5 on 1 and 38 DF,  p-value: < 2.2e-16
```

```
summary(lm_gpu_add <- lm(log10(Runtime)~as.numeric(MatrixSize),data=data[data$Processor == "GPU" & data$
```

```
##
```

```
## Call:
## lm(formula = log10(Runtime) ~ as.numeric(MatrixSize), data = data[data$Processor ==
##     "GPU" & data$MatrixOperation == "Addition", ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.22827 -0.05771  0.04545  0.06655  0.14214
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -1.2307302  0.0231517  -53.16   <2e-16 ***
## as.numeric(MatrixSize)  0.0006241  0.0000443   14.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.116 on 38 degrees of freedom
## Multiple R-squared:  0.8393, Adjusted R-squared:  0.835
## F-statistic: 198.4 on 1 and 38 DF,  p-value: < 2.2e-16
```

```
summary(lm_gpu_mult <- lm(log10(Runtime)~as.numeric(MatrixSize),data[data$Processor == "GPU" & data$Mat
```

```
##
## Call:
## lm(formula = log10(Runtime) ~ as.numeric(MatrixSize), data = data[data$Processor ==
##     "GPU" & data$MatrixOperation == "Multiplication", ])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.189553 -0.042371 -0.006256  0.062402  0.136090
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -1.148e+00  1.822e-02  -63.01   <2e-16 ***
## as.numeric(MatrixSize)  1.715e-03  3.487e-05   49.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09131 on 38 degrees of freedom
## Multiple R-squared:  0.9845, Adjusted R-squared:  0.9841
## F-statistic:  2419 on 1 and 38 DF,  p-value: < 2.2e-16
```

```
summary(lm_gpu_inv <- lm(log10(Runtime)~as.numeric(MatrixSize),data[data$Processor == "GPU" & data$Matr
```

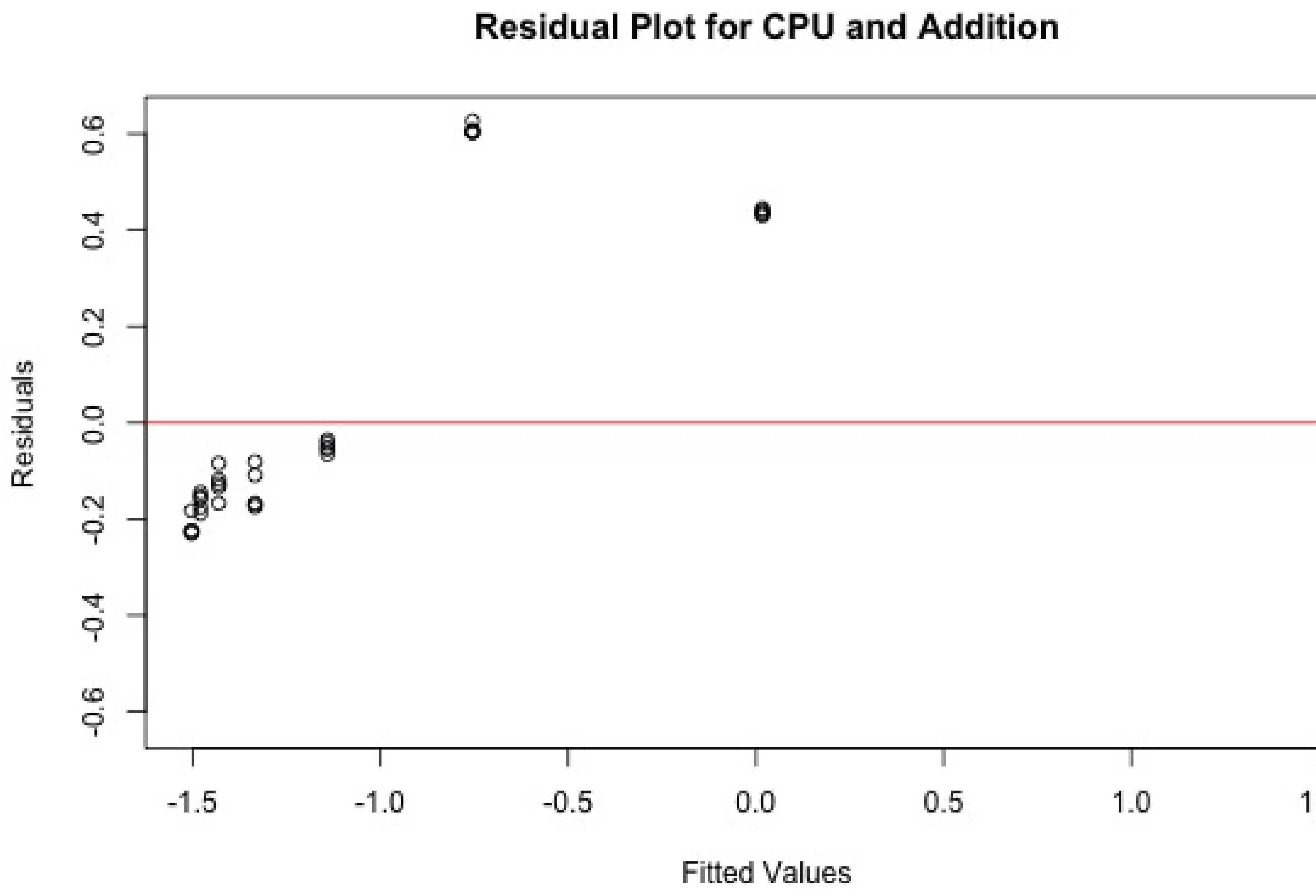
```
##
## Call:
## lm(formula = log10(Runtime) ~ as.numeric(MatrixSize), data = data[data$Processor ==
##     "GPU" & data$MatrixOperation == "Inversion", ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.36329 -0.27442  0.00045  0.24043  0.39749
##
## Coefficients:
```

```
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.3755794  0.0557293   6.739 5.55e-08 ***
## as.numeric(MatrixSize) 0.0015252  0.0001066  14.301  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2793 on 38 degrees of freedom
## Multiple R-squared:  0.8433, Adjusted R-squared:  0.8392
## F-statistic: 204.5 on 1 and 38 DF,  p-value: < 2.2e-16
```

Plot LM

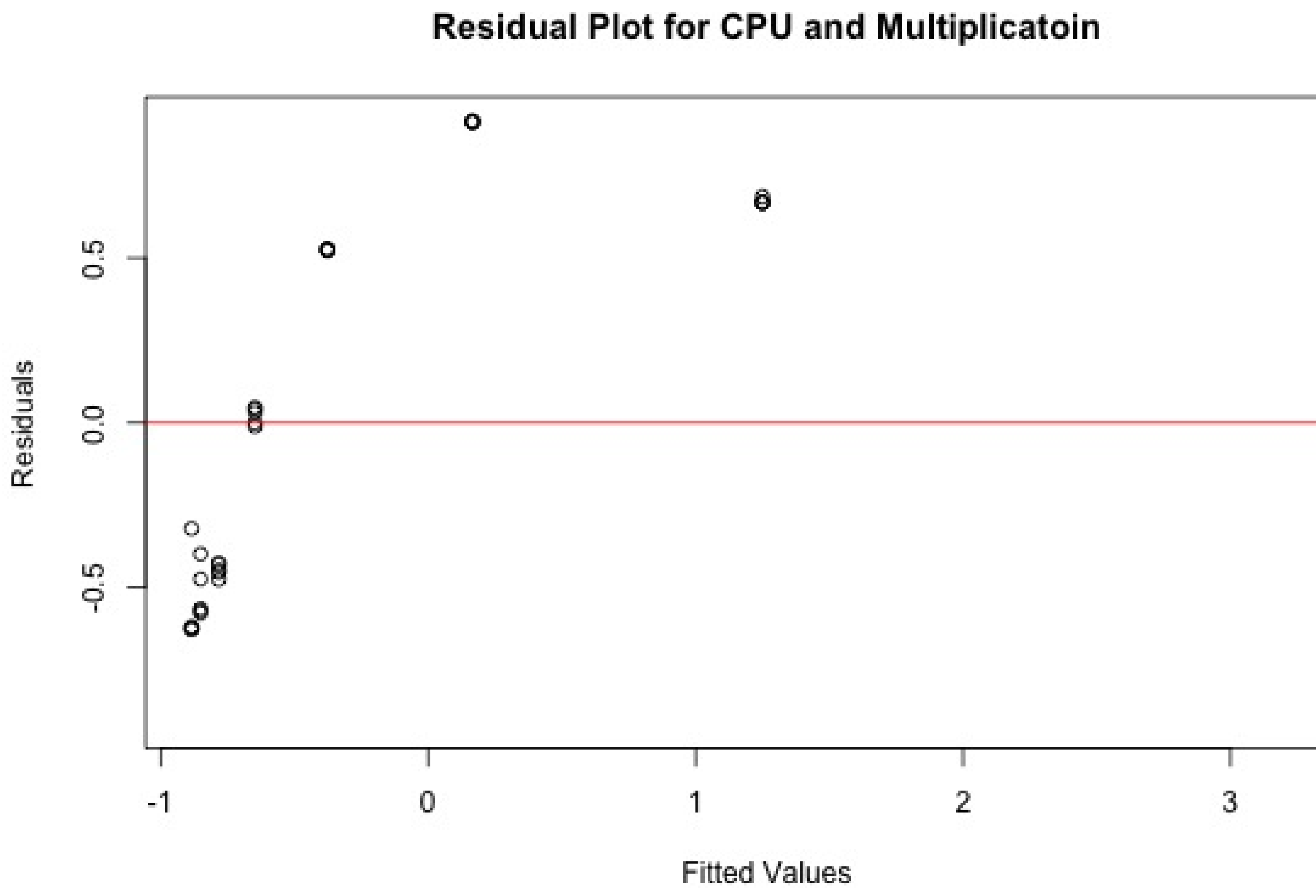
CPU Addition

```
jpeg(filename = "../figs/res_lm_cpu_add.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(lm_cpu_add$residuals))
plot(lm_cpu_add$fitted.values, lm_cpu_add$residuals, ylim=c(-m, m), main = "Residual Plot for CPU and Ac
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_lm_cpu_add.jpeg")
```



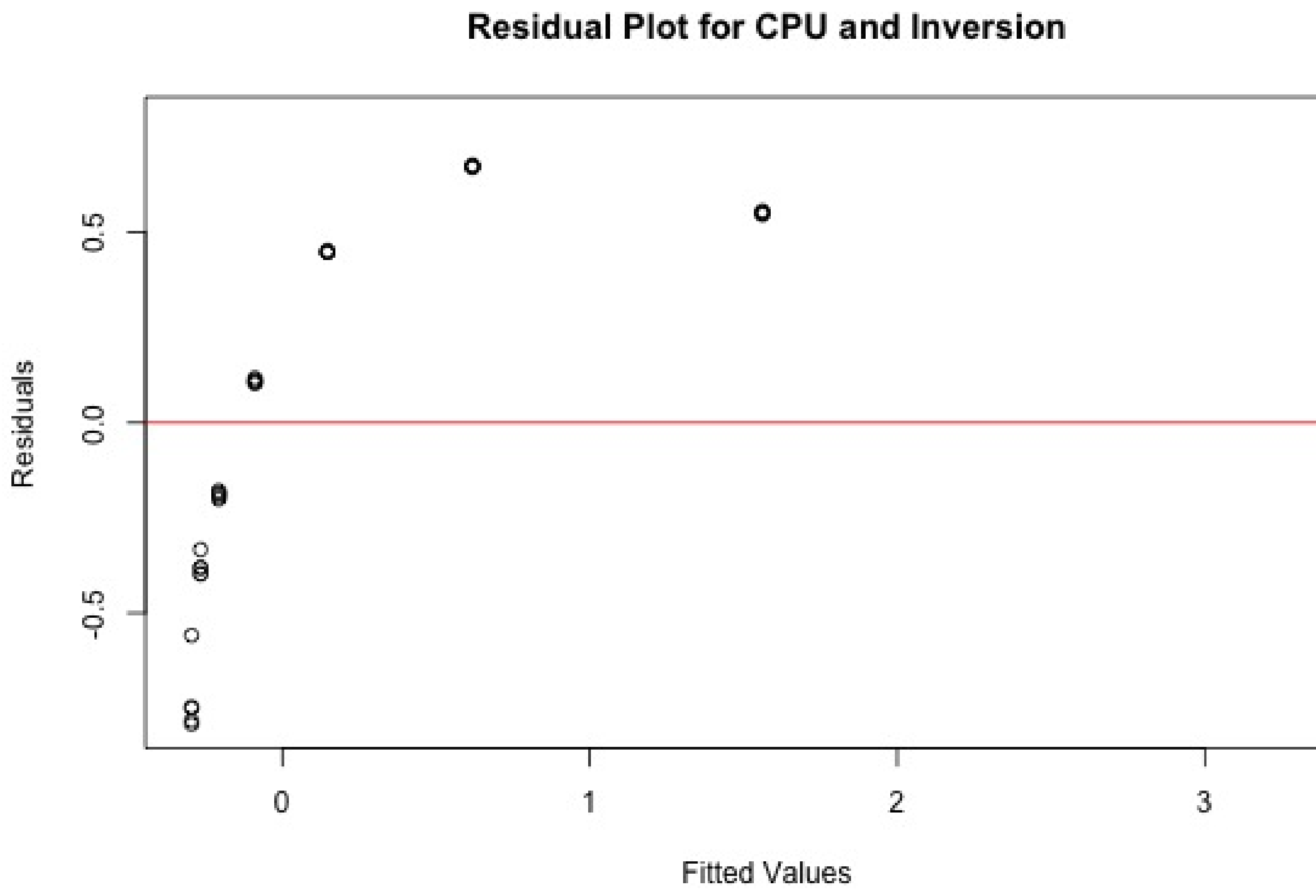
### CPU Multiplication

```
jpeg(filename = "../figs/res_lm_cpu_mult.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(lm_cpu_mult$residuals))
plot(lm_cpu_mult$fitted.values, lm_cpu_mult$residuals, ylim=c(-m, m), main = "Residual Plot for CPU and
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_lm_cpu_mult.jpeg")
```



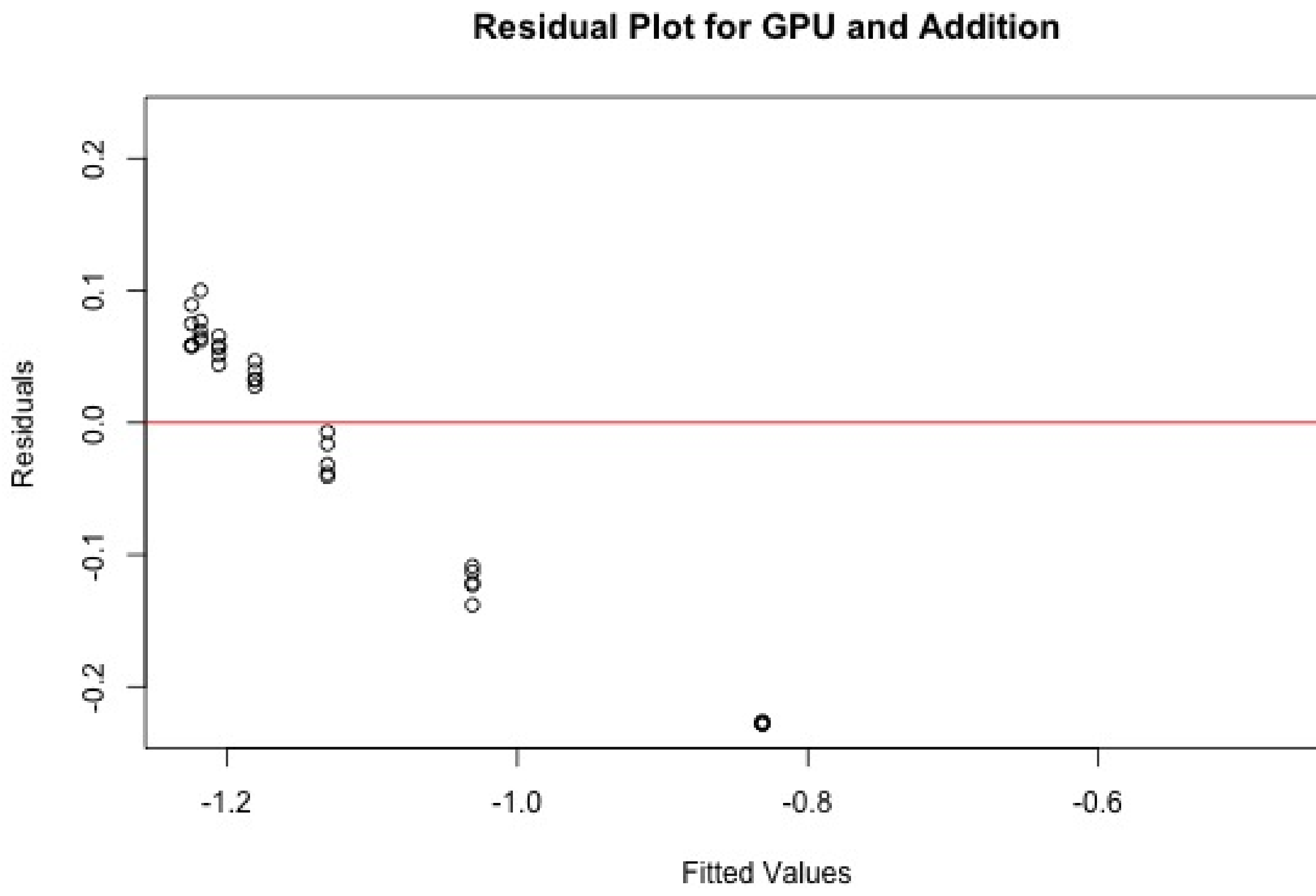
CPU Inversion

```
jpeg(filename = "../figs/res_lm_cpu_inv.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(lm_cpu_inv$residuals))
plot(lm_cpu_inv$fitted.values, lm_cpu_inv$residuals, ylim=c(-m, m), main = "Residual Plot for CPU and I
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_lm_cpu_inv.jpeg")
```



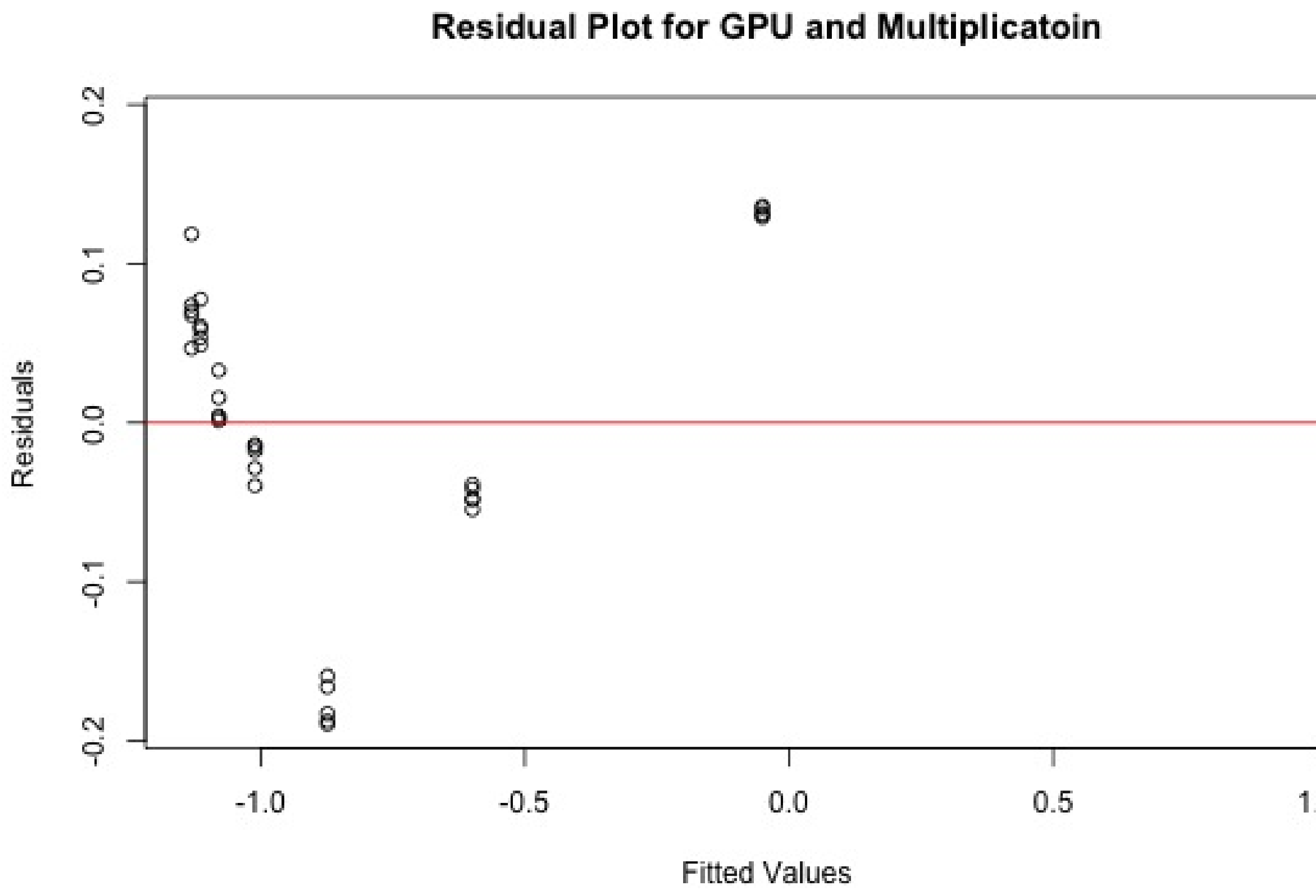
GPU Addition

```
jpeg(filename = "../figs/res_lm_gpu_add.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(lm_gpu_add$residuals))
plot(lm_gpu_add$fitted.values, lm_gpu_add$residuals, ylim=c(-m, m), main = "Residual Plot for GPU and Ac
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_lm_gpu_add.jpeg")
```



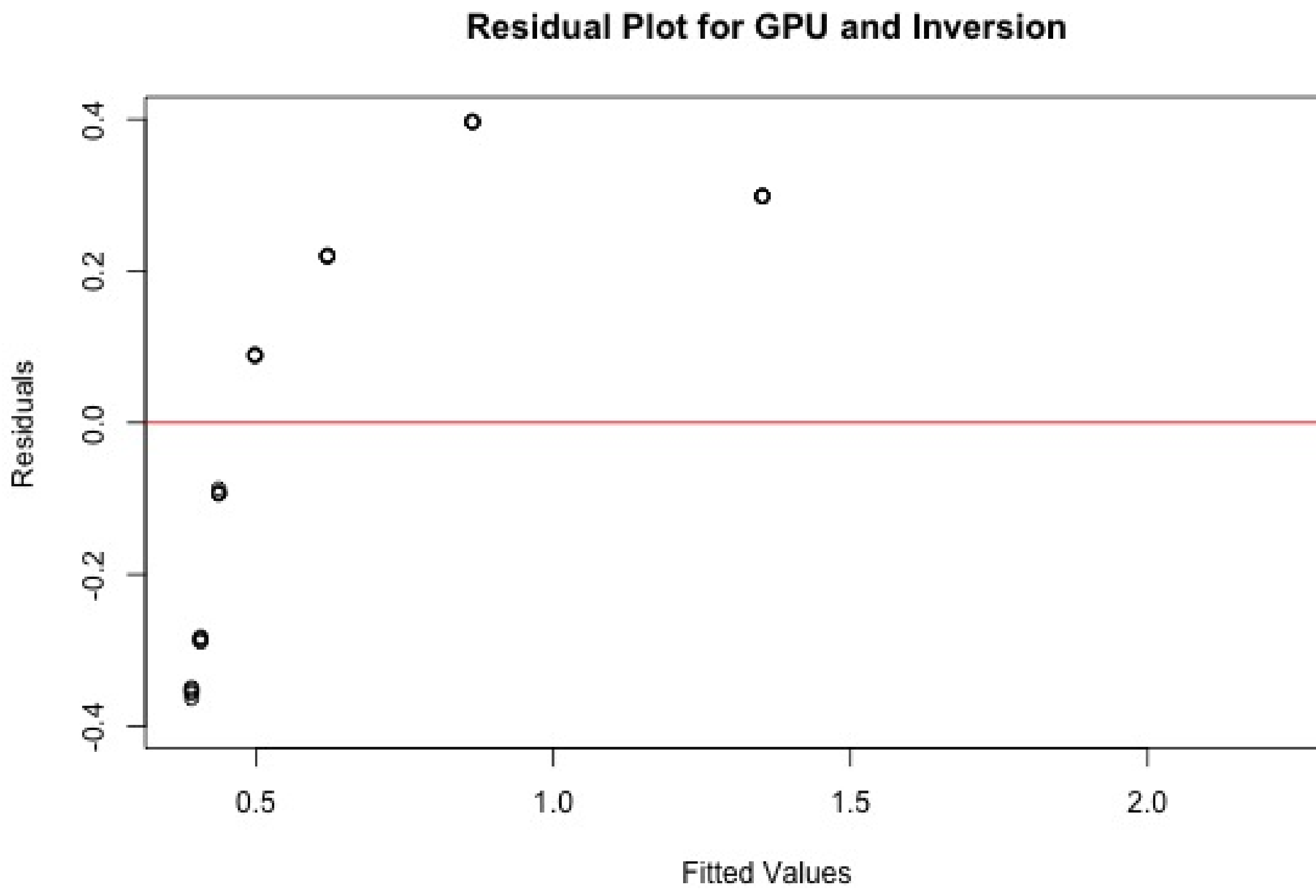
### GPU Multiplication

```
jpeg(filename = "../figs/res_lm_gpu_mult.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(lm_gpu_mult$residuals))
plot(lm_gpu_mult$fitted.values, lm_gpu_mult$residuals, ylim=c(-m, m), main = "Residual Plot for GPU and
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_lm_gpu_mult.jpeg")
```

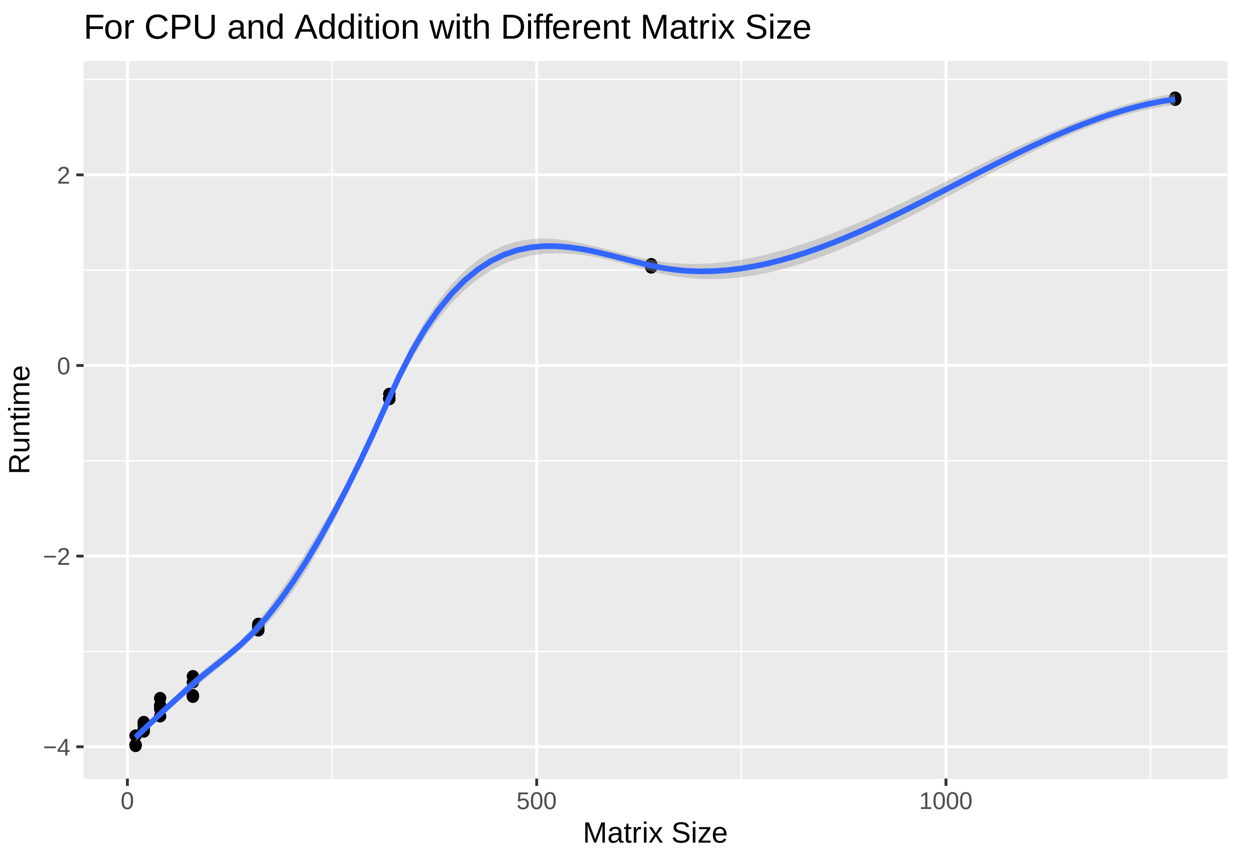


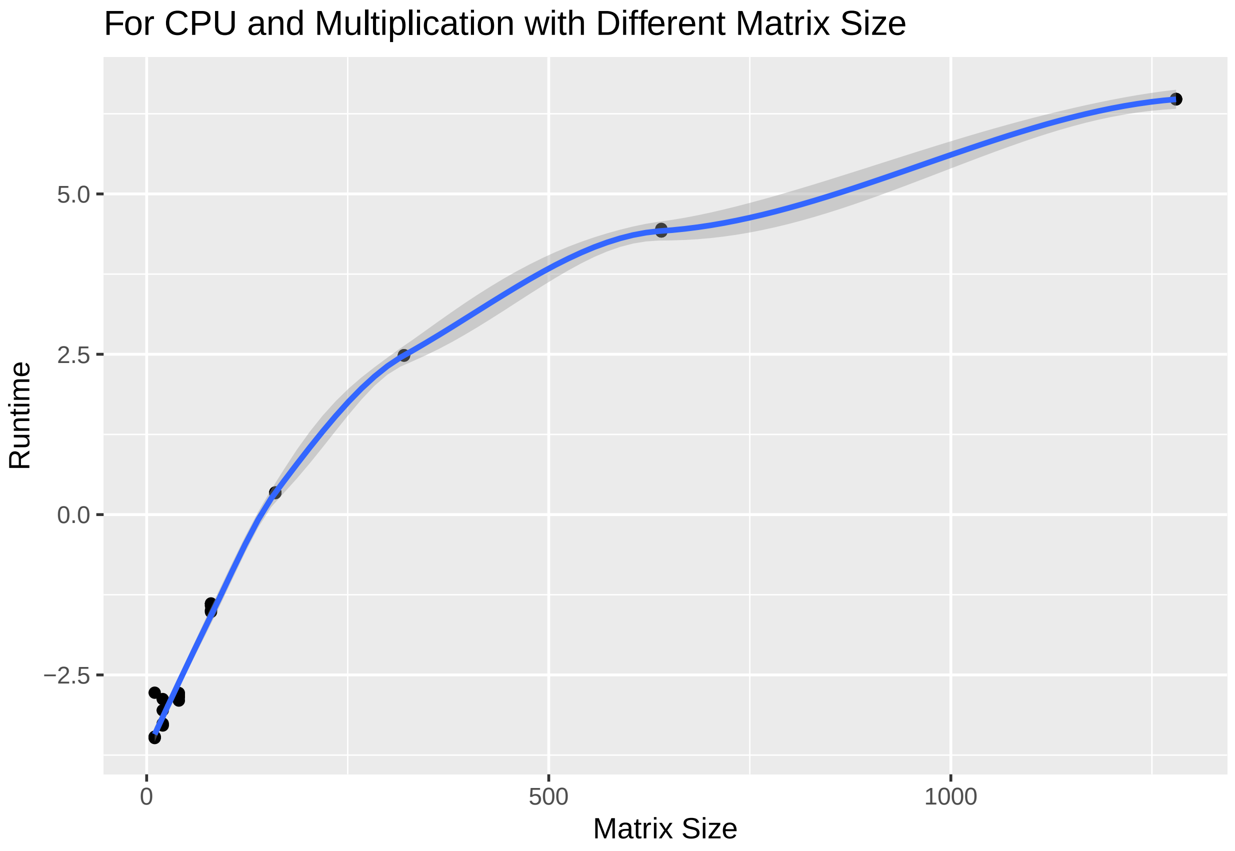
GPU Inversion

```
jpeg(filename = "../figs/res_lm_gpu_inv.jpeg", width = 600, height = 400,quality = 10000)
m <- max(abs(lm_gpu_inv$residuals))
plot(lm_gpu_inv$fitted.values, lm_gpu_inv$residuals, ylim=c(-m, m), main = "Residual Plot for GPU and In
abline(h=0, col="red")
while (!is.null(dev.list()))  dev.off()
knitr::include_graphics("../figs/res_lm_gpu_inv.jpeg")
```

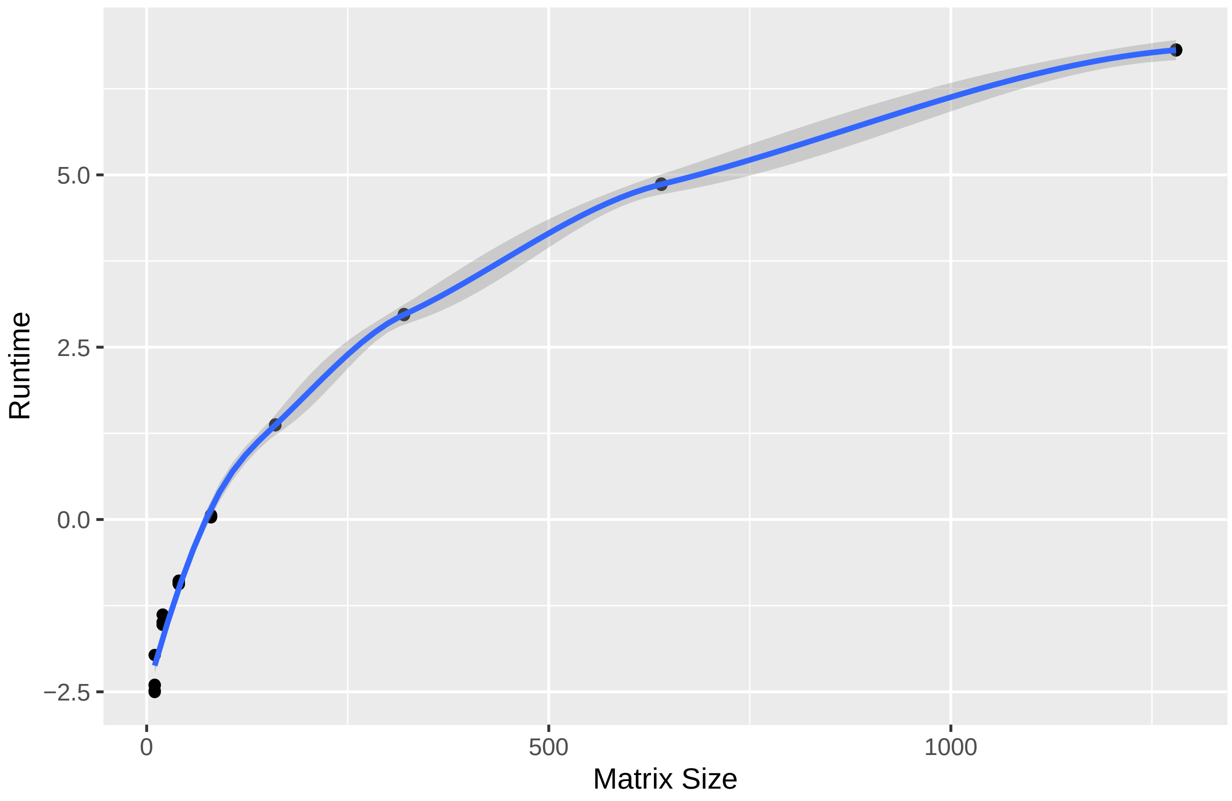


At the level of matrix size=1280, avoid using CPU for inversion and multiplication because its run-times are much bigger.

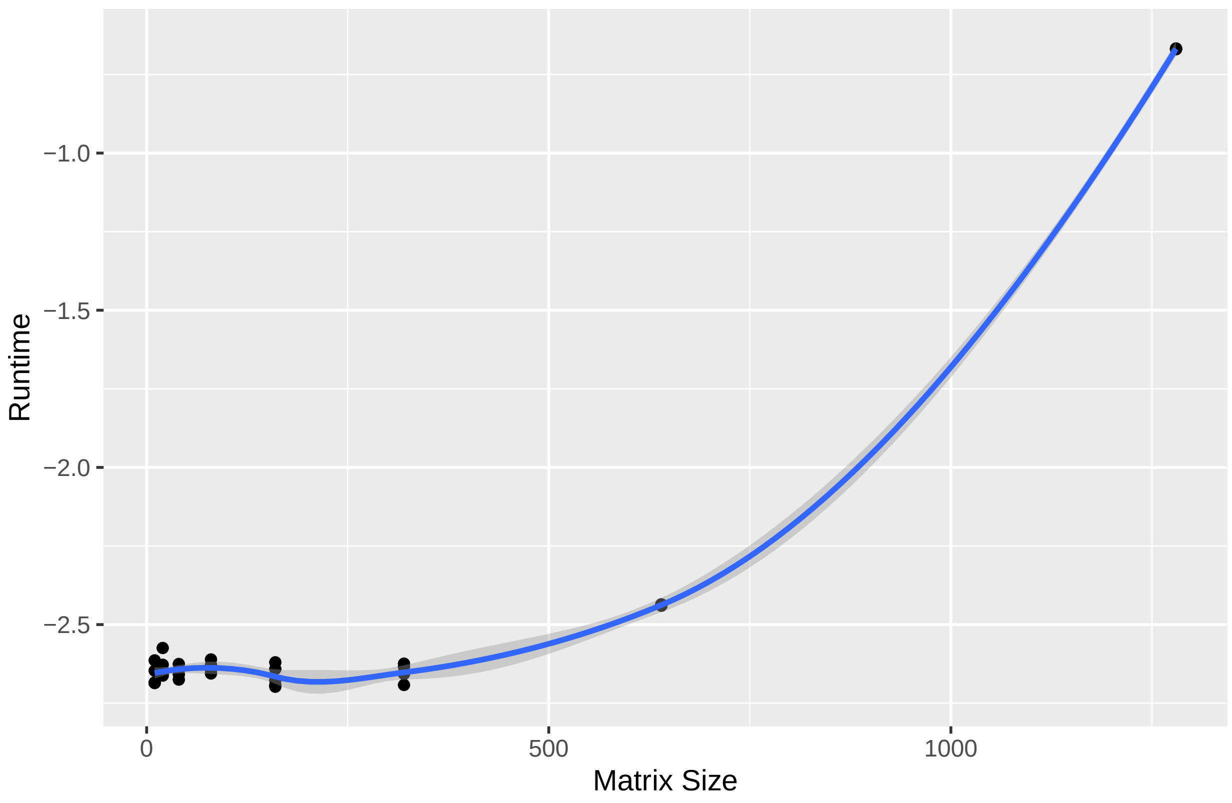

For CPU and Addition with Different Matrix Size
Runtime
2
0
−2
−4
0
500
1000
Matrix Size


For CPU and Multiplication with Different Matrix Size
Runtime
5.0
2.5
0.0
−2.5
0
500
1000
Matrix Size

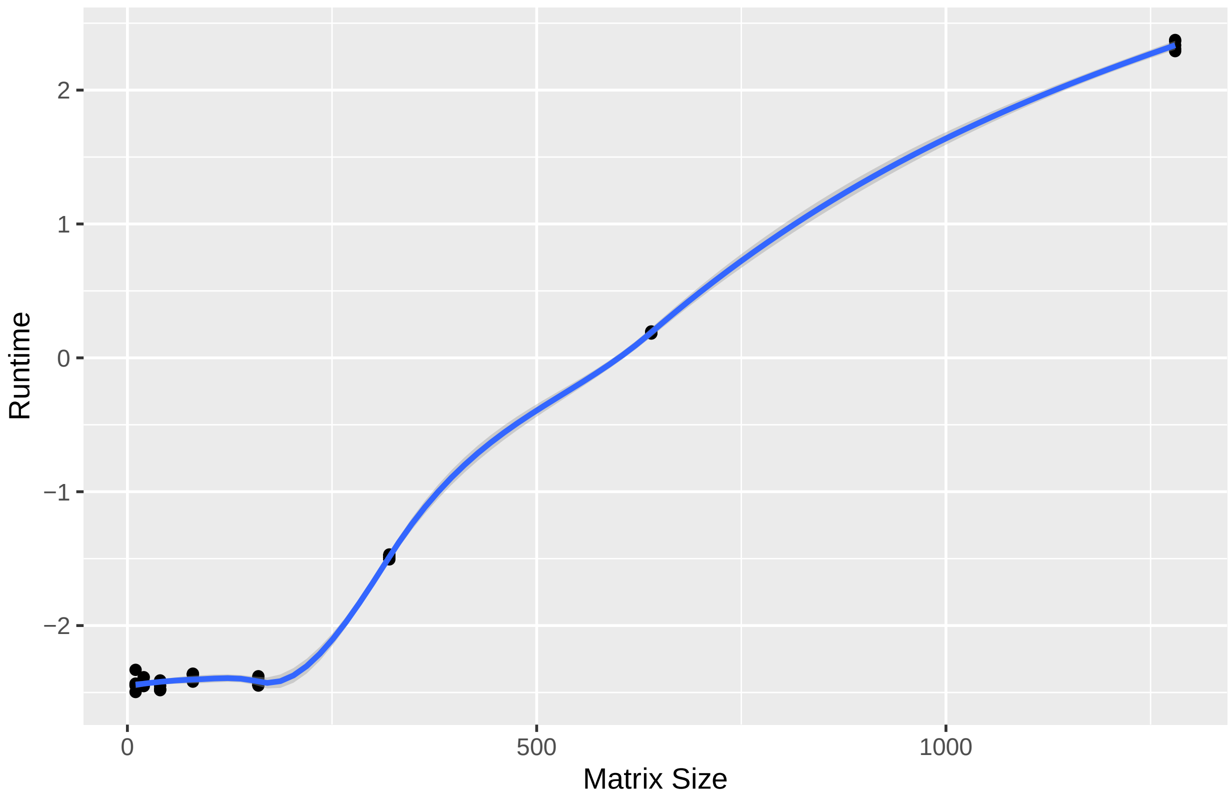
For CPU and Invertion with Different Matrix Size



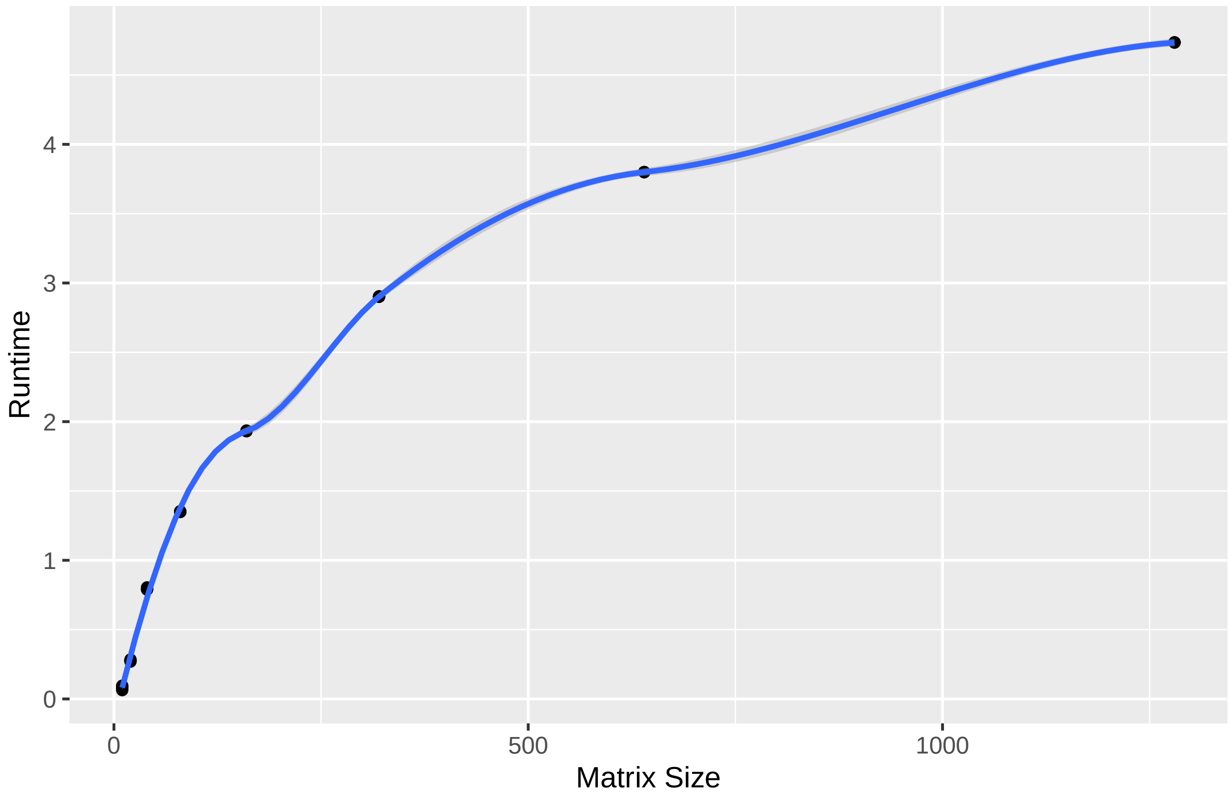
For GPU and Addition with Different Matrix Size

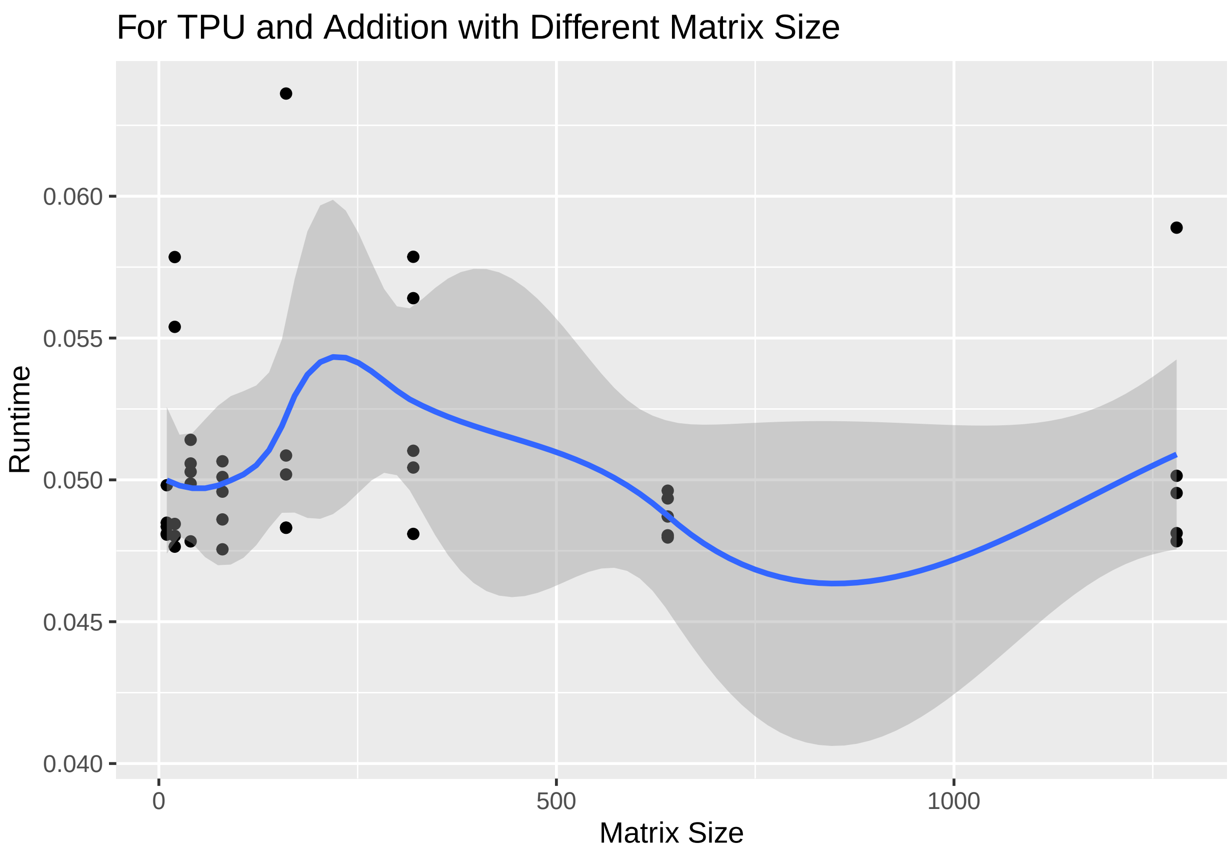


For GPU and Multiplication with Different Matrix Size

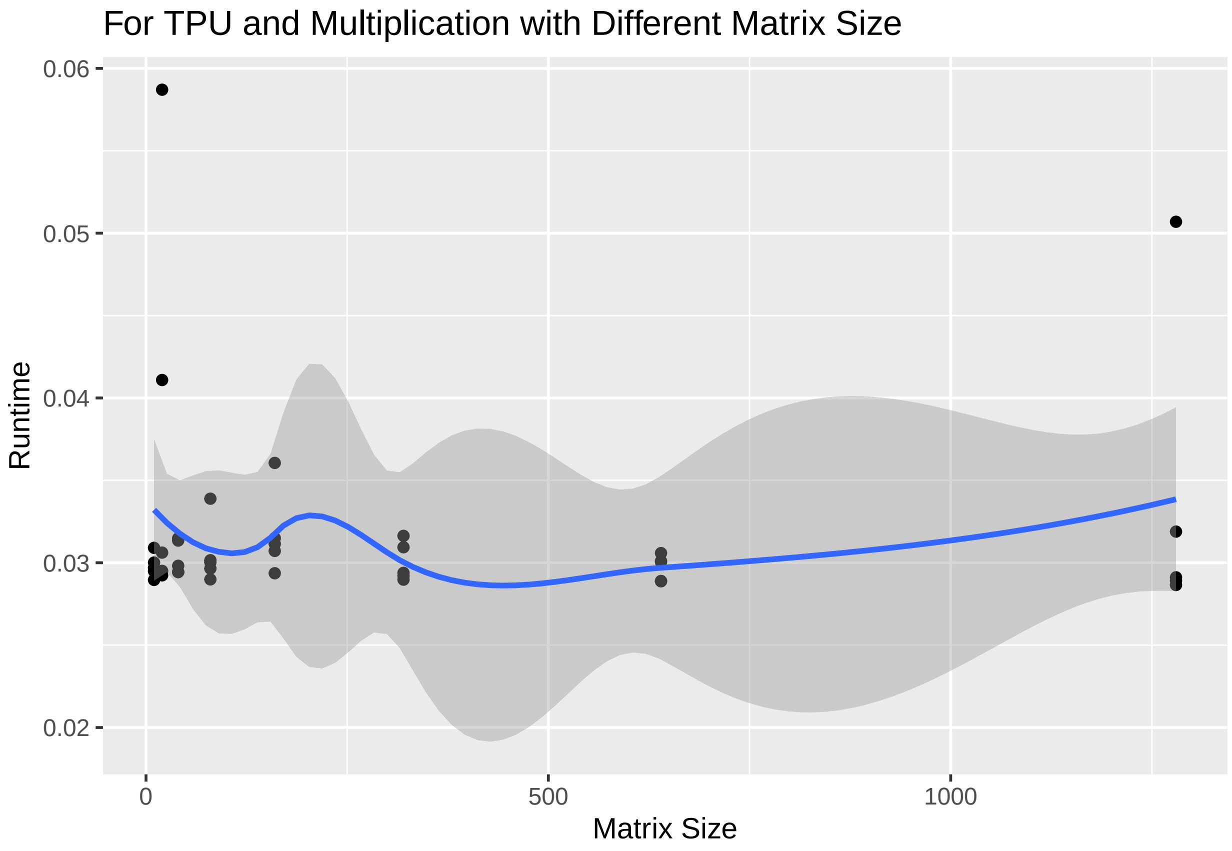


For GPU and Invertion with Different Matrix Size



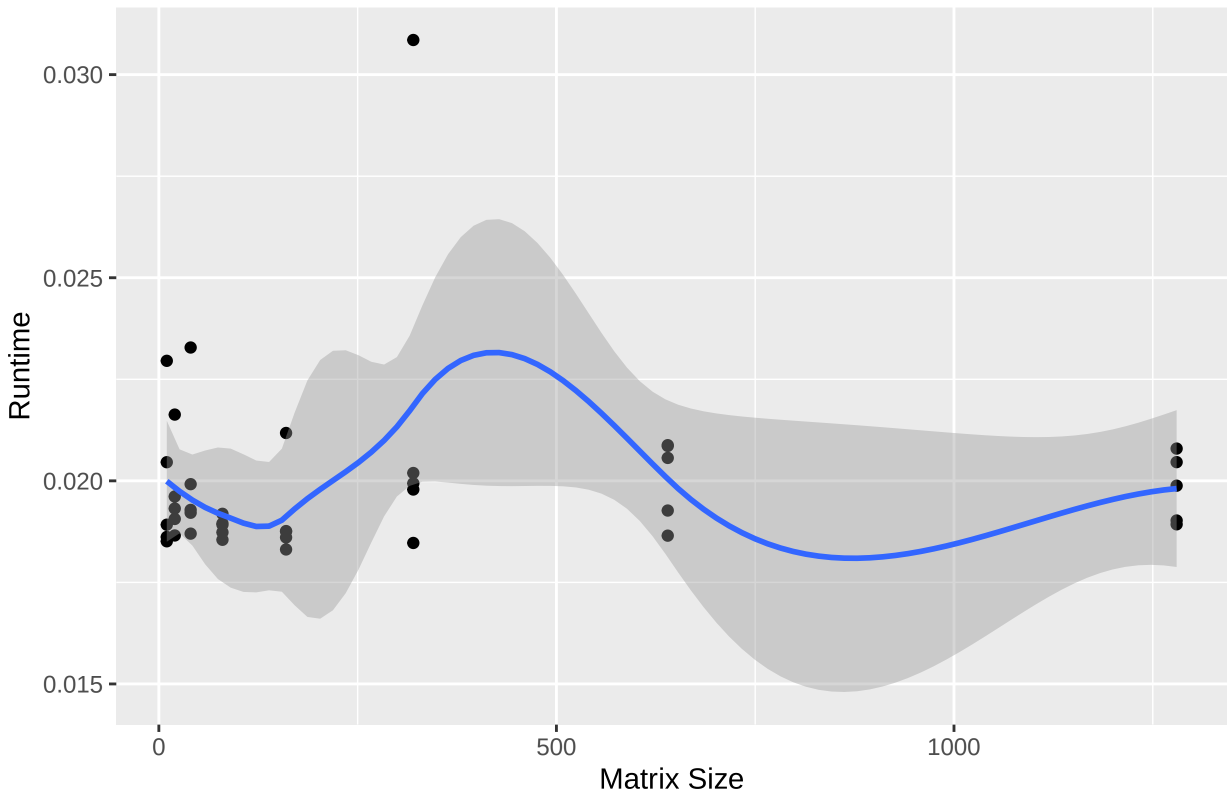


For TPU and Addition with Different Matrix Size
0.060
0.055
0.050
0.045
0.040
Runtime
0
500
1000
Matrix Size



For TPU and Multiplication with Different Matrix Size
0.06
0.05
0.04
0.03
0.02
Runtime
0
500
1000
Matrix Size

For TPU and Invertion with Different Matrix Size


Runtime
Matrix Size