

Diseño de Experimentos

Tarea 2

Andrey Arguedas Espinoza - 2020426569

Abstract—Resolución de la tarea 2 del curso de Diseño de Experimentos - Tecnológico de Costa Rica.

Index Terms—Experimentos, Datos, Gráficos, Histogramas, Modelos, Anova, Residuos, Leveane, R.

I. INTRODUCCIÓN

Un científico de la computación, apasionado por el área de gráficos por computadora, se propone crear un algoritmo de procesamiento híbrido (hace uso del CPU y GPU integrado) para mejorar el desempeño en computadoras que no cuentan con GPU discreto. El algoritmo de sintetizado de imágenes por computadora que utilizó hace uso de las propiedades físicas de la luz para generar la imagen, por lo que clásicamente se han utilizado CPUs cuando se hace uso de este algoritmo, sin embargo, en investigaciones recientes se han utilizado GPUs modernos para sintetizar esta clase de imágenes.

Dado que el objetivo de su investigación es mejorar el rendimiento del sintetizado de imágenes en sistemas que únicamente cuentan con GPU integrado (conocidos en la industria como SoC o APUs), la variable de respuesta de su experimento es tiempo de sintetizado. Con la finalidad de hacer la comparación de la manera más justa posible, utilizó un CPU, APU y GPU del mismo rango de precios. El algoritmo híbrido corre únicamente en el APU.

Varios factores impactan el tiempo de sintetizado: la cantidad de objetos presentes en la escena (la escena en sí es definida por este factor), los efectos visuales de la escena: anti-aliasing (AA), reflexiones (RE) y transparencias (TR). De igual manera, decidió sintetizar la misma imagen en tres resoluciones diferentes: 1280x720, 1440x900 y 1920x1080. Por último, la misma escena, con los mismos escenarios se ejecutaron en las tres diferentes arquitecturas (CPU, GPU y APU).

El razonamiento del científico para incluir tantos factores en su experimento es que las imágenes sintetizadas por computadora son muy diversas por naturaleza, y quería estar completamente seguro de que su propuesta de algoritmo tenía la capacidad de brindar mejor rendimiento en distintos escenarios, y no convenientemente en uno específico.

El científico automatizó la ejecución de los experimentos, garantizando aleatoriedad, y que cada escenario específico corriera un número repetido de veces (de las cuales no se acuerda).

Peor aún, los datos que el científico recabó no están listos para ingresarse a R, sino que se necesita de un preprocesamiento.

Cada medición tomada cuenta de cuatro líneas, como las que se muestran a continuación:

APU image written, Rays: 2758193

real 6,53

user 50,25

sys 0,33

Cada medición tomada cuenta de cuatro líneas, como las que se muestran a continuación:

APU image written, Rays: 2758193

real 6,53

user 50,25

sys 0,33

10167_16000APU010-1280x720.sc

Las líneas en rojo no sirven para el análisis que se desarrolla en este documento y se pueden eliminar. La línea que dice "real" contiene el tiempo de sintetizado de la imagen (nuestra variable de respuesta).

De la última línea, la información relevante es la que se encuentra después del _.

La nomenclatura es _(cantidad de objetos)(arquitectura)(efectos en la escena)-(resolución de la escena).

Los efectos de la escena tienen una codificación binaria, donde 1 significa presente y 0 no presente. El orden es (anti-aliasing)(transparencias)(reflexiones). 000 significa no hay ningún efecto y 111 que todos los efectos están presentes.

II. PROCEDIMIENTO

1) Inicialización:

a) Inclusión de paquetes y librerías a utilizar:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(car)){install.packages("car")}
if(!require(multcompview)){install.packages("multcompview")}
if(!require(multcomp)){install.packages("multcomp")}
if(!require(lsmmeans)){install.packages("lsmmeans")}
if(!require(phia)){install.packages("phia")}
if(!require(stringr)){install.packages("stringr")}

library(rcompanion)
library(ggplot2)
library(car)
library(repr)
library(dplyr)
library(stringr)
library(data.table)
```

b) Carga de los datos desde el archivo de text:

```
# Directorio donde se encuentra el archivo
setwd(this.path:here())

# Leemos los datos sin pre-procesar
original_data <- readlines("Datos_tarea_2.txt")

filtered_data = original_data
```

c) Datos originales cargados:

```
[1] "APU image written, Rays: 2758193" "real 6,53"
[3] "user 50,25" "sys 0,33"
[5] "10167_16000APU010-1280x720.sc" "APU image written, Rays: 2048122"
[7] "real 4,22" "user 30,90"
[9] "sys 0,11" "10184_1000APU011-1440x900.sc"
[11] "APU image written, Rays: 6708090" "real 6,13"
[13] "user 46,22" "sys 0,17"
[15] "10204_1000APU101-1280x720.sc" "APU image written, Rays: 19085495"
[17] "real 21,75" "user 171,67"
[19] "sys 0,19" "10210_16000APU110-1440x900.sc"
[21] "APU image written, Rays: 1765644" "real 4,47"
[23] "user 33,40" "sys 0,13"
[25] "10216_4000APU001-1280x720.sc" "APU image written, Rays: 2847385"
[27] "real 19,76" "user 147,53"
[29] "sys 0,14" "10368_260000APU001-1280x720.sc"
[31] "APU image written, Rays: 21486720" "real 76,95"
[33] "user 603,60" "sys 0,21"
[35] "10471_260000APU110-1440x900.sc" "APU image written, Rays: 1834065"
[37] "real 5,93" "user 44,36"
[39] "sys 0,12" "1048_65000APU000-1280x720.sc"
[41] "APU image written, Rays: 10240564" "real 8,43"
[43] "user 63,26" "sys 0,27"
[45] "10731_1000APU111-1440x900.sc" "APU image written, Rays: 18824750"
```

2) Limpieza de datos:

a) Eliminando datos que no serán utilizados:

```
#Eliminando datos no necesarios
filtered_data = str_replace_all(filtered_data, "user.*$", "")
filtered_data = str_replace_all(filtered_data, "sys.*$", "")
filtered_data = str_replace_all(filtered_data, "real ", "")
filtered_data = str_replace_all(filtered_data, regex("(?=(image written).*$"), "")
filtered_data = str_remove_all(filtered_data, regex("(?=_)"))

#Eliminando todos las líneas que quedaron en blanco
empty_lines = grepl('\s*$', filtered_data)
filtered_data[! empty_lines]

filtered_data
```

```
[1] "APU image written, Rays: 2758193" "real 6,53"
[3] "user 50,25" "sys 0,33"
[5] "10167_16000APU010-1280x720.sc" "APU image written, Rays: 2048122"
[7] "real 4,22" "user 30,90"
[9] "sys 0,11" "10184_1000APU011-1440x900.sc"
[11] "APU image written, Rays: 6708090" "real 6,13"
[13] "user 46,22" "sys 0,17"
[15] "10204_1000APU101-1280x720.sc" "APU image written, Rays: 19085495"
[17] "real 21,75" "user 171,67"
```

Notar que ahora tenemos solo los datos necesarios pero todavía necesitan separación.

3) Limpieza y separación de los datos necesarios mediante la utilización de expresiones regulares:

```
#Datos filtrados para obtener solo el tiempo de ejecucion
only_time_data = str_remove_all(filtered_data, regex("(?=_).*"))
only_time_data = str_replace_all(only_time_data, ",", ".")
#Eliminando todos las líneas que quedaron en blanco
empty_lines = grepl('\s*$', only_time_data)
only_time_data = only_time_data[! empty_lines]

only_time_data

#Datos filtrados para obtener solo la columna de mas informacion
only_technical_data = str_remove_all(filtered_data, regex("^A[0-9].*(?=).*"))
#Eliminando todos las líneas que quedaron en blanco
empty_lines = grepl('\s*$', only_technical_data)
only_technical_data = only_technical_data[! empty_lines]

only_technical_data

#Convirtiendo las líneas de tipo
#(cantidad de objetos)(arquitectura)(efectos en la escena)-(resolución de la escena)
#en datos separados

#Obteniendo la resolución
resolucion = str_remove_all(only_technical_data, regex("(?=-).*"))
resolucion = str_replace_all(resolucion, "-", ".")

#Obteniendo los efectos de la escena
efectos = str_remove_all(only_technical_data, regex("(?=-).*"))
efectos = str_remove_all(efectos, regex("(?<=GPU).*"))
efectos = str_remove_all(efectos, regex("(?<=APU).*"))
efectos = str_remove_all(efectos, regex("(?<=CPU).*"))


#Obteniendo los ARQUITECTURA de la escena
arquitectura = str_remove_all(only_technical_data, regex("(?<=GPU).*"))
arquitectura = str_remove_all(arquitectura, regex("(?<=GPU).*"))
arquitectura = str_remove_all(arquitectura, regex("(?<=APU).*"))
arquitectura = str_remove_all(arquitectura, regex("(?<=APU).*"))
arquitectura = str_remove_all(arquitectura, regex("(?<=CPU).*"))
arquitectura = str_remove_all(arquitectura, regex("(?<=CPU).*"))

#Obteniendo la cantidad de objetos de la escena
objetos = str_remove_all(only_technical_data, regex("(?<=GPU).*"))
objetos = str_remove_all(objetos, regex("(?<=CPU).*"))
objetos = str_remove_all(objetos, regex("(?<=APU).*"))
objetos = str_replace_all(objetos, "-", ".")
```

Finalmente creamos una tabla con los datos limpios y los guardamos en un archivo de texto con el formato adecuado que necesitamos.

```
# Create a data table from a data frame
data_frame <- data.frame(TiempoResolucion=rep(c(only_time_data)),
                        Objetos=rep(c(objetos)),
                        Arquitectura=rep(c(arquitectura)),
                        Efectos=rep(c(efectos)),
                        Resolucion=rep(c(resolucion)))

write.table(data_frame, "Datos2kTabla.txt", quote = FALSE, row.names=FALSE)
```

 Datos2kTabla: Bloc de notas

Archivo Edición Formato Ver Ayuda

TiempoResolucion Objetos Arquitectura Efectos Resolucion

```
6.53 16000 APU 010 1280x720.sc
4.22 1000 APU 011 1440x900.sc
6.13 1000 APU 101 1280x720.sc
21.75 16000 APU 110 1440x900.sc
4.47 4000 APU 001 1280x720.sc
19.76 260000 APU 001 1280x720.sc
76.95 260000 APU 110 1440x900.sc
5.93 65000 APU 000 1280x720.sc
8.43 1000 APU 111 1440x900.sc
14.02 4000 APU 110 1920x1080.sc
6.78 1000 APU 110 1280x720.sc
7.34 1000 APU 110 1440x900.sc
9.66 4000 APU 100 1920x1080.sc
8.60 65000 APU 010 1280x720.sc
16.80 260000 APU 010 1280x720.sc
7.70 65000 APU 000 1920x1080.sc
6.88 65000 APU 000 1440x900.sc
124.75 260000 APU 101 1920x1080.sc
8.01 65000 APU 000 1920x1080.sc
10.00 16000 APU 100 1280x720.sc
4.99 4000 APU 000 1920x1080.sc
28.53 260000 APU 001 1920x1080.sc
5.37 4000 APU 010 1440x900.sc
4.76 16000 APU 000 1280x720.sc
29.47 65000 APU 011 1920x1080.sc
28.64 260000 APU 001 1920x1080.sc
3.67 1000 APU 001 1280x720.sc
7.61 16000 APU 001 1440x900.sc
136.74 65000 APU 111 1920x1080.sc
```

Estos datos ya limpios los cargamos para iniciar el análisis.

```
# Se leen los datos y se guardan en la variable Data
my_data <- read.table("Datos2kTabla.txt", header=TRUE,
                    colClasses = c("numeric", "factor", "factor", "factor", "factor"))

Data = my_data

# Se eliminan los datos originales de memoria
rm(my_data)
rm(data_frame)
rm(objetos)
rm(arquitectura)
rm(efectos)
rm(resolucion)
rm(only_technical_data)
rm(filtered_data)
rm(original_data)

headTail(Data)
str(Data)
summary(Data)
```

```
> headTail(Data)
  TiempoResolucion Objetos Arquitectura Efectos Resolucion
1             6.53  16000          APU    010 1280x720.sc
2             4.22   1000          APU    011 1440x900.sc
3             6.13   1000          APU    101 1280x720.sc
4            21.75  16000          APU    110 1440x900.sc
...             ...      <NA>      <NA>    <NA>    <NA>
1797          11.75  16000          GPU    010 1920x1080.sc
1798          31.93   4000          GPU    111 1440x900.sc
1799          47.3  260000          GPU    000 1440x900.sc
1800           5.44   1000          GPU    101 1440x900.sc

> str(Data)
'data.frame': 1800 obs. of  5 variables:
 $ TiempoResolucion: num  6.53 4.22 6.13 21.75 4.47 ...
 $ Objetos          : Factor w/ 5 levels "1000","16000",...: 2 1 1 2 4 3 3 5 1 4 ...
 $ Arquitectura     : Factor w/ 3 levels "APU","CPU","GPU": 1 1 1 1 1 1 1 1 1 ...
 $ Efectos          : Factor w/ 8 levels "000","001","010",...: 3 4 6 7 2 2 7 1 8 7 ...
 $ Resolucion       : Factor w/ 3 levels "1280x720.sc",...: 1 2 1 2 1 1 2 1 2 3 ...

> summary(Data)
  TiempoResolucion  Objetos  Arquitectura  Efectos      Resolucion
Min.   : 0.880    1000   :360    APU:600    000   :225    1280x720.sc :600
1st Qu.: 7.315    16000  :360    CPU:600    001   :225    1440x900.sc  :600
Median : 20.130   260000:360    GPU:600    010   :225    1920x1080.sc:600
Mean   : 66.326    4000   :360          011   :225
3rd Qu.: 64.073   65000  :360          100   :225
Max.   :1271.990                (other):450
```

4) ¿Es el tiempo de sintetizado lo único que se debe verificar en este experimento?

No, debemos considerar todos los factores y la interacción entre los mismos ya que el tiempo de sintetizado es la variable que estaría dependiendo de los demás factores como lo son la cantidad de objetos, resolución, efectos y la arquitectura sobre la cual se ejecutó el experimento.

5) ¿Cuántas repeticiones por escenario realizó el científico?

Para esto debemos ver cuantos escenarios distintos existían (entendiendo escenario como el conjunto de datos técnicos como lo son cantidad de objetos, efectos y resolución). Siguiendo con un **Summarize del Tiempo de Resolución dado los Datos Técnicos** podemos ver que **cada escenario fue ejecutado 5 veces**, por lo que en total son **360 ejecuciones del experimento**.

```
Summarize(TiempoResolucion ~ TechnicalData, data=Data, digits=4)

(Top Level) ±

Terminal Background Jobs

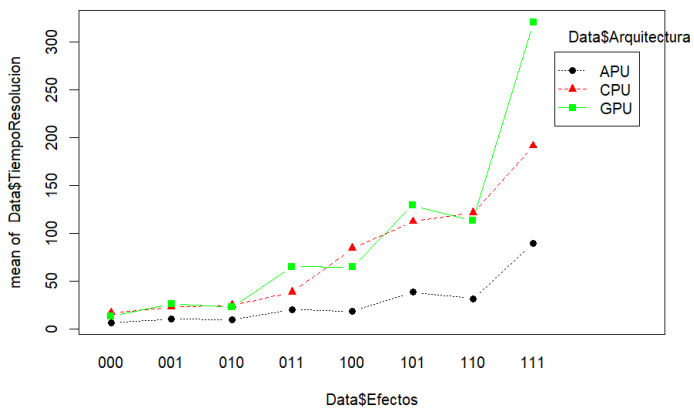
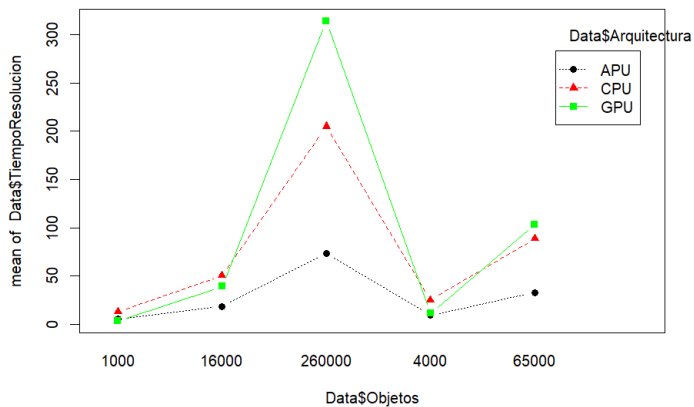
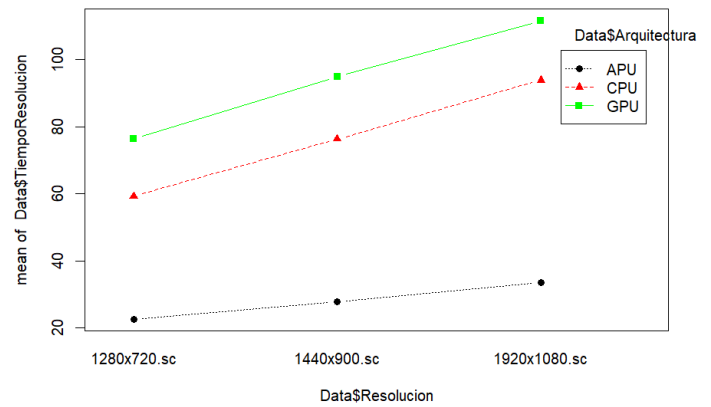
4.2.2 C:/Users/Andrey/Desktop/ExperimentDesign/Tareas/Tarea 2/
TechnicalData mean sd min Q1 median Q3 max
_1000APU000-1280x720.sc 5 3.474 0.0378 3.41 3.48 3.48 3.49 3.51
_1000APU000-1440x900.sc 5 3.924 0.1189 3.81 3.86 3.86 3.99 4.10
_1000APU000-1920x1080.sc 5 4.166 0.0764 4.09 4.10 4.15 4.23 4.26
_1000APU001-1280x720.sc 5 3.580 0.0682 3.50 3.53 3.58 3.62 3.67
_1000APU001-1440x900.sc 5 3.992 0.1062 3.87 3.91 4.02 4.02 4.14
_1000APU001-1920x1080.sc 5 4.238 0.1141 4.12 4.18 4.18 4.31 4.40
_1000APU010-1280x720.sc 5 3.742 0.0492 3.70 3.70 3.74 3.75 3.82
_1000APU010-1440x900.sc 5 4.090 0.1755 3.87 4.01 4.03 4.27 4.27
_1000APU010-1920x1080.sc 5 4.280 0.1149 4.13 4.23 4.26 4.35 4.43
_1000APU011-1280x720.sc 5 3.802 0.0460 3.74 3.78 3.80 3.83 3.86
_1000APU011-1440x900.sc 5 4.194 0.0635 4.09 4.18 4.22 4.23 4.25
_1000APU011-1920x1080.sc 5 4.502 0.0396 4.44 4.49 4.51 4.53 4.54
_1000APU100-1280x720.sc 5 5.324 0.1521 5.15 5.20 5.36 5.38 5.53
```

6) Gráficos de interacción para cada factor

```
interaction.plot(x.factor= Data$Objetos,
               trace.factor= Data$Arquitectura,
               response = Data$TiempoResolucion,
               fun=mean, type="b", col=c("black", "red", "green"),
               pch=c(19,17,15), fixed=TRUE, leg.bty="o")

interaction.plot(x.factor= Data$Efectos,
               trace.factor= Data$Arquitectura,
               response = Data$TiempoResolucion,
               fun=mean, type="b", col=c("black", "red", "green"),
               pch=c(19,17,15), fixed=TRUE, leg.bty="o")

interaction.plot(x.factor= Data$Resolucion,
               trace.factor= Data$Arquitectura,
               response = Data$TiempoResolucion,
               fun=mean, type="b", col=c("black", "red", "green"),
               pch=c(19,17,15), fixed=TRUE, leg.bty="o")
```



Seguido generamos el model lineal y ejecutamos el Anova

```
#Definimos Modelo Lineal y Anova
model = lm(TiempoResolucion ~ Objetos * Arquitectura * Efectos * Resolucion
           , data=Data)

Anova(model, type="II")
```

	Sum Sq	Df	F value	Pr(>F)
Objetos	8753306	4	4852765.9	< 2.2e-16 ***
Arquitectura	1415467	2	1569448.3	< 2.2e-16 ***
Efectos	6124196	7	1940118.5	< 2.2e-16 ***
Resolucion	216629	2	240195.2	< 2.2e-16 ***
Objetos:Arquitectura	2477612	8	686784.5	< 2.2e-16 ***
Objetos:Efectos	7321468	28	579852.3	< 2.2e-16 ***
Arquitectura:Efectos	1591765	14	252132.1	< 2.2e-16 ***
Objetos:Resolucion	220264	8	61056.3	< 2.2e-16 ***
Arquitectura:Resolucion	38619	4	21410.0	< 2.2e-16 ***
Efectos:Resolucion	193152	14	30594.8	< 2.2e-16 ***
Objetos:Arquitectura:Efectos	2595724	56	102789.3	< 2.2e-16 ***
Objetos:Arquitectura:Resolucion	51689	16	7164.0	< 2.2e-16 ***
Objetos:Efectos:Resolucion	227462	56	9007.4	< 2.2e-16 ***
Arquitectura:Efectos:Resolucion	46891	28	3713.7	< 2.2e-16 ***
Objetos:Arquitectura:Efectos:Resolucion	72187	112	1429.3	< 2.2e-16 ***
Residuals	649	1440		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

7) ¿Cómo es el comportamiento de la normalidad y homocedasticidad en los datos?

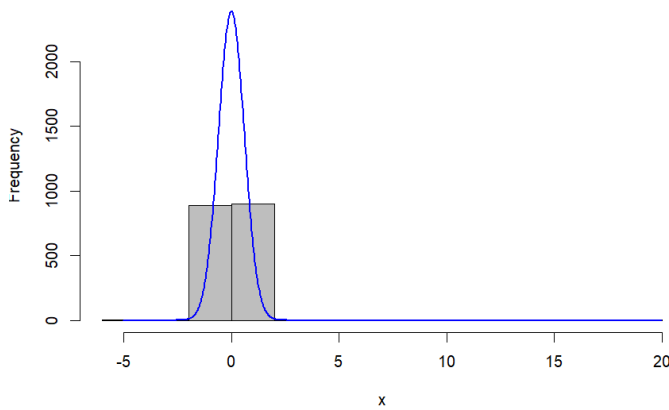
Para poder saber esto vamos a ejecutar el análisis de los supuestos:

```
#Evaluacion de los supuestos
x = residuals(model)
plotNormalHistogram(x)

#Ver el patron homcedasteicidad
plot(fitted(model), residuals(model))

plot(model)
```

- 8) ¿Existe alguna transformación que cumplir los supuestos necesarios? De existir, aplíquela a los datos. Con la transformación por raíz cuadrada podemos obtener mejor homocedasticidad.



```
#Transformacion de datos - Por raíz cuadrada
T.sqrt = sqrt(Data$TiempoResolucion)

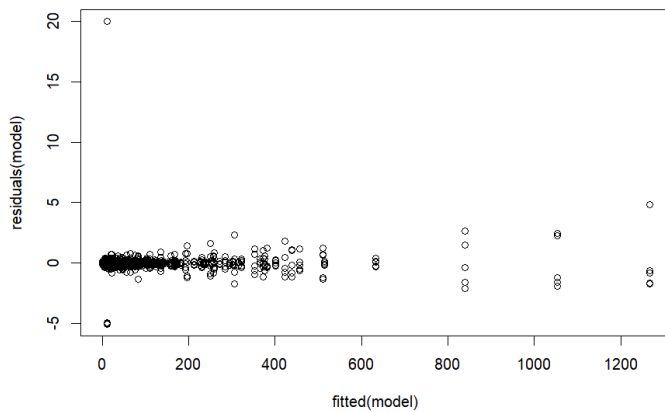
model = lm(T.sqrt ~ Objetos * Arquitectura * Efectos * Resolucion, data=Data)

Anova(model, type="II")

x = residuals(model)
plotNormalHistogram(x)

#Ver el patron homocedasticidad
plot(Fitted(model), residuals(model))

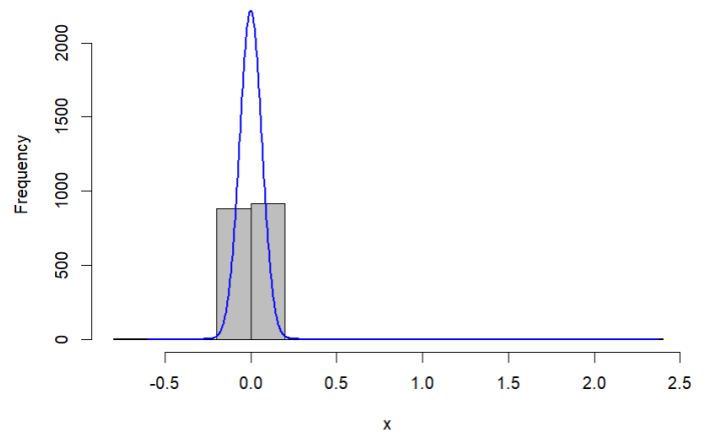
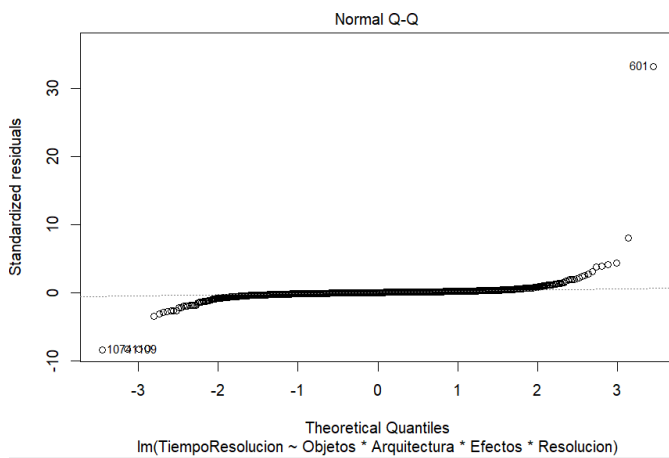
plot(model)
```



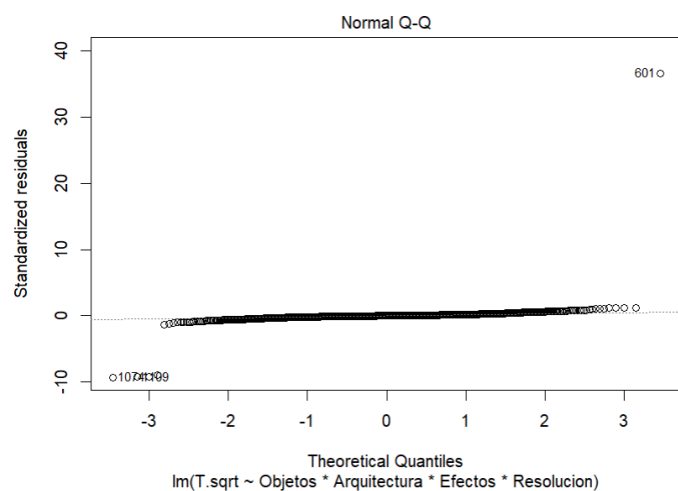
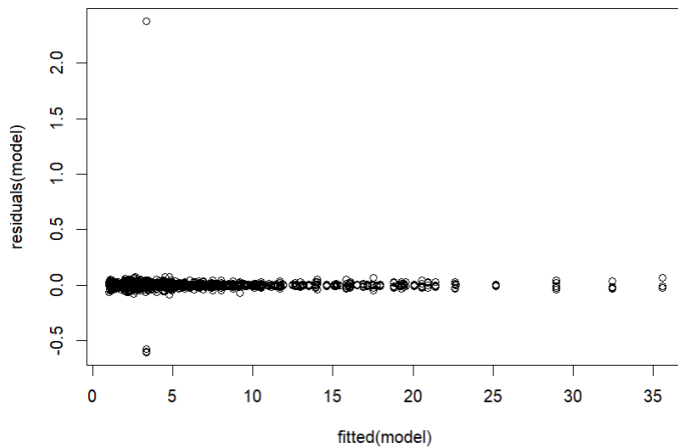
Response: T.sqrt

	Sum Sq	Df	F value	Pr(>F)	
objetos	21603.4	4	1.0306e+06	< 2.2e-16	***
Arquitectura	2980.9	2	2.8440e+05	< 2.2e-16	***
Efectos	12945.3	7	3.5288e+05	< 2.2e-16	***
Resolucion	463.2	2	4.4194e+04	< 2.2e-16	***
Objetos:Arquitectura	2944.3	8	7.0228e+04	< 2.2e-16	***
Objetos:Efectos	5538.6	28	3.7745e+04	< 2.2e-16	***
Arquitectura:Efectos	1284.6	14	1.7508e+04	< 2.2e-16	***
Objetos:Resolucion	151.0	8	3.6010e+03	< 2.2e-16	***
Arquitectura:Resolucion	41.5	4	1.9806e+03	< 2.2e-16	***
Efectos:Resolucion	111.5	14	1.5191e+03	< 2.2e-16	***
Objetos:Arquitectura:Efectos	750.7	56	2.5580e+03	< 2.2e-16	***
Objetos:Arquitectura:Resolucion	13.2	16	1.5706e+02	< 2.2e-16	***
Objetos:Efectos:Resolucion	57.4	56	1.9547e+02	< 2.2e-16	***
Arquitectura:Efectos:Resolucion	11.8	28	8.0599e+01	< 2.2e-16	***
Objetos:Arquitectura:Efectos:Resolucion	7.4	112	1.2528e+01	< 2.2e-16	***
Residuals	7.5	1440			

signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1



Con los resultados obtenidos podemos ver que la normalidad podría ser mucho mejor, por lo que buscaremos transformar los datos.



Como podemos observar gráficamente mejora un poco la normalidad, por lo que utilizaremos la prueba **LEVENE** para poder tener más información.

```
#Prueba de LEVENE para homocedasticidad |
leveneTest(T.sqrt ~ Objetos * Arquitectura * Efectos * Resolucion, data=Data)
```

```
Levene's Test for Homogeneity of Variance (center = median)
Df F value Pr(>F)
group 359 1.0262 0.3718
1440
```

Para la prueba de LEVENE se cumple el supuesto con valores grandes, por lo que podemos afirmar que estamos cumpliendo los principios de homocedasticidad con la transformación

9) Análisis post-hoc con los datos transformados.

```
#Análisis post-hoc con los datos transformados
marginal = lsmeans(model, pairwise ~ Arquitectura,
  adjust = "tukey")
CLD = cld(marginal, alpha=0.05, Letters = letters, adjust="tukey")
CLD
```

```
marginal = lsmeans(model, pairwise ~ Efectos,
  adjust = "tukey")
```

```
CLD = cld(marginal, alpha=0.05, Letters = letters, adjust="tukey")
```

```
CLD
```

```
marginal = lsmeans(model, pairwise ~ Resolucion,
  adjust = "tukey")
```

```
CLD = cld(marginal, alpha=0.05, Letters = letters, adjust="tukey")
```

```
CLD
```

```
> CLD
Arquitectura lsmean      SE   df lower.CL upper.CL .group
APU          4.441 0.002955 1440  4.434   4.448    a
GPU          7.059 0.002955 1440  7.052   7.066    b
CPU          7.271 0.002955 1440  7.264   7.278    c
```

Results are averaged over the levels of: Objetos, Efectos, Resolucion
Confidence level used: 0.95
Conf-level adjustment: sidak method for 3 estimates
P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05
NOTE: If two or more means share the same grouping symbol,
then we cannot show them to be different.
But we also did not show them to be the same.

```
Efectos lsmean      SE   df lower.CL upper.CL .group
000      3.097 0.004826 1440  3.084   3.111    a
010      3.804 0.004826 1440  3.791   3.817    b
001      3.846 0.004826 1440  3.833   3.859    c
011      5.349 0.004826 1440  5.336   5.363    d
100      6.279 0.004826 1440  6.265   6.292    e
110      7.974 0.004826 1440  7.961   7.987    f
101      8.111 0.004826 1440  8.098   8.125    g
111     11.597 0.004826 1440 11.583  11.610    h
```

```
> CLD
Resolucion lsmean      SE   df lower.CL upper.CL .group
1280x720.sc 5.623 0.002955 1440  5.616   5.630    a
1440x900.sc  6.284 0.002955 1440  6.277   6.291    b
1920x1080.sc 6.865 0.002955 1440  6.858   6.872    c
```

Con el análisis post-hoc podemos observar que no hay grupos estadísticamente similares en los factores con los datos transformados.

10) Gráficos de medias y bigotes para cada factor.

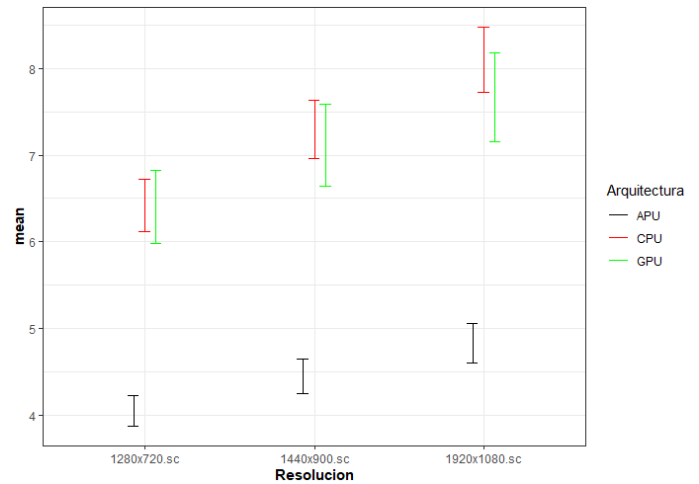
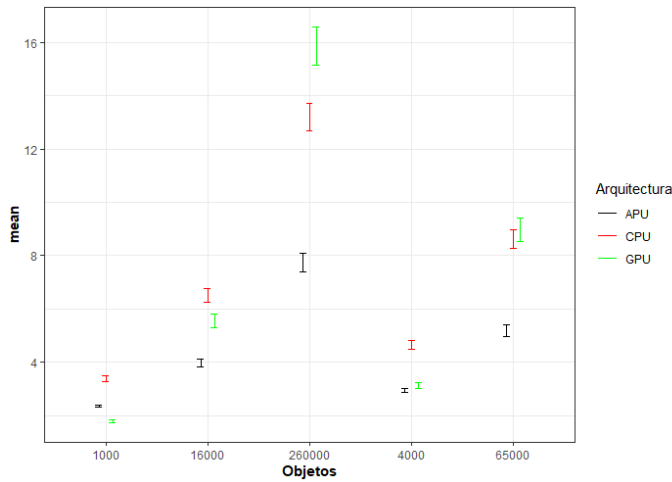
```
#Gráficos Finales
#Factor de objetos
Sum = Summarize(T.sqrt ~ Objetos + Arquitectura, data=Data, digits=4)

SumIsse = SumIsd / sqrt(SumIn)
SumIsse = signif(SumIsse, digits=3)
Sum

SumObjetos = factor(SumObjetos, levels=unique(SumObjetos))

pd=position_dodge(.2)

ggplot(Sum, aes(x = objetos, y = mean, color = Arquitectura)) +
  geom_errorbar(aes(ymin = mean - se, ymax = mean + se), width = 0.2, size = 0.7, position = pd) +
  theme_bw() + theme(axis.title = element_text(face="bold")) +
  scale_color_manual(values=c("black", "red", "green")) +
  ylab("Objetos")
```

```
#Factor de efectos
Sum = Summarize(T.sqrt ~ Efectos + Arquitectura, data=Data, digits=4)

SumSse = SumSsd / sqrt(SumSn)
SumSse = signif(SumSse, digits=3)
Sum

SumSEfectos = factor(SumSEfectos, levels=unique(SumSEfectos))

pd=position_dodge(.2)

ggplot(Sum, aes(x = Efectos, y = mean, color = Arquitectura)) +
  geom_errorbar(aes(ymin = mean - se, ymax = mean + se), width = 0.2, size = 0.7, position = pd) +
  theme_bw() + theme(axis.title = element_text(face="bold")) +
  scale_color_manual(values=c("black", "red", "green"))
ylab("Efectos")
```

```
#GRAFICO PRINCIPAL - PROMEDIOS TRANSFORMADOS

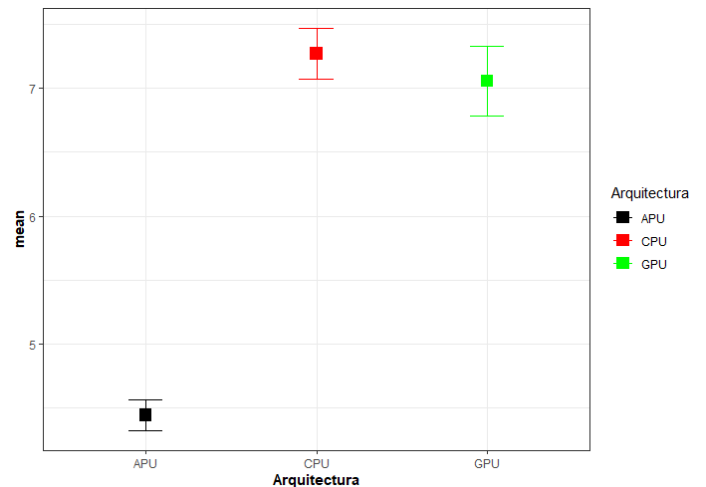
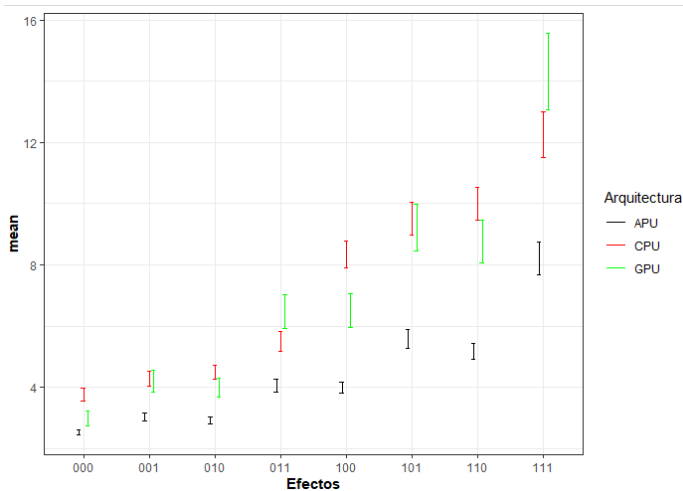
Sum = Summarize(T.sqrt ~ Arquitectura, data=Data, digits=3)

SumSse = SumSsd / sqrt(SumSn)
SumSse = signif(SumSse, digits=3)
Sum

SumSArquitectura = factor(SumSArquitectura, levels=unique(SumSArquitectura))

pd=position_dodge(.2)

ggplot(Sum, aes(x = Arquitectura, y = mean, color = Arquitectura)) +
  geom_errorbar(aes(ymin = mean - se, ymax = mean + se), width = 0.2, size = 0.7, position = pd) +
  geom_point(shape=15, size=4, position=pd) +
  theme_bw() + theme(axis.title = element_text(face="bold")) +
  scale_color_manual(values=c("black", "red", "green"))
ylab("Raíz cuadrada de arquitectura")
```



```
#Factor de resolución
Sum = Summarize(T.sqrt ~ Resolución + Arquitectura, data=Data, digits=4)

SumSse = SumSsd / sqrt(SumSn)
SumSse = signif(SumSse, digits=3)
Sum

SumSResolución = factor(SumSResolución, levels=unique(SumSResolución))

pd=position_dodge(.2)

ggplot(Sum, aes(x = Resolución, y = mean, color = Arquitectura)) +
  geom_errorbar(aes(ymin = mean - se, ymax = mean + se), width = 0.2, size = 0.7, position = pd) +
  theme_bw() + theme(axis.title = element_text(face="bold")) +
  scale_color_manual(values=c("black", "red", "green"))
ylab("Resolución")
```

11) Gráfico principal de promedios transformados.

Mediante este gráfico podemos observar como el error estándar de la solución en APU es más bajo, este gráfico generado con los datos transformados por raíz cuadrada logran que la prueba sea más difícil de pasar por lo que podemos concluir que los originales también cumplen la diferencia, para estos podemos destransformar.

De manera intuitiva se observa que los datos obtenidos mediante la solución ejecutada en la arquitectura APU son más bajos que la competencia con CPU y GPU en la mayoría de casos.

12) Gráfico principal de promedios destransformados.

```
#Ahora des-transformemos
Sum = Summarize(T.sqrt ~ Arquitectura, data=Data, digits=3)

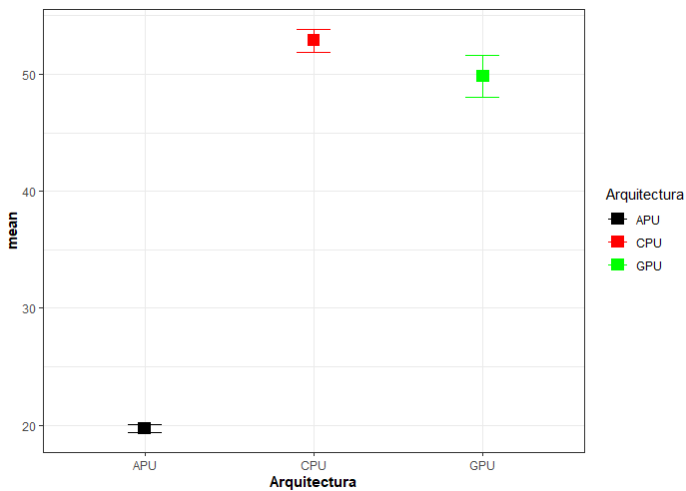
Sum$mean = Sum$mean ^ 2
Sum$sd = Sum$sd ^ 2
Sum

Sum$se = Sum$sd / sqrt(Sum$n)
Sum$se = signif(Sum$se, digits=3)
Sum

Sum$Arquitectura = factor(Sum$Arquitectura, levels=unique(Sum$Arquitectura))

pd=position_dodge(.2)

ggplot(Sum, aes(x = Arquitectura, y = mean, color = Arquitectura)) +
  geom_errorbar(aes(ymin = mean - se, ymax = mean + se), width = 0.2, size = 0.7, position = pd) +
  geom_point(shape=15, size=4, position=pd) +
  theme_bw() + theme(axis.title = element_text(face="bold")) + |
  scale_color_manual(values=c("black", "red", "green"))
ylab("Originales de Arquitectura")
```



Mediante este gráfico obtenemos una visión muy similar a los datos transformados por la raíz cuadrada por lo que podemos concluir que APU se comporta mejor y es estadísticamente distinto a las soluciones corridas en CPU y GPU.

13) ¿Qué considera que se está pasando por alto?

Se están pasando por alto consideraciones de los equipos donde fueron corridos los experimentos, también se dice que en el caso del CPU y GPU tienen el mismo precio de compra pero para proyectos relacionados a imágenes los resultados que se pueden obtener con un CPU de 800 dólares no son los mismo que con una GPU del mismo precio. Además posiblemente otros datos técnicos pudieron haber sido extraviados por la forma en que están generados los datos. También cada escenario fue ejecutado un numero de veces impar, por lo que una de las arquitecturas siempre va a tener al menos una corrida menos de cada escenario.

14) ¿Considera que la comparación entre arquitecturas es justa?

Considero que no por que se debería también correr el algoritmo híbrido o una versión del mismo en otros sets de CPU o GPU solamente para comparar. A su vez se debieron probar un número de veces iguales para cada arquitectura.

15) ¿Qué información falta con respecto a lo que corre en el CPU y GPU?

Factores relacionados al consumo de CPU y GPU durante el tiempo que se tomó en sintetizar la imagen.