Instituto Tecnológico de Costa Rica

Escuela de Computación

Maestría en Ciencias de la Computación

Diseño de Experimentos

Transformación de datos y diseños multifactoriales 2^k

Estudiante: Yirlania Mejías Rodríguez

Profesor: Ernesto Rivera Alvarado

Abril, 2023

Tabla de Contenidos

Diseños Multifactoriales 2^k

- Ampliamente utilizados
- Se utilizan para los k factores que únicamente tienen dos niveles
- Se exploran un gran número de factores que únicamente tienen dos niveles
- Incluimos factores relevantes para nuestro experimento

Tenemos dos factores: el de inclinación y el de altura. A cada celda le asignamos una letra distinta

Inclinación	Altura
30 (a)	10 (x)
60 (b)	20 (y)

La idea detrás de esto es condensar estos dos factores en uno solo. Entonces un nombre que le podríamos poner a este nuevo factor condensado sería I-A.

¿Cuáles van a ser los posibles niveles que va a tener el factor condensado en 2^k de estos dos?
Básicamente todas las posibles combinaciones de estos. Si agregamos un tercer factor , serían 8 (2^3), si tenemos 4 entonces serían 16 (2^4), es decir, 2^k posibles combinaciones.

Inclinación	Altura
a	x
a	y
b	x
b	y

Ejemplo de ANOVA multifactorial 2^k

Nomenclatura

NA: Acelerador no aplica
SW: Aceleración por software
HW: Aceleración por hardware

Software	Hardware
NA	NA
SW	HW

Tabulado

Volvemos a nuestro experimento para evaluar algoritmos en la red neuronal. Queremos evaluar el desempeño al entrenar con dos conjuntos de datos distintos y el hecho de utilizar un acelerador de software o hardware.

	Factores		
	Método de entrenamiento	Conjunto de Datos	Acelerador
Niveles	Algoritmo A	MT500	NA-NA
	Algoritmo B	MT1000	NA-SW
	Algoritmo C	MT5000	HW-NA
		MT50000	HW-SW

Ejecutamos el código que hemos utilizado siempre, para importar las bibliotecas, cargar la información y verificar que esta sea correcta.

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(car)){install.packages("car")}
if(!require(multcompView)){install.packages("multcompView")}
```

```
if(!require(lsmeans)){install.packages("lsmeans")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

```
     ln=("Algoritmo       Entrenamiento       Rendimiento   Acelerador
         'Algoritmo A'        MT500           12000          NA-NA
         'Algoritmo A'        MT500           14005          NA-NA
         'Algoritmo A'        MT500           13508          NA-NA
         'Algoritmo A'        MT500            9503          NA-NA
         'Algoritmo A'        MT500           14004          NA-NA
         'Algoritmo A'        MT1000          11502          NA-NA
         'Algoritmo A'        MT1000          13006          NA-NA
         'Algoritmo A'        MT1000          13252          NA-NA
         'Algoritmo A'        MT1000          14253          NA-NA
         'Algoritmo A'        MT1000          15003          NA-NA
         'Algoritmo A'        MT5000          12504          NA-NA
         'Algoritmo A'        MT5000          11504          NA-NA
         'Algoritmo A'        MT5000           9500          NA-NA
         'Algoritmo A'        MT5000          11506          NA-NA
         'Algoritmo A'        MT5000          16000          NA-NA
         'Algoritmo A'        MT50000         13008          NA-NA
         'Algoritmo A'        MT50000         10506          NA-NA
         'Algoritmo A'        MT50000         13005          NA-NA
         'Algoritmo A'        MT50000         17002          NA-NA
         'Algoritmo A'        MT50000         13008          NA-NA
         'Algoritmo B'        MT500           11005          NA-NA
         'Algoritmo B'        MT500           12007          NA-NA
         'Algoritmo B'        MT500           12509          NA-NA
         'Algoritmo B'        MT500           10504          NA-NA
         'Algoritmo B'        MT500           12002          NA-NA
         'Algoritmo B'        MT1000          12504          NA-NA
         'Algoritmo B'        MT1000          13501          NA-NA
         'Algoritmo B'        MT1000          13501          NA-NA
         'Algoritmo B'        MT1000          13252          NA-NA
         'Algoritmo B'        MT1000          15256          NA-NA
         'Algoritmo B'        MT5000          12253          NA-NA
         'Algoritmo B'        MT5000          11255          NA-NA
         'Algoritmo B'        MT5000          10006          NA-NA
         'Algoritmo B'        MT5000          11252          NA-NA
         'Algoritmo B'        MT5000          14004          NA-NA
         'Algoritmo B'        MT50000         12007          NA-NA
         'Algoritmo B'        MT50000         11505          NA-NA
         'Algoritmo B'        MT50000         14009          NA-NA
         'Algoritmo B'        MT50000         15000          NA-NA
         'Algoritmo B'        MT50000         12009          NA-NA
         'Algoritmo C'        MT500            9000          NA-NA
         'Algoritmo C'        MT500           11003          NA-NA
         'Algoritmo C'        MT500           11505          NA-NA
         'Algoritmo C'        MT500            9509          NA-NA
         'Algoritmo C'        MT500           11003          NA-NA
         'Algoritmo C'        MT1000          11508          NA-NA
         'Algoritmo C'        MT1000          12508          NA-NA
         'Algoritmo C'        MT1000          12506          NA-NA
         'Algoritmo C'        MT1000          12254          NA-NA
         'Algoritmo C'        MT1000          13253          NA-NA
         'Algoritmo C'        MT5000          11255          NA-NA
         'Algoritmo C'        MT5000          10257          NA-NA
```

'Algoritmo C'	MT5000	9500	NA-NA
'Algoritmo C'	MT5000	9255	NA-NA
'Algoritmo C'	MT5000	12009	NA-NA
'Algoritmo C'	MT50000	11000	NA-NA
'Algoritmo C'	MT50000	9509	NA-NA
'Algoritmo C'	MT50000	13009	NA-NA
'Algoritmo C'	MT50000	14005	NA-NA
'Algoritmo C'	MT50000	11001	NA-NA
'Algoritmo A'	MT500	12046	NA-SW
'Algoritmo A'	MT500	14589	NA-SW
'Algoritmo A'	MT500	13723	NA-SW
'Algoritmo A'	MT500	9799	NA-SW
'Algoritmo A'	MT500	14715	NA-SW
'Algoritmo A'	MT1000	11144	NA-SW
'Algoritmo A'	MT1000	13920	NA-SW
'Algoritmo A'	MT1000	13226	NA-SW
'Algoritmo A'	MT1000	14845	NA-SW
'Algoritmo A'	MT1000	15142	NA-SW
'Algoritmo A'	MT5000	12352	NA-SW
'Algoritmo A'	MT5000	11296	NA-SW
'Algoritmo A'	MT5000	9737	NA-SW
'Algoritmo A'	MT5000	11129	NA-SW
'Algoritmo A'	MT5000	16409	NA-SW
'Algoritmo A'	MT50000	13872	NA-SW
'Algoritmo A'	MT50000	10100	NA-SW
'Algoritmo A'	MT50000	13419	NA-SW
'Algoritmo A'	MT50000	17398	NA-SW
'Algoritmo A'	MT50000	13164	NA-SW
'Algoritmo B'	MT500	11047	NA-SW
'Algoritmo B'	MT500	12226	NA-SW
'Algoritmo B'	MT500	12105	NA-SW
'Algoritmo B'	MT500	10418	NA-SW
'Algoritmo B'	MT500	12446	NA-SW
'Algoritmo B'	MT1000	12156	NA-SW
'Algoritmo B'	MT1000	13968	NA-SW
'Algoritmo B'	MT1000	13891	NA-SW
'Algoritmo B'	MT1000	13778	NA-SW
'Algoritmo B'	MT1000	15448	NA-SW
'Algoritmo B'	MT5000	12441	NA-SW
'Algoritmo B'	MT5000	11767	NA-SW
'Algoritmo B'	MT5000	10340	NA-SW
'Algoritmo B'	MT5000	11306	NA-SW
'Algoritmo B'	MT5000	14565	NA-SW
'Algoritmo B'	MT50000	12725	NA-SW
'Algoritmo B'	MT50000	11169	NA-SW
'Algoritmo B'	MT50000	14749	NA-SW
'Algoritmo B'	MT50000	15566	NA-SW
'Algoritmo B'	MT50000	12239	NA-SW
'Algoritmo C'	MT500	9082	NA-SW
'Algoritmo C'	MT500	11887	NA-SW
'Algoritmo C'	MT500	11799	NA-SW
'Algoritmo C'	MT500	9300	NA-SW
'Algoritmo C'	MT500	11049	NA-SW
'Algoritmo C'	MT1000	11378	NA-SW
'Algoritmo C'	MT1000	12659	NA-SW
'Algoritmo C'	MT1000	12905	NA-SW
'Algoritmo C'	MT1000	12782	NA-SW

'Algoritmo C'	MT1000	13196	NA-SW
'Algoritmo C'	MT5000	11795	NA-SW
'Algoritmo C'	MT5000	10316	NA-SW
'Algoritmo C'	MT5000	9947	NA-SW
'Algoritmo C'	MT5000	9420	NA-SW
'Algoritmo C'	MT5000	12699	NA-SW
'Algoritmo C'	MT50000	11024	NA-SW
'Algoritmo C'	MT50000	9556	NA-SW
'Algoritmo C'	MT50000	13900	NA-SW
'Algoritmo C'	MT50000	14006	NA-SW
'Algoritmo C'	MT50000	11738	NA-SW
'Algoritmo A'	MT500	126572	HW-NA
'Algoritmo A'	MT500	140058	HW-NA
'Algoritmo A'	MT500	139580	HW-NA
'Algoritmo A'	MT500	92583	HW-NA
'Algoritmo A'	MT500	148057	HW-NA
'Algoritmo A'	MT1000	110078	HW-NA
'Algoritmo A'	MT1000	131942	HW-NA
'Algoritmo A'	MT1000	133797	HW-NA
'Algoritmo A'	MT1000	140026	HW-NA
'Algoritmo A'	MT1000	155479	HW-NA
'Algoritmo A'	MT5000	125809	HW-NA
'Algoritmo A'	MT5000	114264	HW-NA
'Algoritmo A'	MT5000	98797	HW-NA
'Algoritmo A'	MT5000	113400	HW-NA
'Algoritmo A'	MT5000	168898	HW-NA
'Algoritmo A'	MT50000	133452	HW-NA
'Algoritmo A'	MT50000	101641	HW-NA
'Algoritmo A'	MT50000	133155	HW-NA
'Algoritmo A'	MT50000	175156	HW-NA
'Algoritmo A'	MT50000	131945	HW-NA
'Algoritmo B'	MT500	110317	HW-NA
'Algoritmo B'	MT500	129244	HW-NA
'Algoritmo B'	MT500	127966	HW-NA
'Algoritmo B'	MT500	109783	HW-NA
'Algoritmo B'	MT500	122936	HW-NA
'Algoritmo B'	MT1000	128830	HW-NA
'Algoritmo B'	MT1000	134437	HW-NA
'Algoritmo B'	MT1000	138321	HW-NA
'Algoritmo B'	MT1000	132000	HW-NA
'Algoritmo B'	MT1000	157693	HW-NA
'Algoritmo B'	MT5000	121964	HW-NA
'Algoritmo B'	MT5000	119872	HW-NA
'Algoritmo B'	MT5000	106654	HW-NA
'Algoritmo B'	MT5000	112666	HW-NA
'Algoritmo B'	MT5000	145535	HW-NA
'Algoritmo B'	MT50000	127938	HW-NA
'Algoritmo B'	MT50000	115179	HW-NA
'Algoritmo B'	MT50000	143021	HW-NA
'Algoritmo B'	MT50000	150357	HW-NA
'Algoritmo B'	MT50000	121216	HW-NA
'Algoritmo C'	MT500	95474	HW-NA
'Algoritmo C'	MT500	113776	HW-NA
'Algoritmo C'	MT500	117473	HW-NA
'Algoritmo C'	MT500	92900	HW-NA
'Algoritmo C'	MT500	115582	HW-NA
'Algoritmo C'	MT1000	115279	HW-NA

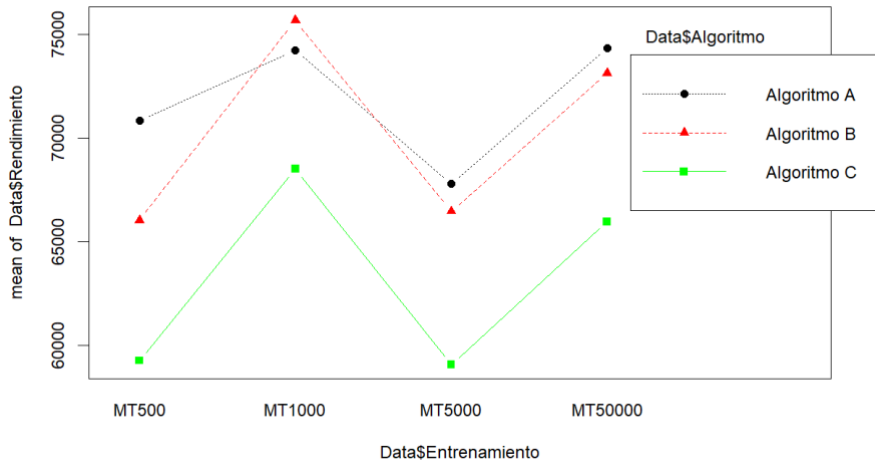
'Algoritmo C'	MT1000	122184	HW-NA
'Algoritmo C'	MT1000	124770	HW-NA
'Algoritmo C'	MT1000	128403	HW-NA
'Algoritmo C'	MT1000	135219	HW-NA
'Algoritmo C'	MT5000	112562	HW-NA
'Algoritmo C'	MT5000	108736	HW-NA
'Algoritmo C'	MT5000	91064	HW-NA
'Algoritmo C'	MT5000	98171	HW-NA
'Algoritmo C'	MT5000	120277	HW-NA
'Algoritmo C'	MT50000	111299	HW-NA
'Algoritmo C'	MT50000	90193	HW-NA
'Algoritmo C'	MT50000	135178	HW-NA
'Algoritmo C'	MT50000	146158	HW-NA
'Algoritmo C'	MT50000	113845	HW-NA
'Algoritmo A'	MT500	124252	HW-SW
'Algoritmo A'	MT500	143833	HW-SW
'Algoritmo A'	MT500	138907	HW-SW
'Algoritmo A'	MT500	91010	HW-SW
'Algoritmo A'	MT500	143901	HW-SW
'Algoritmo A'	MT1000	116563	HW-SW
'Algoritmo A'	MT1000	136455	HW-SW
'Algoritmo A'	MT1000	130411	HW-SW
'Algoritmo A'	MT1000	140060	HW-SW
'Algoritmo A'	MT1000	154308	HW-SW
'Algoritmo A'	MT5000	124480	HW-SW
'Algoritmo A'	MT5000	111552	HW-SW
'Algoritmo A'	MT5000	99135	HW-SW
'Algoritmo A'	MT5000	110208	HW-SW
'Algoritmo A'	MT5000	167228	HW-SW
'Algoritmo A'	MT50000	134267	HW-SW
'Algoritmo A'	MT50000	102119	HW-SW
'Algoritmo A'	MT50000	138036	HW-SW
'Algoritmo A'	MT50000	171632	HW-SW
'Algoritmo A'	MT50000	130666	HW-SW
'Algoritmo B'	MT500	116942	HW-SW
'Algoritmo B'	MT500	129721	HW-SW
'Algoritmo B'	MT500	128834	HW-SW
'Algoritmo B'	MT500	100390	HW-SW
'Algoritmo B'	MT500	127771	HW-SW
'Algoritmo B'	MT1000	121789	HW-SW
'Algoritmo B'	MT1000	135311	HW-SW
'Algoritmo B'	MT1000	136587	HW-SW
'Algoritmo B'	MT1000	139664	HW-SW
'Algoritmo B'	MT1000	151543	HW-SW
'Algoritmo B'	MT5000	128962	HW-SW
'Algoritmo B'	MT5000	110157	HW-SW
'Algoritmo B'	MT5000	106129	HW-SW
'Algoritmo B'	MT5000	114634	HW-SW
'Algoritmo B'	MT5000	143337	HW-SW
'Algoritmo B'	MT50000	129292	HW-SW
'Algoritmo B'	MT50000	117502	HW-SW
'Algoritmo B'	MT50000	143687	HW-SW
'Algoritmo B'	MT50000	153488	HW-SW
'Algoritmo B'	MT50000	129773	HW-SW
'Algoritmo C'	MT500	99920	HW-SW
'Algoritmo C'	MT500	110833	HW-SW
'Algoritmo C'	MT500	117879	HW-SW

```
          'Algoritmo C'        MT500            96441          HW-SW
          'Algoritmo C'        MT500           119688          HW-SW
          'Algoritmo C'        MT1000          117995          HW-SW
          'Algoritmo C'        MT1000          122984          HW-SW
          'Algoritmo C'        MT1000          120317          HW-SW
          'Algoritmo C'        MT1000          120213          HW-SW
          'Algoritmo C'        MT1000          137806          HW-SW
          'Algoritmo C'        MT5000          117014          HW-SW
          'Algoritmo C'        MT5000          105529          HW-SW
          'Algoritmo C'        MT5000           98755          HW-SW
          'Algoritmo C'        MT5000           96010          HW-SW
          'Algoritmo C'        MT5000          126548          HW-SW
          'Algoritmo C'        MT50000         113527          HW-SW
          'Algoritmo C'        MT50000          99385          HW-SW
          'Algoritmo C'        MT50000         136573          HW-SW
          'Algoritmo C'        MT50000         141965          HW-SW
         'Algoritmo C'        MT50000         111994          HW-SW")


Data = read.table(textConnection(ln), header = TRUE)
Data$Entrenamiento = factor(Data$Entrenamiento,
                            levels=unique(Data$Entrenamiento))
library(psych)
headTail(Data)
str(Data)
summary(Data)
```

Gráfico de Interacción

Realizamos el gráfico de interacción por medio del siguiente código:

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



Los algoritmos tienen un mejor rendimiento cuando no tienen acelerador o cuando el acelerador es de software. Y parece que cuando hay un acelerador de hardware empeora los resultados. Además se puede observar que la diferencia entre no tener acelerador y tener un acelerador de software de 10k dólares, es muy poca, por lo tanto parece que no vale la pena gastar este dinero.

Modelo lineal y ANOVA

Realizamos el modelo lineal utilizando *, para poder analizar todos los factores y las interacciones.

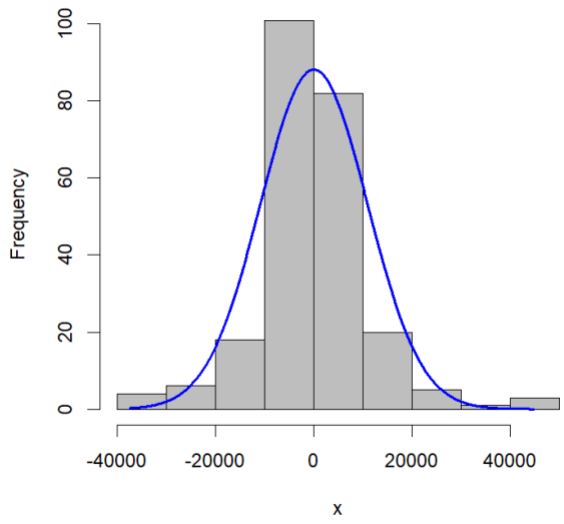
```
model = lm(Rendimiento ~ Entrenamiento * Algoritmo * Acelerador, data=Data)
library(car)
Anova(model, type = “II”)
```

```
Anova Table (Type II tests)

Response: Rendimiento
                                    Sum Sq  Df   F value    Pr(>F)
Entrenamiento                    3.1056e+09   3    7.0491 0.0001608 ***
Algoritmo                        3.3845e+09   2   11.5231 1.879e-05 ***
Acelerador                       7.5562e+11   3 1715.0914 < 2.2e-16 ***
Entrenamiento:Algoritmo          2.4891e+08   6    0.2825 0.9447359
Entrenamiento:Acelerador         2.0023e+09   9    1.5149 0.1449261
Algoritmo:Acelerador             2.2782e+09   6    2.5855 0.0197057 *
Entrenamiento:Algoritmo:Acelerador 2.0204e+08  18    0.0764 0.9999999
Residuals                        2.8196e+10 192
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

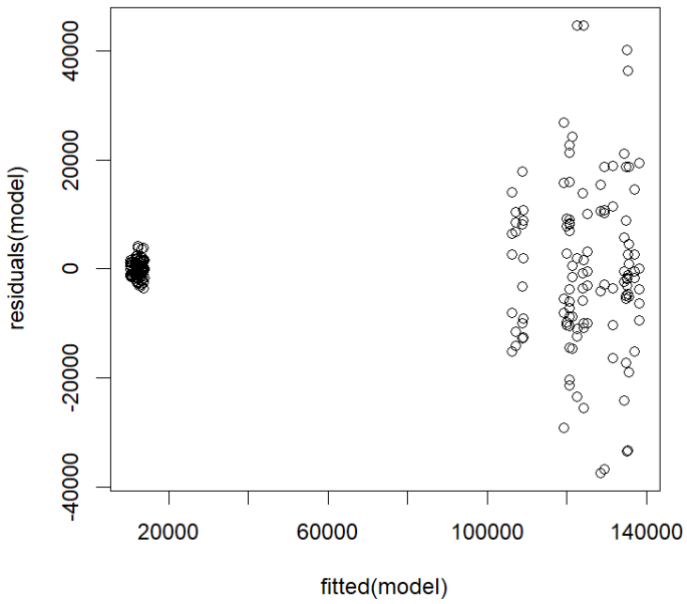
Al realizar el análisis ANOVA se puede observar que los primeros 3 datos, tienen p-values cercanos a 0, es decir hay una diferencia estadística significativa entre los grupos de entrenamiento, los algoritmos y los aceleradores. Pero no sabemos quién es peor o mejor, solamente sabemos que hay diferencia estadística. También lo podemos interpretar como que estos factores impactan la variable de respuesta y las interacciones no lo impactan.

Análisis de los supuestos

```
x = residuals(model)
library(rcompanion)
plotNormalHistogram(x)
plot(fitted(model), residuals(model))
```



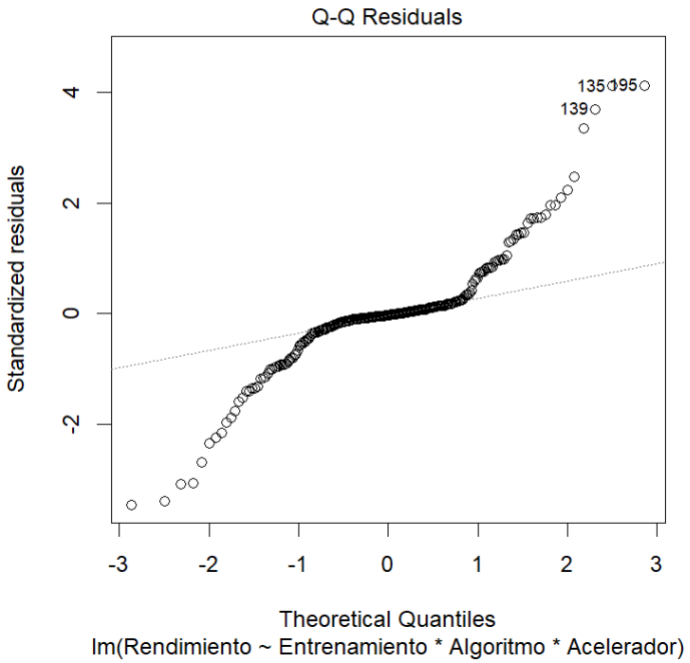
El histograma de residuos se ve bastante bien, se encuentra entre lo esperado, sin embargo, al realizar el análisis de homocedasticidad observamos un gráfico no ideal. Se puede ver un patrón tipo creciente a la derecha, como un cono. Además, otra forma que podemos ver que no cumple con homocedasticidad es por que vemos como se encuentran agrupados. Deberían verse como los de la derecha, con cierto distanciamiento entre estos.



Gráficos adicionales

Corremos el siguiente código para ver el gráfico de cuartiles. En este podemos observar como los datos están bastante alejados de la normalidad.

```
plot(model)
```



En estos casos que no se cumple con los supuestos realizamos transformación de datos

Transformación de Datos

Primer intento: Transformación por raíz cuadrada

Básicamente estas dos líneas son las únicas que implican la transformación de datos. Importar la biblioteca y aplicar la función raíz cuadrada a la variable dependiente que en este caso es rendimiento.

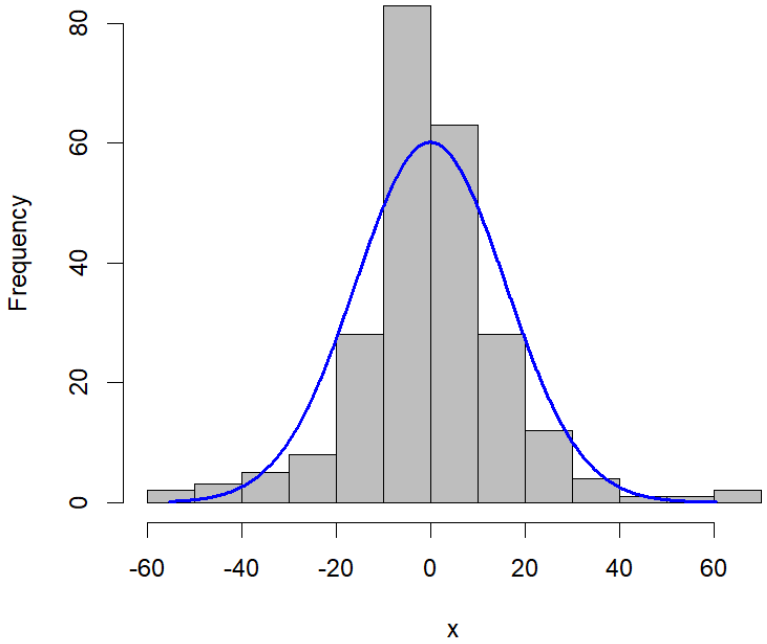
```
library(rcompanion)
T_sqrt = sqrt(Data$Rendimiento)

Notemos que ahora el modelo lineal lo hacemos sobre la transformación por
raíz cuadrada del rendimiento

model = lm (T_sqrt ~ Entrenamiento * Algoritmo * Acelerador, data=Data)

library(car)
Anova(model, Type="II")
```

```
Anova Table (Type II tests)

Response: T_sqrt
                                 Sum Sq  Df   F value    Pr(>F)
Entrenamiento                      9545   3   10.0836 3.356e-06 ***
Algoritmo                          9671   2   15.3253 6.684e-07 ***
Acelerador                      3492852   3 3689.9304 < 2.2e-16 ***
Entrenamiento:Algoritmo             767   6    0.4052    0.8750
Entrenamiento:Acelerador           2223   9    0.7827    0.6326
Algoritmo:Acelerador               2626   6    1.3871    0.2216
Entrenamiento:Algoritmo:Acelerador  255  18    0.0448    1.0000
Residuals                         60582 192
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La transformación por raíz cuadrada es la menos agresiva de todas, hay varias más, pero se inicia por este tipo de transformación y nos detenemos en el punto donde ya se cumplen los supuestos.

Ahora en el histograma de los residuos de igual manera podemos observar que se ve bien.

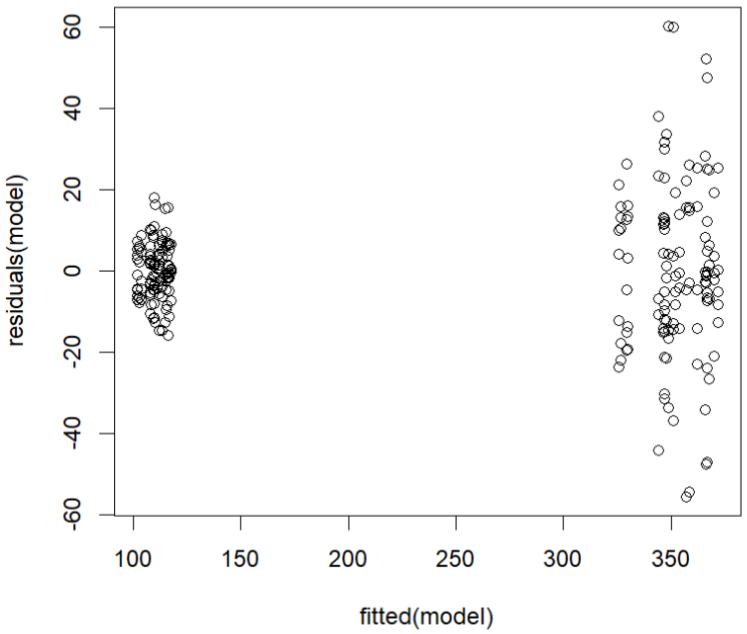
```
x=residuals(model)

library(rcompanion)
plotNormalHistogram(x)
```



Ahora analizamos la homocedasticidad

```
plot(fitted(model), residuals(model))
```

Al ver el gráfico podemos notar que algo mejoró, sin embargo no es suficiente. Aún podemos observar la tendencia creciente. Entonces aún necesitamos realizar más transformación de datos



Segundo Intento: Transformación por raíz cúbica

Tenemos que hacer la raíz cúbica de esta manera, para eliminar las soluciones con números imaginarios

```
library(rcompanion)
T_cub = sign(Data$Rendimiento) * abs(Data$Rendimiento)^(1/3)
```

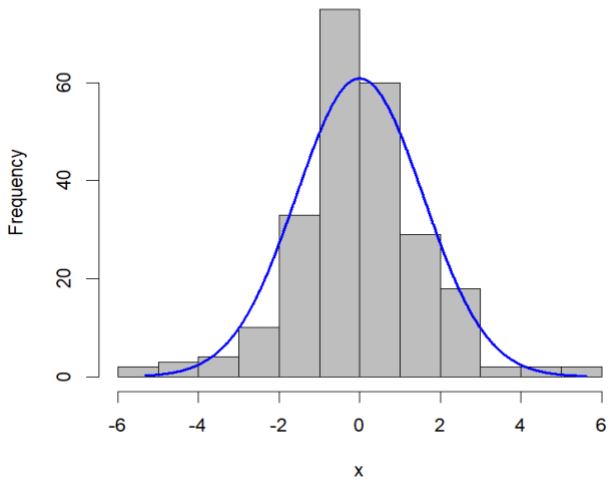
Sacamos el modelo lineal de la transformación de los datos con raíz cúbica

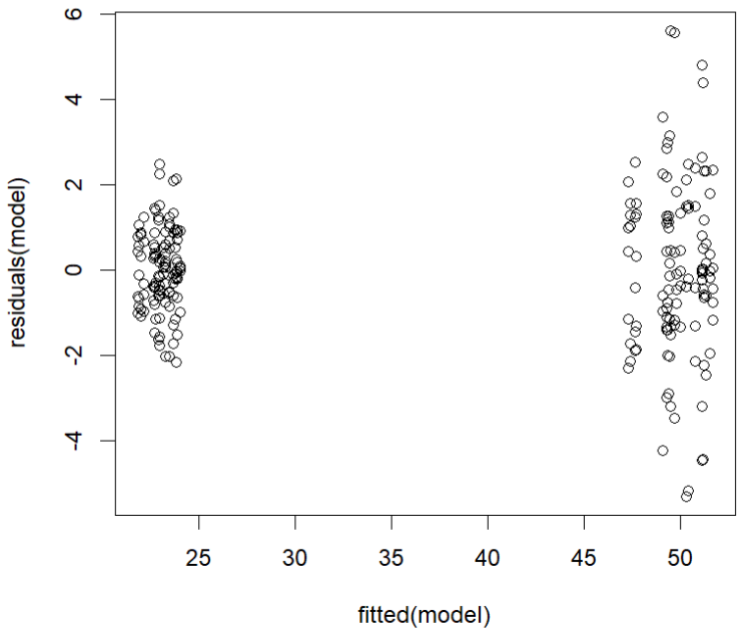
```
model = lm(T_cub ~ Entrenamiento * Algoritmo * Acelerador, data=Data)
library(car)
Anova(model, type =”II”)
x=residuals(model)
```

```
Anova Table (Type II tests)

Response: T_cub
                                 Sum Sq  Df   F value    Pr(>F)
Entrenamiento                       108   3   11.7101 4.428e-07 ***
Algoritmo                           107   2   17.2949 1.241e-07 ***
Acelerador                        43031   3 4649.8989 < 2.2e-16 ***
Entrenamiento:Algoritmo               9   6    0.4766    0.8252
Entrenamiento:Acelerador             11   9    0.4078    0.9300
Algoritmo:Acelerador                 14   6    0.7802    0.5864
Entrenamiento:Algoritmo:Acelerador    2  18    0.0303    1.0000
Residuals                           592 192
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

De igual forma el histograma se ve normal , y al analizar el gráfico de la homocedasticidad igual se ve mejora pero aún no es homogéneo

```
library(rcompanion)
plotNormalHistogram(x)
plot(fitted(model), residuals(model))
```





Tercer Intento: Transformación por logaritmo

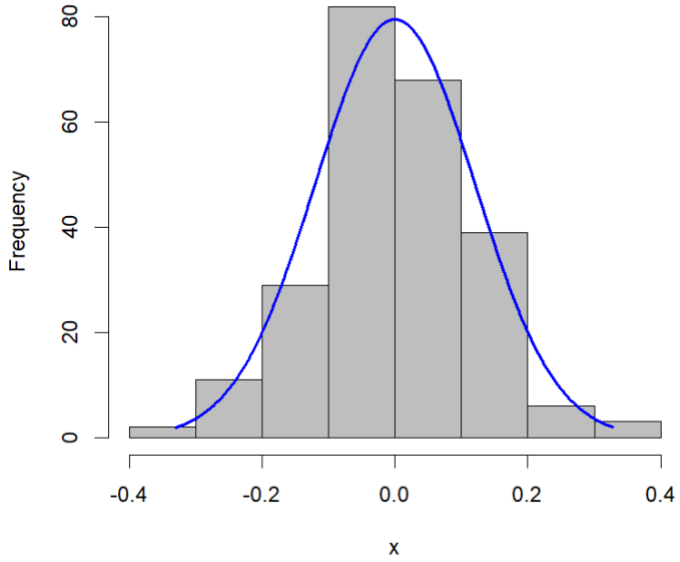
```
library(rcompanion)
T_log = log(Data$Rendimiento)

model = lm(T_log ~ Entrenamiento * Algoritmo * Acelerador, data=Data)

library(car)
Anova(model, type ="II")

x=residuals(model)
```

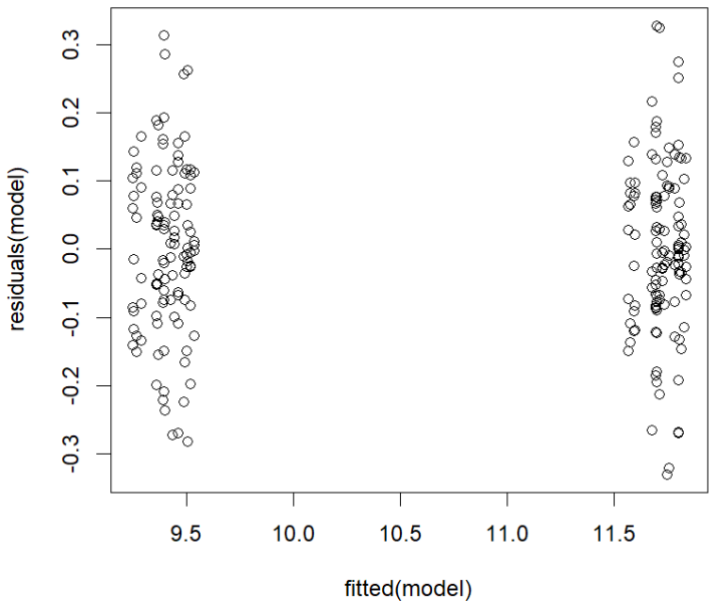
```
Anova Table (Type II tests)

Response: T_log
                                  Sum Sq  Df   F value    Pr(>F)
Entrenamiento                       0.78   3   14.4599 1.566e-08 ***
Algoritmo                           0.73   2   20.1075 1.179e-08 ***
Acelerador                        321.12   3 5928.9761 < 2.2e-16 ***
Entrenamiento:Algoritmo             0.07   6    0.6144    0.7187
Entrenamiento:Acelerador            0.01   9    0.0355    1.0000
Algoritmo:Acelerador                0.00   6    0.0291    0.9999
Entrenamiento:Algoritmo:Acelerador  0.01  18    0.0216    1.0000
Residuals                           3.47 192
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
library(rcompanion)
```

```
plotNormalHistogram(x)
```

De igual forma el histograma se ve normal y este caso se puede observar como se ha ido achatando, y al analizar el gráfico de la homocedasticidad ya se puede observar que si se cumple estos dos supuestos.





Prueba de Levene

Esta permite analizar la homocedasticidad

```
leveneTest(T_log ~ Entrenamiento * Algoritmo * Acelerador, data=Data)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value Pr(>F)
group  47  0.4057 0.9998
      192
```

La hipótesis nula en la prueba de Levene es que no hay diferencia entre la varianza de los grupos, entonces un p-value alto es lo que queremos como resultado para la homocedasticidad. Al observar el resultado vemos que el p-value es super buena.

Características de Transformación de Datos

- Lo hacemos cuando no logramos cumplir los supuestos
- También se utiliza para obligar a que las variables tengan distribuciones similares
- Típicamente solo vamos a transformar la variable dependiente.
- En modelos complejos y regresión múltiple se puede transformar tanto la dependiente como las independientes
- No hay nada prohibido en transformar variables
- Lo único es que debemos ser cuidadosos en cómo reportamos los resultados. No se vale hacer transformación de datos y presentar los resultados con los datos originales
- Por ejemplo: se identificó una diferencia significativa en la transformación logarítmica del rendimiento
- Para estadísticas resumen, podemos presentar los promedios de los valores transformados
- O des-transformar esos promedios a sus unidades originales. Si se vale ejecutar todo el análisis con los datos transformados y al final los promedio que obtuvimos de los datos transformados destransformarlos para reportar con las unidades originales
- Algunas medidas son normalmente distribuidas
- Otras en su comportamiento natural son logarítmicamente distribuidas. Ejemplo: las pérdidas o ganancias de decibeles o la contaminación de agua
- Se nota con valores muy bajos en conjunto con altos y muy altos
- Para distribuciones sesgadas las transformaciones que vimos sirven
- Con logaritmo, se suele sumar una constante para evitar el caso de log(0), para el cual el logaritmo se indefine
- Si ninguna de las transformaciones anteriores no funcionó, existen otros métodos de transformaciones:
    - Tukeys Ladder of Powers
    - Box-Cox
    - R ya tiene herramientas integradas para hacer uso de dichas transformaciones

Las transformaciones en sí que son? Lo que estamos haciendo es literalmente sacar raíz cuadrada, cúbica o logaritmo a nuestra variable dependiente, en este caso a los datos de rendimiento.

Orden de la transformación de los datos

Por qué tenemos que seguir el orden de la transformación (de la menos agresiva a la más)? Esto es porque estas transformaciones reducen las distancias entre los datos, y entre más agresiva la transformación de datos, la prueba para medir diferencia entre los grupos la estamos haciendo más difícil de pasar. Cada vez estamos minimizando la posibilidad de que los grupos sean distintos, que es lo que queremos analizar con el ANOVA. Además hay casos en los que una raíz cúbica más bien nos arruinan los datos y el logaritmo aún peor.

Reanudando el ANOVA

Análisis post-hoc

```
library(lsmeans)
```

El modelo de aquí es el que utiliza la transformación del algoritmo

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

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

```
 Algoritmo   lsmean      SE  df lower.CL upper.CL .group
 Algoritmo C  10.49 0.01502 192    10.45    10.52  a
 Algoritmo B  10.59 0.01502 192    10.56    10.63   b
 Algoritmo A  10.61 0.01502 192    10.58    10.65   b

Results are averaged over the levels of: Entrenamiento, Acelerador
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.
```

Podemos observar que B y A pertenecen al mismo grupo. Además, podemos ver que el algoritmo con mejor rendimiento es el C, ya que es el que tiene un promedio menor y se encuentra en un grupo distinto.

Realizamos el análisis post-hoc ahora por entrenamiento

```
library(multcomp)
```

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

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

```
 Entrenamiento lsmean      SE  df lower.CL upper.CL .group
 MT5000         10.50 0.01735 192    10.46    10.55  a
 MT500          10.52 0.01735 192    10.47    10.56  a
 MT50000        10.60 0.01735 192    10.56    10.65   b
 MT1000         10.64 0.01735 192    10.59    10.68   b

Results are averaged over the levels of: Algoritmo, Acelerador
Confidence level used: 0.95
Conf-level adjustment: sidak method for 4 estimates
P value adjustment: tukey method for comparing a family of 4 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.
```

En este caso vemos que MT5000 y MT500 pertenecen al mismo grupo y además son los que cuentan con el promedio más bajo. Como el MT500 consiste en un grupo de datos significativamente menor a MT5000 entonces este sería la mejor opción.

Seguidamente, hacemos el mismo análisis para el acelerador

```
library(multcomp)
marginal = lsmeans(model, pairwise ~ Acelerador, adjust="tukey")
```

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

```
 Acelerador lsmean      SE  df lower.CL upper.CL .group
 NA-NA       9.399 0.01735 192    9.355    9.443  a
 NA-SW       9.418 0.01735 192    9.375    9.462  a
 HW-NA      11.720 0.01735 192   11.676   11.763   b
 HW-SW      11.724 0.01735 192   11.681   11.768   b

Results are averaged over the levels of: Entrenamiento, Algoritmo
Confidence level used: 0.95
Conf-level adjustment: sidak method for 4 estimates
P value adjustment: tukey method for comparing a family of 4 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.
```

En este podemos observar que cuando no esta el acelerador de hardware tenemos valores menores y el hecho que este o no presente el acelerador de software no es importante ya que pertenece al mismo grupo, es decir, no hay diferencia significativa entre estos.

Gráficos finales

Creamos un dato llamado sum con promedios y SE (error estándar)

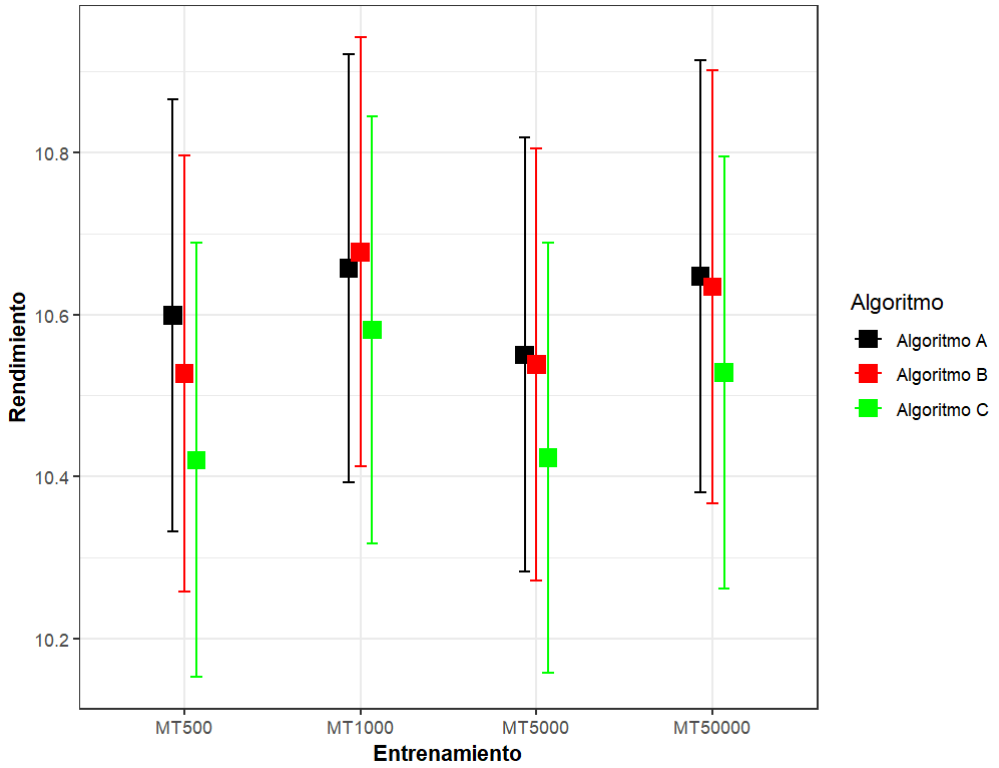
```
library(FSA)
Sum = Summarize(T_log ~ Entrenamiento + Algoritmo, data = Data, digits = 3)
```

Agregamos el SE (error estándar)

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

Ordenamos y graficamos

```
Sum$Entrenamiento = factor(Sum$Entrenamiento, levels =
unique(Sum$Entrenamiento))

library(ggplot2)
pd = position_dodge(.2)
ggplot(Sum, aes(x=Entrenamiento,
                y = mean,
                color = Algoritmo)) +
  geom_errorbar(aes(ymin=mean-se,
                    ymax=mean + se),
                width=.2, size=0.7, position=pd) +
  geom_point(shape=15, size=4, position = pd) +
  theme_bw() +
  theme(axis.title = element_text(face="bold")) +
  scale_colour_manual(values = c("black", "red", "green")) +
  ylab("Rendimiento")
```



Podemos observar que C se desempeña mejor en los datos de 5000 y 500 y por el argumento de que es mejor utilizar menos datos para entrenar para reducir el tiempo de entrenamiento, entonces este sería el elegido.

Seguidamente realizamos el gráfico para Algoritmo y Acelerador y podemos observar que los aceleradores no nos sirven.

```
library(FSA)
Sum = Summarize(T_log ~ Acelerador + Algoritmo, data = Data, digits = 3)


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

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

library(ggplot2)
pd = position_dodge(.2)
ggplot(Sum, aes(x=Acelerador,
                y = mean,
                color = Algoritmo)) +
  geom_errorbar(aes(ymin=mean-se,
                    ymax=mean + se),
                width=.2, size=0.7, position=pd) +
  geom_point(shape=15, size=4, position = pd) +
  theme_bw() +
  theme(axis.title = element_text(face="bold")) +
  scale_colour_manual(values = c("black", "red", "green")) +
  ylab("Rendimiento")
```

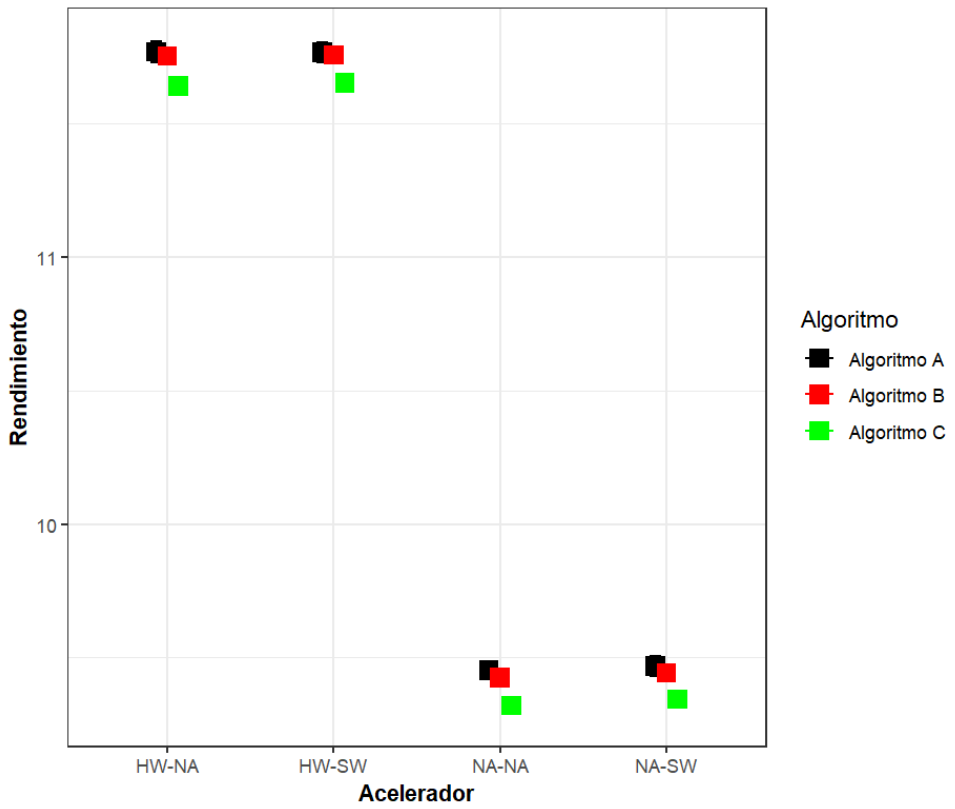


Gráfico de promedio transformados

```
library(FSA)
Sum = Summarize(T_log ~ Algoritmo, data = Data, digits = 3)
```

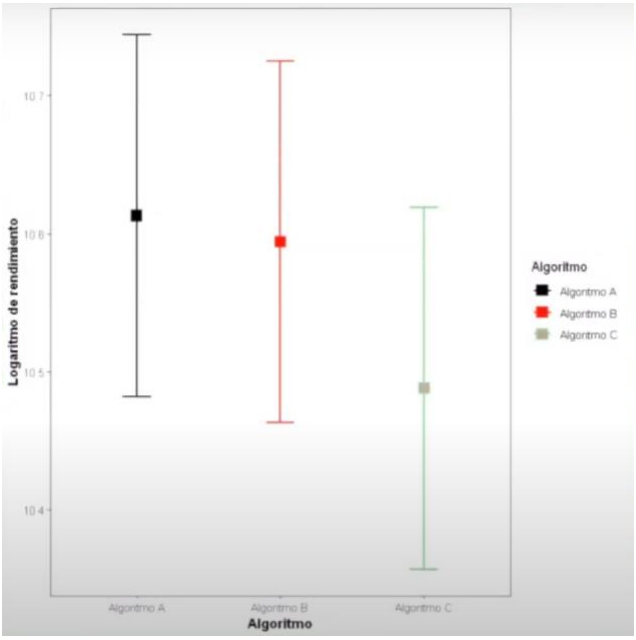
Agregamos el SE (error estándar)

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

Ordenamos y graficamos

```
Sum$Entrenamiento = factor(Sum$Entrenamiento, levels =
unique(Sum$Entrenamiento))

library(ggplot2)
pd = position_dodge(.2)

ggplot(Sum, aes(x = Entrenamiento, y = mean, color = Algoritmo)) +
        geom_errorbar(aes(ymin = mean-se, ymax = mean + se)
width = .2, size = 0.7, position = pd) +
geom_point(shape=15, size=4, position=pd) + theme_bw() +
theme(axis.title = element_text(face = "bold")) +
scale_colour_manual(values=c("black", "red", "green")) +
ylab("Logaritmo de rendimiento")
```

Con esto obtenemos el rendimiento promedio por algoritmo. Este gráfico se interpreta en conjunto con la tabla del CLD. Con ambas cosas podemos llegar a la conclusión que para todas las combinaciones que hicimos el algoritmo C es el que se porta mejor en promedio con las combinaciones de aceleradores y de datos de entrenamiento. También podemos decir que los algoritmos A y B no son estadísticamente diferentes.



Des-transformado promedios

Creamos un dato llamado sum con promedios y SE

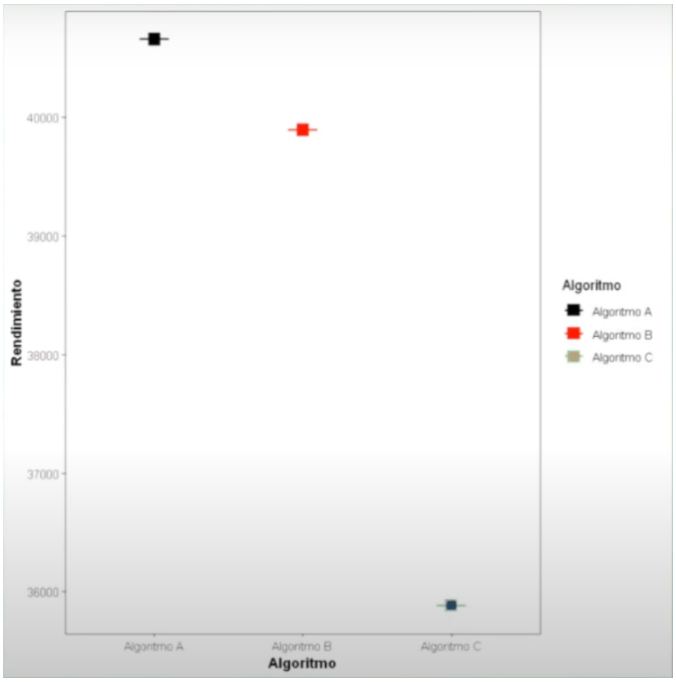
```
library(FSA)
Sum = Summarize(T_log ~ Entrenamiento + Algoritmo, data = Data, digits = 3)
```

Des-transformamos elevando a la e

```
Sum$mean = exp(Sum$mean)
Sum$sd = exp(Sum$sd)
```

Agregamos el SE (error estándar)

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



Al graficar los datos des-transformados podemos observar como aumentan las distancias entre los datos. Podemos ver que la distancia entre A y B aumentó, e incluso la distancia con C aumenta sustancialmente.

Nota: Al aplicar transformación de los datos, también minimizamos el error estándar por el mismo principio que disminuimos las distancias.

¿Por qué podemos llegar a conclusiones utilizando la transformación de datos? Al realizar la transformación estamos haciendo la prueba más difícil y aún así los datos pasan mi prueba, eso quiere decir que los datos si son estadísticamente diferentes.

Reportando resultados con transformación de datos

- Si el análisis se hizo con los datos transformados, presentar los datos sin transformar es incorrecto
- Podemos des-transformar varianza, promedios y presentar eso (algunos casos)
- Truco: los datos des-transformados cuentan la misma historia?
- Si ese no es el caso, es mejor los resultados con los datos transformados

El ejemplo que vimos NO cuentan la misma historia, por que en los destransformados vemos como que A y B si son distintos y además vemos que C está muchísimo más distanciado de A y B que como se observa en el de datos transformados

Presentando resultados

“El ANOVA se realizó con el logaritmo de los datos para cumplir con los supuestos”

“Toda la información que se presenta fue generada y procesada con la transformación al logaritmo de la variable de respuesta”

“Todos los promedios presentados en las figuras son los resultados des-transformando obtenidos en el análisis”

Aleatorización y Replicaciones

- Buena ciencia tiene réplicas, mala ciencia no y es solo una anécdota
- Para cada nivel de los factores estudiados, el experimento debe repetirse muchas veces
- Error experimental: variaciones generadas por factores incontrolables o no considerados
- Debemos poder estimar este error. Por ejemplo, en el proyecto 1 la curva de la distribución normal no era igual en cada repetición.
- Se debe estimar la media de la variable de respuesta con esa combinación de los niveles de los factores
- Entre más réplicas resultados más robustos
- Entre más réplicas, experimentos más caro

¿Cuántas réplicas son suficientes?

- Analice el contexto, porque va a depender este
- La cantidad necesaria que nos permite estimar la varianza. Mínimo 3
- En computación entre 5 y 15 es como un rango aceptable
- ¿Cómo saber entonces si 5 réplicas es mejor que 15 para estimar la varianza?

    - Realizamos un experimento previo para uno de los escenarios
    - Si sacamos la varianza para 20 y 5 datos y los intervalos de confianza nos dan básicamente iguales, esto es un buen indicador para decir que 5 es suficiente.

Aleatoriedad

- Imprescindible en el diseño de experimentos
- Revolvemos - entremezclamos - barajamos el material del experimento
- El orden del experimento debe ser aleatorio
- Las observaciones deben ser variables aleatorias distribuidas de manera independiente
- Al aleatorizar el orden de las réplicas distribuimos el efecto de los factores externos (ruido) no considerados en el experimento

Consideraciones éticas

Definiciones

Moral

Conjunto de normas y principios que se basan en la cultura y costumbres de determinado grupo social. Por ejemplo el divorcio.

Ética

Estudio y reflexión sobre la moral

Experimento

Buscar “animal experiments” en google imágenes y ver varias imágenes.

Preguntas del experimento cualitativo

¿Cómo se sintió? ¿Es eso correcto o ético? ¿Cuáles cuestionamientos vienen a su mente? ¿Es algo que apoya? ¿Qué se puede hacer?

Se tiene dos posturas: a favor y en contra de la experimentación en animales

Argumento en contra

- Experimentar en animales es inaceptable siempre porque
- Causa sufrimiento a estos
- Los beneficios para los humanos no están probados
- Cualquier beneficio obtenido a través de las pruebas en animales, se puede obtener de otras maneras

Daño vs Beneficio

A favor

Los experimentos en animales dan tantos beneficios que es totalmente aceptable lastimar a unos cuantos animales

En contra

El sufrimiento causado y el número de animales involucrados es tan alto que los beneficios para la humanidad no dan una justificación moral

Fármacos seguros para el ser humano

- Los experimentos de fármacos en animales no son utilizados para probar que los fármacos son seguros para los seres humanos
- Se utilizan para decidir si un fármaco debe ser probado en personas o no
- Si un fármaco pasa las pruebas en animales, se prueba en un pequeñísimo grupo de personas
- Eso antes de hacer pruebas a gran escala

Análisis de Caso

Se tienen 4 fármacos para curar el sida:

- Fármaco A mató a todas las ratas, ratones y perros
- Fármaco B mató todos los perros y ratas
- Fármaco C mató a todos los ratones y ratas
- Fármaco D se le dio a todos los animales hasta dosis muy altas y no causó efectos negativos

¿Cuál probaríamos en voluntarios jóvenes?
Tenemos dos posibles respuestas
Fármaco D y ninguno, porque de igual manera este puede ser tóxico

Fármaco D debe darse en condiciones controladas y reguladas, en una dosis muy baja a humanos voluntarios