Resumen de la clase de semana 10 (20/04/2023)
Curso: Diseño de experimentos

Estudiante: Gabriel Gutiérrez Arguedas

Índice

---

1. Presentaciones de papers

La clase inició con las presentaciones por parte de los estudiantes de los siguientes papers:

- Key Steps in Developing a Cognitive Vaccine against Traumatic Flashbacks: Visuospatial Tetris versus Verbal Pub Quiz. Link a la presentación: `https://youtu.be/BdAG4TqgyPc`
- One Rule to Grow Them All: A General Theory of Neuronal Branching and Its Practical Application. Link a la presentación: `https://youtu.be/xG86iV5ntQw`

2. Introducción a la clase

La clase inicia con una recapitulación de las tareas cortas asignadas en semana 9, que trataban el tema de diseño de bloques y que se se concluyó en que sí había interacción entre los factores al realizar la modificación al modelo lineal en el script de R, y que por lo tanto no se debió analizar por bloques. Para corregir esto, se necesita el anova multifactorial.

Por otra parte, se hace mención a la importancia de cómo no se deben realizar las tareas. Se da el ejemplo del área de las ingenierías electrónica y eléctrica cuando un experimento o tarea sale mal, se puede ocasionar un fallo físico, e inclusivo peligroso. Se contrasta con los fallos en las ciencias de la computación, de que un fallo resulta en errores de compilación y en esa línea. La idea es dar a entender que en ocasiones las cosas se pueden estar realizando de manera errónea y no percatarse.

3. Importancia de la interacción en bloques y anova

Está directamente relacionado a los errores que se pueden cometer durante un análisis. Se retoma el ejercicio de las tareas cortas; se tenía tres algoritmos, dos computadoras que se consideraron como factores de bloques, se realizó el análisis monofactorial con factores de bloque y resultó en que los algoritmos no eran distintos y las computadoras no influían en el desempeño. Esto es un enfoque errado, ya que la interacción entre los factores es importante en este caso.

Una interacción de factores es importante según el contexto; la idea era usar la computadora como factor de bloque en el caso de que hubieran diferencias físicas, y que estas fueran consideradas en el resultado del algoritmo. Se retoma el ejemplo de los algoritmos y computadoras y cómo los algoritmos pueden ejecutarse mejor en una computadora u otra. El análisis de la interacción permite decir si un algoritmo corre mejor en una computadora o la otra.

4. Cuadros latinos

- Se utilizan cuando hay dos factores de bloque.
  - Entiéndase cuando tenemos dos fuentes de variabilidad indeseada
- Debe cumplirse el supuesto de que no hay interacción entre los factores
  - De lo contrario el error se infla

La idea es tener un experimento más complejo, en el que los factores de bloques generan efecto en la variable de respuesta. No es lo que se quiere estudiar, pero se buscan resultados que consideren el efecto de los factores de bloques. El supuesto de que no hay interacción entre factores es de suma importancia, si no se cumple, se debe realizar un análisis multifactoral en lugar de análisis de bloque. Si hay interacción entre los factores, el error se infla.

4.1. Ejemplo

Consideremos una fábrica con 4 operadores y 4 máquinas. Las columnas son los operadores y las filas las máquinas. Para garantizar la aletorización debemos:

- Asignar aleatoriamente los operadores a las filas
- Asignar aleatoriamente las máquinas a las columnas
- Asignar aleatoriamente cada uno de los tratamientos a las letras

Si la máquina y el operador son los factores de bloque, utilizar este diseño permite eliminar del análisis la variación debida a ellos.

Se busca cual proceso se desempeña mejor. Graficando se obtiene el cuadro 1.

		Máquinas			
		2	1	3	4
Operadores	3	A	B	C	D
	4	B	C	D	A
	1	C	D	A	B
	2	D	A	B	C

Cuadro 1: Caudro latino para el ejemplo de las máquinas y operarios.

Se quiere validar los procesos A, B, C y D, pero los operadores pueden generar efecto sobre los resultados. Se ejemplifica con el caso de que el operador 3 acaba de tener un bebé y este lo despierta durante la madrugada. El desempeño de este operador se puede ver afectado por la falta de sueño, y por lo tanto cambiar los resultados. Mientas, el operador 4 podría no tener este incoveniente, por ejemplo.

Asismismo, las máquinas también podrían influir. Aunque se supone que las máquinas son iguales, el mismo modelo, si una máquina tiene 10 años de uso y otra solamente uno, esto va a llegar a influir. Los aspectos del estado físico de la máquina influyen.

Lo que se busca con este tipo de diseño es ecualizar, considerar o eliminar la variación que un operador en específico y una máquina en específico dan. En este caso los procesos se asignaron de forma aleatoria a las letras.

El diseño de experimentos por cuadros latinos cumplen una caracterísitica; lo que se debe cumplir es **que no se repita letra en fila y que no se repita letra en columna**. Sigue el patrón del *logical shift left circular*.

Nota: La operación lógica del *shift left* consiste en simplemente mueve un conjunto de bits a la izquierda. La operación lógica desplaza un bit a la izquierda (MSB -Most Significant Bit-) e introduce un cero por la derecha (LSB -Least Significant Bit-). Esta operación binaria equivale a una multiplicación por un entero de valor 2. La figura 1 es una ejemplo de esta operación binaria.

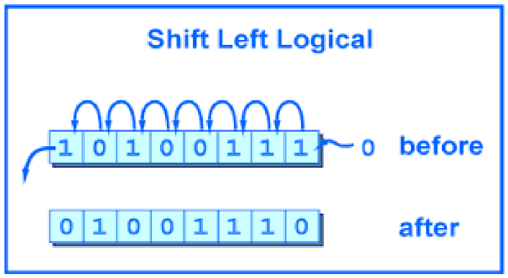


Figura 1: Ejemplo de la operación lógiva *shift left*.

4.2. Análisis

- Una observación por tratamiento en cada fila y columna.
  - Las columnas y filas se incluyen para disminuir la variación debida al error.
- La aleatorización permite inferir correctamente sobre esas pruebas.
  - Causa que el efecto del error se distribuya.
- Hacer el análisis en R es muy similar al análisis del ejemplo anterior, solo que con un factor más.
  - Es solamente agregar el factor con el operador + en la creación del modelo lineal.
  - El cuadro latino es el **cómo se van a recabar los datos**, aquí está la importancia y no en el cambio en la sintaxis de una línea en el script.
  - Es importante porque no se pueden probar todas las combinaciones; en el ejemplo hay 4 operadores, hay 4 máquinas y 4 procesos, por lo que se tiene $4 * 4 * 4$, un factorial que da 64 combinaciones. Con el cuadro latino se realizan 16 experimentos; es un experimento muy eficiente que permite distribuir el error de los factores bloques de manera eficiente. Con 16 experimentos, se puede llegar a conclusiones muy similires a los que producen el conjunto total de combinaciones.
- Notemos que es un **diseño incompleto** (no se prueba todas las posibles combinaciones).

5. Cuadros greco-latinos

Consiste en un cuadro latino como el de la sección 4.1 pero con tres factores de bloque. Volviendo al ejemplo de la sección 4.1:

- Consideremos que ahora tenemos 3 factores de bloque.
  - Máquina, operador y día en que se hace la observación.
  - Esto se considera si en un día no es posible obtener todas las observaciones.
  - Debemos tener razones para pensar que no todos los días son iguales.
- Podríamos intentar completar el experimento en un día y replicarlo 4 veces (64 observaciones).

Si la máquina, operador y días son los factores de bloque, utilizar este diseño permite eliminar del análisis la variación debido a ellos. Notar la eficiencia de t^2 observaciones.

Se busca desarrollar un experimento que permita distribuir el error entre todas las observaciones, que contemple la variación que el factor de bloque genera y que sea económico de realizar (reducir el tamaño del experimento).

No todos los días son iguales; por ejemplo, los lunes se puede realizar el experimento a las 2 a.m. y los martes a la hora de almuerzo. Pueden haber bastantes sospechas que el día puede influir en los resultados del experimento, por eso se agrega como factor bloque.

5.1. Volviendo al ejemplo

Consideremos una fábrica con 4 operadores y 4 máquinas. Las columnas son los operadores y las filas las máquinas. Para garantizar la aleatorización debemos:

- Asignar aleatoriamente los operadores a las filas.
- Asignar aleatoriamente las máquinas a las columnas.
- Asignar aleatoriamente cada uno de los tratamientos a las letras.
- Asignar aleatoriamente cada uno de los días a las letras griegas.

Si la máquina, operador y día son factores de bloque, utilizar este diseño permite eliminar del análisis la variación debido a ellos. Notar la eficiencia de t^2 observaciones. El resultado se muestra en el cuadro 2.

		Máquinas			
		2	1	3	4
Operadores	3	Aα	Bβ	Cγ	Dδ
	4	Bδ	Cγ	Dβ	Aα
	1	Cβ	Dα	Aδ	Bγ
	2	Dγ	Aδ	Bα	Cβ

Cuadro 2: Cuadro greco-latino para el ejemplo de las máquinas, operarios y día del experimento.

Nuevamente, se debe garantizar que las letras griegas no se repitan. Se puede seguir el procedimeinto del *shift left* similar a como se muestra en la figura 3. Una forma de leer el cuadro, por ejemplo, sería que el operador 3 va en la máquina 2 el lunes a realizar el proceso A. Este diseño es valioso porque es eficiente, lo que es eficiente es barato y se ahorra en costos (recursos económicos y tiempo). Si se quiere evaluar 4 procesos (t en este caso) se necesitan hacer t^2 observaciones.

Hay una limitante; ¿qué pasa si los factores de bloque no alcanzan para el cuadro? Su tamaño, el tamaño de cuadro greco-latino, lo define el factor bajo estudio. Por ejemplo, si el factor de estudio son los procesos A, B, C, D y E, el tamaño del cuadro es 25. Los factores bloque pueden faltar o sobrar; si faltan, aleatoriamente se asignan a los espacios para completar el cuadro. Si sobran, aleatoriamente se deciden cuales se van a considerar.

6. Otros diseños

6.1. Diseño cruzados

Es cuando se tiene la misma unidad experimental para múltiples experimentos; es muy común en el área de la medicina o para probar un fármaco. En la figura 2 se observa un ejemplo de prueba de medicamento:

1. Primer tratamiento: Se tiene una persona a la cual se le aplica el tratamiento.
2. Observación: Se observa y toma nota sobre el efecto del fármaco.
3. Período de lavado: Lapso de tiempo en el cual no se expone al individuo bajo estudio a ningún medicamento, y que más bien su cuerpo lo proceso y saque el fármaco de su organismo.
4. Segundo tratamiento: Una vez limpio, se le aplica al paciente de nuevo el tratamiento.
5. Observación: Observación para tomar nota sobre la respuesta del individuo al fármaco en estudio.

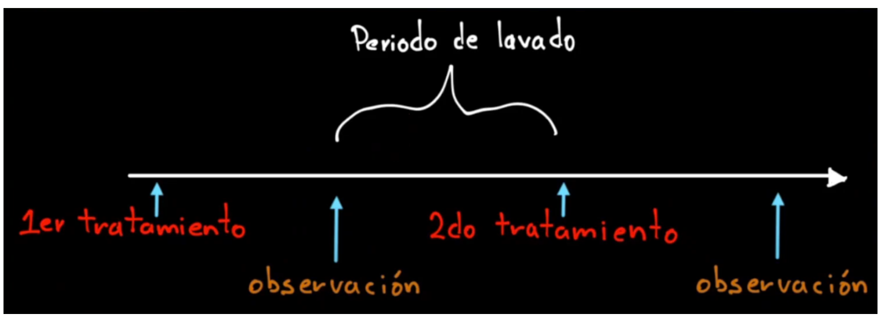


Figura 2: Ejemplo de clase de aplicar el diseño cruzado.

Si este procedimiento se realiza sobre una sola persona, no es un experimento sino una anécdota. Así no se pueden derivar conclusiones, se debe aplicar un experimento de esta clase en distintas personas. El cuadro 3 muestra cómo se aplica el experimento en varias personas y el orden en el que se le van a aplicar los fármacos se va a alternar. Se le asigna el tratamiento de manera cruzada.

		Máquinas									
		1	2	3	4	5	6	7	8	9	10
Operadores	1	A	B	A	B	A	B	A	B	A	B
	2	B	A	B	A	B	A	B	A	B	A

Cuadro 3: Ejemplo de diseño cruzado.

6.2. Diseños de bloques incompletos

Se dan cuando por alguna razón no se puede completar el cuadro latino o greco-latino. Se utiliza cuando hay alguna razón práctica o de incompatibilidad que no permite ejecutar el experimento. Por ejemplo, según el cuadro 4 el tratamiento 1 no es compatible con la máquina 3, físicamente no se puede correr ahí y por lo tanto ese dato no se puede recabar.

		Bloques			
		1	2	3	4
Tratamiento	1	1	1	0	1
	2	0	1	1	1
	3	2	1	1	0
	4	1	0	1	1

Cuadro 4: Ejemplo de diseño de experimento de bloques incompletos.

7. Diseños multifactoriales

La recomendación es plantear los experimentos de esta clase. En esta clase de diseños:

- Se analizan varios factores a la vez.
  - Se estudian los efectos de estos factores en la variable de respuesta.
- Es de lo más común su utilización en experimentos.
- Aparte de los factores, también se analiza su interacción.
- Se realiza análisis post-hoc si se encuentran diferencias significativas en los promedios de los grupos (depende del resultado del anova).

7.1. Análisis post-hoc

- No se aplica si no hay diferencia significativa en los factores o las observaciones.
- Cuando uno o más de los factores principales son significativos, pero la interacción no, sólo se debe hacer el análisis en los factores principales.
- Cuando el efecto de la interacción es estadísiticamente significativo, se aplica análisis post-hoc a la interacción.
- La recomendación es hacerle análisis post-hoc a todo lo que el anova tire como significativo.

7.2. Ejemplo de anova multifactorial

7.2.1. Escenario tabulado

Volviendo al experimento para evaluar algoritmos en la red neuronal, se quiere evaluar el desempeño al entrenar con dos conjuntos de datos distintos. En el cuadro 5 se muestra los 3 algoritmos a comparar y los 4 conjuntos de datos. Una red neuronal generaliza mejor el problema y se desempeña mejor, generalmente. Se busca alcanzar una métrica establecida con la menor cantidad de datos posibles, para tratar de ahorrar recursos. Algunos conjuntos de datos ya son suficientes y más de eso ya es un desperdicio. La idea es alcanzar la métrica de la manera más conveniente posible.

	Factores	
	Método de entrenamiento	Conjunto de datos
Niveles	Algoritmo A Algoritmo B Algortimo C	MT500 MT1000 MT5000 MT50000

Cuadro 5: Factores y niveles del ejemplo de anova multifactorial.

7.2.2. Anova monofactorial en R

A continuación se muestran los segmentos de código necesarios para ejecutar un anova multifactorial:

1. Importar las librerías:

```
#Instalando paquetes requeridos

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

2. Cargar los datos en R:

```
#Cargando datos

Datos = ("
Algoritmo        Entrenamiento       Rendimiento
'Algoritmo A'         MT500             12000
'Algoritmo A'         MT500             14005
'Algoritmo A'         MT500             13508
'Algoritmo A'         MT500              9503
'Algoritmo A'         MT500             14004
'Algoritmo A'         MT1000             11502
'Algoritmo A'         MT1000             13006
'Algoritmo A'         MT1000             13252
'Algoritmo A'         MT1000             14253
'Algoritmo A'         MT1000             15003
'Algoritmo A'         MT5000            12504
'Algoritmo A'         MT5000            11504
'Algoritmo A'         MT5000             9500
'Algoritmo A'         MT5000            11506
'Algoritmo A'         MT5000            16000
'Algoritmo A'         MT50000            13008
'Algoritmo A'         MT50000            10506
'Algoritmo A'         MT50000            13005
'Algoritmo A'         MT50000            17002
'Algoritmo A'         MT50000            13008
'Algoritmo B'         MT500              11005
```

```
'Algoritmo B'          MT500          12007
'Algoritmo B'          MT500          12509
'Algoritmo B'          MT500          10504
'Algoritmo B'          MT500          12002
'Algoritmo B'          MT1000         12504
'Algoritmo B'          MT1000         13501
'Algoritmo B'          MT1000         13501
'Algoritmo B'          MT1000         13252
'Algoritmo B'          MT1000         15256
'Algoritmo B'          MT5000         12253
'Algoritmo B'          MT5000         11255
'Algoritmo B'          MT5000         10006
'Algoritmo B'          MT5000         11252
'Algoritmo B'          MT5000         14004
'Algoritmo B'          MT50000        12007
'Algoritmo B'          MT50000        11505
'Algoritmo B'          MT50000        14009
'Algoritmo B'          MT50000        15000
'Algoritmo B'          MT50000        12009
'Algoritmo C'          MT500           9000
'Algoritmo C'          MT500          11003
'Algoritmo C'          MT500          11505
'Algoritmo C'          MT500           9509
'Algoritmo C'          MT500          11003
'Algoritmo C'          MT1000         11508
'Algoritmo C'          MT1000         12508
'Algoritmo C'          MT1000         12506
'Algoritmo C'          MT1000         12254
'Algoritmo C'          MT1000         13253
'Algoritmo C'          MT5000         11255
'Algoritmo C'          MT5000         10257
'Algoritmo C'          MT5000          9500
'Algoritmo C'          MT5000          9255
'Algoritmo C'          MT5000         12009
'Algoritmo C'          MT50000        11000
'Algoritmo C'          MT50000         9509
")
```

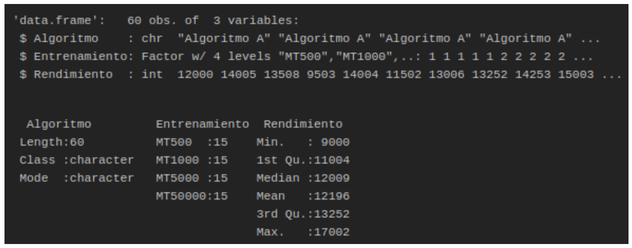
3. Leer la tabla para procesarla y ordenando los datos según fueron ingresados:

```
#Leer la tabla para procesarla

Data = read.table(textConnection(Datos), header = TRUE)

#Ordenando los datos segun los ingresamos

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

4. Verificar que todo esté bien y liberar la tabla (15 datos por conjunto de datos entrenamiento):

```
#Verificando que todo este bien y liberamos la tabla

library(psych)
```

```r
headTail(Data)
str(Data)
summary(Data)
rm(Datos)
```



```
'data.frame':   60 obs. of  3 variables:
 $ Algoritmo    : chr  "Algoritmo A" "Algoritmo A" "Algoritmo A" "Algoritmo A" ...
 $ Entrenamiento: Factor w/ 4 levels "MT500","MT1000",..: 1 1 1 1 1 2 2 2 2 2 ...
 $ Rendimiento  : int  12000 14005 13508 9503 14004 11502 13006 13252 14253 15003 ...

  Algoritmo          Entrenamiento  Rendimiento
 Length:60          MT500  :15     Min.   : 9000
 Class :character   MT1000 :15     1st Qu.:11004
 Mode  :character   MT5000 :15     Median :12009
                    MT50000:15     Mean   :12196
                                   3rd Qu.:13252
                                   Max.   :17002
```

Figura 3: Resultado en consola de verificar que todo esté bien y liberar la tabla.

5. Gráfico simple de interacción (concepto nuevo):

```r
#Grafico simple de interaccion

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" )
```

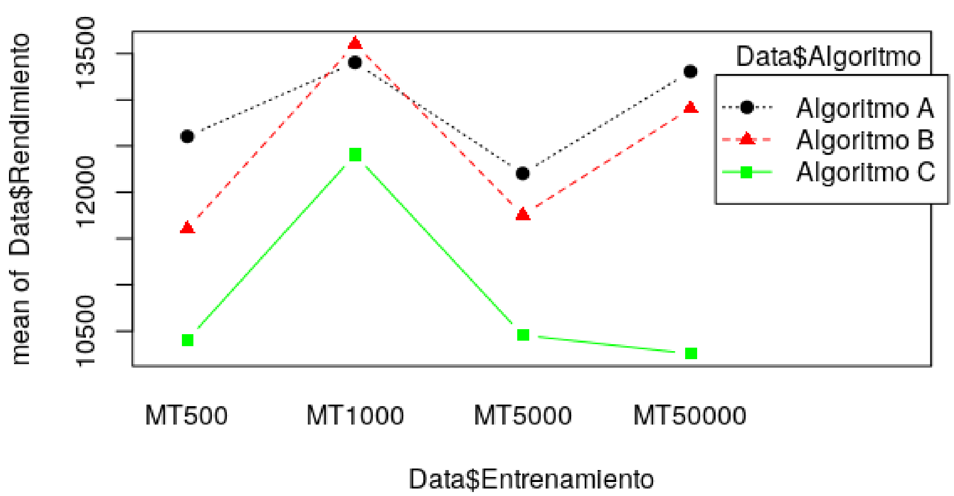


Figura 4: Gráfico simmple de interacción.

El gráfico simple de interacción se realiza antes del análisis de varianza, antes del modelo lineal, antes de evaluación de supuestos. Antes se hizo gráficos de cajas para tratar de intuir el resultado, ahora se tiene gráficos de interacción para ver cómo se comportan las interacciones. Vemos que el efecto del algoritmo que se utilizó no es consistente en todos los métodos de entrenamiento. Puede que haya interacciones significativas que debemos explorar. La imagen 4 se puede interpretar como que para el conjunto de datos MT500 el algoritmo C tuvo el mejor rendimiento (promedios), seguido por el algoritmo B y el algoritmo A fue el peor. Las interacciones se dan cuando un nivel de un factor a otro los puntos se invierten.

6. Análisis de varianza, construcción del modelo lineal y el anova

```
#Analisis de varianza

#Definiendo el modelo lineal

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

#Ejecutando el anova

library(car)

Anova(model, #Tipo 2 es el por defecto
      type ="II") #Suma de cuadrados
```

	Sum Sq	Df	F value	Pr(>F)
	<dbl>	<dbl>	<dbl>	<dbl>
Entrenamiento	30621741	3	4.2341389	0.009822338
Algoritmo	28982927	2	6.0113044	0.004679860
Entrenamiento:Algoritmo	2444325	6	0.1689912	0.983866401
Residuals	115713695	48	NA	NA

Figura 5: Resultado de ejecutar el anova.

Nota: El código utiliza la versión simplificada para combinar las diferentes interacciones de los factores. Si la manaera en la que se recaban los datos es balanceada, no importa el orden de las interacciones en el que se ingresen para el modelo (todos tienen las mismas observaciones). El problema si las observaciones son distintas; la recomendación es poner de primero el factor principal.

Los p-values de Entrenamiento y Algoritmo muestra que hay diferencia estadísticamente significativa. La interacción entre factores muestra lo contrario, por lo que no vale la pena aplicar análisis post-hoc para ellas.

7. Histograma de residuos

```
#Histograma de residuos (supuesto)

x = residuals(model)

library(rcompanion)
```

```
#Histograma de residuos

plotNormalHistogram(x)
```

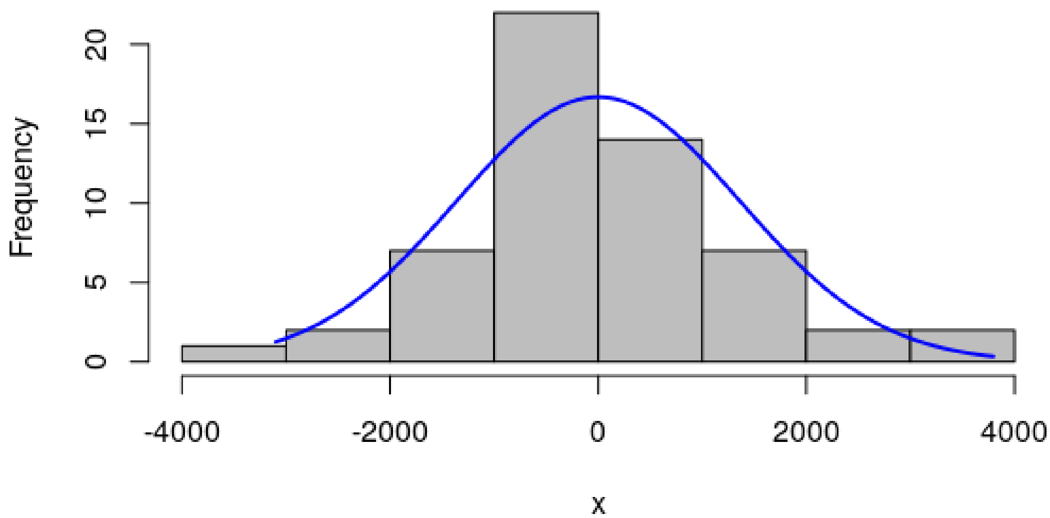


Figura 6: Histograma de residuos.

8. Varianza de residuos y mínimos cuadrados para Algoritmo

```
#Graficacion de la varianza de residuos
#Fitted value: Prediccion del modelo de la respuesta promedio cuando se
#ingresa un valor predicor, nivel de factor, o componentes en el modelo

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

plot(model)

#Analisis post-hoc para Algoritmo

#Minimos cuadrados para multiples comparaciones

library(multcompView)
library(lsmeans)

marginal = lsmeans(model, ~ Algoritmo)

#Ejecuta todas las comparaciones de pares
#para la variable Algoritmo

```

```
#Ajuste de Tukey minimiza error tipo 1

pairs(marginal,
      adjust = "sidak")
```

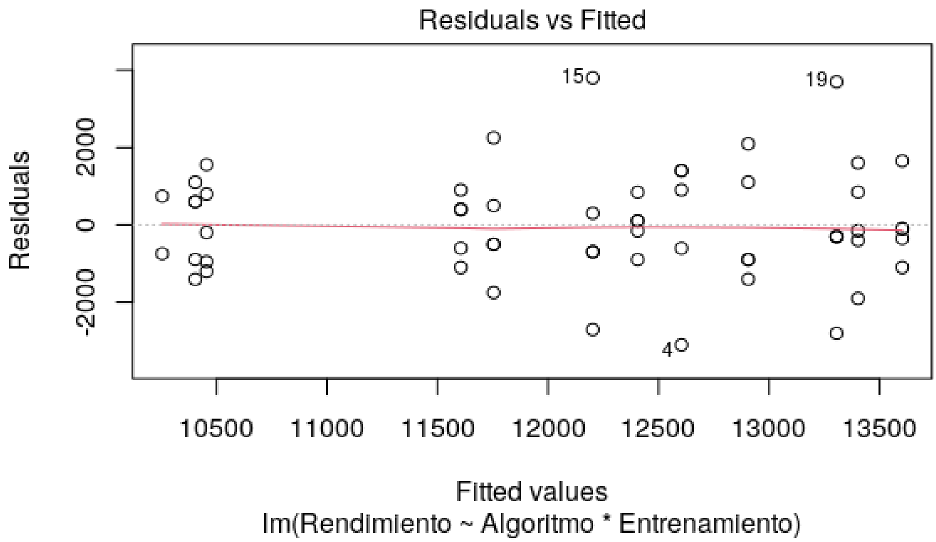


Figura 7: Varianza de los residuos.

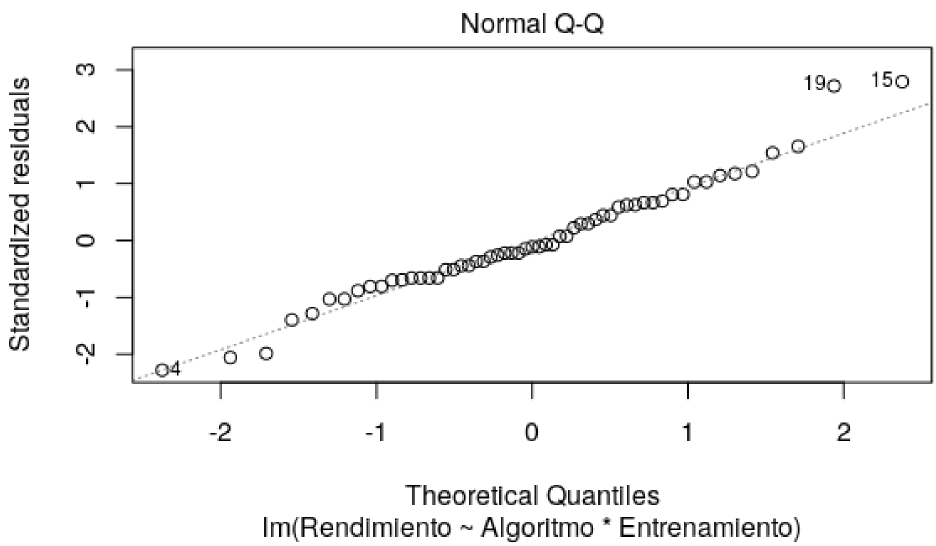


Figura 8: Gráfica Q-Q de los residuos.

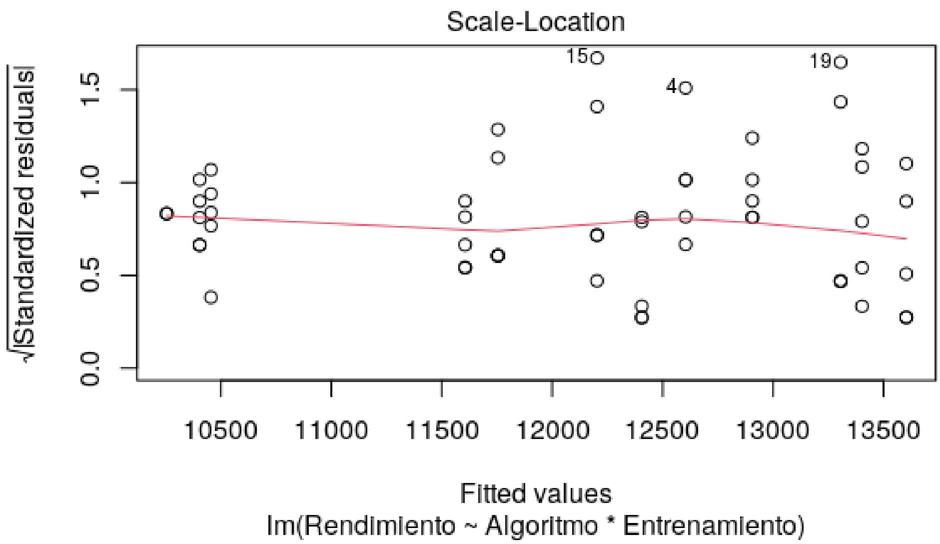


Figura 9: Varianza de los residuos.

9. Letras de separación entre grupos para algoritmo

```
#Letras de separacion entre grupos

library(multcomp)
CLD = cld(marginal,
          alpha = 0.05,
          Letters = letters,
          adjust = "sidak")

#CLD nos muestra despliegue por letras

print(CLD)
```

	Algoritmo	lsmean	SE	df	lower.CL	upper.CL	.group
	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
3	Algoritmo C	11242.45	347.1817	48	10383.55	12101.35	a
2	Algoritmo B	12467.05	347.1817	48	11608.15	13325.95	b
1	Algoritmo A	12878.95	347.1817	48	12020.05	13737.85	b

Figura 10: Letras de separación para Algoritmo.

10. Varianza de residuos y mínimos cuadrados para Entrenamiento

```
#Minimos cuadrados para multiples comparaciones

library(multcompView)
library(lsmeans)

```

```
marginal = lsmeans(model, ~ Entrenamiento)

#Ajuste de Tukey minimiza error tipo 1

pairs(marginal,
      adjust = "sidak")
```

```
 contrast          estimate  SE df t.ratio p.value
 MT500 - MT1000     -1599.5 567 48  -2.821  0.0339
 MT500 - MT5000        67.1 567 48   0.118  0.9994
 MT500 - MT50000    -1101.1 567 48  -1.942  0.2245
 MT1000 - MT5000     1666.6 567 48   2.940  0.0251
 MT1000 - MT50000     498.4 567 48   0.879  0.8156
 MT5000 - MT50000   -1168.2 567 48  -2.061  0.1808

Results are averaged over the levels of: Algoritmo
P value adjustment: tukey method for comparing a family of 4 estimates
```

Figura 11: Mínimos cuadrados para entrenamiento.

11. Letras de separación entre grupos para entrenamiento

```
#Letras de separacion entre grupos

library(multcomp)
CLD = cld(marginal,
          alpha = 0.05,
          Letters = letters,
          adjust = "sidak")

#CLD nos muestra despliegue por letras

print(CLD)
```

	Entrenamiento	lsmean	SE	df	lower.CL	upper.CL	.group
	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
3	MT5000	11470.67	400.8908	48	10433.24	12508.09	a
1	MT500	11537.80	400.8908	48	10500.38	12575.22	a
4	MT50000	12638.87	400.8908	48	11601.44	13676.29	ab
2	MT1000	13137.27	400.8908	48	12099.84	14174.69	b

Figura 12: Letras de separación para Entrenamiento.

El conjunto de datos de MT50000 pertenece a los dos grupos, pero esto no implica que a y b sean iguales.

12. Variable con promedios y error estándar (para los bigotes)

```
#Creando un dato llamado sum con promedios y se (standard error)

library(FSA)

Sum = Summarize(Rendimiento ~ Entrenamiento + Algoritmo,
```

```
                    data = Data,
                    digits = 3)

#Agregando el se

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

print(Sum)

#Ordenando el grafico

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")
```

Entrenamiento	Algoritmo	n	mean	sd	min	Q1	median	Q3	max	se
<fct>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
MT500	Algoritmo A	5	12604.0	1918.400	9503	12000	13508	14004	14005	858
MT1000	Algoritmo A	5	13403.2	1330.104	11502	13006	13252	14253	15003	595
MT5000	Algoritmo A	5	12202.8	2386.861	9500	11504	11506	12504	16000	1070
MT50000	Algoritmo A	5	13305.8	2332.843	10506	13005	13008	13008	17002	1040
MT500	Algoritmo B	5	11605.4	822.896	10504	11005	12002	12007	12509	368
MT1000	Algoritmo B	5	13602.8	1010.473	12504	13252	13501	13501	15256	452
MT5000	Algoritmo B	5	11754.0	1488.960	10006	11252	11255	12253	14004	666
MT50000	Algoritmo B	5	12906.0	1514.683	11505	12007	12009	14009	15000	677
MT500	Algoritmo C	5	10404.0	1084.210	9000	9509	11003	11003	11505	485
MT1000	Algoritmo C	5	12405.8	625.963	11508	12254	12506	12508	13253	280
MT5000	Algoritmo C	5	10455.2	1166.849	9255	9500	10257	11255	12009	522
MT50000	Algoritmo C	5	11704.8	1789.237	9509	11000	11001	13009	14005	800

Figura 13: Resumen de la variable.

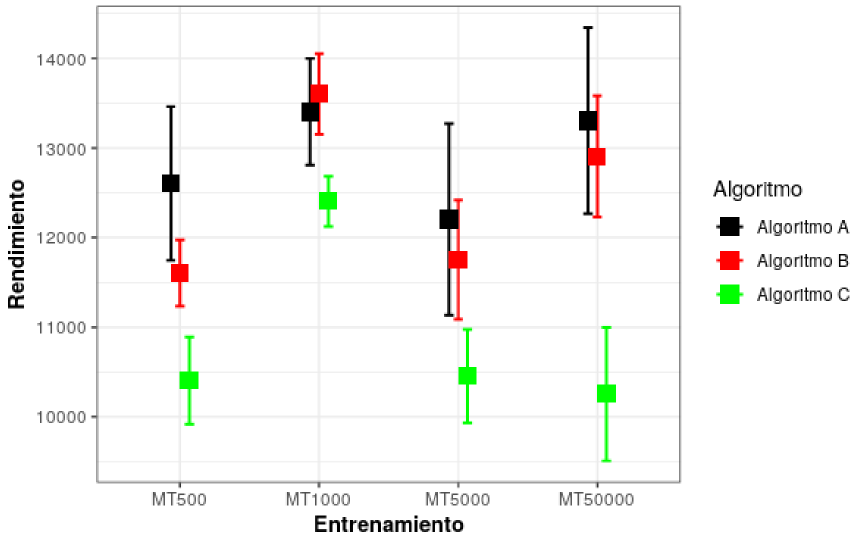


Figura 14: Comparación por letras para algoritmo y entrenamiento, usando el error estándar para los bigotes.

En la figura 14 el eje x es entrenamiento y color es algoritmo; para un mismo x podemos tener varios algoritmo. Como se busca compara rendimiento (tiempo), menos es mejor. También, ahora los bigotes son el error estándar. El error estándar (relacionado con la desviación estándar) genera un intervalo de confianza menor que usando la desviación estándar, la información que da es que el promedio real de la población puede estar en dicho intervalo. Los intervalos de confianza contiene el error estándar, pero hay que recordar que tiene un porcentaje de error asociado.

Se pueden derivar conclusiones como que C, en cuanto a rendimiento, es mejor con cualquier conjunto de datos. También se puede concluir que el conjunto de datos MT1000 emperora el desempeño de los algoritmos por ejemplo.

8. Estrategias de experimentación

8.1. Ejemplo: ¿Qué factores influyen en el desempeño de mini-golf

Sin considerar habilidad, ¿qué nos impacta el desempeño?

- Palo propio o prestado.
- Oportunidad de repetir el tiro.
- Clima.
- Obstáculos en el campo.
- Bola propia o prestada.
- Uso de guantes.

8.1.1. Simplificando el experimento

Eliminamos factores que nos impactan.

- Jugamos en un campo de mini-golf cerrado.
- Usamos las bolas provistas por el dueño.
- Utilizamos un sólo obstáculo como presente y no presente.
- ¿Se puede hacer?
  - Sí se puede, pero todas estas decisiones se tienen que documentar. A veces por razones prácticas hay que simplificar el experimento.
- Criterio de experiencia, instuición, pruebas informales.
  - A veces hay que usar la intuición para concluir a partir de datos incompletos.
- Economizamos recursos (tiempo, dinero)

8.1.2. Según los criterios escogidos

Es **importante** anotar y justificar las decisiones tomadas. Se debe estar preparado a reconsiderar y reajustar el experimento.

8.1.3. ¿Cómo vamos a desarrollar el experimento?

Aspectos que se deben considerar en cuenta:

- Dinero.
- Tiempo.
- Aspectos éticos y legales.
  - Animales, personas, etc. Posible causa de daños.
- ¿Es el experimento destructivo?
  - Pruebas de choques de autos, disparo de munición.
- Disponibilidad de recursos.
- Precisión e importancia.
  - A veces un pequeño experimento más sencillo puede servir de guía para plantear un mejor experimento.

8.2. Mejor adivinanza

Continuando con el ejemplo de la sección 8.1:

- De los factores que tenemos a investigar, tomamos una combinación arbitraria:
  1. Sin guantes.
  2. Con palo propio.
  3. Sin obstáculo.
  4. Con posibilidad de repetición.
- Se hacen 15 tiros, de los cuales se anotan 5.
- Se nota que las manos sudan y el palo de golf resbala (observación).

La observación ayuda a tomar una decisión para la siguiente iteración del experimento.

8.2.1. Mejor adivinanza (segundo intento)

- Se reajusta el experimento usando guantes:
  1. Con guantes.
  2. Con palo propio.
  3. Sin obstáculo.
  4. Con posibilidad de repetición.
- De los 15 tiros ahora se anotan 8.
- Se crea una nueva combinación cambiando uno (a lo sumo dos) de los factores y se repite.
- Son ampliamente utilizados en la ingeniería, medicina y ciencias.

Se nota que hay una mejoría. Estos métodos no son fuertes estadísiticamente pero funcionan como experimentos previos para un experimento más formal.

8.2.2. Ventajas de mejor adivinanza

- Son experimentos fáciles de entender.
- Son relativamente baratos.
- Pueden descubrir tendencias.
- Últiles para filtros preliminares.
- Se puede hacer uso de conocimiento previo.
- Puede ser la única opción.

8.2.3. Desventajas

- Llegar a un callejón sin salida.
- La estrategia de las adivinanzas podría no llegar a nada.
- Es posible que no encontremos el óptimo.
- Los resultados pueden tener poca validez estadística.

8.3. Un factor a la vez

- Establecemos una combinación inicial de los factores.
- Este es llamado la línea base.
- Estudiamos el efecto de un factor a la vez.
- Variamos este factor en todos los rangos aceptables.
- Los demás factores los mantenemos en línea base.
- La cantidad de experimentos depende de cuanto variamos cada factor

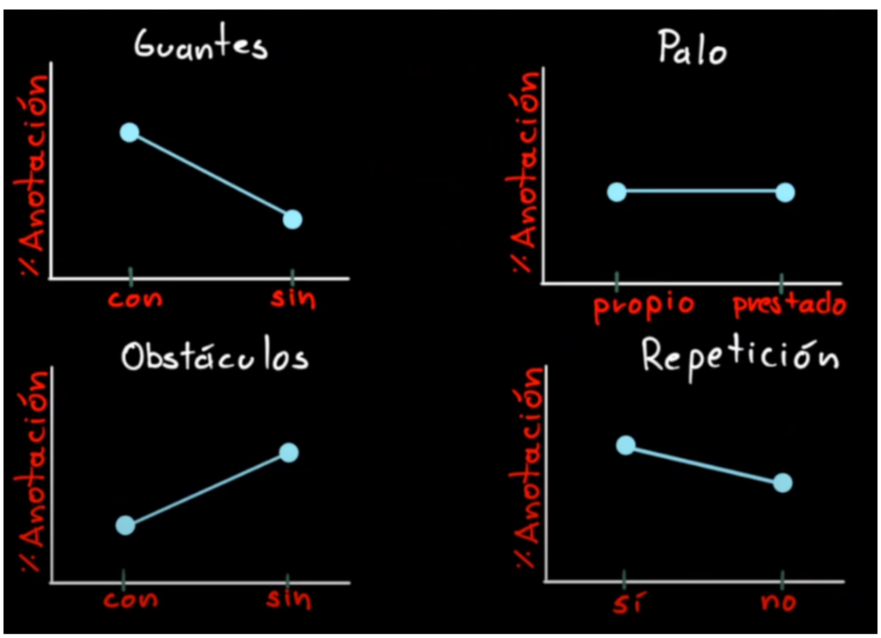


Figura 15: Aplicación del método un factor a la vez para el ejemplo de la sección 8.1.

La figura 15 muestra que en una de las iteraciones había una línea base y en una de las iteraciones se variaron los guantes. Luego se verifica con palo propio y prestado, lo mismo con los obstáculos y la oportunidad de repetir el tiro. Sólo estos factores se variarion de uno en uno por experimento, todos los demás se quedan igual.

8.3.1. Ventajas

- Fácil de entender, definir, interpretar, explicar.
- Podemos obtener resultados valiosos.
- Es relativamente económico.

8.3.2. Desventajas

- Los resultados no son estadísticamente válidos.
- No se puede estudiar interacciones entre factores.
- Es muy sensible a la definición de línea base.
- Posiblemente más caro que “mejor adivinanza”.