CAPÍTULO 3

Análisis de variables continuas de experimentos en arreglos factoriales

José Bruno Malaquias, Tatiane Caroline Grella, Jéssica Karina da Silva Pachú, Milton Fernando Cabezas Guerrero

Resumen

En esta sección, estamos exponiendo un ejemplo de análisis de base de datos hipotética que se delineó en bloques aleatorios y con una estructura de tratamiento en arreglo factorial 2 x 2. Dado que la variable hipotética es continua, nuevamente exploramos algunas líneas de comando que son esenciales para la normalidad y homogeneidad de pruebas de varianza, para un análisis más detallado de las pruebas de varianza y de comparación de medias. Presentamos todas las líneas de comando con un paso a paso de forma comentada y detallada. Usaremos los siguientes paquetes: **readxl**, **MASS** e **ExpDes.pt**. Los comandos a ejecutar en R se muestran en color azul.

Palabras claves: análisis factorial; ANOVA; dos factores; variable continua; desdoblar; interacción.

1. Importación de base de datos

Después de descargar los archivos utilizados en este capítulo haciendo **clic aquí**, es necesario verificar en qué lugar de la computadora (directorio) se encuentra el archivo a analizar.

Para eso, usemos el comando **getwd()**, que verificará en qué directorio estás trabajando: **getwd()**

Si necesita cambiar el directorio, simplemente siga los pasos:

Session -> Set working Directory -> Choose Directory

Luego de elegir un directorio es posible ver qué archivos existen en él, para eso usamos la función: **list.files()** la cual mostrará la lista de archivos dentro de su directorio.

Después de elegir el directorio, leamos el archivo que se analizará.

Para leer el archivo de Excel, necesitaremos el paquete **readxl** [1].

Una forma elegante de cargar el paquete es usar la línea de comando (esta línea también funciona si el paquete no está instalado, ya que se instalará automáticamente):

```
if (!require("readxl")) install.packages("readxl"); require(readxl)
```

El signo de exclamación indica denegación, por lo que la línea de comando anterior se traduce como: *"si el paquete requerido (readxl) no está instalado, instale el paquete, luego cárguelo"*

Exploremos ahora otra forma de leer la base de datos, pero usando el mismo paquete. Primero, pongamos nuestra ruta (**path**) para leer el archivo, veamos que asigna mi **path** con el nombre *MyFolderANDFile* (cada uno puede asignar el nombre que quiera):

```
MinhaPastaEArquivo <- "C:/Users/Bruno/Google Drive/Grupo de Discussão/Grupo - Linguagem R/Material-01082020/BD1-Anovatwoway.xlsx"
```

Vea qué parte del **path** es personal: **"C:/Users/Bruno/Google Drive/Grupo de Discussão/Grupo - Linguagem R/Material-01082020**

Puede extraer esta parte, simplemente ejecutando el comando **getwd()**, copiando y pegando lo que aparece.

La otra parte la extraemos del resultado de **list.files()**

Para mostrar los nombres de las hojas disponibles, es necesario ejecutar la siguiente línea de comando:

```
excel_sheets(path=MinhaPastaEArquivo)
```

Usemos ahora la función **read_excel** para leer el archivo y especificar la hoja de cálculo que nos interesa.

```
df <-read_excel(path=MinhaPastaEArquivo, sheet = "Planilha1")
```

Vea que estamos asignando el nombre de la base de datos ahora desde *df*

head(df): para leer el encabezado del dataframe.

View(df): para ver la base de datos en otra ventana

attach(df): para enviar la base de datos a la memoria

2. Pruebas de suposición ANOVA

Para probar la homogeneidad de las varianzas, usaremos la prueba de Bartlett, con la función **bartlett.test**

Vresp: es la variable de respuesta de interés o variable dependiente

Factor1: es el factor 1 o la variable independiente 1

Factor2: es el factor 2 o la variable independiente 2

Bloque: es el factor bloque

bartlett.test(df$Vresp, df$Factor1)	Prueba de Bartlett para un estudio realizado en DCA - - con un solo factor.
bartlett.test(df$Vresp, df$Factor1, df$Bloque)	Prueba de Bartlett para un estudio realizado en DBCA, con un solo factor.
bartlett.test(df$Vresp, df$Factor1, df$Factor2)	Test de Bartlett para un estudio realizado en DCA en arreglo factorial (2 factores).
bartlett.test(df$Vresp, df$Factor1, df$Factor2, df$Bloco)	Prueba de Bartlett para un estudio realizado en DBCA en arreglo factorial (2 factores).

Prueba de Bartlett: si el valor *p* es mayor que 0.05, las varianzas son homogéneas.

Prueba de Shapiro: **shapiro.test**

Si el valor *p* es mayor que 0.05, los datos son normales.

3. ANOVA con dos factores (two-way ANOVA)

Este es el modelo en el que trabajaremos para ANOVA bidireccional:

Modelofactorial<-aov(ALT~FACT1*FACT2+BLOC, data=df)

Tenga en cuenta que usamos la función **aov** para ejecutar el ANOVA.

ALT: es la variable de respuesta.

FACT1: es el factor 1.

FACT2: es el factor 2.

BLOC: es el factor de bloque.

Estas designaciones ALT, FACT1, FACT2 y BLOC provienen de la siguiente base de datos:

FACT1	FACT2	BLOC	ALT
A	1	1	80
A	1	2	60
A	1	3	69
A	1	4	87
A	2	1	94
A	2	2	83
A	2	3	81
A	2	4	80
B	1	1	91
B	1	2	103
B	1	3	98
B	1	4	107
B	2	1	95
B	2	2	94
B	2	3	96
B	2	4	85

summary(Modelofactorial): muestra el resumen del ANOVA

```
            Df Sum Sq Mean Sq F value  Pr(>F)
FAT1         1 1139.1  1139.1  16.257 0.00198 **
FAT2         1   10.6    10.6   0.151 0.70523
BLOC         1    0.0     0.0   0.000 0.98958
FAT1:FAT2    1  315.1   315.1   4.497 0.05752 .
Residuals   11  770.7    70.1
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Tenga en cuenta que este análisis contiene un error. El error de esta salida está en el número de grados de libertad para el factor **BLOC** (que corresponde a bloques). El número correcto sería **3**. La fórmula de grados de libertad es n-1, en este caso tenemos 4 bloques, lo que haría BLOC = 3.

Antes de continuar con el análisis, debemos preguntarnos si es un factor, porque tenemos variables dependientes e independientes, cuando son independientes debe ser un factor, para eso usamos el comando:

```
is.factor(df$FACT1)
is.factor(df$FACT2)
is.factor(df$BLOC)
```

Cuando las variables son independientes es necesario convertir, para eso necesitamos usar la función **as.factor** y para convertir a factor a cada variable.

```
df$FACT1<-as.factor(df$FACT1)
df$FACT2<-as.factor(df$FACT2)
df$BLOC<-as.factor(df$BLOC)
```

Atención: no realice este procedimiento de conversión de factores para sus variables dependientes (variables de respuesta).

Después de convertir las variables independientes en un factor, necesitamos probar nuevamente si las variables son factores:

```
is.factor(df$FACT1)
is.factor(df$FACT2)
is.factor(df$BLOC)
```

el resultado debería ser **TRUE**

Ahora que sus variables independientes son factores, puede ejecutar el ANOVA bifactorial.

```
Modelofactorial<-aov(ALT~FACT1*FACT2+BLOC, data=df)
summary(Modelofactorial)
```

¡Tenga en cuenta que el número de grados de libertad del factor BLOC ahora es correcto!

```
            Df Sum Sq Mean Sq F value  Pr(>F)
FAT1         1 1139.1  1139.1  14.813 0.00391 **
FAT2         1   10.6    10.6   0.137 0.71949
BLOC         3   78.7    26.2   0.341 0.79637
FAT1:FAT2    1  315.1   315.1   4.097 0.07362 .
Residuals    9  692.1    76.9
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ahora podemos aplicar las pruebas de comparación de medias. Para esto, usemos la función **fact2.rbd** y cargamos el paquete de nuestra elegante manera:

if(!require("ExpDes")) install.packages("ExpDes"); require(ExpDes)

fat2.rbd(FACT1, FACT2, BLOC, ALT, quali = c(TRUE, TRUE), mcomp = "tukey", fac.names = c("FACTOR 1", "FACTOR 2"), sigT = 0.05, sigF = 0.05)

Como la interacción no fue significativa (podemos ver esto en la línea "FACTOR1* FACTOR2"), solo se analizarán los efectos simples.

4. Otros ejemplos

Para acceder al script y la base de datos, haga **clic aquí**. Elijamos el directorio en el que queremos trabajar:

Session -> Set working Directory -> Choose Directory

Abra la carpeta que contiene su base de datos.

Usemos el comando **list.files()** para ver los nombres de todos los archivos contenidos en la carpeta seleccionada.

Cuando su base de datos está en formato Excel es necesario ejecutar el comando **df<-read_excel("BD.xlsx", sheet = 1)**, donde:

- df es el nombre que se le da al dataframe (puede poner el nombre que desee);
- read_excel es el comando para leer el archivo de Excel;
- BD. es el nombre de su archivo dentro de la carpeta seleccionada;
- xlsx es la extensión del archivo con el que está trabajando;
- sheet = 1 se refiere a qué hoja de su archivo de Excel desea analizar.

Para ver el encabezado de la base de datos, la función utilizada es: **head(df, n=2)** en este caso, en particular, solo se mostrarán 2 líneas de su base de datos.

Antes de realizar las pruebas de normalidad y homogeneidad, es necesario verificar si las variables independientes son factores.

En este caso, las variables son: Ins, Fung y Bloque

Ins	Fung	Bloc
A	X	1
A	X	2
A	X	3
A	X	4
B	X	1
B	X	2
B	X	3

Para realizar la verificación es necesario utilizar los comandos:

```
is.factor(df$Ins)
is.factor(df$Fung)
is.factor(df$Bloc)
```

La respuesta para la verificación debe ser: **TRUE**

Si la respuesta es: **FALSE**, es necesario convertir las variables independientes en factores, para eso usamos la función “as.factor”:

```
df$Ins<-as.factor(df$Ins)
df$Fung<-as.factor(df$Fung)
df$Bloc<-as.factor(df$Bloc)
```

entonces debe verificar si la conversión fue exitosa:

```
is.factor(df$Ins)
is.factor(df$Fung)
is.factor(df$Bloc)
```

Luego debe seleccionar qué variable desea analizar

```
df$VRESP<-df$D
```

En este caso trabajaremos con la variable “D”, procedente de la base de datos.

Atención: el nombre de la variable debe ser idéntico al escrito en la base de datos.

Después de seleccionar la variable que queremos analizar, podemos realizar las pruebas de normalidad y homogeneidad:

Para normalidad usamos el comando:

shapiro.test(df$VRESP)

Shapiro-Wilk normality test

data: df$VRESP
W = 0.95084, **p-value = 0.5031**

Para que los datos se consideren normales, el valor de “p” debe ser mayor que 0.05. Así, podemos verificar que el análisis en cuestión pasó la prueba de normalidad, ya que *p* = 0,5031.

Para homogeneidad usamos el comando:

bartlett.test(df$VRESP, df$Ins, df$Fung, df$Bloc)

Bartlett test of homogeneity of variances

data: df$VRESP and df$Ins
Bartlett's K-squared = 2.3704, df = 1, **p-value = 0.1237**

Para que los datos se consideren homogéneos, el valor de “p” debe ser superior a 0,05. Así, podemos verificar que el análisis en cuestión pasó la prueba de homogeneidad, ya que *p* = 0,1237.

NOTA: si los datos no cumplen con la normalidad y homogeneidad, es necesario transformarlos, verifique cómo se hace a continuación.

Luego podemos ejecutar el ANOVA, para eso usaremos la función "aov":

```
Modelofactorial<-aov(VRESP~Ins*Fung+Bloc, data=df)
summary(Modelofactorial)
```

Luego hacemos un Resumen para verificar el resultado de la prueba anterior:

```
           Df Sum Sq Mean Sq F value   Pr(>F)
Ins         1   2233    2233   41.25 0.000122 ***
Fung        1   1828    1828   33.77 0.000256 ***
Bloco       3   2841     947   17.50 0.000426 ***
Ins:Fung    1   7183    7183  132.72 1.09e-06 ***
Residuals   9    487      54
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

A través de la prueba realizada, podemos observar que existe una diferencia significativa en los grupos cuando se analizan por separado y también en la interacción insecticida versus fungicida.

NOTA: cuando la interacción no es significativa, no es necesario dividir la interacción insecticida versus fungicida.

Cuando ocurre una diferencia significativa, podemos ejecutar la prueba de Tukey, que hace múltiples comparaciones, para eso es necesario instalar el paquete “ExpDes” [2], usando el comando:

```
if(!require("ExpDes")) install.packages("ExpDes"); require(ExpDes)
```

Este comando verifica si tiene el paquete necesario y si no, descarga e instala ese paquete.

Después de instalar el paquete, podemos ejecutar la prueba de Tukey, con el siguiente comando:

```
fact2.rbd(Ins, Fung, Bloco, VRESP, quali = c(TRUE, TRUE), mcomp = "tukey",
      fac.names = c("Insecticida", "Fungicida"), sigT = 0.05, sigF = 0.05)
```

Este comando ya especifica que la importancia de los resultados es del 95%
Como resultado de la prueba de Tukey, obtenemos los desdoblamientos:

Insecticida dentro del nivel X de Fungicida

Grupos	Tratamientos	Medias
a	1	52.5
b	2	33.75

Insecticida dentro del nivel Y de Fungicida

Grupos	Tratamientos	Medias
a	2	97.5
b	1	31.5

Fungicida dentro del nivel A de Insecticida

Grupos	Tratamentos	Medias
a	1	52.5
b	2	31.5

Fungicida dentro del nivel B de Insecticida

Grupos	Tratamentos	Medias
a	2	97.5
b	1	33.75

Letras diferentes indican que hubo una diferencia significativa entre tratamientos. Para presentar los resultados finales en los artículos utilizamos estas letras, pero para obtener y presentar la media junto con una medida de variabilidad como la desviación estándar o error estándar es necesario instalar el paquete "Rmisc" y ejecutar un último comando:

```
if (!require("Rmisc")) install.packages("Rmisc"); require(Rmisc)

summarySE(df, measurevar = "VRESP", groupvars = c("Ins", "Fung"))
```

Obtenemos como resultado:

	Ins	Fung	N	VRESP	sd	se	ci
1	A	X	4	**52.50**	17.07825	**8.539126**	27.17531
2	A	Y	4	**31.50**	18.33939	**9.169696**	29.18207
3	B	X	4	**33.75**	13.76893	**6.884463**	21.90943
4	B	Y	4	**97.50**	17.07825	**8.539126**	27.17531

Esta es la media

Este es el error estándar

Un ejemplo de tabla que se puede construir a partir de los resultados obtenidos es:

	X	Y
A	52.50±8,53Aa	31.50±9,16Bb
B	33.75±6,88Bb	97.50±8,53Aa

Las letras mayúsculas comparan columnas.
Las letras minúsculas comparan líneas.

5. Cuando necesita transformación

Haga **Clique aquí** para acceder a la base de datos y al script.

Leamos la base de datos:

```
df<-read_excel("BD.xlsx", sheet = 1)
head(df, n=2)
```

Probemos las suposiciones:

```
shapiro.test(df$VRESP)
```

```
        Shapiro-Wilk normality test

data:  df$VRESP
W = 0.86057, p-value = 0.01957
```

```
bartlett.test(df$VRESP, df$Ins, df$Fung, df$Bloc)
```

```
        Bartlett test of homogeneity of variances

data:  df$VRESP and df$Ins
Bartlett's K-squared = 4.3204, df = 1, p-value = 0.03766
```

Nótese que los datos no cumplen con los supuestos de normalidad y homogeneidad de varianzas (p <0.05). Por tanto, es necesario transformarlos, para eso necesitaremos el paquete MASS [3] y usaremos el comando:

```
library(MASS)

Box = boxcox(VRESP ~ Ins*Fung+Bloc,
       data = df,
       lambda = seq(-6,6,0.1))

Cox = data.frame(Box$x, Box$y)
(Cox2 = Cox[with(Cox, order(-Cox$Box.y)),])
Cox2[1,]
(lambda = Cox2[1, "Box.x"]+0.0000001)
```

: esta línea de comando muestra el valor de lambda.

(df$VRESP_box = (df$VRESP ^ lambda - 1)/lambda): aquí introducimos la fórmula de Box-Cox [5].

Para otras bases de datos es necesario cambiar:

Nombres de: variables dependientes e independientes.

Nombre del dataframe.

Luego de la transformación, debemos repetir las pruebas de “Shapiro” y “Bartlett” para verificar que los datos cumplen con los supuestos de normalidad y homogeneidad (ATENCION AL ESCRIBIR EL COMANDO, PORQUE AHORA SE TRANSFORMAN LOS DATOS, ENTONCES ES NECESARIO AGREGAR "box”).

```
shapiro.test(df$VRESP_box)
bartlett.test(df$VRESP_box, df$Ins, df$Fung, df$Bloc)
```

Después de eso, ejecutemos el análisis de varianza:

```
if(!require("ExpDes")) install.packages("ExpDes"); require(ExpDes)

fact2.rbd(Ins, Fung, Bloco, VRESP_box, quali = c(TRUE, TRUE), mcomp = "tukey",
      fac.names = c("Insecticida", "Fungicida"), sigT = 0.05, sigF = 0.05)


if (!require("Rmisc")) install.packages("Rmisc"); require(Rmisc)
summarySE(df, measurevar = "VRESP", groupvars = c("Ins", "Fung"))
```

Cuando sea necesario transformar los datos, debemos prestar atención a los valores medios que ponemos en la tabla de resultados final, ya que no deben ser de los datos transformados, sino de los originales, que se obtienen en el análisis descriptivo arriba con la función summarySE del paquete Rmisc [4].

6. Referencias de paquetes usados

[1] Wicham H., Bryan J. readxl: Read Excel Files. R package version 1.3.1. 2019. https://CRAN.R-project.org/package=readxl

[2] Ferreira E.B., Cavalcanti P.P., Nogueira D.A. ExpDes.pt: Pacote Experimental Designs (Portuguese). R package version 1.2.0. 2018. https://CRAN.R-project.org/package=ExpDes.pt

[3] Venables W.N., Ripley B.D. Modern Applied Statistics with S. Fourth Edition. Springer, New York. 2002.

[4] Hope R.M. Rmisc: Rmisc: Ryan Miscellaneous. R package version 1.5. https://CRAN.R-project.org/package=Rmisc. 2013.

[5] Box G., Cox DR. An analysis of transformations. Journal of the Royal Society,

26: 211-252. 1964.

6. Referencias recomendadas

Crawley, M.J. The R book. John Wiley & Sons, 2012.

Matloff Norman. The art of R programming: A tour of statistical software design. No Starch Press, 2011.

Peternelli L.A., Mello M.P. Conhecendo o R: uma visão estatística. Viçosa: UFV, v. 1, 2011.