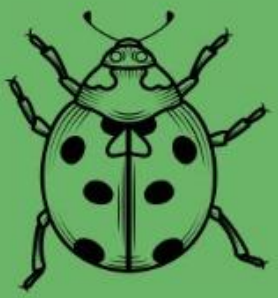
# R PARA ENTOMÓLOGOS

## NIVEL BÁSICO




**ORGANIZADORES:**

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



MERIDA PUBLISHERS

# R para Entomólogos:

## nivel básico

José Bruno Malaquias
Jéssica Karina da Silva Pachú
Tatiane Caroline Grella
Milton Fernando Cabezas Guerrero
(Orgs)

Canoas
2022

MÉRIDA
PUBLISHERS

**R para Entomólogos: nivel básico**
© 2022 Mérida Publishers
https://doi.org/10.4322/mp.978-65-994457-4-3

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

**Adaptación de la portada y diseño gráfico**
Luis Miguel Guzmán
Adriana Romero Domínguez


Canoas - RS - Brasil
**contact@meridapublishers.com**
**www.meridapublishers.com**

Todos los derechos de autor pertenecen a Mérida Publishers. Se permite la reproducción total o parcial de las obras publicadas siempre que se otorguen créditos a los autores.

**Catalogación en Publicación**
**(eDOC BRASIL)**

R111 R para entomólogos [libro electrónico] : nivel básico / Organizadores José Bruno Malaquias... [et al.]. – Canoas, RS: Mérida Publishers, 2022.

Formato: PDF
Requisitos del sistema: Adobe Acrobat Reader
Modo de acceso: World Wide Web
Incluye bibliografía
Título original: R para entomologistas
ISBN 978-65-994457-4-3

1. Estadísticas. 2. Entomología. 3. R (lenguaje de programación). I. Malaquias, José Bruno. II. Pachú, Jéssica Karina da Silva. III. Grella, Tatiane Caroline. IV. Guerrero, Milton Fernando Cabezas.
CDD 595.7

**Elaborado por Maurício Amormino Júnior – CRB6/2422**

## Presentación

Este trabajo tiene un carácter introductorio y está destinado a entomólogos que comienzan a utilizar el programa R. En este libro se exploran los primeros pasos del programa R, como lectura de archivos, instalación y carga de paquetes, análisis descriptivo, pruebas de normalidad y homogeneidad de varianzas, análisis de varianza, pruebas de comparación de medias e introducción a modelos lineales generalizados para conteo de datos.

El objetivo de este libro es presentar ideas iniciales para cada análisis mencionado anteriormente. A través de algunos ejemplos de bases de datos que fueron generados de forma simulada y obtenidas de forma experimental, presentamos gradualmente la programación necesaria para ejecutar cada análisis.

Al inicio de cada capítulo, hay un resumen y secuencia de líneas de comando comentadas y discutidas de manera práctica y sencilla, cuyo objetivo es orientar al usuario para iniciar el trabajo estadístico y que pueda asociarlo con bases de datos y situaciones similares.

Este trabajo nació de la experiencia, en el día a día en cursos de extensión sobre R para entomólogos. Así, el presente trabajo, a pesar de ser introductorio, es fruto en gran medida de la colaboración de varios compañeros, profesores y colaboradores, a los que ofrecemos nuestro más sincero agradecimiento.

Por ello, destacamos que el objetivo no es solo enseñar el programa R o estadística, sino proponer una guía para los interesados en iniciar su trabajo con R y sus herramientas.

Dr. José Bruno Malaquias

Instituto de Biociencias
Departamento de Bioestadistica UNESP
Botucatu - SP, Brasil.

## Presentación de los autores

**Filipe Lemos Jacques**
Magister en Agronomia Universidad Estadal de Ponta Grossa, Brasil.

**Jéssica Karina da Silva Pachú**
Doctora en Entomología por la Universidad de São Paulo, USP, Brasil.

**José Bruno Malaquias**
Doctor en Entomología por la Universidad de São Paulo, USP, Brasil.

**Milton Fernando Cabezas Guerrero**
Doctor en Entomología por la Escuela Superior de Agricultura Luiz de Queiroz, Universidad de São Paulo (Esalq/USP), Brasil. Profesor de la Universidad Técnica Estatal de Quevedo, Ecuador.

**Paulo Eduardo Degrande**
Doctor en Ciencias Biológicas (Zoologia). Profesor Titular de la Universidad Federal de Grande Dourados (UFGD), Brasil.

**Tatiane Caroline Grella**
Magister en Agricultura y Ambiente por la Universidad Federal de São Carlos, Brasil.

# Índice

# CAPÍTULO 1

## Introducción al ambiente R

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

**Resumen**

R cuenta con una gama de herramientas como pruebas paramétricas y no paramétricas, análisis de regresión lineal y no lineal, análisis de supervivencia, análisis multivariado, producción gráfica, entre otros. Es importante mencionar que R no es simplemente un programa estadístico. En este capítulo, presentamos aspectos introductorios al entorno R, como instalar el programa, cargar e instalar paquetes. Los comandos a ejecutar en R se muestran en color azul.

**Palabras claves**: lenguaje R; presentación; instalación; cargando; paquetes.

### 1. Introducción

La estadística se puede definir como matemáticas aplicadas a datos de observación [1]. A través de la Estadísticas obtenemos descripciones claras, sintéticas y objetivas [2].

Desde una perspectiva entomológica, la estadística es una condición fundamental ya que trabaja con métodos científicos para recolectar, organizar, resumir y presentar datos, además, de obtener conclusiones y tomar decisiones.

Las dificultades a las que se enfrentan los estudiantes e investigadores entomológicos que necesitan hacer uso de la inferencia estadística parece ser el resultado de la falta de libros didácticos escritos en un lenguaje adecuado para ellos, libres de los algebraísmos que suelen emplear los estadísticos matemáticos. Para cubrir esa brecha, este libro tiene como objetivo dar a conocer los comandos básicos para el análisis descriptivo, análisis de varianza y análisis de desviación, con datos entomológicos en lenguaje R. La estrategia empleada


Malaquias J.B., *et al*. (Orgs.) © 2020 Mérida Publishers CC-BY 4.0

para lograr este propósito consistió en preparar *scripts* en un lenguaje sencillo y con ejemplos entomológicos.

El programa estadístico que ha sido más útil en el análisis estadístico de datos biológicos es el basado en el entorno R, que es un lenguaje de programación estadística de acceso abierto que se ha vuelto cada vez más popular.

Por sus características, el *R Development Core Team* lo clasifica como un entorno R [3].

R es un software gratuito para el análisis de datos. Esto significa que se puede utilizar, copiar, distribuir, modificar y mejorar libremente. Fue desarrollado en 1996, por los profesores de estadística Ross Ihaka y Robert Gentleman, que trabajaron y fueron colegas en el departamento de estadística de la Universidad de Auckland, los dos compartieron un interés en la estadística computacional y vieron la necesidad de un mejor entorno de software para el área. Así que desarrollaron un nuevo lenguaje computacional, similar al lenguaje S [3].

El software es un entorno de investigación y desarrollo colaborativo internacional formalmente mantenido por la Fundación R, que ofrece una amplia gama de técnicas estadísticas y gráficas, que incluyen modelado lineal y no lineal, pruebas estadísticas clásicas, análisis de series de tiempo, clasificación, agrupamiento y otros. Puede "descargarlo" de Internet, sin costo (www.r-project.org). Un conjunto básico de paquetes viene integrado en la instalación de R, con muchos más disponibles en la red de distribución de R (CRAN). Investigaciones realizadas con profesionales en el área que la popularidad de R ha aumentado sustancialmente en los últimos años [4]. A partir de 2017, R ya tiene más de 10000 paquetes disponibles [5]. Un paquete es un conjunto de funciones a las que se puede acceder a su código simplemente importándolo. Hay paquetes para resolver diversos problemas como gráficos, análisis descriptivo, análisis de varianza, etc.

## 2. Motivos para el uso de R

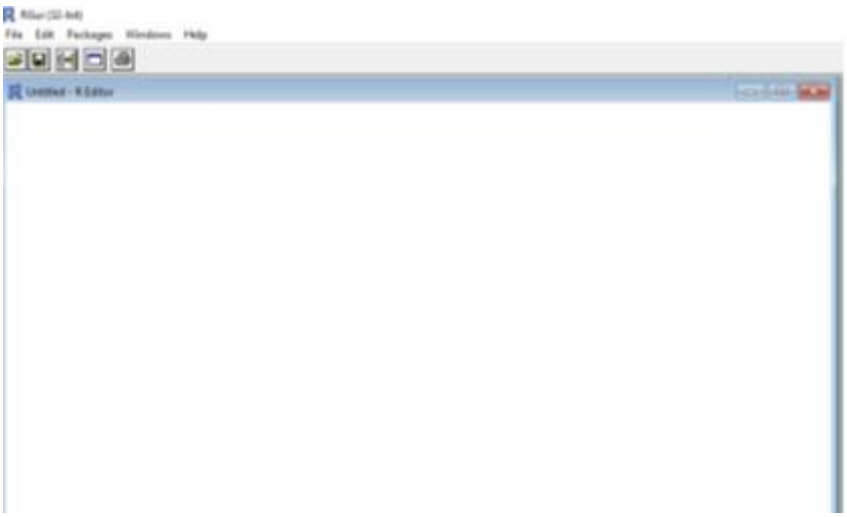
- Todos los códigos son abiertos, reproducibles y adaptables;
- Compatible con Windows, Mac, Linux y etc;
- Se utiliza para crear otro software;
- Las empresas públicas y privadas están prestando atención a R;

- Enlace con C, C ++, Java, entre otros;
- Gran documentación (cambios de versión y ayuda);
- Comunidad académica difundiendo conocimientos;
- Altamente extensible;
- La comunidad lo apoya continuamente, con correcciones rápidas de errores y nuevas funciones.

## 3. Instalación de R

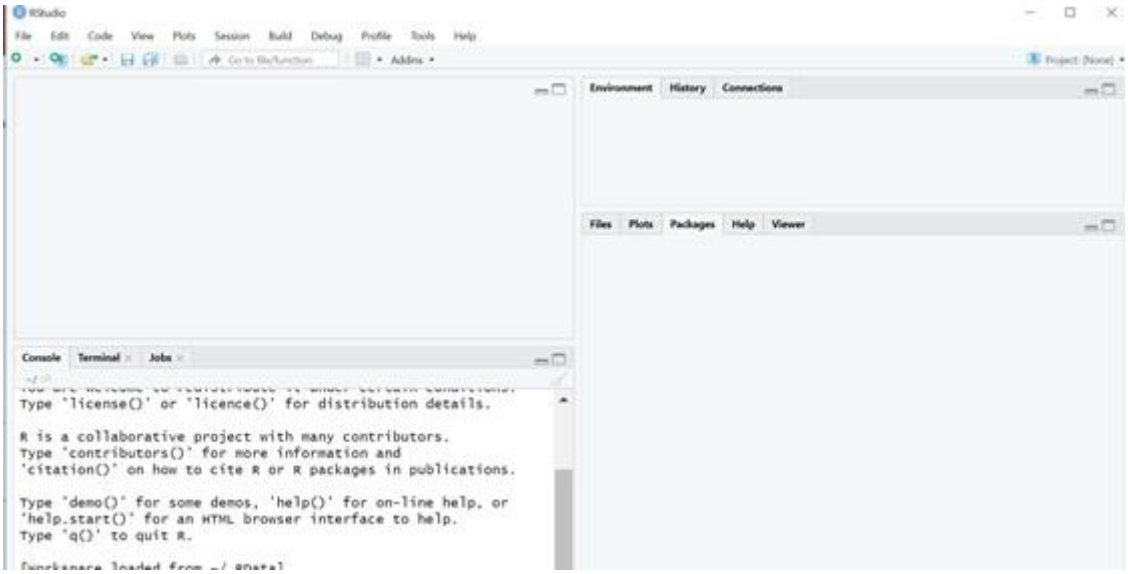
- Haga clic en CRAN;
- Elija el espejo de su elección (CRAN mirrors);
- Haga clic en Windows;
- Haga clic en la base y guarde el archivo R para Windows;
- Entonces simplemente ejecute el archivo.

## 4. Consola de software "R"



## 5. Uso de R como un entorno de desarrollo integrado (IDE)

RStudio es una interfaz integrada para gráficos y cálculos estadísticos. Accesorio escrito en lenguaje C++, está disponible en dos ediciones: RStudio Desktop, que se ejecuta localmente como una aplicación de escritorio estándar; y RStudio Server, que le permite acceder a RStudio mediante un navegador web.




Para **bajar RStudio** haga clic en: https://www.rstudio.com/products/rstudio/download/ y escoja la opción *Rstudio desktop*

## 6. Script

Para abrir y practicar en Rstudio con un nuevo script, vaya a:

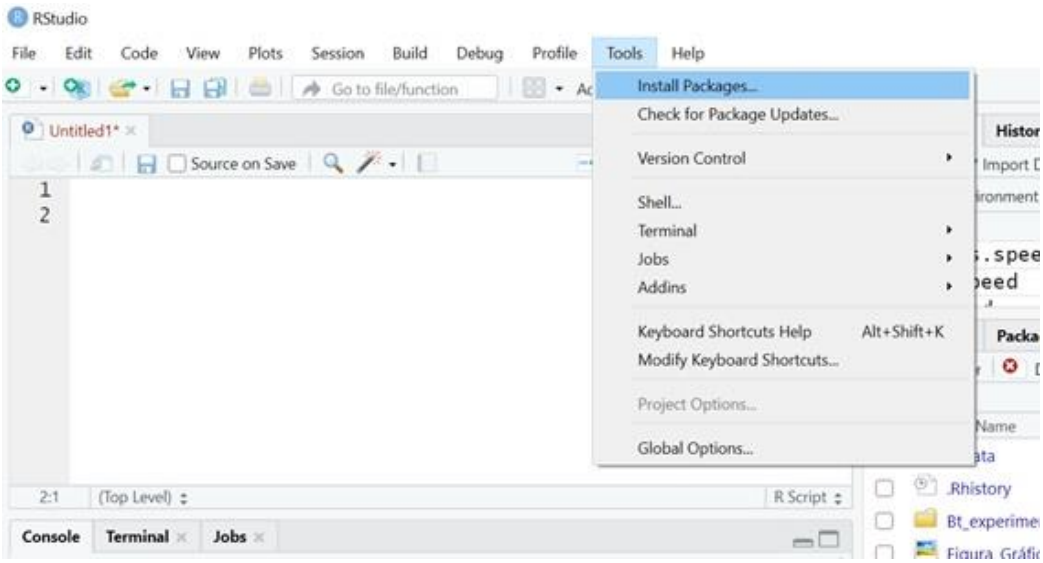
- File > New File >R Script

Si prefiere abrir un script que ya ha guardado, vaya a:

- Open File> seleccione el archivo > Abrir

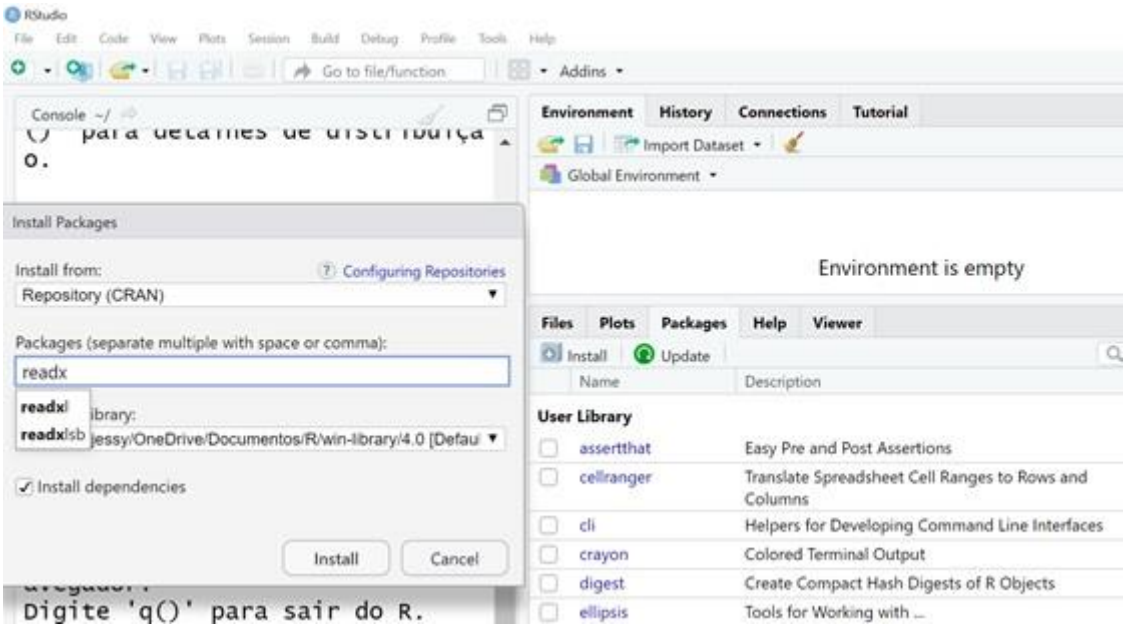
En el script puede escribir comandos a ejecutar y también comentarios. Los comandos se escriben como en la consola y todo lo que se escribe después del # se considera solo como comentarios.


R es un programa que ocupa poco espacio y memoria y generalmente se ejecuta rápido, eso se debe a que los paquetes vienen en la instalación base, es decir, configuraciones mínimas para funcionar. Para realizar tareas más complejas puede ser necesario instalar paquetes adicionales (packages), este es uno de los grandes beneficios de R, su gran colección de paquetes. Cualquiera puede enviar un paquete y cada código enviado está disponible en Internet. Sin embargo, existe un proceso de evaluación por el que pasa el código y se deben cumplir ciertas reglas estrictas con respecto al formato del código, el manual de usuario y la forma de actualizar el paquete.



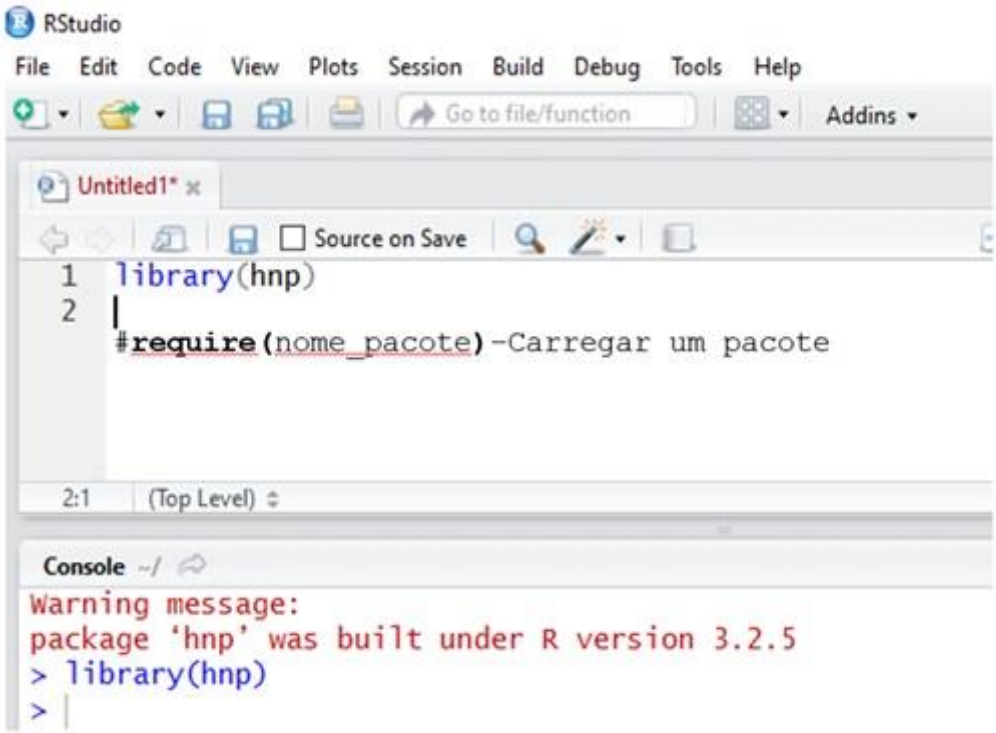

Otra forma es, en el lado derecho de la pantalla, hacer clic en:

- Packages>Install> ingrese el nombre del paquete e instálelo.




## 7. Cargando un paquete

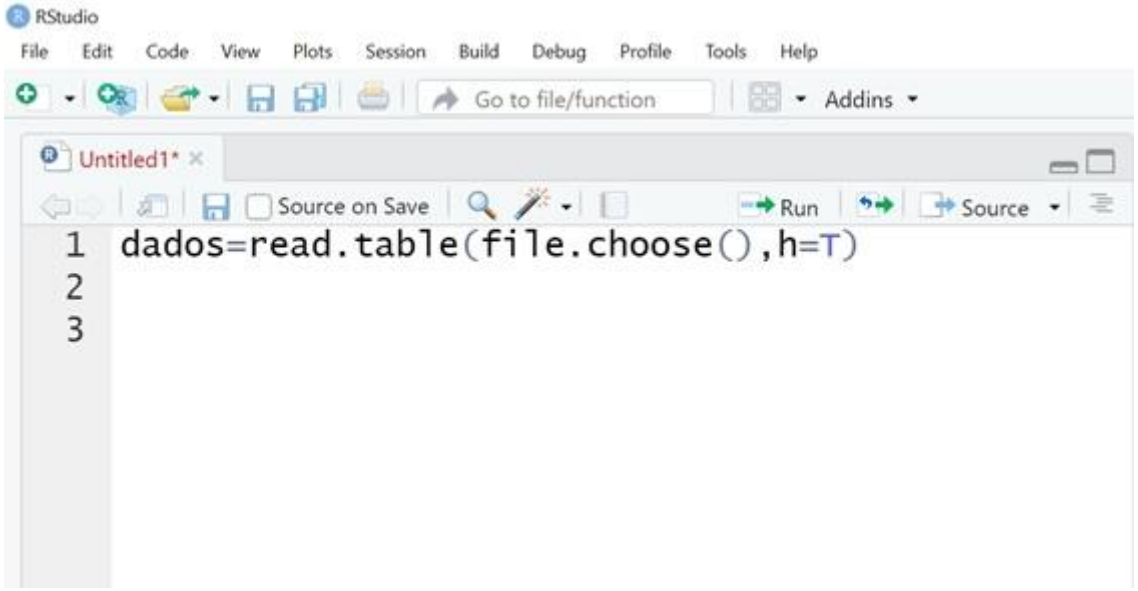
Para cargar un paquete dentro de un script, usamos la función **library()** o **require()**, dentro del paréntesis el nombre del paquete que se debe cargar.



## 8. Leer un archivo

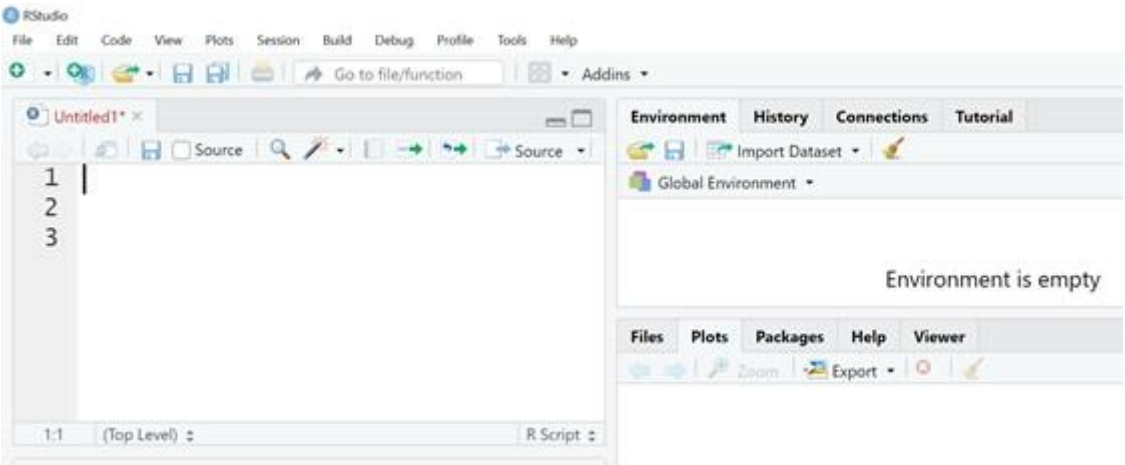
Para leer una base de datos, use la función **read.table** para importar los datos.

La línea de comando ejemplificada **dados=read.table(file.choose(),h=T** lo dirige a la carpeta que contiene su base de datos en el entorno R (datos), la función para importar los datos (read.table) y la función para elegir la carpeta donde están sus datos (file.choose).




## 9. Visualización de un Plot

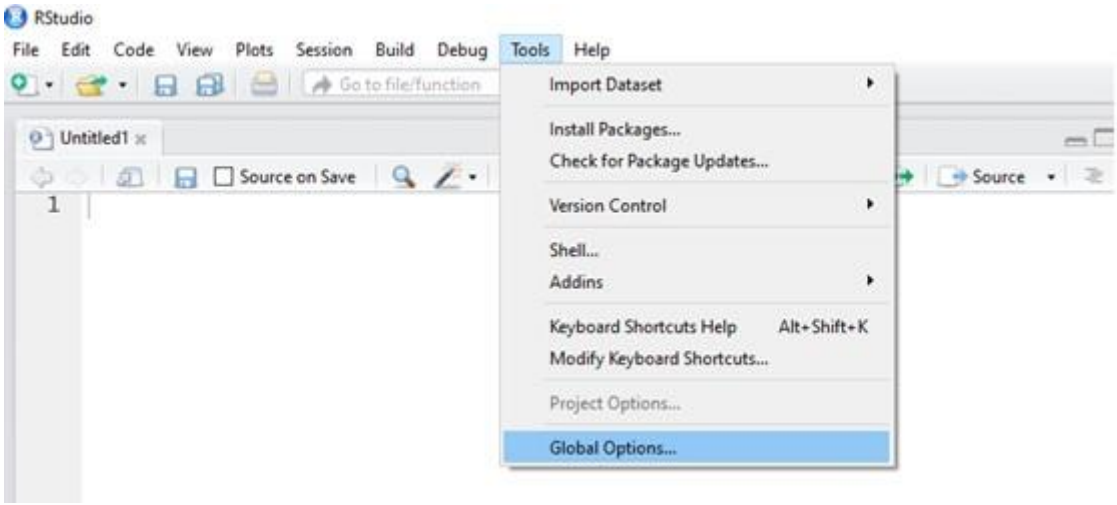
Para ver sus gráficos, haga clic en la pestaña plots en la fila superior de la consola o en la pestaña plots en el lado derecho, de esta manera puede usar la escoba para limpiar o hacer zoom para ampliar su plot.



## 10. Personaliza tu RStudio

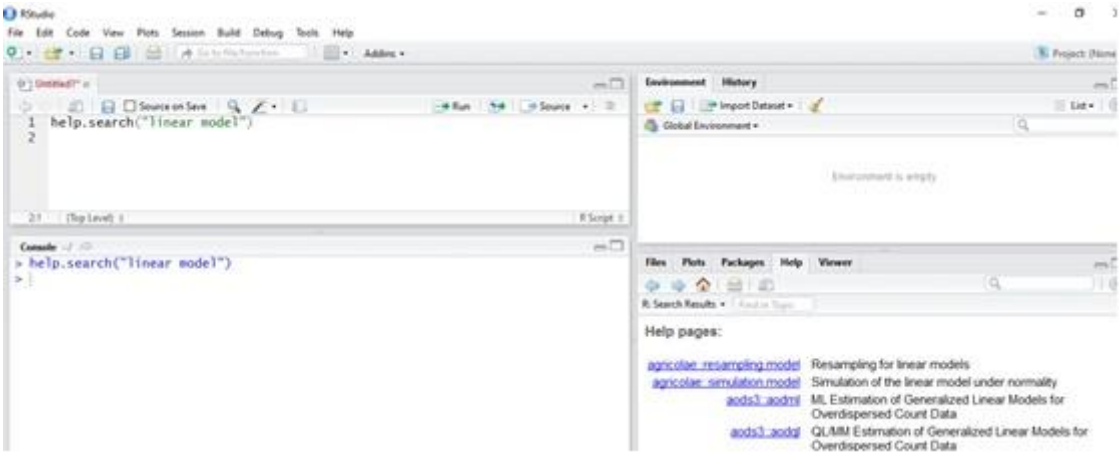
En esta opción puede cambiar y personalizar RStudio según sus preferencias, por ejemplo, modificar la fuente, tamaño, color, etc. Haga clic en:

- Tools> Global Options>




## 11. Buscando ayuda

Existen algunas formas de obtener ayuda para ejecutar comandos y paquetes. Por ejemplo, haciendo clic en la pestaña Help o escribiendo la línea de comando help.search ()




## 12. Algunas funciones y comandos básicos

Para ver la base de datos

**View ()**

Limpiar la consola

**Ctrl+L**

Para leer las líneas de comando

**Ctrl+Enter ou Ctrl+R**

Cambio de directorio

**Ctrl+Shift+H:**

**getwd()**

Para ver los nombres de los archivos dentro de la carpeta elegida para trabajar:

**list.files()**

## 13. Referencias de los paquetes utilizados

[1] Pimentel-Gomes F. Curso de estatística experimental. 15. Ed. Piracicaba: FEALQ, 2009.

[2] Nazareth H. Curso básico de estatística. Ática, São Paulo, 2003.

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

[4] Muenchen R.A. The popularity of data analysis software. URL http://r4stats.com/popularity, 2012.

[5] Smith D. CRAN now has 10,000 R packages. Here's how to find the ones you need, 2017.

## 14. Referencias recomendadas

Borcard D., Gillet F., Legendre P. Numerical ecology with R. Springer, 2018.

British Ecological Society. A guide to data management in ecology and evolution. 2014.

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.

# CAPÍTULO 2

## Análisis descriptivo y pruebas de supuestos para análisis de varianza

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

**Resumen**

En este capítulo, describimos de una manera simple y objetiva elementos esenciales para el análisis descriptivo en R usando boxplot y también exploramos algunas líneas de comando que son esenciales para las pruebas de normalidad y homogeneidad de varianzas, para un posterior análisis de varianza y la prueba de comparación de dos medias, en este caso prueba t, y comparación múltiple de medias - prueba de Tukey. Todas las líneas de comando se presentan paso a paso de forma comentada y detallada. Usaremos los siguientes paquetes: **readxl** y **ExpDes.pt**. Los comandos a ejecutar en R se muestran en color azul.

**Palabras claves**: boxplot; normalidad; homocedasticidad; variables continuas; pruebas de comparação; ANOVA.

### 1. Lectura de archivos en R

Para acceder a la base de datos y el script usados como ejemplo en este capítulo, haga **clic aquí**.

Para iniciar el análisis, es necesario descargar la base de datos y el script (disponible arriba) y luego verificar en qué parte 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**


Malaquias J.B., *et al*. (Orgs.) © 2020 Mérida Publishers CC-BY 4.0

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, leemos 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 siguiente línea de comando (esta línea también funciona si el paquete no está instalado, ya que la instalación se llevará a cabo 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"*.

## 2. Análisis descriptivo con bloxplot

Luego de cargar el paquete, necesitamos leer el archivo, para eso usaremos la línea de comando:

**df<-read_excel("BD1.xls ", sheet = 1)**

- 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;
- **BD1** es el nombre de su archivo dentro de la carpeta seleccionada;
- xls es la extensión del archivo con el que está trabajando;
- sheet = 1 se refiere a qué hoja de su archivo de Excel desea analizar.

Leamos el Archivo: **BD1.xls**.

En este ejemplo didáctico se analizó el efecto de dos tratamientos sobre el peso de los insectos (Tabla 1). El diseño experimental fue completamente al azar y con solo 3 repeticiones.

**Tabla 1**. Peso (g) de una especie de insecto hipotética sometida a dos tratamientos.

| TRATAMIENTO | REPETICIÓN | PESO |
|---|---|---|
| A | 1 | 0,0750 |
| A | 2 | 0,0810 |
| A | 3 | 0,0820 |
| B | 1 | 0,0780 |
| B | 2 | 0,0790 |
| B | 3 | 0,0800 |

Para ver el encabezado usamos la función: **head(df)**

Para ver la base de datos usamos la función: **View(df)**

Para expresar los datos en un boxplot y construirlo, usamos la función:

**boxplot(PESO~TRATAMIENTO, data=df)**

Después de la construcción, es posible cambiar varios elementos, como:

- nombre en los ejes, usando el comando:

```
boxplot(PESO~TRATAMIENTO, data=df,
        xlab = "Tratamientos",
        ylab = "Matéria seca (g)")
```

En este ejemplo, los nombres de los ejes son "Tratamientos" "Materia seca (g)"

- nombre en los ejes y con límite inferior y superior.

```
boxplot(PESO~TRATAMIENTO, data=df,
        xlab = "TratamIentos",
        ylab = "Matéria seca (g)"
        , ylim = c(0.07,0.085))
```

- nombre en los ejes y con límite superior e inferior y sumando la media dentro de boxplot.

```
boxplot(PESO~TRATAMIENTO, data=df, col= "white",
      xlab= "Tratamientos", ylab= "Matéria Seca (g)", ylim = c(0.07,0.085))
points(1:nlevels(TRATAMIENTO), tapply(PESO, TRATAMIENTO, mean),
data=df)
abline(mean(TRATAMIENTO), data=df)
```

Para exportar la imagen en el formato deseado una sugerencia es el formato tiff con 300 dpi, que suele ser el formato solicitado por las publicaciones científicas.

Para cambiar los tamaños, use: **cex.main** (título), **cex.lab** (etiquetas) y **cex.axis** (tamaño de los ejes).

```
tiff("Figura_BoxPlot_.tiff", width=12, height=8, units="in", res=300)
boxplot(PESO~TRATAMIENTO, data=df, col= "white",
      xlab= "Tratamiento", ylab= "Matéria Seca (g)",
      ylim = c(0.07,0.085),
      cex.main=1.5, cex.lab=1.5, cex.axis=1.5)
        dev.off()
```

NOTA: tenga en cuenta su directorio, ya que la imagen se exportará directamente a esta carpeta.

Para elegir el directorio use **Control + Shift + H** o simplemente use **getwd()**

La Figura 1 es el boxplot exportado. Cada flecha expresa un parámetro del análisis descriptivo, es decir, los valores mínimo, máximo y cuartil (1º, 2º y 3º).

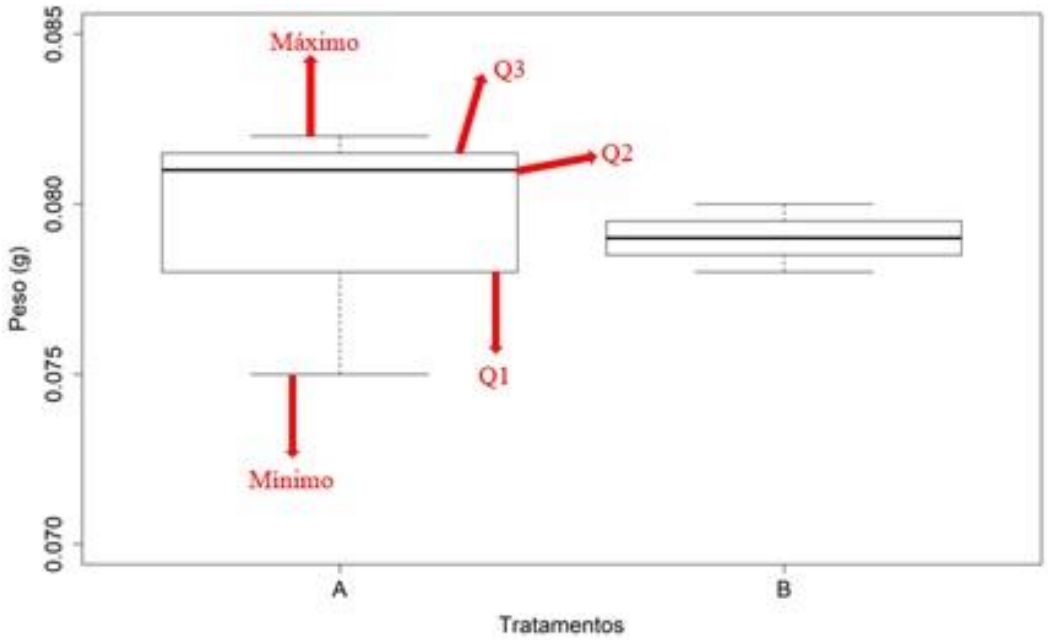



**Figura 1**. Diagrama de caja que representa el efecto de dos tratamientos sobre el peso (g) de una especie de insecto hipotética. **Q1**: primer cuartil. **Q2**: segundo cuartil o mediana. **Q3**: tercer cuartil.

Usando la línea de comando a continuación, será posible ver los mismos valores expresados en el diagrama de caja, es decir, valor mínimo (Min), primer cuartil (1st Qu.), Mediana (Mediana), media (Media), tercer cuartil (3rd Qu.) y máximo (Max), por lo que se mostrará un resumen del análisis descriptivo del peso (g) de una hipotética especie de insecto sometida a cada uno de los dos tratamientos.

**tapply(df$PESO, df$TRATAMIENTO, summary)**

```
$A
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.07500 0.07800 0.08100 0.07933 0.08150 0.08200

$B
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0780  0.0785  0.0790  0.0790  0.0795  0.0800
```

### 3. Prueba de los supuestos del modelo ANOVA

Usaremos la misma base de datos usada previamente, así que lea el archivo: BD1.xls (Tabla 1) usando el script que está disponible haciendo clic aquí. Como se muestra en la Tabla 1 y se comentó anteriormente, en este ejemplo didáctico se analizó el efecto de dos tratamientos sobre el peso de los insectos. El diseño experimental fue completamente al azar y con solo 3 repeticiones.

Cargue el paquete **readxl** para leer el archivo de Excel:

**require(readxl)** si tiene preguntas sobre la carga e instalación de paquetes, consulte el tema "Lectura de archivos en R".

Leer la base de datos: **df<- read_excel("BD1.xlsx", sheet = 1)**

Ver encabezado: **head(df)**

Ver la base de datos: **View(df)**

Antes de realizar el análisis **ANOVA**, es necesario probar la normalidad y homogeneidad de las varianzas.

Para probar la normalidad, usamos la prueba de Shapiro Wilk, usando:

**shapiro.test(df$PESO)**

Prueba de Shapiro: Si el valor *p* es mayor que 0.05 (5%), hay normalidad de los datos.

Para probar la homogeneidad, en un estudio realizado en DCA, con un solo factor, utilizamos la prueba de Bartlett, utilizando:

**bartlett.test(df$PESO, df$TRATAMIENTO)**

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

NOTA: si sus datos son variables continuas y no cumplen con la normalidad y/u homogeneidad, serán necesarias transformaciones.

## 4. Prueba *t*

Continuaremos usando la misma base de datos que se utilizó anteriormente, pero con la modificación del script al que se puede acceder haciendo clic aquí, luego leer el archivo: **BD1.xls** (Tabla 1). Como solo hay dos tratamientos, aplicaremos una prueba *t*. Pero antes de eso, necesitaremos probar los supuestos de normalidad y homogeneidad de las varianzas. Para obtener más detalles, consulte el tema: “Prueba de los supuestos del modelo Anova”.

Cargue el paquete **readxl** para leer el archivo de Excel:
**require(readxl)** si tiene preguntas sobre la carga e instalación de paquetes, consulte el tema "Lectura de archivos en R".

Leer la base de datos: **df<- read_excel("BD1.xlsx", sheet = 1)**

Ver encabezado: **head(df)**

Ver la base de datos: **View(df)**

Prueba de Shapiro Wilk: **shapiro.test(df$PESO)**

Prueba de Bartlett para un estudio realizado en DCA - - con un solo factor:

**bartlett.test(df$PESO, df$TRATAMIENTO)**

Para aplicar la prueba *t*, use la función **t.test.**

- El peso es la variable de respuesta;
- El tratamiento es la variable independiente.

Puedes usar las opciones: "**two.sided**", "**less**" ou "**greater**".

**t.test(PESO ~ TRATAMIENTO, data = df, alternative = "two.sided")**

## 5. Prueba de Tukey

Usemos la base de datos presente en el Archivo: **Anova1.xlsx** (Tabla 2). El diseño experimental fue completamente al azar con 4 repeticiones. Como hay más de dos tratamientos, aplicaremos una prueba de comparaciones múltiples, en este caso la prueba de Tukey. Recuerde probar los supuestos de normalidad y homogeneidad de las varianzas, para obtener más detalles, consulte el tema: "Prueba de los supuestos del modelo ANOVA".

**Tabla 2**. Diámetro (mm) del *pronotum* de una hipotética especie de insecto sometida a tres tratamientos

| TRATAMIENTO | REPETICION | DIAMETRO |
|---|---|---|
| X | 1 | 30 |
| X | 2 | 40 |
| X | 3 | 20 |
| X | 4 | 67 |
| Y | 1 | 20 |
| Y | 2 | 20 |
| Y | 3 | 35 |
| Y | 4 | 45 |
| Z | 1 | 60 |
| Z | 2 | 40 |
| Z | 3 | 50 |
| Z | 4 | 30 |

Cargue el paquete **readxl** para leer el archivo de Excel:

**require(readxl)** si tiene preguntas sobre la carga e instalación de paquetes, consulte el tema "Lectura de archivos en R".

Leer la base de datos: **df<- read_excel("BD1.xlsx", sheet = 1)**

Ver encabezado: **head(df)**

Ver la base de datos: **View(df)**

Prueba de Shapiro Wilk: **shapiro.test(df$DIAMETRO)**

Prueba de Bartlett para un estudio realizado en DCA, con un solo factor:

**bartlett.test(df$DIAMETRO, df$TRATAMIENTO)**

Para este análisis usaremos el paquete ExpDes [2]: **require(ExpDes)**

Para enviar los comandos escritos (data frame) a la memoria: **attach(df)**

La prueba se realizó en un DCA, por lo que usaremos la función **crd**

#en **mcomp**, escoja el método: "**tukey**"

**crd(TRATAMENTO, DIAMETRO, quali = TRUE, mcomp = "tukey", nl=FALSE, sigT = 0.05, sigF = 0.05)**

Este es el resultado del análisis:

```
De acordo com o teste F, as medias nao podem ser consideradas diferentes.
------------------------------------------------------------------------
  Niveis Medias
1      x  39.25
2      y  30.00
3      z  45.00
------------------------------------------------------------------------
```

Del análisis, queda claro que no hay evidencia de diferencias entre tratamientos. Si desea presentar los resultados usando letras, simplemente asigne la misma letra a las 3 medias, aunque esto sea considerado redundante.

## 6. Referencias de los paquetes utilizados

[1] WICKHAM 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

## 7. Referencias recomendadas

CRAWLEY, Michael 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, Luiz Alexandre; MELLO, MP de. Conhecendo o R: uma visão estatística. Viçosa: UFV, v. 1, 2011.

VENABLES W.N., RIPLEY B. D. Modern Applied Statistics with S. Fourth Edition. Springer, New York. 2002.

# 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**


Malaquias J.B., *et al*. (Orgs.) © 2020 Mérida Publishers CC-BY 4.0

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.

Para otras bases de datos es necesario cambiar:

Nombres de: variables dependientes e independientes.

Nombre del dataframe.

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

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.

# CAPÍTULO 4

## Aplicaciones de Modelos Lineales Generalizado (GLM) para datos de conteo

José Bruno Malaquias, Jéssica Karina da Silva Pachú, Filipe Lemos Jacques, Paulo Eduardo Degrande, Milton Fernando Cabezas Guerrero

**Resumen**

En esta última sección, presentamos dos ejemplos de datos que son de naturaleza discreta. Para ello, utilizamos la prueba de ajuste de Modelos Lineales Generalizados (GLM) con las siguientes distribuciones: Poisson, cuasi-Poisson y Binomial Negativo. Una de las bases de datos tiene un solo factor, mientras que la segunda base de datos está formada por un bifactorial. Usaremos los siguientes paquetes R: **readxl**, **hnp**, **MASS**, **multcomp** y **Rmisc**. Los comandos a ejecutar en R se muestran en color azul.

**Palabras claves**: desviación; variable discreta; número de insectos; contrastes; glm.

### 1. Importación de base de datos

La base de datos a trabajar es la ocurrencia de *Scaptocoris castanea* Perty, 1830 (Hemiptera: Cydnidae) en diferentes cultivos. Para acceder a la base de datos haga clic aquí, la misma que también se expone en el apéndice de este capítulo.

- *Host* es la planta huésped;
- *Rep* corresponde al bloque (repetición) y
- *y* es la variable de respuesta que se estudió como ocurrencia de insectos.

Antes de iniciar el análisis debemos ejecutar el comando: **rm(list=ls(all=TRUE))** que sirve para borrar la memoria del programa y evitar errores.


Malaquias J.B., *et al*. (Orgs.) © 2020 Mérida Publishers CC-BY 4.0

Después de cambiar nuestro directorio (seleccione la carpeta que contiene los archivos que usaremos): **Session -> Set working Directory -> Choose Directory**

Podemos ver los nombres de todos los archivos contenidos en la carpeta seleccionada: **list.files()**

Luego debemos usar el comando para cargar el paquete que lee la hoja de Excel, que es el paquete readxl [1]: **require(readxl)**

Luego de cargar el paquete, necesitamos leer el archivo, para eso usaremos la línea de comando:

**data<-read_excel("Dados-Abundancia.xlsx", sheet = 1)**

- data 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;
- Abundancia de datos es el nombre de su archivo dentro de la carpeta seleccionada;
- xlsx es la extensión del archivo con el que está trabajando;
- sheet = 1 hace referencia a qué pestaña de su archivo de Excel desea analizar.

## 2. Análisis descriptivo

El primer paso es la producción de un análisis descriptivo, para ello usaremos el paquete **Rmisc** [2], y para cargarlo usaremos el comando: **require(Rmisc)**.

Luego con la función **summarySE** obtendremos los valores medios, desviación estándar, error estándar e intervalos de confianza asociados a la media, para eso debemos ejecutar la línea de comando:

**(descriptiva<- summarySE(data, measurevar="y", groupvars=c("Host")))**

El resultado del análisis descriptivo se muestra a continuación:

| | Host | N | y | sd | se | ci |
|---|---|---|---|---|---|---|
| 1 | Algodão | 10 | 194.8 | 53.1722568 | 16.8145440 | 38.0371411 |
| 2 | Brachiaria | 10 | 17.5 | 6.7535999 | 2.1356758 | 4.8312343 |
| 3 | Crotalaria | 10 | 25.5 | 19.9568980 | 6.3109253 | 14.2763048 |
| 4 | Girassol | 10 | 0.6 | 0.6992059 | 0.2211083 | 0.5001818 |
| 5 | Guandu | 10 | 1.9 | 1.7288403 | 0.5467073 | 1.2367379 |
| 6 | Milho | 10 | 366.2 | 98.5391067 | 31.1608016 | 70.4906305 |
| 7 | Mucuna Preta | 10 | 1.6 | 1.8378732 | 0.5811865 | 1.3147353 |
| 8 | Pousio | 10 | 0.2 | 0.4216370 | 0.1333333 | 0.3016210 |
| 9 | Soja | 10 | 58.3 | 29.4016250 | 9.2976102 | 21.0326555 |
| 10 | Sorgo | 10 | 1.6 | 2.0110804 | 0.6359595 | 1.4386403 |

El siguiente paso es elegir el modelo que mejor se adapte a este conjunto de datos.

### 3. Prueba de ajuste del modelo

Cuando la variable es independiente, debemos verificar si son factores, para eso usamos el comando:

**is.factor(data$Host)**

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”: **data$Host<-as.factor(data$Host)**

A continuación, compruebe si la conversión se realizó correctamente: **is.factor(data$Host)**

Con una conversión exitosa, podemos analizar cuál de los 4 modelos se ajusta mejor a los datos, para eso usaremos los modelos:

**model1<-glm(y~Host+Rep, data=data):** Modelo gaussiano (no es necesario probar este modelo ya que se aplica a variables continuas.
**model2<-glm(y~Host+Rep, family="poisson", data=data)**: Modelo Poisson.
**model3<-glm(y~Host+Rep, family="quasipoisson", data=data)**: Modelo quasi-Poisson.

Algunos modelos necesitan paquetes específicos (como model4, modelo binomial negativo). Para ello es necesario cargar el paquete **MASS** [3] para trabajar con la función **glm.nb**

**Require(MASS)**

**model4<-glm.nb(y~Host+Rep, data=data)**: Modelo Binomial Negativo.

Luego, es necesario cargar el paquete "**hnp**" para analizar la calidad de ajuste de los modelos a los datos y usamos: **require(hnp)**. El paquete hnp fue desarrollado por Moral *et al*. [4], para más detalles sobre media normal con envelope simulado, ver Demetrius [5] y Demetrius *et al*. [6].

Para ver los gráficos en otra ventana, use el comando: **dev.new()**
Para agrupar los 4 gráficos en una cuadrícula de 2 x 2: **par(mfrow=c(2,2))**

Para probar el ajuste de los diferentes modelos usaremos diferentes líneas de comando:

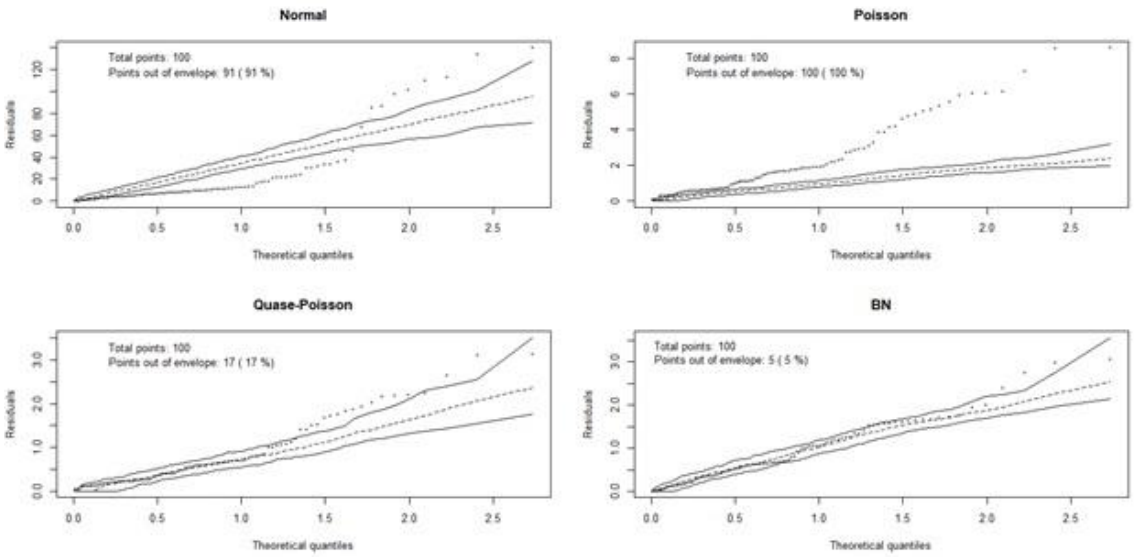
**hnp(model1, print.on="T", main="Normal")**: Prueba de bondad de ajuste del modelo Normal

**hnp(model2, print.on="T", main="Poisson")**: Prueba de bondad de ajuste del modelo Poisson

**hnp(model3, print.on="T", main="QuasePoisson")**: Prueba de bondad de ajuste del modelo quase-Poisson

**hnp(model4, print.on="T", main="BN")**: Prueba de bondad de ajuste del modelo Binomial Negativo

Este es el resultado de ajustar los modelos a los datos de conteo:



Se puede observar que el modelo **Binomial Negativo (BN)** fue el modelo que mejor se ajustó a los datos de conteo de *S. castanea*, ya que solo el 5% de los puntos estaban fuera del envelope. Entonces esta será el modelo seleccionado.

El siguiente paso es ejecutar el comando anova con la prueba F para mostrar el análisis de *desviación* del modelo seleccionado. Para ello usaremos el comando:

**anova(model4, test="F")**

```
Analysis of Deviance Table

Model: Negative Binomial(3.3323), link: log

Response: y

Terms added sequentially (first to last)

     Df Deviance Resid. Df Resid. Dev        F Pr(>F)
NULL                    99    1189.58
Host  9  1071.87        90     117.71 119.0965 <2e-16 ***
Rep   9     3.54        81     114.17   0.3935 0.9389
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

De acuerdo con el análisis de *desviación*, es posible verificar que existe evidencia de una diferencia significativa entre las plantas hospedantes (factor *Host*), ya que Pr (> F) es menor a 0.05 (5%).

Ahora comparemos los tratamientos usando el paquete "**multcomp**" [7], para eso ejecutaremos la línea de comando: **require(multcomp)**

Para visualizar las diferencias usaremos la función **glht** y el método Tukey, para eso es necesario ejecutar el comando:

**(Comparaciones <- summary(glht(model4, linfct = mcp(Host = "Tukey"))))**
**Comparaciones <- summary(Comparaciones, test = univariate())**

A continuación, imprimiremos los resultados expresando las letras de cada tratamiento:

**(Letras.mP<-cld(Comparaciones, level=0.05, Letters= c(LETTERS, letters), decreasing=TRUE))**

De la prueba anterior, obtendremos como resultado las comparaciones. Aquellas con las mismas letras se consideran significativamente iguales:

```
      Algodão    Brachiaria     Crotalaria       Girassol         Guandu          Milho
          "B"           "D"            "D"           "FG"            "E"           "A"
Mucuna Preta        Pousio           Soja          Sorgo
         "EF"           "G"            "C"           "EF"
```

## 4. Importación de base de datos - Ejemplo 2

Ejemplo 2 – Datos de conteo en un ensayo en arreglo factorial que contabilizó el número de larvas de *Spodoptera frugiperda* (Lepidoptera: Noctuidae) en diferentes plantas hospedantes (factor 1) en tres estadios larvales (factor 2). Para obtener la base de datos y el script, haga **clic aquí**

Para leer los archivos de Excel, necesitamos el paquete "readxl" [1], para eso usamos el comando: **library(readxl)**.

Para leer la base de datos, use el comando:
**df <- read_excel("DadosConteo.xlsx", sheet = 1)**

Para ver el encabezado: **head(df, n=2)**

Antes de proceder con el análisis es necesario comprobar si las variables son factores, para ello usamos el comando:
**is.factor(df$Estadio)**: Preguntar si el Estadio es un factor
**is.factor(df$Bloque)**: Preguntar si el Bloque es un factor
**is.factor(df$Cultivo)**:Preguntar si el Cultivo es un factor

Si la respuesta es "FALSE", es necesario convertir las variables.
**df$Estadio<-as.factor(df$Estadio)**: Conversión de la variable Estadio en un factor
**df$Bloco<-as.factor(df$Bloque)**: Conversión de la variable Bloque en un factor
**df$Cultura<-as.factor(df$Cultivo)**: Conversión de la variable Cultivo en un factor

Después de la conversión es necesario comprobar si funcionó.

**is.factor(df$Estadio)**: Preguntar de nuevo si el Estadio es un factor
**is.factor(df$Bloque)**: Preguntar de nuevo si el Bloque es un factor
**is.factor(df$Cultivo)**: Preguntar de nuevo si el Cultivo es un factor

Si la respuesta es "TRUE", podemos continuar con el análisis.

**NOTA**: si tiene dudas sobre los pasos descritos anteriormente, consulte los capítulos anteriores.

### 5. Prueba de ajuste

Probemos el ajuste de los modelos de Poisson, Quasi-Poisson y Binomial negativo:

```
Modelo.poisson<-glm(SfrugTot~Estadio*Cultivo+Bloque, family= "poisson", data=df)
Modelo.qpoisson<-glm(SfrugTot~Estadio*Cultivo+Bloque, family= "quasipoisson", data=df)
```

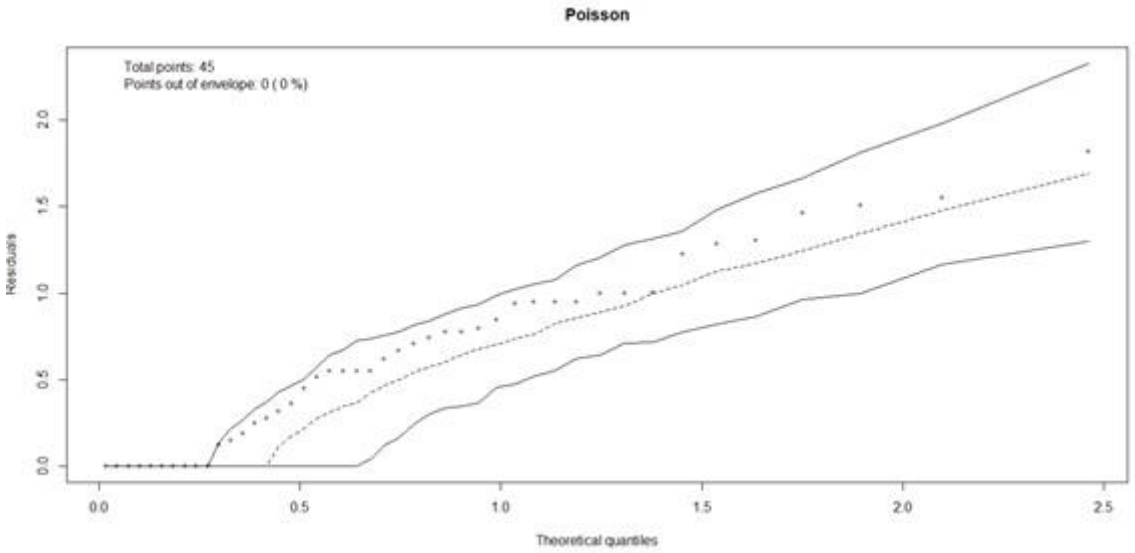
**library(MASS)** – recuerde que necesita cargar el paquete MASS [2] para probar el modelo binomial negativo.

```
Modelo.BN<-glm.nb(SfrugTot~Estadio*Cultivo+Bloque, data=df)
```

Ahora probemos si el modelo se ajusta a los datos, para eso ejecutaremos el paquete hnp [3] con el siguiente comando:

```
require(hnp)
hnp(Modelo.poisson, print.on="T", main="Poisson")
```

Es posible observar que el modelo de Poisson se ajustó a los datos, ya que ninguno de los puntos estaba fuera de las líneas, por lo que usaremos este modelo.




Después de elegir el modelo, podemos ejecutar el comando ANOVA. Como el modelo de Poisson se ajusta a los datos, usaremos la prueba de chi-cuadrado (Chisq):

**anova(Modelo.poisson, test="Chisq")**

Este es el resumen del análisis de *desviación*. Se puede observar que hubo una interacción significativa entre los factores Estadio y Cultivo.

Tabla de análisis de *desviación* - Model: poisson, link: log

Respuesta: SfrugTot

```
Analysis of Deviance Table

Model: poisson, link: log

Response: SfrugTot

Terms added sequentially (first to last)


               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                             44     89.170
Estadio         2   4.9117        42     84.259  0.085792 .
Cultura         2  18.6056        40     65.653 9.117e-05 ***
Bloco           4  13.4147        36     52.238  0.009418 **
Estadio:Cultura 4  24.2078        32     28.031 7.257e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## 6. Fraccionamiento de la interacción

Como la interacción fue significativa, realizamos el desdoblamiento de la interacción y para ello necesitamos conocer los niveles de cada factor. Para esto usaremos la función **sqldf** del paquete **sqldf** [8]

**sqldf::sqldf("select distinct Estadio from df")**: Verifica los niveles de los factores del Estadio

**sqldf::sqldf("select distinct Cutivo from df")**: Verifica los niveles del factor de Cultivo

- Para el factor Estadio tenemos: V5, V7 y V8.
- En cuanto a Cultivo tenemos: Milhoconv; CornBt y Urochloa

Primero comparemos las etapas dentro de cada cultivo.

Primero, crearemos un subconjunto dentro de cada cultivo:

**EstadiosMC<-subset(df, df$Cultivo=="Maizoconv")**

**EstadiosMBt<-subset(df, df$Cultivo=="MaizBt")**

**EstadiosU<-subset(df, df$Cultivo=="Urochloa")**

Entonces necesitaremos crear un modelo para el cultivo.

**Modelo.EstadiosMC<-glm(SfrugTot~Estadio+Bloque, family= "poisson", data=EstadiosMC)**

**Modelo.EstadiosMBt<-glm(SfrugTot~Estadio+Bloque, family= "poisson", data=EstadiosMBt)**

**Modelo.EstadiosU<-glm(SfrugTot~Estadio+Bloque, family= "poisson", data=EstadiosU)**

Para realizar las comparaciones necesitaremos el paquete "**multcomp**" [7], para ello ejecutaremos la línea de comando (cargará o instalará el paquete si es necesario):

**if(!require(multcomp)){install.packages("multcomp")}**

Modelo para datos de maíz convencional:

**Comp.EstadiosMC<-summary(glht(Modelo.EstadiosMC,linfct=mcp(Estadio="Tukey")))**

Modelo para datos de maíz Bt:

**Comp.EstadiosMBt<-summary(glht(Modelo.EstadiosMBt, infct=mcp(Estadio="Tukey")))**

Modelo para datos de Urochloa:

**Comp.EstadiosU<-summary(glht(Modelo.EstadiosU, linfct=mcp(Estadio="Tukey")))**

Resultados de comparaciones en forma de probabilidad:

**(Comp.EstadiosMC<-summary(Comp.EstadiosMC, test = univariate()))**

```
Linear Hypotheses:
             Estimate Std. Error z value Pr(>|z|)
V7 - V5 == 0  -1.3863     0.7906  -1.754   0.0795 .
V8 - V5 == 0  -0.9808     0.6770  -1.449   0.1474
V8 - V7 == 0   0.4055     0.9129   0.444   0.6569
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Univariate p values reported)
```

**(Comp.EstadiosMBt<-summary(Comp.EstadiosMBt, test = univariate()))**

```
Linear Hypotheses:
                 Estimate Std. Error z value Pr(>|z|)
V7 - V5 == 0  2.472e+01  8.153e+04       0        1
V8 - V5 == 0  3.203e-09  1.153e+05       0        1
V8 - V7 == 0 -2.472e+01  8.153e+04       0        1
(Univariate p values reported)
```

**(Comp.EstadiosU<-summary(Comp.EstadiosU, test = univariate()))**

```
Linear Hypotheses:
             Estimate Std. Error z value Pr(>|z|)
V7 - V5 == 0   1.7918     1.0801   1.659   0.0971 .
V8 - V5 == 0   2.8332     1.0290   2.753   0.0059 **
V8 - V7 == 0   1.0415     0.4749   2.193   0.0283 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Univariate p values reported)
```

De los resultados de las comparaciones en forma de probabilidad, podemos ver que hubo una diferencia significativa solo en la tercera comparación en “V8 - V5” y “V8 - V7”.

Para visualizarlo con mayor claridad, es posible asignar letras a las diferencias significativas, con los siguientes comandos:

**(Letras.EstadiosMC<-cld(Comp.EstadiosMC, level=0.05, Letters= c(LETTERS, letters), decreasing=TRUE))**

| V5 | V7 | V8 |
|---|---|---|
| "A" | "A" | "A" |

**(Letras.EstadiosMBt<-cld(Comp.EstadiosMBt, level=0.05, Letters= c(LETTERS, letters), decreasing=TRUE))**

| V5 | V7 | V8 |
|---|---|---|
| "A" | "A" | "A" |

**(Letras.EstadiosU<-cld(Comp.EstadiosU, level=0.05, Letters= c(LETTERS, letters), decreasing=TRUE))**

| V5 | V7 | V8 |
|---|---|---|
| "B" | "B" | "A" |

Asignando letras podemos confirmar la diferencia estadística que se produce en la última comparación.

Ahora comparemos los cultivos dentro de cada Estadio.

Haremos un subconjunto dentro de cada Estadio:

**CultivosV5<-subset(df, df$Estadio=="V5")**

**CultivosV7<-subset(df, df$Estadio=="V7")**

**CultivosV8<-subset(df, df$Estadio=="V8")**

Posteriormente, necesitaremos crear un modelo para cada Estadio.

**Modelo.V5<-glm(SfrugTot~Cultivo+Bloque, family= "poisson", data=CultivosV5)**

**Modelo.V7<-glm(SfrugTot~Cultivo+Bloque, family= "poisson", data=CultivosV7)**

**Modelo.V8<-glm(SfrugTot~Cultivo+Bloque, family= "poisson", data=CultivosV8)**

Realicemos las comparaciones utilizando el mismo método que el anterior, solo contrastando el factor “Cultivo”.

**Comp.CultivosV5<-summary(glht(Modelo.V5, linfct=mcp(Cultivo="Tukey")))**

**Comp.CultivosV7<-summary(glht(Modelo.V7, linfct=mcp(Cultivo="Tukey")))**

**Comp.CultivosV8<-summary(glht(Modelo.V8, linfct=mcp(Cultivo="Tukey")))**

Resultados de comparaciones en forma de probabilidad:

**(Comp.CultivosV5<-summary(Comp.CultivosV5, test = univariate()))**

```
Linear Hypotheses:
                        Estimate Std. Error z value Pr(>|z|)
Milhoconv - MilhoBt == 0   22.425  15878.588   0.001   0.9989
Urochloa - MilhoBt == 0    20.345  15878.588   0.001   0.9990
Urochloa - Milhoconv == 0  -2.079      1.061  -1.961   0.0499 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Univariate p values reported)
```

**(Comp.CultivosV7<-summary(Comp.CultivosV7, test = univariate()))**

```
Linear Hypotheses:
                        Estimate Std. Error z value Pr(>|z|)
Milhoconv - MilhoBt == 0  -0.4055     0.9129  -0.444    0.657
Urochloa - MilhoBt == 0    0.6931     0.7071   0.980    0.327
Urochloa - Milhoconv == 0  1.0986     0.8165   1.346    0.178
(Univariate p values reported)
```

**(Comp.CultivossV8<-summary(Comp.CultivosV8, test = univariate()))**

```
Linear Hypotheses:
                           Estimate Std. Error z value Pr(>|z|)
Milhoconv - MilhoBt == 0  2.234e+01  2.485e+04   0.001  0.99928
Urochloa - MilhoBt == 0   2.407e+01  2.485e+04   0.001  0.99923
Urochloa - Milhoconv == 0 1.735e+00  6.262e-01   2.770  0.00561 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Univariate p values reported)
```

De los resultados de las comparaciones en forma de probabilidad, podemos ver que hubo una diferencia significativa solo en la primera comparación en “Urochloa - Maizconv” y en la última comparación en “Urochloa - Maizconv”.

Para visualizarlo con mayor claridad, es posible asignar letras a las diferencias significativas, con los siguientes comandos:

**(Letras.CultivosV5<-cld(Comp.CultivosV5, level=0.05, Letters= c(letters, letters), decreasing=TRUE))**

| MaizBt | Maizconv | Urochloa |
|---|---|---|
| "ab" | "a" | "b" |

**(Letras.CultivosV7<-cld(Comp.CultivosV5, level=0.05, Letters= c(letters, letters), decreasing=TRUE))**

| MaizBt | Maizconv | Urochloa |
|---|---|---|
| "a" | "a" | "a" |

**(Letras.CultivosV8<-cld(Comp.CultivosV5, level=0.05, Letters= c(letters, letters), decreasing=TRUE))**

| MaizBt | Maizconv | Urochloa |
|---|---|---|
| "ab" | "a" | "b" |

Con la asignación de letras podemos confirmar la diferencia estadística que se produce en la última comparación.

Ahora realicemos un análisis descriptivo, para eso necesitamos el paquete “Rmisc” [2]. Ejecutemos la línea de comando que vemos a continuación, el paquete se instalará, si es necesario.

**if(!require(Rmisc)){install.packages("Rmisc")}:** usemos el paquete "Rmisc"
Luego, hacemos un "summary" para ver un resumen del análisis:

**summarySE(df, measurevar = "SfrugTot", groupvars = c("Estadio", "Cultivo"))**

```
Estadio   Cultura N SfrugTot        sd        se        ci
     V5   MilhoBt 5      0.0 0.0000000 0.0000000 0.0000000
     V5 Milhoconv 5      1.6 1.1401754 0.5099020 1.4157148
     V5  Urochloa 5      0.2 0.4472136 0.2000000 0.5552890
     V7   MilhoBt 5      0.6 1.3416408 0.6000000 1.6658671
     V7 Milhoconv 5      0.4 0.8944272 0.4000000 1.1105780
     V7  Urochloa 5      1.2 0.8366600 0.3741657 1.0388506
     V8   MilhoBt 5      0.0 0.0000000 0.0000000 0.0000000
     V8 Milhoconv 5      0.6 0.5477226 0.2449490 0.6800874
     V8  Urochloa 5      3.4 2.7018512 1.2083046 3.3547914
```

Los valores que no tienen variabilidad no se pueden considerar en las comparaciones. Los valores obtenidos del análisis descriptivo se pueden colocar en una tabla de la siguiente manera:

| Cultivo | Estadios | | |
|---|---|---|---|
| | V5 | V7 | V8 |
| Maíz Bt | 0.00±0.00 * | 0.60±0.60 a A | 0.00±0.00 * |
| Maíz convencional | 1.60±0.50 a A | 0.40±0.40 a A | 0.60±0.24 a A |
| Urochloa | 0.20±0.20 b B | 1.20±0.37 a B | 3.40±1.20 b A |

Las letras minúsculas comparan los cultivos dentro del factor Estadio (comparación dentro de las columnas), mientras que las letras mayúsculas comparan los Estadios dentro de cada cultivo (comparación dentro de las filas). * Dado que no hubo variabilidad en este tratamiento, no se considerará en las comparaciones.

## Anexo - Base de datos 1

| Host | Rep | y |
|---|---|---|
| Brachiaria | 1 | 13 |
| Brachiaria | 2 | 20 |
| Brachiaria | 3 | 24 |
| Brachiaria | 4 | 17 |
| Brachiaria | 5 | 22 |
| Brachiaria | 6 | 2 |
| Brachiaria | 7 | 17 |
| Brachiaria | 8 | 15 |
| Brachiaria | 9 | 26 |
| Brachiaria | 10 | 19 |
| Crotalaria | 1 | 36 |
| Crotalaria | 2 | 50 |
| Crotalaria | 3 | 1 |
| Crotalaria | 4 | 34 |
| Crotalaria | 5 | 2 |
| Crotalaria | 6 | 48 |
| Crotalaria | 7 | 6 |
| Crotalaria | 8 | 45 |
| Crotalaria | 9 | 27 |
| Crotalaria | 10 | 6 |
| Algodón | 1 | 216 |
| Algodón | 2 | 180 |
| Algodón | 3 | 197 |
| Algodón | 4 | 124 |
| Algodón | 5 | 180 |
| Algodón | 6 | 281 |
| Algodón | 7 | 203 |
| Algodón | 8 | 100 |
| Algodón | 9 | 228 |
| Algodón | 10 | 239 |
| Soya | 1 | 29 |
| Soya | 2 | 4 |
| Soya | 3 | 61 |
| Soya | 4 | 50 |
| Soya | 5 | 34 |
| Soya | 6 | 77 |
| Soya | 7 | 84 |
| Soya | 8 | 70 |
| Soya | 9 | 103 |
| Soya | 10 | 71 |
| Maíz | 1 | 222 |
| Maíz | 2 | 218 |

| | | |
|---|---|---|
| Maíz | 3 | 373 |
| Maíz | 4 | 400 |
| Maíz | 5 | 478 |
| Maíz | 6 | 454 |
| Maíz | 7 | 400 |
| Maíz | 8 | 263 |
| Maíz | 9 | 472 |
| Maíz | 10 | 382 |
| Gandul | 1 | 2 |
| Gandul | 2 | 2 |
| Gandul | 3 | 4 |
| Gandul | 4 | 0 |
| Gandul | 5 | 2 |
| Gandul | 6 | 0 |
| Gandul | 7 | 5 |
| Gandul | 8 | 3 |
| Gandul | 9 | 0 |
| Gandul | 10 | 1 |
| Mucuna Negra | 1 | 0 |
| Mucuna Negra | 2 | 0 |
| Mucuna Negra | 3 | 0 |
| Mucuna Negra | 4 | 2 |
| Mucuna Negra | 5 | 4 |
| Mucuna Negra | 6 | 0 |
| Mucuna Negra | 7 | 1 |
| Mucuna Negra | 8 | 1 |
| Mucuna Negra | 9 | 5 |
| Mucuna Negra | 10 | 3 |
| Sorgo | 1 | 2 |
| Sorgo | 2 | 0 |
| Sorgo | 3 | 5 |
| Sorgo | 4 | 4 |
| Sorgo | 5 | 0 |
| Sorgo | 6 | 4 |
| Sorgo | 7 | 0 |
| Sorgo | 8 | 0 |
| Sorgo | 9 | 1 |
| Sorgo | 10 | 0 |
| Girasol | 1 | 2 |
| Girasol | 2 | 0 |
| Girasol | 3 | 1 |
| Girasol | 4 | 0 |
| Girasol | 5 | 1 |
| Girasol | 6 | 0 |
| Girasol | 7 | 0 |
| Girasol | 8 | 0 |

| | | |
|---|---|---|
| Girasol | 9 | 1 |
| Girasol | 10 | 1 |
| Barbecho | 1 | 0 |
| Barbecho | 2 | 0 |
| Barbecho | 3 | 1 |
| Barbecho | 4 | 0 |
| Barbecho | 5 | 1 |
| Barbecho | 6 | 0 |
| Barbecho | 7 | 0 |
| Barbecho | 8 | 0 |
| Barbecho | 9 | 0 |
| Barbecho | 10 | 0 |

## Anexo – Banco de Dados 2

| Estadio | Cultivo | Bloque | SfrugTot |
|---|---|---|---|
| V5 | Maizconv | 1 | 2 |
| V5 | Maizconv | 2 | 0 |
| V5 | Maizconv | 3 | 1 |
| V5 | Maizconv | 4 | 2 |
| V5 | Maizconv | 5 | 3 |
| V5 | MaizBt | 1 | 0 |
| V5 | MaizBt | 2 | 0 |
| V5 | MaizBt | 3 | 0 |
| V5 | MaizBt | 4 | 0 |
| V5 | MaizBt | 5 | 0 |
| V5 | Urochloa | 1 | 1 |
| V5 | Urochloa | 2 | 0 |
| V5 | Urochloa | 3 | 0 |
| V5 | Urochloa | 4 | 0 |
| V5 | Urochloa | 5 | 0 |
| V7 | Maizconv | 1 | 0 |
| V7 | Maizconv | 2 | 0 |
| V7 | Maizconv | 3 | 0 |
| V7 | Maizconv | 4 | 0 |
| V7 | Maizconv | 5 | 2 |
| V7 | MaizBt | 1 | 3 |
| V7 | MaizBt | 2 | 0 |
| V7 | MaizBt | 3 | 0 |
| V7 | MaizBt | 4 | 0 |
| V7 | MaizBt | 5 | 0 |
| V7 | Urochloa | 1 | 1 |
| V7 | Urochloa | 2 | 2 |

| | | | |
|---|---|---|---|
| V7 | Urochloa | 3 | 1 |
| V7 | Urochloa | 4 | 2 |
| V7 | Urochloa | 5 | 0 |
| V8 | Maizconv | 1 | 1 |
| V8 | Maizconv | 2 | 1 |
| V8 | Maizconv | 3 | 0 |
| V8 | Maizconv | 4 | 0 |
| V8 | Maizconv | 5 | 1 |
| V8 | MaizBt | 1 | 0 |
| V8 | MaizBt | 2 | 0 |
| V8 | MaizBt | 3 | 0 |
| V8 | MaizBt | 4 | 0 |
| V8 | MaizBt | 5 | 0 |
| V8 | Urochloa | 1 | 7 |
| V8 | Urochloa | 2 | 3 |
| V8 | Urochloa | 3 | 0 |
| V8 | Urochloa | 4 | 2 |
| V8 | Urochloa | 5 | 5 |

## 7. Referencias de paquetes usados

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

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

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

[4] Moral, R.A., Hinde, J., Demétrio, C.G.B. Half-normal plots and overdispersed models in R: The hnp package. Journal of Statistical Software, v. 81, n. 10, 2017.

[5] Demétrio, C.G.B. Modelos lineares generalizados em experimentação agronômica. USP/ESALQ, 2001.

[6] Demétrio, C.G., Hinde, J., Moral, R.A. Models for overdispersed data in entomology. In: Ecological modelling applied to entomology. Springer, Cham, 2014. p. 219-259.

[7] Hothorn, T., Bretz F., Westfall P. Simultaneous inference in general parametric models. Biometrical Journal 50(3), 346--363. 2008.

[8] Grothendieck, G. sqldf: Manipulate R Data Frames Using SQL. R package version 0.4-11. https://CRAN.R-project.org/package=sqldf. 2017.

## 8. Referencias recomendadas

Demétrio, C.G.B. Modelos lineares generalizados em experimentação agronômica. USP/ESALQ, 2001.

Demétrio, C.G., Hinde, J., Moral, R.A. Models for overdispersed data in entomology. In: Ecological modelling applied to entomology. Springer, Cham, 2014. p. 219-259.










MERIDA
PUBLISHERS
www.meridapublishers.com