

# Una Introduccion a R

**Edgar Acuna**

**([academic.uprm.edu/eacuna/introRacuna2018.pdf](http://academic.uprm.edu/eacuna/introRacuna2018.pdf))**

**Departament of Mathematics**

**University of Puerto Rico at Mayaguez**

**Enero 2020**

# Contenido-1

- Introduccion
- Interfaces graficos para R
- Personalizando R y Rstudio
- Operadores aritmeticos, relacionales, logicos y de asignacion
- Funciones numericas y de caracteres
- Vectores, Factores, Matrices y DataFrames
- Construyendo funciones en R
- Uso y construccion de Librerias(Paquetes)

# Contenido-2

- Leyendo datos en R y RStudio
- Conexion a base de datos usando R
- Graficas :GGplot2 y ggplotgui
- Funciones Estadisticas Basicas
- Estadistica Inferencial
- Modelos de Prediccion
- Preprocesamiento de datos con dyplr
- Manipulacion de Datos temporales con lubridate
- Tidyverse
- Haciendo aplicaciones de R en la web: Shiny

# Introduccion

R (Ihaka and Gentleman, 1994) es una implementacion gratuita del programa de computacion estadistica, S, el cual se origino a principios de los 80's. S-Plus una implementacion comercial de S que incluye un interface grafico(GUI), estuvo disponible desde los inicios de los 90's hasta el 2010 aprox.

R mayormente usa comandos de linea e incluye un limitado GUI. Hay varias propuestas para GUI's en R, siendo Rstudio, y Rcmdr los mas usados.

R tiene excelente capacidades de graficas.

R esta disponible para Windows, MacOS. Unix/Linux.

R tiene muy buena documentacion y ayuda disponible.

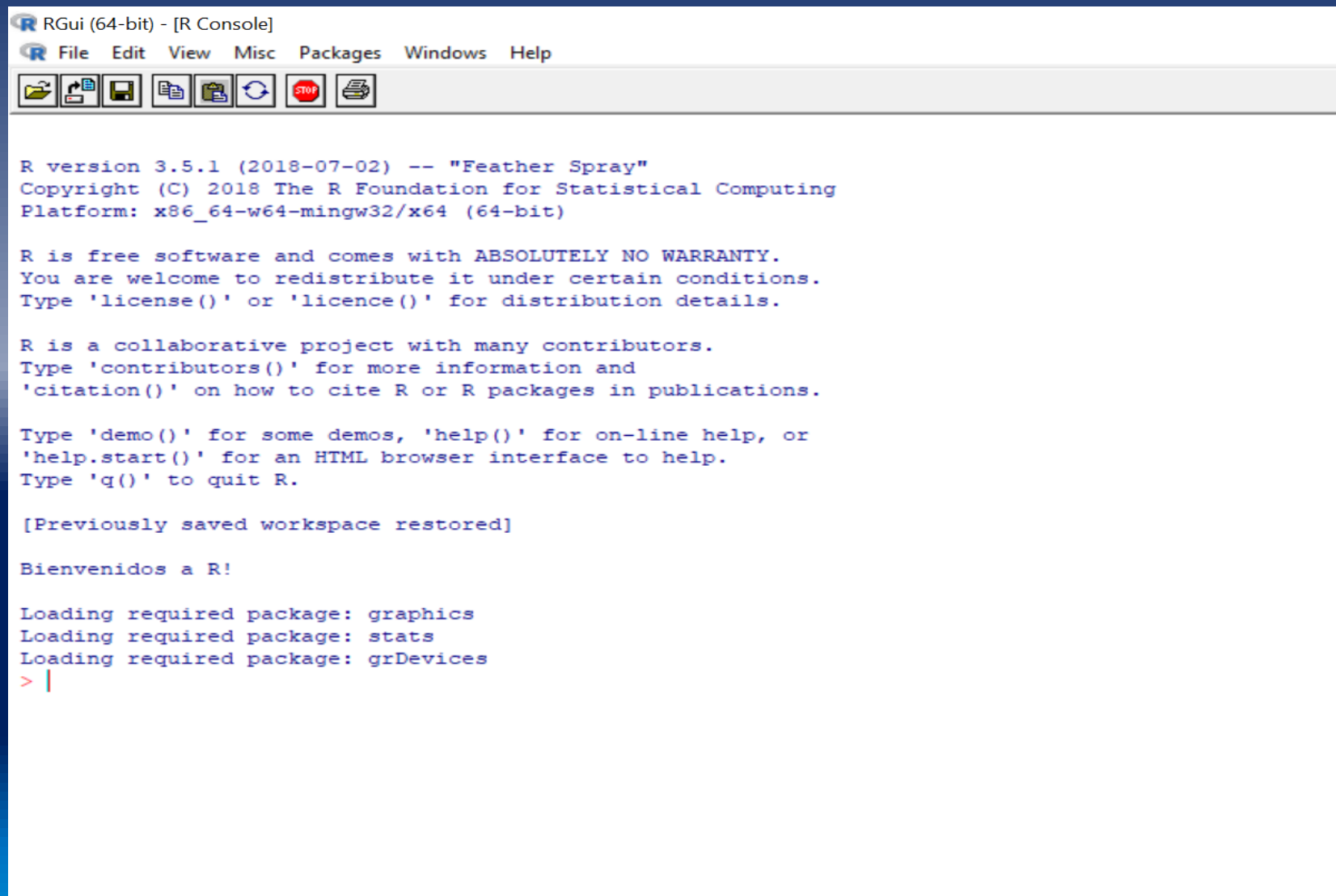
# Porque usar R?

- Los metodos estadisticos mas recientes aparecen primero en R.
- Existen muchas librerias disponibles (12,897, Enero 2018) para aplicar diversos metodos estadisticos.
- Crea excelente graficas con relativa facilidad.
- Es facil de usar.
- Puede leer datos de diferente sistemas de bases de datos (SQL, Oracle,etc) y en diferentes formatos csv, xml, json.
- Puede interactuar con muchos lenguajes de Programacion: Java, C++, Python, etc.
- Es gratis.

# Instalando R

- 1-Entrar Website: [cran.r-project.org](https://cran.r-project.org).
- 2-Escoger el sistema operativo en donde va a usar R: Linux, MacOS o Windows.
3. En la pantalla R for Windows escoger el subdirectorio base
4. En la pantalla R-3.6.2 for Windows hacer un click a Download R-3.6.2 for Windows para descargar el archivo R-3.6.2-win.exe
5. Localizar el archivo R-3.6.2-win.exe en su computadora y ejecutarlo eso instalara R

# El ambiente grafico de R



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help

R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

Bienvenidos a R!

Loading required package: graphics
Loading required package: stats
Loading required package: grDevices
> |
```

# Ambientes graficos (GUI) de R

1-R commander Rcmdr es un paquete que se instala dentro de R (John Fox, 2007)

2-Para instalar Rstudio entrar a [www.rstudio.com](http://www.rstudio.com)  
( Hadley Wickham, 2011)

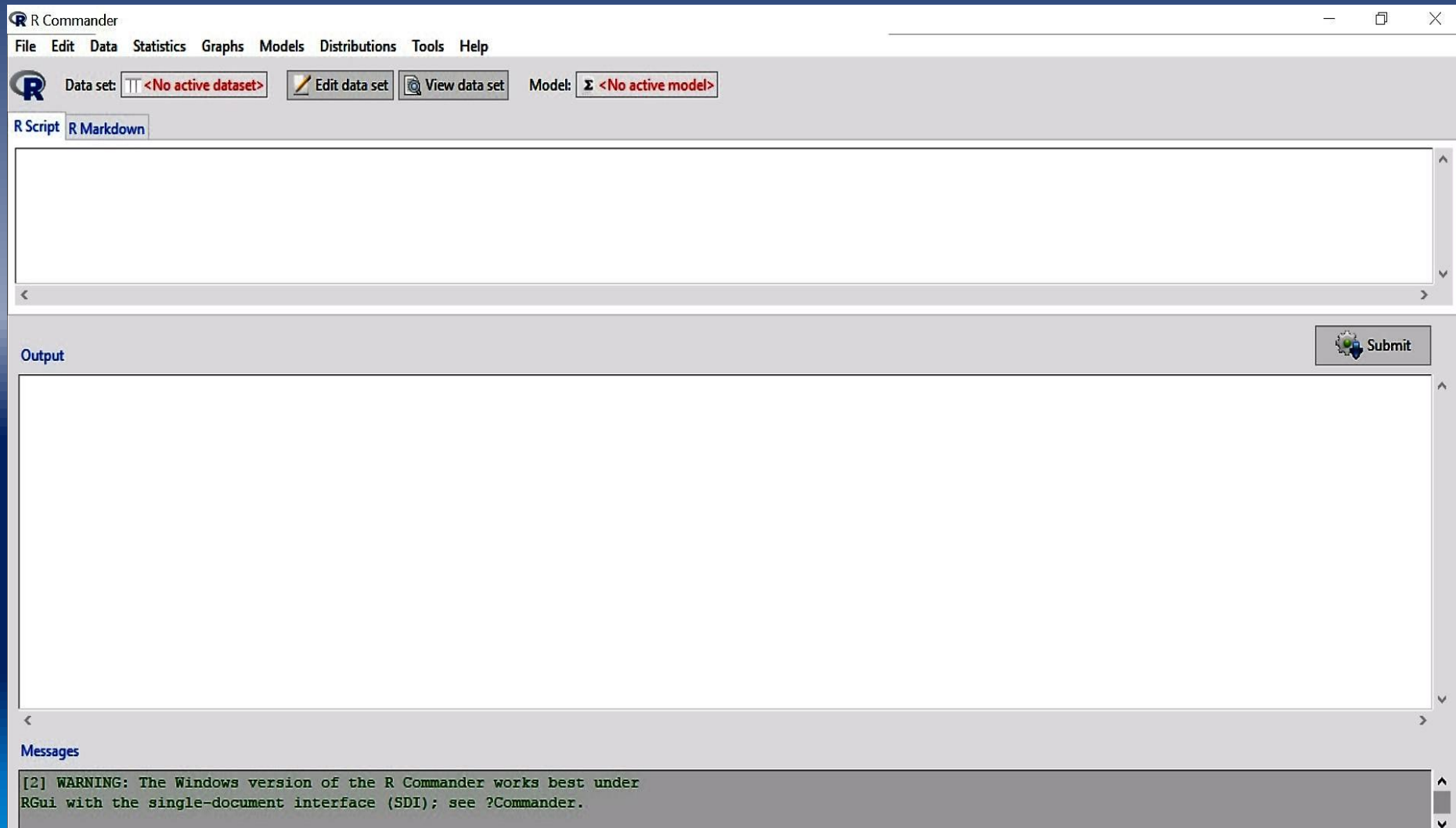
3. Para instalar Deducer entrar a [www.deducer.org](http://www.deducer.org). (Ian Fellows, 2011)

4. Para instalar R Analytic Flow entrar a <https://r.analyticflow.com/en/> (Ef-Prime Inc desde 2007)

De todos ellos Rstudio es el que ha ganado mas popularidad



# El ambiente grafico Rcmdr de R



# Usando el gui R Commander

The screenshot displays the R Commander interface. The top menu bar includes File, Edit, Data, Statistics, Graphs, Models, Distributions, Tools, and Help. Below the menu, the 'Data set:' field shows 'bupa' with buttons for 'Edit data set' and 'View data set'. The 'Model:' field shows '<No active model>'. The main workspace is divided into three panes: 'R Script' (empty), 'Output' (empty), and 'Messages' (containing two messages: [1] NOTE: R Commander Version 2.4-1: Thu Jan 11 19:03:30 2018 and [2] WARNING: The Windows version of the R Commander works best under RGui with the single-document interface (SDI); see ?Commander.). A 'bupa' data viewer window is open, showing a table with 30 rows and 7 columns (V1-V7). A 'Submit' button is visible on the right side of the interface.

	V1	V2	V3	V4	V5	V6	V7
1	85	92	45	27	31	0.0	1
2	85	64	59	32	23	0.0	2
3	86	54	33	16	54	0.0	2
4	91	78	34	24	36	0.0	2
5	87	70	12	28	10	0.0	2
6	98	55	13	17	17	0.0	2
7	88	62	20	17	9	0.5	1
8	88	67	21	11	11	0.5	1
9	92	54	22	20	7	0.5	1
10	90	60	25	19	5	0.5	1
11	89	52	13	24	15	0.5	1
12	82	62	17	17	15	0.5	1
13	90	64	61	32	13	0.5	1
14	86	77	25	19	18	0.5	1
15	96	67	29	20	11	0.5	1
16	91	78	20	31	18	0.5	1
17	89	67	23	16	10	0.5	1
18	89	79	17	17	16	0.5	1
19	91	107	20	20	56	0.5	1
20	94	116	11	33	11	0.5	1
21	92	59	35	13	19	0.5	1
22	93	23	35	20	20	0.5	1
23	90	60	23	27	5	0.5	1
24	96	68	18	19	19	0.5	1
25	84	80	47	33	97	0.5	1
26	92	70	24	13	26	0.5	1
27	90	47	28	15	18	0.5	1
28	88	66	20	21	10	0.5	1
29	91	102	17	13	19	0.5	1
30	87	41	31	19	16	0.5	1

**File:** Menu de opciones para cargar y guardar archivos log/script. Guardar las salidas y el espacio de trabajo de R y salir.

**Edit:** Opciones para editar el contenido de las ventanas output y log/script..

**Data:** contiene opciones para leer y manipular datos.

**Statistics:** Submenus conteniendo opciones para analisis estadistico basico.

**Graphs:** Contiene opciones para crear graficas estadisticas.

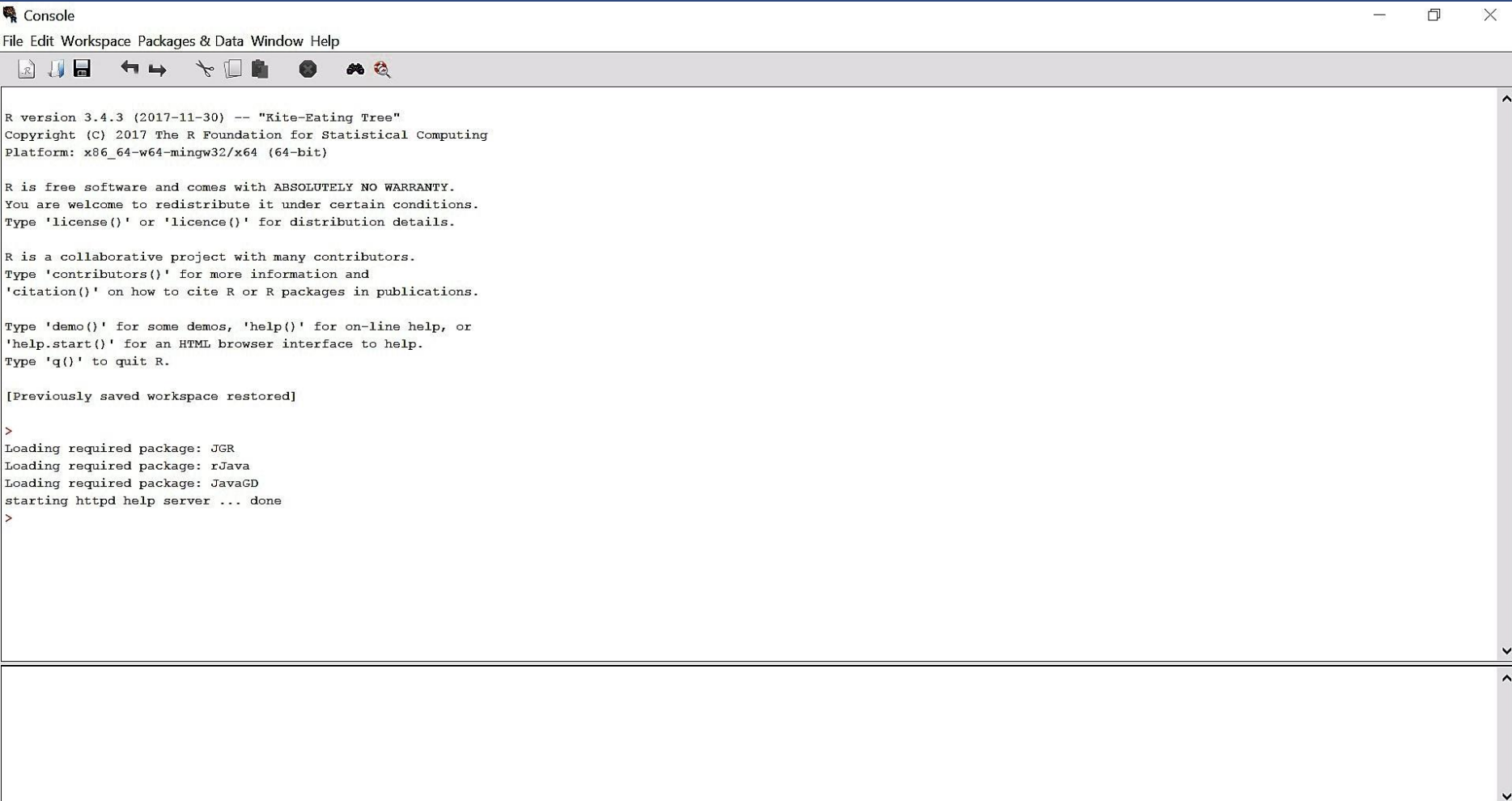
**Models:** Contiene opciones para obtener resúmenes numéricos, hacer pruebas de hipótesis, intervalos de confianza y modelos de regresión.

**Distributions:** Contiene opciones para calcular probabilidades, obtener cuantiles, and graficas de distribuciones estadísticas conocidas.

**Tools:** Menu de opciones para cargar paquetes de R y modificar opciones de Rcmdr

**Help:** Menu de opciones para obtener informacion acerca del Rcmdr

# El ambiente grafico Deducer de R



The screenshot shows the R Console window with the following text:

```
R version 3.4.3 (2017-11-30) -- "Kite-Eating Tree"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

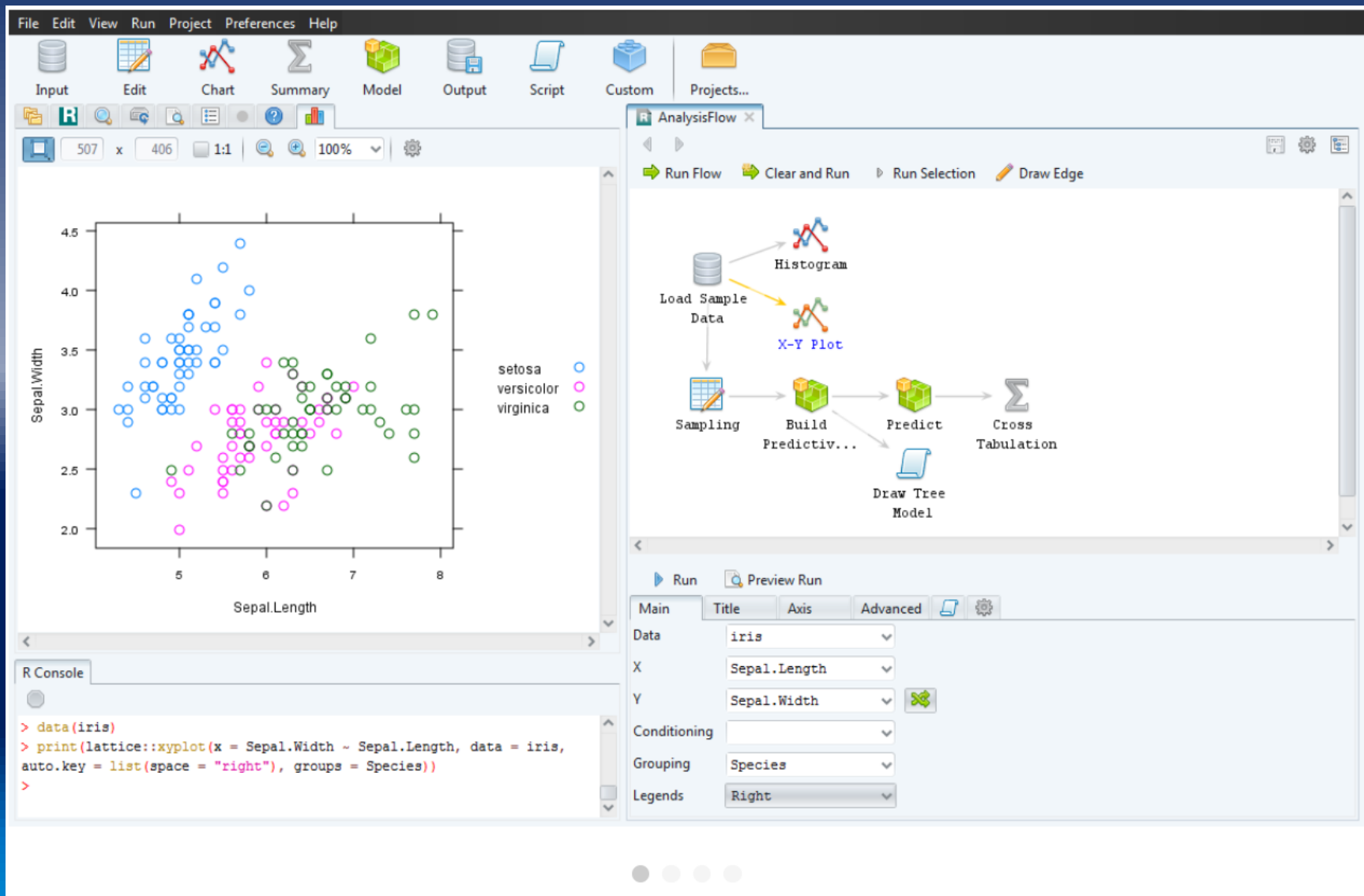
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

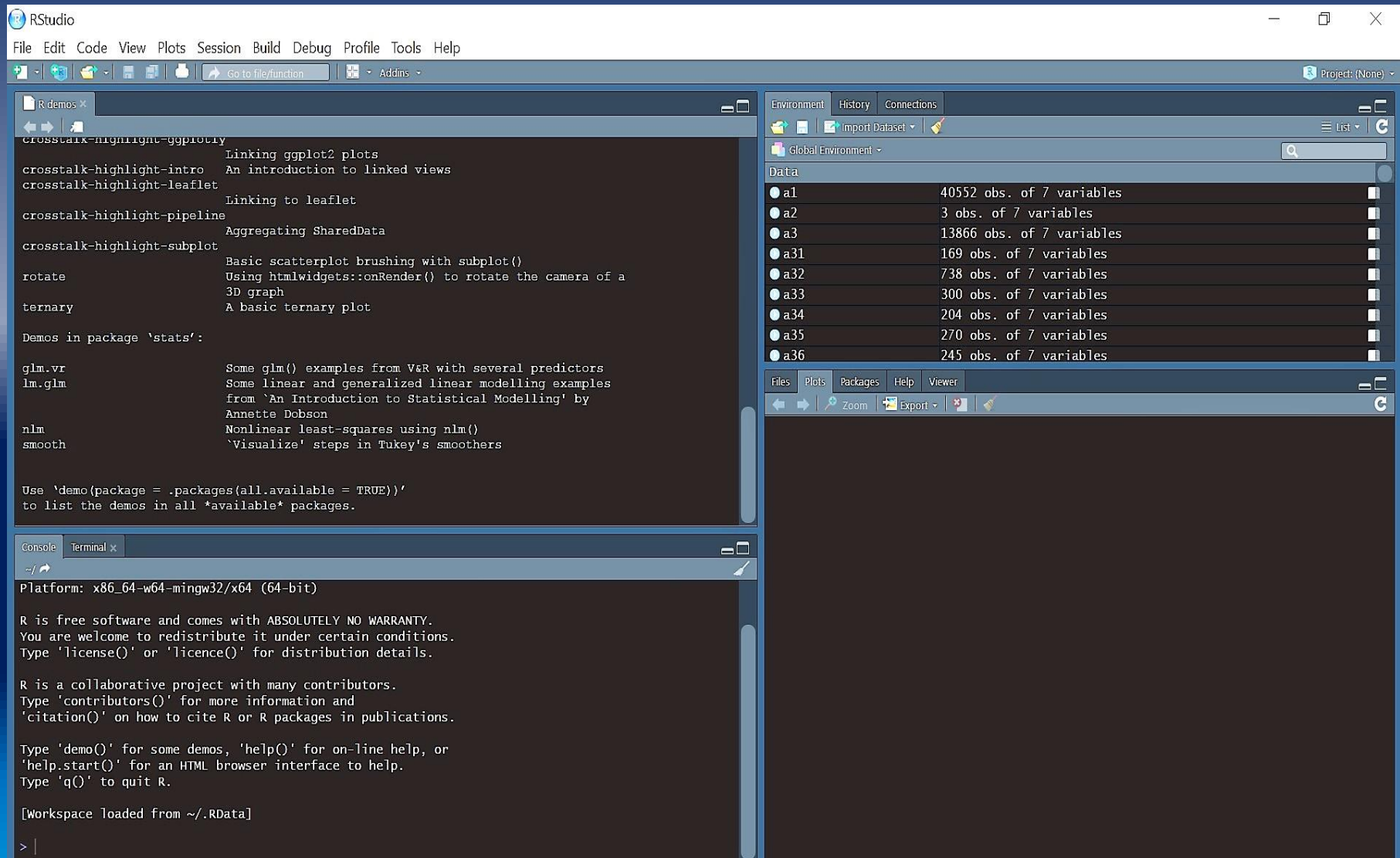
[Previously saved workspace restored]

>
Loading required package: JGR
Loading required package: rJava
Loading required package: JavaGD
starting httpd help server ... done
>
```

# El ambiente grafico de RAnalyticF



# El entorno de Desarrollo (IDE) Rstudio



# Microsoft R Open

Microsoft R Application Network

Home About R Microsoft R Open R Packages R Community R Tools

Find an R Package

## R Packages

Search all MRAN(13,494 CRAN) packages  All

Display as: ☒ List View ☐ Tile View

Package Name	Published	Authors	Title	Vignettes	Task View
<a href="#">dprep</a>	2015-11-24	Edgar Acuna and the CASTLE research group at The...	Data Pre-Processing and Visualization Functions for Classification	0	

Previous 1 Next Showing 1 to 1 of 1 packages (13494 total entries)

Microsoft R Open

- About
- Download
- Install
- Release History

Packages

- Explore Packages
- Explore Task Views
- CRAN Time Machine

Connect

- Contact
- Licenses

# Personalizando R: Rprofile.site, .Rprofile, .First

Orden	File/Funcion	Que hace y donde esta
1	Rprofile.site	Es un file de texto en directorio <code>...\etc\</code> de su disco duro y que contiene comandos de R que son ejecutados cada vez que se llama a R. El simbolo <code>"..."</code> se refiere al directorio que contiene el programa R.
2	.Rprofile	Es un file de texto en directorio <code>home</code> de R que contiene comandos que son ejecutados cada vez que se llama a R. El directorio <code>home</code> se halla con el R comando <code>Sys.getenv()["HOME"]</code>
3	.First	Es un funcion en el espacio de trabajo de R y que contiene comandos que son ejecutados cada vez que se llama a R.



# Ejemplo Rprofile.site

```
# Comentario: Opciones que se pueden cambiar

# options(papersize="a4")
# options(editor="notepad")
# options(pager="internal")

# set the default help type
# options(help_type="text")
options(help_type="html")

# set a site library
# .Library.site <- file.path(chartr("\\", "/", R.home()), "site-library")

# set a CRAN mirror
# local({r <- getOption("repos")
#   r["CRAN"] <- "http://my.local.cran"
#   options(repos=r)})
#Dar el commando ?Startup para mayor informacion
```

# Ejemplo de function .First

```
.First=function(){  
  #Imprime um aviso de Bienvenida en la consola seguido  
  de dos lineas vacias  
  cat("Bienvenidos a R!\n\n")  
  options(digits=5)  
  library(rpart)}
```

# Personalizando RStudio

Las opciones para configurar RStudio se obtienen siguiendo la secuencia **Tools > Global Options** e incluye las siguientes categorías:

**General:** Default CRAN mirror, initial working directory, workspace and history behavior.

**Source Code Editing:** Enable/disable line numbers, selected word, line and console syntax highlighting; configure tab spacing; set default text encoding.

**Appearance and Themes:** Specify font size and visual theme for the console and source editor.

**Pane Layout:** Locations of console, source editor, and tab panes; set which tabs are included in each pane.

**Packages:** Set default CRAN repository

**Sweave:** Configure Sweave compiling options and PDF previewing.

**Spelling:** Choose main dictionary language and specify spell checking options.

**Publishing:** Enable publishing apps and documents from IDE. Set account.

# Operadores aritmeticos

$2+3$  #Suma

$2-3$  #Resta

$2*3$  #Producto

$2/3$  # Division

$2^3$  #Potencia

$2**3$  #Potencia

$5\%\%2$  #Modulo

$(4^2) - (3*2)$  #Operaciones combinadas

$2^{-3}$

# Operadores de asignacion

`<-`, `<<-`, `=`    Asignacion hacia la izquierda  
`->`, `->>`        Asignacion hacia la derecha

Ejemplos:

`x<-3`

`x=5.9`

`8-> x`

`15.3 ->>`

# Operadores relacionales

< #Menor que

<= #Menor o igual que

> #Mayor que

>= #Mayor o igual que

= #Exactamente Igual a

!= #No igual a

Ejemplo:

`gpa<-3.5`

`gpa>3`

# Operadores logicos

`!x` #No x

`x | y` # x o y, o logico elemento por elemento

`a & b` # a y b, y logico elemento por elemento

#Cero es considerado Falso y No-Cero es Cierto

`isTRUE(x)` #Prueba si x es cierto

`a || b` # o logico solo compara los primeros elementos de los vectores a y b y su resultado es un vector logico de un elemento

`a && b` # y logico

# Imprimiendo valores en la consola de R

```
> x<-"This is a string"
```

```
> x
```

```
[1] "This is a string"
```

```
> print(x)
```

```
[1] "This is a string"
```

```
> print(x,quote=F)
```

```
[1] This is a string
```

```
> cat(x,"\n")
```

```
This is a string
```



# Funciones

- Todos los calculos en R son realizados mediante el llamado a funciones
- Una llamada a una funcion en R recibe ningun o varios argumentos y devuelve uno o mas valores.
- Para anadir mayor funcionalidad a R un usuario deberia escribir sus propias funciones.
- Funciones definidas por usuarios tienen el mismo nivel que las funciones que vienen con R.

# Funciones numericas-1

```
> sqrt(9) #raíz cuadrada  
[1] 3  
> exp(3) #exponencial  
[1] 20.08554  
> cos(pi) #coseno  
[1] -1  
> sin(pi/2) #seno  
[1] 1  
> tan(pi/4) #tangente  
> abs(-3.5) #valor absoluto  
[1] 3.5
```

# Funciones numericas-2

```
> ceiling(9.23) #El próximo entero que sigue al numero
[1] 10
> floor(3.71) #El entero que antecede al numero
[1] 3
> trunc(3.89) #redondea
[1] 4
> round(pi,2) #redondea a dos decimales
[1] 3.14
> log(10) #logaritmo natural, base=e
[1] 2.3026
> log(10,b=10) #logaritmo en base 10
[1] 1
```

# Funciones para caracteres-1

`substr(x, start=n1, stop=n2)` #extrae o reemplaza sub-strings  
del string x

```
x <- "abcdef"
```

```
> substr(x, 2, 4)
```

```
[1] "bcd"
```

```
> substr(x, 2, 4) <- "222"
```

```
> x
```

```
[1] "a222ef"
```

`toupper(x)` # escribe el string x en mayusculas

`tolower(x)` # escribe el string x en minusculas

# Funciones para caracteres-2

`strsplit("abc", split="")` #Divide un string en substrings de acuerdo al simbolo que aparece en string

```
[[1]]
```

```
[1] "a" "b" "c"
```

`paste( ...)` #concatena los argumentos para generar un string que es una combinacion de todos ellos, de acuerdo al argumento `sep` o `collapse`

```
> paste('x',1:10,sep="")
```

```
[1] "x1" "x2" "x3" "x4" "x5" "x6" "x7" "x8" "x9" "x10"
```

```
> paste(c('este','es','un','ejemplo'),collapse="")
```

```
[1] "esteesun ejemplo"
```

```
> paste(c('este','es','un','ejemplo'),collapse="-")
```

```
[1] "este-es-un-ejemplo"
```

```
> paste(c('este','es','un','ejemplo'),collapse=" ")
```

```
[1] "este es un ejemplo"
```

# Funciones para manipular el workspace

```
ls() #Lista todos las variables(objetos) en el workspace  
rm(x) # remueve la variable(objeto) x del workspace  
rm(list=ls()) # remueve todas las variables del workspace  
getwd() # Informa acerca del directorio home de R
```

# Usando la ayuda de R

```
help(plot)
```

```
?plot
```

`help.search("plot")` #lista todas las funciones que tiene el string "plot". Un comando similar es

```
apropos("plot")
```

Tambien hay un menu de help, en donde hay manuales en formatos pdf e informacion acerca de comandos y paquetes en formato html.

```
help(package=Rcmdr) # da ayuda acerca el paquete Rcmdr.
```

# La ventana de ayuda para plot (vista parcial)

plot

package:graphics

R Documentation

## Generic X-Y Plotting

### Description:

Generic function for plotting of R objects. For more details about the graphical parameter arguments, see 'par'.

### Usage:

```
plot(x, y, ...)
```

### Arguments:

x: the coordinates of points in the plot. Alternatively, a single plotting structure, function or any R object with a 'plot' method\_ can be provided.

y: the y coordinates of points in the plot, optional if 'x' is an appropriate structure.



# Estructuras de Control

Permiten controlar el flujo de la ejecución de un script típicamente dentro de una función en R. Entre las más usadas están:

- if, else
- for
- while
- repeat
- break
- next
- return

# If,else

```
if (condition) {  
  # do something  
} else {  
  # do something else  
}
```

Ejemplo:

```
> GPA=2.95  
> if(GPA>3.00){  
  print("Graduando es contratado")  
} else {  
  print("Graduando no es contratado")  
}  
[1] "Graduando no es contratado"
```

# If,else-2

```
gpa = 1.4
if(gpa >= 2.5)
{cat("Bienvenido al Colegio!")}else
{cat("su solicitud fue denegada")}
```

**Notar que el else tiene que estar en la misma linea del bracket que Cierra el if**

```
Ana=3
```

```
Rosa=25
```

```
if (Ana <= 5 && Rosa >= 10 || Rosa == 500 && Ana != 5)
{print ("Ana and Rosa")}
```

# for

```
x <- c( "Paulo", "Renato", "Camila", "Ana")  
for (i in 1:4) {  
  print(x[i])  
}  
[1] "Paulo"  
[1] "Renato"  
[1] "Camila"  
[1] "Ana"  
for (a in x) {  
  print(a)}  
[1] "Paulo"  
[1] "Renato"  
[1] "Camila"  
[1] "Ana"
```

# For-2

```
for(x in 1:4){  
  cat( x,"al cuadrado es", x^2, "\n") }
```

Output:

```
1 al cuadrado es 1  
2 al cuadrado es 4  
3 al cuadrado es 9  
4 al cuadrado es 16
```

```
x=c(3,7,12)  
for( i in x){  
  cat( i,"al cuadrado es", i^2, "\n") }
```

```
3 al cuadrado es 9  
7 al cuadrado es 49  
12 al cuadrado es 144
```

# While-1

```
i<-1
while(i<5){
  print(i^2)
  i<-i+1
}
[1] 1
[1] 4
[1] 9
[1] 16
```

# While-2

```
number = 1
while(number < 200){
    print(number)
    number = number * 2}
```

1] 1

[1] 2

[1] 4

[1] 8

[1] 16

[1] 32

[1] 64

[1] 128

# Repeat, break

```
x <- 1
repeat {
  print(x)
  x = x+1
  if (x == 6){
    break
  }
}
```



# Vectores-1

```
x=c(1,2,3,4) #combinado varios valores
```

```
x=1:4      # vector formado por una secuencia
```

```
x=rep(1,4) # vector de 4 unos
```

```
x=seq(2, 8, by=2) # vector:2,4,6,8
```

```
x=seq(0, 1, length=11) #vector desde 0 ... hasta 1.0
```

#Tambien se puede copiar datos de EXCEL o WORD usando la function scan

```
x=scan()
```

```
> x=scan()
```

```
1: 6
```

```
2: 7
```

```
3: 4
```

```
4:
```

```
Read 3 items
```

# Vectores-2

`x[2]`      # el segundo elemento del vector `x`  
`x[c(2,4,6)]` # vector conteniendo los elementos 2,4 y 6 de `x`  
`x[-c(1,3)]` # vector sin incluir los elementos 1 y 3 de `x`  
`x[x < 4]`    # vector que contiene los elementos de `x` t.q  $x < 4$ .  
`x[x != 4]` # vector que contiene los elementos de `x` distintos de 4.  
`y=x/2`     # divide los elementos del vector `x` por 2  
`z=x+y`    #suma los vectores `x` y `y`  
`log(x, 10)` #logaritmos en base 10  
`y = sqrt(x)` #raiz cuadrada

# Vectores-3

```
> x=c(10,9,3,17,89,19)
```

```
> sort(x) #ordena los datos en forma creciente
```

```
[1] 3 9 10 17 19 89
```

```
> sort(x, decreasing=T) # ordena los datos en forma decreciente
```

```
[1] 89 19 17 10 9 3
```

```
> rev(x) #retorna x en sentido contrario
```

```
[1] 19 89 17 3 9 10
```

```
> order(x) #da las posiciones de los datos ordenados de menor a mayor
```

```
[1] 3 2 1 4 6 5
```

```
> which(x>10) # Indica los indices de los valores de x que son mayores que 10
```

```
[1] 4 5 6
```

```
> y=c(4,5,5,9,4,6,2,2,4)
```

```
> unique(y) # reporta solo los valores distintos del vector y
```

```
[1] 4 5 9 6 2
```

```
> table(y) # cuenta veces se repite cada valor de y
```

# Factores

```
> colores=c("red","green","blue","red","blue") #vector de caracteres
> colores
[1] "red"  "green" "blue"  "red"  "blue"
> colores_factor=factor(colores) #convierte el vector en factor
> colores_factor
[1] red  green blue red  blue
Levels: blue green red
> summary(colores)
  Length  Class  Mode
    5 character character
> summary(factor(colores))
blue green  red
   2    1    2
```

# Matrices-1

```
x=c(1,4,3,7,5,8)
```

```
xmat=matrix(x,nrow=2,ncol=3) #convirtiendo el  
vector en matriz
```

```
xmat
```

	[,1]	[,2]	[,3]
[1,]	1	3	5
[2,]	4	7	8

```
dim(xmat) #Tamano de la matriz(filas y columnas)
```

```
xmat[1,2] # el elemento de la matriz en la posicion (1,2)
```

```
[1] 3
```

```
xmat[,3] # la tercera columna de la matriz
```

```
[1] 5 8
```

```
xmat[2,] # la segunda fila de la matriz
```

```
[1] 4 7 8
```

# Matrices-2

```
> addcol=c(9,2)
>
> newmat=cbind(xmat,addcol)
> newmat
```

				addcol
[1,]	1	5	7	9
[2,]	3	4	8	2

```
> mat2=newmat[,-c(3,4)] #elimina columnas 3 y 4
> mat2
```

[1,]	1	5
[2,]	3	4

# Matrices-3

```
> addrow=c(9,2,7)
> newmat=rbind(xmat,addrow)
> newmat
  V1 V2 V3
1  1  3  5
2  4  7  8
3  9  2  7
>mat2=newmat[-c(1,2),] #elimina  filas 1 y 2
> mat2
  V1 V2 V3
3  9  2  7
>m1=matrix(1:9,byrow=TRUE, nrow=3) #crea una matriz
3X3 con elementos del 1 al 9 entrados fila por fila
```

# Matrices-4

$m1+m2$  #suma de matrices

$t(m1)$  # transpuesta de una matriz

$t(m1)\%*\%m1$  # producto de matrices  $t(m1)$  y  $m1$

$\det(m1)$  # determinante de una matriz

$\text{solve}(m1)$  #inversa de una matriz

$\text{eigen}(m1)$  # produce los valores y vectores propios de la matriz  $m1$



# Matrices-5

`colSums(m1)` # suma de columnas

`rowSums(m1)` # suma de filas

`apply(m1,2,sum)` # suma de columnas

`apply(m1,1,sum)` # suma de filas

`apply(m1,1,max)` # maximo de las filas

`apply(m1,2,min)` # minimo de las columnas

`apply(m1,2,mean)` # media de las columnas

# Data-frames-1

Un data frame es una version extendida de una matriz, donde no solamente se pueden usar distinto tipos de variables no solamente numericas. Tambien puede tener missing values

El siguiente es el data frame **mtcars** que viene con R

```
> mtcars
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
.....											

# Data frames-2

Para crear un data frame se usa la function `data.frame()`

```
> nombre=c("Juan","Pedro","Rosa","Elena")
```

```
> profesion=c("Ingeniero","Medico","Sicologa","Abogada")
```

```
> Salario=c(50,90,40,80)
```

```
> personal=data.frame(cbind(nombre,profesion,Salario))
```

```
> personal
```

	nombre	profesion	Salario
--	--------	-----------	---------

1	Juan	Ingeniero	50
---	------	-----------	----

2	Pedro	Medico	90
---	-------	--------	----

3	Rosa	Sicologa	40
---	------	----------	----

4	Elena	Abogada	80
---	-------	---------	----

## Data frame-3

```
xmat=as.data.frame(xmat) #convierte la matriz xmat en data frame
```

```
xmat
```

```
  V1 V2 V3
```

```
1 85 86 87
```

```
2 85 91 98
```

```
> rownames(xmat) #nombres de las filas
```

```
[1] "1" "2"
```

```
> colnames(xmat) #nombres de las columnas
```

```
[1] "V1" "V2" "V3"
```

En un dataframe las columnas y las filas tienen nombres y se pueden hacer las mismas operaciones que con una matriz.

# Lists-1

Una lista es una coleccion de objetos de distinto tipo:

```
my.list=list(comp1,comp2,comp3)
```

```
# Vector con numeros del from 1 al 10
```

```
my.vector <- 1:10
```

```
# Matrix 3X3 con numeros del 1 al 9
```

```
my.matrix <- matrix(1:9, ncol = 3)
```

```
# Las primeras 3 filas del data frame mtcars
```

```
my.df <- mtcars[1:3,]
```

```
# Construyendo una lista con los tres componentes anteriores
```

```
My.list <- list(my.vector, my.matrix, my.df)
```

# Listas-2

```
#imprimiendo la lista
```

```
> My.list
```

```
[[1]]
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
[[2]]
```

```
  [,1] [,2] [,3]
```

```
[1,]  1   4   7
```

```
[2,]  2   5   8
```

```
[3,]  3   6   9
```

```
[[3]]
```

```
      mpg cyl  disp  hp drat   wt  qsec vs am gear carb
```

```
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46 0  1   4   4
```

```
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02 0  1   4   4
```

```
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61 1  1   4   1
```

# Listas-3

#Tambien se puede dar nombre a cada compoente de la lista

```
>my.list=list(matriz=my.matrix,cars=my.df)
```

```
>my.list
```

```
$`matriz`
```

```
  [,1] [,2] [,3]
```

```
[1,]  1   4   7
```

```
[2,]  2   5   8
```

```
[3,]  3   6   9
```

```
$cars
```

```
      mpg cyl  disp  hp drat   wt  qsec vs am gear carb
```

```
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46 0  1   4   4
```

```
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02 0  1   4   4
```

```
Datsun 710           22.8   4 108.0  93 3.85 2.320 18.61 1  1   4   1
```

# Corriendo scripts en R

Se puede escribir varias líneas de comandos y guardarlos en un archivo con la extensión R. Por ejemplo, el archivo comandos1.R contiene las siguientes líneas

```
x=c(1,2,3,4) # combinar
cat("\nEste es el vector x\n")
x
y=1:4      # vector formado por una secuencia
cat("El vector y es:\n")
print(y)
x1=seq(1,4) # vector de 4 unos
cat("\n El vector x1 es\n")
print(x1)
x2=seq(2, 8, by=2) # vector:2,4,6,8
cat("\n El vector x2 es",x2,"\n")
x3=seq(0, 1, length=11)
cat("El vector x3 es",x3,"\n")
```



# Corriendo scripts en R(cont)

Para correr el script, abrir R y cuando aparece el prompt escribir el comando

`Source("c://com1.R")` # escribir el path adecuado  
o del menu File elegir la opcion Source R code

Otra opcion es abrir Rstudio y en el Menu File elegir Open File y acceder al file com1.R; Luego marcar todas las lineas que quiere ejecutar y darle run or darle CTRL+SHIFT+ENTER para ejecutar todo. Los resultados apareceran en la ventana Console

# Construyendo funciones

```
moda=function(x)
{
#Funcion que encuentra la moda de un vector x
  m1=sort(table(x),decreasing=T)
  moda=names(m1[m1==m1[1]])
  moda=as.numeric(moda)
  return(moda)
}
> x1=c(2,3,4,4,5,2,3,3,8)
> moda(x1)
[1] 3
> x
[1] 1 3 4 5 3 2 4 5 7
> moda(x)
[1] 5 4 3
```

# Construyendo funciones-2

```
tablafreq=function (x)
{#Tabla de frecuencias para datos discretos
n=length(x)
frec.abs=table(x)
frec.rel.porc=table(x)*100/n
frec.abs.acum=cumsum(frec.abs)
frec.rel.acum=cumsum(frec.rel.porc)
tabla=cbind(frec.abs,frec.rel.porc,frec.abs.acum,frec.rel.acum)
return(tabla)
}
```

# Paquetes(librerias)

- Un Paquete es un conjunto de funciones que realizan ciertas tareas específicas y que han sido construidas por diversos usuarios de R.
- Hay mas de 11,000 paquetes disponibles en el website de R.
- La mayoría de ellos se instalan eligiendo primero el menu Packages y luego la opcion Install Packages from CRAN.
- La calidad y la cantidad de funciones incluidas en los paquetes varia bastante.

```
>library(Rcmdr) # carga el paquete Rcmdr
```

```
>help(package="paquete") # da ayuda de como usar el paquete
```

# Algunos paquetes disponibles

<u>fBasics</u>	Financial Software Collection - fBasics
<u>foreign</u>	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase,
<u>cluster</u>	Funciones para hacer clustering
<u>dplyr</u>	Funciones para subsetting, summarizing, y juntar datasets
<u>ggplot2</u>	Funciones para hacer graficas de alta calidad
<u>lubridate</u>	Funciones para trabajar facilnente con datos que tienen fecha y hora
<u>manipulater</u>	Funciones para plots interactivos en Rstudio
<u>Rcmdr</u>	R Commander
<u>Rcpp</u>	Funciones para llamar en R programs escritos en c++
<u>vcd</u>	Visualizing Categorical Data

# Construyendo librerías

Aquí solo explicaremos como hacer una librería local. Hacer una librería para ponerlo en el cran es mas tedioso

1. Descargar Rtools del cran de R
2. Modificar su path. Abrir Control Panel, luego ir a System and Security y luego a System > Advanced System Settings > Environment Variables. Hallar la variable "Path" y añadir C:\Program Files\R\R\ - 3.3\bin\x64;C:\Rtools\bin

## 2. Construir el esqueleto (versión básica) del paquete

Supongamos que tiene varias funciones en su medio ambiente ("environment") de R, digamos "fun1", "fun2", ..., "funN" las cuales han sido corridas individualmente y que se las quiere ensamblar y ponerlas todas a la vez en una librería llamada "mipaquete", la cual va a estar localizada en el directorio ("c:\Rpaquetes". Supongamos además que se usan los conjuntos de datos "dat1", ..., "datN".

- a. Entrar al medio ambiente de R
- b. Escribir el siguiente comando de línea dentro de R

```
package.skeleton(name="mipaquete",  
list=c("fun1", "fun2", ..., "funN", "dat1, ... "datN"), path="c://Rpaquetes").
```

# Construyendo librerías

Por ejemplo

```
package.skeleton(name="mipaquete",  
list=c("moda","tablafreq"), path="c://Rpaquetes").
```

En el subdirectorío mipaquete van a crearse dos subdirectoríos R y man, en uno están los códigos de las funciones y en otra la ayuda

3- Finalmente se construye el paquete dando en la Ventana de terminal el comando

```
Rcmd build --binary mipaquete
```

Seguido del comando

```
Rcmd INSTALL mynewpackage_1.0.tar.gz
```

4- Después de esto se puede entrar a R y dar el comando  

```
library(mipaquete)
```

# La Libreria Dprep

Package: dprep

Type: Package

Title: Data Pre-Processing and Visualization Functions for Classification Version: 3.0.2

Date: 2015-11-14

Author: Edgar Acuna and the CASTLE research group at The University of Puerto Rico-Mayaguez

Maintainer: Edgar Acuna [edgar.acuna@upr.edu](mailto:edgar.acuna@upr.edu)

Description: Data preprocessing techniques for classification. Functions for normalization, handling of missing values, discretization, outlier detection, feature selection, and data visualization are included.

Depends: R ( $\geq 3.1.0$ ), graphics, stats Imports: MASS, e1071, class, nnet, rpart, FNN, StatMatch, rgl, methods

License: GPL

LazyLoad: yes

NeedsCompilation: yes

Packaged: 2015-11-24 00:51:58 UTC; Edgar

Repository: CRAN

Date/Publication: 2015-11-24 07:46:38

Built: R 3.4.0; x86\_64-w64-mingw32; 2017-06-21 03:40:56 UTC; windows

Archs: i386, x64



# Leyendo datos de un archivo

`read.table("c://esma3016/colon.txt")` #lee los datos que estan en colon.txt localizado en c://esma3016.

`clase=read.table("http://academic.uprm.edu/eacuna/clase97.txt", header=T)` #lee los datos clase97.txt que estan en mi pagina de internet y los guarda en el objeto clase. En la primera fila aparecen los nombres de las variables.

`head(clase)` # muestra las seis primeras filas de clase

	edad	sexo	escuela	programa	creditos	gpa	familia	hestud	htv
1	21	f	publ	biol	119	3.6	3	35	10
2	18	f	priv	mbio	15	3.6	3	30	10
3	.....								

`tail(clase)` #muestra las ultimas seis filas de clase

# Leyendo datos de un archivo (cont)

Otra forma de leer datos de la internet es usando la funcion `getURL` de la libreria `RCurl()`.

```
getURL("http://academic.uprm.edu/eacuna/clase97.txt")
```

- Tambien hay interfaces que permiten leer datos de otros programas estadisticos como SAS, SPSS y MINITAB.
- Para leer datos guardados Excel es mejor tenerlo en el fomato csv y usar el commando `read.csv("c://datos1.csv", sep="; ")`
- Otra alternativas para leer datos son usar las librerias `data.table`, `foreign`, `xlsx` o `RODBC`.
- Se pueden leer tablas de datos directamente de la internet usando las librerias `rvest` y `XML`.
- Cuando los datos estan en un paquete (libreria de R) se usa simplemente: `data(datospaq)`, donde `datospaq` es un conjunto de datos que viene con el paquete.

# Subsetting un dataframe

```
clase=read.table("http://academic.uprm.edu/eacuna/clase97.txt",header=T)
```

```
clase.pub=subset(clase,escuela=="pub") #solo los estudiantes de publica
```

```
clase.pubyf= subset(clase,escuela=="pub" & sexo=="f") #solo los estudiantes de publica y mujeres
```

```
clase.biologpa= subset(clase,programa=="biol" | gpa>3.25) #solo los estudiantes de biol o con gpa mayor de 3.25
```

# Subsetting un dataframe

#Extrae todos los estudiantes que son mujeres y que tienen promedio mas de 3.20

```
clase1=clase[which(clase$sexo=='f' & clase$gpa > 3.20),]
```

#Extrae todos los estudiantes que son mujeres o que tienen promedio

```
clase2=clase[which(clase$sexo=='f' | clase$gpa > 3.20),]
```

#divide al conjunto clase en dos los datos de las mujeres y los varones

```
split(clase,as.factor(clase$sexo))
```

# Funciones estadísticas básicas

```
x=c(18,24,17,23,23,21,19,18,24,21)
```

```
mean(x) #calcula la media
```

```
median(x) #calcula la mediana
```

```
var(x) #calcula la varianza de x
```

```
sd(x) #calcula la desviación estándar
```

```
quantile(x,prob=c(.1,.9)) # percentiles del 10 y 90%
```

# Funciones estadísticas básicas

`summary(x)` # calcula varias medidas estadísticas

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
------	---------	--------	------	---------	------

13.00	18.50	23.50	23.00	27.75	32.00
-------	-------	-------	-------	-------	-------

`sort(x)` #ordena los valores de x en forma creciente

`sort(x,decreasing=T)` # ordena en forma decreciente

`table(x)` #muestra las frecuencias absolutas de x

# Graficas estadisticas univariadas

```
edad=c(18,24,19,23,22,32,17,21,23,20)
```

```
hist(edad) #hace un histogram
```

```
boxplot(edad,horizontal=T) #Hace un diagrama de caja  
#Muestra las dos figuras en la misma pantalla
```

```
par(mfrow=c(1,2))
```

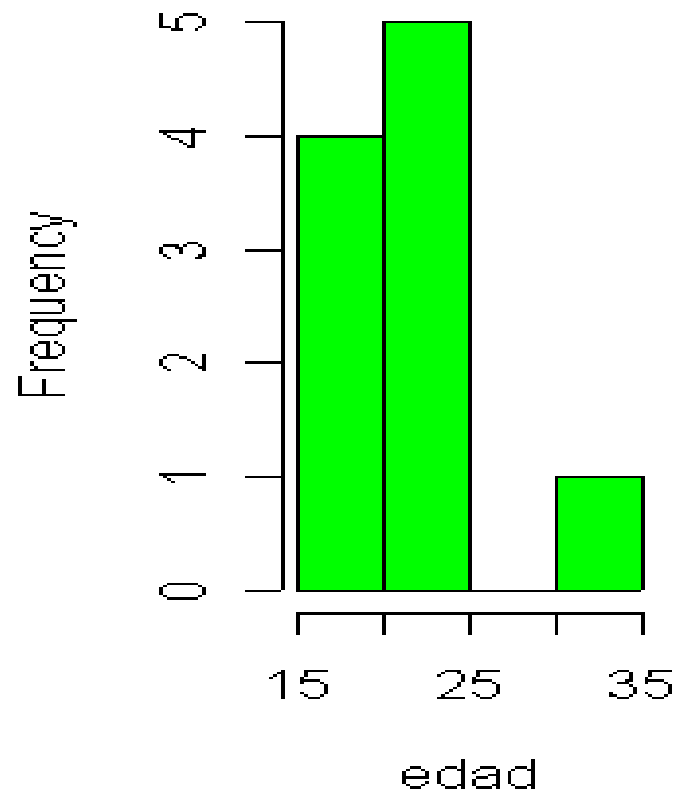
```
hist(edad,main="histograma de edad",col="green")
```

```
boxplot(edad,main="boxplot de edad", col=4)
```

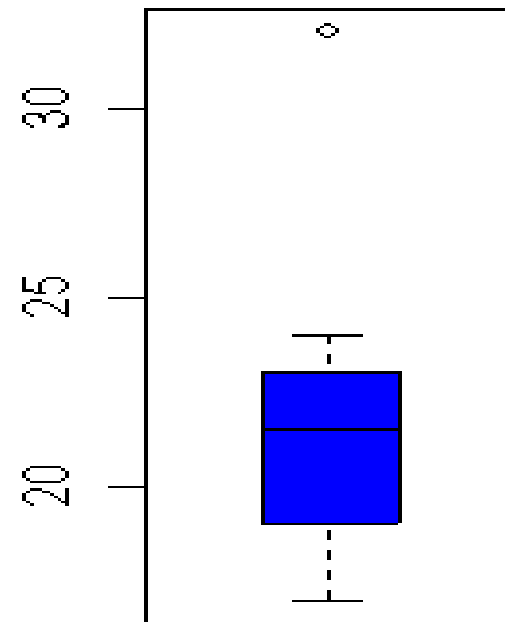
```
programas=c("bio","sico","adem","sico","bio","sico","adem","ade  
m","sico")
```

# Graficas estadísticas univariadas

**histograma de edad**

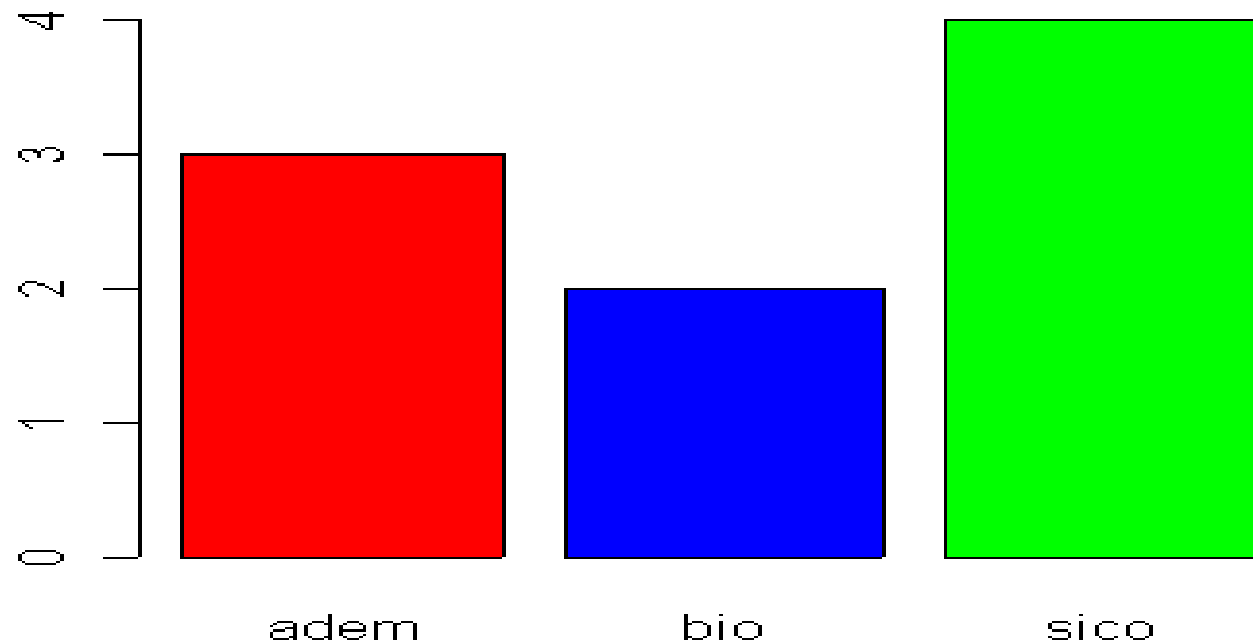


**boxplot de edad**





# Grafica de Barras



```
barplot(table(programas),col=c("red","blue","green"))
```

# Pie-charts



```
pie(table(programas),col=c("red","blue","green"),main="distribucion de  
estudiantes por programa",cex.main=.8)
```

# Grafica en dos dimensiones

```
htv=c(16,18,19,21,23,24,25,27,28,30)
```

```
gpa=c(3.17,3.45,2.95,2.71,2.64,2.65,2.37,2.68,2.11,2.09)
```

```
genero=c("M", "V", "V", "M", "M", "V", "M", "M", "M",  
"V")
```

```
plot(htv,gpa,main="htv versus gpa",col="red")
```

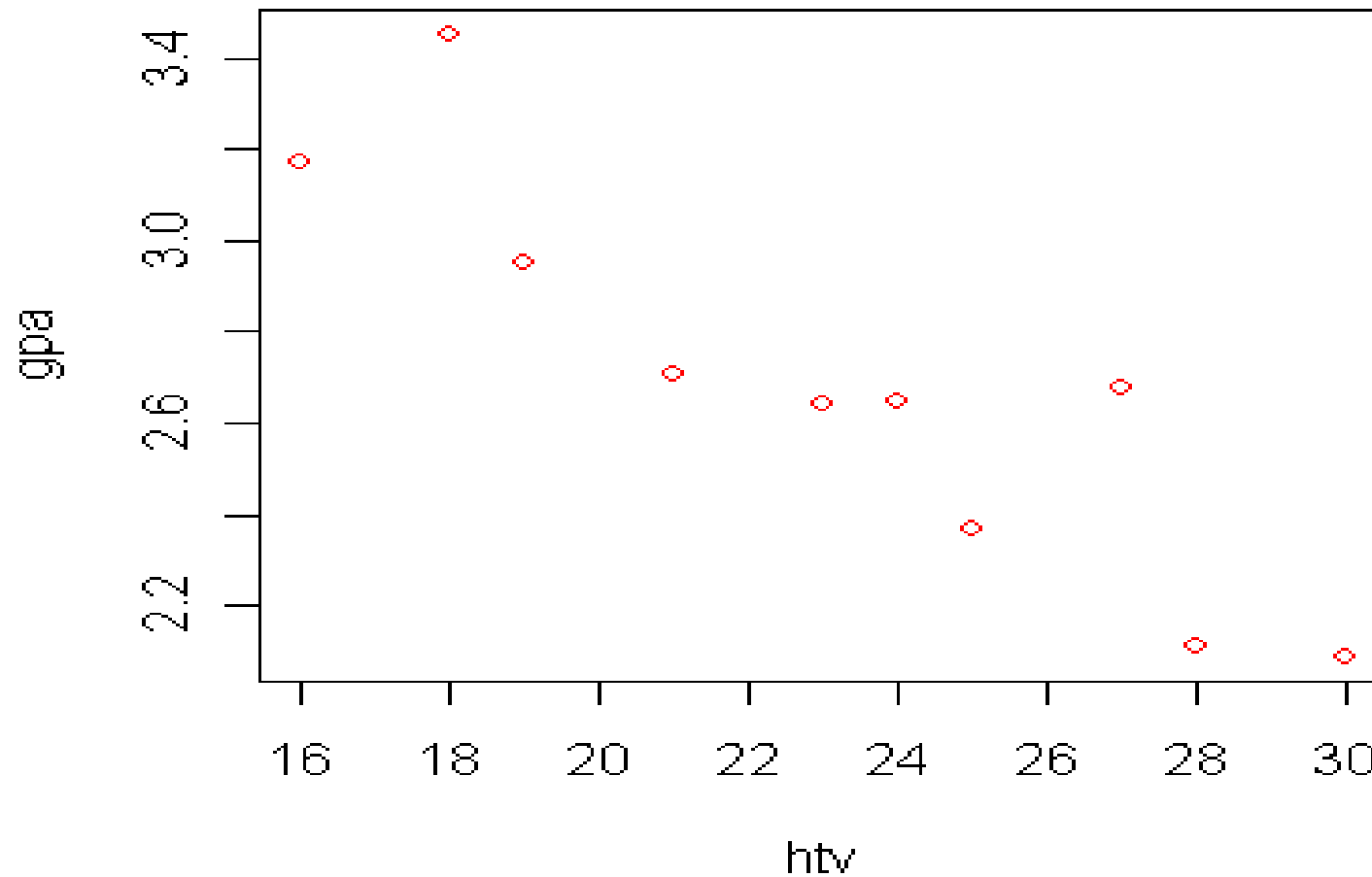
```
boxplot(gpa~as.factor(genero),col="green")
```

```
title("GPA por genero")
```

```
edad=c(20,17,21,23,22,24,21,18,19,22)
```

# Scatterplot

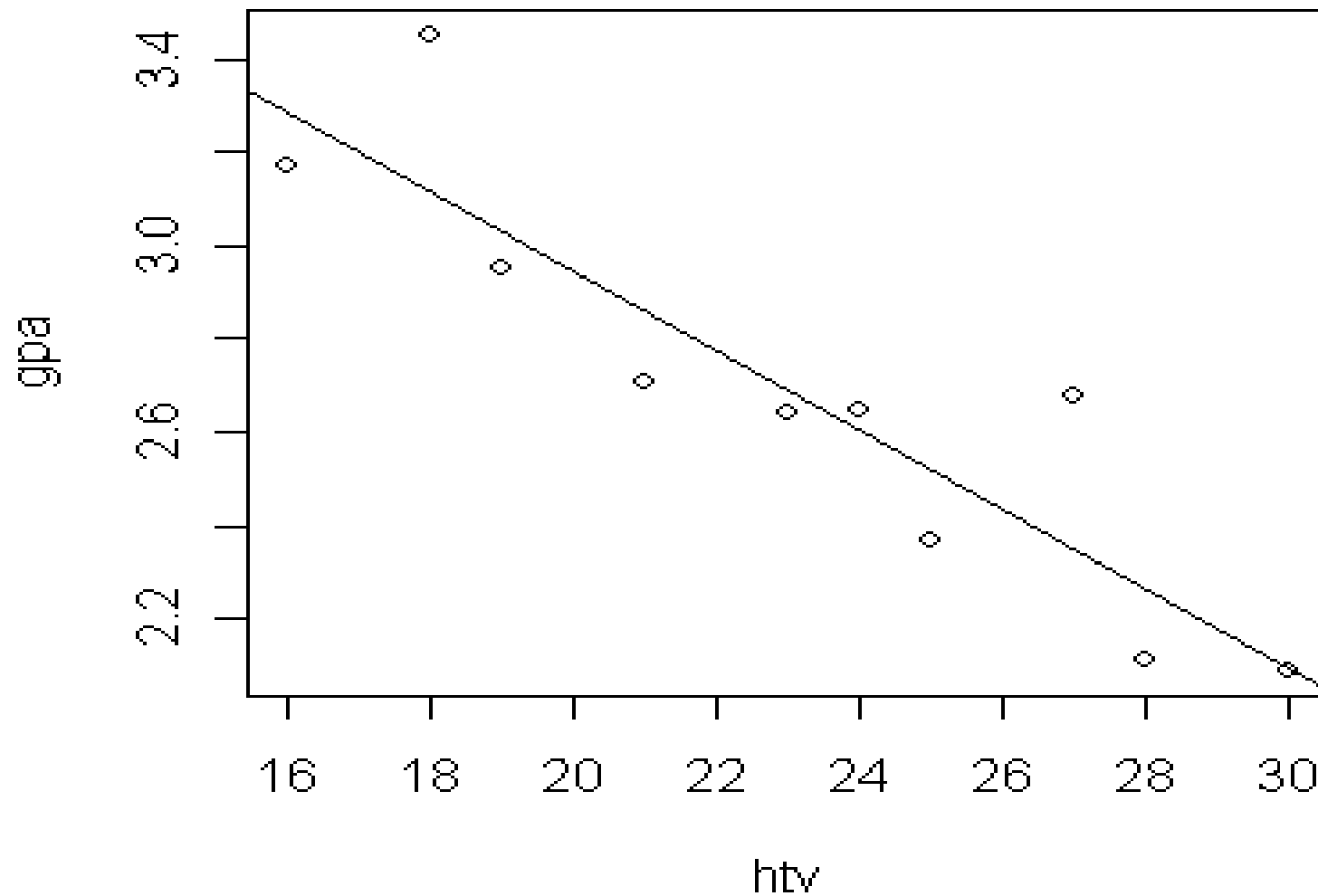
**htv versus gpa**



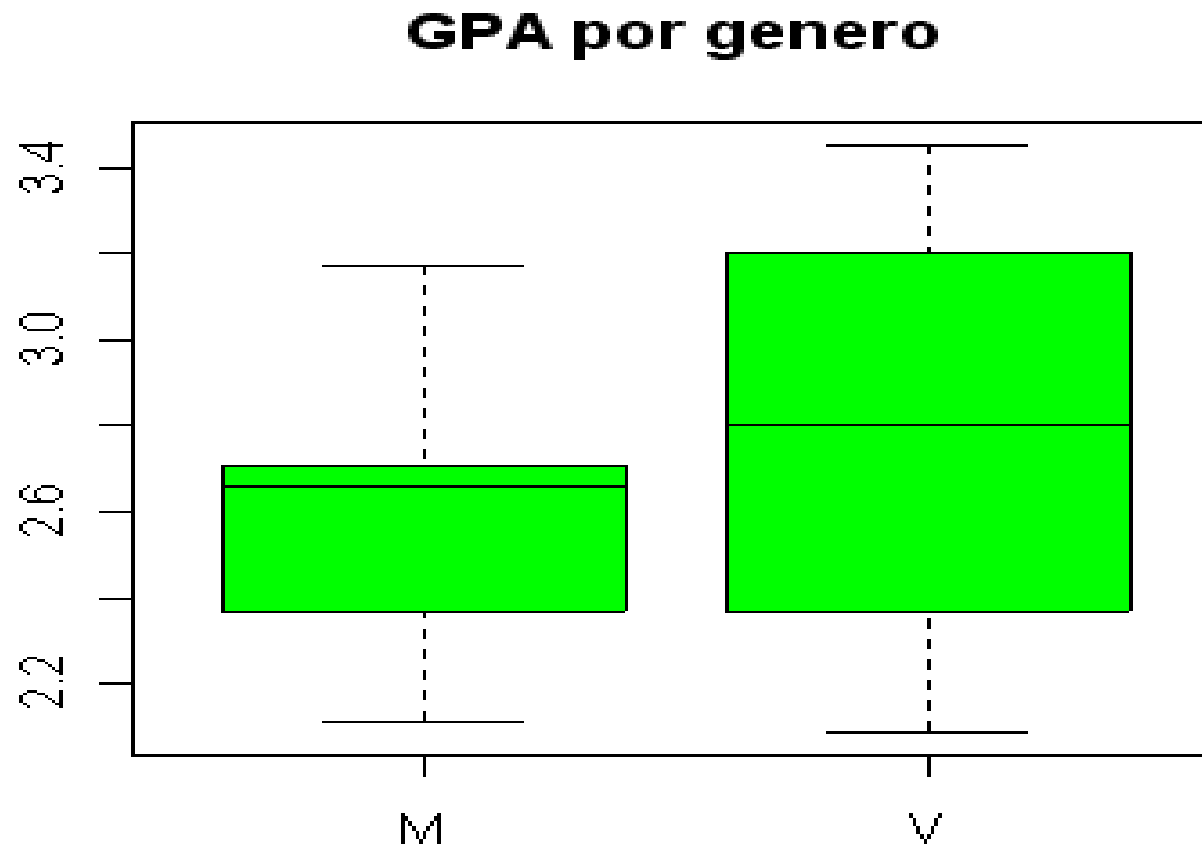
# Regresion y correlacion lineal simple

```
> cor(htv,gpa)
[1] -0.9015101
> rl=lm(gpa~htv)
> summary(rl)
Call:
lm(formula = gpa ~ htv)
Residuals:
    Min     1Q   Median     3Q    Max
-0.15562 -0.14167 -0.06545  0.03194  0.33462
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.64495   0.33899  13.702 7.76e-07 ***
htv          -0.08498   0.01442  -5.892 0.000365 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.1982 on 8 degrees of freedom
Multiple R-Squared: 0.8127,    Adjusted R-squared: 0.7893
F-statistic: 34.72 on 1 and 8 DF, p-value: 0.000365
```

**linea de regresion mostrando la relacion entre htv y gpa**



# Boxplots para comparar grupos



# Comparando dos grupos

```
t.test(gpa~as.factor(genero))
```

Welch Two Sample t-test

data: gpa by as.factor(genero)

$t = -0.5372$ ,  $df = 4.592$ ,  $p\text{-value} = 0.6161$

alternative hypothesis: true difference in means is not equal to  
0

95 percent confidence interval:

-1.0156057 0.6722724

sample estimates:

mean in group M mean in group V

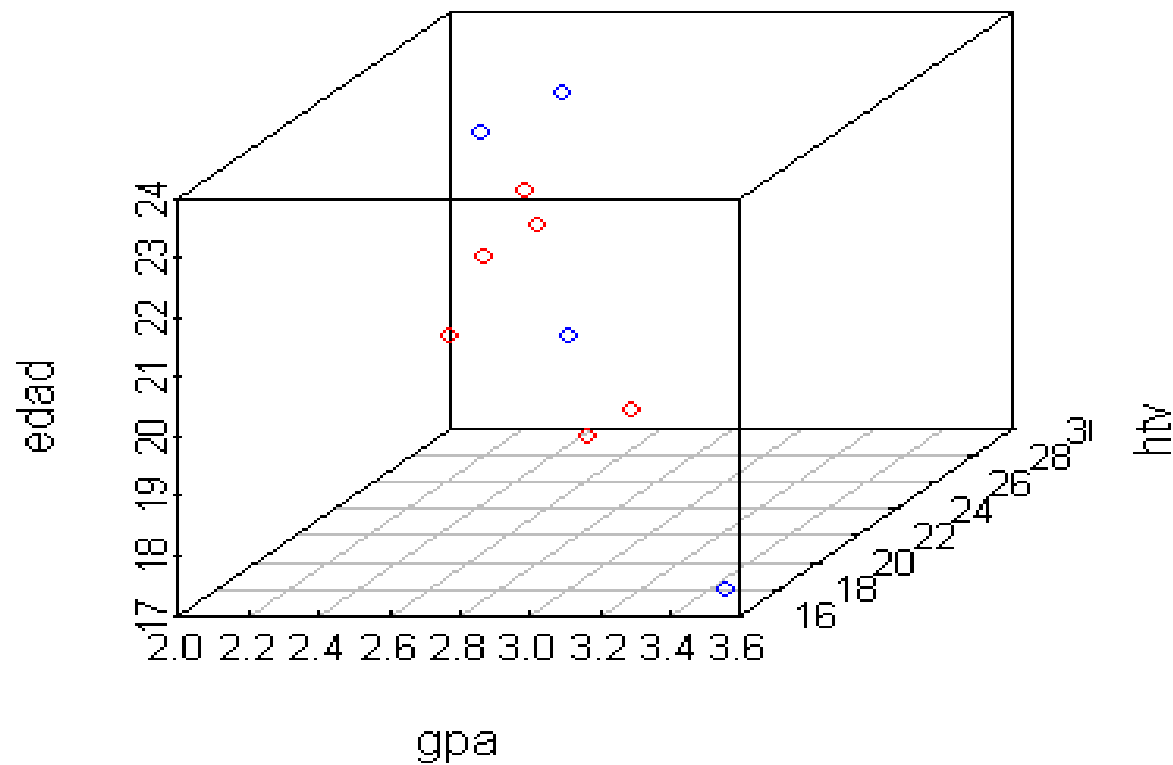
2.613333 2.785000



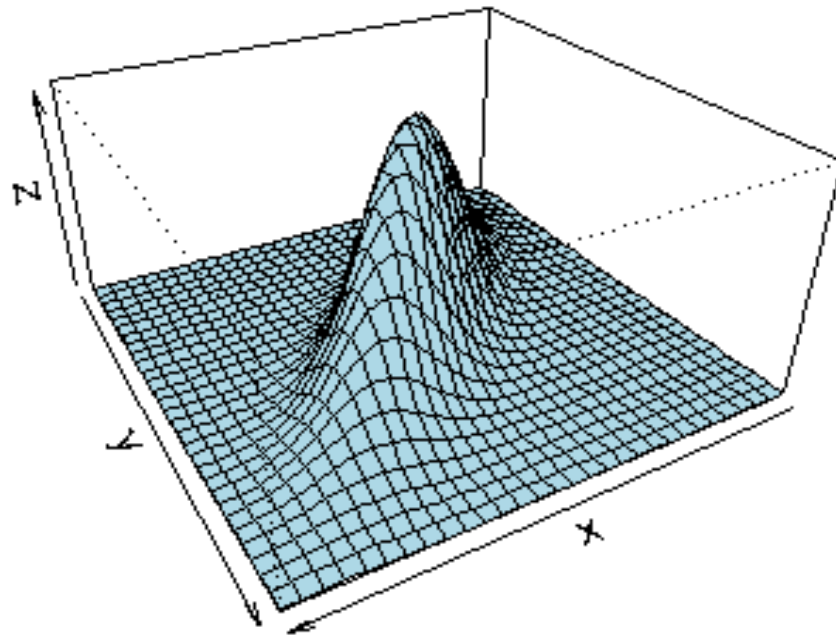
# Plot en 3 dimensiones

```
color=c("red","blue")[as.factor(genero)]
> color
[1] "red" "blue" "blue" "red" "red" "blue" "red" "red"
"red" "blue"
>scatterplot3d(gpa,htv,edad,color) #requiere instalar el paquete
scatterplot3d
x =seq(-3, 3, length= 30)
y=x
f=function(x,y) { (1/(2*pi*.6))*exp(-(1/.72)*(x^2-1.6*x*y+y^2)
)} # definiendo la función normal bivariada
z = outer(x, y, f) # calculando la funcion en cada par (x,y)
persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col =
"lightblue")
```

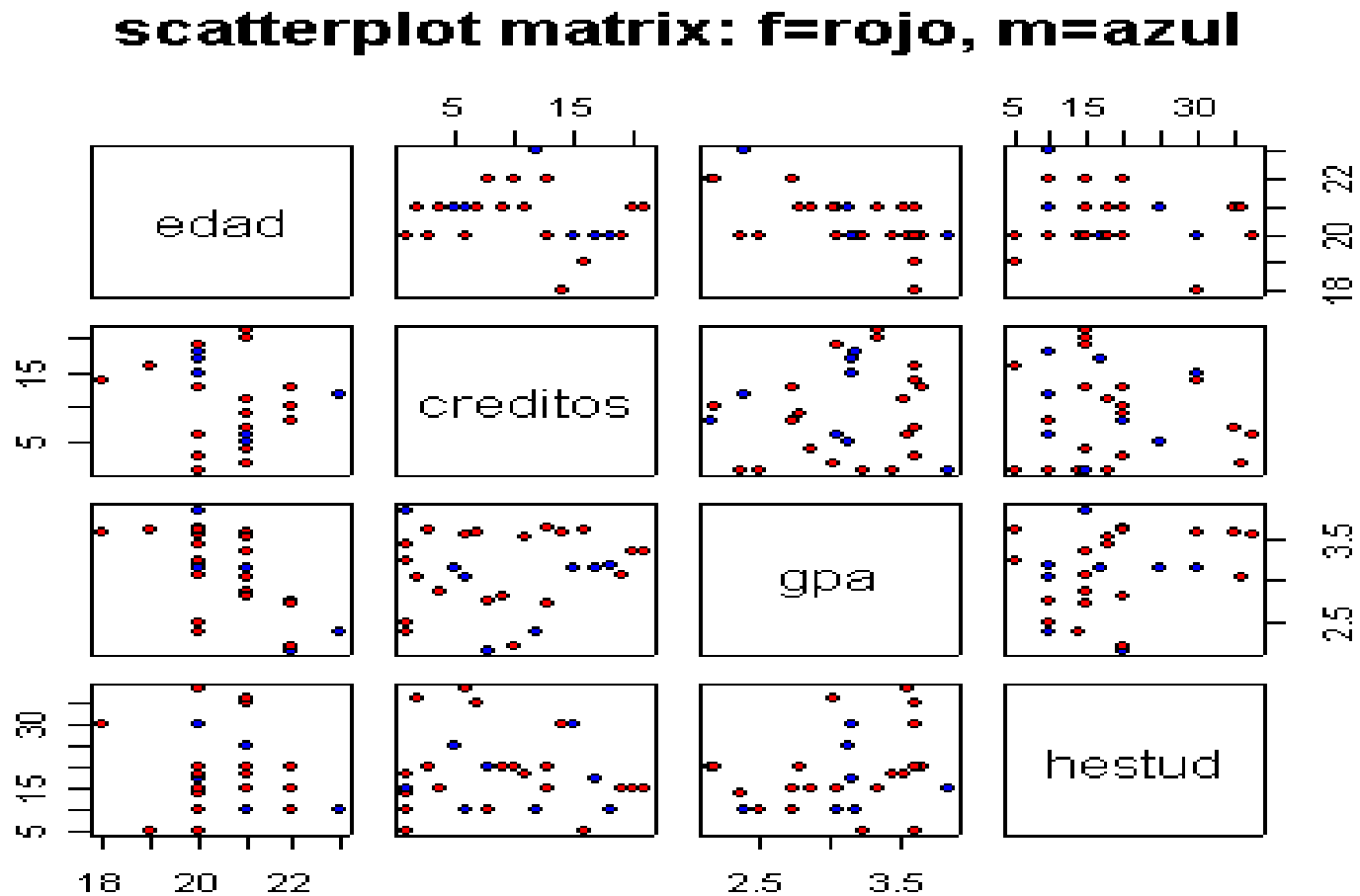
# Plot en 3 dimensiones



# Grafica de una densidad normal bivariada (0,0,1,1,.8)



# Grafica en mas de tres dimensiones



```
pairs(clase[,c(1,5,6,8)], pch=21, bg=c("red","blue")[unclass(clase$sexo)])
```

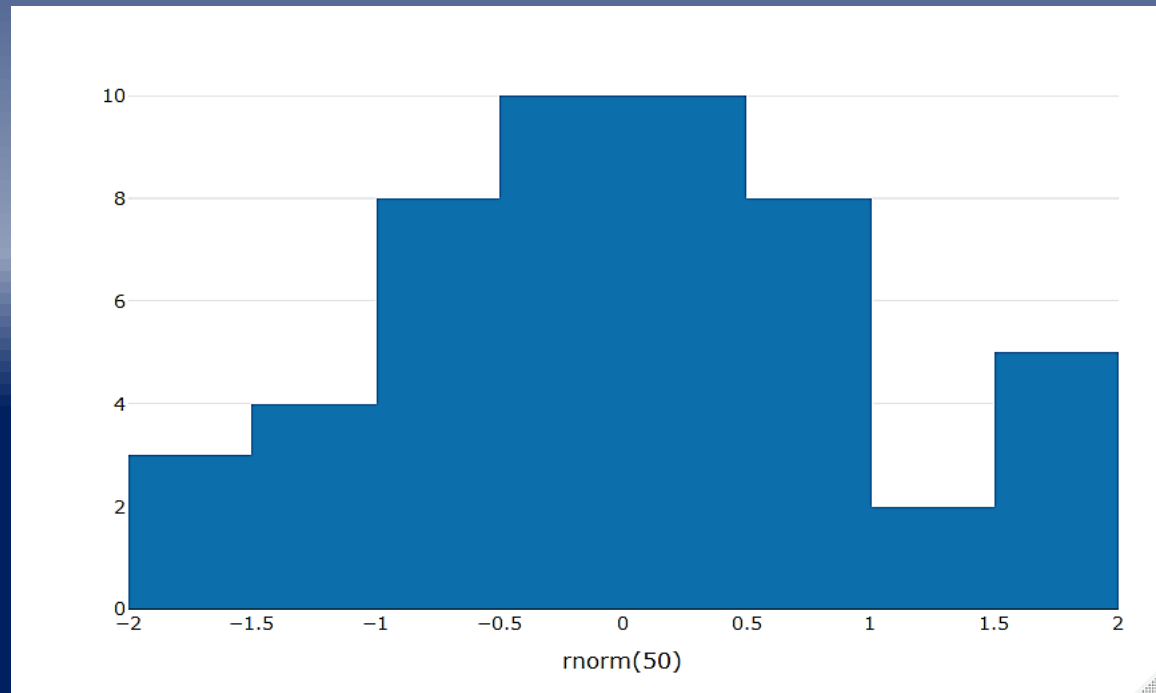
# Librerías para gráficas en R

Ggplot2 ( Hadley Wickham, 2009)

Plolty (2014)

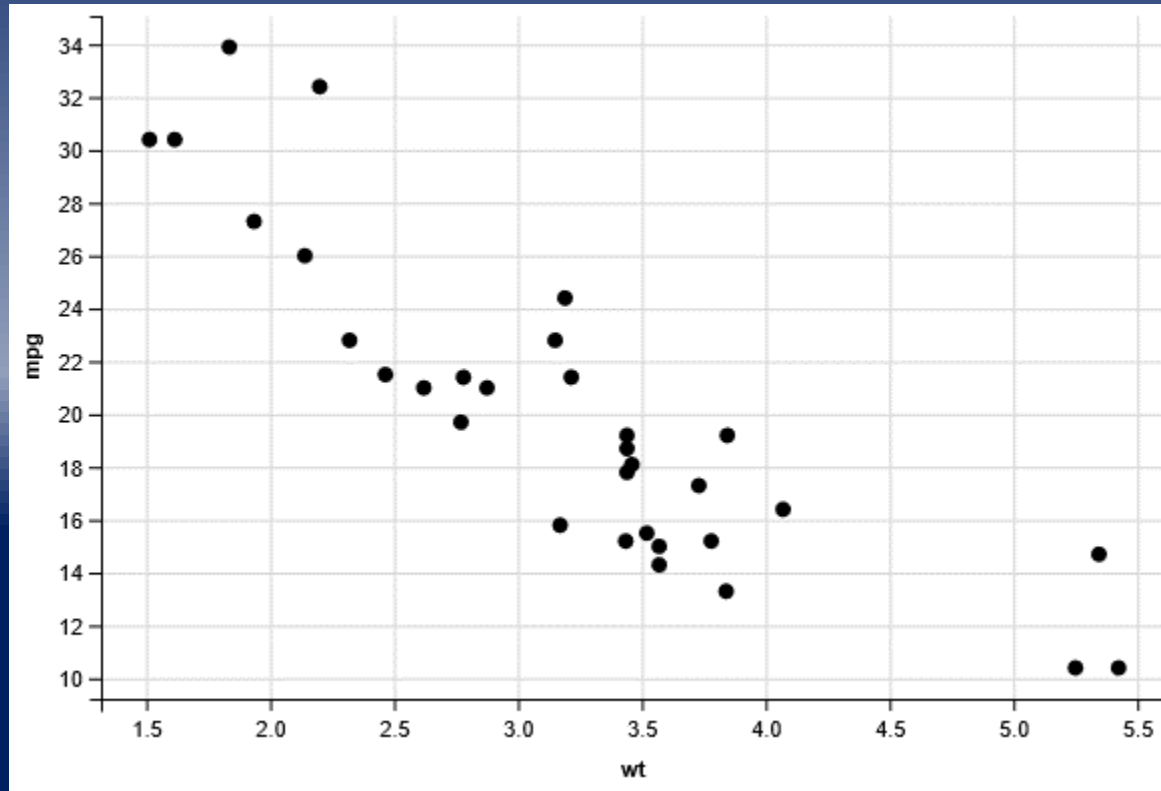
Ggvis (Hadley Wickham , 2014)

# Histograma en Plotly



```
library(plotly)
p <- plot_ly(x = ~rnorm(50), type = "histogram")
p
```

# Scatterplot en ggvis



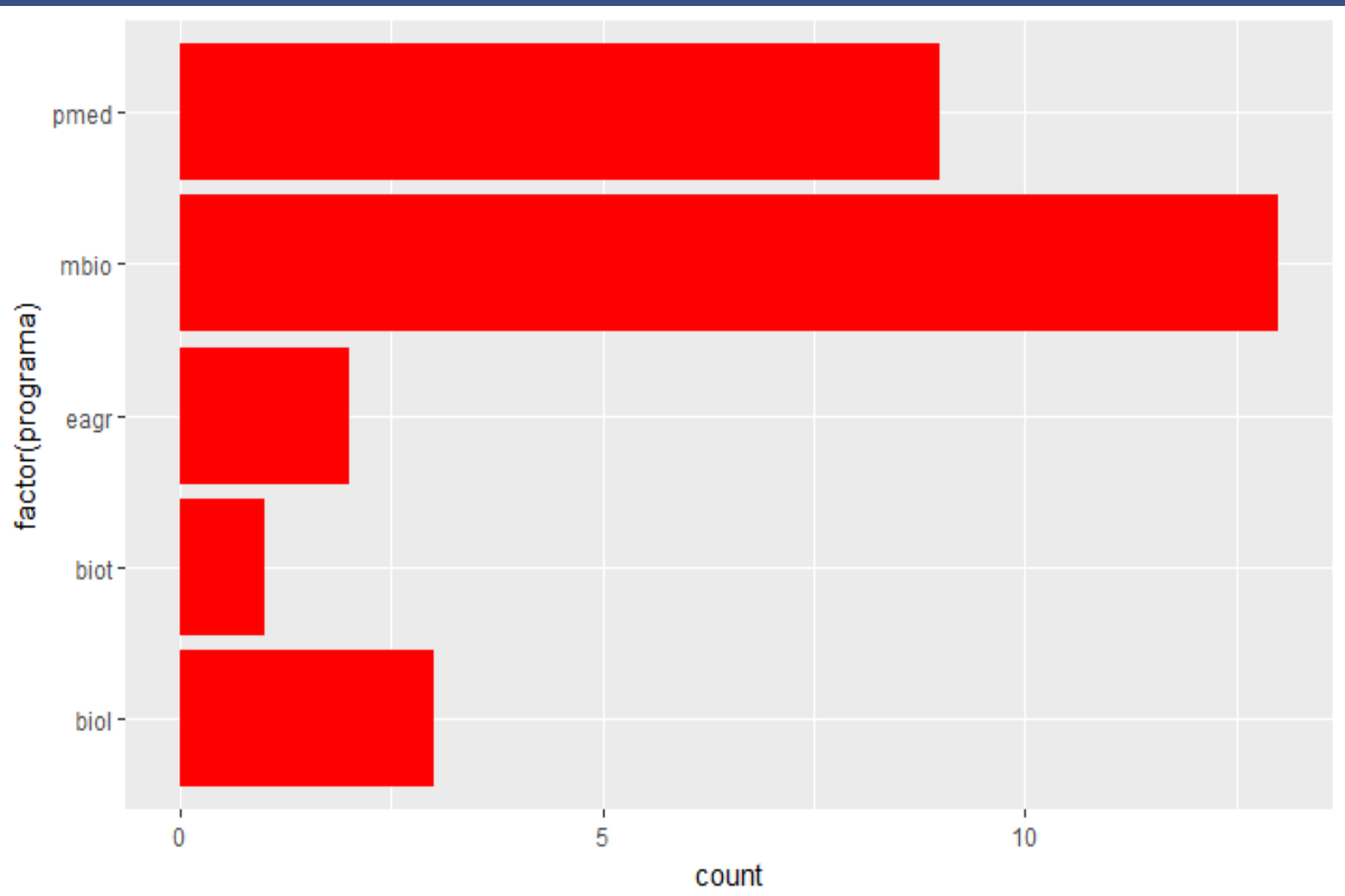
```
library(ggvis)  
data(mtcars)  
ggvis(mtcars, ~wt, ~mpg)
```

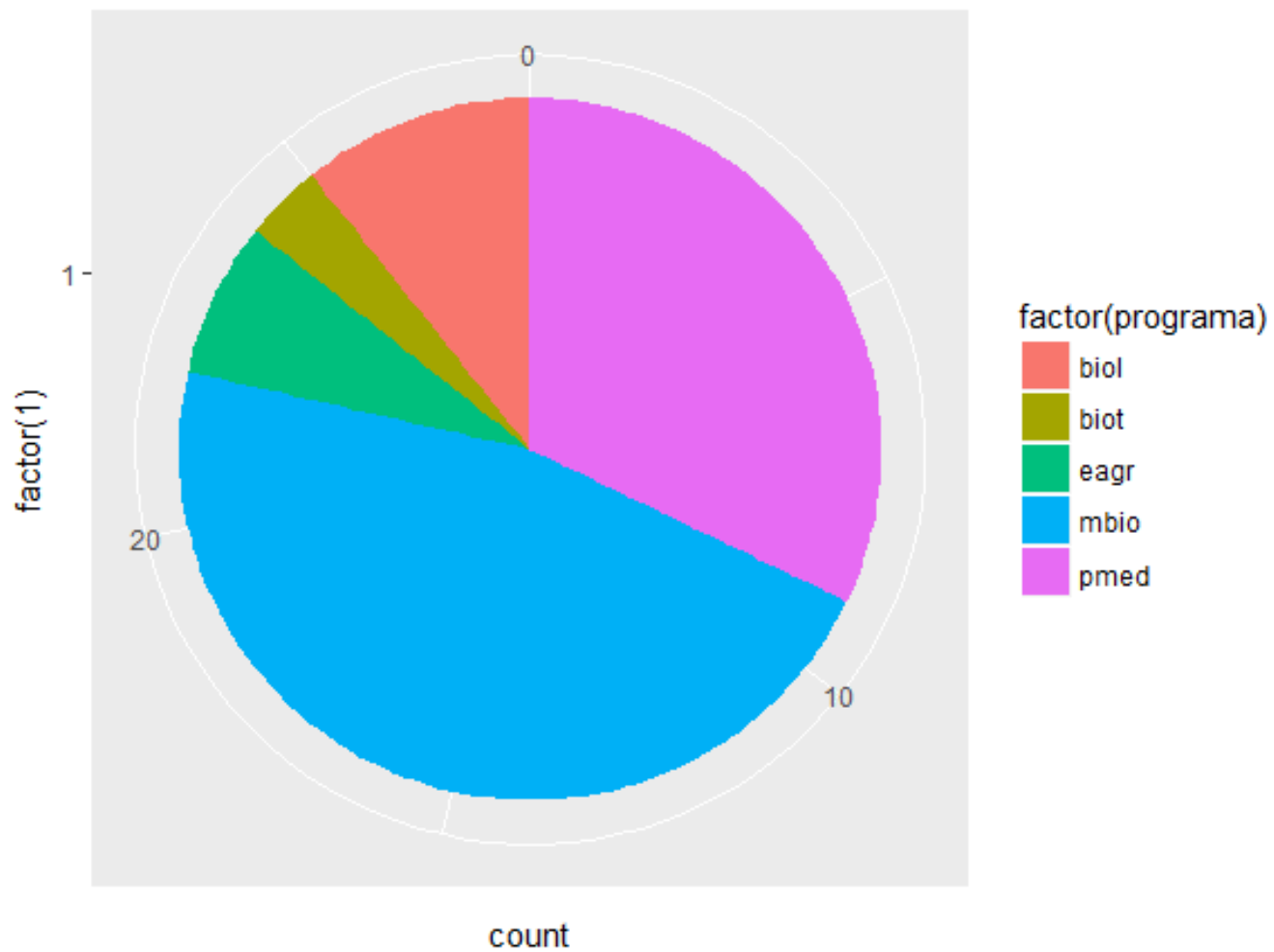
# ggplot2



# Graficas de barras y pie-chart

```
datos=read.table("http://academic.uprm.edu/eacuna/clase97.txt",  
na.strings="*",header=T)  
attach(datos)  
#Usando la libreria ggplot2  
library(ggplot2)  
#grafica de barras verticales  
ggplot(datos, aes(factor(programa)))+geom_bar(fill="green")  
#Grafica de barras horizontales  
ggplot(datos, aes(factor(programa)))+geom_bar(fill="red")+coord_flip()  
#Grafica de pie-chart  
barras=ggplot(datos,aes(x=factor(1),fill=factor(programa)))+geom_bar(  
width=1)  
barras+ coord_polar("y")
```





# Boxplots usando ggplot

#Boxplot elegante

```
(datos,aes(x=1,y=edad))+geom_boxplot()+theme(axis.text.x=element_blank(),
```

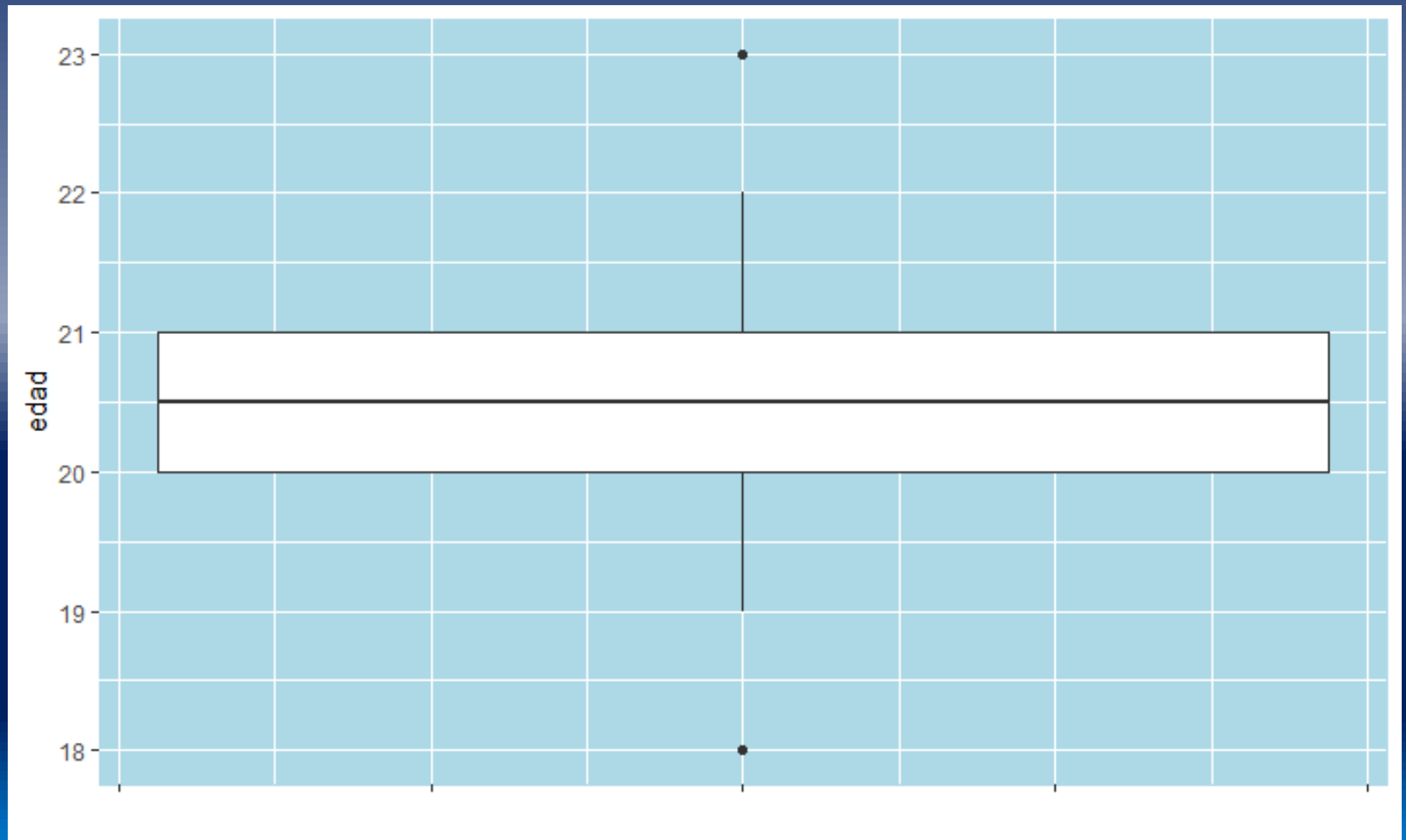
```
panel.background = element_rect(fill = "lightblue",colour = "lightblue",size = 0.5, linetype = "solid"),
```

```
panel.grid.major = element_line(size = 0.5, linetype = 'solid', colour = "white"),
```

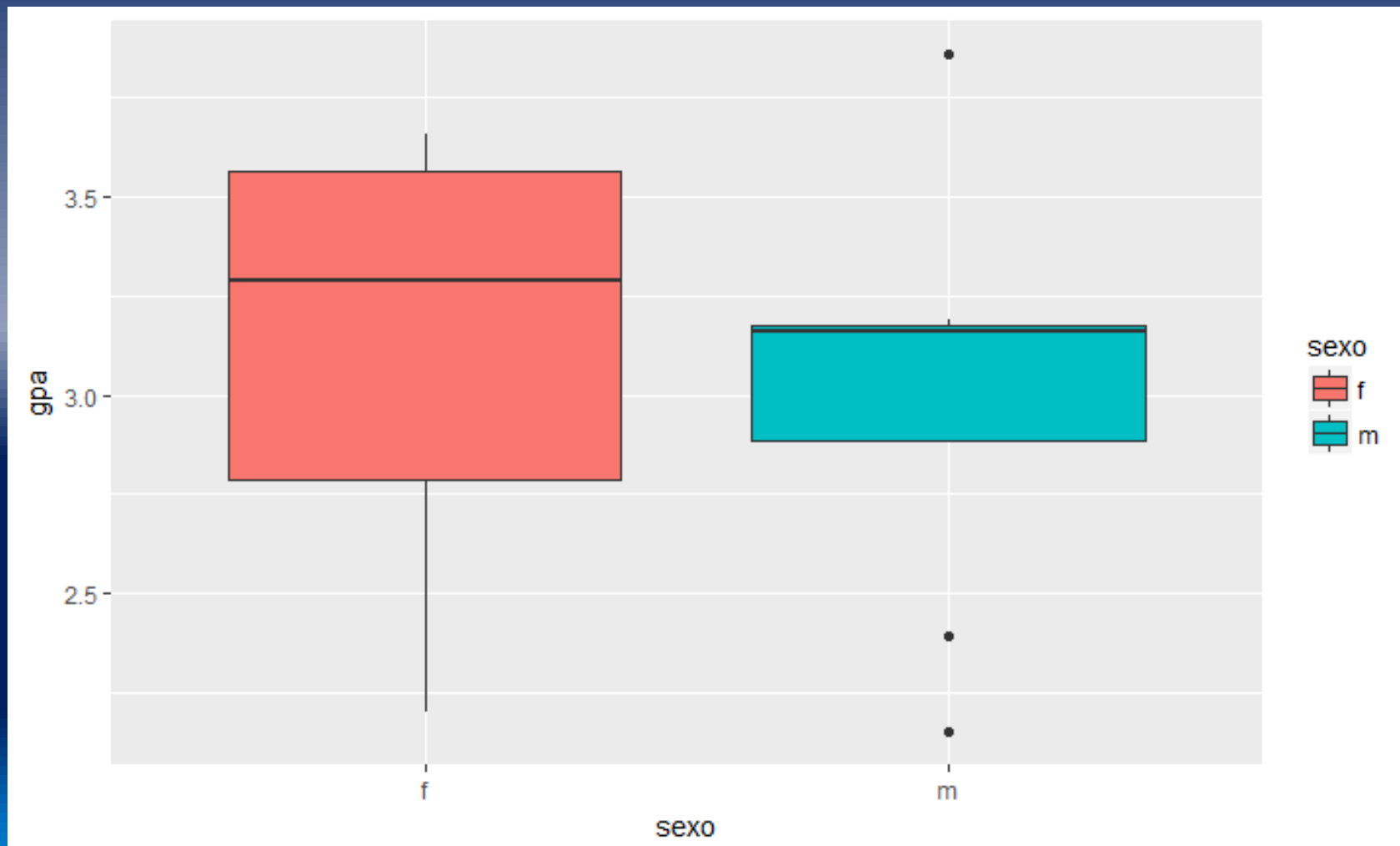
```
panel.grid.minor = element_line(size = 0.25, linetype = 'solid',colour = "white"))+labs(x=" ")
```

#Boxplot para comparar dos grupos

```
ggplot(datos,aes(x=sexo,y=gpa,fill=sexo))+geom_boxplot()
```



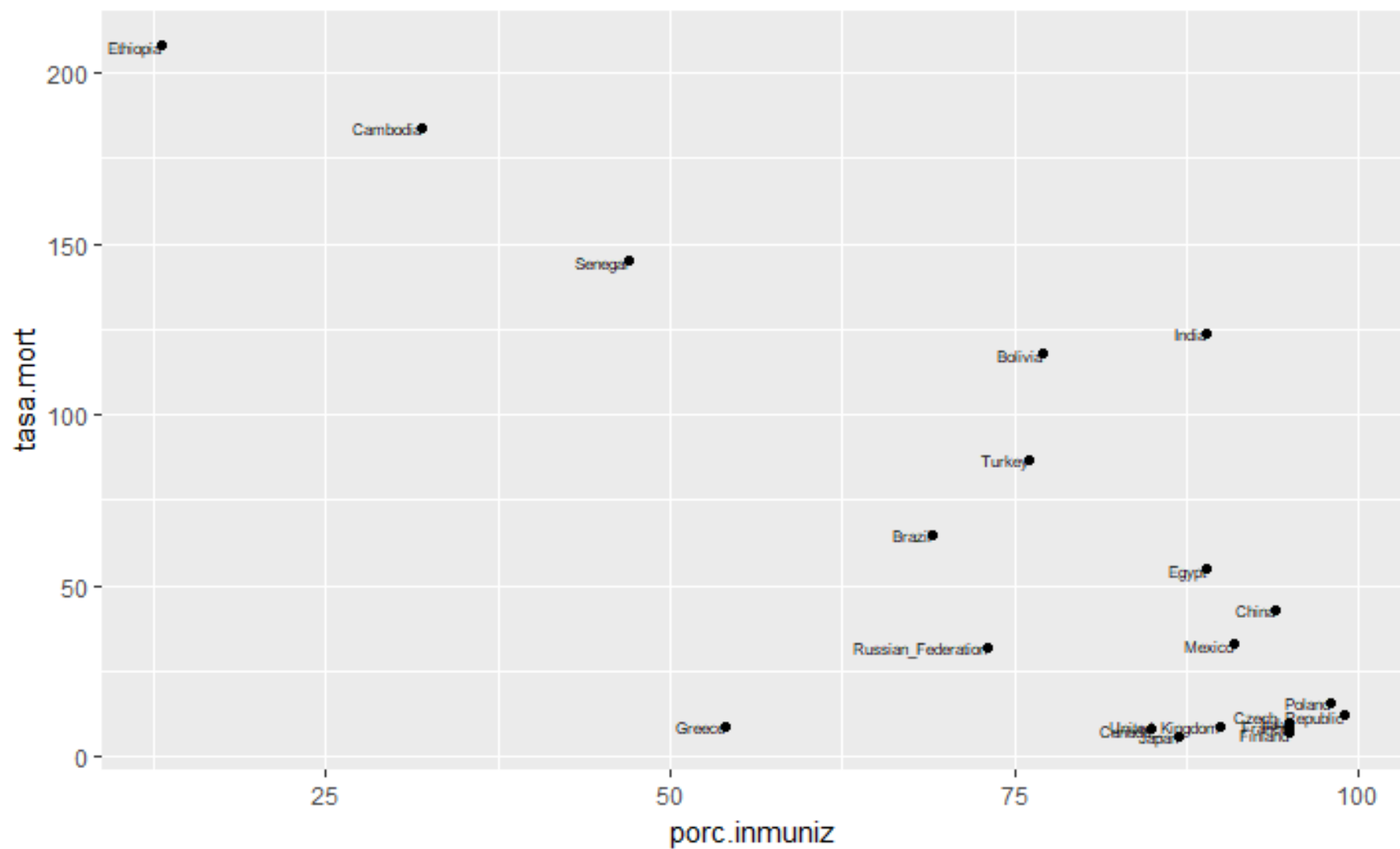
UPRM, Enero 2020  
Edgar Acuna



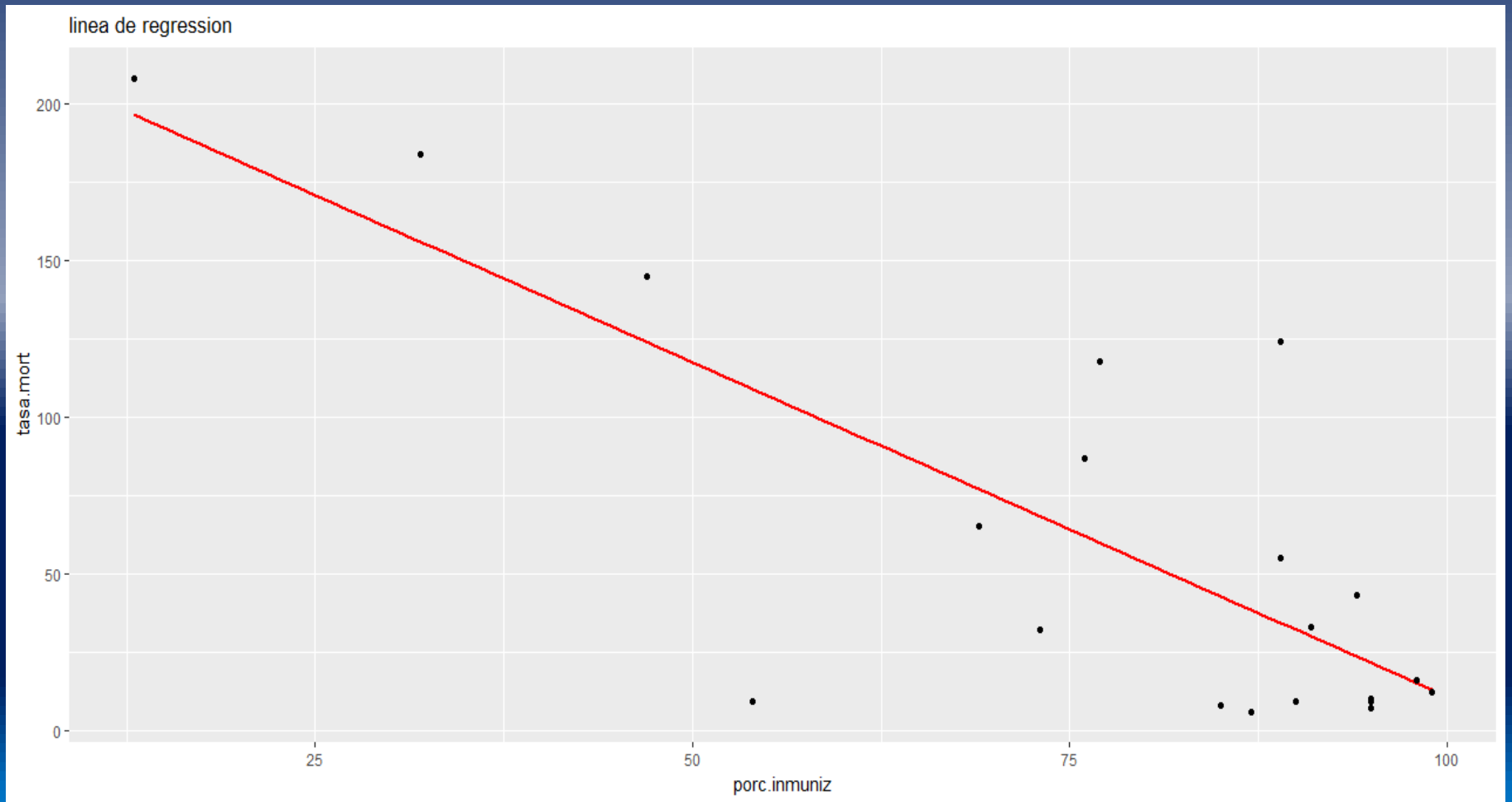
# Scatterplot y regression lineal

```
datos1=read.table("http://academic.uprm.edu/eacuna/mortalidad.txt",header=T)
attach(datos1)
#Usando la libreria ggplot2
ggplot(datos1,aes(x=porc.inmuniz,y=tasa.mort,label=nacion))+
  geom_point()+geom_text(hjust=1,size=2)

#scatter plot incluyendo la linea de regresion
p=ggplot(datos1,aes(x=porc.inmuniz,y=tasa.mort))+geom_smooth(method="lm",se=FALSE,color="red",formula=y~x)+geom_point()
p +ggtitle("linea de regression")
```







# ggplotGUI

Introducida en Julio el 2017.

Facilita el analisis grafico de bases de datos Requiere que tenga instalado la libreria shiny

Despues de dar `library(shiny)` y `library(ggplotgui)`

Se activa usando el commando

`ggplot_shiny("filename")`. Se puede usar desde R o Rstudio.

Usar la opcion Data Upload para cargar su propios datos

# ggplot GUI

Create visualization

Type of graph:  

Boxplot

Y-variable  

x.1

X-variable  

No x-var

Group (or colour)  

classes

Facet Row  

No groups

Facet Column  

No groups

☐ Show data points (jittered)

☐ Notched box plot

For more info see the 'Info'-tab or visit <https://github.com/gertstulp/ggplotgui>

Data upload

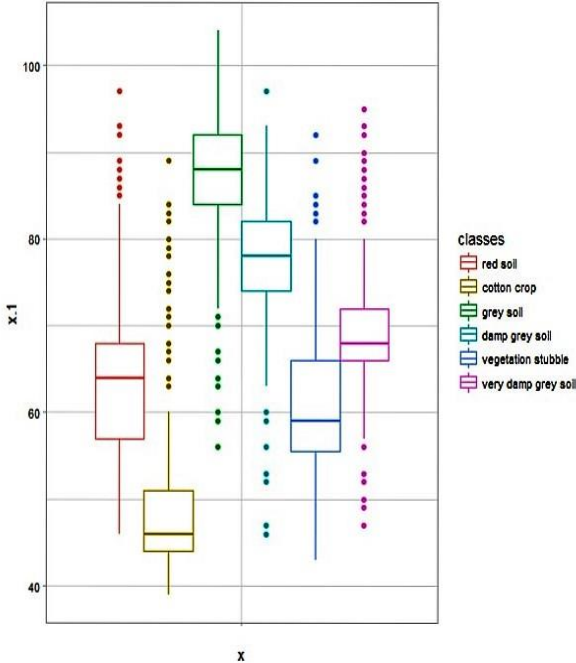
ggplot

Plotly

R-code

Info

Download pdf of figure



Change aesthetics

Text

Theme

Legend

Size

☐ Change colours

☐ Remove gridlines

Theme  

light

UPRM, Enero 2020

Edgar Acuna

# Graficas interactivas usando Shiny

Shiny es un paquete de R que permite hacer graficas interactivas y aplicaciones web.

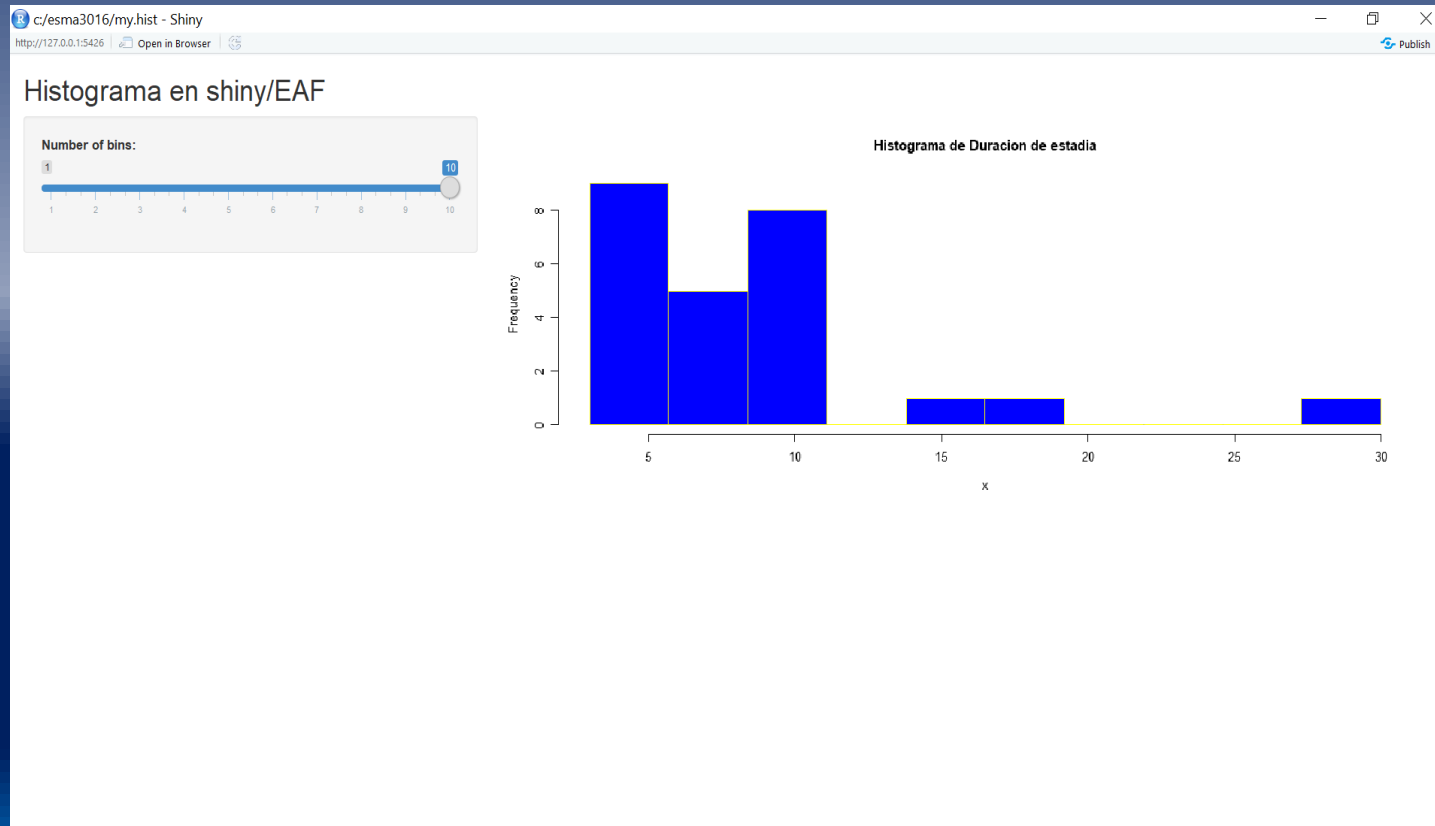
Crear un archivo server.R y otro ui.R dentro de un directorio my.app

Ver varios ejemplos en la pagina de shiny

(<https://shiny.rstudio.com/gallery/>)

```
shiny::runApp("c://my.app")
```

# Graficas interactivas usando Shiny



Shiny::runApp("c://HP-PR/my.hist  
UPRM, Enero 2020

Edgar Acuna

# Aplicaciones Web Shiny

Aqui les muestro una aplicacion de mi Proyecto que va acompañada de una presentacion

Para las graficas de shiny uso tres archivos global.R, server.R y ui.R

Copiarlos en un directorio “dosgrafos” y luego de llamar a la libreria shiny darle el comando

`Shiny::runApp(“dosgrafos”)`

# Aplicaciones Web Shiny

## Bee activity

Choose Bee data File

Browse...

data\_24hrs.csv

Upload complete

Type of graph (Actogram only for data with  
24 hours by day)

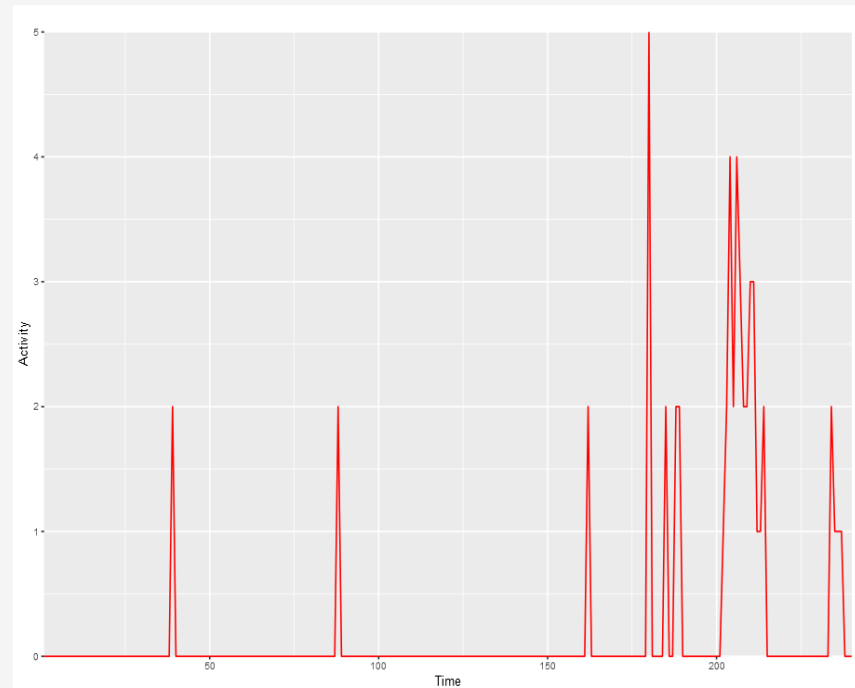
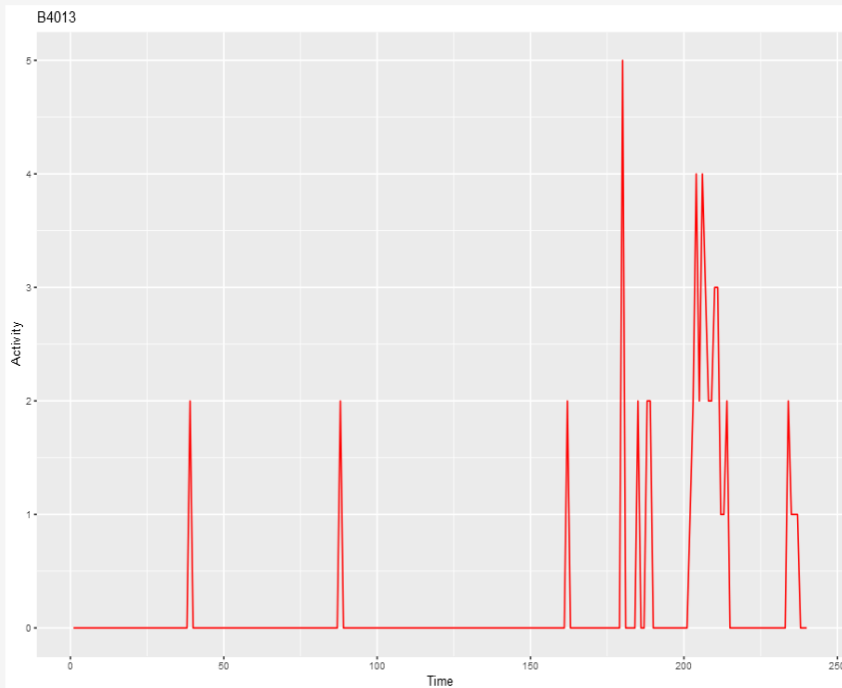
☒ Series

☐ Actogram

Bee ID:

B4013

Left plot controls right plot



# Aplicaciones Web Shiny

## Bee activity

Choose Bee data File

Browse...

data\_24hrs.csv

Upload complete

Type of graph (Actogram only for data with  
24 hours by day)

☐ Series

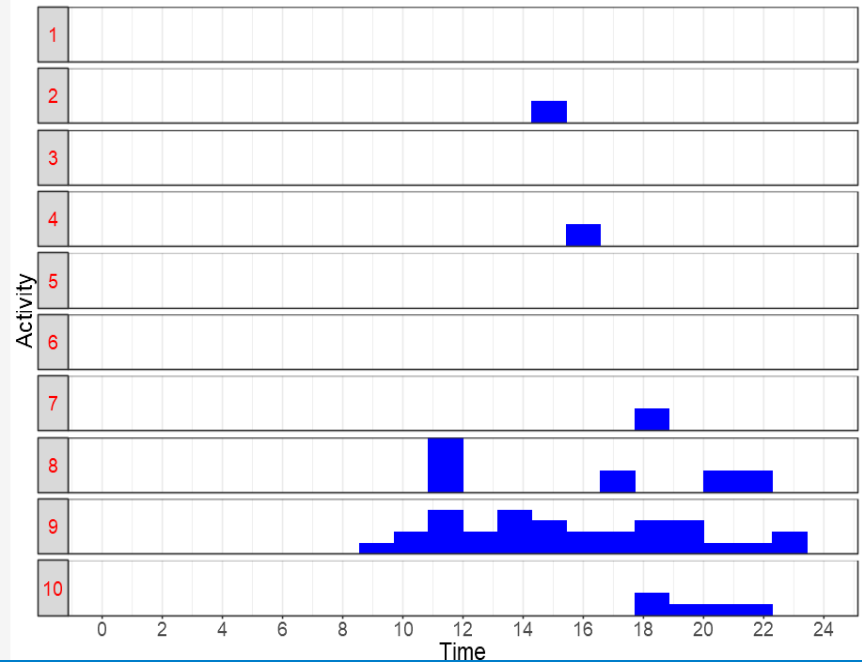
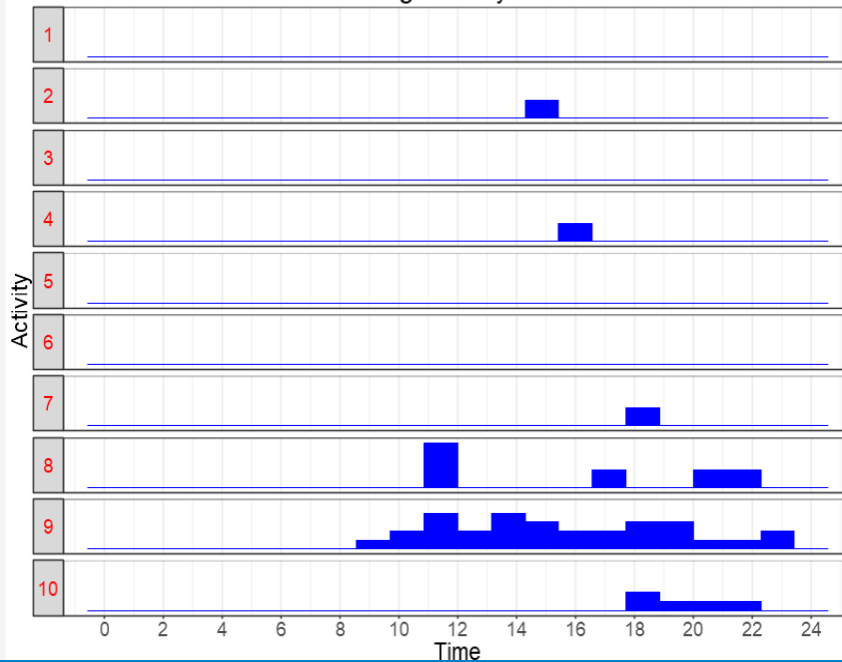
☒ Actogram

Bee ID:

B4013

Left plot controls right plot

Actograma by hour





# Usando lubridate-1

Permite manipular facilmente datos de tiempo (fechas y hora)

#Convirtiendo year- month-day en un objeto tipo Date.

```
ymd(20101215)
```

```
"2010-12-15 "
```

#Convirtiendo month/day/year en un objeto tipo Date.

```
mdy("4/1/17")
```

```
[1] "2017-04-01"
```

```
begin <- c("May 11, 1996", "September 12, 2001", "July 1, 1988")
```

```
end <- c("7/8/97","10/23/02","1/4/91")
```

#Convirtiendo un vector de fechas un objeto tipo Date.

```
begin <- mdy(begin)
```

```
begin
```

```
[1] "1996-05-11" "2001-09-12" "1988-07-01"
```

```
end <- mdy(end)
```

# Usando lubridate-2

```
time.interval <- begin %--% end
> time.interval
[1] 1996-05-11 UTC--1997-07-08 UTC 2001-09-12 UTC--2002-10-23
UTC
[3] 1988-07-01 UTC--1991-01-04 UTC
> time.duration <- as.duration(time.interval)
> time.duration
[1] "36547200s (~1.16 years)" "35078400s (~1.11 years)"
[3] "79228800s (~2.51 years)"
#duracion en semanas
time.duration/dweeks(1)
[1] 60.429 58.000 131.000
```

# Usando lubridate-3

```
#Tiempos en horas minutos y segundos
time1 <- c("1:13", "0:58", "1:01")
time2 <- c("12:23:11", "09:45:31", "12:05:22")
time1 <- ms(time1)
time2 <- hms(time2)
> as.numeric(time1)
[1] 73 58 61
> as.numeric(time2)
[1] 44591 35131 43522
#Fecha completa incluyendo time zone
> start <- mdy_hm("3-11-2017 5:21", tz = "US/Eastern")
> start
[1] "2017-03-11 05:21:00 EST"
```

# Usando dplyr-1

La gente ha estado usando por varias decadas SQL para analizar datos. Todos los programas actuales para analizar datos tales como R, SAS y Python hacen conexion con bases de datos en SQL, pero SQL no fue creado para analizar datos fue creado para manipular y hacer consultas acerca de los mismos.

Hay muchas operaciones de analisis de datos donde SQL falla o se complica en hacerlas.

En cambio el paquete dplyr de R si fue disenado para analisis de datos analizar datos ademas de hacer las mismas operaciones de SQL.

# Usando dplyr-2

<b>Funcion</b>	<b>Descripcion</b>	<b>SQL</b>
<b>select()</b>	<b>Selecciona columns</b>	<b>SELECT</b>
<b>filter()</b>	<b>Selecciona filas</b>	<b>WHERE</b>
<b>arrange()</b>	<b>Reordena filas</b>	<b>ORDER BY</b>
<b>mutate()</b>	<b>Crea nuevas columnas</b>	<b>COLUMN ALIAS</b>
<b>summarise()</b>	<b>Resume datos</b>	
<b>group_by()</b>	<b>Hace operaciones por grupos</b>	<b>GROUP BY</b>

# Usando dplyr-3

```
# Eligiendo 4 filas al azar de flight delays 2008
```

```
> sample_n(X2008_csv,4)
```

```
# A tibble: 4 x 29
```

```
  Year Month DayOfMonth DayOfWeek DepTime CRSDepTime ArrTime  
  <int> <int>    <int>    <int>    <int>    <int>    <int>  
1  2008   10      2      4    1016      1023    1215  
2  2008    3      9      7    1745      1745    1825  
3  2008    1     11      5    1325      1329    1657  
4  2008   12     27      6     932       921    1048  
# ... with 22 more variables: CRSArrTime <int>, UniqueCarrier <chr>,  
# FlightNum <int>, TailNum <chr>, ActualElapsedTime <int>,  
# CRSElapsedTime <int>, AirTime <int>, ArrDelay <int>,  
# DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>,  
# TaxiIn <int>, TaxiOut <int>, Cancelled <int>,  
# CancellationCode <chr>, Diverted <int>, CarrierDelay <int>,  
# WeatherDelay <int>, NASDelay <int>, SecurityDelay <int>,  
# LateAircraftDelay <int>
```

# Usando dplyr-4

```
#Selecciona, algunas columnas de X2008_csv
select(X2008_csv, UniqueCarrier, Dest:Cancelled)
#Eliminando algunas columnas de X2008_csv
select(X2008_csv, -OriginUniqueCarrier, -AirTime)
#Seleccionando las columnas que empiezan con D
select(X2008_csv, starts_with("D"))
#Seleccionando las columnas que contienen T
select(X2008_csv, contains("T"))
#Seleccionando las filas que satisfacen la condicion que el vuelo sale de SJU
filter(X2008_csv, Origin=="SJU")
# Seleccionando las filas que satisfacen la condicion que el vuelo sale de SJU o
BQN
filter(X2008_csv, Origin%in% c("SJU", "BQN"))
```

# Usando dplyr-5

#Renombrando una columna

```
rename(X2008_csv, Distancia=Distance)
```

#Extrayendo los vuelos que salen de SJU y BQN que tienen delay

```
filter(X2008_csv, Origin %in% c("SJU", "BQN") & DepDelay > 15)
```

#Extrayendo los vuelos que salen de SJU y BQN que tienen delay o que son cancelados

```
filter(X2008_csv, Origin %in% c("SJU", "BQN") & (DepDelay > 15 |  
Cancelled==1))
```

#Filtrando los vuelos con código de origen que contiene el string "SJ"

```
filter(X2008_csv, grepl("SJ", Origin))
```

#Hallando la media y desviación estándar de la variable DepDelay

```
summarise(X2008_csv, delay_mean = mean(DepDelay, na.rm=T),  
delay_sd = sd(DepDelay, na.rm=T))
```



# Usando dplyr-6

```
#Resumir mas de una variable. Calculando la media y mediana de las
columnas AirTime y ArrDelay
summarise_at(X2008_csv, vars(AirTime, ArrDelay),
funs(mean(.,na.rm=TRUE), median(.,na.rm=TRUE)))
#Hallando el numero de missings en la variable CarrierDelay
summarise_all(X2008_csv["CarrierDelay"], funs(nmiss=sum(is.na(.))))
#Ordenando los datos por las variables AirTime and ArrivalDelay
arrange(X2008_csv,AirTime,ArrDelay)
```

# Usando dplyr-7

El Operador Pipe %>%

Permite escribir sub-consultas como en SQL.

Lo usan varios paquetes de R pero fue introducido inicialmente en el paquete magrittr

Los siguientes dos comandos seleccionan al azar 10 valores de las columnas ArrDelay y DepDelay de X2008\_csv

```
dt = sample_n(select(X2008_csv, ArrDelay, DepDelay),10)
```

O tambien

```
dt = X2008_csv %>% select(ArrDelay, DepDelay) %>% sample_n(10)
```

# Usando dplyr-8

#Calculando el numero de vuelos y la media de ArrDelay y DepDelay por Carrier

```
summarise_at(group_by(X2008_csv, UniqueCarrier), vars(ArrDelay, DepDelay), funs(n(), mean(., na.rm=TRUE)))
```

O usando el operador pipe

```
X2008_csv %>% group_by(UniqueCarrier) %>%  
  summarise_at(vars(ArrDelay, DepDelay), funs(n(), mean(., na.rm = TRUE)))
```

Anadiendo una nueva columna demora=DepDelay+ArrDelay

```
mutate(X2008_csv, demora= DepDelay+ArrDelay)
```

# Usando dplyr-9

```
df1 = data.frame(ID = c(1, 2, 3, 4, 5),  
                  w = c('a', 'b', 'c', 'd', 'e'),  
                  x = c(1, 1, 0, 0, 1),  
                  y=c(1.1,1.5,2.5,3.5,5.5),  
                  z=letters[1:5])
```

```
df2 = data.frame(ID = c(1, 7, 3, 6, 8),  
                  a = c('z', 'b', 'k', 'd', 'l'),  
                  b = c(1, 2, 3, 0, 4),  
                  c =c(3.5,4.5,2.9,3.9,4.9),  
                  d =letters[2:6])  
inner_join(df1,df2,by="ID")
```

# Usando dplyr-10

```
left_join(df1,df2,by="ID")
```

```
ID w x y z a b c d
1 1 a 1 1.1 a z 1 3.5 b
2 2 b 1 1.5 b <NA> NA NA <NA>
3 3 c 0 2.5 c k 3 2.9 d
4 4 d 0 3.5 d <NA> NA NA <NA>
5 5 e 1 5.5 e <NA> NA NA <NA>
```

```
intersect(df1,df2)
```

```
union(df1,df2)
```

```
setdiff(df1,df2)
```

# Usando tidyverse

Es una coleccion de paquetes en R que son usados en Data Science. Incluye los siguientes paquetes:

- Dplyr : Para manipulacion de datos
- Ggplot2: Para graficas de datos
- Readr: Lectura de datos que vienen en tablas
- Tibble: version mejorada de data frame
- Stringr: Libreria para trabajar con strings
- Tidyr: Libreria para organizar los datos
- Purr: Libreria que mejora la programacion funcional de R
- Forcats: Library para trabajar con factores

# Conclusion

R es flexible y poderoso

- Facil de leer datos.
- Bastante capacidad de manipular datos.
- Enorme capacidad para hacer graficas.
- Un rango bien amplio de funciones estadisticas.
- Un gran numero de paquetes disponibles.
- Se puede guardar todo el trabajo que se hace en una sesion.