Introducción al Lenguage Estadístico R

Gabriel Nuñez Antonio

ITAM

XXIII Foro Nacional de Estadística.
Boca del Río, Veracruz, 2008.

Contenido

- Introducción.
- Manipulación de Datos.
- Gráficos.
- Análisis Estadístico.

Introducción

- **Instalación:**
  1. Google: R. (`http:www.r-project.org`) || CRAN
  2. *Elegir alguna dirección URL* || `http://lib.stat.cmu.edu/R/CRAN/`
  3. *Elegir algún sistema operativo* || Widows || base
  4. Bajar el ejecutable
- R ofrece una gran cantidad de funciones para realizar análisis gráfico y estadístico.
- ¿Cómo trabaja R?
- R es un lenguaje orientado a objetos. Es un intérprete no un compilador.
- Una vez que se abre R aparece el prompt de default "$>$", lo que indica que R espera algún comando.

Introducción

- El nombre de un objeto debe empezar con una letra (A-Z y a-z) y puede incluir digitos y puntos.
- R discrimina para el nombre de los objetos letras mayúsculas de minúsculas, por lo que **x** y **X** nombrarán a diferentes objetos.
- En R para ejecutar una función, ésta siempre se debe escribir con paréntesis aunque no haya nada dentro de ellos. Por ejemplo:

  ```
  ls()
  ```

  desplegará el contenido del directorio de trabajo actual.
- Los argumentos de una función pueden ser en si objetos (datos, fórmulas, matrices, tablas, etc.)

Creando Objetos

- La forma de asignar objetos en R es a través del simbolo $<-$. Por ejemplo:

```
> x<- 56

> X<- 23
>x;X
[1] 56 [1] 23
>
>  n<- sqrt(X)
>  n
[1] 4.795832
>  m<- 3+n
>  m
[1] 7.795832
>  m.aux<-10*n
>  m.aux
```

Creando Objetos

- Para borrar objetos de la memoria se usa la función `rm()`

```
> ls()
[1] "m"        "m.aux"    "n"        "x"        "X"
> rm(X)
> rm(n,x)
> ls()
[1] "m"        "m.aux"    "m.aux.2"
>
```

Lectura de Datos

- Para leer datos desde un archivo se pueden utilizar las funciones `read()`, `table()` o `scan()`. Esta última es más flexible y permite especificar el tipo de cada variable.

```
> mydata<-scan(file="./NMV.dat2", what=list("",0,0))
Read 15 records
> mydata
```

```
[[1]]
[1] "18.61664" "19.43575" "20.20695" "21.84337" "21.34864"
[[2]]
[1] 20.48832 17.99986 21.38629 17.97225 22.99391
[[3]]
[1]18.70510 20.18638 21.75702 22.66418 20.04545
```

- Se puede notar que la primera variables es de tipo caracter y las otras dos de tipo numérico

Respaldo de Datos

- Para salvar o respaldar datos en un archivo uno de los comandos que se puede utilizar es la función `write()`.

```
> write(datos,file="./salida.R", ncol=2)
```

Ayudas

- R ofrece ayuda en línea a través de la función `help()`.

```
>help(rm)
>?rm
```

- R hace también búsquedas inteligentes

```
>help.search("rose diagram")
```

```
Help files with alias or concept or title matching 'rose diagram'
using fuzzy matching:

rose.diag(CircStats)      Rose Diagram
rose.diag(circular)       Rose Diagram

Type 'help(FOO, package = PKG)' to inspect entry 'FOO(PKG) TITLE'.
```

- Adicionalmente, se pueden hacer búsquedas en el sitio web de R.

```
>RSiteSearch("rose diagram")
```

Generación de Datos

Secuencias Regulares

- Algunas susesiones se pueden generar de la siguiente manera:

```
> x<-1:15
> x
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
> y<-seq(1,5,0.5)
> y
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> w<-seq(length=9,from=1,to=5)
> w
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

Generación de Variables Aleatorias

- Generando observaciones de variable aleatorias

```
> x<-rnorm(1000)
> y<-rnorm(1000,5,3)
> u<-runif(2000,-1,1)
> z<-rexp(1000,2)
```

- No sólo se pueden generar observaciones de variable aleatorias. También se puede obtener la densidad, la probabilidad acumulada y los cuantiles de la correspondiente variable.

```
dnorm(x, mean=0, sd=1)
pnorm(q, mean=0, sd=1)
qnorm(p, mean=0, sd=1)
rnorm(n, mean=0, sd=1)
```

Manipulación de vectores y matrices

- La forma más común de crear vectores es con la función `c()`.

```
> x<-c(1,2,3,4,5,6)
> x
[1] 1 2 3 4 5 6
> y<-c(6,7)
> z<-c(y,x,y)
> z
 [1] 6 7 1 2 3 4 5 6 6 7
```

- Las operaciones aritméticas $(+, -, *, /, \%$, etc.) entre vectores se realizan elemento a elemento.

```
> X<-c(10,11,12,100,-5,-6)
> x*X
[1]  10  22  36 400 -25 -36
> X+1
[1]  11  12  13 101  -4  -5
```

Manipulación de vectores y matrices

- Las matrices pueden ser creadas a partir de vectores o directamente usando la función `matrix()`.

```
> matrix(data=5,nrow=2,ncol=3)
     [,1] [,2] [,3]
[1,]    5    5    5
[2,]    5    5    5
```

```
> matrix(1:6,2,3)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
> matrix(0,ncol=3,nrow=2)
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
```

```
> matrix(1:6,2,3,byrow=T)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Manipulación de vectores y matrices

- La multiplicación de matrices, de dimensiones adecuadas, se realiza a través de la función %*%.

- La transpuesta de una matriz se obtiene con la función `t()`.

```
> t(M2)
     [,1] [,2]
[1,]    1    1
[2,]    1    1
[3,]    2    2
[4,]    2    2
```

```
> M1%*%M2
     [,1] [,2] [,3] [,4]
[1,]    2    2    4    4
[2,]    2    2    4    4
[3,]    4    4    8    8
[4,]    4    4    8    8
```

```
> M2%*%M1
     [,1] [,2]
[1,]   10   10
[2,]   10   10
```

Accesando los valores de un objeto

- El sistema de índices: La nomenclatura `[i,]` y `[,j]` es usada en R para hacer referencia a un renglón completo o a una columna completa de una matrix.

```
> M<-matrix(1:20,4,5)
> M
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

```
> Mrow1<-M[1,]

> Mrow1
[1]  1  5  9 13 17
```

```
> M[,2]

[1] 5 6 7 8
```

Funciones

- R tiene funciones especiales para matrices, por ejemplo `solve()` para invertir, `qr()` para la descomposición QR, `eigen()` para obtener los eigenvalores y eigenvectores, `svd()` para obtener la descomposición de valor singular, etc.
- En R uno puede encontrar:
  - Funciones matemáticas básicas: `log, exp, log10, log2, sin, cos, tan, asin, acos, abs, sqrt`, etc.
  - Funciones especiales: `gamma, digamma, beta, besselI`, ...
  - Funciones estadísticas: `mean, median, lm`,...
  - Algunas otras funciones como: `sum(x)`=suma de los elementos de `x, max(x), min(x), wich, wich.max`, etc.

Creando tus propias funciones

- Una función en R puede tener cualquier número de argumentos y las operaciones que esta realice pueden ser producto de expresiones en R.
- La sintaxis general para la definición de una función es:

```
function(arguments){expression}
```

  donde **arguments** son los argumentos de la función separados por comas y **expression** es cualquier estructura permitida en R. El valor de la última línea dentro de la estructura **expression** será el valor que retorne la función.

Creando tus propias funciones

Ejemplo 1.

- La siguiente función retorna la suma de los cuadrados de los elementos del vector `x`.

```
> myfunction<-function(x){ sum(x*x) }
```

- La función `myfunction` ahora puede ser usada de la siguiente manera:

```
> z<-1:50
> y<-myfunction(z)
> y
[1] 42925
```

Graficando con R

- R ofrece una gran variedad de gráficos, aunado a la posibilidad y flexibilidad de crearlos y personalizarlos.
- Para tener una idea de las gráficas que ofrece R se puede ejecuta el siguiente comando:

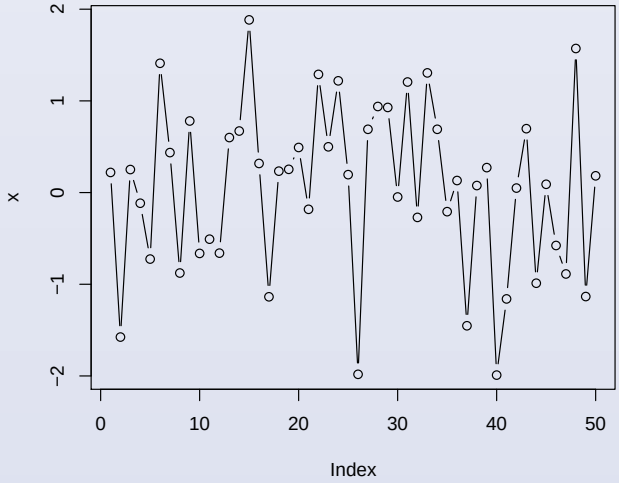
```
> demo(graphics)
```

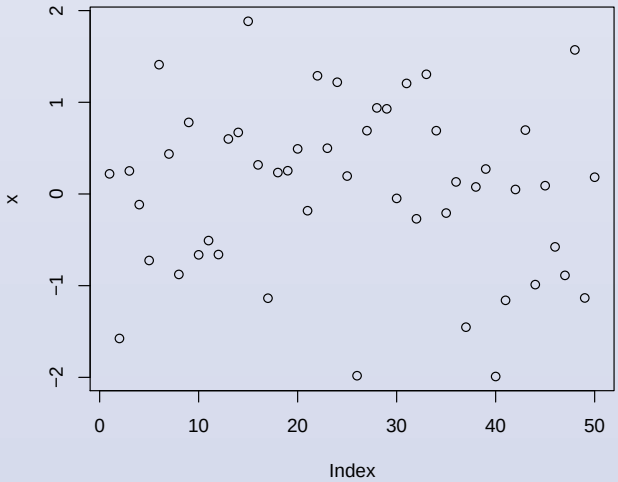
- Sería difícil exponer en esta presentación todas las opciones y posibilidades que ofrece R en terminos gráficos. De manera particular, cada función gráfica tiene un gran número de opciones (argumentos), lo que resulta en una amplia flexibilidad en la construcción de gráficos.

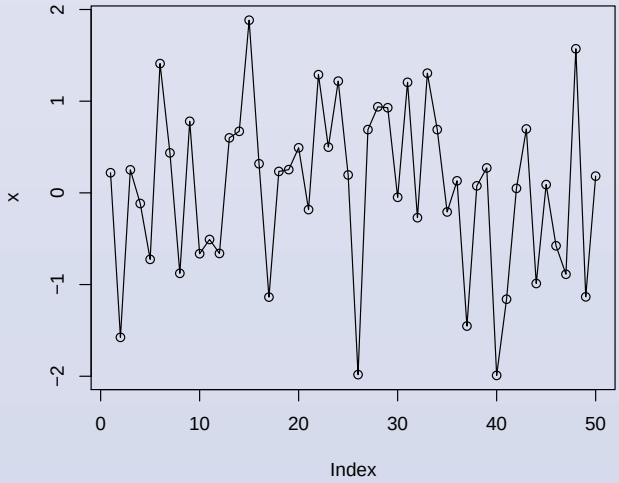
Gráficos en R

- Notemos la flexibilidad de las funciones gráficas.

```
> x<-rnorm(50)
> plot(x)
> plot(x,type="b")
> plot(x,type="o")
```

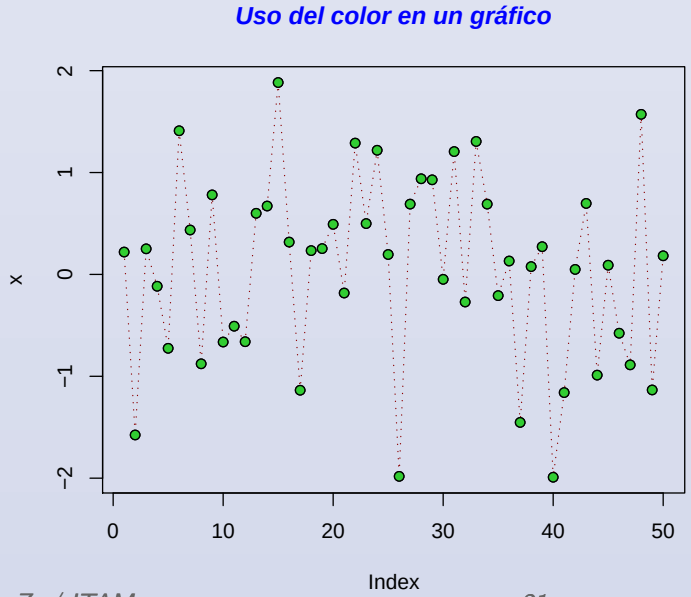






- Uso del color en los gáficos.

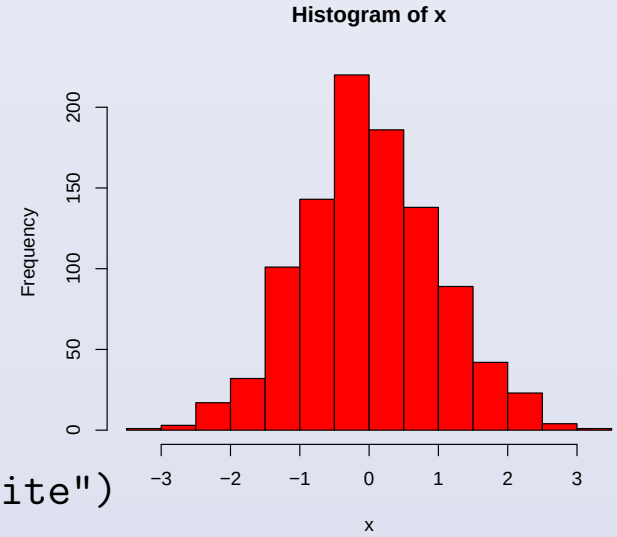
```
>plot(x)
>lines(x, col = "red4", lty = "dotted")
>points(x, bg="limegreen", pch = 21)
>title(main = "Uso del color en un  grafico",cex.main =
+ 1.2,font.main = 4,col.main = "blue")
```

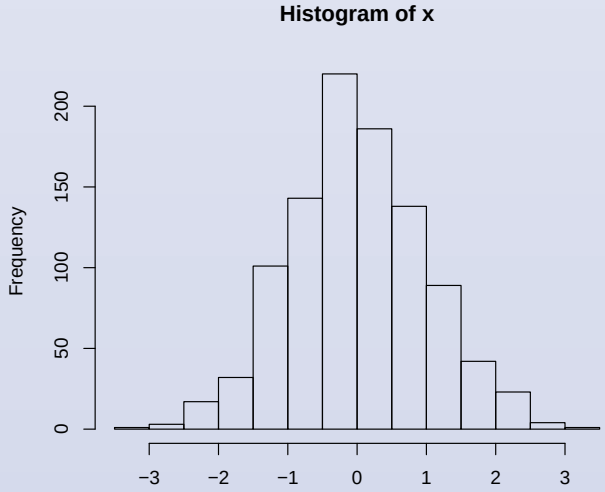


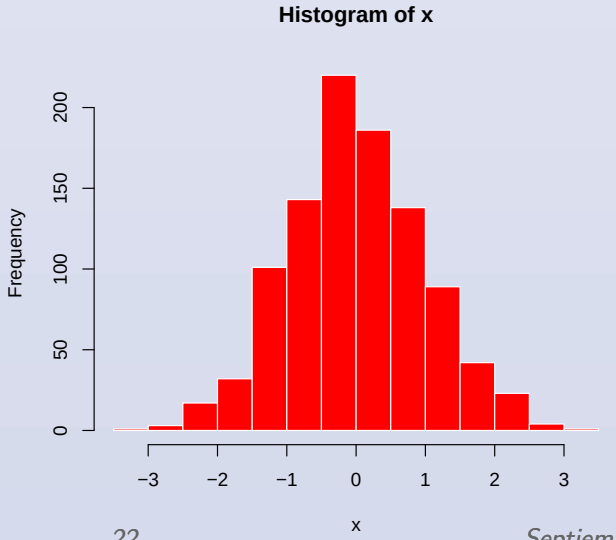
Gráficos en R

- Continuemos con la flexibilidad de las funciones gráficas.

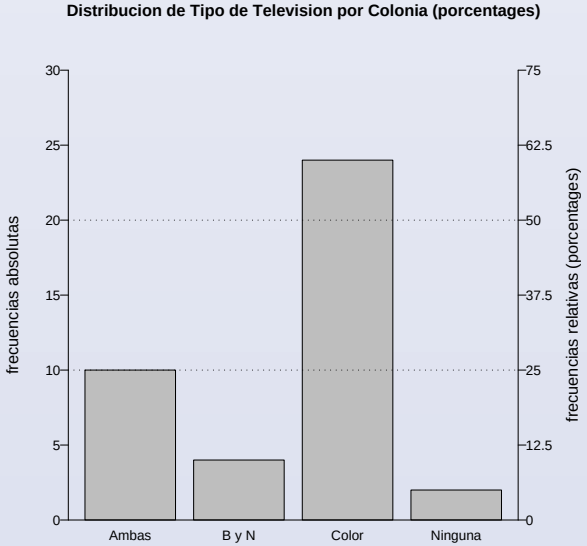
```
> x<-rnorm(1000)
>
>
> hist(x)
> hist(x,col="red")
> hist(x,col="red",border="white")
```

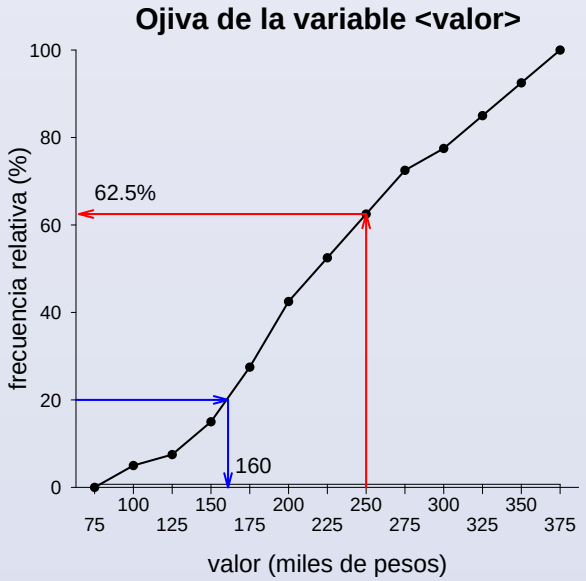




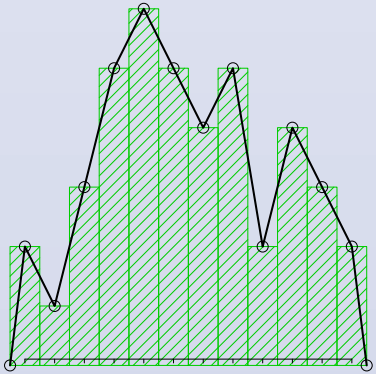


Gráficos Personalizados

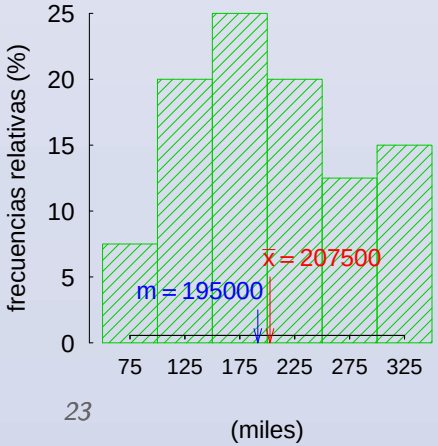




Histograma y poligono de frecuencias



Medidas de tendencia central



Análisis Estadísticos usando R

- R no sólo ofrece una gran flexibilidad en la construcción de gráficos, también ofrece una amplia gama de posibilidades para realizar análisis estadísticos (tanto descriptivos como inferenciales). A continuación se muestran sólo algunos ejemplos.

Ejemplo 1

```
>datos<-rnorm(100, 2, 4)

#Muestra de 100 observaciones normales con media 2 y desv. est.
```

- La función `summary()` calcula algunas estadísticas descriptivas.

```
   >summary(datos)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-6.2090  0.2729  2.5220  2.6040  5.0040 11.5000
```

Ejemplo 2

- A continuación se muestra el ajuste de un modelo de regresión lineal múltiple. Los datos asociados a las variable independientes (x) y a la variable dependiente (y) se pueden leer de una base de datos con ayuda de las funciones `scan()` o `read.tabla()`. Para este ejemplo, se generaron de la siguiente manera.

```
> y=rnorm(10)
> x1=c(1:10)
> x2=c(rep(1,5),rep(0,5))
> x3=rnorm(10)
```

- La función `lm()` es usada para ajustar modelos lineales.

```
>modelo1<-lm(y~x1+x2+x3)   # Se define y ajusta el modelo.
> modelo1
Call: lm(formula = y ~ x1 + x2 + x3)
Coefficients: (Intercept)       x1          x2          x3
                2.8520      -0.3599     -1.8483      0.3328
```

  ⋆ Observe de qué tipo es el objeto `modelo1`.

Ejemplo 2 (Continuación ...)

- Con la ayuda de la función genérica `summary()` se pueden obtener algunos resultados resumen del ajuste.

```
>summary(modelo1)
```

```
Call: lm(formula = y ~ x1 + x2 + x3)

Residuals:
    Min      1Q  Median      3Q     Max
-0.6709 -0.3335 -0.0218  0.2858  0.7165

Coefficients:
            Estimate    Std. Error  t value  Pr(>|t|)
(Intercept)      2.8520     1.0690   2.668   0.0371 *
x1              -0.3599     0.1305  -2.759   0.0329 *
x2              -1.8483     0.7467  -2.475   0.0481 *
x3               0.3328     0.3616   0.920   0.3930
---
Residual standard error: 0.5758 on 6 degrees of freedom Multiple
R-Squared: 0.5797,     Adjusted R-squared: 0.3695 F-statistic: 2.758
on 3 and 6 DF,  p-value: 0.1342
```

Ejemplo 2 (Continuación ...)

- Se puede notar que la función `summary()` se puede aplicar tanto a un vector de datos como a un modelo. He aquí la flexibilidad de R.
- La tabla de ANOVA para un análisis de regresión múltiple se obtiene usando la función `anova()`

```
> anova(modelo1)
```

```
Analysis of Variance Table

Response: y
          Df  Sum Sq   Mean Sq   F value  Pr(>F)
x1         1  0.17703  0.17703   0.5339  0.49250
x2         1  2.28595  2.28595   6.8942  0.03929 *
x3         1  0.28074  0.28074   0.8467  0.39299
Residuals  6  1.98945 0.33158
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ejemplo 2 (Continuación ...)

- En realidad `modelo1` es un objeto tipo lista, todos sus componentes se pueden obtener con ayuda de la función `names()`.

```
> names(modelo1)
 [1] "coefficients"  "residuals"   "effects"     "rank"
 [5] "fitted.values" "assign"      "qr"          "df.residual"
 [9] "xlevels"       "call"        "terms"       "model"
```

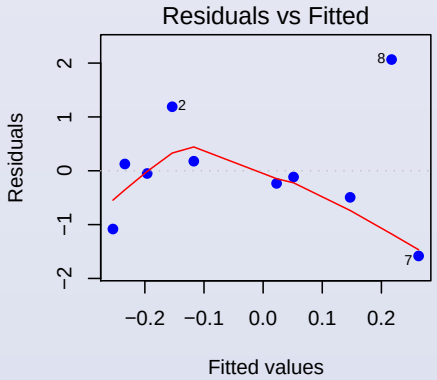
- Así , sus componentes se pueden accesar, por ejemplo, con:

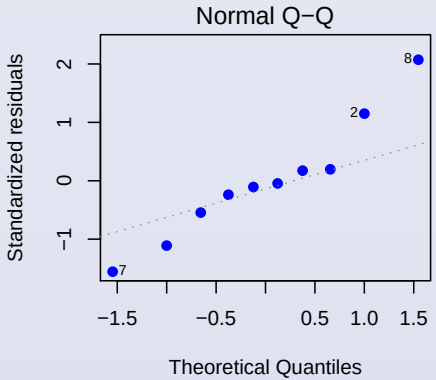
```
> modelo1$residuals
          1           2           3           4           5
-0.23464471  1.18946063 -0.05046907  0.17749660 -1.08184345
          6           7           8           9          10
-0.49368395 -1.58228384  2.06572986 -0.11623393  0.12647186
```

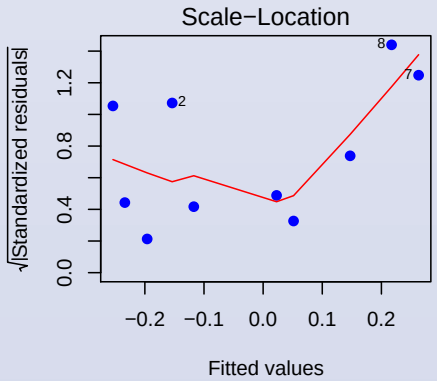
Ejemplo 2 (Continuación ...)

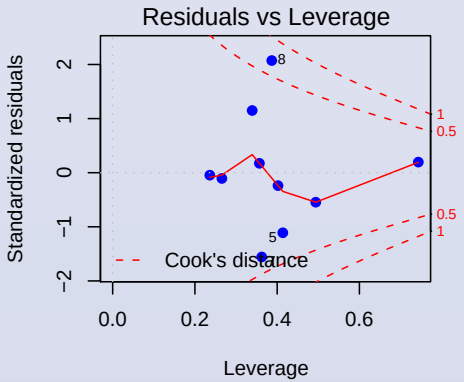
- Ahora, recordando que modelo1 es un objeto de tipo lista, al que se le asigno un ajuste de regresión lineal, ...

```
>plot(modelo1)
```









Análisis Estadísticos usando R

- Pruebas de hipótesis, paramétricas y no paramétricas:

```
t.test(), var.test(), cor()
wilcox.test(), cor.test(datos,method="spearman")
binom.test(), kruskal.test()
⋮
```

- Análisis y ajuste de modelos:

`lm()`	Modelos de regresión
`glm()`	Modelos lineales generalizados
`survfit(), coxph()`	Análisis de supervivencia
`prcomp()`	Análisis multivariado
`hclust(), plclust`	
`density()`	Estimadores de kernel
`ts.plot(), acf()`	Series de tiempo
⋮	

Paquetes Adicionales (Contribuciones)

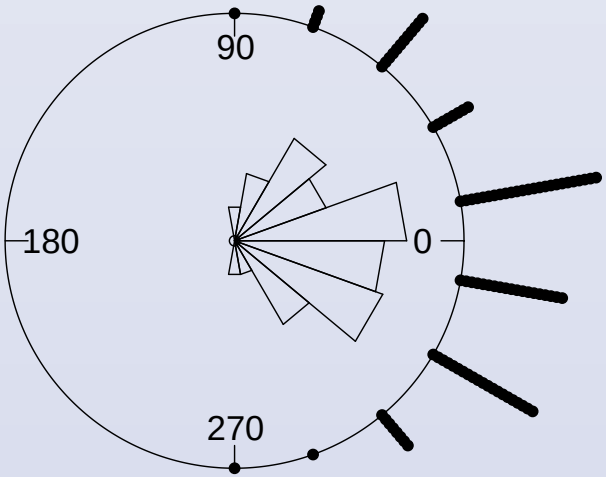
- En R:
  1. Packages|| Install package(s) || *elegir el paquete*
  2. Packages|| load package(s) || *elegir el paquete*

Paquete: **CircStats**

```
> help(rvm)
> . . .
> data.vm <- rvm(100, 0, 3)

> rose.diag(data.vm, bins = 18,
+ pts = TRUE, shrink=1.5,prop=1.5)

> title("Grafica de Datos Circulares")
```



Comentarios Finales y Referencias

- A la fecha existen existen más de 1500 paquetes, en el sitio web, que ajustan modelos específicos, grafican datos muy particulares (datos direccionales), etc.
- R resulta no sólo en una opción sino una **buena opción** para la graficación y el análisis estadístico.
  - Ihaka, R. y Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, **5**, 3, 299-314.
  - `http://cran.r-project.org`
  - Dalgaard, P. (2002). *Introductory Statistics with R*. Springer-Verlag. New York.
  - Albert, J. (2007). *Bayesian Computation with R*. Springer-Verlag. New York.