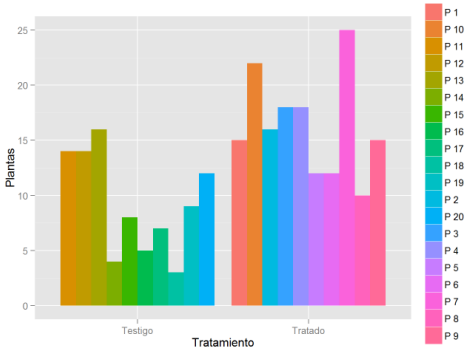
Perú

# R para universitarios

R es un entorno de trabajo que permite realizar cálculos matemáticos, estadísticos y gráficos de forma potente.

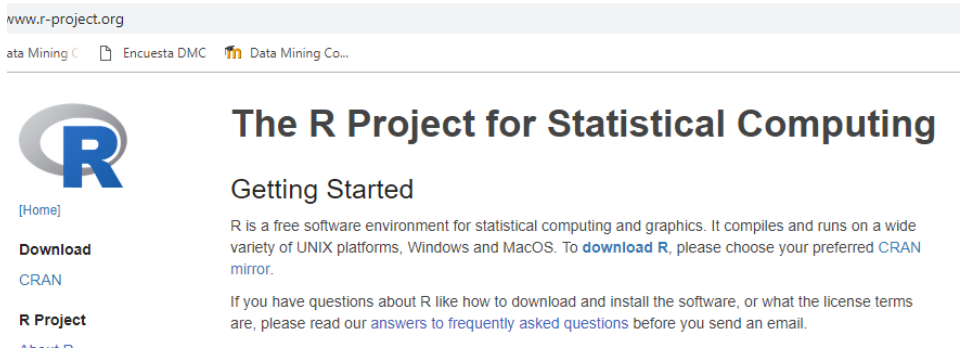
## Ventajas de la programación en R:

- Es un software libre y de mayor uso en estadística.
- Esta avalado por una comunidad científica mundial.
- Los usuarios pueden manipular y adaptar los paquetes.
- Es compatible con equipos Mac, Windows y Linux.



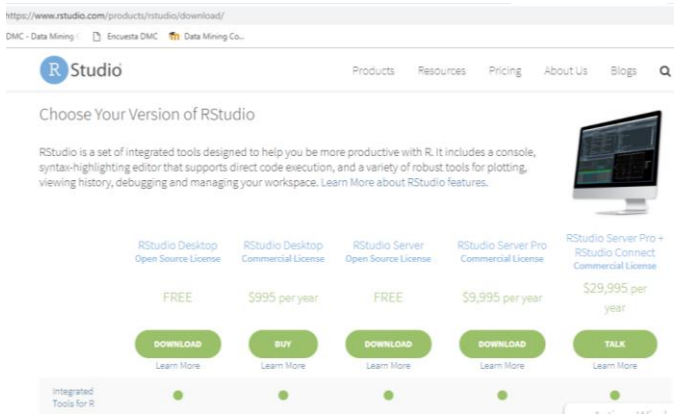

# Instalación R

https://cran.r-project.org

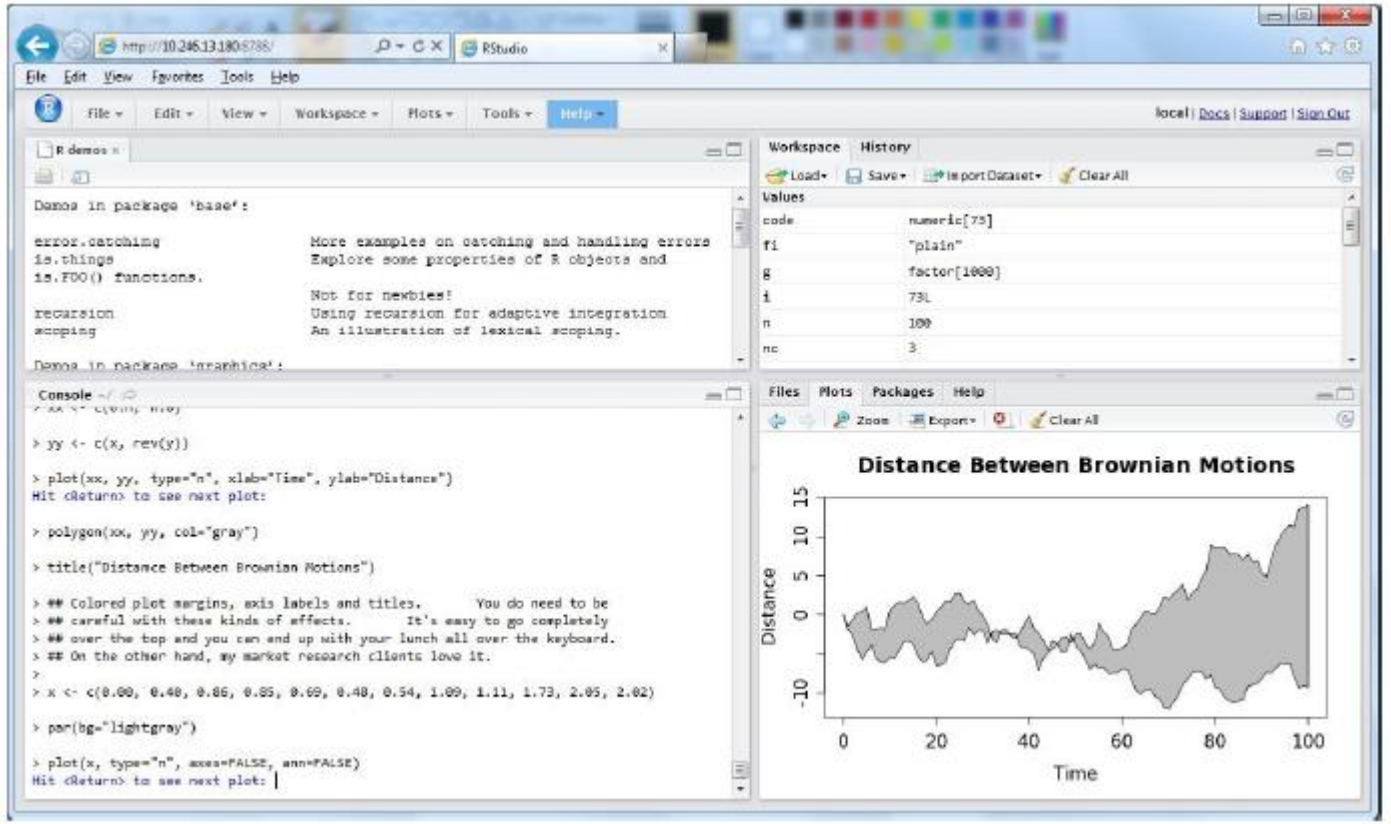



# Instalación RStudio

https://www.rstudio.com/products/rstudio/download/

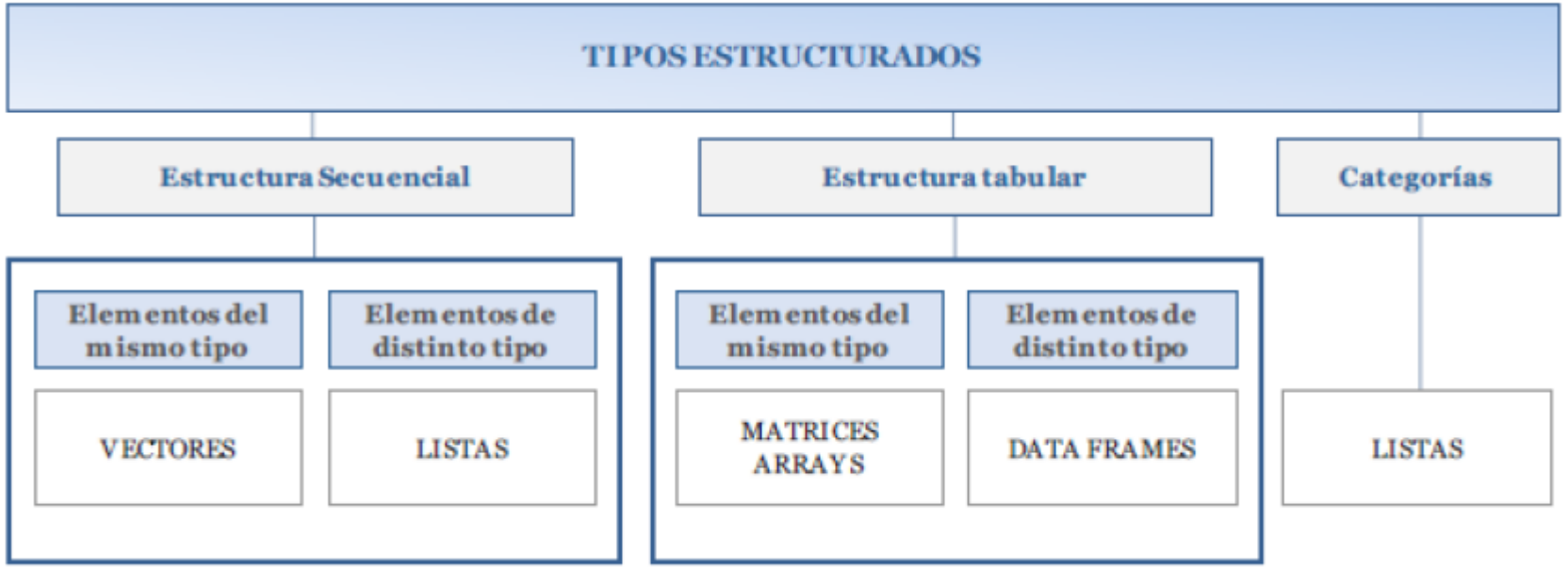


# Introducción a R y RStudio:



# Programación en R:

## Estructura de datos con R



# Programación en R:

## Funciones con vectores

| | | |
|---|---|---|
| `sum(x)` | `prod(x)` | Sum/product of the elements of `x` |
| `cumsum(x)` | `cumprod(x)` | Cumulative sum/product of the elements of `x` |
| `min(x)` | `max(x)` | Minimum/Maximum element of `x` |
| `mean(x)` | `median(x)` | Mean/median of `x` |
| `var(x)` | `sd(x)` | Variance/standard deviation of `x` |
| `cov(x,y)` | `cor(x,y)` | Covariance/correlation of `x` and `y` |
| `range(x)` | | Range of `x` |
| `quantile(x)` | | Quantiles of `x` for the given probabilities |
| `fivenum(x)` | | Five number summary of `x` |
| `length(x)` | | Number of elements in `x` |
| `unique(x)` | | Unique elements of `x` |
| `rev(x)` | | Reverse the elements of `x` |
| `sort(x)` | | Sort the elements of `x` |
| `which()` | | Indices of `TRUE`s in a logical vector |
| `which.max(x)` | `which.min(x)` | Index of the max/min element of `x` |
| `match()` | | First position of an element in a vector |
| `union(x, y)` | | Union of `x` and `y` |
| `intersect(x, y)` | | Intersection of `x` and `y` |
| `setdiff(x, y)` | | Elements of `x` that are not in `y` |
| `setequal(x, y)` | | Do `x` and `y` contain the same elements? |

# Programación en R:

## Trabajando con matrices

| | |
|---|---|
| `t(A)` | Transpose of `A` |
| `det(A)` | Determinate of `A` |
| `solve(A, b)` | Solves the equation `Ax=b` for `x` |
| `solve(A)` | Matrix inverse of `A` |
| `MASS::ginv(A)` | Generalized inverse of `A` (MASS package) |
| `eigen(A)` | Eigenvalues and eigenvectors of `A` |
| `chol(A)` | Choleski factorization of `A` |
| `diag(n)` | Create a `n`×`n` identity matrix |
| `diag(A)` | Returns the diagonal elements of a matrix `A` |
| `diag(x)` | Create a diagonal matrix from a vector `x` |
| `lower.tri(A),upper.tri(A)` | Matrix of logicals indicating lower/upper triangular matrix |
| `apply()` | Apply a function to the margins of a matrix |
| `rbind(...)` | Combines arguments by rows |
| `cbind(...)` | Combines arguments by columns and |
| `dim(A)` | Dimensions of `A` |
| `nrow(A), ncol(A)` | Number of rows/columns of `A` |
| `colnames(A), rownames(A)` | Get or set the column/row names of `A` |
| `dimnames(A)` | Get or set the dimension names of `A` |

# Programación en R:

## Creación de Funciones

Las funciones son creadas usando function() directamente y son alamacenadas como objetos de R. En particular, las funciones son objetos de R de la clase "function".

```
f <- function(<arguments>) {
##Sentencias
}
```

Las funciones en R son "first class objects", lo que significa que pueden ser tratados en R como cualquier objetos.

- Las funciones pueden pasar como argumentos a otras funciones.
- Las funciones pueden estar anidadas, es decir se puede definir una función dentro de otra.
- El valor retorno de una función es la última expresión evaluada en el cuerpo de la función.

```
superfRect=function(base,altura){
                    S=base*altura
                    return(S)
                   }
```


```
superfRect(2,3)
```

```
## [1] 6
```

```
superfRect(5,6)
```

```
## [1] 30
```

# Programación en R:

## Manejo de archivos de datos

### Archivos *.csv (comma separated values)

- Son archivos planos sin estructura adicional
- Sus columnas se separan por comas
- Pueden incluir nombres para las filas y columnas
- La función que se emplea es *read.csv(file="")*

### Archivos *.sav (archivos del SPSS)

- Son archivos generados en SPSS
- Requiere el paquete foreign
- La función que se emplea es read.spss(file="")
- Argumentos más usados: use.value.labels=TRUE, max.value.labels=TRUE, to.data.frame=TRUE

### Archivos *.txt (archivos de texto)

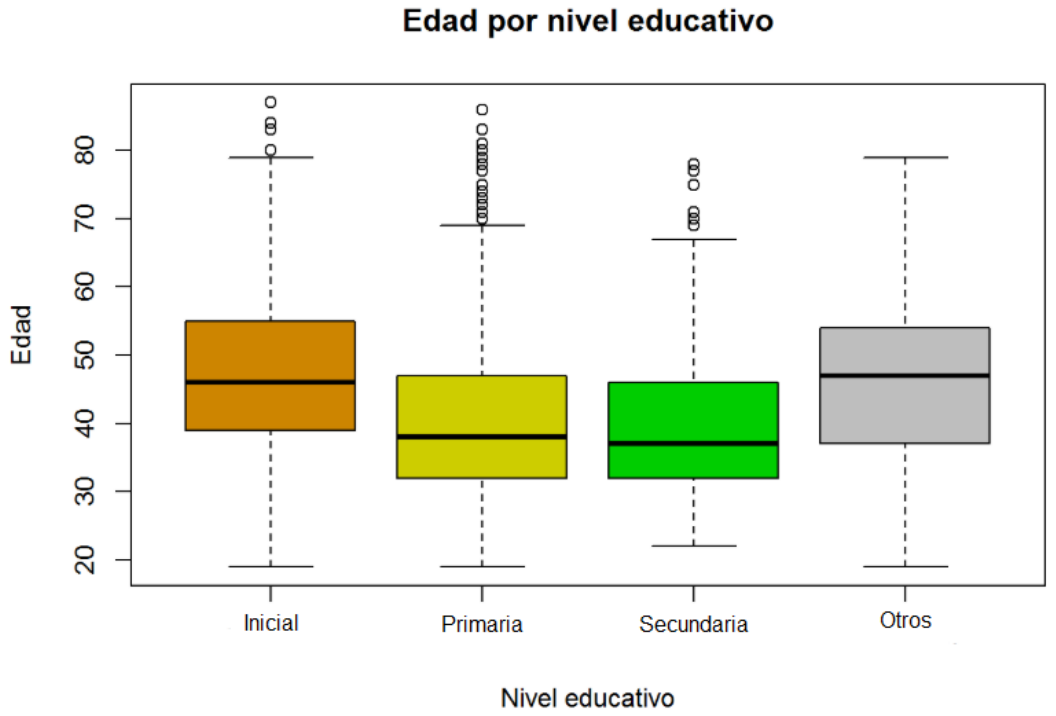
- Son archivos separados por tabulaciones, coma (,) o punto y coma (;)
- La función que se emplea es read.table(file="") *o* read.delim()
- Argumentos más usados: header=FALSE, sep="", dec="."

# ¿Qué es un valor atípico?

Valores que se encuentran alejados del resto de observaciones con más de 3 desviaciones respecto de su media.

## Causas:

- Errores de codificación.
- Observaciones ajenas a la población en estudio.
- Observaciones atípicas debidas a la distribución de la variable.



# ¿Qué es un valor perdido?

Valores que para una variable determinada no constan en algunas filas o patrones

## Consecuencias:

- La perdida de información que producen
- Pueden introducir un sesgo considerable
- Hacen el manejo del análisis más complicado

## Tratamiento:

- Eliminar los registros que tengan datos faltantes
- Imputar los valores perdidos con estimaciones

# Programación en R:

## Tratamiento de datos atípicos y perdidos

Cargamos set de datos disponibles

```
Library (datasets)
```

Cargamos set de datos “airquality”

```
base = read.csv('Base Credit Card.csv')
```

Obtenemos las características de la variable “**Ozono**”

```
summary(base$EDAD)
boxplot(base$EDAD)
```

**Eliminamos filas con nulos en columnas concretas**

```
base <- base[!is.na(base$EDAD),]
```

**Eliminamos todas las filas que contengan algún valor nulo**

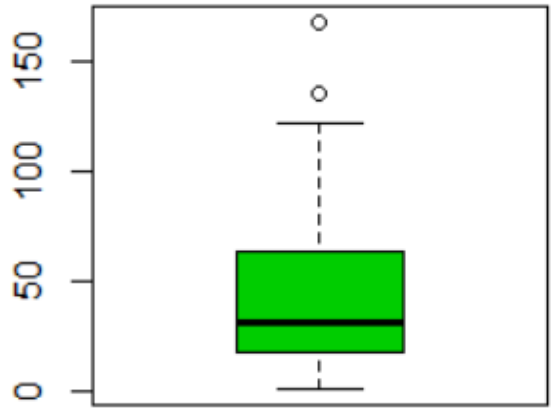
```
base <- na.omit(base)
```

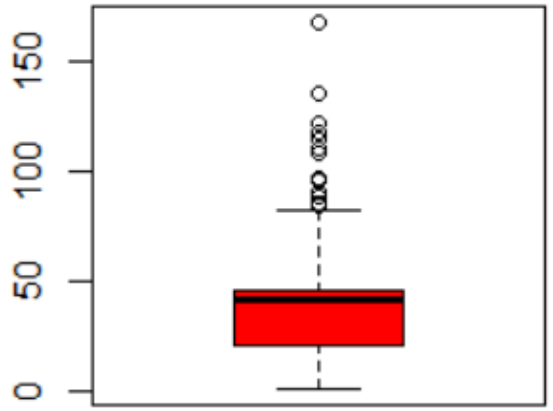
# Programación en R:

## Tratamiento de datos atípicos y perdidos

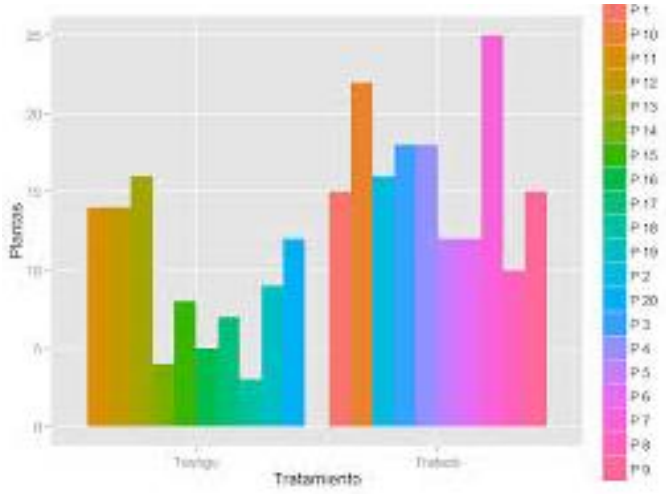
### Estimación de perdidos y atípicos

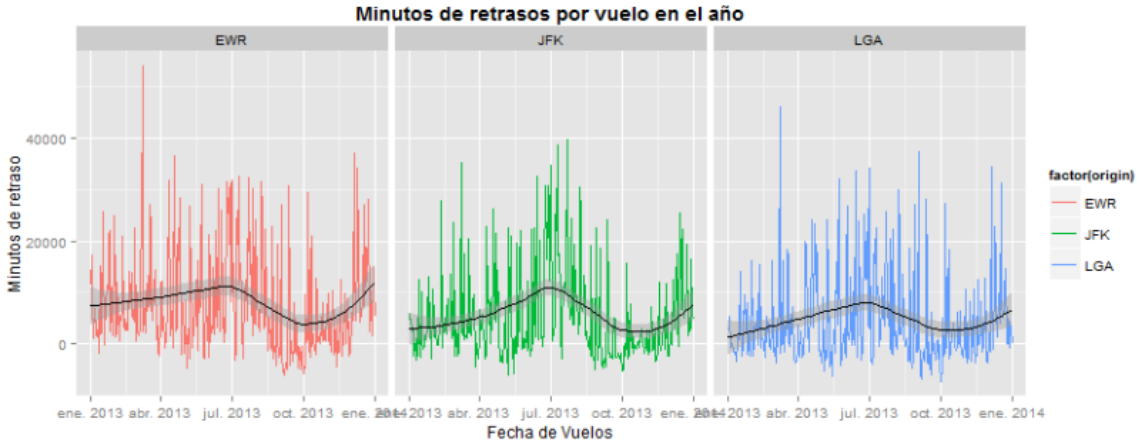
```
media = mean(base$EDAD[!is.na(base$EDAD)])
base$EDAD2 = base$EDAD
base$EDAD2[is.na(base$EDAD)] = media

Q3 = quantile(base$EDAD2,0.75)
Q1 = quantile(base$EDAD2,0.25)

base$EDAD3[base$EDAD3>(Q3+1.5*(Q3-Q1))] = (Q3+1.5*(Q3-Q1))

par(mfrow = c(1,2))
boxplot(base$EDAD2, main = "edad imputado", col  = 3)
boxplot(base$EDAD3, main = "edad sin atípicos", col = 2)
```

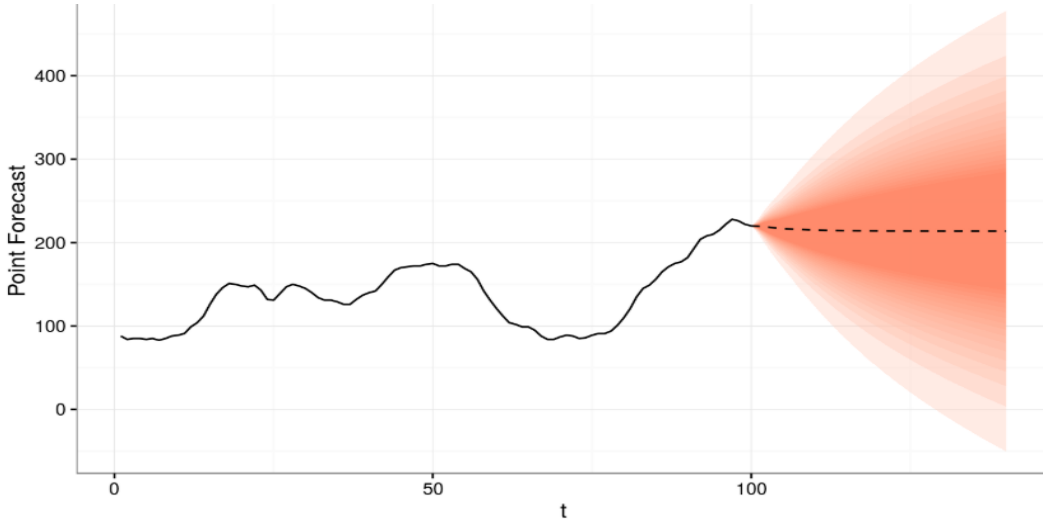


# Importancia del análisis visual

# Programación en R:

## Tipos de gráficos:

plot()
hist()
barplot()
pie()
boxplot()

## Argumentos de gráficos:

| | | | |
|---|---|---|---|
| Controlan la extensión de los ejes X, Y | xlim()<br>ylim() | Indica un título para el gráfico | main = |
| Controlan Las etiquetas de los ejes X, Y | xlab =<br>ylab = | Indica un subtítulo para el gráfico | sub = |

# Programación en R:

## Ejemplos de gráficos:

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

### Histograma:

```
hist(base$EDAD, xlim = c(0,100),ylim = c(0,190),
    xlab = "EDAD", ylab = "Frecuencia", main = "histograma")
```

### Dispersión:

```
plot(base$EDAD, base$Total_facturacion, xlim = c(0,100),ylim = c(0,190),
    xlab = "edad", ylab = "Frecuencia", main = "Dispersión")
```

### Caja:

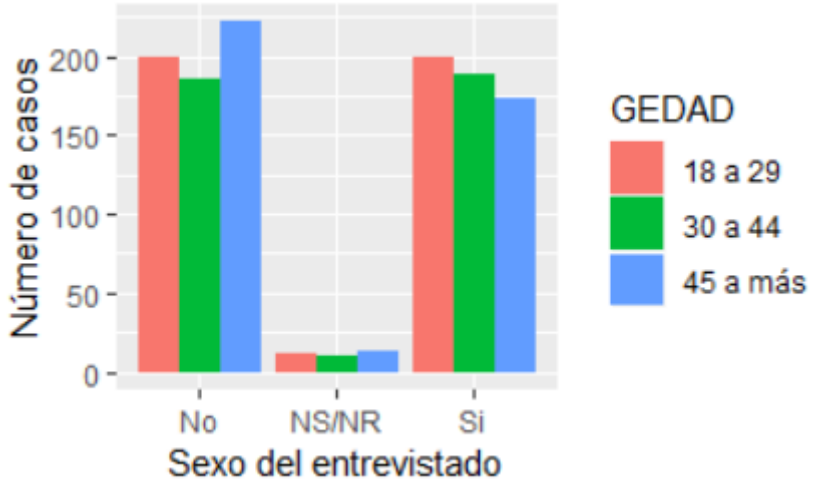
```
boxplot(base$EDAD,xlab = "edad",
      ylab = "Frecuencia", main = "Caja")
```

# Programación en R:

## Uso del paquete ggplot2:

### Barras:

```
gg1 = ggplot(datos,aes(GEDAD))+ geom_bar(width = 0.6, fill = "blue") #cambiamos el ancho de las barras
gg1 + xlab("Sexo del entrevistado") + ylab("Número de casos")
gg1 + ggtitle("Genero del entrevistado")

gg1 = ggplot(datos,aes(P36, fill = GEDAD))+ geom_bar(position = "dodge")
gg1 + xlab("Sexo del entrevistado") + ylab("Número de casos")
gg1 + ggtitle("Genero del entrevistado")
```

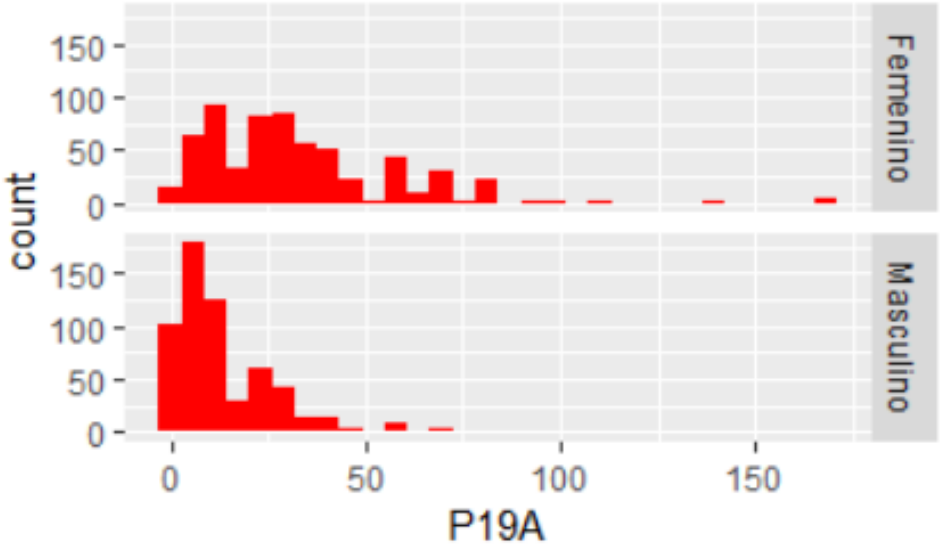


# Programación en R:

Uso del paquete ggplot2:

Histogramas:

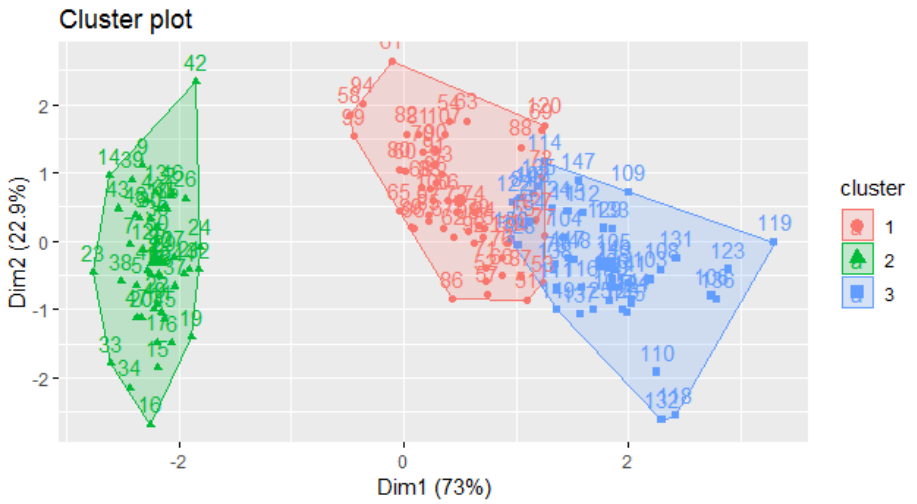
```
ggplot(datos, aes(P19A)) + geom_histogram()

hist1 <- ggplot(datos, aes(P19A)) + geom_histogram(fill =
"Red")
hist1 + facet_grid(SEXO ~.)
```



# Programación en R:

## Análisis Cluster: K - medias

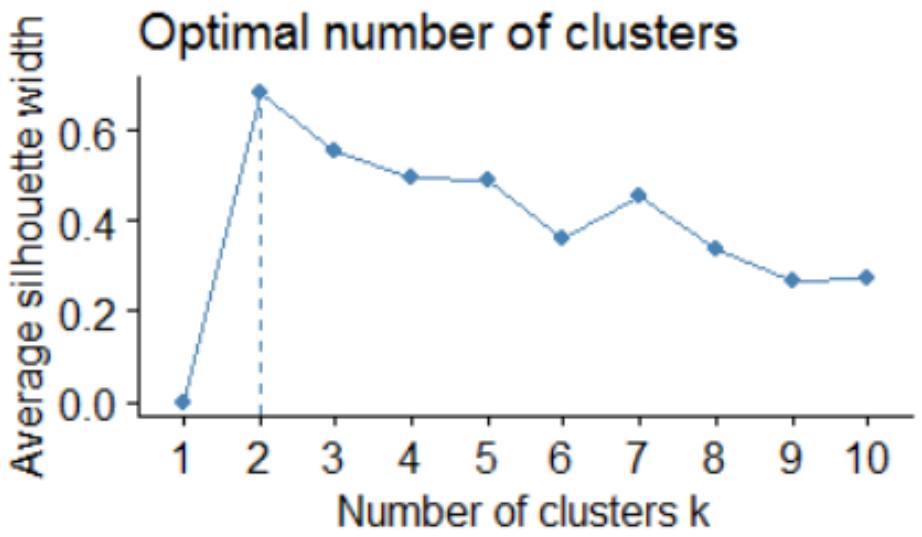
```
library(datasets)
library(cluster)
library(factoextra)

iriscluster <- kmeans(iris[,3:4],3,nstart = 20)

base = data.frame(iris)
base$cluster = iriscluster$cluster

ggplot(base, aes(Petal.Length,Petal.Width, color =
base$cluster)) + geom_point()

fviz_cluster(iriscluster,data = iris[,-5])
```

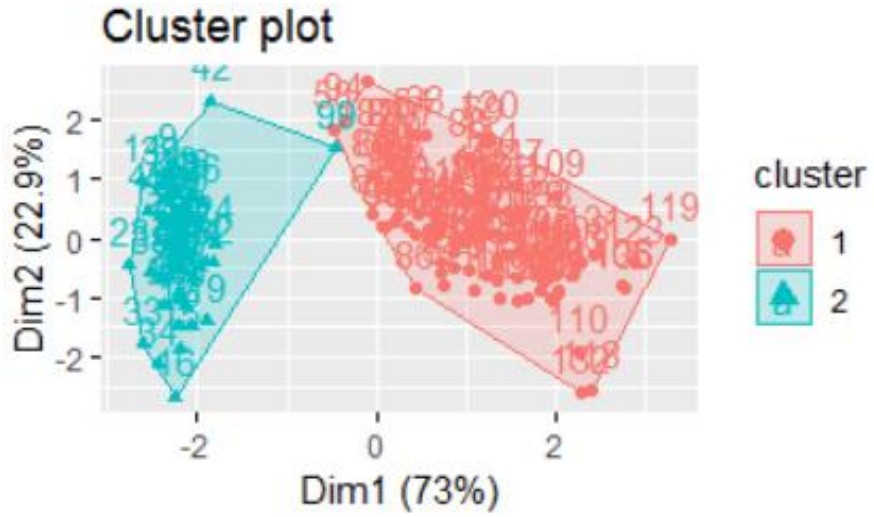


# Programación en R:

## Análisis Cluster: K - medias

```
#Número optimo de clusters
fviz_nbclust(iris[,-5],kmeans,method = "wss")
fviz_nbclust(iris[,-5],kmeans,method = "silhouette")
```

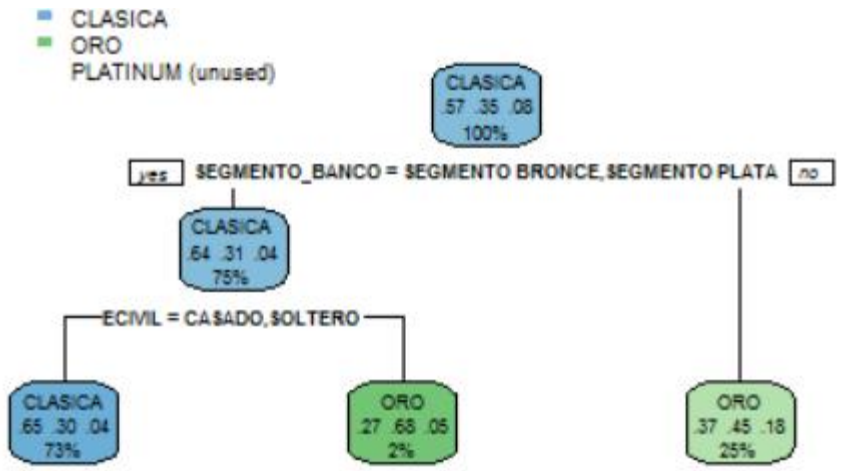


# Programación en R:

## Arboles de clasificación

```
library(rpart)
library(rpart.plot)
library(caret)

arbol1 <- rpart(formula = TIPO_TARJETA~SEGMENTO_BANCO+
                ECIVIL , data = base)
rpart.plot(arbol1)
```



# Programación en R:

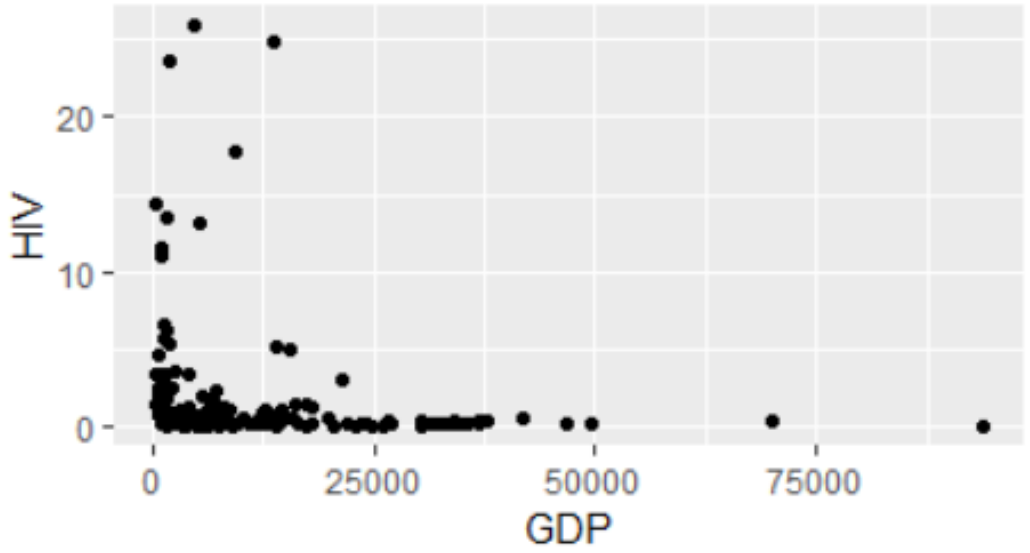
## Regresión lineal – Regresión logística:

### Correlación

```
pairs(x = base[,-1], lower.panel = NULL)
cor(x = base[,-1],method = "pearson")
```

### Normalidad

```
hist(base$GDP, main = "GDP")
qqnorm(base$GDP,main = "GDP")
shapiro.test(base$GDP)
```




### Homocedasticidad

```
ggplot(base,aes(x=GDP, y=HIV))+geom_point()
```

# Programación en R:

## Regresión lineal:

```
regre01 <- lm(Life_expentacy ~ GDP + HIV,
              data = base)
summary(regre01)

regre02 <- lm(Life_expentacy ~log(GDP + HIV),
              data = base)
summary(regre02)
```

## Regresión logística

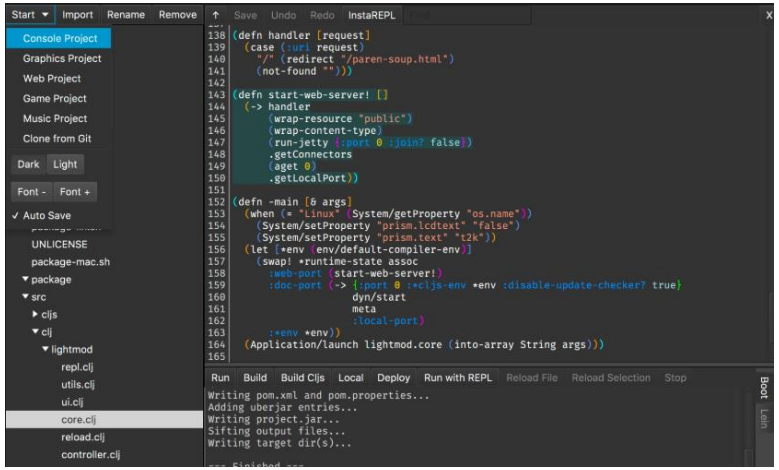
```
logi01 <- glm(Y ~ GDP + HIV,
              data = base,family = "binomial")
```

# RATTLE en R

Interface gráfica de código abierto y gratuita para el software R. Se caracteriza por apuntar y hacer clic en las tareas de minería de datos.

## GUI (interface gráfica de usuario)


## IDE (entorno de desarrollo integrado)



# Programación en R:

## RATTLE: Instalación y puesta en marcha

```
install.packages("rattle")
library(rattle)

#Para activar el visualizador
rattle()
```