



UNIVERSIDAD TÉCNICA DE MACHALA

Maestría en Software

Asignatura:

**Ciencia de Datos e Inteligencia de
Negocios**

Tema:

Tarea Práctica 4. Técnicas de Minería de Datos

Docente: Ing. Bertha Mazón, Mg. Inf.

Estudiantes:

ESTEBAN FABRICIO GONZABAY JIMENEZ

JIMMY FERNANDO CASTILLO Crespín

CESAR DAVID SANTILLAN VILLOTA

JORGE LUIS MIRANDA GALLEGOS

2021-2022

¿Que es un archivo “csv”?

De las siglas Comma-Separated values, es un archivo de texto que almacena los datos en forma de columnas, separadas por coma y las filas se distinguen por saltos de línea.

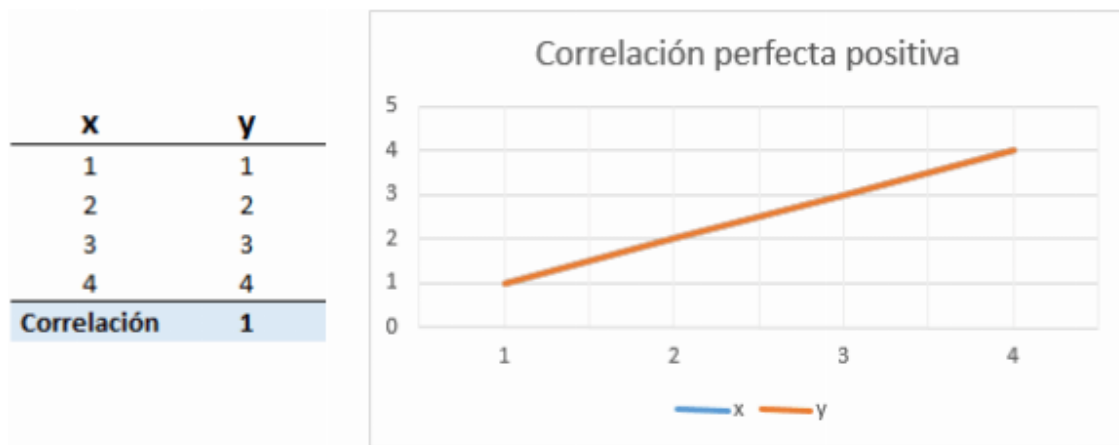
Estos archivos normalmente se usan para importar o exportar de base de datos de unas aplicaciones.

Libreria “Readr”

Readr es una componente del denominado “tidyverso”, un conjunto de librerías que todo usuario de R debería si no dominar, al menos conocer, para así resolver ciertas situaciones de la manera más sencilla posible.

Análisis de Correlación

El análisis de correlación es un enfoque estadístico que se utiliza para determinar la relación entre variables cuantitativas o categóricas.



Packages (“corrplot”)

El paquete corrplot es una representación gráfica de una matriz de una correlación, intervalo de confianza. También contiene algunos algoritmos para ordenar matrices. Además, corrplot es bueno para los detalles, incluida la elección del color, las etiquetas de texto, las etiquetas de color, el diseño, etc.

Métodos de visualización “corrplot”

Hay siete modos de visualización (parámetro method) en el paquete corrplot, llamado “circle”, “square”, “ellipse”, “number”, “shade”, “color”, “pie”.

Correlación método de pearson

El coeficiente de correlación de Pearson es una prueba que mide la relación estadística entre dos variables continuas. Si la asociación entre los elementos no es lineal, entonces el coeficiente no se encuentra representado adecuadamente.

La escala de medida debe ser una escala de intervalo o relación.

- [illegible]

En la prueba estadística el Coeficiente de Concordancia de Kendall (W), ofrece el valor que posibilita decidir el nivel de concordancia entre los expertos. El valor de W oscila entre 0 y 1. El valor de 1 significa una concordancia de acuerdos total y el valor de 0 un desacuerdo total. La tendencia a 1 es lo deseado pudiéndose realizar nuevas rondas si en la primera no es alcanzada significación en la concordancia.

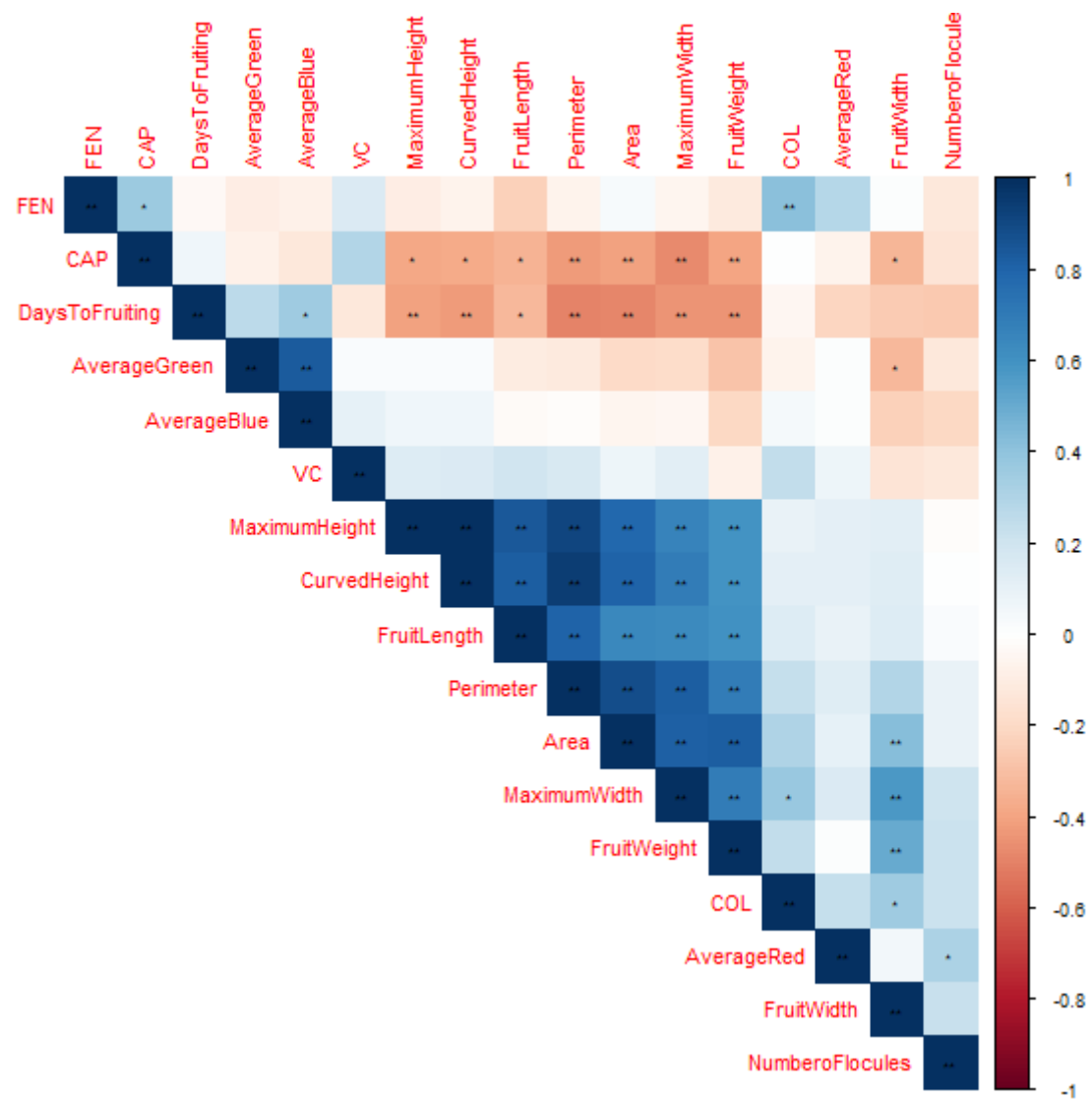
```

33 #Correlación método de Kendall
34 library("corrplot")
35 M<-cor(data, method = "kendall")
36 head(round(M,1),71)
37 res1 <- cor.mtest(data, method = "kendall", conf.level = .95)
38 cex.before <- par("cex")
39 par(cex = 0.7)
40 corrplot(M, p.mat = res1$p, method = "color", type = "upper",
41          sig.level = c(.01, .05), pch.cex = .7,
42          insig = "label_sig", pch.col = " black", order = "AOE")
43
44 library(corrplot)
45 M<-cor(data, method = "pearson")
46 #Visualizar Matriz de valores de correlacion
47 head(round(M,2),71)
48 <

```

44:1 (Top Level) ↕

Console	Terminal ×	
~/		
MaximumWidth	0.3 -0.3	-0.2 0.4 0.5 0.5 0.1
MaximumHeight	0.1 -0.3	-0.2 0.7 0.1 0.4 0.0
CurvedHeight	0.1 -0.3	-0.2 0.7 0.1 0.4 0.0
AverageRed	0.2 0.1	-0.1 0.0 0.1 0.0 0.3
AverageGreen	-0.1 -0.1	0.2 0.0 -0.2 -0.2 -0.1
AverageBlue	0.0 0.0	0.2 0.0 -0.2 -0.2 -0.1
VC	0.1 0.2	-0.1 0.1 0.0 -0.1 -0.1
FEN	0.3 0.2	-0.1 -0.2 0.0 -0.2 -0.1
COL	1.0 0.1	0.0 0.1 0.3 0.1 0.2
CAP	0.1 1.0	0.1 -0.2 -0.3 -0.4 0.0
DaysToFruiting	0.0 0.1	1.0 -0.1 -0.1 -0.2 -0.2
FruitLength	0.1 -0.2	-0.1 1.0 0.2 0.4 0.0
Fruitwidth	0.3 -0.3	-0.1 0.2 1.0 0.4 0.2
Fruitweight	0.1 -0.4	-0.2 0.4 0.4 1.0 0.2
NumeroFlocules	0.2 0.0	-0.2 0.0 0.2 0.2 1.0
> res1 <- cor.mtest(data, method = "kendall", conf.level = .95)		
There were 50 or more warnings (use warnings() to see the first 50)		
> cex.before <- par("cex")		
> par(cex = 0.7)		
> corrplot(M, p.mat = res1\$p, method = "color", type = "upper",		
+ sig.level = c(.01, .05), pch.cex = .7,		
+ insig = "label_sig", pch.col = " black", order = "AOE")		



Mapa de calor y Cluster en base a la correlación de pearson

Los mapas de calor visualizan los datos a través de variaciones en el color. Cuando se aplican a un formato tabular, los mapas de calor son útiles para el examen cruzado de datos multivariados, mediante la colocación de variables en las filas y columnas, y coloreando las celdas dentro de la tabla.

Los mapas de calor son buenos para mostrar la varianza a través de múltiples variables, revelando cualquier patrón, mostrando si las variables son similares entre sí y para detectar si existen correlaciones entre ellas.

```

44 library(corrplot)
45 M<-cor(data, method = "pearson")
46 #Visualizar Matriz de valores de correlacion
47 head(round(M,2),71)
48 col<- colorRampPalette(c("blue", "white", "red"))(20)
49 heatmap(x = M, col = col, symm = TRUE)
50 |

```

50:1 (Top Level) ▾

Console Terminal x

```

~/

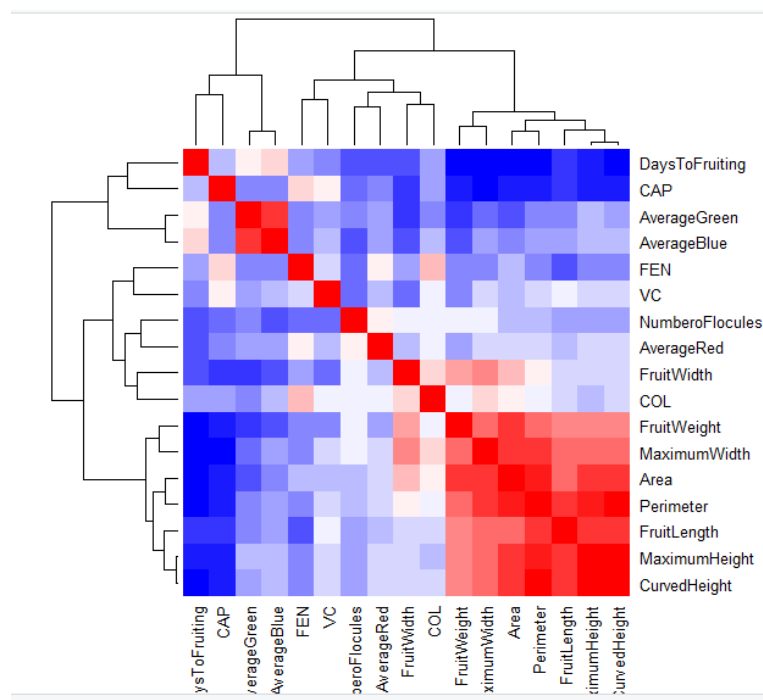
```

	COL	CAP	DaysToFruiting	FruitLength	Fruitwidth	Fruitweight	NumeroFlocules
Fruitweight	0.70	0.83	0.70	0.59	0.60	0.01	-0.29
NumeroFlocules	0.10	0.10	0.21	-0.02	0.01	0.31	-0.12
Perimeter	0.23	-0.42	-0.50	0.80	0.29	0.70	0.10
Area	0.30	-0.41	-0.48	0.64	0.43	0.83	0.10
Maximumwidth	0.37	-0.48	-0.44	0.63	0.58	0.70	0.21
MaximumHeight	0.10	-0.38	-0.40	0.84	0.13	0.59	-0.02
CurvedHeight	0.12	-0.38	-0.43	0.83	0.13	0.60	0.01
AverageRed	0.23	-0.07	-0.22	0.10	0.06	0.01	0.31
AverageGreen	-0.07	-0.08	0.27	-0.11	-0.32	-0.29	-0.12
AverageBlue	0.05	-0.12	0.35	-0.02	-0.23	-0.21	-0.20
VC	0.24	0.29	-0.12	0.20	-0.14	-0.07	-0.13
FEN	0.42	0.37	-0.04	-0.24	0.02	-0.11	-0.12
COL	1.00	0.01	-0.05	0.14	0.36	0.24	0.22
CAP	0.01	1.00	0.07	-0.34	-0.34	-0.39	-0.15
DaysToFruiting	-0.05	0.07	1.00	-0.32	-0.26	-0.44	-0.26
FruitLength	0.14	-0.34	-0.32	1.00	0.15	0.61	0.02
Fruitwidth	0.36	-0.34	-0.26	0.15	1.00	0.50	0.23
Fruitweight	0.24	-0.39	-0.44	0.61	0.50	1.00	0.21
NumeroFlocules	0.22	-0.15	-0.26	0.02	0.23	0.21	1.00

```

> col<- colorRampPalette(c("blue", "white", "red"))(20)
> heatmap(x = M, col = col, symm = TRUE)
> |

```



Paquete "Hmisc"

El lenguaje R funciona mediante la adición de paquetes elaborados por diferentes usuarios. Cada paquete realiza operaciones o cálculos específicos. La biblioteca Hmisc contiene funciones útiles para análisis de datos, como ofrecer una matriz de correlaciones de Pearson.

```

52 #instalar paquete una sola vez
53 #install.packages("Hmisc")
54 library("Hmisc")
55 res<- rcorr(as.matrix(data) , type ="pearson")
56 #muestra matriz de coeficientes de correlación
57 res$r
58 write.csv(res$r, "correlacion_pearson.csv")
59 #muestra matriz de p-values
60 res$p
61 write.csv(res$p, "correlacion_pearson_pvalues.csv")
62
<
62:1 (Top Level)

```

Console Terminal x

```

~/
> res$r

```

	Perimeter	Area	Maximumwidth	MaximumHeight	CurvedHeight	AverageRed	AverageGreen	AverageBlue
Perimeter	1.00000000	0.88211328	0.82475615	0.91858051	0.943019707	0.13946913	-0.11529640	-0.01753745
Area	0.88211328	1.00000000	0.81945801	0.78640681	0.802321236	0.10568016	-0.19834741	-0.05136440
Maximumwidth	0.82475615	0.81945801	1.00000000	0.66279273	0.698285528	0.15070951	-0.18207461	-0.04561751
MaximumHeight	0.91858051	0.78640681	0.66279273	1.00000000	0.991213560	0.11897513	0.02684452	0.06937904
CurvedHeight	0.94301971	0.80232124	0.69828553	0.99121356	1.000000000	0.11944196	0.02024560	0.06574699
AverageRed	0.13946913	0.10568016	0.15070951	0.11897513	0.119441965	1.00000000	0.01939772	0.01578577
AverageGreen	-0.11529640	-0.19834741	-0.18207461	0.02684452	0.020245600	0.01939772	1.00000000	0.83243368
AverageBlue	-0.01753745	-0.05136440	-0.04561751	0.06937904	0.065746989	0.01578577	0.83243368	1.00000000
VC	0.16495630	0.07032584	0.12282600	0.14082334	0.151073701	0.07193092	0.02237036	0.10020325
FEN	-0.06522750	0.03745239	-0.05141263	-0.09362812	-0.067299367	0.28372081	-0.09881974	-0.07589244
COL	0.23180475	0.30286027	0.37151758	0.09974995	0.119754431	0.23300908	-0.06964926	0.04997118
AP	-0.42152720	-0.40811592	-0.47733316	-0.38222827	-0.375920940	-0.06793352	-0.07785205	-0.12272857
DaysToFruiting	-0.49918051	-0.48023860	-0.44077430	-0.40411715	-0.427180090	-0.21534581	0.26587193	0.35197808
FruitLength	0.80410224	0.64127127	0.63009731	0.84218624	0.829043966	0.09544760	-0.10635562	-0.02157050
Fruitwidth	0.29178160	0.42760450	0.57730836	0.12769695	0.133354627	0.05706321	-0.32115167	-0.23355684
Fruitweight	0.69911255	0.82797148	0.69603724	0.59158960	0.599228622	0.01227511	-0.28935888	-0.20780472
NumberofLocules	0.09783039	0.09850930	0.20849834	-0.01912344	0.006103734	0.31114823	-0.12522335	-0.20034693
VC		FEN	COL	CAP	DaysToFruiting	FruitLength	Fruitwidth	Fruitweight
Perimeter	0.16495630	-0.06522750	0.231804752	-0.421527205	-0.49918051	0.80410224	0.29178160	0.69911255
Area	0.07032584	0.03745239	0.302860267	-0.408115920	-0.48023860	0.64127127	0.42760450	0.82797148
Maximumwidth	0.12282600	-0.05141263	0.371517575	-0.477333160	-0.44077430	0.63009731	0.57730836	0.69603724
MaximumHeight	0.14082334	-0.09362812	0.099749952	-0.382228267	-0.40411715	0.84218624	0.12769695	0.59158960

Cluste Jerarquico

La representación de la jerarquía de cluster se representa por medio de un “dendograma”, en el que las sucesivas fusiones de las ramas a los distintos niveles nos informan de las sucesivas fusiones de los grupos en grupos de superior nivel(mayor tamaño, menos homogeneidad) sucesivamente.

```

52 #instalar paquete una sola vez
53 #install.packages("Hmisc")
54 library("Hmisc")
55 res<- rcorr(as.matrix(data) , type ="pearson")
56 #muestra matriz de coeficientes de correlación
57 res$r
58 write.csv(res$r, "correlacion_pearson.csv")
59 #muestra matriz de p-values
60 res$p
61 write.csv(res$p, "correlacion_pearson_pvalues.csv")
62
63 #Dendograma vertical básico
64 #clúster mediante método ecludean y dendogram ward.D2
65 dd <- dist(scale(data), method = "euclidean")
66 hc <- hclust(dd, method = "ward.D2")
67 plot(hc, hang = -1)
68 |
<
68:1 (Top Level)

```

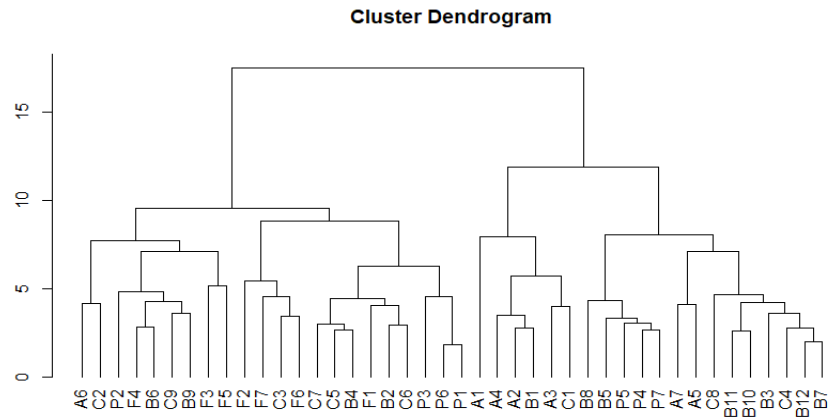
Console Terminal x

```

~/
> write.csv(res$p, "correlacion_pearson_pvalues.csv")
> #Dendograma vertical básico
> #clúster mediante método ecludean y dendogram ward.D2
> dd <- dist(scale(data), method = "euclidean")
> hc <- hclust(dd, method = "ward.D2")
> plot(hc, hang = -1)
> |

```

	Area
Maximumwidth	0.18515042
MaximumHeight	0.90432091
CurvedHeight	0.96939805
AverageRed	0.04488708
AverageGreen	0.43952348
AverageBlue	0.20331733
VC	0.42056675
FEN	0.43159122
COL	0.16983484
AP	0.34713561
DaysToFruiting	0.09631303
FruitLength	0.88736916
Fruitwidth	0.14985493
Fruitweight	0.17852472
NumberofLocules	NA

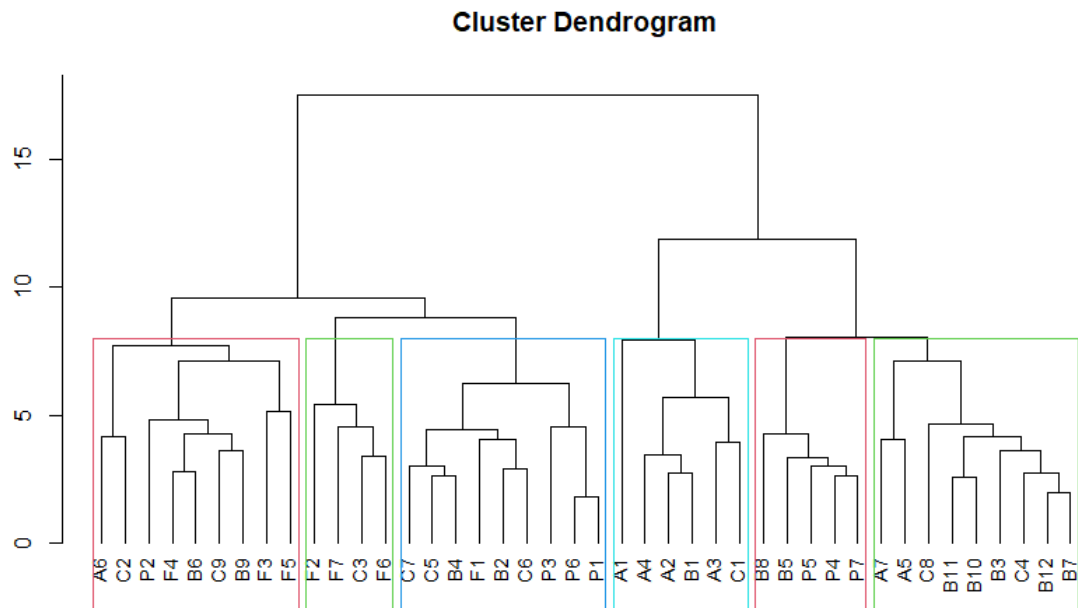


Cluster jerárquico vertical con grupos

```

69
70 #cluster jerárquico vertical con grupos
71 dd <- dist(scale(data), method = "euclidean")
72 hc <- hclust(dd, method = "ward.D2")
73 plot(hc, hang = -1, cex = 0.9)
74 9
75 res.hc <- hclust(dd, method = "ward.D2" )
76 grp <- cutree(res.hc, k = 6)
77 # Visualize plot(res.hc, cex = 0.6)
78 # plot tree
79 rect.hclust(res.hc, k = 6, border = 2:5) # add rectangle
80 |
<
0:1 (Top Level)
nsole Terminal x
ysToFruiting      0.09631303
uitLength         0.88736916
uitwidth          0.14985493
uitweight         0.17852472
mberoFlocules     NA
write.csv(res$P, "correlation_pearson_pvalues.csv")
#Dendograma vertical básico
#clúster mediante método euclidean y dendrogram ward.D2
dd <- dist(scale(data), method = "euclidean")
hc <- hclust(dd, method = "ward.D2")
plot(hc, hang = -1)
#cluster jerárquico vertical con grupos
dd <- dist(scale(data), method = "euclidean")
hc <- hclust(dd, method = "ward.D2")
plot(hc, hang = -1, cex = 0.9)
9
] 9
res.hc <- hclust(dd, method = "ward.D2" )
grp <- cutree(res.hc, k = 6)
# Visualize plot(res.hc, cex = 0.6)
# plot tree
rect.hclust(res.hc, k = 6, border = 2:5) # add rectangle

```

Número óptimo de clusters

La agrupación en clúster es una parte importante del proceso de Machine Learning para empresas comerciales o científicas que utilizan la Ciencia de Datos. Como su nombre lo indica, ayuda a identificar congregaciones de puntos de datos estrechamente relacionados, por alguna medida de distancia, en un conjunto de datos, los cuales, de otra manera, serían difíciles de entender.

Sin embargo, en la mayoría de los casos, el proceso de agrupación cae dentro del ámbito de Aprendizaje no Supervisado. Acá no hay respuestas o etiquetas conocidas para guiar el proceso de optimización o para medir nuestro éxito. Estamos en el territorio inexplorado.

Por lo tanto, no es de extrañar que un método tan popular como la agrupación K Means no parezca proporcionar una respuesta completamente satisfactoria cuando nos hacemos la pregunta básica ¿cómo sabríamos el número real de grupos para empezar?

Esta cuestión es de importancia crítica debido al hecho de que el proceso de agrupamiento es a menudo un precursor del procesamiento posterior de los datos individuales de las agrupaciones y, por lo tanto, la cantidad de recursos computacionales puede depender de esta medición.

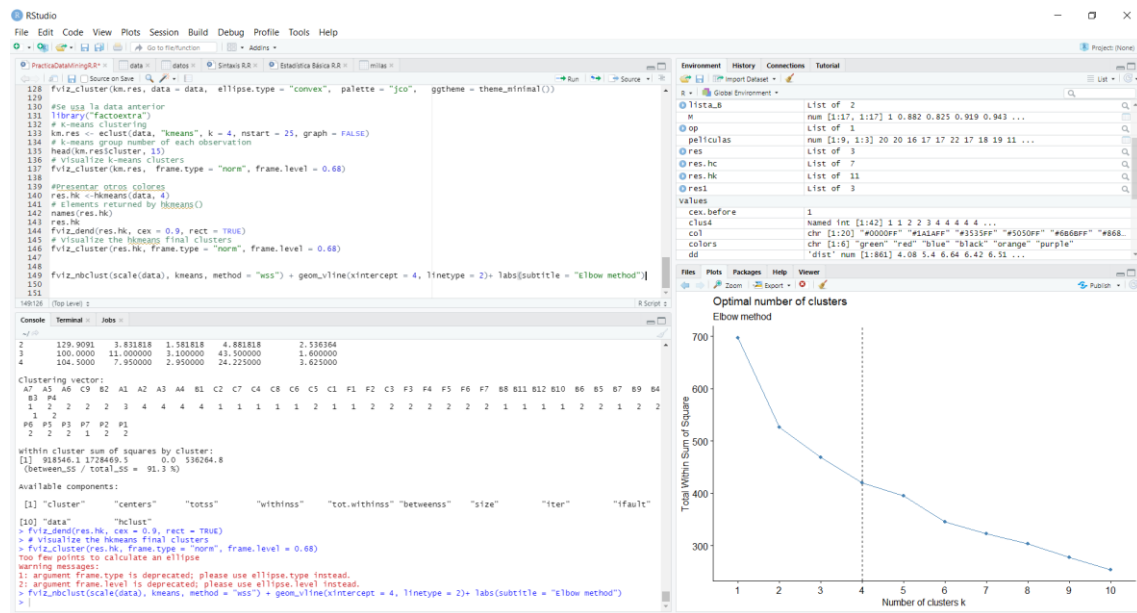
Hay múltiples métodos que puedes utilizar para determinar cuál es el número óptimo de clústeres para tus datos:

- Método del codo
- Método de la Silueta
- Método de estadística de la brecha

Elbow method

Este método utiliza los valores de la inercia obtenidos tras aplicar el K-means a diferente número de Clusters (desde 1 a N Clusters), siendo la inercia la suma de las distancias al cuadrado de cada objeto del Cluster a su centroide:

```
fviz_nbclust(scale(data), kmeans, method = "wss") + geom_vline(xintercept = 4,  
linetype = 2)+ labs(subtitle = "Elbow method")
```



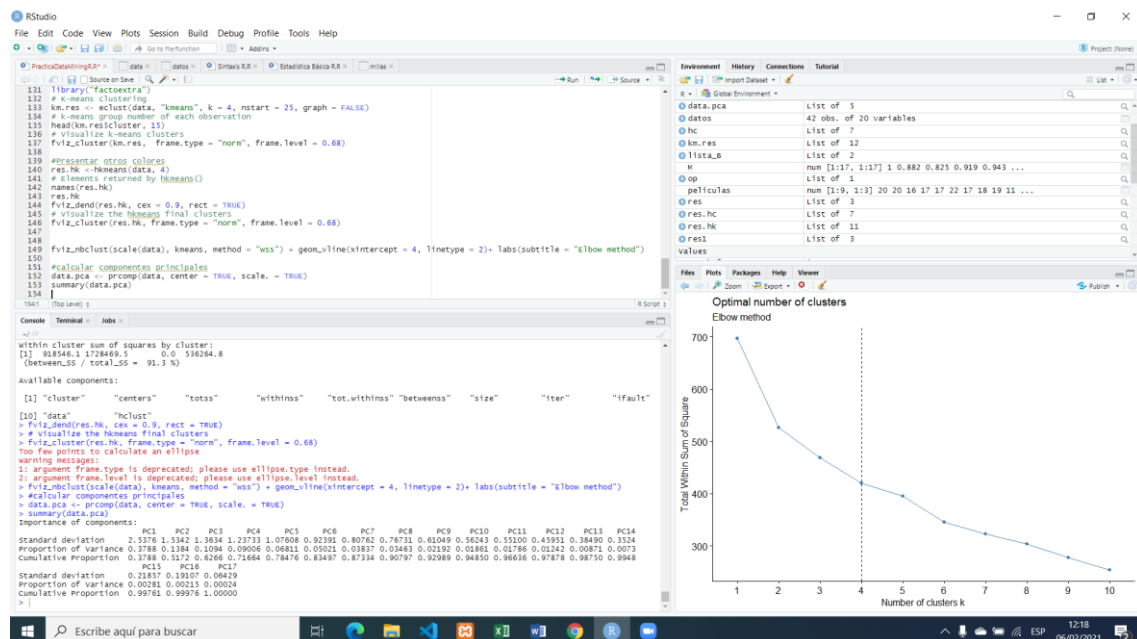
Análisis de componentes

El análisis de componentes principales (PCA) es una técnica útil para el análisis de datos exploratorios, que le permite visualizar mejor la variación presente en un conjunto de datos con muchas variables. Es particularmente útil en el caso de conjuntos de datos "amplios", donde tiene muchas variables para cada muestra.

#calcular componentes principales

```
data.pca <- prcomp(data, center = TRUE, scale. = TRUE)
```

```
summary(data.pca)
```



#instalar una sola vez la siguiente libreria

```
install_github("vqv/ggbiplot")
```

```
library(devtools)
```

El objetivo devtoolses facilitarle la vida como desarrollador de paquetes proporcionando funciones R que simplifican muchas tareas comunes

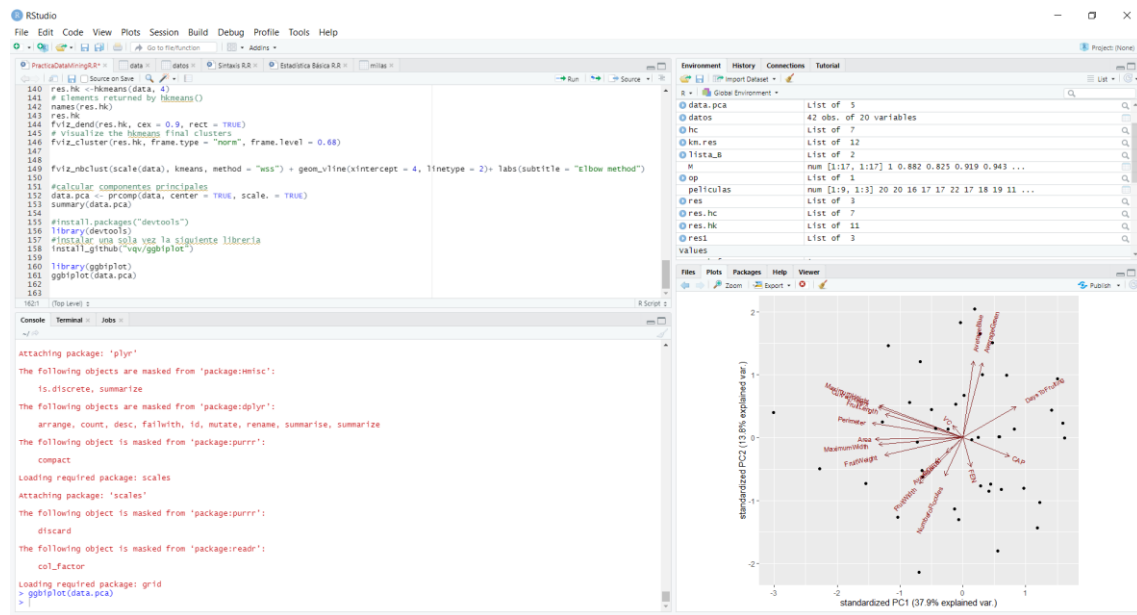
#instalar una sola vez la siguiente libreria

```
#install_github("vqv/ggbiplot")
```

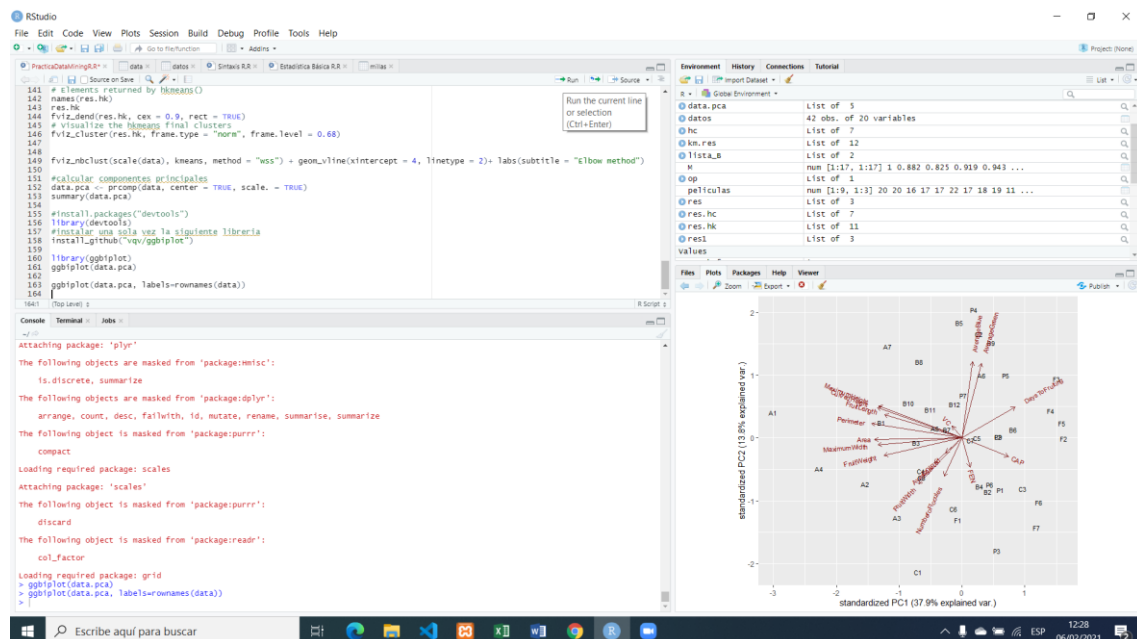
```
library(ggbiplot)
```

es un paquete de visualización de datos para el lenguaje R que implementa lo que se conoce como la "Gramática de los Gráficos"

ggbiplot(data.pca)

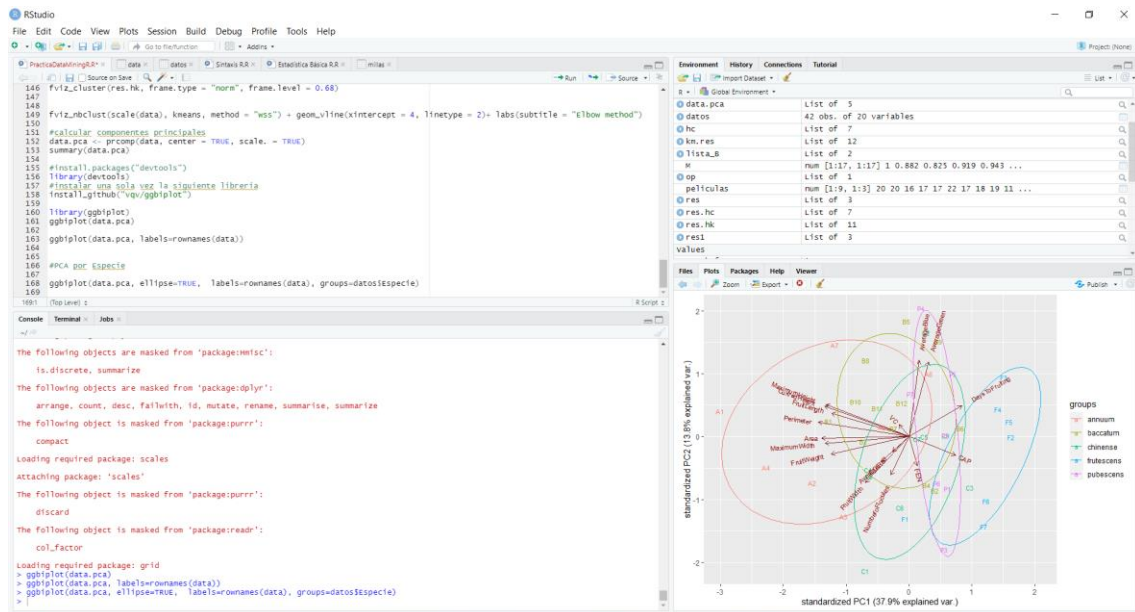


ggbiplot(data.pca, labels=rownames(data))



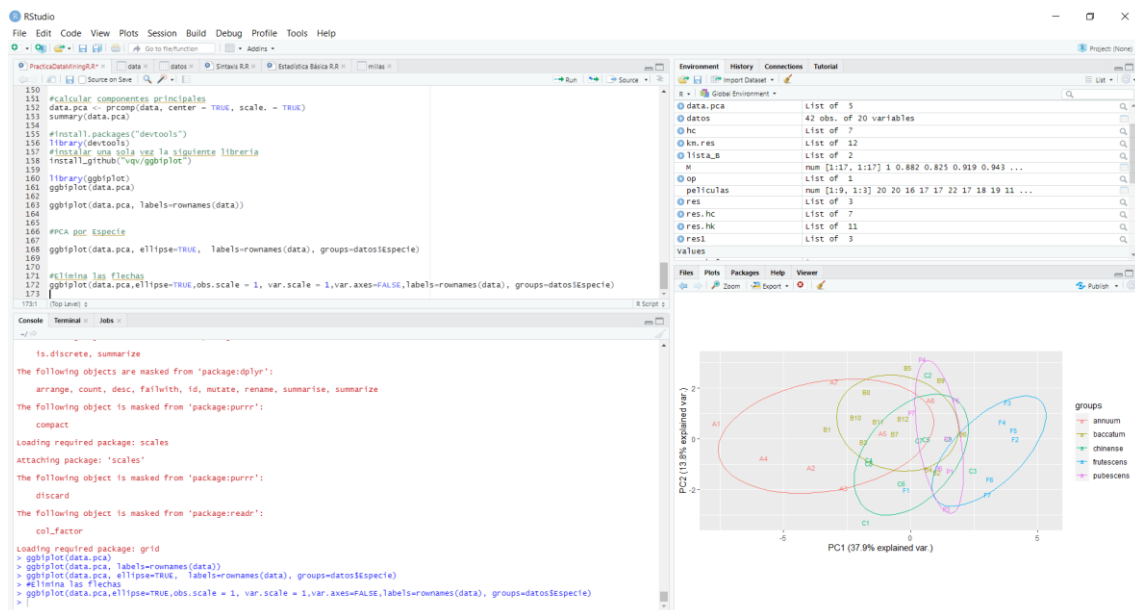
#PCA por Especie

ggbiplot(data.pca, ellipse=TRUE, labels=rownames(data), groups=datos\$Especie)



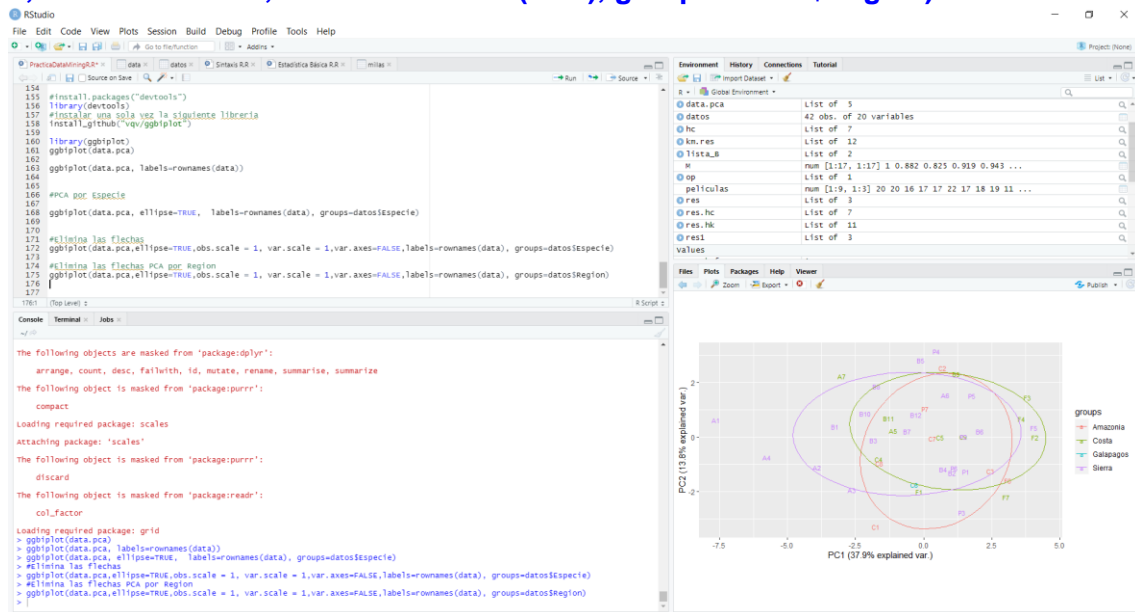
#Elimina las flechas

ggbiplot(data.pca,ellipse=TRUE,obs.scale = 1, var.scale = 1,var.axes=FALSE,labels=rownames(data), groups=datos\$Especie)



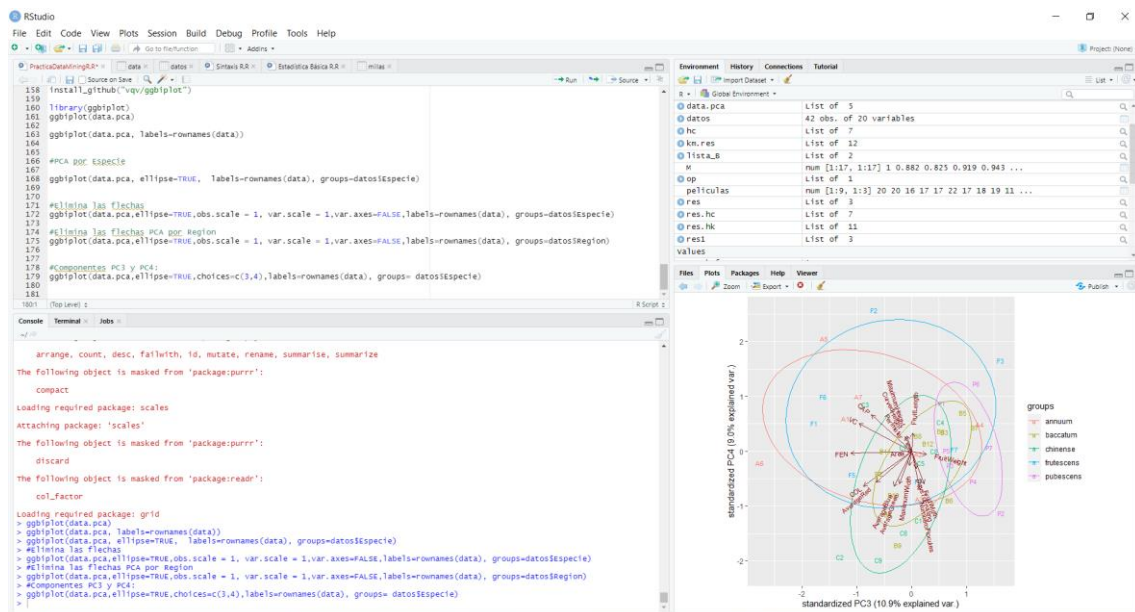
#Elimina las flechas PCA por Region

ggbiplot(data.pca,ellipse=TRUE,obs.scale = 1, var.scale = 1,var.axes=FALSE,labels=rownames(data), groups=datos\$Region)



#Componentes PC3 y PC4:

ggbiplot(data.pca,ellipse=TRUE,choices=c(3,4),labels=rownames(data), groups=datos\$Especie)



Técnicas Predictivas

Arboles de Decisión. Los árboles de decisión son un método usado en distintas disciplinas como modelo de predicción. Estos son similares a diagramas de flujo, en los que llegamos a puntos en los que se toman decisiones de acuerdo a una regla.

#librerías, instalar alguna si es necesario

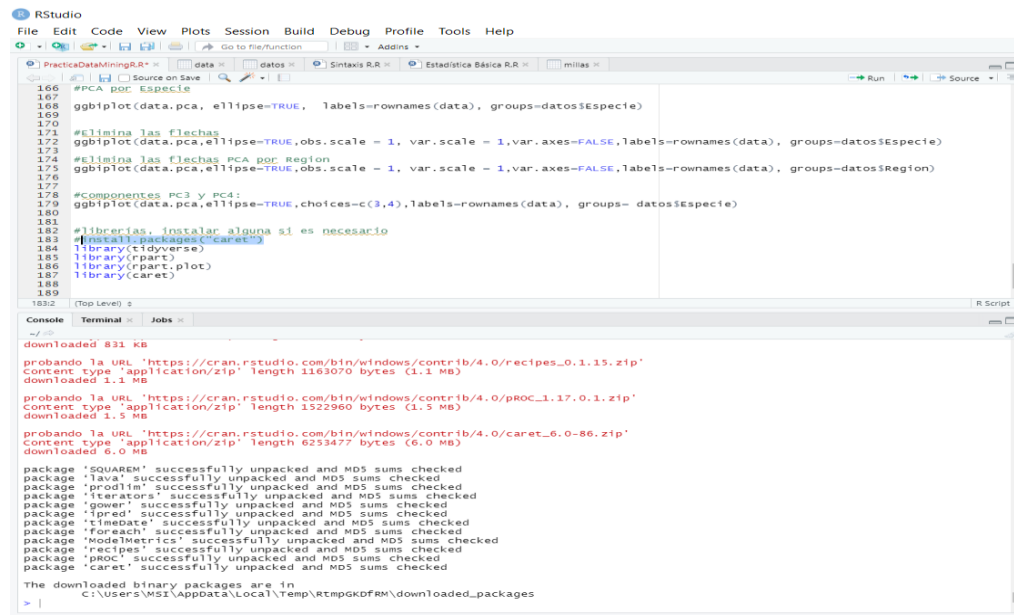
library(tidyverse)

library(rpart)

library(rpart.plot)

library(caret)

Instalamos librerías faltantes como se muestra a continuación



```
#PCA por Especie
166 #PCA por Especie
167
168 ggbiplot(data.pca, ellipse=TRUE, labels=rownames(data), groups=datos$Especie)
169
170 #Elimina las flechas
171 ggbiplot(data.pca, ellipse=TRUE, obs.scale = 1, var.scale = 1, var.axes=FALSE, labels=rownames(data), groups=datos$Especie)
172
173 #Elimina las flechas PCA por Region
174 ggbiplot(data.pca, ellipse=TRUE, obs.scale = 1, var.scale = 1, var.axes=FALSE, labels=rownames(data), groups=datos$Region)
175
176 #Componentes PC3 y PC4:
177 ggbiplot(data.pca, ellipse=TRUE, choices=c(3,4), labels=rownames(data), groups= datos$Especie)
178
179 #Instalar paquetes que faltan
180
181 #librerías, instalar alguna si es necesario
182 library(tidyverse)
183 library(rpart)
184 library(rpart.plot)
185 library(caret)
186
187
188
189
```

downloaded 831 KB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/recipes_0.1.15.zip'
Content type 'application/zip' length 1163070 bytes (1.1 MB)
downloaded 1.1 MB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/prodlim_1.17.0.1.zip'
Content type 'application/zip' length 1522960 bytes (1.5 MB)
downloaded 1.5 MB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/caret_6.0-86.zip'
Content type 'application/zip' length 6253477 bytes (6.0 MB)
downloaded 6.0 MB

package 'lava' successfully unpacked and MD5 sums checked
package 'prodlim' successfully unpacked and MD5 sums checked
package 'iterators' successfully unpacked and MD5 sums checked
package 'gower' successfully unpacked and MD5 sums checked
package 'lme4' successfully unpacked and MD5 sums checked
package 'timeDate' successfully unpacked and MD5 sums checked
package 'foreach' successfully unpacked and MD5 sums checked
package 'ModelMetrics' successfully unpacked and MD5 sums checked
package 'recipes' successfully unpacked and MD5 sums checked
package 'rpart' successfully unpacked and MD5 sums checked
package 'caret' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
c:\Users\MSI\AppData\Local\Temp\RtmpGKDFRM\downloaded_packages

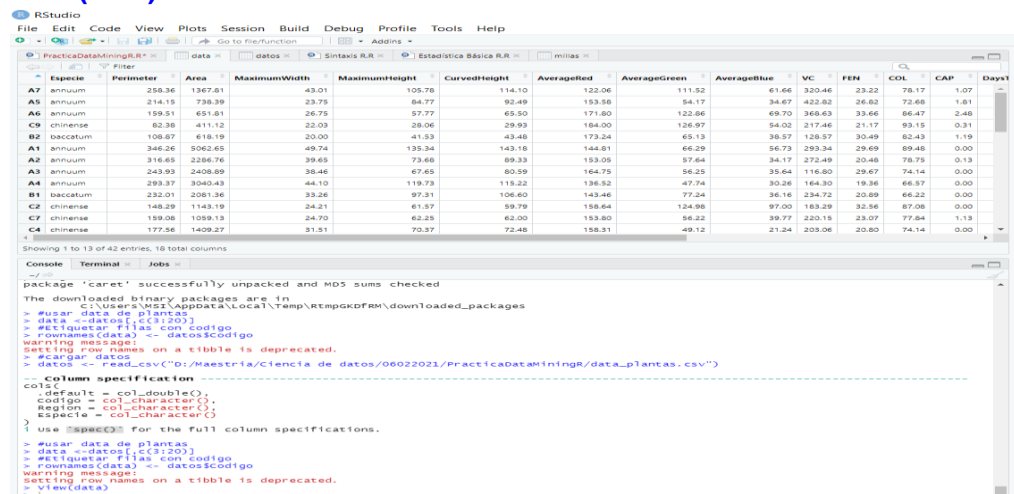
#usar data de plantas

data <- datos[,c(3:20)]

#Etiquetar filas con código

rownames(data) <- datos\$Codigo

View(data)



```
package 'caret' successfully unpacked and MD5 sums checked
The downloaded binary packages are in
c:\Users\MSI\AppData\Local\Temp\RtmpGKDFRM\downloaded_packages
> #usar data de plantas
> data <- datos[,c(3:20)]
> #Etiquetar filas con código
> rownames(data) <- datos$Codigo
warning message:
setting row names on a tibble is deprecated.
> data
> read.csv("D:\maestria\Ciencia de datos\06022021\PracticaDataMiningR\data_plantas.csv")
# Column specification -----
#> # A tibble: 16 x 16
#>   Especie Perimeter Area MaximumWidth MaximumHeight CurvedHeight Averagelength AveragelGreen Averagelhue VC FEN COL CAP Days1
#>   <fct>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 A7 annuum 258.36 1367.81 43.01 105.76 114.10 122.06 111.52 61.66 320.46 23.22 76.17 1.07
#> 2 A8 annuum 214.15 736.39 23.75 84.77 92.49 153.58 54.17 34.67 422.82 26.82 72.68 1.81
#> 3 A9 annuum 199.51 651.81 26.75 57.77 69.50 171.80 122.86 69.70 366.63 33.66 86.47 2.48
#> 4 C9 chinense 62.38 411.12 22.03 28.06 29.93 164.00 126.97 54.02 217.46 21.17 93.15 0.31
#> 5 B2 baccatum 108.67 618.19 20.00 41.53 43.48 173.24 65.13 38.57 128.57 30.49 82.43 1.19
#> 6 A1 annuum 346.26 3062.65 49.74 135.34 143.18 144.81 66.29 56.79 294.34 29.69 89.48 0.00
#> 7 A2 annuum 316.65 2284.76 39.45 73.68 89.33 153.05 87.64 34.17 272.49 20.48 75.75 0.13
#> 8 A3 annuum 243.93 2408.89 38.46 67.65 80.59 164.75 56.25 35.64 116.80 29.67 74.14 0.00
#> 9 A4 annuum 289.37 3040.43 44.10 119.73 119.22 136.52 47.74 30.36 164.30 19.26 66.87 0.00
#> 10 B1 baccatum 232.01 2081.36 33.28 97.31 106.60 143.46 77.24 36.16 234.72 20.89 66.22 0.00
#> 11 C7 chinense 148.29 1143.19 24.21 61.57 59.79 158.64 124.98 97.00 183.29 32.56 87.08 0.00
#> 12 C8 chinense 159.08 1059.13 24.70 62.25 62.00 153.80 56.22 39.77 220.15 23.07 77.84 1.13
#> 13 C4 chinense 177.56 1409.27 31.51 70.37 72.48 158.31 49.12 21.24 205.06 20.80 74.14 0.00
#>
#> Showing 1 to 13 of 42 entries. 16 total columns
```

```
set.seed(1649)
```

```
train <- sample_frac(data, .7)
```

```
test <- setdiff(data, train)
```

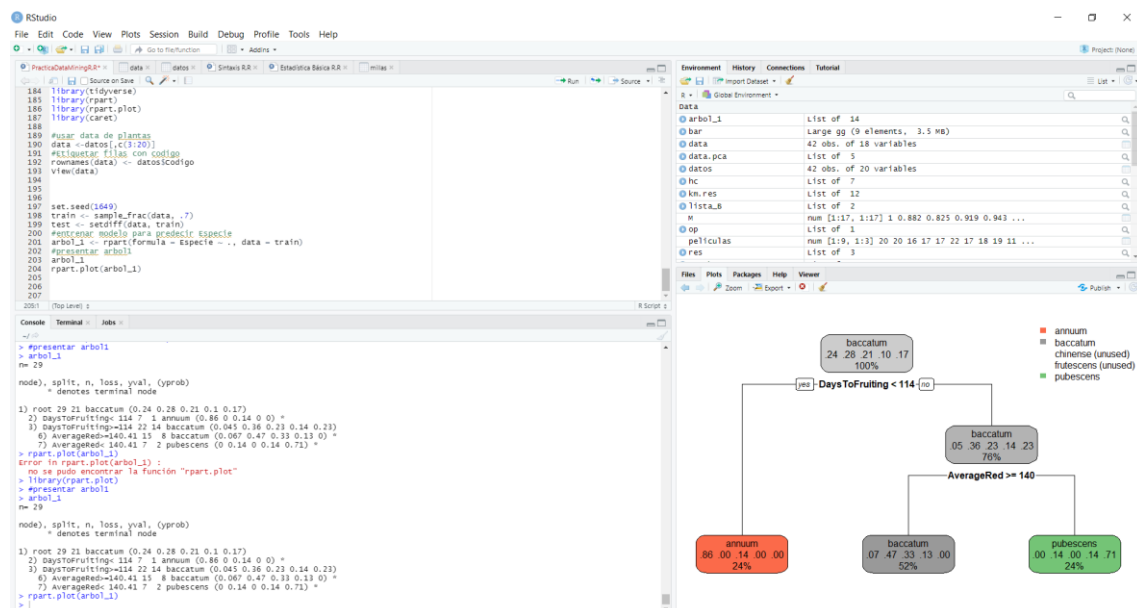
```
#entrenar modelo para predecir Especie
```

```
arbol_1 <- rpart(formula = Especie ~ ., data = train)
```

```
#presentar arbol1
```

```
arbol_1
```

```
rpart.plot(arbol_1)
```



```
#Generamos un segundo árbol, usando sets de entrenamiento y prueba diferentes.
```

```
set.seed(7439)
```

```
train2 <- sample_frac(data, .7)
```

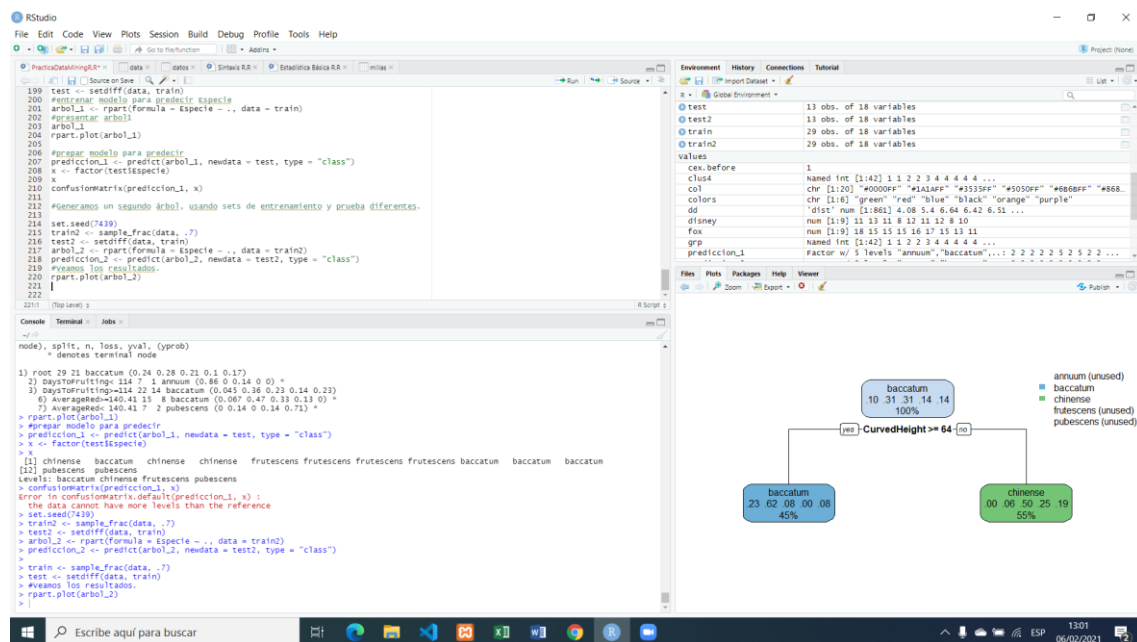
```
test2 <- setdiff(data, train)
```

```
arbol_2 <- rpart(formula = Especie ~ ., data = train2)
```

```
prediccion_2 <- predict(arbol_2, newdata = test2, type = "class")
```

```
#Veamos los resultados.
```

```
rpart.plot(arbol_2)
```

#Generamos un tercer árbol, usando sets de entrenamiento y prueba diferentes.

```
set.seed(8476)
```

```
train3 <- sample_frac(data, .7)
```

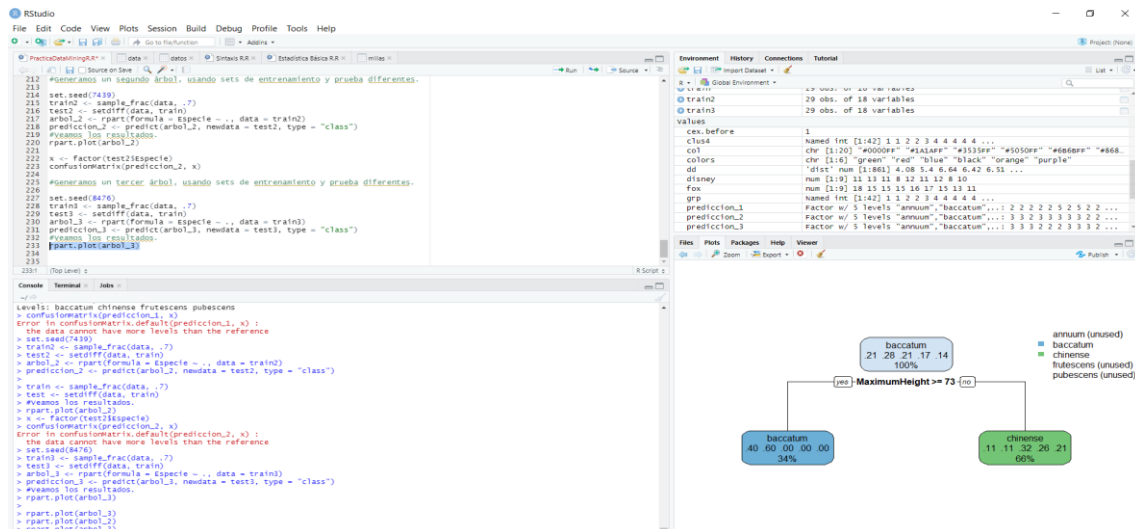
```
test3 <- setdiff(data, train)
```

```
arbol_3 <- rpart(formula = Especie ~ ., data = train3)
```

```
prediccion_3 <- predict(arbol_3, newdata = test3, type = "class")
```

#Veamos los resultados.

```
rpart.plot(arbol_3)
```



Regresión lineal simple

La regresión lineal simple consiste en generar un modelo de regresión (ecuación de una recta) que permita explicar la relación lineal que existe entre dos variables. A la variable dependiente o respuesta se le identifica como “Y” y a la variable predictora o independiente como “X”.

Ejemplo práctico en R

#Aplicamos la data de plantas, volver a cargar si es necesario

Elegir Variables CurvedHeight y MaximumHeight

```
> Variables <-c(7,8)
> Entrenamiento <-datos[, c(7,8)]
> Entrenamiento
# A tibble: 42 x 2
  MaximumHeight CurvedHeight
  <dbl> <dbl>
1 106. 114.
2 84.8 92.5
3 57.8 65.5
4 28.1 29.9
5 41.5 43.5
6 135. 143.
7 73.7 89.3
8 67.6 80.6
9 120. 115.
10 97.3 107.
# ... with 32 more rows
> |
```

MaximumHeight es la variable independiente y CurvedHeight la variable dependiente

```

Call:
lm(formula = CurvedHeight ~ MaximumHeight, data = Entrenamiento)

Coefficients:
(Intercept)  MaximumHeight
      2.280         1.022

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

```

`confint(modelo, level = 0.95)`

```

> confint(modelo, level = 0.95)
                2.5 %    97.5 %
(Intercept)  -0.5263995  5.085880
MaximumHeight  0.9783373  1.065493

```

#Ejemplo de predicción, asignar un valor a la variable independiente, MaximumHeight = 100

```

> predict(object = modelo, newdata = data.frame (MaximumHeight = c(100)),
+         interval = "confidence", level = 0.95)
      fit      lwr      upr
1 104.4712 102.1521 106.7904

```

#Ejemplo de predicción para varios datos

```

> Test <- data.frame(MaximumHeight = seq(100, 150, by=5))
> p<-predict(modelo, Test, interval = "prediction", level = 0.95)
> p<-data.frame (Test,p)
> p
  MaximumHeight      fit      lwr      upr
1          100 104.4712  95.48002 113.4625
2          105 109.5808 100.54122 118.6204
3          110 114.6904 105.59746 123.7833
4          115 119.8000 110.64882 128.9511
5          120 124.9096 115.69539 134.1237
6          125 130.0191 120.73728 139.3010
7          130 135.1287 125.77458 144.4828
8          135 140.2383 130.80740 149.6692
9          140 145.3479 135.83584 154.8599
10         145 150.4574 140.86002 160.0548
11         150 155.5670 145.88006 165.2539

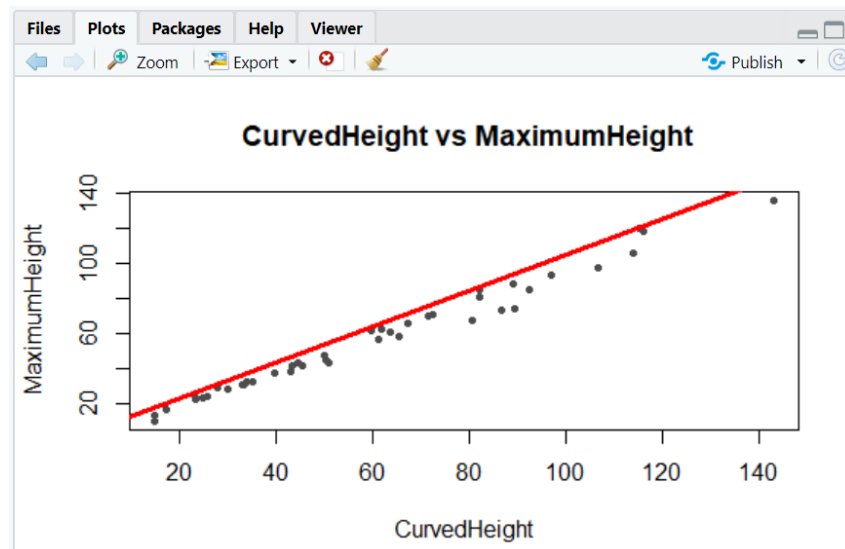
```

#Ejemplo de gráfico 1

```

plot(x = Entrenamiento$CurvedHeight, y = Entrenamiento$MaximumHeight, main = "
CurvedHeight vs MaximumHeight", xlab = "CurvedHeight", ylab = "MaximumHeight",
pch = 20, col = "grey30") abline(modelo, lwd = 3, col = "red")

```

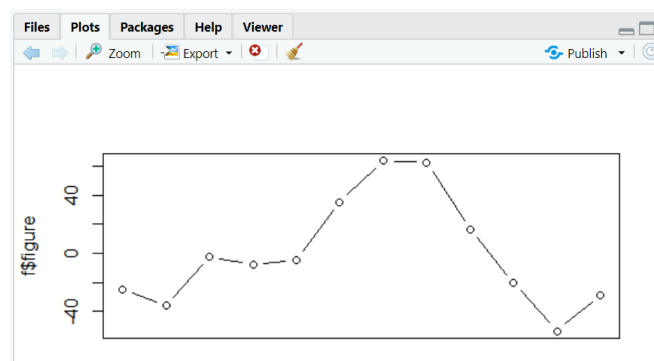


Series temporales

Una serie temporal se define como una colección de observaciones de una variable recogidas secuencialmente en el tiempo. Estas observaciones se suelen recoger en instantes de tiempo equiespaciados. Si los datos se recogen en instantes temporales de forma continua, se debe o bien digitalizar la serie, es decir, recoger sólo los valores en instantes de tiempo equiespaciados, o bien acumular los valores sobre intervalos de tiempo.

```
> f <- decompose(AirPassengers)
> # seasonal figures
> f$figure
[1] -24.748737 -36.188131 -2.241162 -8.036616 -4.506313 35.402778
[7] 63.830808 62.823232 16.520202 -20.642677 -53.593434 -28.619949
```

```
plot(f$figure, type="b", xaxt="n", xlab="")
```



```
# get names of 12 months in English words
```

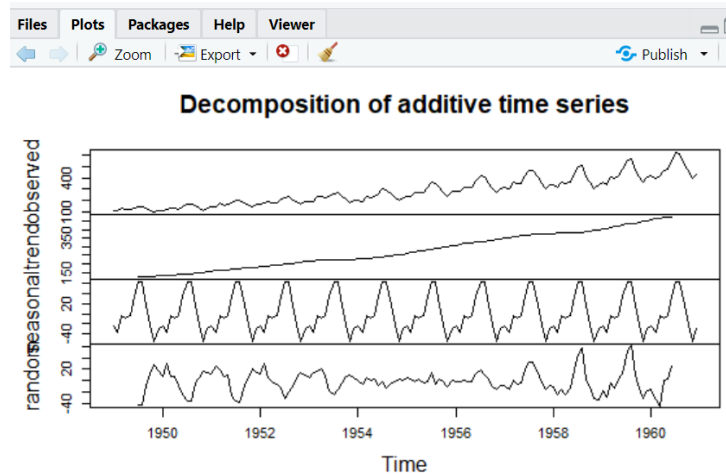
```
monthNames <- months(ISOdate(2011,1:12,1))
```

```
# label x-axis with month names
```

las is set to 2 for vertical label orientation

```
axis(1, at=1:12, labels=monthNames, las=2)
```

```
plot(f)
```



#Predicción

```
fit <- arima(AirPassengers, order=c(1,0,0), list(order=c(2,1,0), period=12))
```

```
fore <- predict(fit, n.ahead=24)
```

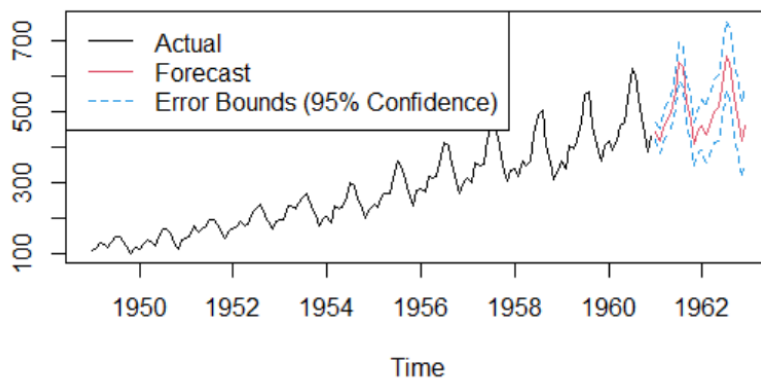
error bounds at 95% confidence level

```
U <- fore$pred + 2*fore$se
```

```
L <- fore$pred - 2*fore$se
```

```
ts.plot(AirPassengers, fore$pred, U, L, col=c(1,2,4,4), lty = c(1,1,2,2))
```

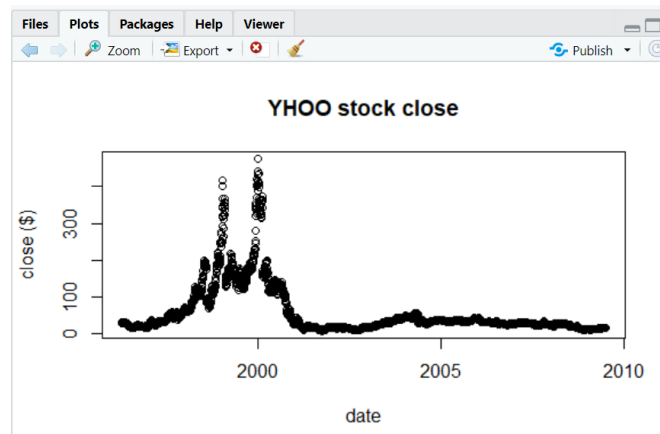
```
legend("topleft", c("Actual", "Forecast", "Error Bounds (95% Confidence)"), col =  
c(1,2,4), lty = c(1,1,2))
```



Ejemplos # 2 yahoo_series temporales.csv

#graficar la serie

```
plot(x=yahoo$date, y=yahoo$close, main='YHOO stock close', xlab='date',  
ylab='close ($)')
```

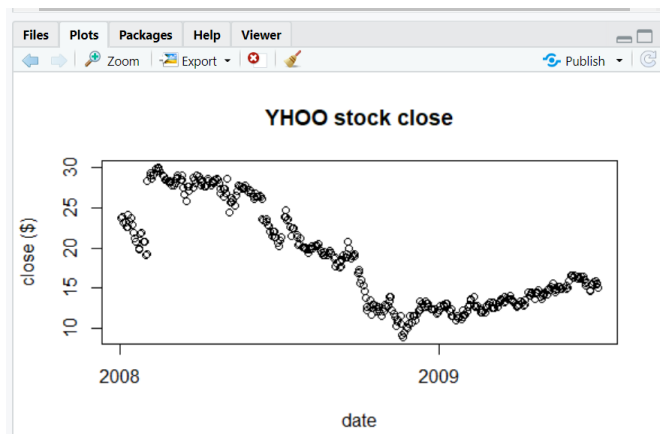


#filtrar datos y graficar la serie

```
yahoo2 <- yahoo[ yahoo$date >= as.Date('2008-01-01'), ]
```

```
plot(x=yahoo2$date, y=yahoo2$close,
```

```
main='YHOO stock close', xlab='date', ylab='close ($)')
```

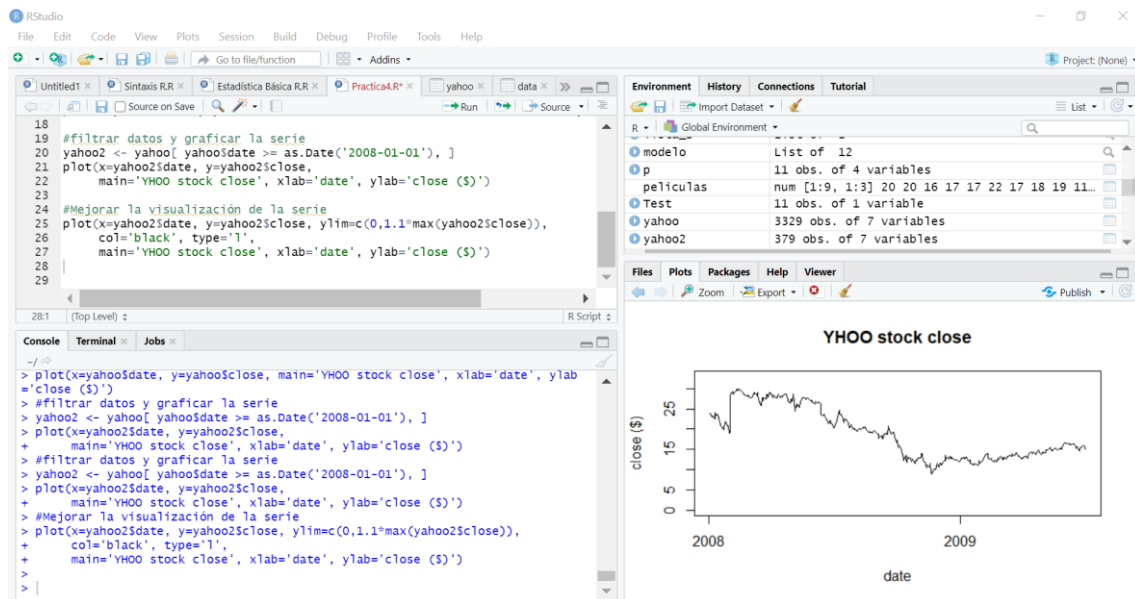


#Mejorar la visualización de la serie

```
plot(x=yahoo2$date, y=yahoo2$close, ylim=c(0,1.1*max(yahoo2$close)),
```

```
col='black', type='l',
```

```
main='YHOO stock close', xlab='date', ylab='close ($)')
```



Ejemplo de Predicciones #2

```

1 #Leer archivo de datos
2 library(readr)
3 yahoo <- read_csv("G:/BI/DataMiningR/yahoo_seriesTemporales.csv")
4 view(yahoo)
5 #ordenar por fecha
6 colnames(yahoo) <- tolower( colnames(yahoo) )
7 yahoo$date <- as.Date( as.character( yahoo$date ) )
8 # order yahoo into the same way we want to display it
9 yahoo <- yahoo[ order(yahoo$date), ]
10 #graficar la serie
11 plot(x=yahoo$date, y=yahoo$close, main='YHOO stock close', xlab='date', ylab='close ($)')
12

```

En la instrucción de la línea View(yahoo) – obtendremos este resultado.

	Date	Open	High	Low	Close	Volume	Adj Close
56	2009-04-15	13.93	14.09	13.77	14.02	12363200	14.02
57	2009-04-14	14.40	14.42	14.00	14.07	15151700	14.07
58	2009-04-13	14.02	14.54	13.86	14.42	35067600	14.42
59	2009-04-09	13.14	13.59	13.07	13.47	17285800	13.47
60	2009-04-08	12.90	13.01	12.75	12.92	11241000	12.92
61	2009-04-07	13.00	13.10	12.68	12.81	12290500	12.81
62	2009-04-06	13.08	13.24	12.99	13.23	11935700	13.23
63	2009-04-03	12.95	13.39	12.78	13.34	18534900	13.34
64	2009-04-02	13.04	13.14	12.80	12.95	28800900	12.95
65	2009-04-01	12.70	13.12	12.60	12.75	14540400	12.75
66	2009-03-31	12.76	13.10	12.67	12.81	11882100	12.81
67	2009-03-30	12.93	13.14	12.51	12.70	16556800	12.70
68	2009-03-27	13.17	13.61	13.12	13.18	22426200	13.18

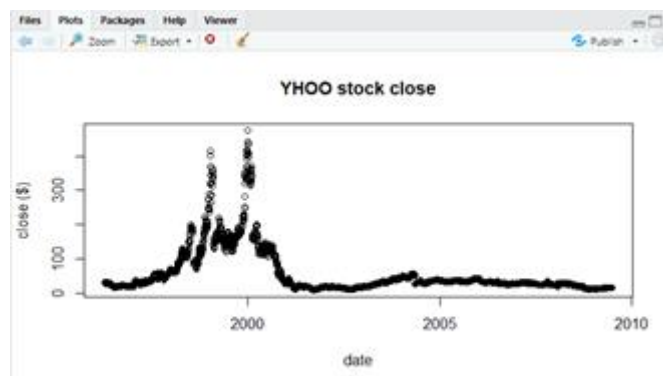
Showing 56 to 69 of 3,329 entries, 7 total columns

```

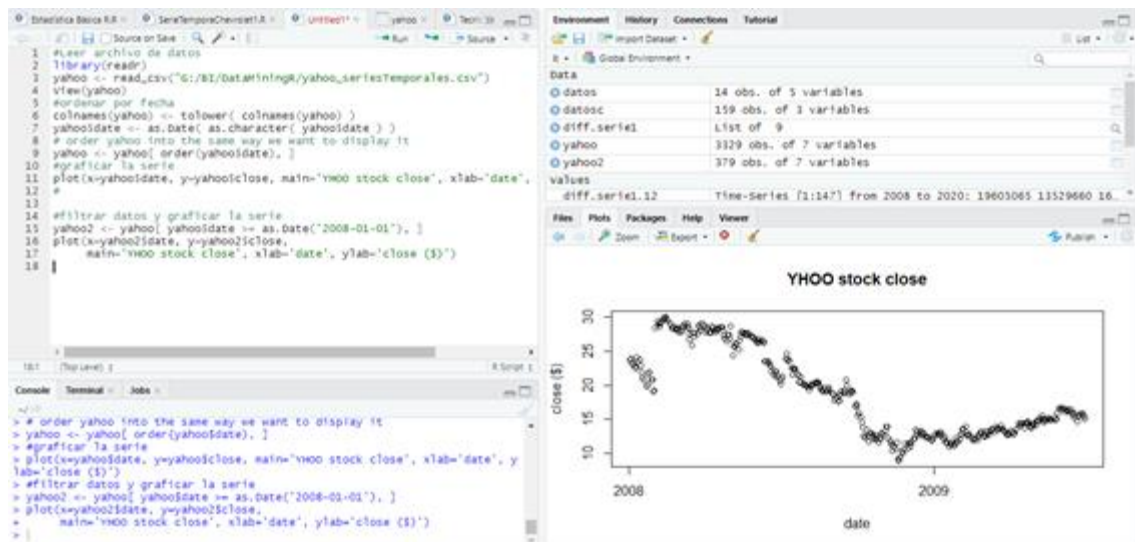
Open = col_double(),
High = col_double(),
Low = col_double(),
Close = col_double(),
Volume = col_double(),
'Adj close' = col_double()
)
> view(yahoo)
>

```

Al regresar a nuestra ventana de Código R continuar la ejecución – al ejecutar la línea `plot(x=yahoo$date, y=yahoo$close, main='YHOO stock close', xlab='date', ylab='close ($'))` nos mostrara lo siguiente:



Filtrar los datos y graficar la serie:



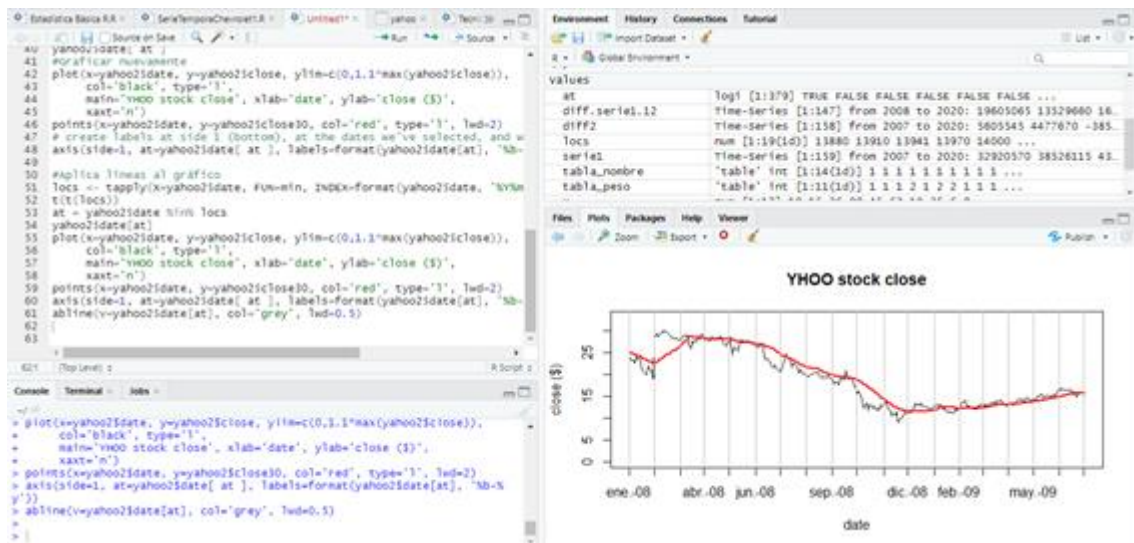
Mejorar la Visualización de la Serie



Graficar el promedio móvil.



Resultado final del Ejercicio #2

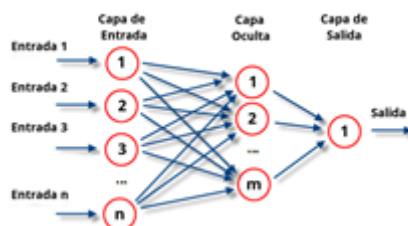


Redes Neuronales

Las redes neuronales artificiales son un modelo inspirado en el funcionamiento del cerebro humano. Está formado por un conjunto de nodos conocidos como neuronas artificiales que están conectadas y transmiten señales entre sí. Estas señales se transmiten desde la entrada hasta generar una salida.

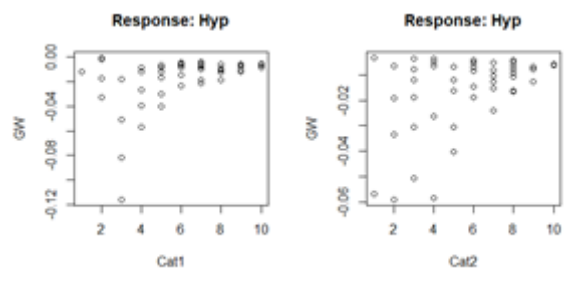
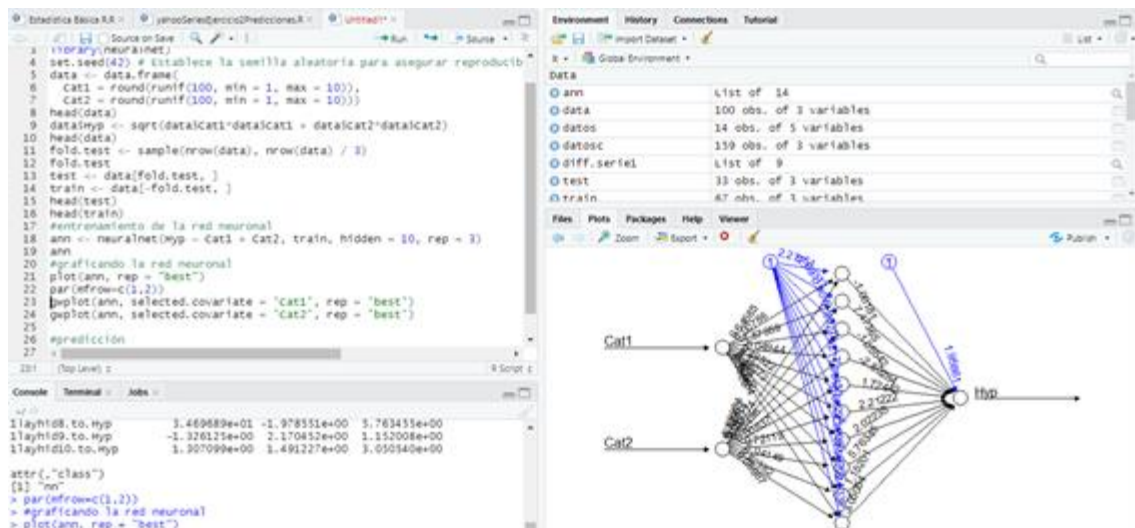
¿Cómo funcionan las redes neuronales?

Como se ha mencionado el funcionamiento de las redes se asemeja al del cerebro humano. Las redes reciben una serie de valores de entrada y cada una de estas entradas llega a un nodo llamado neurona. Las neuronas de la red están a su vez agrupadas en capas que forman la red neuronal. Cada una de las neuronas de la red posee a su vez un peso, un valor numérico, con el que modifica la entrada recibida. Los nuevos valores obtenidos salen de las neuronas y continúan su camino por la red. Este funcionamiento puede observarse de forma esquemática en la siguiente imagen.

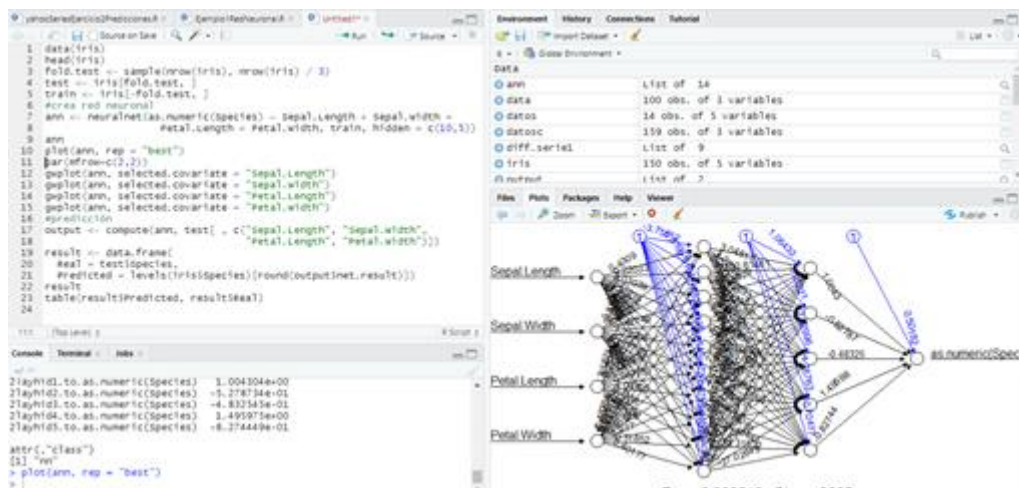


Una vez que se ha alcanzado el final de la red se obtiene una salida que será la predicción calculada por la red. Cuantas más capas posea la red y más compleja sea, también serán mas complejas las funciones que pueda realizar.

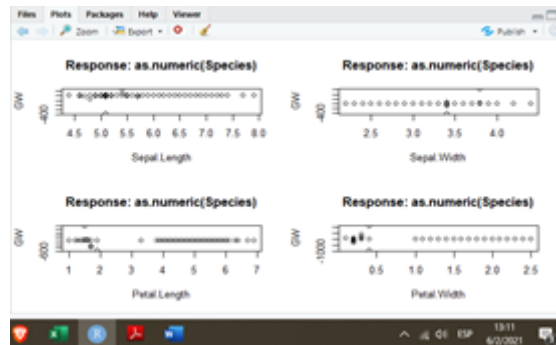
Ejemplo # 1 – Usando lenguaje R



Ejemplo#2 Red Neuronal.



Predicción:



Ejemplo #3 Red Neuronal:

