

Técnicas y Herramientas Modernas^{*}

Ricardo Palma¹[0000-1111-2222-3333], Gustavo Masera^{2,3}[1111-2222-3333-4444], and Rodolfo Echegaray³[2222-3333-4444-5555]

¹ Universidad Nacional de Cuyo CP 5502 Arg.

² Instituto de Ingeniería Industrial

rpalma@uncu.edu.ar

<http://themys.sid.uncu.edu.ar>

³ Laboratorio Industria 4.0, Facultad de Ingeniería, Mendoza

ricardo.rpalma@gmail.com

Abstract. Este módulo establece los primeros pasos de programación en R-Cran utilizando la plantilla LNCS de Springer.

Keywords: Programing Lenguaje · R-Cran · RMarkdown

1 Elementos básicos de programación

En este breve tutorial examinaremos algunos elementos del lenguaje de programación R y como valernos de ello para resolver problemas de la vida cotidiana. Apelaremos a ejemplos bien conocidos, pero además mostraremos las soluciones que desarrollaremos contra las misma que ya están implementadas en R. Comparando el costo computacional, medido como tiempo de ejecución. Esto nos permitirá entender la calidad del algoritmo que implementemos. Como excusa para introducirnos propondremos realizar tres experimentos y medir el tiempo ejecución.

Veremos:

- Generar un vector secuencia
- Implementación de una serie Fibonacci
- Ordenación de un vector por método burbuja
- La penitencia de Newton
- Algoritmo de funciones estadísticas (kmeans)

1.1 Algunas ideas de como medir el tiempo de ejecucion

Es muy frecuente la pregunta sobre **cuál algoritmo es el mejor** para realizar una tarea específica o resolver un problema. Una de las técnicas que suele utilizarse para esto es el benchmarking. No necesariamente un algoritmo es siempre mejor que otro. Por el contrario existen métricas para hacer el benchmarking que

^{*} Universidad Nacional de Cuyo

comparan el uso del procesador, la memoria requerida, el tiempo que tarda en resolver el problema, la exactitud de la solución, etc. El benchmarking no me dice cual es el mejor, pero me da información sobre cuál es el más conveniente según las características del problema y el recurso físico (máquina) que tengo.

Muchos de ustedes están familiarizados con Octave o Matlab. Algunos recordarán que para invertir matrices y saber que método era más eficiente su utilizaban los comandos tic y tac. Por ejemplo se generaba una matriz A, se ejecutaba el comando tic que disparaba una especie de cronómetro interno, luego se invertía siguiendo una algoritmo de determinante y finalmente se ejecutaba el tomando toc que detenía el reloj y entregaba el tiempo de ejecución. Luego se repetía el mismo procedimiento, pero en lugar de hacerlo con determitante se usaba un algoritmo de matriz LU.

Una búsqueda rápida en línea nos revela al menos tres paquetes R para comparar performance del código R (rbenchmark, microbenchmark y tictoc). Estos además de medir el tiempo nos indican porcentaje de memoria y microprocesador utilizados.

Además, la base R proporciona al menos dos métodos para medir el tiempo de ejecución del código R (Sys.time y system.time), que es una aproximación bastante útil para un curso como el que desarrollamos.

A continuación, paso brevemente por la sintaxis del uso de cada una de las cinco opciones, y presento mis conclusiones al final.

Usando Sys.time El tiempo de ejecución de un fragmento de código se puede medir tomando la diferencia entre el tiempo al inicio y al final del fragmento de código leyendo los registros del RTC (Real Time Clock. Simple pero flexible: como un relojito de arena :).

```
sleep_for_a_minute <- function() { Sys.sleep(14) }

start_time <- Sys.time()
sleep_for_a_minute()
end_time <- Sys.time()

end_time - start_time
```

```
## Time difference of 14.0184 secs
```

Hemos generado una función que antes no existía y la hemos usado. Deficiencias: Si usas el comando dentro de un documento en R-Studio te demorarás mucho tiempo cuando compiles un PDF o una presentación.

Biblioteca tictoc Esto de usar una biblioteca es llamar u cargar una procedimientos que generará comando nuevos en R. Como ya fue comentado, cargar una biblioteca implica ejecutar el comando install.packages() o usar en r-studio el menú de Herramientas y Luego Instalar paquetes. Las funciones tic y toc

son de la misma biblioteca de Octave/Matlab y se usan de la misma manera para la evaluación comparativa que el tiempo de sistema recién demostrado. Sin embargo, tictoc agrega mucha más comodidad al usuario y armonía al conjunto.

La versión de desarrollo más reciente de tictoc se puede instalar desde github:

```
install.packages (tictoc)
```

```
library(tictoc)

tic("sleeping")
A<-20
print("dormire una siestita...")
```

```
## [1] "dormire una siestita..."
```

```
Sys.sleep(2)
print("...suena el despertador")
```

```
## [1] "...suena el despertador"
```

```
toc()
```

```
## sleeping: 2.012 sec elapsed
```

Uno puede cronometrar solamente un fragmento de código a la vez:

Biblioteca rbenchmark La documentación de la función benchmark del paquete rbenchmark R lo describe como “un simple contenedor alrededor de system.time”. Sin embargo, agrega mucha conveniencia en comparación con las llamadas simples a system.time. Por ejemplo, requiere solo una llamada de referencia para cronometrar múltiples repeticiones de múltiples expresiones. Además, los resultados devueltos se organizan convenientemente en un marco de datos.

Recuerda antes de ejecutar

```
##library ( cualquiercosa )##
```

debes haber cargado en tu máquina la biblioteca que quieres invocar usando **install.packages (cualquiercosa)** .

```
«bench_mark,echo=TRUE»=
```

```
##           test replications elapsed relative user.self sys.self
## 3 linear system      1000    0.119     1.000     0.118    0.001
## 1                lm      1000    0.749     6.294     0.739    0.009
## 2 pseudoinverse     1000    0.148     1.244     0.137    0.012
```

En el informe de salida nos dice que cantidad de tiempo consume cada parte del código.

1.2 Biblioteca Microbenchmark

La versión de desarrollo más reciente de microbenchmark se puede instalar desde github:

Al igual que el punto de referencia del paquete rbenchmark, la función microbenchmark se puede usar para comparar tiempos de ejecución de múltiples fragmentos de código R. Pero ofrece una gran comodidad y funcionalidad adicional. Es más “beta” (inestable), pero como todo lo que hoy es nuevo poco a poco se hará más estable y no complicará tanto las cosas para el usuario final.

Una cosa interesante es que se puede ver la salida gráfica del uso de recursos. Ver líneas finales del código.

Me parece que una característica particularmente agradable de microbenchmark es la capacidad de verificar automáticamente los resultados de las expresiones de referencia con una función especificada por el usuario. Esto se demuestra a continuación, donde nuevamente comparamos tres métodos que computan el vector de coeficientes de un modelo lineal.

```
library(microbenchmark)

set.seed(2017)
n <- 10000
p <- 100
X <- matrix(rnorm(n*p), n, p)
y <- X %*% rnorm(p) + rnorm(100)

check_for_equal_coefs <- function(values) {
  tol <- 1e-12
  max_error <- max(c(abs(values[[1]] - values[[2]]),
                     abs(values[[2]] - values[[3]]),
                     abs(values[[1]] - values[[3]])))
  max_error < tol
}

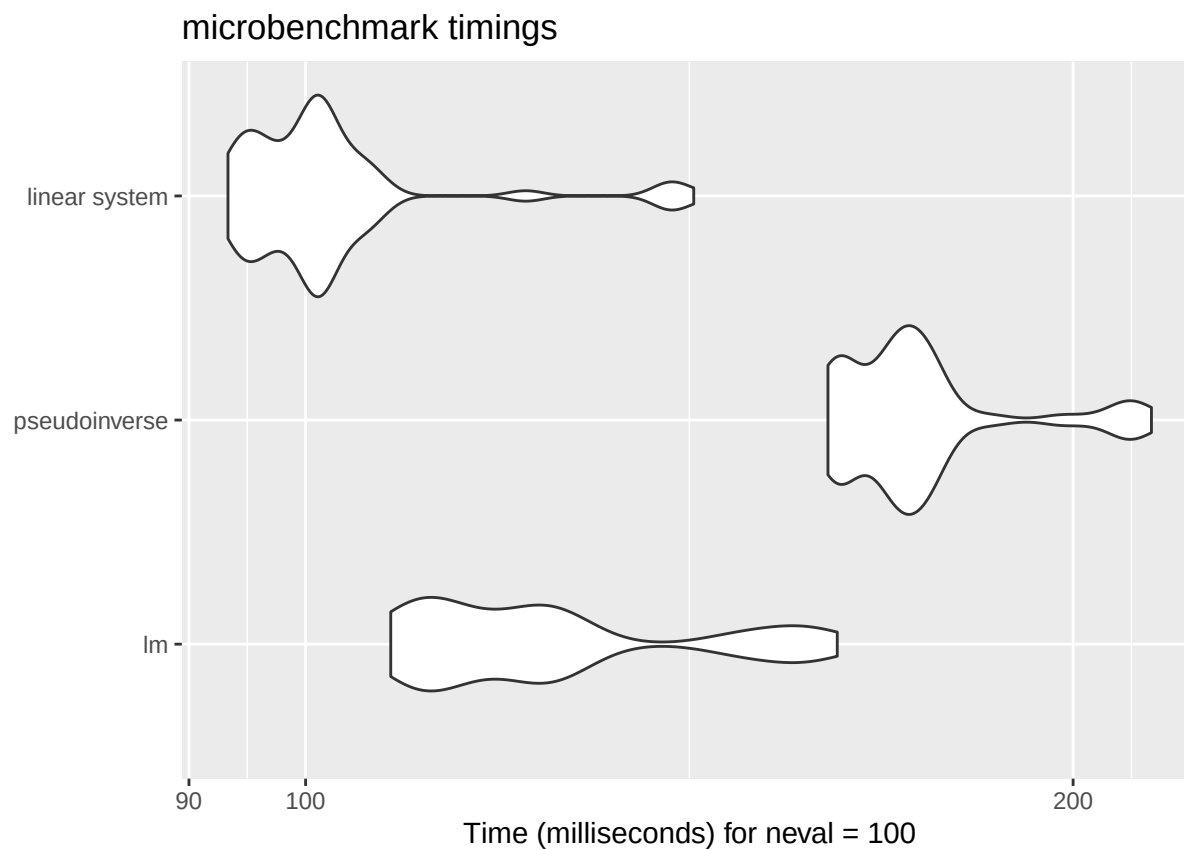
mbm <- microbenchmark("lm" = { b <- lm(y ~ X + 0)$coef },
  "pseudoinverse" = {
    b <- solve(t(X) %*% X) %*% t(X) %*% y
  },
  "linear system" = {
    b <- solve(t(X) %*% X, t(X) %*% y)
  },
  check = check_for_equal_coefs)

mbm

## Unit: milliseconds
##          expr          min          lq          mean         median          uq         max neval
```

```
##          lm 107.99171 111.90941 125.0261 121.9367 127.1246 161.6320 100
## pseudoinverse 160.27765 163.79209 174.1863 171.2880 175.3848 214.6600 100
## linear system 93.22768 96.34663 102.4206 100.5656 102.4196 141.9815 100
## cld
## a
## b
## c
```

```
library(ggplot2)
autoplot(mbm)
```



1.3 Consigna de trabajo

El trabajo de hoy que presentar implica revisar los algoritmos que se presentan a continuación. Deberá ejecutarlos primero en la línea de comando de la consola.

Luego deberá elegir alguno de los métodos vistos para medir la performance y comparar los resultados con otros compañeros que hayan usado otros métodos para medir la performance.

Luego todo deberá entregarse en un informe en formato pdf construido con RStudio, archivo RMarkdown.

1.4 Generar un vector secuencia

De echo R. tiene un comando para generar secuencias llamado “seq’’. Recomendamos ejecutar la ayuda del comando en RStudio.

Pero utilizaremos el clásico método de secuencias de anidamiento for, while, do , until.

Generaremos una secuencia de números que de dos en dos entre 1 y 100.000.

```
A <- 0
for (i in 1:50000) { A[i] <- (i*2)}
head (A)
```

Secuencias generada con for

```
## [1]  2  4  6  8 10 12
```

```
tail (A)
```

```
## [1] 99990 99992 99994 99996 99998 100000
```

Secuencia generada con R

```
## [1]  1  3  5  7  9 11
```

```
## [1] 999989 999991 999993 999995 999997 999999
```

CONSIGNA: Comparar la performance con systemtime

1.5 Implementación de una serie Fibonacci o Fibonacci

En matemáticas, la sucesión o serie de Fibonacci es la siguiente sucesión infinita de números naturales:

0,1,1,2,3,5,8 ... 89,144,233 ...

La sucesión comienza con los números 0 y 1,2 a partir de estos, «cada término es la suma de los dos anteriores», es la relación de recurrencia que la define.

A los elementos de esta sucesión se les llama números de Fibonacci. Esta sucesión fue descrita en Europa por Leonardo de Pisa, matemático italiano del siglo XIII también conocido como Fibonacci. Tiene numerosas aplicaciones en ciencias de la computación, matemática y teoría de juegos. También aparece

en configuraciones biológicas, como por ejemplo en las ramas de los árboles, en la disposición de las hojas en el tallo, en las flores de alcachofas y girasoles, en las inflorescencias del brécol romanesco, en la configuración de las piñas de las coníferas, en la reproducción de los conejos y en como el ADN codifica el crecimiento de formas orgánicas complejas. De igual manera, se encuentra en la estructura espiral del caparazón de algunos moluscos, como el nautilus.

Original de la Biblioteca Uninersidad de Florencia. Liber Abachi - Autor Fibonacci

1.6 Definición matemática recurrente

$$f_0 = 0 \quad (1)$$

$$f_1 = 1 \quad (2)$$

$$f_{n+1} = f_n + f_{n-1} \quad (3)$$

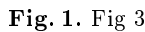
Ejemplo inicial del algoritmo

```
## [1] 0 1 1 2
## [1] 4
## [1] 0 1 1 2 3
## [1] 5
## [1] 0 1 1 2 3 5
## [1] 6
## [1] 0 1 1 2 3 5 8
## [1] 7
```

CONSIGNA: ¿Cuántas iteraciones se necesitan para generar un número de la serie mayor que 1.000.000 ?

1.7 Ordenación de un vector por método burbuja

La Ordenación de burbuja (**Bubble Sort en inglés**) es un sencillo algoritmo de ordenamiento. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada. Este algoritmo obtiene su nombre de la forma con la que suben por la lista los elementos durante los intercambios, como si fueran pequeñas *burbujas*. También es conocido como el método del intercambio directo. Dado que solo usa comparaciones para operar elementos, se lo considera un algoritmo de comparación, siendo uno de los más sencillos de implementada.



```

# Tomo una muestra de 10 números ente 1 y 100
x<-sample(1:100,10)
# Creo una funci?n para ordenar
burbuja <- function(x){
  n<-length(x)
  for(j in 1:(n-1)){
    for(i in 1:(n-j)){
      if(x[i]>x[i+1]){
        temp<-x[i]
        x[i]<-x[i+1]
        x[i+1]<-temp
      }
    }
  }
  return(x)
}
res<-burbuja(x)
#Muestra obtenida
x

```

```
## [1] 7 71 10 72 37 28 64 82 19 88
```

```

#Muestra Ordenada
res

```

```
## [1] 7 10 19 28 37 64 71 72 82 88
```

```

#Ordanaci?n con el coamando SORT de R-Cran
sort(x)

```

```
## [1] 7 10 19 28 37 64 71 72 82 88
```

CONSIGNA: Compara la performance de ordenación del método burbuja vs el método sort de R

Usar método microbenchmark para una muestra de tamaño 20.000

1.8 La penitencia de Newton

Comentan algunos historiadores que Newton tenía en le escuela primaria un profesor muy exigente que reclamaba constantemente la atención de los alumnos a lo que el estaba enseñando. Tan pronto alguien cometía un acto “improcedente” lo castigaba con una penitencia que consistía en traer para el día siguiente el resultado de la suma de desde el número 1 hasta cierto número aleatoriamente

elegido. Por supuesto esta tarea demandaría quedarse toda la noche haciendo las sumas parciales, tarea que el profesor ya había realizado previamente y que tenía registrado en un voluminoso cuaderno que (al esltio tabla de logaritmos) cargaba religiosamente en su portafolios de cuero.

Consigna de trabajo : Desarrolla dos algoritmos que hagan este trabajo , por ejemplo para sumar desde 1 hasta $1 * 10^{10}$ y verifica cuál de los dos es más eficiente.

2 Agrupación K-Means

En el enfoque de partición de R, las observaciones se dividen en grupos K y se reorganizan para formar los grupos más cohesivos posibles de acuerdo con un criterio dado. Hay dos métodos: K-medias y partición alrededor de mediodes *PAM*. En este artículo, basado en el capítulo 16 de *R in Action*, Second Edition, el autor Rob Kabacoff analiza la agrupación de K-means.

El algoritmo k-means podría utilizarse en el contexto del mantenimiento industrial.

Supongamos que tienes un conjunto de datos históricos sobre el rendimiento y la vida útil de diferentes componentes de maquinaria en una planta industrial. Estos datos pueden incluir variables como la temperatura de funcionamiento, la vibración, el consumo de energía, la carga de trabajo, entre otros.

El objetivo sería utilizar el algoritmo k-means para identificar grupos o clusters de componentes que tengan un comportamiento similar en términos de su rendimiento y vida útil. Esto puede ayudar a los ingenieros y operarios de mantenimiento a entender mejor el comportamiento de los componentes y tomar decisiones más informadas sobre el mantenimiento preventivo o la sustitución de los mismos.

El proceso podría seguir los siguientes pasos:

- Preparación de datos: Reunir y preparar los datos históricos relevantes de los componentes de maquinaria. Esto puede incluir la recopilación de variables, como la temperatura, la vibración, el consumo de energía, etc., para cada componente en diferentes momentos.
- Selección de características: Identificar las características relevantes para el análisis. Puedes realizar un análisis de correlación o utilizar el conocimiento experto para seleccionar las variables más significativas que pueden influir en el rendimiento y la vida útil de los componentes.
- Normalización de datos: Normalizar los datos para asegurarse de que todas las características tengan la misma escala y rango. Esto es importante para que todas las variables tengan una influencia equitativa en el algoritmo k-means.
- Elección del número de clusters (k): Determinar el número de clusters que deseas encontrar. Puedes utilizar métodos como el método del codo (elbow method) o la silueta (silhouette) para evaluar diferentes valores de k y seleccionar el más adecuado.

- Aplicación del algoritmo k-means: Ejecutar el algoritmo k-means en los datos normalizados, agrupando los componentes en los k clusters determinados. El algoritmo asignará cada componente al cluster más cercano en función de las características seleccionadas.
- Análisis e interpretación de los resultados: Analizar los resultados del clustering para identificar patrones y comportamientos similares entre los componentes. Puedes examinar las características de cada cluster para entender qué factores pueden estar relacionados con un mayor rendimiento o una vida útil más larga. Esto puede ayudar a tomar decisiones de mantenimiento más precisas y eficientes.

Atención que este es solo un ejemplo de cómo se podría aplicar el algoritmo k-means en el mantenimiento industrial. La selección de variables, la normalización de datos y la interpretación de los resultados pueden variar según el contexto y los objetivos específicos del estudio de mantenimiento. No es lo mismo utilizar esta técnica para mantenimiento en la industria alimenticia que en la industria nuclear. Cada campo de aplicación debería respetar los procedimientos sugeridos por las normas y buenas prácticas recomendadas.

2.1 Bibliografía recomendada

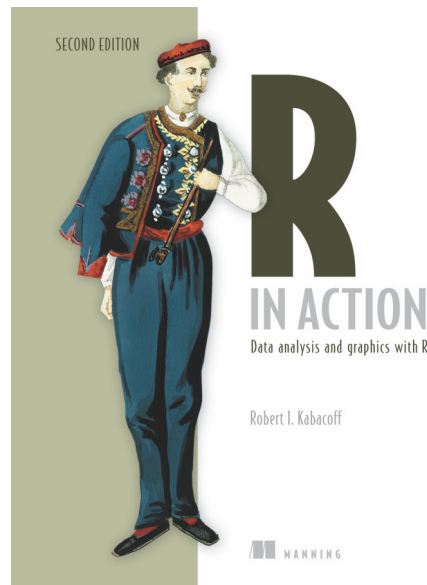


Fig. 2. R In Action por Robert Kabacoff

PAM es por lejos el más comunmente utilizado, pero existen otros como *CLARA*, *MELISSA*, y otros tantos que se adaptan a diferentes problemas. En

el caso de la localización de las balizas de satélites como el ARSAT e incluso de los nano satélites como el FossaSat-1 (cubesat) este método es muy utilizado. Ver Satelites para internet de las cosas Industriales. Todas estas tecnologías de satélites de bajo costo están permitiendo mejorar la competitividad y eficiencia de las cadenas de suministros en todo el planeta. En especial esta tipo de tecnología sirve para ver el deterioro de las propiedades alimenticias de cargas en línea y en tiempo real. A pesar de ello en gran parte de latinoamérica y caribe estas tecnología aún están por descubrirse. Se han utilizado varios métodos del **skimr()** (un sustituto de **summary()**), pero el paquete que utilizaremos para esta tarea específica será **cluster()** paquete exitosamente para problemas de congestión de puertos y misiones de vacunación post covid de el paquete : ClusterR

2.2 Agrupación K-means

El método de partición más común es el análisis de clúster de K-means. Conceptualmente, el algoritmo K-means:

1. Selecciona K centroides (K filas elegidas al azar)
2. Asigna cada punto de datos a su centroide más cercano
3. Recalcula los centroides como el promedio de todos los puntos de datos en un clúster (es decir, los centroides son vectores medios de p-longitud, donde p es el número de variables)
4. Asigna puntos de datos a sus centroides más cercanos
5. Continúa los pasos 3 y 4 hasta que no se reasignen las observaciones o se alcance el número máximo de iteraciones (R usa 10 como valor predeterminado). [bezerra_extracting_2019]

Los detalles de implementación de este enfoque pueden variar. R utiliza un algoritmo eficiente de Hartigan y Wong (1979) que divide las observaciones en grupos k de tal manera que la suma de cuadrados de las observaciones a sus centros de clúster asignados es un mínimo. Esto significa que en los pasos 2 y 4, cada observación se asigna al clúster con el valor más pequeño de:

$$SS_{(k)} = \sum_{i=1}^n \sum_{j=1}^p (x_{ij} - \bar{x}_{kj})^2$$

Donde k es un cluster, x_{ij} es el valor de la variable j ésima para la observación i ésima, y $\bar{x}_{kj}$ es la media de la variable j para el cluster k .

La agrupación en clústeres K-means puede manejar conjuntos de datos más grandes que los enfoques de clúster jerárquicos. Además, las observaciones no se comprometen permanentemente a un grupo. Se mueven cuando al hacerlo mejora la solución general. Sin embargo, el uso de medoides implica que todas las variables deben ser continuas y el enfoque puede verse gravemente afectado por valores atípicos. Por ejemplo k-means no es aplicable para localización de pallets en sistemas de posición flotante si dentro del edificio hay muros o barreras que impiden la circulación entre racks.[runkler_data_2020]

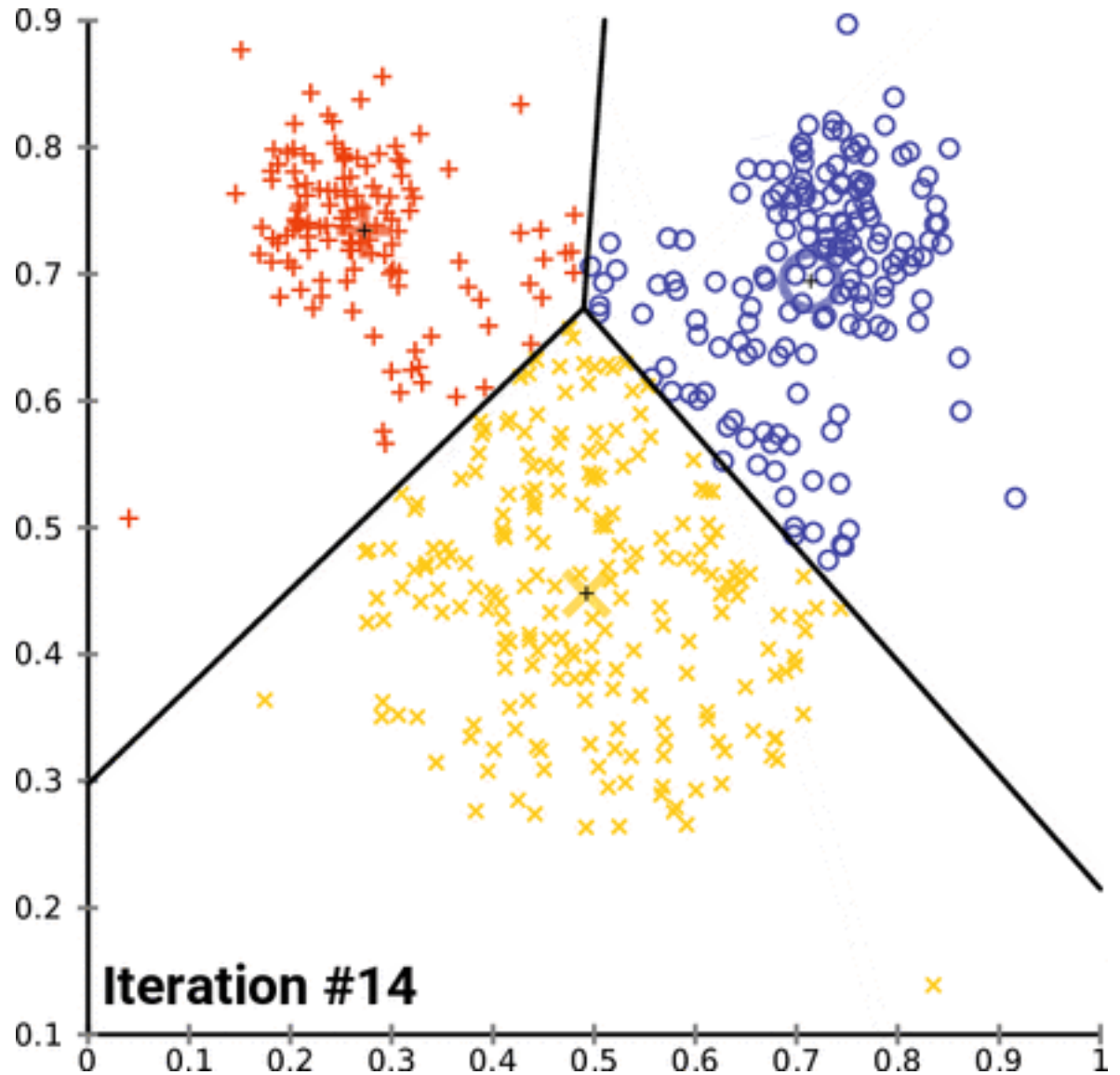
También se desempeñan mal en presencia de grupos no convexos (por ejemplo, en forma de U o celdas de manufactura flexible). El formato de la función K-means en R es:

$$kmeans(x, centros)$$

donde x es un conjunto de datos numéricos (matriz o data.frame) y $centros$ es el número de clústeres a extraer. Es un parámetro arbitrario que elijo y con el que pruebo o simulo alternativas. Por su bajo costo computacional converge razonablemente bien a una solución.

La función devuelve la pertenencia de casos al clúster, los centroides, las sumas de cuadrados (dentro, entre y total) y los tamaños de cada clúster.

Dado que el análisis de clústeres de K-medias comienza con k centroides elegidos al azar, se puede obtener una solución diferente cada vez que se invoca la función. Utilice la función `set.seed()` para garantizar que los resultados sean reproducibles, en especial si publicas en papers.



Además, este enfoque de agrupación puede ser sensible a la selección inicial de centroides. La función `kmeans()` tiene una opción o parámetro `nstart` que intenta varias configuraciones iniciales e informa sobre la mejor. Por ejemplo, agregando `nstart = 25` generará 25 configuraciones iniciales. Este enfoque a menudo es muy recomendable. A diferencia de la agrupación jerárquica, la agrupación en clústeres de *K-means* requiere que el número de clústeres que se extraer se especifique de antemano. Una vez más, el paquete `NbClust()` se puede utilizar como guía para estimar un número inicial de clusters aceptable. Además, puede ser útil un gráfico de las sumas totales de cuadrados dentro de los grupos contra el

número de clústeres en una solución de K-means. Una curva en el gráfico puede sugerirnos el número apropiado de clústeres. [li_nonparametric_2021]

El gráfico se puede producir mediante la siguiente función.

A efectos de sostener un estilo único en los gráficos amenudo conviene crear una función de ploteo en la que repito los parámetros.

El parámetro de datos es el conjunto de datos numéricos que se analizará, *nc* es el número máximo de grupos a considerar y la semilla es una semilla de número aleatorio. Aquí, se analiza un conjunto de datos que contiene 13 mediciones químicas en 178 muestras de vino italiano. Los datos provienen originalmente del Repositorio de Aprendizaje Automático de la UCI (<http://www.ics.uci.edu/~mllearn/MLRepository.html>), pero también se pueden acceder a ellos a través del paquete de rattle. En el listado se proporciona un análisis de conglomerado de K-means para los datos.[kumar_colour_2022]

```
## Rows: 177 Columns: 14
## -- Column specification -----
## Delimiter: ","
## dbl (14): 1, 14.23, 1.71, 2.43, 15.6, 127, 2.8, 3.06, .28, 2.29, 5.64, 1.04,...
##
## i Use 'spec()' to retrieve the full column specification for this data.
## i Specify the column types or set 'show_col_types = FALSE' to quiet this message.
```

Dimensiones:

columna	Propiedad
1)	Alcohol
2)	Ácido Malico
3)	Cenizas
4)	Alcalinidad de las cenizas
5)	Magnecio
6)	Fenoles Totales
7)	Flavonoides
8)	Fenoles no flavonoides
9)	Proantocianinas
10)	Intensidad de Color
11)	Tinte
12)	OD280/OD315 de vinos diluidos
13)	Prolina

Generaremos un reporte exploratorio automático con la biblioteca skim

Table 2: Data summary

Name	wine
Number of rows	177

Number of columns	14

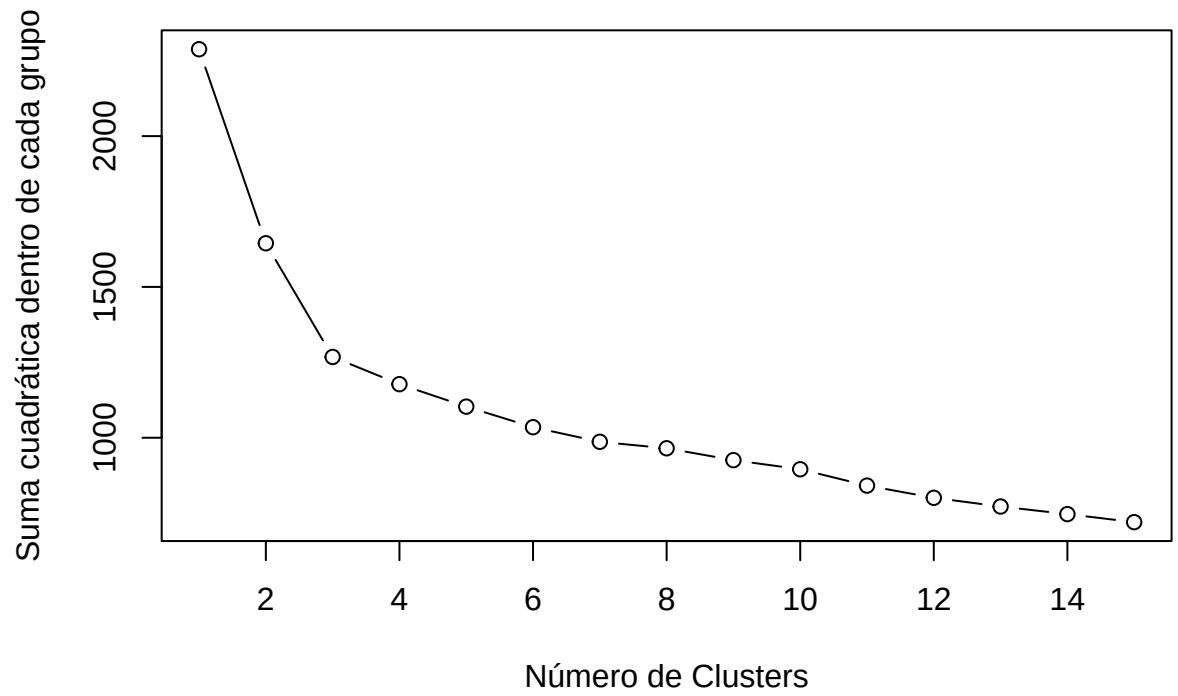
Column type frequency:	
numeric	14

Group variables	None

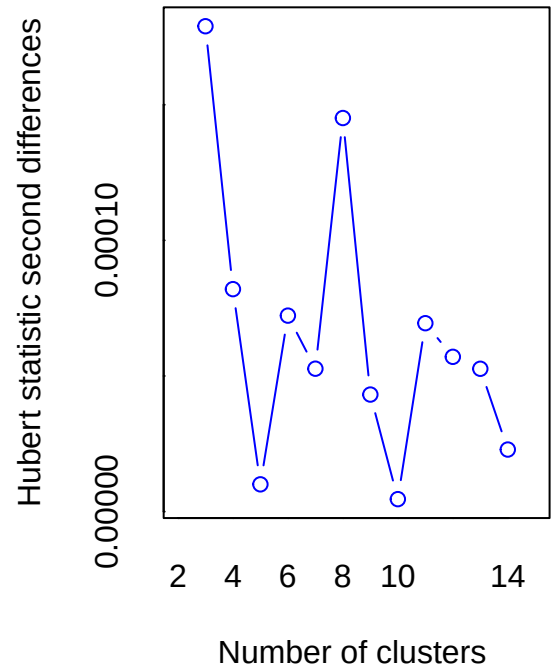
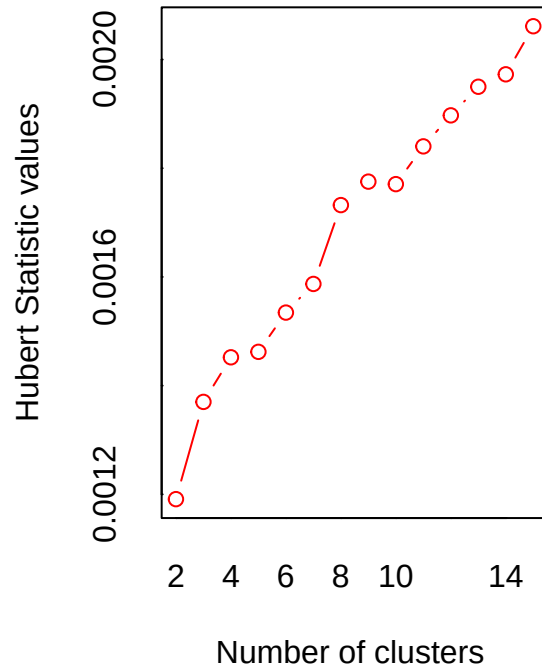
Variable type: numeric

skim_variable	n_missing	n_complete	mean	sd	p0	p25	p50	p75	p100	hist
Type	0	1	1.94	0.77	1.00	1.00	2.00	3.00	3.00	
Alcohol	0	1	12.99	0.81	11.03	12.36	13.05	13.67	14.83	
Ácido Malico	0	1	2.34	1.12	0.74	1.60	1.87	3.10	5.80	
Cenizas	0	1	2.37	0.28	1.36	2.21	2.36	2.56	3.23	
Alcalinidad de las cenizas	0	1	19.52	3.34	10.60	17.20	19.50	21.50	30.00	
Magnecio	0	1	99.59	14.17	70.00	88.00	98.00	107.00	162.00	
Fenoles Totales	0	1	2.29	0.63	0.98	1.74	2.35	2.80	3.88	
Flavonoides	0	1	2.02	1.00	0.34	1.20	2.13	2.86	5.08	
Fenoles no flavonoides	0	1	0.36	0.12	0.13	0.27	0.34	0.44	0.66	
Proantocianinas	0	1	1.59	0.57	0.41	1.25	1.55	1.95	3.58	
Intensidad de Color	0	1	5.05	2.32	1.28	3.21	4.68	6.20	13.00	
Tinte	0	1	0.96	0.23	0.48	0.78	0.96	1.12	1.71	
OD280/OD315 de vinos diluïdos	0	1	2.60	0.71	1.27	1.93	2.78	3.17	4.00	
Prolina	0	1	745.10	14.82	78.00	100.00	172.00	285.00	680.00	

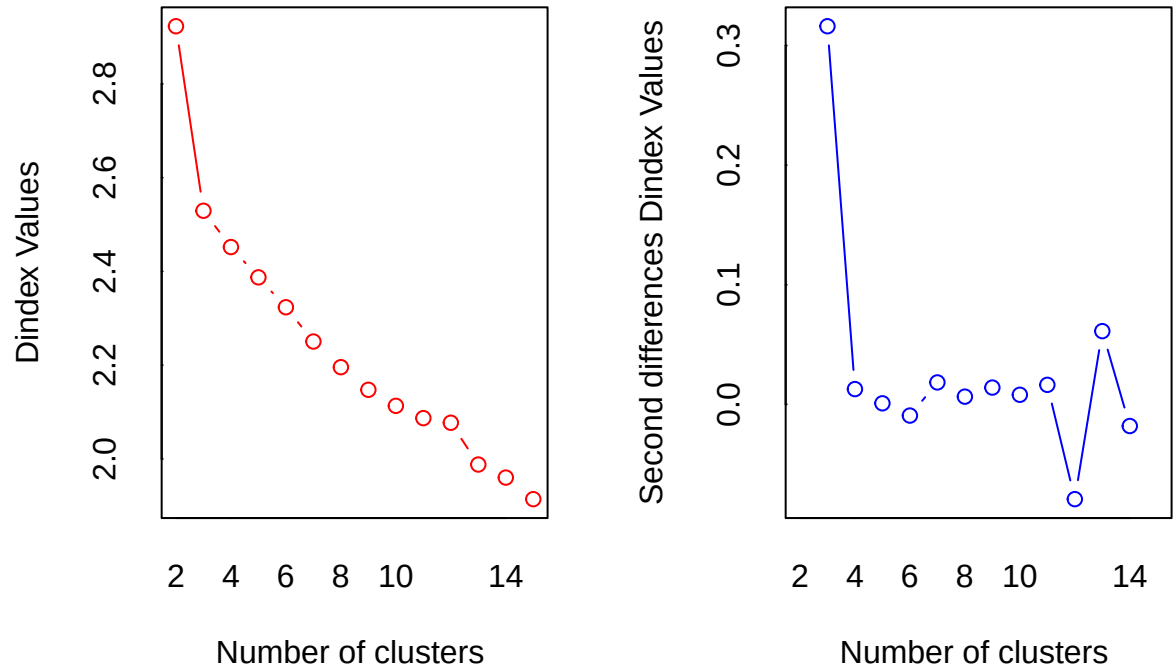
Procederemos a escalar el dataset, que es un poco menos eficiente que normalizar, pero es más rápido y nos muestra algunas de las herramientas de la biblioteca **skimr**. [chen_evidential_2018]



Cargaremos la biblioteca **NbClust()** para usar el método kmeansn que ella proporciona.



```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##       In the plot of Hubert index, we seek a significant knee that corresponds
##       significant increase of the value of the measure i.e the significant peak.
##       index second differences plot.
##
```



```
## *** : The D index is a graphical method of determining the number of clusters.
##           In the plot of D index, we seek a significant knee (the significant peak
##           second differences plot) that corresponds to a significant increase of t
##           the measure.
##
## *****
## * Among all indices:
## * 1 proposed 2 as the best number of clusters
## * 19 proposed 3 as the best number of clusters
## * 1 proposed 9 as the best number of clusters
## * 2 proposed 15 as the best number of clusters
##
##           ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is 3
##
## *****
```

```
##
##  0  1  2  3  9 15
##  2  1  1 19  1  2
```

Dado que las variables varían en rango, se estandarizan antes de la agrupación (función *scale()*) es muy importante utilizarla ya que esta biblioteca es muy sensible al error si no se aplica normalizado.

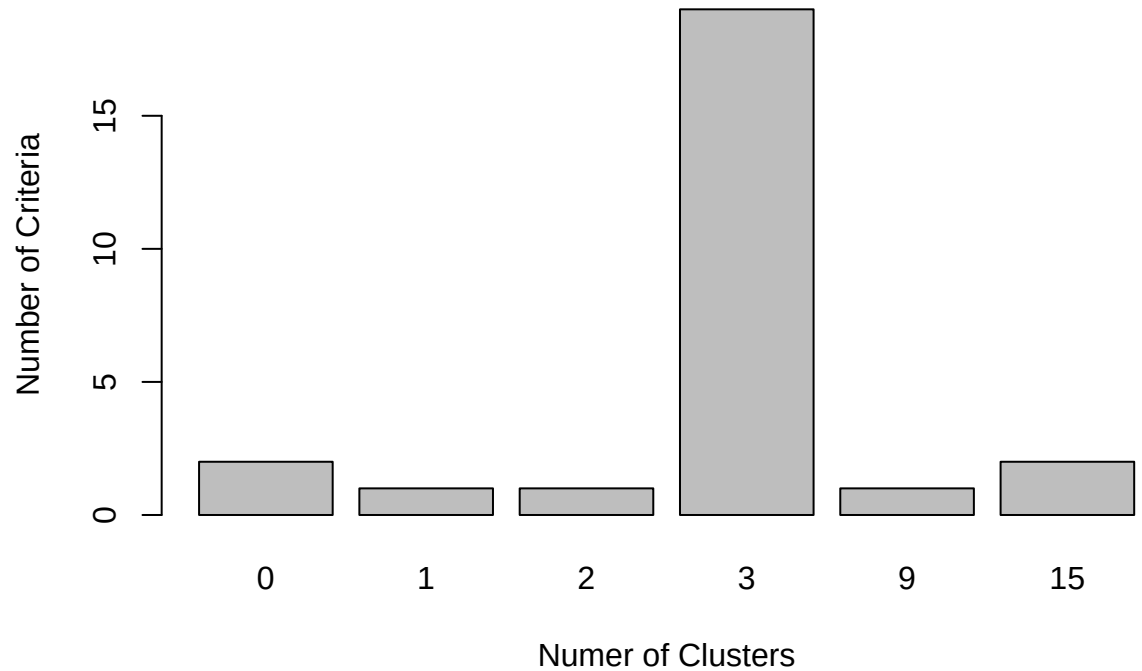
A continuación, el número de clústeres se determina mediante las funciones *wvsplot()* y *NbClust()*. La Figura indica que hay una clara caída en la suma de cuadrados dentro de los grupos cuando se pasa de 1 a 3 grupos. Después de tres clústeres, esta disminución disminuye, lo que sugiere que una solución de 3 clústeres puede ser una buena opción para los datos.

En la siguiente figura, 14 de los 24 criterios proporcionados por el paquete *NbClust* sugieren una solución de 3 clústeres.

Tenga en cuenta que no se pueden calcular los 30 criterios para cada conjunto de datos. Se obtiene una solución final de clúster con la función *kmeans()* y se imprimen los centroides de clúster.

Dado que los centroides proporcionados por la función se basan en datos estandarizados, la función *aggregate()* se utiliza junto con las pertenencias al clúster para determinar las medias variables para cada clúster en la métrica original (o desescalando).

Number of Clusters Chosen by 26 Criteria



¿Cuántas muestras hay en cada grupo?

```
## [1] 61 65 51
```

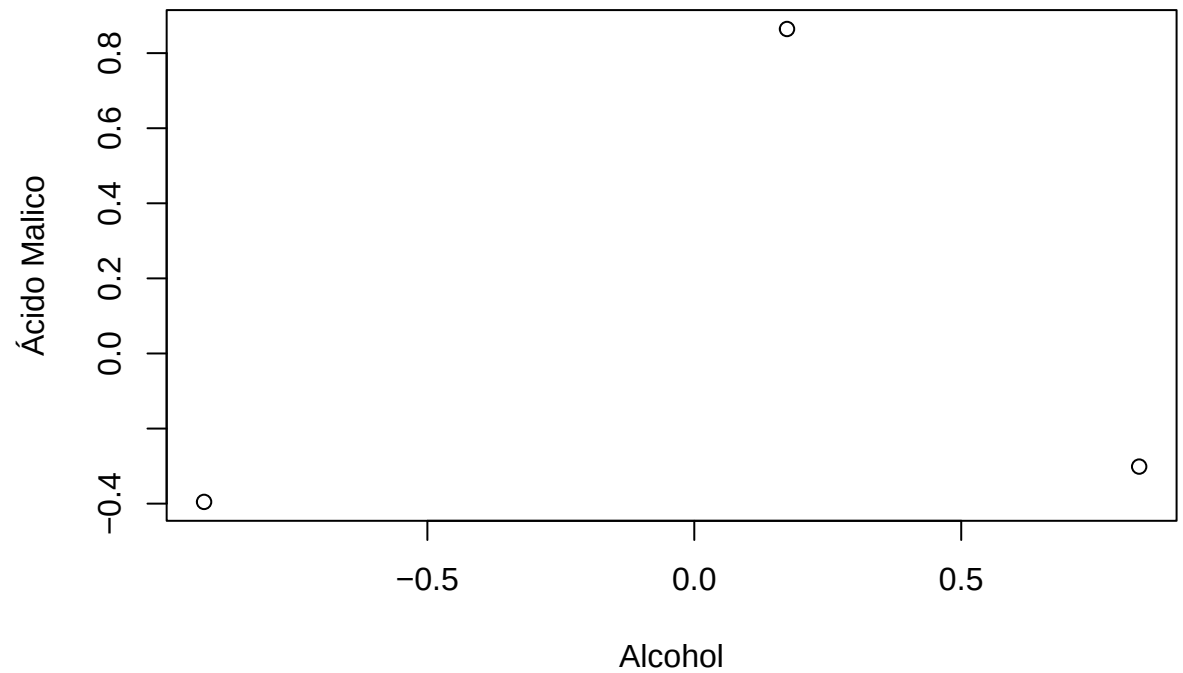
Dónde estan los centroides?

```
##      Alcohol  Ácido Malico      Cenizas Alcalinidad de las cenizas      Magnesio
## 1  0.8333649  -0.3013131  0.3661731                -0.6065538  0.56922228
## 2 -0.9183253  -0.3953334 -0.4905017                0.1637039 -0.48321576
## 3  0.1736447   0.8642504  0.1871775                0.5168437 -0.06497127
##  Fenoles Totales Flavonoides Fenoles no flavonoides Proantocianinas
## 1    0.88768039  0.98016451                -0.56173008    0.57583669
## 2   -0.07114136  0.02658937                -0.03709561    0.06509498
## 3   -0.97106500 -1.20624204                0.71915195   -0.77171004
##  Intensidad de Color      Tinte OD280/OD315 de vinos diluídos      Prolina
## 1         0.1702296  0.4753467                0.7753334  1.1296451
## 2        -0.8955790  0.4614076                0.2823571 -0.7460740
## 3         0.9378162 -1.1566204                -1.2872265 -0.4002655
```

Mira con detenimiento los valores entregados por

fit

(...)



Matriz de confusión

```
##
##      1  2  3
##  1 58  0  0
##  2  3 65  3
##  3  0  0 48
```

Como podemos ver se nos ha indicado que es muy probable que existan **3 categorías** o grupos en los que podemos clasificar o etiquetar este dataset. La matriz de confusión es muy buena.