Aplicaciones del análisis multivariante con R

Autores

Laura Gómez Garrido Miguel Lentisco Ballesteros Antonio Martín Ruiz Daniel Pozo Escalona Francisco Javier Saéz Maldonado

Dependencias

Instalar paquetes.

```
install.packages("MVA")
install.packages("HSAUR2")
install.packages("car")
install.packages("MASS")
```

```
Updating HTML index of packages in '.Library'

Making 'packages.html' .
 done

Updating HTML index of packages in '.Library'

Making 'packages.html' .
 done

Updating HTML index of packages in '.Library'

Making 'packages.html' .
 done

Updating HTML index of packages in '.Library'

Making 'packages.html' .
 done
```

```
library("MVA")
library("HSAUR2")
library("car")
library("MASS")
```

```
Loading required package: HSAUR2

Loading required package: tools

Loading required package: carData
```

Introducción

Qué es R

R es un entorno y lenguaje de programación enfocados a la computación estadística y de gráficos. Surge como una reimplementación libre del lenguaje y entorno S. Fue creado en 1993 por Ross Ihaka y Robert Gengleman en el de partamento de estadística de la universidad de Auckland, Nueva Zelanda.

Actualmente está desarrollado por el R Development Core Team. Existe también una activa comunidad que contribuye mediante el reporte de fallos y creación de nuevas funcionalidades.

Su código fuente está escrito principalmente en C, Fortran y el mismo R, y está disponible como software libre bajo los términos de la GNU General Public License de la Free Software Foundation. Puede ser compilado y ejecutado en una gran cantidad de plataformas UNIX, Windows y MacOs.

En este trabajo trataremos de abordar algunas de las funcionalidades básicas que nos ofrece R orientadas a la estadística multivariante.

Extensiones y paquetes

R forma parte de un proyecto colaborativo y abierto donde sus propios usuarios pueden publicar paquetes que extienden su funcionalidad básica.

Para facilitar el desarrollo de nuevos paquetes, se ha puesto a servicio de la comunidad una forja de desarrollo, conocida como **R-Forge**, que facilita las tareas relativas a dicho proceso. Esto es, en definitiva, una plataforma central basada en **FusionForge** para el desarrollo de paquetes, de software relacionado y futuros proyectos.

A fecha de Diciembre de 2019, el repositorio oficial de paquetes de **R** tiene disponibles 15315 paquetes, los cuales han sido organizados en varias vistas, o temas, que permiten agruparlos según su naturaleza o función. Algunos vistas serían:

- Bayesian : Vista dedicada a la Inferencia Bayesiana en la que podemos encontrar paquetes para ajustes generalizados de modelos, modelos o métodos específicos, aprender Estadística Bayesiana o que enlazan con motores de muestreo.
- ChemPhys : En esta vista nos centramos en la Quimiometría y la Física Computacional. Estas disciplinas se ocupan del análisis de los datos que surgen en experimentos de química y física, así como la simulación de los sistemas fisicoquímicos.

- ClinicalTrials : Vista dedicada al diseño, monitorización y análisis de Ensayos Clínicos.
- Cluster : Aquí se centran en el Análisis de Clusters y Modelos de Mezcla Finita.
- Databases : Paquetes relacionados con la accesibilidad a diferentes bases de datos, sin incluir la importación/exportación de datos o su control.
- DifferentialEquations : Paquetes dedicados al análisis y resolución de Ecuaciones Diferenciales.
- Distribution : Para las distribuciones clásicas, base R implementa algunas funcionalidades básicas. Estos paquetes están dedicados a aumentar la cantidad de funcionalidades disponibles para el análisis de dichas distribuciones, en particular para las distribuciones multivariantes.
- Genetics : El enfoque de esta vista está centrada en los paquetes de R que implementan métodos y algoritmos estadísticos para el análisis de datos genéticos y para el estudio de la genética de poblaciones motivada por los grandes avances en el campo del análisis genético de los últimos años.
- MachineLearning : Dentro de este tema, los paquetes pueden ser divididos en varios topics como pueden ser *Redes Neuronales y Deep Learning*, *Particionamiento Recursivo* o *Bosques Aleatorios*, entre otros muchos.
- Multivariate : Agrupación de paquetes dedicados a la Estadística Multivariante, siendo que algunos de ellos pueden ser encontrados en algunas otras vistas al dedicarse a varios ámbitos.

Operaciones básicas

Hola mundo

Comenzamos viendo las funcionalidades más básicas. Para asignar valores a varibales podemos utilizar los operadores `<-`, `->` o `=`. Para hacer comentarios se utiliza `#`.

```
# Esto es un comentario
miString <- "Hola mundo"
print(miString)
```

```
[1] "Hola mundo"
```

Tipos

Para la definición de variables R cuenta con tipos dinámicos, por lo que no es necesario especificar el tipo al declarar la variable. Las variables toman el tipo del valor que se les asigna. Existen séis tipos básicos.

```
v <- TRUE
print(class(v))
v <- 42
print(class(v))
v <- 2L
print(class(v))
v <- 4+2i
print(class(v))
v <- "multivariante"
print(class(v))
v <- charToRaw("multivariante")
print(class(v))
```

```
[1] "logical"
[1] "numeric"
[1] "integer"
[1] "complex"
[1] "character"
[1] "raw"
```

Estructuras de datos

R implementa las siguientes estructuras de datos

- Vectores. Concatenación de datos del mismo tipo. Si los datos son de distinto tipo, R realiza una transformación automática.
- Listas. Concatenaciones de datos de diferentes tipos, generalmente identificados con un nombre.
- Matrices. Conjunto de datos 2-dimensional rectangular con datos del mismo tipo.
- Arrays. Conjunto de datos n-dimensional.
- Data frames. Objetos tabulares. Cada columna puede contener datos de un tipo diferente.

```
vector <- c('v1', 'v2', 'v3')
print(vector)
lista <- list(c(1,2,3), 2+3i, "multivariante")
print(lista)
matriz <- matrix(c(1,2,3,4,5,6), nrow = 2, ncol = 3)
print(matriz)
array <- array(c('hola', 'adios'), dim=c(3,2,3))
```

```r
print(array)
marco <- data.frame(
    nombre = c("A", "B", "C"),
    peso = c(1,2,3),
    cantidad = c(4,5,6)
)
print(marco)
```

```
[1] "v1" "v2" "v3"
[[1]]
[1] 1 2 3

[[2]]
[1] 2+3i

[[3]]
[1] "multivariante"

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

     [,1]    [,2]
[1,] "hola"  "adios"
[2,] "adios" "hola"
[3,] "hola"  "adios"

, , 2

     [,1]    [,2]
[1,] "hola"  "adios"
[2,] "adios" "hola"
[3,] "hola"  "adios"

, , 3

     [,1]    [,2]
[1,] "hola"  "adios"
[2,] "adios" "hola"
[3,] "hola"  "adios"

  nombre peso cantidad
1      A    1        4
2      B    2        5
3      C    3        6
```

Operadores

R implementa los siguientes operadores, todos válidos para vectores

Aritméticos

- `+` Suma
- `-` Resta
- `*` Producto
- `/` División
- `%%` Módulo
- `%/%` División entera
- `^` Potencia

Relacionales

- `<` Menor que
- `>` Mayor que
- `==` Igualdad
- `<=` Menor o igual
- `>=` Mayor o igual
- `!=` Diferente

Lógicos

- `&` AND
- `|` OR
- `!` NOT
- `&&` AND entre los primeros elementos de los vectores
- `||` OR entre los primeros elementos de los vectores

Misceláneos

- `->`, `<-`, `=`, `->>`, `<<-` Asignación
- `:` Crea un vector con todos los valores entre los dados
- `%in%` Identifica si un elemento está en un vector
- `%%*%%` Multiplica una matriz con su traspuesta

Estructuras de decisión

R implementa las siguientes estructuras de decisión

- if
- if..else
- switch

```
if (3 > 2) {
    print("3>2")
}
if (2 > 3){
    print("2>3")
}

i = 4
if(i > 5){
    print("i>5")
} else {
    print("i<5")
}

x <- switch(2, "a", "b", "c")
print(x)
```

```
[1] "3>2"
[1] "i<5"
[1] "b"
```

Bucles

R implementa los siguientes bucles

- repeat
- while
- for

```
i = 0
repeat{
    i = i+1
    if(i > 3){
        break
    }
}
print(i)

i = 0
while(i < 5){
    i = i+1
}
print(i)
```

```r
v <- 1:4
for (i in v){
    v[i] = i+1
}
print(v)
```

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

Lectura de datos

Podemos leer datos de multitud de bases de datos ofrecidas tanto en web, como que podamos tener en nuestro propio dispositivo.

Existen multitud de funciones para leer los datos según el fichero de donde las queramos leer. Algunos ejemplos son:

- `read.csv` o `read.csv2`
- `read.delim`
- `read.table`

Como ejemplo, vamos a usar `read.table(fuente,separador)` para leer una base de datos:

```r
wine <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data'
```

La base de datos que hemos leído contiene los datos de las concentraciones de 13 tipos de productos químicos en vinos fabricados en la misma región de *Italia*. El primer elemento es solo un identificador de la fábrica que puede tomar dos valores: 1 ó 3. Es por ello que podemos considerar que vamos a trabajar con un vector aleatorio $X = (X_2, \ldots, X_{14})$

Podemos ver el contenido de esta base de datos escribiendo el nombre de la variable en la que la hemos guardado:

```r
wine
```

Dibujando los datos

Ahora, mostraremos una librería y ciertas funciones que nos permiten dibujar de diversas formas la muestra que hemos obtenido en los datos.

Utilizaremos la librería `car`, y dentro de ella la función `scatterplotMatrix`. Hay que llamar a esta función utilizando la variable donde tenemos la base de datos y un rango, que nos indica el rango de variables que queremos mostrar. Por ejemplo, lo hacemos para las dos primeras variables:

In [4]:
```
wine
```

A data.frame: 178 × 14

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>
1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480

Figure 1: Wine dataset

```
library("car")
scatterplotMatrix(wine[2:3])
```

Cálculo usando los datos

Una vez hemos obtenido los datos y los hemos representado para poder tener una visión de los mismos, pasamos a realizar cálculos con ellos. Utilizaremos la función `sapply` para aplicar una función sobre cada columna de nuestro *data frame*. Utilizaremos funciones pre-definidas en *R* para obtener información sobre nuestro conjunto de datos.

Media y desviación típica muestrales

Calculamos primero la media muestral. La función que tiene *R* para hacerlo es `mean()`. Usando esta función y la anteriormente mencionada, podemos calcular la media muestral da todas nuestras variables aleatorias. Lo haremos para el conjunto total de los datos, y luego hacemos una prueba en la que seleccionamos únicamente las muestras que han sido obtenidas en la *fábrica numero 1*

```
print("Media muestral del conjunto completo")
sapply(wine[2:14],mean)
selection1 <- wine[wine$V1 == "1",]
selection3 <- wine[wine$V1 == "3",]
print("Comparativa de medias muestrales de vinos en primera y tercera fábricas")
mean1 <- sapply(selection1[2:14],mean)
mean2 <- sapply(selection3[2:14],mean)

chemical <- c(2,3,4,5,6,7,8,9,10,11,12,13,14)
plot(chemical,mean1,col = "red")
points(chemical, mean2, col="blue", pch="*")
legend(2,1000,legend=c("Medias en Fabrica 1","Medias en Fabrica 2"), col=c("red","blue"),
```

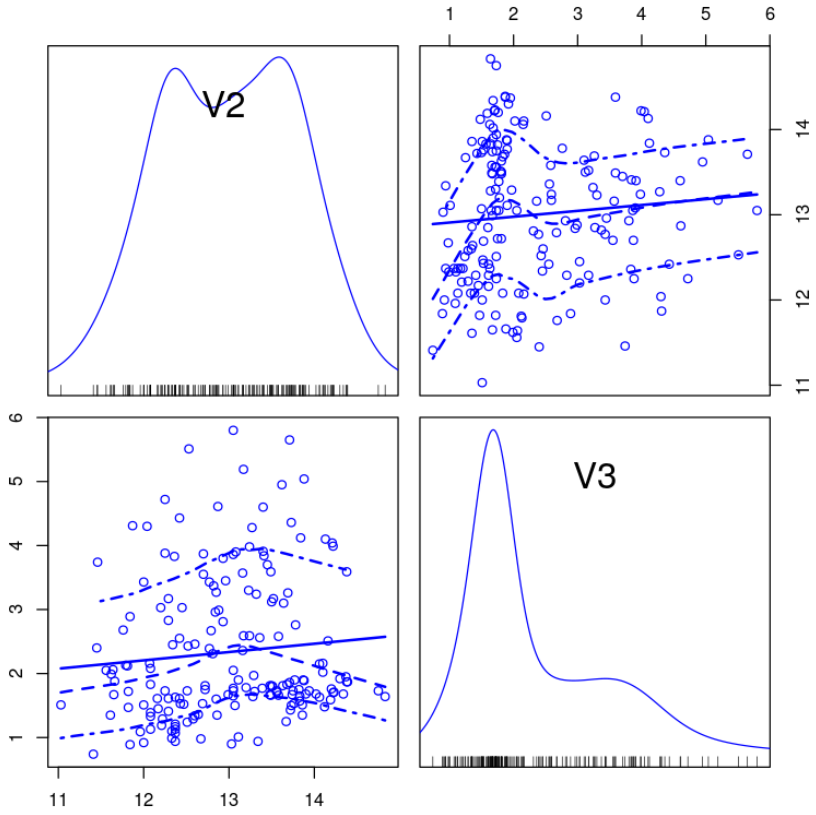

Figure 2: Gráficos

```
                                    pch=c("o","*"))

sapply(wine[2:14],sd)
```

```
[1] "Media muestral del conjunto completo"
```

V2	13.0006179775281
V3	2.33634831460674
V4	2.36651685393258
V5	19.4949438202247
V6	99.7415730337079
V7	2.29511235955056
V8	2.02926966292135
V9	0.36185393258427
V10	1.59089887640449
V11	5.05808988202247
V12	0.957449438202247
V13	2.61168539325843
V14	746.893258426966

```
[1] "Comparativa de medias muestrales de vinos en primera y tercera fábricas"
```

V2	0.811826538005857
V3	1.11714609761446
V4	0.274344009060815
V5	3.3395637671735
V6	14.2824835152957
V7	0.625851048833989
V8	0.998858685016947
V9	0.124453340296679
V10	0.572358862674761
V11	2.31828587182241
V12	0.228571565829823
V13	0.70999042876505
V14	314.907474276849

Figure 3: Medias

Implementación en R

Diagramas de dispersión

Vamos a ver como podemos representar diagramas de dispersión (*scatterplots*), de dos variables de X, que podrán ser muy útiles para ver mejor visualmente los datos, o para encontrar valores atípicos (*outliers*), puntos que son numéricamente distintos al resto.

Por ejemplo tomemos de *USairpollution* las variables *popul* (nº de población del censo de 1970) y *manu* (empresas productoras con al menos 20 trabajadores), donde las representamos con `plot`:

```
plot(popul ~ manu, data = USairpollution)
```

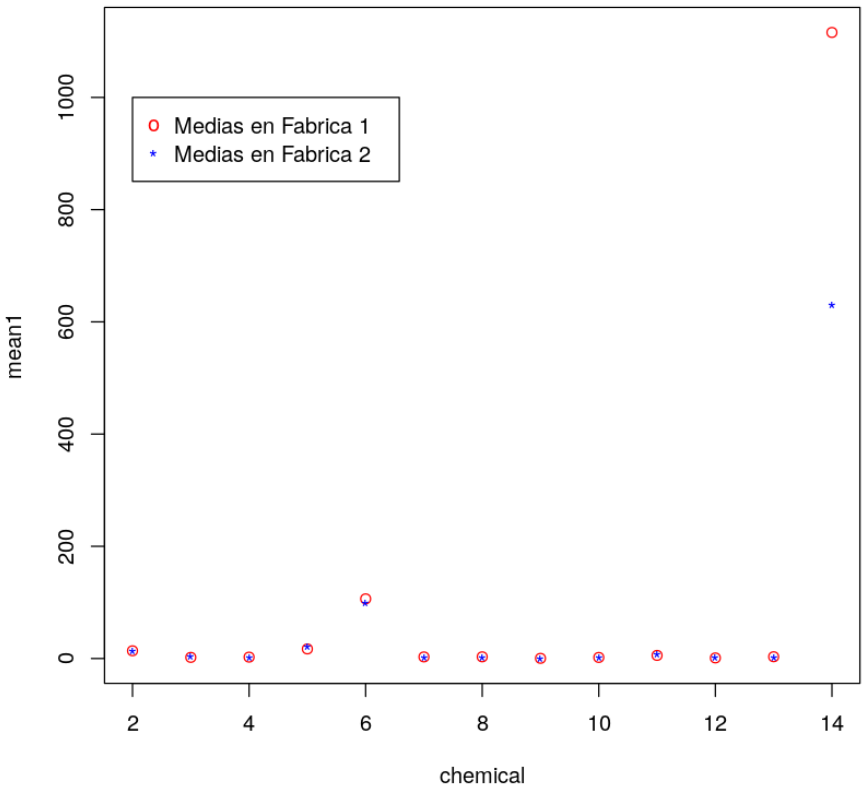

Figure 4: Medias fábrica 1 y 3

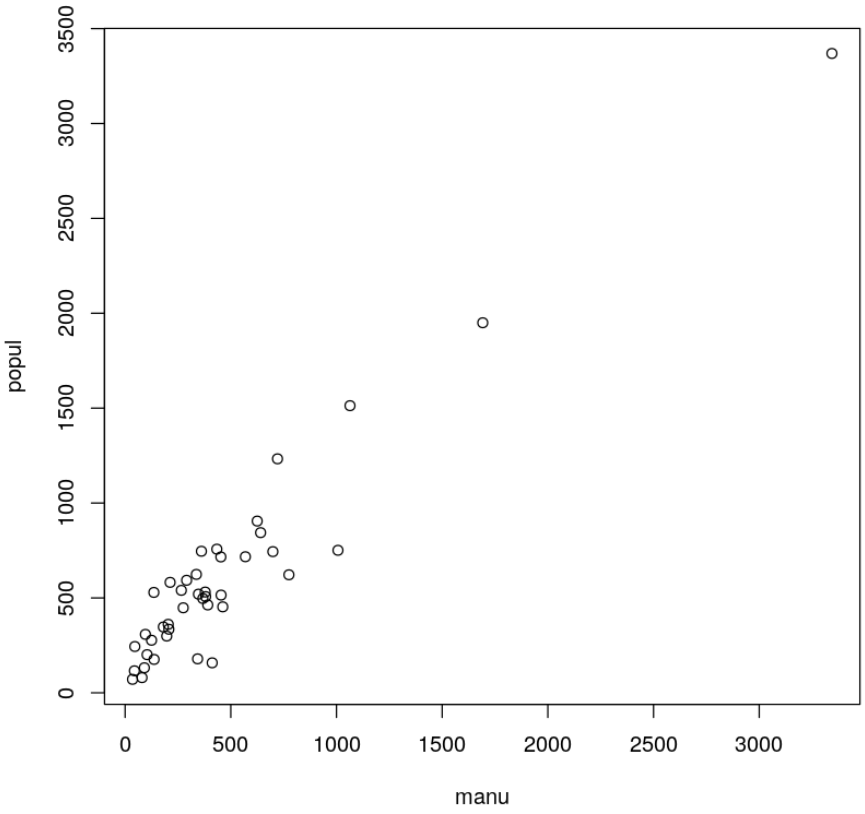

Figure 5: Scatterplot

También podemos representar todas los scatterplots de todas las parejas posibles a la vez con `pairs`:

```
pairs(USairpollution, pch = ".", cex = 1.5)
```

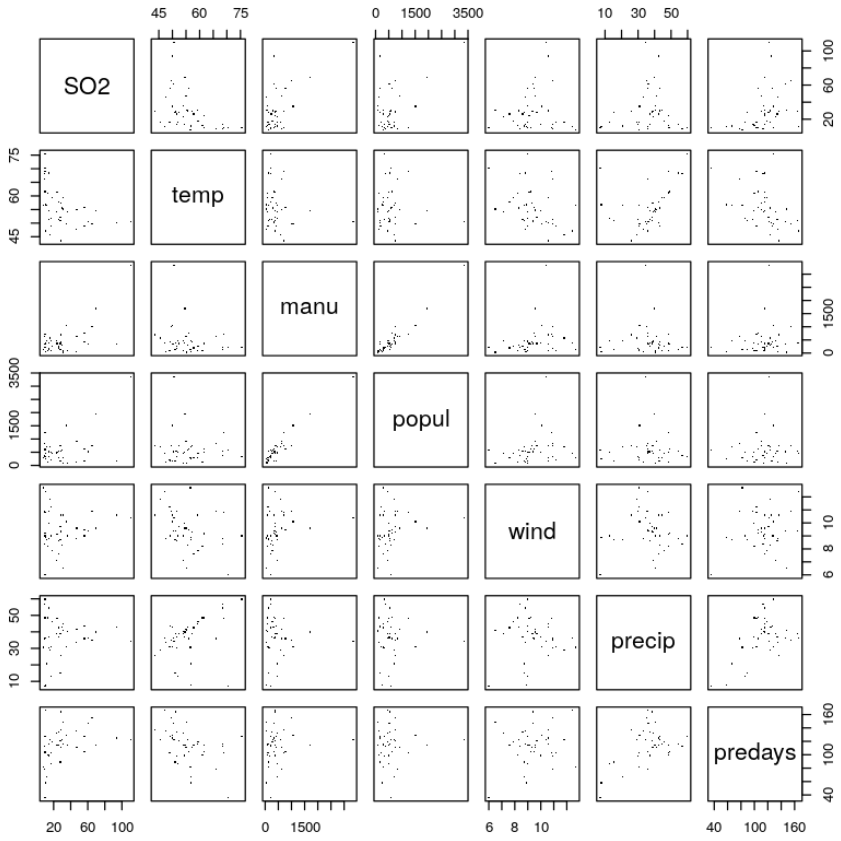

Figure 6: Parejas de scatterplots

Además podemos trazar las rectas de regresión (obteniéndolas con `lm`):

```
pairs(USairpollution,
      panel = function(x, y, .) {
          points(x, y, .)
          abline(lm(y ~ x), col = "grey")
      }, pch = ".", cex = 1.5)
```

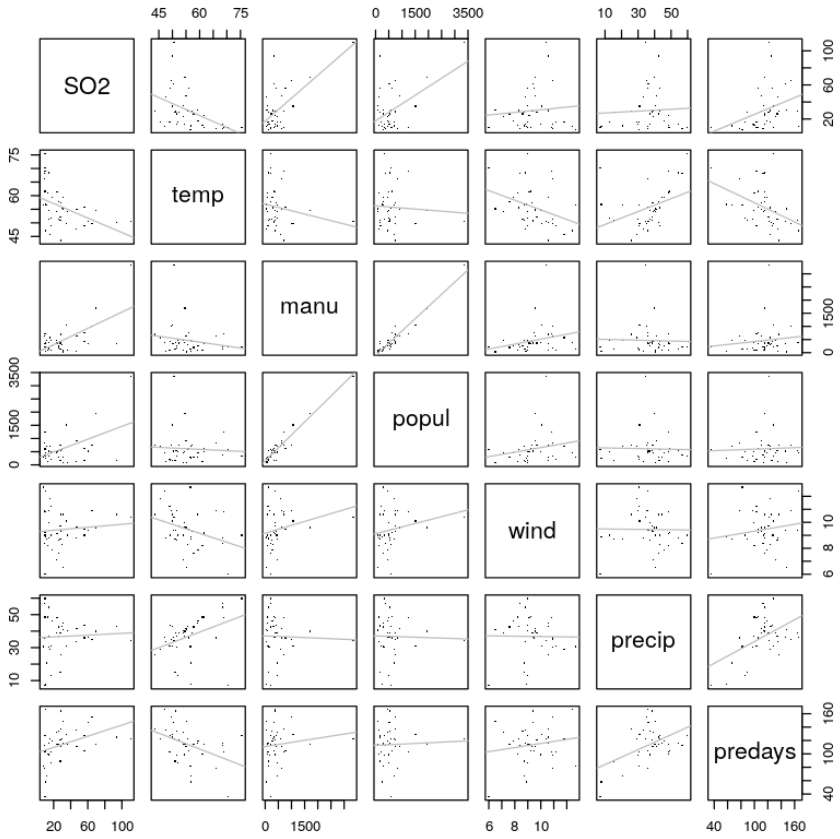

Figure 7: Parejas de scatterplots con regresión

Estandarización de variables

En muchas ocasiones, como cuando queremos comparar variables con distintas unidades o realizar un *análisis de componentes principales (PCA)*, necesitamos que nuestras variables tengan $\mu = 0$ y $\sigma = 1$.

Para ello, utilizamos la función `scale()` que tenemos en *R*.

Nota.- La función `scale()` no devuelve un objeto *dataframe* como los que hemos estado usando hasta ahora. Por tanto, tenemos que utilizar la función `as.data.frame()` para obtener un objeto de este tipo y poder trabajar con él como lo hacíamos antes.

```
standardised_wine <- as.data.frame(scale(wine[2:14]))
standardised_wine
```

A data.frame: 178 × 13

V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1.51434077	-0.56066822	0.23139979	-1.16630317	1.90852151	0.8067217	1.0319081	-0.65770780	1.22143845	0.251008784	0.3611585	1.84272147
0.24559683	-0.49800856	-0.82566722	-2.48384052	0.01809398	0.5670481	0.7315653	-0.81841060	-0.54318872	-0.292496232	0.4049085	1.11031723
0.19632522	0.02117152	1.10621386	-0.26798225	0.08810981	0.8067217	1.2121137	-0.49700500	2.12995937	0.268262912	0.3174085	0.78636920
1.68679140	-0.34583508	0.48655389	-0.80697481	0.92829983	2.4844372	1.4623994	-0.97911340	1.02925134	1.182731669	-0.4263410	1.18074072
0.29486844	0.22705328	1.83522559	0.45067448	1.27837900	0.8067217	0.6614853	0.22615759	0.40027531	-0.318377423	0.3611585	0.44833648
1.47738706	-0.51591132	0.30430096	-1.28607930	0.85828399	1.5576991	1.3622851	-0.17559941	0.66234866	0.729810822	0.4049085	0.33565890
1.71142720	-0.41744613	0.30430096	-1.46574348	-0.26196936	0.3273744	0.4912911	-0.49700500	0.67982021	0.082781041	0.2736585	1.36384178
1.30493643	-0.16680747	0.88751034	-0.56742256	1.48842650	0.4871569	0.4812796	-0.41665360	-0.59560339	-0.003489596	0.4486584	1.36384178
2.25341491	-0.62332789	-0.71631546	-1.64540766	-0.19195352	0.8067217	0.9518167	-0.57735640	0.67982021	0.061213382	0.5361584	0.33565890
1.05857838	-0.88291793	-0.35180959	-1.04652705	-0.12193769	1.0943301	1.1220109	-1.13981619	0.45268998	0.932546820	0.2299086	1.32158768
1.35420804	-0.15785609	-0.24245783	-0.44764644	0.36817315	1.0463954	1.2922052	-1.13981619	1.37868246	0.298457635	1.2799079	0.78636920
1.37884384	-0.76654998	-0.16955666	-0.80697481	-0.33198519	-0.1519728	0.4011882	-0.81841060	-0.03651359	-0.025057256	0.9299081	0.29340481
0.92308146	-0.54276546	0.15849862	-1.04652705	-0.75208020	0.4871569	0.7315653	-0.57735640	0.38280376	0.233754657	0.8424082	0.40608239
2.15487169	-0.54276546	0.08559744	-2.42395246	-0.61204853	1.2860690	1.6626279	0.54756319	2.12995937	0.147484019	1.2799079	0.16664254
1.69910930	-0.41744613	0.04914686	-2.24428828	0.15812565	1.6056339	1.6125708	-0.57735640	2.39203271	1.053325713	1.0611581	0.54692935
0.77526663	-0.47115441	1.21556562	-0.68719868	0.85828399	0.8866129	0.8817367	-0.49700500	-0.22870071	0.967055075	1.4111579	0.37791299
1.60056608	-0.37268923	1.28846679	0.15123418	1.41841067	0.8067217	1.1119995	-0.25595080	0.66234866	0.492566569	0.4924084	0.05396496
1.02162467	-0.68598755	0.92396093	0.15123418	1.06833150	1.0463954	1.3722966	0.30650899	0.22555975	0.665107844	0.7549083	-0.05871261
1.46506916	-0.66808479	0.41365272	-0.89680690	0.57822065	1.6056339	1.9029021	-0.33630220	0.47016154	1.570949537	1.1924080	0.29340481
0.78758453	0.68357369	0.70525741	-1.28607930	1.13834733	0.6469393	1.0018738	-1.54157319	0.12073042	0.018078063	0.0111587	1.05397844
1.30493643	-0.63227927	-0.31535901	-1.04652705	1.83850567	1.1262866	1.1420338	-0.97911340	0.88947889	0.255322316	0.5799084	1.54694284

Figure 8: Wine estandarizado

Si nos fijamos en el contenido del *dataframe* `wine` que obtuvimos al principio, este posee algunos valores muy grandes en algunas variables. Podemos observarlo fácilmente en que algunos de los vectores de medias tienen valores superiores a 700. Tratemos de aplicar ahora la media y la desviación típica a nuestro conjunto de datos estandarizado:

```
print("Medias estandarizadas")
sapply(standardised_wine,mean)
print("Desviacion típica estandarizadas")
sapply(standardised_wine,sd)
```

```
[1] "Medias estandarizadas"
```

V2	-8.59176620688482e-16
V3	-6.77644630873763e-17
V4	8.0451760363661e-16
V5	-7.72049403184734e-17
V6	-4.07393534579009e-17
V7	-1.39556006322723e-17
V8	6.95826335844938e-17
V9	-1.04218629278997e-16
V10	-1.22136897486352e-16
V11	3.64937578439868e-17
V12	2.09374057963209e-16
V13	3.00345929462872e-16
V14	-1.03442937443795e-16

```
[1] "Desviacion típica estandarizadas"
```

V2	1
V3	1
V4	1
V5	1
V6	1
V7	1
V8	1
V9	1
V10	1
V11	1
V12	1
V13	1
V14	1

Figure 9: Media estandarizada

Vemos que para todas las variables, $\mu_i \sim 0$ y $\sigma_i = 1,\ \forall i = 2, \ldots, 12$, con lo que nuestras variables quedaron estandarizadas.

Implementaciones del temario EMV

Distribución Normal Multivariante

Representamos una distribución normal multivariante de dimensión p mediante el vector de medias y la matriz de varianzas-covarianzas (incluiremos el p para que quede claro cual es la dimensión de la DNM).

Veamos como definimos la clase DNM, con lo que ya hemos mencionado:

```
DNM <- setRefClass("DNM",
                   fields = list(p = "numeric",
                                 media = "matrix",
                                 cov = "matrix"))
```

Necesitamos pasar la dimensión, el vector de medias y la matriz de varianzas-covarianzas; la dimensión es redundante pero así queda más claro visualmente. Se deja a responsabilidad del usuario que las dimensiones de la media y las covarianzas coincidan (y que la matriz de varianzas-covarianzas sea $>= 0$)

Nota: p es de tipo `numeric` por lo cual puede aceptar reales, se ha hecho así porque si se pone de tipo `integer` se tienen que hacer casteos todo el rato con `integer()` por lo que es mejor dejarlo en `numeric`y se deja al usuario que no ponga otra cosa que no sean enteros.

Ejemplo de creacion de la DNM $\boldsymbol{X} = (X_1, X_2, X_3)^T \sim N_3(\begin{pmatrix} 2 \\ 3 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 \\ 0 & 5 & -2 \\ 1 & -2 & 2 \end{pmatrix})$

```
# Vector medias
means <- matrix(c(2,3,-1), nrow = 3, ncol = 1)
# Matriz covarianzas
cov <- matrix(c(1, 0, 1, 0, 5, -2, 1, -2, 2), nrow = 3, ncol = 3)
# Creamos la DNM
X <- DNM$new(p = 3, media = means, cov = cov); X
```

```
Reference class object of class "DNM"
Field "p":
[1] 3
Field "media":
     [,1]
[1,]    2
[2,]    3
[3,]   -1
```

```
Field "cov":
     [,1] [,2] [,3]
[1,]    1    0    1
[2,]    0    5   -2
[3,]    1   -2    2
```

Función característica

Implementamos la **función caracteristica** de la DNM $\boldsymbol{X} = (X_1, \ldots, X_p)^T \sim N_p(\boldsymbol{\mu}, \Sigma)$, dada por

$$\Psi_{\boldsymbol{X}}(\boldsymbol{t}) = \exp\left(i\boldsymbol{t}^T\boldsymbol{\mu} - \frac{1}{2}\boldsymbol{t}^T\Sigma\boldsymbol{t}\right),\ \boldsymbol{t} \in \mathbb{R}^p$$

```
# Pre: X es DNM, t es matriz
funcion_caracteristica <- function(X, t) {
    # Comprobación de rango
    if (dim(t)[1] != X$p)
        stop("Filas de t no coinciden con dimensión de X")
    if (dim(t)[2] != 1)
        stop("t no es vector columna")
    # psi_X(t)
    exp(as.complex(1i * t(t) %*% X$media - 0.5 * t(t) %*% X$cov %*% t))
}
```

Ejemplo con la $\boldsymbol{X}$ anterior y $\boldsymbol{t} = \begin{pmatrix} 5 \\ 0 \\ 0 \end{pmatrix}$:

```
t = matrix(c(5, 0, 0), nrow = 3, ncol = 1)
psi_t <- funcion_caracteristica(X, t); psi_t
```

-3.12692857543366e-06-2.02737799857363e-06i

Función densidad

Implementamos la **función densidad** de una DNM $\boldsymbol{X} = (X_1, \ldots, X_n)^T \sim N_p(\boldsymbol{\mu}, \Sigma)$, con $\Sigma > 0$, que está definida para un

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right),\ \boldsymbol{x} \in \mathbb{R}^p$$

```
# Pre: X es DNM, x matriz
funcion_densidad <- function(X, x) {
    if (dim(x)[1] != X$p)
        stop("Filas de x no coinciden con dimensión de X")
    if (dim(x)[2] != 1)
```

```
        stop("x no es vector columna")
    if (det(X$cov) <= 0)
        stop("Matriz de covarianzas no es definida positiva")
    # f_X(x)
    exp(-0.5 * as.numeric(t(x - X$media) %*% solve(X$cov)
      %*% (x - X$media))) / ((2 * pi)^(X$p / 2) * sqrt(det(X$cov)))
}
```

Ejemplo con la $\boldsymbol{X}$ anterior y $\boldsymbol{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$:

```
x = matrix(c(1, 1, 1), nrow = 3, ncol = 1)
f_x <- funcion_densidad(X,x); f_x
```

1.43516532458314e-07

Transformación lineal

Implementamos la **transformación lineal**, que necesita como argumentos una DNM $\boldsymbol{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$, una matriz $B \in \mathcal{M}_{qxp}$ $(q \leq p)$ y un vector $\boldsymbol{b} \in \mathbb{R}^q$. Entonces devuelve la DNM definida como $\boldsymbol{Y} = B\boldsymbol{X} + \boldsymbol{b}$, entonces $\boldsymbol{Y} \sim N_q(B\boldsymbol{\mu} + \boldsymbol{b}, B\Sigma B^T)$.

```
# Pre: X es DNM; B, b matrices
transformacion_lineal <- function(X, B, b) {
    # Comprobaciones de rango
    if (dim(B)[1] > dim(B)[2])
        stop("Dimensión de B incorrecta (q > p)")
    if (dim(B)[2] != X$p)
        stop("Columnas de B no coinciden con dimensión de X")
    if (dim(b)[1] != dim(B)[1])
        stop("Filas de B no coinciden con dimensión de b")
    if (dim(b)[2] != 1)
        stop("b no es un vector columna")
    # Nueva DNM
    media_y = matrix(B %*% X$media + b, ncol = 1)
    cov_y = matrix(B %*% X$cov %*% t(B), nrow = dim(B)[1])
    DNM$new(p = dim(B)[1], media = media_y, cov = cov_y)
}
```

Ejemplo con la $\boldsymbol{X}$ anterior, $B = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}$, $b = \begin{pmatrix} 0 \end{pmatrix}$:

```
B = matrix(c(1, 1, 0), nrow = 1, ncol = 3)
b = matrix(0, nrow = 1, ncol = 1)
Y <- transformacion_lineal(X, B, b); Y
```

```
Reference class object of class "DNM"
Field "p":
[1] 1
```

```
Field "media":
     [,1]
[1,]    5
Field "cov":
     [,1]
[1,]    6
```

Marginalización

Implementamos la **marginalización** de una DNM $\boldsymbol{X} = (X_1, \ldots, X_p)^T \sim N_p(\boldsymbol{\mu}, \Sigma)$, que consiste en tomar el subvector $\boldsymbol{X_r} = (X_{r_1}, \ldots, X_{r_q})^T$ con $\boldsymbol{r} = (r_1, \ldots, r_q)^T, r_1, \ldots, r_q \in \{1, \ldots, q\}, q \leq p$ obteniendo $\boldsymbol{X_r} \sim N_q(\boldsymbol{\mu_r}, \Sigma_{\boldsymbol{r}})$ donde: - $\boldsymbol{\mu_r}$ es el subvector de $\boldsymbol{\mu}$ correspondiente a $\boldsymbol{r}$. - $\Sigma_{\boldsymbol{r}}$ es la submatriz de Σ definida por las filas y columnas correspondientes a $\boldsymbol{r}$.

```
# Pre: X es DNM, r es matriz
marginalizar <- function(X, r) {
    # Comprobar rangos
    if (dim(r)[2] != 1)
        stop("r no es vector columna.")
    if (dim(r)[1] > X$p)
        stop("Filas de r más que dimensión de X")
    if (any(r < 1 || r > X$p))
        stop("Algún r_i es < 1 ó > p")
    # mu_r
    media_r <- matrix(X$media[r, ], ncol = 1)
    # sigma_r
    cov_r <- matrix(X$cov[r, r], nrow = dim(r)[1])
    # X_r
    DNM$new(p = dim(r)[1], media = media_r, cov = cov_r)
}
```

Ejemplo con la $\boldsymbol{X}$ anterior, y $\boldsymbol{r} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$:

```
r = matrix(c(3, 1), nrow = 2, ncol = 1);
X_r <- marginalizar(X, r); X_r
```

```
Reference class object of class "DNM"
Field "p":
[1] 2
Field "media":
     [,1]
[1,]   -1
[2,]    2
Field "cov":
     [,1] [,2]
```

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

Partición

Dada una DNM $\boldsymbol{X} = (X_1, \ldots, X_p)^T \sim N_p(\boldsymbol{\mu}, \Sigma)$ con $p > 1$ y $\Sigma > 0$ podemos realizar una partición de $\boldsymbol{X} = (\boldsymbol{X}_{(1)}^T, \boldsymbol{X}_{(2)}^T)^T$ con $\boldsymbol{\mu} = (\boldsymbol{\mu}_{(1)}^T, \boldsymbol{\mu}_{(2)}^T)^T$ y

$\Sigma = \begin{pmatrix} \Sigma_{(11)} & \Sigma_{(12)} \\ \Sigma_{(21)} & \Sigma_{(22)} \end{pmatrix}$ de manera que $\boldsymbol{X}_{(1)} = (X_1, \ldots, X_q)^T$, y $\boldsymbol{X}_{(2)} = (X_{q+1}, \ldots, X_p)$ $(1 \leq q < p)$.

Implementamos un método para devolver las **DNM basadas en las particiones independientes**, que son $\boldsymbol{X}_{(1)} \sim N_q\left(\boldsymbol{\mu}_{(1)}, \Sigma_{(11)}\right)$ y $\boldsymbol{X}_{(2)} - \Sigma_{(21)}\Sigma_{(11)}^{-1}\Sigma_{(12)} \sim N_{p-q}\left(\boldsymbol{\mu}_{(2)} - \Sigma_{(21)}\Sigma_{(11)}^{-1}\boldsymbol{\mu}_{(1)}, \Sigma_{(22)} - \Sigma_{(21)}\Sigma_{(11)}^{-1}\Sigma_{(12)}\right)$

```
# Pre: X es DNM, q un entero
particiones_independientes <- function(X, q) {
    # Comprobaciones de rango
    if (q < 1 || q >= X$p)
        stop("q no puede ser < 1 ó >= p")
    if (det(X$cov) <= 0)
        stop("Sigma no es definida positiva")
    # X_1
    media_1 = matrix(X$media[1:q], ncol = 1)
    cov_11 = matrix(X$cov[1:q,1:q], nrow = q)
    X_1 <- DNM$new(p = q, media = media_1, cov = cov_11)
    # X_2 - cov_21 * cov_11^-1 * cov_12
    media_2 = matrix(X$media[(q+1):X$p], ncol = 1)
    cov_22 = matrix(X$cov[(q+1):X$p, (q+1):X$p], nrow = X$p - q)
    cov_12 = matrix(X$cov[1:q, (q+1):X$p], nrow = q)
    cov_21 = matrix(X$cov[(q+1):X$p, 1:q], ncol = q)
    X_2 <- DNM$new(p = X$p - q,
                   media = media_2 - cov_21 %*% solve(cov_11) %*% media_1,
                   cov = cov_22 - cov_21 %*% solve(cov_11) %*% cov_12)
    # Devolvemos
    c(X_1, X_2)
}
```

Ejemplo con la $\boldsymbol{X}$ anterior y $q = 2$:

```
particiones_independientes(X, 2)
```

```
[[1]]
Reference class object of class "DNM"
Field "p":
[1] 2
```

```
Field "media":
     [,1]
[1,]    2
[2,]    3
Field "cov":
     [,1] [,2]
[1,]    1    0
[2,]    0    5

[[2]]
Reference class object of class "DNM"
Field "p":
[1] 1
Field "media":
     [,1]
[1,] -1.8
Field "cov":
     [,1]
[1,]  0.2
```

Distribución condicionada

En las condiciones del apartado anterior, tenemos que la **distribución condicionada** de $\boldsymbol{X}_{(2)}$ dado $\boldsymbol{X}_{(1)} = \boldsymbol{x}_{(1)}$ es una DNM con $\boldsymbol{X}_{(2)} \sim N_{p-q}(\boldsymbol{\mu}_{(2)} + \Sigma_{(21)}\Sigma_{(11)}^{-1}(\boldsymbol{x}_{(1)} - \boldsymbol{\mu}_{(1)}), \Sigma_{(22)} - \Sigma_{(21)}\Sigma_{(11)}^{-1}\Sigma_{(12)})$.

Alternativamente, la distribución condicionada de $\boldsymbol{X}_{(1)}$ dado $\boldsymbol{X}_{(2)} = \boldsymbol{x}_{(2)}$ es una DNM con $\boldsymbol{X}_{(1)} \sim N_q(\boldsymbol{\mu}_{(1)} + \Sigma_{(12)}\Sigma_{(22)}^{-1}(\boldsymbol{x}_{(2)} - \boldsymbol{\mu}_{(2)}), \Sigma_{(11)} - \Sigma_{(12)}\Sigma_{(22)}^{-1}\Sigma_{(21)})$.

El último parametro es un booleano, si es `TRUE` entonces hace la condicionada dado $\boldsymbol{x}_{(2)}$, en caso contrario la hace dado $\boldsymbol{x}_{(1)}$.

```r
# Pre: X es DNM, q es entero, x es matriz, dado_x2 es booleano
particion_condicionada <- function(X, q, x, dado_x2) {
    # Comprobaciones de rango
    if (q < 1 || q >= X$p)
        stop("q no puede ser < 1 ó >= p")
    if (dado_x2 && dim(x)[1] != (X$p - q))
        stop("Filas de x_2 no coinciden con (p-q)")
    if (!dado_x2 && dim(x)[1] != (X$p - q))
        stop("Filas de x_1 no coinciden con q")
    if (det(X$cov) <= 0)
        stop("Sigma no es definida positiva")
    # Partición
    media_1 = matrix(X$media[1:q], ncol = 1)
    media_2 = matrix(X$media[(q+1):X$p], ncol = 1)
```

```
    cov_11 = matrix(X$cov[1:q,1:q], nrow = q)
    cov_22 = matrix(X$cov[(q+1):X$p, (q+1):X$p], nrow = X$p - q)
    cov_12 = matrix(X$cov[1:q, (q+1):X$p], nrow = q)
    cov_21 = matrix(X$cov[(q+1):X$p, 1:q], ncol = q)
    # DNM condicionada
    if (dado_x2) {
        p_cond = q
        media_cond = media_1 + cov_12 %*% solve(cov_22) %*% (x - media_2)
        cov_cond = cov_11 - cov_12 %*% solve(cov_22) %*% cov_21
    } else {
        p_cond = X$p - q
        media_cond = media_2 + cov_21 %*% solve(cov_11) %*% (x - media_1)
        cov_cond = cov_22 - cov_21 %*% solve(cov_11) %*% cov_12
    }
    DNM$new(p = p_cond, media = media_cond, cov = cov_cond)
}
```

Ejemplo con la $\boldsymbol{X}$ anterior y $q = 2$, dado $x_{(2)} = \begin{pmatrix}1\end{pmatrix}$:

```
X_cond <- particion_condicionada(X, 2, matrix(1), TRUE); X_cond
```

```
Reference class object of class "DNM"
Field "p":
[1] 2
Field "media":
     [,1]
[1,]    3
[2,]    1
Field "cov":
     [,1] [,2]
[1,]  0.5    1
[2,]  1.0    3
```

Normalización

Sea $\boldsymbol{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$ con $\Sigma > 0$ devolvemos la matriz $A \in M_{pxp}$ no singular tal que $A(\boldsymbol{X} - \boldsymbol{\mu}) \sim N_p(\boldsymbol{0}, I_p)$ (lo conseguimos con la factorización de Cholesky)

```
matriz_normalizacion <- function(X) {
    solve(t(chol(X$cov)))
}
```

Ejemplo con la $\boldsymbol{X}$ anterior:

```
A <- matriz_normalizacion(X)
print(round(A %*% X$cov %*% t(A)))
```

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

```
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

Inferencia en la DNM

Con $\boldsymbol{X} = (X_1, \ldots, X_p)^T \sim N_p(\boldsymbol{\mu}, \Sigma)$, $\Sigma > 0$, consideramos una muestra aleatoria simple dada por $X = (\boldsymbol{X}_1^T, \ldots, \boldsymbol{X}_N^T)^T$, donde N es el tamaño muestral; de manera que cada observación se representa con $\boldsymbol{X}_\alpha, \alpha \in \{1, \ldots, N\}$.

Veremos como hacer inferencia sobre la DNM que determina X.

Muestra aleatoria simple

Dada $\boldsymbol{X} \sim N_p(\boldsymbol{\mu}, \Sigma), \Sigma > 0$, podemos generar una muestra aleatoria simple de $\boldsymbol{X}$ de tamaño N, mediante la siguiente función:

```
muestra_aleatoria <- function(N, media, cov) {
    mvrnorm(N, media, cov)
}

muestra_aleatoria_EMV <- function(N, X) {
    mvrnorm(N, X$media, X$cov)
}
```

Ejemplo de la m.a.s de tamaño 14 de la DNM $\boldsymbol{X} = (X_1, X_2, X_3)^T \sim (\begin{pmatrix} 2 \\ 3 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & -0.5 \\ 1 & -0.5 & 2 \end{pmatrix})$:

```
X_DNM <- DNM$new(p = 3, media = matrix(c(2,3,-1), ncol = 1),
  cov = matrix(c(1,0,1,0,1,-0.5,1,-0.5, 2), ncol = 3, nrow = 3))
X <- muestra_aleatoria_EMV(20, X_DNM); X[1:10,]
```

Media muestral

Para la media muestral $\bar{\boldsymbol{X}} = \frac{1}{N}\sum_{\alpha=1}^{N} \boldsymbol{X}_\alpha = \begin{pmatrix} \bar{X}_1 \\ \vdots \\ \bar{X}_p \end{pmatrix}$ usamos `colMeans`(la media de cada variable/columna), además este es el estimador máximo verosímil ($\hat{\boldsymbol{\mu}} = \bar{\boldsymbol{X}}$).

```
media_muestral <- function(X) {
    matrix(colMeans(X), ncol = 1)
}
```

A matrix: 10 × 3 of type dbl

2.00863770	2.820845	-0.9536311
2.63738700	3.910852	-0.1236769
2.79081625	2.245214	0.5986766
3.41135996	2.534212	2.1532126
3.12791616	1.909552	0.6877082
-0.09439048	1.975222	-2.6908610
1.06985983	3.158471	-0.3117945
2.12817850	2.391889	-0.9534681
1.82641447	2.238796	-0.5196546
2.84639585	2.609973	0.5903323

Figure 10: Muestra aleatoria simple

Ejemplo con X anterior:

```
media_muestral(X)
```

A matrix: 3 ×

1 of type dbl

2.2408009

3.1898752

-0.7302118

Matrices sobre covarianzas/correlaciones

Implementamos las siguientes matrices:

- Matriz de dispersiones muestral: $A = \sum_{\alpha=1}^{N} (\boldsymbol{X}_\alpha - \bar{\boldsymbol{X}})(\boldsymbol{X}_\alpha - \bar{\boldsymbol{X}})^T$
- Matriz de covarianzas muestral: $S_N = \frac{1}{N} A$
- Matriz de cuasi-covarianzas muestral: $S_{N+1} = \frac{1}{N-1} A$ (también llamada matriz de covarianzas muestral)
- Matriz de correlaciones muestral: $R = D^{1/2} S_N D^{-1/2}$, donde D es la diagonal de S_N

Sabemos que $\hat{\Sigma} = S_N$, $T = S_{N+1}$ es estimador eficiente de Σ, y $\hat{p} = R$ (coeficientes de correlacion lineal de Pearson).

```
disp_muestral <- function(X) {
    cov(X) * (nrow(X) - 1)
}

cov_muestral <- function(X) {
    disp_muestral(X) / nrow(X)
}

cuasicov_muestral <- function(X) {
    disp_muestral(X) / (nrow(X)-1)
}

cor_muestral <- function(X) {
    cor(X)
}
```

Ejemplo con la X anterior:

```r
print(disp_muestral(X))
print(cov_muestral(X))
print(cuasicov_muestral(X))
print(cor_muestral(X))
```

```
          [,1]      [,2]      [,3]
[1,] 26.613912  4.367076 22.593071
[2,]  4.367076 17.626476 -6.097261
[3,] 22.593071 -6.097261 39.876757
          [,1]       [,2]       [,3]
[1,] 1.3306956  0.2183538  1.1296535
[2,] 0.2183538  0.8813238 -0.3048631
[3,] 1.1296535 -0.3048631  1.9938379
          [,1]       [,2]       [,3]
[1,] 1.4007322  0.2298461  1.1891090
[2,] 0.2298461  0.9277093 -0.3209085
[3,] 1.1891090 -0.3209085  2.0987767
          [,1]       [,2]       [,3]
[1,] 1.0000000  0.2016293  0.6935232
[2,] 0.2016293  1.0000000 -0.2299811
[3,] 0.6935232 -0.2299811  1.0000000
```

Contraste sobre $\boldsymbol{\mu}$

Sobre $\boldsymbol{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$, $\Sigma > 0$ y $\{\boldsymbol{X}_\alpha : \alpha = 1, \ldots, N\}$, con $N > p$ una m.a.s de $\boldsymbol{X}$ nos planteamos el problema de contrastre

$$\begin{cases} H_0 : \boldsymbol{\mu} = \boldsymbol{\mu}_0 \\ H_1 : \boldsymbol{\mu} \neq \boldsymbol{\mu}_0 \end{cases} , \boldsymbol{\mu}_0 \in \mathbb{R}^p \text{ dado.}$$

Σ conocida Para Σ conocida usaremos el estadístico de Wishart $W = N(\bar{\boldsymbol{X}} - \boldsymbol{\mu}_0)^T \Sigma^{-1} (\bar{\boldsymbol{X}} - \boldsymbol{\mu}_0)$ sigue una $\chi^2_p(\delta)$ con $\delta = N(\boldsymbol{\mu} - \boldsymbol{\mu}_0)^T \Sigma^{-1} (\boldsymbol{\mu} - \boldsymbol{\mu}_0)$, y la función test para el problema es:

$$\Phi(X) = \begin{cases} 1 \text{ si } W > \chi^2_{p;\alpha} \\ 0 \text{ si } W \leq \chi^2_{p;\alpha} \end{cases} ,$$

donde $\chi^2_{p;\alpha}$ representa el valor de una distribución χ^2_p que deja a su derecha una probabilidad α.

En nuestro caso si no proporcionamos la probabilidad α nos devolverá el p-value, que es la probabilidad que deja W a su derecha.

```r
media_test_sigma <- function(X, media_0, cov, alpha = NA) {
    if (!is.na(alpha) && (alpha > 1 || alpha < 0))
```

```
        stop("Error, 0 <= alpha <= 1")
    N <- nrow(X)
    p <- ncol(X)
    media <- media_muestral(X)
    W <- N * (t(media - media_0) %*% solve(cov) %*% (media - media_0))
    p_value <- 1 - pchisq(W, p)
    if (!is.na(alpha))
        cat("\nResultado del test: ", as.logical(p_value < alpha))
    else
        cat("\np-value: ", p_value)
}
```

Probamos el test con el X anterior, y vemos como el p-value se hace más grande (cuanto mayor valor más pequeño es W y por tanto tenemos menos significancia estadística de rechazar la hipótesis nula) conforme nos acercamos a la media muestral de X.

En el caso de poner una media cualquiera con significancia $\alpha = 0.05$, vemos que el test nos da TRUE (rechazamos hipótesis nula):

```
print(X_DNM$media)
print(media_muestral(X))
media_test_sigma(X, X_DNM$media, X_DNM$cov)
media_test_sigma(X, media_muestral(X), X_DNM$cov)
media_test_sigma(X, c(2.5, 3.5, -1.2), X_DNM$cov)
media_test_sigma(X, X_DNM$media, X_DNM$cov, alpha = 0.05)
```

```
     [,1]
[1,]    2
[2,]    3
[3,]   -1
           [,1]
[1,]  2.0124724
[2,]  2.9600257
[3,] -0.8978561

p-value:  0.9830968
p-value:  1
p-value:  0.0004866499
Resultado del test:  FALSE
```

Σ desconocida En el caso de que no sepamos quien es Σ usaremos el estadístico de T^2 de Hotelling, con $T^2 = N(\bar{\boldsymbol{X}} - \boldsymbol{\mu}_0)^T S_{N-1}^{-1}(\bar{\boldsymbol{X}} - \boldsymbol{\mu}_0)$, donde $\frac{T^2}{N-1}\frac{N-p}{p} \sim F_{p;N-p}(\delta)$ con $\delta = N(\boldsymbol{\mu} - \boldsymbol{\mu}_0)^T \Sigma^{-1}(\boldsymbol{\mu} - \boldsymbol{\mu}_0)$, y entonces la función test para el problema es:

$$\Phi(X) = \begin{cases} 1 \text{ si } (N-p)T^2 > (N-1)pF_{p;N-p;\alpha} \\ 0 \text{ si } (N-p)T^2 \leq (N-1)pF_{p;N-p;\alpha} \end{cases},$$

donde $F_{p;N-p;\alpha}$ representa el valor de una distribución $F_{p;N-p}$ que deja a su derecha una probabilidad α.

```
media_test <- function(X, media_0, alpha = NA) {
    if (!is.na(alpha) && (alpha > 1 || alpha < 0))
        stop("Error, 0 <= alpha <= 1")
    N <- nrow(X)
    p <- ncol(X)
    media <- media_muestral(X)
    S <- cuasicov_muestral(X)
    T <- N * (t(media - media_0) %*% solve(S) %*% (media - media_0))
    T_val <- T * (N - p) / (p * (N - 1))
    p_value <- 1 - pf(T_val, p, N - p)
    if (!is.na(alpha))
        cat("\nResultado del test: ", as.logical(p_value < alpha))
    else
        cat("\np-value: ", p_value)
}
```

Probamos con los ejemplos anteriores, y vemos que obtenemos unos resultados parecidos:

```
print(X_DNM$media)
print(media_muestral(X))
media_test(X, X_DNM$media)
media_test(X, media_muestral(X))
media_test(X, c(2.1, 3.1, -1.2))
media_test(X, c(3, 0, 0), alpha = 0.05)
```

```
     [,1]
[1,]    2
[2,]    3
[3,]   -1
           [,1]
[1,]  2.0124724
[2,]  2.9600257
[3,] -0.8978561

p-value:  0.9861966
p-value:  1
p-value:  0.4934115
Resultado del test:  TRUE
```

Superficies de confianza

Σ conocida Para formar las regiones de confianza para el vector de medias $\boldsymbol{\mu}$ consideramos que:

$$P[W \leq \chi^2_{p;\alpha}] = 1 - \alpha,$$

donde W era el estadístico de Wishart definido en el test de contraste.

Tenemos entonces que la región de confianza al $100(1-\alpha)\%$ del vector de medias $\boldsymbol{\mu}$ está definida por todos los $\boldsymbol{\mu}_0 \in \mathbb{R}^p$ tales que cumplen:

$$N(\bar{\boldsymbol{X}} - \boldsymbol{\mu}_0)^T \Sigma^{-1} (\bar{\boldsymbol{X}} - \boldsymbol{\mu}_0) \leq \chi^2_{p;\alpha}$$

Para representarla visualmente, tomamos dos variables X_i, X_j de $\boldsymbol{X}$ y representamos la región de confianza en $\mathbb{R}^2$ para esas dos variables, que toma la forma de una elipse. Usando la matriz de covarianza Σ y el radio $\sqrt{\frac{1}{N}\chi^2_{p;\alpha}}$, la representamos.

```
elipse_medias_sigma <- function(X, id, cov, alpha, col = "black", pch = 1, draw = FALSE) {
    p <- ncol(X)
    N <- nrow(X)
    media <- media_muestral(X)
    radio <- sqrt(qchisq(alpha,p) / N)
    elipse <- ellipse(center = media[id], shape = cov[id,id], radius = radio,
      draw = FALSE, pch = pch, col = col)
    if (!draw)
        plot(elipse, pch = pch, col = col)
    else
        points(elipse, pch = pch, col = col)
    points(matrix(media[id], ncol = 2), pch = 2, col = "red")
}
```

Veamos el ejemplo con X, trazando 3 regiones de confianza con $\alpha \in \{0.99, 0.95, 0.90\}$. Representamos todos los puntos de las variables X_1 y X_3, ponemos la media muestral con el triángulo rojo y la media de $\boldsymbol{X}$ con la cruz azul; finalmente representamos las 3 elipses de confianza.

```
plot(X[, c(1,3)], xlab = "1", ylab = "3")
elipse_medias_sigma (X, c(1,3), X_DNM$cov, 0.99, col = "orange", draw = TRUE)
points(matrix(X_DNM$media[c(1,3)], ncol=2), pch = 1, col = "blue")
elipse_medias_sigma (X, c(1,3), X_DNM$cov, 0.90, col = "green", pch = 1, draw = TRUE)
elipse_medias_sigma (X, c(1,3), X_DNM$cov, 0.95, col = "yellow", pch = 1, draw = TRUE)
legend("bottomright", legend = c("0.99", "0.95", "0.90"), pch = c(1, 1, 1),
  col = c("orange", "yellow", "green"), cex = 2.0)
```

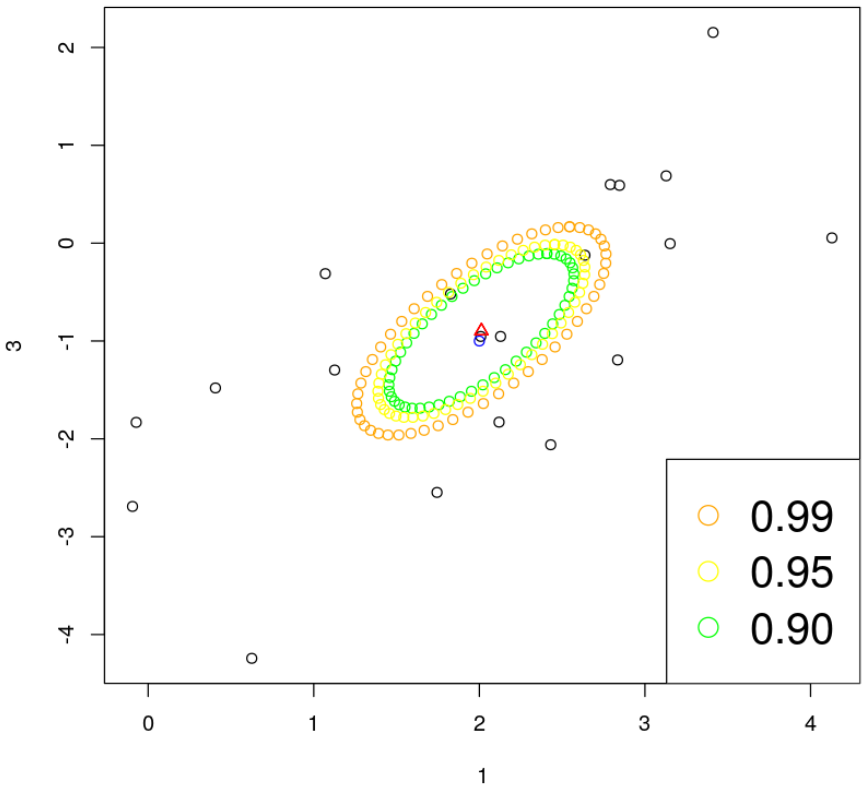

Figure 11: Superficie de confianza

Σ **desconocida** Para formar las regiones de confianza para el vector de medias $\boldsymbol{\mu}$ consideramos que:

$$P\left[T^2 \leq \frac{p(N-1)}{N-p} F_{p;N-p;\alpha}\right] = 1-\alpha,$$

donde T^2 era el estadístico de Hotelling definido en el test de contraste.

Tenemos entonces que la región de confianza al $100(1-\alpha)\%$ del vector de medias $\boldsymbol{\mu}$ está definida por todos los $\boldsymbol{\mu}_0 \in \mathbb{R}^p$ tales que cumplen:

$$N(\bar{\boldsymbol{X}} - \boldsymbol{\mu}_0)^T S_{N-1}^{-1} (\bar{\boldsymbol{X}} - \boldsymbol{\mu}_0) \leq \frac{p(N-1)}{N-p} F_{p;N-p;\alpha}$$

Para representarla visualmente, tomamos dos variables X_i, X_j de $\boldsymbol{X}$ y representamos la región de confianza en $\mathbb{R}^2$ para esas dos variables, que toma la forma de una elipse. Usando la matriz de covarianza S_{N-1} y el radio $\sqrt{\frac{p(N-1)}{N(N-p)} F_{p;N-p;\alpha}}$, la representamos.

```
elipse_medias <- function(X, id, alpha, col = "black", pch = 1, draw = FALSE) {
    p <- ncol(X)
    N <- nrow(X)
    media <- media_muestral(X)
    S <- cuasicov_muestral(X)
    radio <- sqrt(p*(N-1)*qf(alpha,p,N-p)/(N*(N-p)))
    elipse <- ellipse(center = media[id], shape = S[id,id], radius = radio,
      draw = FALSE, pch = pch, col = col)
    if (!draw)
        plot(elipse, pch = pch, col = col)
    else
        points(elipse, pch = pch, col = col)
    points(matrix(media[id], ncol = 2), pch = 2, col = "red")
}
```

Veamos el ejemplo con X, trazando 3 regiones de confianza con $\alpha \in \{0.99, 0.95, 0.90\}$. Representamos todos los puntos de las variables X_1 y X_3, ponemos la media muestral con el triángulo rojo y la media de $\boldsymbol{X}$ con la cruz azul; finalmente representamos las 3 elipses de confianza.

```
plot(X[, c(1,3)], xlab = "1", ylab = "3")
elipse_medias(X, c(1,3), 0.99, col = "orange", draw = TRUE)
points(matrix(X_DNM$media[c(1,3)], ncol=2), pch = 3, col = "blue")
elipse_medias(X, c(1,3), 0.90, col = "green", pch = 10, draw = TRUE)
elipse_medias(X, c(1,3), 0.95, col = "yellow", pch = 5, draw = TRUE)
legend("bottomright", legend = c("0.99", "0.95", "0.90"), pch = c(1, 1, 1),
  col = c("orange", "yellow", "green"), cex = 2.0)
```

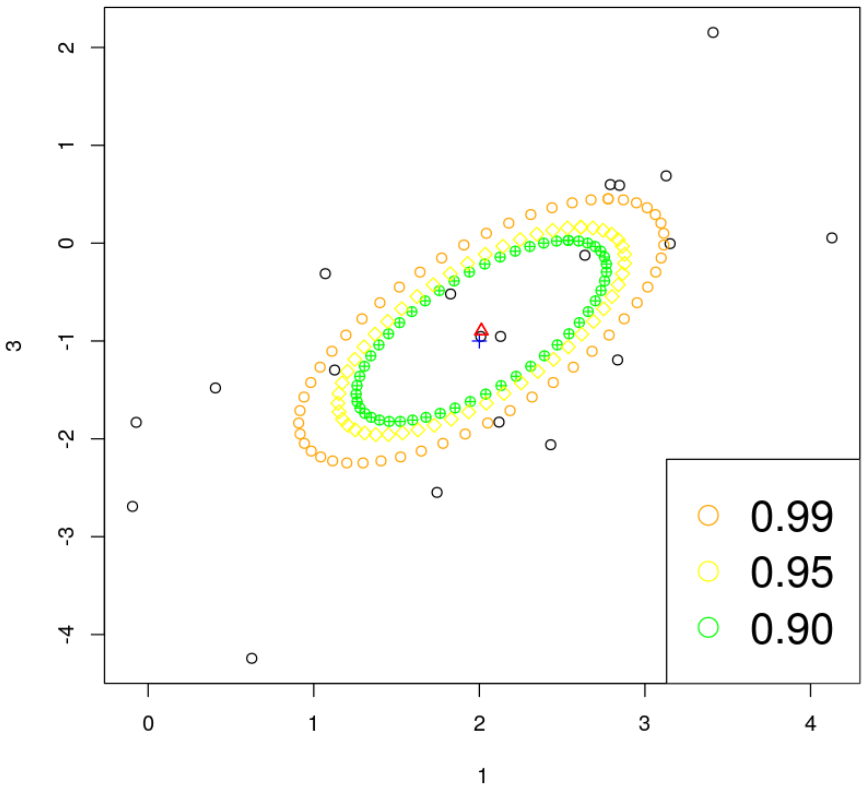

Figure 12: Superficie de confianza

Comparando las dos gráficas podemos ver que si conocemos Σ entonces las regiones de confianza son más pequeñas (tenemos más información acerca donde se encuentra el vector de medias de $\boldsymbol{X}$).