



UNIVERSIDAD
DE GRANADA

Departamento
de Estadística
e Investigación
Operativa

Grado En Estadística

Minería de datos

Tema 1: Preprocesamiento Estadístico



Javier Arnedo
arnedo@ugr.es
Curso 2022-2023

Tema 1

- **Introducción**
- Datos Anómalos
- Imputación Estadística
- Reducción de la Dimensionalidad
- Ingeniería de variables

Introducción

- Dentro de los problemas que podíamos resolver con técnicas estadísticas en el preprocesamiento de un conjunto de datos, tenemos:
- Datos ruidosos:
 - Individuos anómalos \Rightarrow métodos de eliminación de outliers.
 - Variables redundantes/irrelevantes \Rightarrow técnicas de reducción de dimensiones.
- Datos incompletos (valores faltantes) \Rightarrow técnicas de imputación estadística.
- Datos confusos (interpretación poco clara) \Rightarrow transformaciones de variables e ingeniería de variables (feature engineering).

Introducción

En un conjunto de datos, incluso aunque esté perfectamente estructurado, nos podemos encontrar con algunos individuos (instancias) dentro del total de n individuos que pueden ser problemáticos al aplicar técnicas de minería de datos:

- Individuos que toman datos demasiado anómalos (p. ej. contaminación en una muestra de laboratorio, persona que nos está engañando en una encuesta, etc.)
- Individuos que no aportan nada al análisis y que sólo aumentan el volumen de datos a tratar (p. ej. individuos que son muy parecidos a otros).

En esta sesión, nos centraremos en los individuos del primer tipo (anómalos u outliers).

Tema 1

- Introducción
- **Datos Anómalos**
- Imputación Estadística
- Reducción de la Dimensionalidad
- Transformaciones

Reducción de la Dimensionalidad

Introducción

Dentro de los problemas que podíamos resolver con técnicas estadísticas en el preprocesamiento de un conjunto de datos, teníamos:

- Datos ruidosos:
 - **Individuos anómalos** \Rightarrow **métodos de eliminación de outliers.**
 - Variables redundantes/irrelevantes \Rightarrow técnicas de reducción de dimensiones.
- Datos incompletos (valores faltantes) \Rightarrow técnicas de imputación estadística.
- Datos confusos (interpretación poco clara) \Rightarrow transformaciones de variables e ingeniería de variables (feature engineering).

Definición de datos anómalos

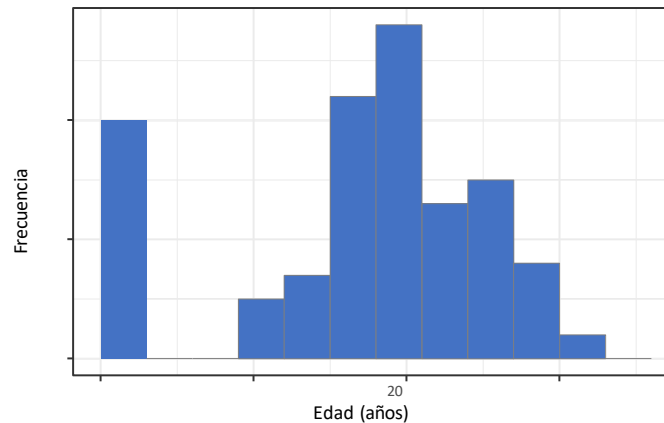
Los datos anómalos u outliers son aquellos individuos cuyo valor o valores son considerablemente distintos al del resto de los individuos. Se pueden dar por diversos mecanismos:

- Errores de codificación (p. ej. erratas).
- Verdaderas ocurrencias de sucesos muy poco probables.
- Individuos que no forman parte de nuestra población objetivo.

Definición de datos anómalos

Los errores de codificación deben de depurarse inspeccionando los datos, ya sea de forma automática o manual.

Ejemplo: un dataset con la variable 'edad' donde se asigna un 0 a quienes no responden

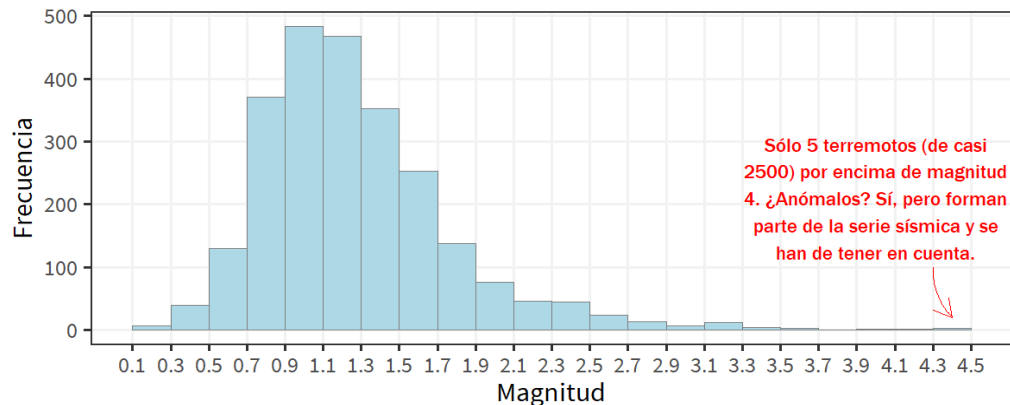


Definición de datos anómalos

Las ocurrencias de sucesos muy poco probables, aunque tengan una influencia demasiado relevante, se deben incluir en el análisis final de todas maneras ya que son datos reales.

Distribución de la magnitud de los terremotos en las proximidades de Granada (España) del 20 de enero al 7 de abril de 2021

Fuente: Catálogo de terremotos del Instituto Geográfico Nacional (IGN).



Longitud mínima: -3.9; Longitud máxima: -3.6. Latitud mínima: 37.15; Latitud máxima: 37.31

Definición de datos anómalos

- Sin embargo, si esos valores anómalos vienen de una población que no es la que queremos estudiar, debemos eliminar dichos individuos del análisis.
- Por ejemplo: realizamos una encuesta para conocer los hábitos de los estudiantes universitarios, pero nos encontramos con que hay cinco personas que han respondido que tienen menos de 15 años.

Detección de datos anómalos

En la detección de individuos anómalos, distinguimos dos posibles situaciones:

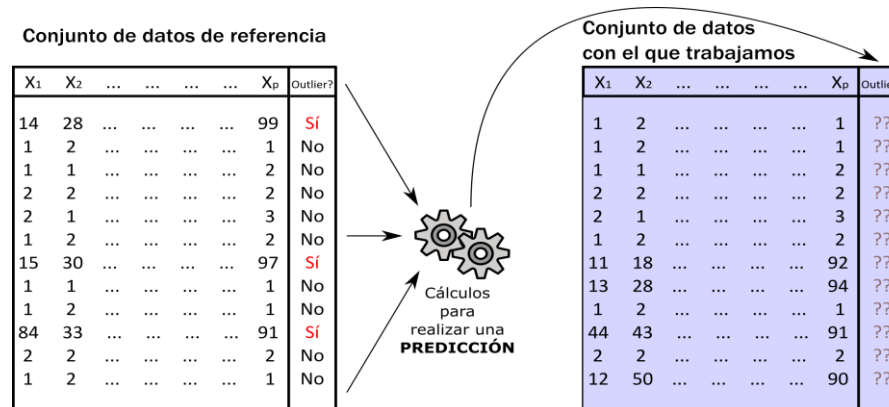
1. Conocemos qué características suelen tener los anómalos.
2. No conocemos las características de los anómalos.

En los ejemplos anteriores conocíamos las características de los datos anómalos, porque los fenómenos han quedado perfectamente descritos. Habitualmente, es común encontrarnos en la segunda situación.

Detección de datos anómalos

Caso 1: conocemos qué características suelen tener los anómalos.

En estos casos, podemos utilizar modelos de regresión logística o clasificación binaria que usen los datos de referencia para proporcionarnos predicciones sobre qué individuos de un dataset con el que vamos a trabajar son anómalos.



Detección de datos anómalos

Caso 2: no conocemos las características de los anómalos.

- Este es el caso más habitual: sabemos que puede haber datos anómalos en nuestro conjunto, pero no tenemos ningún otro conjunto de referencia para hacer el reconocimiento.
- En estos casos, determinaremos que un caso es anómalo según su comportamiento en comparación con el resto de datos.

Detección de datos anómalos

Los métodos estadísticos disponibles para la detección de datos anómalos distinguen entre el caso unidimensional y multidimensional.

- Caso unidimensional
 - Criterio de Tukey
 - Test de Grubbs
 - Test de Tietjen-Moore
 - Test de Rosner
- Caso multidimensional
 - Distancia de Mahalanobis
 - Análisis gráfico

Caso Unidimensional: Criterio de Tukey

Criterio de Tukey para la detección de datos anómalos

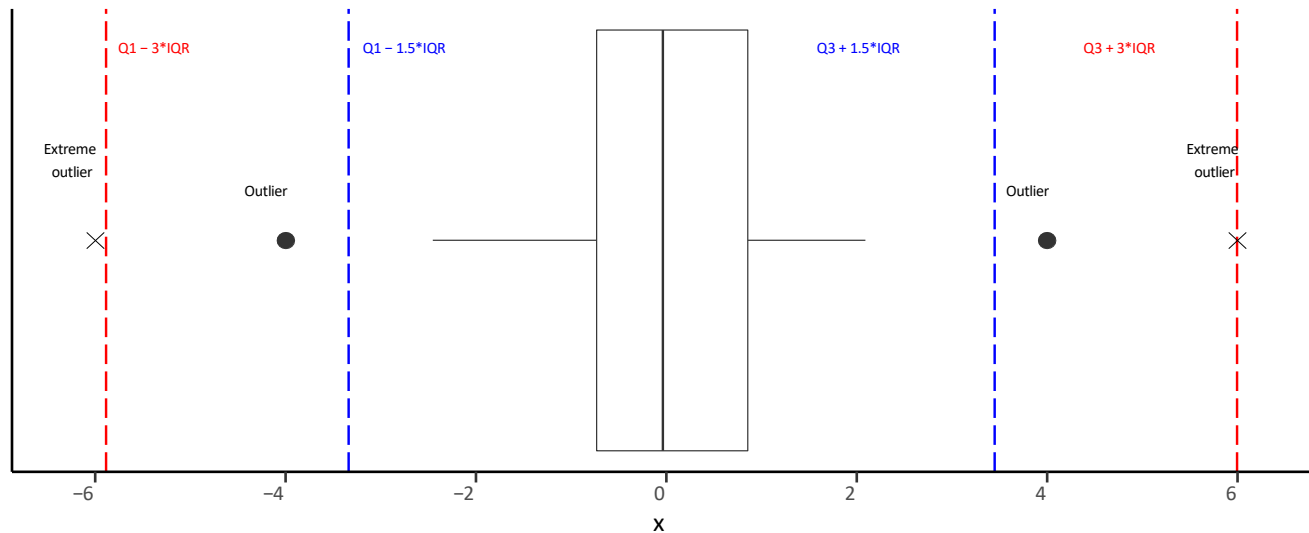
Sea $X = (x_1, \dots, x_n)$ una variable de un conjunto de datos con n individuos, a partir de la cual queremos detectar datos anómalos.

Tukey (1977), al desarrollar el gráfico de caja y bigotes (*boxplot*), hizo las siguientes delimitaciones en función del valor de la variable para un cierto individuo:

- Si $x_i > Q3 + 1.5 \cdot IQR$ ó $x_i < Q1 - 1.5 \cdot IQR \Rightarrow$ el individuo i -ésimo es anómalo (*outlier*).
- Si $x_i > Q3 + 3 \cdot IQR$ ó $x_i < Q1 - 3 \cdot IQR \Rightarrow$ el individuo i -ésimo es extremadamente anómalo (*extreme outlier*).

Caso Unidimensional: Criterio de Tukey

Ejemplo: 50 datos procedentes de una $N(0, 1)$, y cuatro datos puestos artificialmente: $-6, -4, 4, 6$.



Caso Unidimensional: Criterio de Tukey

Ventajas:

- Es computacionalmente simple.
- No requiere de asunciones previas sobre la distribución de probabilidad de X .

Desventajas:

- Criterio exploratorio (no inferencial). Inadecuado cuando n es pequeño.
- En distribuciones asimétricas se vuelve más sensible.

Caso Unidimensional: Test de Grubbs

Test de Grubbs para datos anómalos unidimensionalmente

Sea $X = (x_1, \dots, x_n)$ una variable de un conjunto de datos con n individuos, para la que asumimos que $X \sim N(\mu_x, \sigma_x^2)$ siendo \bar{x} y s_x^2 la media y la cuasivarianza respectivamente de X en nuestro conjunto de datos. En el test de Grubbs, contrastamos las siguientes hipótesis:

- $H_0 : X$ no tiene ningún valor anómalo
- $H_1 : X$ tiene **exactamente un valor** anómalo

El estadístico de prueba, G , es la máxima desviación absoluta estandarizada respecto a la media:

$$G = \frac{\max_{i=1, \dots, n} |x_i - \bar{x}|}{s_x}$$

Caso Unidimensional: Test de Grubbs

La región de rechazo para el estadístico de prueba sería:

$$G > \frac{(n-1)}{\sqrt{n}} \sqrt{\frac{t_{\alpha/2n;n-2}^2}{n-2 + t_{\alpha/2n;n-2}^2}},$$

donde $t_{\alpha/2n;n-2}$ es el cuantil $\alpha/2n$ de una t de Student con $n - 2$ grados de Libertad.

El test tiene también las dos versiones a una cola, para comprobar si el valor mínimo o el máximo de nuestra muestra es un outlier:

$$G = \frac{x_{max} - \bar{X}}{s_X}, \quad G = \frac{\bar{X} - x_{min}}{s_X}$$

Y la región de rechazo sería la misma, pero tomando el cuantil α/n

Caso Unidimensional: Test de Grubbs

Ventaja:

- Criterio inferencial.

Desventajas:

- Se requiere normalidad de los datos.
- Impráctico cuando hay más de un dato anómalo (iterar no es útil).

Caso Unidimensional: Test de Tietjen-Moore

Test de Tietjen-Moore para datos anómalos unidimensionalmente

Sea $X = (x_1, \dots, x_n)$ una variable de un conjunto de datos con n individuos, para la que asumimos que $X \sim N(\mu_x, \sigma_x^2)$ siendo \bar{x} y s_x^2 la media y la cuasivarianza respectivamente de X en nuestro conjunto de datos. En el test de Tietjen-Moore, contrastamos las siguientes hipótesis:

- $H_0 : X$ no tiene ningún valor anómalo
- $H_1 : X$ tiene **exactamente k valores** anómalo

Caso Unidimensional: Test de Tietjen-Moore

Para realizar el test, primero tenemos que definir una nueva variable Z tal que:

Ordenamos la nueva variable de menor a mayor, tal que $z_{(1)}$ es el valor de X con menor desviación con respecto a \bar{x} , y $z_{(n)}$ el valor con mayor desviación. El estadístico de prueba, E_k , sería:

$$E_k = \frac{\sum_{i=1}^{n-k} (z_{(i)} - \bar{z}_k)^2}{\sum_{i=1}^n (z_{(i)} - \bar{z})^2},$$

donde \bar{z} es la media de la nueva variable Z , y \bar{z}_k la media de la nueva variable sin los k valores que presentan la mayor desviación, es decir, sin

$z_{(n-k+1)}, \dots, z_{(n)}$.

Caso Unidimensional: Test de Tietjen-Moore

El estadístico de prueba se computa para una población Normal estándar simulada, y se divide entre el estadístico de prueba de nuestra muestra. Si el cociente es ≈ 1 , no hay outliers. Si está cerca de 0, hay exactamente k outliers.

Ventajas:

- Criterio inferencial.

Desventajas:

- Se requiere normalidad de los datos.
- “Swamping”: el test puede declarar que hay k outliers cuando en realidad hay menos.

Caso Unidimensional: Test de Rosner

Test de Rosner para datos anómalos unidimensionalmente

Sea $X = (x_1, \dots, x_n)$ una variable de un conjunto de datos con n individuos, para la que asumimos que $X \sim N(\mu_x, \sigma_x^2)$ siendo \bar{x} y s_x^2 la media y la cuasivarianza respectivamente de X en nuestro conjunto de datos. En el test de Tietjen-Moore, contrastamos las siguientes hipótesis:

- $H_0 : X$ no tiene ningún valor anómalo
- $H_1 : X$ tiene **hasta k valores** anómalo

Caso Unidimensional: Test de Rosner

El estadístico de prueba se calcula de manera iterativa.

1. Calcular

$$R_i = \frac{\max_j |x_j - \bar{x}|}{s_X}$$

2. Eliminar la observación i -ésima, que maximiza la desviación anterior.
3. Volver al paso 1.
4. Repetir hasta que se hayan eliminado k observaciones. Habremos obtenido k estadísticos de prueba: R_1, R_2, \dots, R_k .

Caso Unidimensional: Test de Rosner

Para computar los valores críticos, calculamos

$$\lambda_i = \frac{(n-i)t_{p;n-i-1}}{\sqrt{(n-i-1+t_{p;n-i-1}^2)(n-i+1)}}, \quad i = 1, 2, \dots, k,$$

donde $t_{p;n-i-1}$ es el cuantil p de la t de Student con $n-i-1$ grados de libertad, y

$$p = 1 - \frac{\alpha}{2(n-i+1)}$$

El número de outliers queda determinado encontrando el mayor valor de i tal que $R_i > \lambda_i$.

Caso Unidimensional: Test de Rosner

Ventajas:

- Criterio inferencial.
- Flexible ante número variante de outliers.

Desventajas:

- Se requiere normalidad de los datos.

Caso Multidimensional

La detección de outliers con varias dimensiones es más compleja, ya que su aparición puede obedecer a dos factores:

- El individuo tiene un dato muy atípico en una variable concreta.
- El individuo presenta una combinación de valores en las variables que, a pesar de que no son atípicos individualmente, sí que es atípica en su conjunto.

Caso Multidimensional

Sea $X_{n \times p} = (x_1, \dots, x_n)$ una matriz de p variables de un conjunto de datos con n individuos, las cuales asumimos que se distribuyen según una Normal multivariante, es decir, $X \sim N(\mu, \Sigma)$.

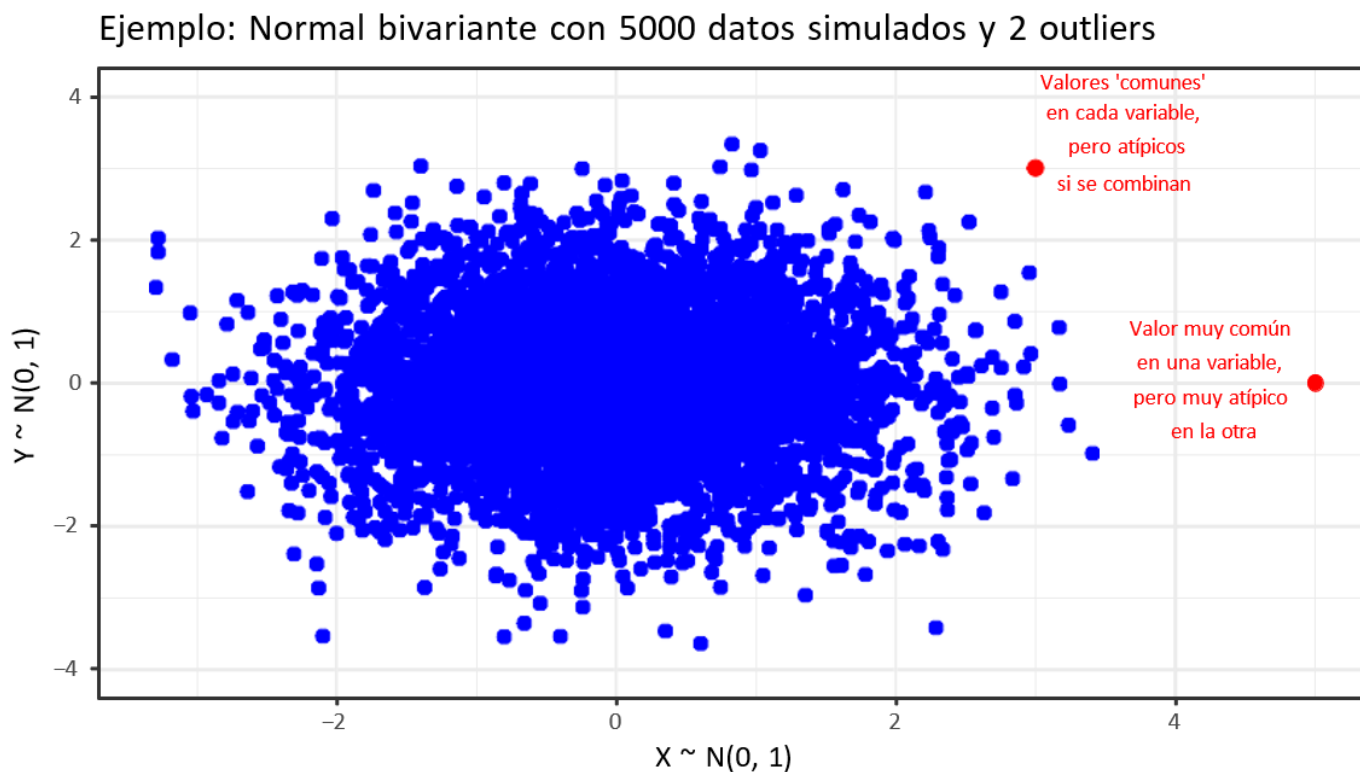
Definimos la matriz de covarianzas muestral:

$$\mathbf{S} = \begin{pmatrix} s_1^2 & s_{12} & \dots & s_{1p} \\ s_{21} & s_2^2 & \dots & s_{2p} \\ \dots & \dots & \dots & \dots \\ s_{p1} & s_{p2} & \dots & s_p^2 \end{pmatrix},$$

donde

$$s_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{n - 1}$$

Caso Multidimensional



Caso Multidimensional: Distancia de Mahalanobis

Distancia de Mahalanobis

Estrategia: medir la distancia de cada punto con respecto al centroide de la Normal multivariante (μ) y ver cuáles se alejan más. . . y si se alejan de una forma demasiado atípica.

La distancia de Mahalanobis para el i -ésimo individuo de una muestra se calcula con la siguiente fórmula:

$$d(\mathbf{x}_i) = \sqrt{(\mathbf{x}_i - \bar{\mathbf{x}})^T \mathbf{S} (\mathbf{x}_i - \bar{\mathbf{x}})}$$

$$d^2(\mathbf{x}_i) = (\mathbf{x}_i - \bar{\mathbf{x}})^T \mathbf{S} (\mathbf{x}_i - \bar{\mathbf{x}})$$

Caso Multidimensional: Distancia de Mahalanobis

La distribución teórica (utilizando μ y Σ en lugar de \bar{x} y S) de la distancia de Mahalanobis es la Chi-Cuadrado con p grados de libertad.

Para el caso de una muestra:

$$d^2(x)_i \approx X_p^2 \quad \frac{np}{n-p} d^2(x_i) \approx F_{p,n-p}$$

Caso Multidimensional: Distancia de Mahalanobis

Por lo tanto, se puede establecer un criterio para etiquetar a un individuo como dato anómalo en función de su valor para la distancia de Mahalanobis y las distribuciones anteriormente descritas:

Si $d^2(x_i) > \chi_{p;1-\alpha_n}^2 \Rightarrow$ el individuo i -ésimo es un outlier.

Si $\left(\frac{np}{n-p}\right) d^2(x_i) > F_{p,n-p;1-\alpha_n} \Rightarrow$ el individuo i -ésimo es un outlier.

α_n representaría el nivel de significación **corregido para comparaciones múltiples**, considerando n comparaciones (ya que aplicamos un contraste por cada individuo de nuestro conjunto).

Caso Multidimensional: Distancia de Mahalanobis

Hay varias correcciones para comparaciones múltiples que podemos utilizar. Dos de ellas son:

- Corrección de Bonferroni: $\alpha_n = \frac{\alpha}{n}$
- Corrección de Šidák: $\alpha_n = (1 - \alpha)^n$

Habitualmente, la corrección de Bonferroni nos devolverá un α_n más bajo que el de la corrección de Šidák, es decir, **la corrección de Bonferroni será más conservadora.**

The Šidák method assumes that each comparison is independent of the others. If this assumption is independence cannot be supported, choose the Bonferroni method, which does not assume independence.

Caso Multidimensional: Distancia de Mahalanobis

Hay varias correcciones para comparaciones múltiples que podemos utilizar. Dos de ellas son:

- Corrección de Bonferroni: $\alpha_n = \frac{\alpha}{n}$
- Corrección de Šidák: $\alpha_n = (1 - \alpha)^n$

Habitualmente, la corrección de Bonferroni nos devolverá un α_n más bajo que el de la corrección de Šidák, es decir, **la corrección de Bonferroni será más conservadora.**

Caso Multidimensional: Distancia de Mahalanobis

- Cuando existen outliers, el centroide de una distribución multivariante **estará artificialmente desplazado** debido a la falta de suficiencia de la media en estos casos (p. ej. salario medio, mediano y modal).
- Para solucionarlo, se puede calcular la distancia de Mahalanobis pero considerando el medoide o la media truncada.

Caso Multidimensional: Análisis gráfico

Análisis gráfico

- Como alternativa al método de la distancia de Mahalanobis (más paramétrico e inferencial), se puede considerar la localización de outliers a través de la **visualización de datos bidimensional**.
- Se reduce el dataset $X_{n \times p}$ a una proyección $X'_{n \times 2}$, de forma que se pueda visualizar en dos ejes, y sobre ellos se localizan los puntos más alejados al centroide.
- Tiene la ventaja de no asumir distribuciones de partida, pero la desventaja de ser subjetivo y depender de la bondad del ajuste de X' .

Ejemplo práctico con R

Vamos a trabajar con la base de datos `airquality`, disponible por defecto en R (paquete `datasets`).

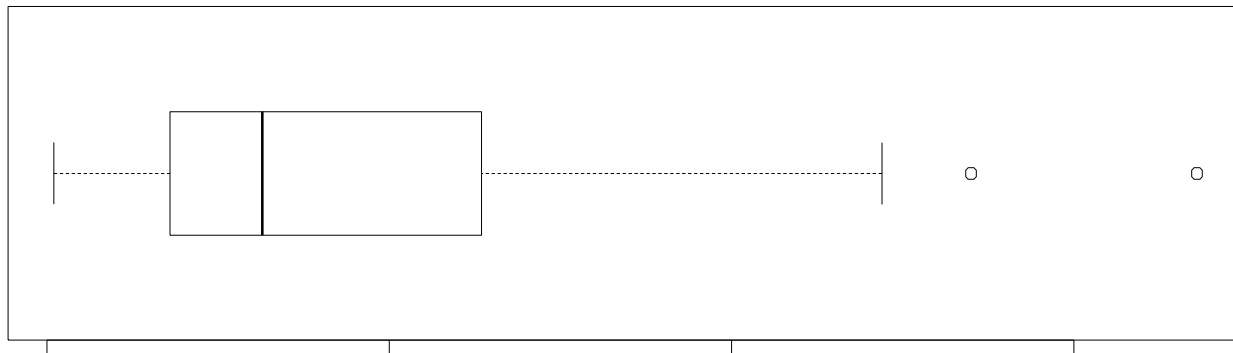
```
summary(airquality)
```

```
##      Ozone      Solar.R      Wind      Temp
##  Min.   : 1.00    Min.   : 7.0    Min.   : 1.700    Min.   :56.00
## 1st Qu.: 18.00    1st Qu.:115.8    1st Qu.: 7.400    1st Qu.:72.00
## Median : 31.50    Median :205.0    Median : 9.700    Median :79.00
## Mean   : 42.13    Mean   :185.9    Mean   : 9.958    Mean   :77.88
## 3rd Qu.: 63.25    3rd Qu.:258.8    3rd Qu.:11.500    3rd Qu.:85.00
## Max.   :168.00    Max.   :334.0    Max.   :20.700    Max.   :97.00
##      NA 's :37    NA 's :7
##      Month      Day
##  Min.   :5.000    Min.   : 1.0
## 1st Qu.:6.000    1st Qu.: 8.0
## Median :7.000    Median :16.0
## Mean   :6.993    Mean   :15.8
## 3rd Qu.:8.000    3rd Qu.:23.0
## Max.   :9.000    Max.   :31.0
##
```

Ejemplo práctico con R

La variable `ozone` refleja la cantidad media de ozono (partes por millón) entre las 13:00 y las 15:00 horas en un día concreto. La examinamos para ver si hay algún valor atípico a través del gráfico de caja y bigotes (función `boxplot()` en R).

```
boxplot(airquality$Ozone, horizontal = T)
```



Ejemplo práctico con R

En el boxplot se aprecian dos valores atípicos por la derecha. Podemos ver qué valores toman, combinando las funciones `sort()` (ordena valores de menor a mayor) y `tail()` (devuelve los últimos valores del vector).

```
tail(sort(airquality$Ozone))
```

```
## [1] 110 115 118 122 135 168
```

Vemos que los valores que toman son 135 y 168. Para saber qué individuos son, utilizamos la función `which()`

```
#Devuelve los índices de quienes tomen  
#un valor en Ozone mayor o igual a 135  
which(airquality$Ozone >= 135)
```

```
## [1] 62 117
```


Ejemplo práctico con R

¿Deberíamos eliminar esos dos individuos? Siguiendo el criterio de Tukey, habría que ver si están por encima de $Q3 + 1.5 \cdot IQR$ o de $Q3 + 1.5 \cdot IQR$

```
Q3 <- quantile(airquality$Ozone, 0.75, na.rm = T)
Q3 + 1.5 * IQR(airquality$Ozone, na.rm = T)
```

```
##      75%
## 131.125
```

```
Q3 + 3 * IQR(airquality$Ozone, na.rm = T)
```

```
## 75%
## 199
```

Según dicho criterio, si descartásemos los valores **anómalos** (a partir de 131.125) habría que descartar ambos. Si sólo descartásemos los valores **extremadamente anómalos** (a partir de 199), no descartaríamos ninguno.

Ejemplo práctico con R

Podemos hacer el test de Grubbs mediante la función `grubbs.test()` del paquete `{outliers}`:

```
library(outliers)
grubbs.test(airquality$Ozone)

##
##  Grubbs test for one outlier
##
## data:  airquality$Ozone
## G = 3.8157, U = 0.8723, p-value = 0.004765
## alternative hypothesis: highest value 168 is an outlier
```

Ejemplo práctico con R

Podemos hacer el test de Rosner mediante la función `rosnerTest()` del paquete `{EnvStats}`, con el argumento K para especificar el número de outliers que vamos a considerar en la hipótesis nula (aquí asumiremos dos, ya que son dos los que hemos identificado en el boxplot).

```
library(EnvStats)
testrosner <- rosnerTest(airquality$Ozone, k = 2)
testrosner$all.stats
```

##	i	Mean.i	SD.i	Value	Obs.Num	R.i+1	lambda.i+1	Outlier	
##	1	0	42.12931	32.98788	168	117	3.815664	3.433961	TRUE
##	2	1	41.03478	30.94447	135	62	3.036575	3.431092	FALSE

Ejemplo práctico con R

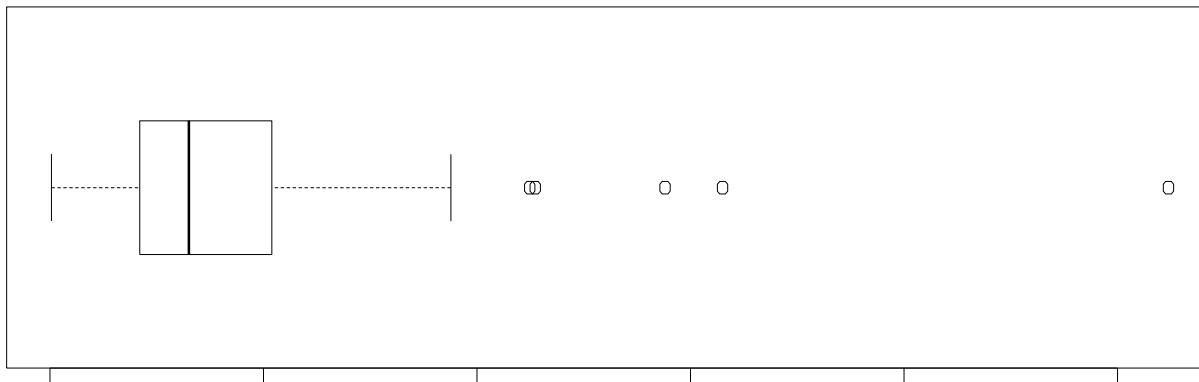
Para detectar outliers multidimensionales, podemos calcular la distancia de Mahalanobis **al cuadrado** ($d^2(x_i)$) utilizando la función `mahalanobis`. Hay que especificarle un vector de medias (parámetro `center`) y la matriz de covarianzas (parámetro `cov`).

La calculamos para las cuatro primeras columnas (ozono, radiación solar, viento y temperatura):

```
#Calculamos el vector de medias con colMeans()  
mu <- colMeans(airquality[,1:4], na.rm = T)  
  
#Calculamos la matriz de covarianzas con cov()  
sigma <- cov(airquality[,1:4], use = "p")
```

Ejemplo práctico con R

```
distancias <- mahalanobis(airquality[,1:4],  
                           center = mu, cov = sigma)  
#Sólo obtenemos las distancias de 111 puntos #(el resto tienen valores faltantes)  
boxplot(distancias, horizontal = T)
```



Ejemplo práctico con R

Si $d^2(x_i) > \chi^2_{p;1-\alpha_n} \Rightarrow$ el individuo i -ésimo es un outlier.

Hemos calculado la distancia con 4 variables, así que $p = 4$.

Para un nivel de confianza del 95% ($\alpha = 0.05$), usaremos la corrección de Bonferroni $\Rightarrow \alpha_n = \alpha/n \Rightarrow 0.05/111 = 0.00045$

```
qchisq(1 - 0.05/111, 4)
```

```
## [1] 20.22683
```

Todos aquellos días para los cuales $d^2(x_i) > 20.23$ serían considerados outliers.

Ejemplo práctico con R

```
which(distancias > qchisq(1 - 0.05/111, 4))
```

```
## [1] 117
```

Sólo el día número 117 sería considerado outlier. Veamos sus valores para las variables que hemos elegido:

```
airquality[117, 1:4]
```

```
##      Ozone Solar.R Wind Temp  
## 117   168     238  3.4   81
```

Tema 1

- Introducción
- Datos Anómalos
- **Imputación Estadística**
- Reducción de la Dimensionalidad
- Ingeniería de variables

Imputación de datos faltantes

Introducción

Dentro de los problemas que podíamos resolver con técnicas estadísticas en el preprocesamiento de un conjunto de datos, teníamos:

- Datos ruidosos:
 - Individuos anómalos \Rightarrow métodos de eliminación de outliers.
 - Variables redundantes/irrelevantes \Rightarrow técnicas de reducción de dimensiones.
- **Datos incompletos (valores faltantes) \Rightarrow técnicas de imputación estadística.**
- Datos confusos (interpretación poco clara) \Rightarrow transformaciones de variables e ingeniería de variables (feature engineering).

Imputación de datos faltantes

Introducción

- En un conjunto de datos, no todos los individuos tienen por qué presentar valores en una variable.
- Hay individuos cuyo valor de X es desconocido, lo cual puede responder a varios motivos:
 - Negativa a la hora de participar en el estudio.
 - Error en la recogida de datos.
 - No procede que el individuo tome valor en esa variable (blanco por flujo).
 - ...

Ozone	Solar.R	Wind	Temp	Month	Day
41	190	7.4	67	5	1
36	118	8.0	72	5	2
12	149	12.6	74	5	3
18	313	11.5	62	5	4
NA	NA	14.3	56	5	5
28	NA	14.9	66	5	6

Imputación de datos faltantes

Introducción

Cuando hay una variable en la cual un individuo no tiene un valor asociado decimos que hay un **dato faltante**. Los datos faltantes son problemáticos:

- Pueden causar estimaciones sesgadas.
- Imposibilitan el uso de algunos algoritmos de minería de datos.
- Provocan pérdida de información.

Imputación de datos faltantes

Introducción

En la literatura podemos encontrar dos tipos de falta de respuesta:

- Falta de respuesta parcial (*item non-response*).
- Falta de respuesta total (*unit non-response*).

En esta sesión nos centraremos en la falta de respuesta parcial: hay pérdida de datos en una o más variables, pero no en todas. En el segundo caso, son recomendables técnicas de reponderación de la muestra.

Imputación de datos faltantes

Mecanismos que provocan datos faltantes

Contamos con un dataset $\mathbf{X}_{n \times p}$ con n individuos y p variables.

Sea X una variable medida para los n individuos, en la cual hay datos perdidos.

Llamamos $\mathbf{X}'_{n \times p-1}$ al dataset sin la variable X .

Denotamos la variable indicadora $I(X)$ como:

$$I_i(X) = \begin{cases} 1 & \text{si la variable } X \text{ presenta dato para el individuo } i \\ 0 & \text{si la variable } X \text{ no presenta ningún dato para el individuo } i \end{cases}$$

Imputación de datos faltantes

Mecanismos que provocan datos faltantes

El mecanismo que provoca que haya un valor faltante en una variable X para un individuo i puede ser de tres tipos:

- Missing Completely At Random (MCAR)
- Missing At Random (MAR)
- Missing Not At Random (MNAR)

Imputación de datos faltantes

Mecanismos que provocan datos faltantes: MCAR

El caso Missing Completely At Random (MCAR) es el más simple de todos: en él, la presencia de datos faltantes en X **no depende de ninguna otra variable**.

Imputación de datos faltantes

Mecanismos que provocan datos faltantes: MCAR

Ejemplos de MCAR:

- Seleccionar mediante muestreo aleatorio simple, en un dataset con $n = 100$ filas, a 10 individuos y quitarles los valores que tienen asignados para una variable X .
- El dataset generado por las mediciones del peso de 100 personas a través de una báscula a la que de vez en cuando le falla la batería.

Imputación de datos faltantes

Mecanismos que provocan datos faltantes: MCAR

En el primer ejemplo, podemos ver que la pérdida de datos no afecta a la estimación (salvo por la pérdida de tamaño de muestra):

```
x <- rnorm(1000) #Generamos 1000 individuos de una N(0,1)  
mean(x)
```

```
## [1] 0.06199874
```

```
x[sample(1000, 100, replace = F)] <- NA  
mean(x, na.rm = T)
```

```
## [1] 0.06560919
```

La media de X con y sin datos perdidos es muy similar.

Imputación de datos faltantes

Mecanismos que provocan datos faltantes: MAR

En el caso Missing At Random (MAR), nos encontramos que la presencia de datos faltantes en X depende de otras variables presentes en el dataset, \mathbf{X} , pero **no** de la propia variable X .

Imputación de datos faltantes

Mecanismos que provocan datos faltantes: MAR

Ejemplos de MAR:

- Una variable X_1 que mida la intención de voto de una persona, donde las personas de mayor edad (variable X_2) sean más reacias a responder.
- El dataset generado por las mediciones del peso de 100 personas a través de una báscula cuya batería falla más frecuentemente al pesar a gente de mayor altura.

Imputación de datos faltantes

Mecanismos que provocan datos faltantes: MAR

Dependiendo de la relación de X_1 con X_2 , este mecanismo podrá provocar más o menos sesgo.

```
y <- rnorm(1000)
x <- y - runif(1000)
mean(x)
```

```
## [1] -0.4934005
```

```
x[which(y > 0.5)] <- NA
mean(x, na.rm = T)
```

```
## [1] -0.9982248
```

Imputación de datos faltantes

Mecanismos que provocan datos faltantes: MNAR

En el caso Missing Not At Random (MNAR), nos encontramos que la presencia de datos faltantes en X depende **directamente de la propia variable**. Este es el caso más problemático de todos.

Imputación de datos faltantes

Mecanismos que provocan datos faltantes: MNAR

Ejemplos de MNAR:

- Una variable X_1 que mida la intención de voto de una persona, donde las personas que quieran votar a un cierto partido prefieran no responder.
- El dataset generado por las mediciones del peso de 100 personas a través de una báscula cuya batería falla más frecuentemente con la gente que más pesa.

Imputación de datos faltantes

Mecanismos que provocan datos faltantes: MNAR

Si los datos faltantes siguen este patrón, se obtendrán estimaciones sesgadas difíciles de ajustar:

```
x <- rnorm(100, 1.80, 0.10) #100 alturas simuladas  
mean(x)
```

```
## [1] 1.797239
```

```
x[which(x > 1.80)] <- NA  
mean(x, na.rm = T)
```

```
## [1] 1.718834
```

Imputación de datos faltantes

Test de Little para datos perdidos

Little (1986) desarrolló un contraste de hipótesis para poder investigar si los datos perdidos se generan con un mecanismo MCAR o no:

- H_0 : los datos perdidos siguen un mecanismo MCAR
- H_1 : los datos perdidos no siguen un mecanismo MCAR

El estadístico de prueba, basado en la razón de verosimilitudes, sigue una Chi-Cuadrado si H_0 es cierta (para más detalles, ver el artículo original: <https://doi.org/10.1080/01621459.1988.10478722>)

Imputación de datos faltantes

Test de Little para datos perdidos

Dicho test se puede aplicar en R con la función `mcar_test` del paquete `{naniar}`:

```
install.packages("naniar")  
library(naniar)  
mcar_test(data)
```

Sólo requiere un argumento: el conjunto de datos a evaluar, en formato `data.frame`.

Imputación de datos faltantes

Test de Little para datos perdidos

Vamos a probar la función con un dataset con dos variables: x (Normal) e y (uniforme), con datos perdidos asignados al azar (MCAR):

```
x <- rnorm(1000)
x[sample(1000, 100, replace = F)] <- NA
y <- runif(1000)
y[sample(1000, 100, replace = F)] <- NA

prueba1 <- data.frame(x, y)
```

Imputación de datos faltantes

Test de Little para datos perdidos

```
mcar_test(prueba1)
```

```
## # A tibble: 1 x 4
## statistic    df p.value missing.patterns
##      <dbl> <dbl>    <dbl>          <int>
## 1    0.621     2    0.733            4
```

Los resultados muestran un p-valor muy por encima de cualquier α (95%, 90%...), por tanto, no podemos rechazar la hipótesis de que los datos perdidos sigan un mecanismo MCAR.

Imputación de datos faltantes

Técnicas para trabajar con datos perdidos

Las técnicas que podemos aplicar para solucionar la presencia de datos perdidos se pueden resumir en las siguientes:

- Eliminación de individuos/variables.
- Imputación de datos faltantes.
- Modelización de datos faltantes.

Imputación de datos faltantes

Eliminación de individuos/variables

En este procedimiento, se eliminan los individuos que tengan datos faltantes en nuestra variable de interés X .

Si se quiere trabajar con todo el dataset, dos posibilidades serían:

- **Eliminar los individuos** con datos perdidos en muchas variables.
- **Eliminar las variables** con datos perdidos en muchos individuos.

Imputación de datos faltantes

Eliminación de individuos/variables

Ejemplo de eliminación de variables: dataset selfreport

En la librería mice, tenemos disponible el dataset selfreport, que contiene datos sobre altura y peso de 2060 personas, con 15 variables:

- src: estudio del que proceden los datos.
- id: número de identificación de la persona.
- pop: población (todos los valores son NL).
- age, sex: edad y sexo.
- hm, hr: altura medida y reportada, respectivamente.
- wm, wr: peso medido y reportado, respectivamente.
- bm, br: IMC medido y reportado, respectivamente.
- prg: estado de embarazo (todos los valores son Not pregnant).
- edu: nivel educativo.
- etn: etnia.
- web: respuesta obtenida a través de encuesta online (sí/no).

Imputación de datos faltantes

Eliminación de individuos/variables

Vamos a analizar qué porcentaje de datos perdidos tiene cada variable. Para ello, primero diseñamos una función que saque la proporción de valores NA para un vector x:

```
porc_perdidos <- function(x) sum(is.na(x))/length(x)
```

Después, utilizando *apply()* pasamos la función a las columnas (segundo argumento = 2) y luego multiplicamos el resultado por 100 para pasarlo a porcentaje:

```
round(100*apply(selfreport, 2, porc_perdidos), 1)
```

```
##  src   id  pop  age  sex   hm   wm   hr   wr  prg
##  0.0   0.0  0.0  0.0  0.0  39.0 39.0  0.0  0.0 80.4
##  edu  etn  web  bm   br
## 61.0 61.0  0.0 39.0  0.0
```

Imputación de datos faltantes

Eliminación de individuos/variables

```
round(100*apply(selfreport, 2, porc_perdidos), 1)
```

```
##  src   id  pop age  sex  hm  wm  hr   wr  prg
##  0.0  0.0  0.0  0.0  0.0 39.0 39.0 0.0  0.0 80.4
##  edu  etn  web  bm  br
## 61.0 61.0  0.0 39.0  0.0
```

- La variable *prg* (estado de embarazo) no tiene ningún valor asociado en 4 de cada 5 individuos.
- Las variables *edu* y *etn* (nivel educativo y etnia respectivamente) no tienen ningún valor asociado en 3 de cada 5 individuos.
- Si basáramos el corte en que tengan más de un 50% de datos faltantes, se eliminarían estas tres variables. El umbral se tendría que definir según las circunstancias.

Imputación de datos faltantes

Eliminación de individuos/variables

Ejemplo de eliminación de individuos: dataset *airquality*

En el conjunto **airquality** teníamos 6 variables, por este orden: ozono, radiación solar, viento, temperatura, mes y día.

Podemos analizar el % de datos perdidos que tienen los individuos, en este caso utilizando *apply()* por filas (segundo argumento = 1).

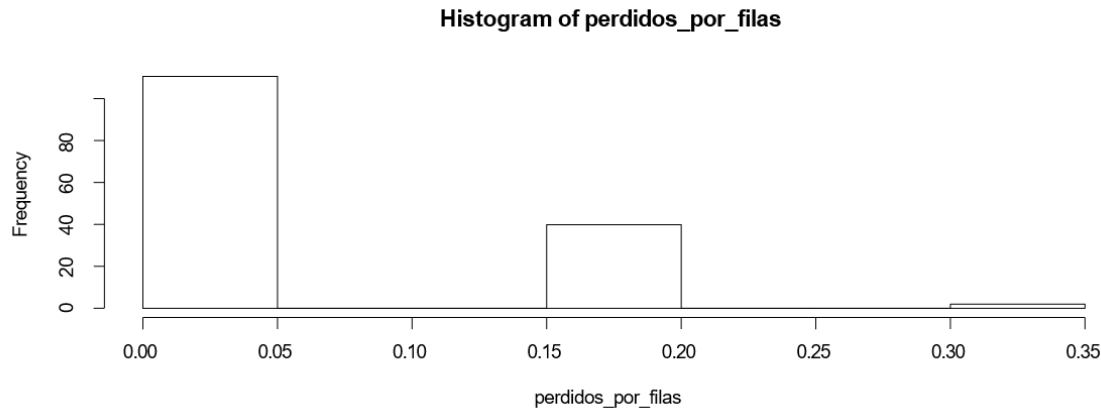
```
perdidos_por_filas <- apply(airquality, 1, porc_perdidos)
```

Imputación de datos faltantes

Eliminación de individuos/variables

Si los representamos en un histograma sencillo:

```
hist(perdidos_por_filas)
```



Vemos que hay algunos individuos con más del 30% de las variables con datos perdidos (2 de 6 variables, en este caso).

Imputación de datos faltantes

Eliminación de individuos/variables

Vemos que hay algunos individuos con más del 30% de las variables con datos perdidos (2 de 6 variables, en este caso). Veamos cuáles son y qué valores presentan:

##		Ozone	Solar.R	Wind	Temp	Month	Day
##	5	NA	NA	14.3	56	5	5
##	27	NA	NA	8.0	57	5	27

Presentan valores perdidos en Ozono y Radiación Solar. La imputación de estas variables en conjunto va a ser complicada, por lo que se podría optar por la eliminación.

Imputación de datos faltantes

Eliminación de individuos/variables

- La eliminación de individuos con valores perdidos es un procedimiento que **puede aumentar el sesgo de los resultados** que se obtengan después, sobre todo si la pérdida de datos no es MCAR.
- Se puede aplicar únicamente en los casos en los que **los individuos afectados sean muy pocos** o en casos en los que sepamos con certeza que no afectará al sesgo de las estimaciones.

Imputación de datos faltantes

Imputación de datos faltantes

- La imputación de datos faltantes consiste en la sustitución de dichos datos por **valores que pueda tomar esa variable**.
- Los métodos para realizar la imputación pueden ser diferentes en función de si se quiere imputar sólo una variable o varias.

Imputación de datos faltantes

Imputación de datos faltantes

Existen diversos métodos de imputación. Algunos de los más relevantes:

- **Hot deck:** se reemplaza el dato faltante por otro dato de ese mismo dataset para esa variable escogido al azar.
- **Media/mediana/moda:** se reemplaza el dato faltante por la media o la mediana de la variable si ésta es numérica, o por la moda si ésta es categórica.
- **Regresión/clasificación:** se ajusta un modelo de regresión utilizando los datos que sí se han podido medir, y se utiliza para predecir los valores de los datos faltantes.

Imputación de datos faltantes

Imputación de datos faltantes: Hot Deck

Hot Deck

- Recibe su nombre de las tarjetas perforadas en las que se almacenaban los datos.
- Se elige un valor al azar del conjunto de datos y se asigna dicho valor al dato faltante.
- Variante **Last Observation Carried Forward** (LOCF): si existe dependencia temporal en los datos, se coge el más reciente.
- Método poco recomendable salvo si el mecanismo es MCAR.

Imputación de datos faltantes

Imputación de datos faltantes: Hot Deck

Variente de Hot Deck: Predictive Mean Matching

- A cada dato perdido se le imputa el valor del individuo más similar a él, en función de una métrica definida.
- Resultados muy favorables, incluso en el caso MAR.
- Se puede realizar en R con el paquete {mice}
- Más información:

<https://stefvanbuuren.name/fimd/sec-pmm.html>

Imputación de datos faltantes

Imputación de datos faltantes: media/mediana/moda

- Si tenemos una variable numérica, sustituimos cada valor perdido por la media o la mediana.
- Si tenemos una variable cualitativa, sustituimos cada valor perdido por el valor más frecuente en el resto de individuos (la moda).

En R se puede realizar fácilmente:

```
head(x)
```

```
## [1] 1.4603030 NA 1.0681639 -0.6769162  
## [5] 1.5535168 -0.2727465
```

```
x[which(is.na(x))] <- mean(x, na.rm = T)
```

```
head(x)
```

```
## [1] 1.4603030 0.0372106 1.0681639 -0.6769162  
## [5] 1.5535168 -0.2727465
```

Imputación de datos faltantes

Imputación de datos faltantes: media/mediana/moda

- La imputación con estos valores puede ser útil en caso de que sea necesario proporcionar un conjunto de datos completo (p. ej. para aplicar un algoritmo).
- Sin embargo, salvo en el caso MCAR, puede provocar sesgos en las estimaciones, especialmente al estudiar relaciones entre variables.

Imputación de datos faltantes

Imputación de datos faltantes: regresión/clasificación

- También se pueden utilizar modelos de regresión o de clasificación para imputar valores faltantes.
- Se basa en considerar que la variable de interés depende del resto de variables del dataset a través de un modelo. Por ejemplo, si asumimos que es un modelo de regresión lineal:

$$X = \beta X'$$

- Ideal cuando el mecanismo de falta de datos es MAR.

Imputación de datos faltantes

Imputación de datos faltantes: regresión/clasificación

Cuando se realiza una imputación con modelos de regresión (u otros como el Predictive Mean Matching), se recomienda **repetir el procedimiento un cierto número de veces** (aunque sean pocos) para poder incluir la variabilidad de la propia modelización.

Es lo que se denomina **imputación múltiple**, frente a la imputación simple (que sería imputar considerando un solo valor posible).

A la hora de hacer el análisis con los datos imputados, se realizaría un promedio (*pooling*) a partir de los resultados obtenidos.

Imputación de datos faltantes

Imputación de datos faltantes: Ejemplos en R

El mejor paquete para realizar imputación de datos en R es el paquete *{mice}*, a través de la función homónima.

```
library(mice)  
mice(data, m = 5, method = NULL, ...)
```

Hay que especificar **el número de veces que repetimos el proceso** (*m*) y el **método de imputación** (*method*).

Imputación de datos faltantes

Imputación de datos faltantes: Ejemplos en R

Vamos a realizar una imputación del dataset **airquality**, con el que hemos trabajado anteriormente. Es un dataset con algunas variables con datos perdidos (*Ozone* y *Solar.R*):

```
apply(airquality, 2, porc_perdidos)
```

```
##      Ozone      Solar.R      Wind      Temp      Month      Day
## 0.24183007 0.04575163 0.00000000 0.00000000 0.00000000 0.00000000
```

Imputación de datos faltantes

Imputación de datos faltantes: Ejemplos en R

Para hacer la imputación, basta con indicar el dataset en la función `mice()` y especificar el número de repeticiones y el método. Dicho método empleará, para cada variable, todas las demás variables para imputar.

```
mice(airquality, m = 5, method = "pmm")
```

Imputación de datos faltantes

Imputación de datos faltantes: Ejemplos en R

Métodos disponibles:

- **pmm**: Predictive Mean Matching (por defecto)
- **mean**: imputación con la media
- **sample**: imputación Hot Deck
- **norm**: regresión lineal
- **logreg**: regresión logística
- **polr**: regresión ordinal (riesgos proporcionales)
- ...

Accediendo a *?mice* podemos encontrar el listado exhaustivo.

Imputación de datos faltantes

Imputación de datos faltantes: Ejemplos en R

Realizamos la imputación de airquality usando Predictive Mean Matching, repitiéndolo 5 veces, y la almacenamos en el objeto imp:

```
imp <- mice(airquality, m = 5,  
            method = "pmm",  
            print = FALSE)  
#Para que no muestre las iteraciones en pantalla
```

Imputación de datos faltantes

Imputación de datos faltantes: Ejemplos en R

Para obtener el dataset completo ya imputado, tenemos que llamar a la función *complete()* e introducir en ella el objeto generado por *mice()*, junto con un número para especificar a qué imputación estamos accediendo (a cuál de las m imputaciones realizadas):

```
complete(imp, 1)
```

Imputación de datos faltantes

Imputación de datos faltantes: Ejemplos en R

Pero de esa manera solo accedemos a una de las imputaciones múltiples, por lo que todavía nos quedaría agregar los resultados (pooling).

```
comp_imp<-complete(imp,"long")
aggregate(comp_imp[,-(1:2)] ,
          by = list(comp_imp$.id),
          FUN= mean)[,-1]
```

Tema 1

- Introducción
- Datos Anómalos
- Imputación Estadística
- **Reducción de la Dimensionalidad**
- Ingeniería de variables

Reducción de la Dimensionalidad

Introducción

Dentro de los problemas que podíamos resolver con técnicas estadísticas en el preprocesamiento de un conjunto de datos, teníamos:

- Datos ruidosos:
 - Individuos anómalos \Rightarrow métodos de eliminación de outliers.
 - **Variables redundantes/irrelevantes \Rightarrow técnicas de reducción de dimensiones.**
- Datos incompletos (valores faltantes) \Rightarrow técnicas de imputación estadística.
- Datos confusos (interpretación poco clara) \Rightarrow transformaciones de variables e ingeniería de variables (feature engineering).

Reducción de la Dimensionalidad

Introducción

- Grandes datasets cada vez más comunes.
- Información redundante o irrelevante para el análisis.
- Volumen de datos inmanejable para algoritmos o métodos estadísticos.
- “Curse of dimensionality”: cuantas más variables tengamos, más individuos necesitaremos para la modelización estadística.

Reducción de la Dimensionalidad

Introducción

Sea $X_{n \times p}$ un dataset con n instancias (individuos) y p variables. Dos formas de reducir la dimensión:

- Reducir las instancias: obtener $X_{n' \times p}$, donde $n' < n$.
- Reducir las variables:
 - Extracción: obtener un nuevo dataset (proyección) $X'_{n \times p'}$, donde $p' < p$.
 - Selección: obtener $X_{n \times p'}$, donde $p' < p$. Se realiza seleccionando las variables relevantes de X sin transformar el dataset.

En esta sesión nos centraremos en reducir las variables.

Reducción de la Dimensionalidad

Introducción: Métodos de extracción

Métodos de extracción para obtener una proyección con menos variables:

- Análisis de Componentes Principales (ACP, PCA en inglés).
- Análisis Factorial.
- Análisis de Correspondencias Múltiple. Escalamiento Multidimensional.
- T-distributed Stochastic Neighbor Embedding (T-SNE).
- UMAP, TriMAP y PaCMAP...

Estos métodos son **no supervisados**, no requieren tener de antemano una variable objetivo que queramos predecir.

Reducción de la Dimensionalidad

Introducción: Métodos de selección

Métodos de selección para eliminar/descartar variables:

- Selección de variables en un modelo lineal (StepWise, LASSO)
- Filtros
- Métricas de importancia
- Eliminación recursiva (**R**ecursive **F**eature **E**limination)
- . . .

Estos métodos son **supervisados**, requieren tener de antemano una variable objetivo que queramos predecir.

Reducción de la Dimensionalidad

Extracción de variables a partir de matrices de distancias

Algunas de las técnicas de reducción de dimensiones no parten del dataset original X , sino de la **matriz de distancias** que se puede derivar del mismo.

La matriz de distancias, D , es una matriz de dimensión $n \times n$ donde cada celda representa la distancia entre dos individuos del dataset.

Por ejemplo, la distancia entre el i -ésimo individuo y el j -ésimo individuo sería el elemento $d_{ij} = g(x_i, x_j)$, donde la función $g()$ es la que representa la distancia entre ellos.

Reducción de la Dimensionalidad

Extracción de variables a partir de matrices de distancias

La función más conocida es la **distancia euclídea**, que para un dataset con p columnas sería:

$$g(x_i, x_j) = \sqrt{(x_{1i} - x_{1j})^2 + (x_{2i} - x_{2j})^2 + \dots + (x_{pi} - x_{pj})^2}$$

Sin embargo, esta distancia puede no ser adecuada para variables de distinta magnitud o no numéricas.

Algunas alternativas son la distancia de **Manhattan**:

$$g(x_i, x_j) = |x_{1i} - x_{1j}| + |x_{2i} - x_{2j}| + \dots + |x_{pi} - x_{pj}|$$

O su generalización: la distancia de **Minkowski**.

$$g(x_i, x_j) = (|x_{1i} - x_{1j}|^p + |x_{2i} - x_{2j}|^p + \dots + |x_{pi} - x_{pj}|^p)^{1/p}$$

*Con p=1 es equivalente a la distancia de Manhattan.

*Con p=2 es equivalente a la distancia euclídea.

Reducción de la Dimensionalidad

Extracción de variables a partir de matrices de distancias

Si el dataset sólo tiene variables no numéricas, podemos binarizarlas y usar la distancia de Jaccard:

$$g(x_i, x_j) = \frac{\sum_{k=1}^p 1_{x_{ki} \neq x_{kj}}}{\sum_{k=1}^p 1_{x_{ki} \neq x_{kj}} + \sum_{k=1}^p 1_{x_{ki}=1 \cup x_{kj}=1}}$$

La función *dist()* en R permite calcular la distancia euclídea, junto con otras (Jaccard, Manhattan, etc.) a especificar en el argumento *method*.

Reducción de la Dimensionalidad

Extracción de variables a partir de matrices de distancias

Si el dataset tiene variables numéricas y no numéricas (habitual), es posible usar la distancia de Gower:

$$g(x_i, x_j) = \frac{\sum_{k=1}^p w_k \delta_{ij}^k d_{ij}^k}{\sum_{k=1}^p w_k \delta_{ij}^k}$$

- w_k es el peso asignado a la variable k .
- δ_{ij}^k toma el valor 0 si la variable k es binaria y $x_{ki} = x_{kj} = 0$, o si x_{ki} y/o x_{kj} es un dato faltante. En otro caso, toma el valor 1.
- d_{ij}^k depende del tipo de variable:
 - Si k es cualitativa, toma el valor 0 si $x_{ki} = x_{kj}$, y 1 en otro caso.
 - Si k es cuantitativa, equivale a $|x_{ki} - x_{kj}|$ dividido entre el rango de la variable.

Reducción de la Dimensionalidad

Extracción de variables a partir de matrices de distancias

Vamos a reducir las dimensiones del dataset **mtcars** (Henderson and Velleman, 1981).

Dataset con 32 individuos y 11 variables ($n = 32$, $p = 11$)

Cada individuo es un automóvil y las variables son atributos referidos a las capacidades del coche (consumo, cilindros, caballos, pesos, etc.)

```
summary(mtcars)
```

```
##           mpg           cyl           disp           hp           drat           wt
##  Min.      :10.40   Min.      :4.000   Min.      : 71.1   Min.      : 52.0   Min.      :2.760   Min.      :1.513
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080   1st Qu.:2.581
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695   Median :3.325
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597   Mean   :3.217
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920   3rd Qu.:3.610
##  Max.    :33.90   Max.    :8.000   Max.    :472.0   Max.    :335.0   Max.    :4.930   Max.    :5.424
##           qsec           vs           am           gear           carb
##  Min.      :14.50   Min.      :0.0000   Min.      :0.0000   Min.      :3.000   Min.      :1.000
##  1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :17.71   Median :0.0000   Median :0.0000   Median :4.000   Median :2.000
##  Mean   :17.85   Mean   :0.4375   Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.    :22.90   Max.    :1.0000   Max.    :1.0000   Max.    :5.000   Max.    :8.000
```

Reducción de la Dimensionalidad

Extracción de variables a partir de matrices de distancias

Las distancias euclídea, de Manhattan, Jaccard y Minkowski (entre otras) se pueden hallar con la función `dist()` implementada de base en R.

#Distancia euclídea

```
dist_euclidea <- dist(mtcars)
```

#Distancia de Jaccard

```
dist_jaccard <- dist(mtcars, method = "binary")
```

#Distancia de Manhattan

```
dist_manhattan <- dist(mtcars, method = "manhattan")
```

#Distancia de Minkowski

```
dist_minkowski <- dist(mtcars, method = "minkowski", p = p)
```

Reducción de la Dimensionalidad

Extracción de variables a partir de matrices de distancias

Para la distancia de **Gower** hay que cargar el paquete *cluster* y utilizar la función *daisy()*, especificando *metric = "gower"*. Adicionalmente, los pesos de cada variable los especificaríamos dando un vector al argumento *weights*.

```
#Distancia de Gower
```

```
library(cluster)
```

```
dist_gower <- daisy(mtcars, metric = "gower")
```

```
#Dando pesos de 0.5 a las 5 primeras variables
```

```
#y pesos de 0.25 a las 6 últimas variables
```

```
pesos <- c(0.5, 0.5, 0.5, 0.5, 0.5,  
          0.25, 0.25, 0.25, 0.25, 0.25, 0.25)
```

```
dist_gower <- daisy(mtcars, metric = "gower",  
                    weights = pesos)
```


Reducción de la Dimensionalidad

Escalamiento Multidimensional (MDS)

La técnica de Escalamiento Multidimensional (MDS por sus siglas en inglés) se basa en la misma idea que el Análisis de Componentes Principales (PCA), pero aquí partimos de una matriz de distancias (MDS métrico) o de una matriz de similitudes (MDS no métrico).

PCA es un subcaso de MDS. PCA obtiene exactamente los mismos resultados que MDS si se aplica con distancia euclídea.

Reducción de la Dimensionalidad

Escalamiento Multidimensional

- Una matriz de distancias, \mathbf{D} , es una matriz en la que cada una de las celdas contiene la distancia entre el individuo i y el individuo j de un conjunto \mathbf{X} , de forma que \mathbf{D} es una matriz $n \times n$.
- Una matriz de similitudes, Δ , es una matriz donde todos sus elementos representan la similitud en una cierta medida entre dos individuos (p. ej. el número de veces que toman la misma decisión).
- El Escalamiento Multidimensional nos permite reducir dicha matriz a una serie de variables que represente los datos en dimensión $k < n$.

Reducción de la Dimensionalidad

Escalamiento Multidimensional Métrico

Partiendo de una matriz D de distancias:

Obtenemos la matriz de similaridades $\mathbf{Q} = -\frac{1}{2} \mathbf{PDP}$ con $P = I - \frac{1}{n} \mathbf{1}\mathbf{1}'$.

Obtenemos los vectores propios, v , y los valores propios, λ , de Q .

Obtenemos las nuevas n variables mediante la fórmula:

$$x_i^i = v_i \lambda_i, i = 1, \dots, n$$

Reducción de la Dimensionalidad

Escalamiento Multidimensional Métrico

Al igual que en PCA, podemos emplear los valores propios para obtener una medida de la precisión en la reducción de dimensiones. Dicha medida es el coeficiente de Mardia:

Porcentaje de precisión de la representación con k variables=

$$100 * \frac{\lambda_1 + \dots + \lambda_k}{\sum_{i=1}^n |\lambda_i|}$$

Reducción de la Dimensionalidad

Escalamiento Multidimensional No Métrico

El MDS no métrico es un método no paramétrico que asume una relación funcional entre similaridades y distancias:

$$\delta_{ij} = f(d_{ij}), i, j = 1, \dots, n$$

$$\delta_{ij} > \delta_{ih} \Leftrightarrow d_{ij} > d_{ih}, i, j, h = 1, \dots, n$$

Se busca entonces encontrar las distancias minimizando las diferencias cuadráticas entre las similaridades y una configuración inicial (criterio STRESS):

$$S^2 = \frac{\sum_{i < j} (\delta_{ij} - \hat{d}_{ij})^2}{\sum_{i < j} \delta_{ij}^2}$$

El proceso se repite durante una serie de iteraciones hasta obtener la mejor solución.

Reducción de la Dimensionalidad

Escalamiento Multidimensional Métrico en R

El MDS métrico se aplica con la función `cmdscale()`, que requiere sólo una matriz de distancias y especificar un número de nuevas variables k que queramos obtener.

```
cmdscale(d,      #Matriz de distancias  
         k = 2,  #Número de variables a obtener (por defecto es 2)  
         eig = FALSE #Indica si queremos que devuelva  
                  #los valores propios  
                  #Por defecto no los devuelve (FALSE)  
         )
```

Reducción de la Dimensionalidad

Escalamiento Multidimensional Métrico en R

Vamos a aplicar el MDS métrico al dataset `mtcars` para obtener 2 nuevas dimensiones. Primero almacenamos la matriz de distancias euclídeas entre coches (según sus variables) usando la función `dist()`:

```
d <- dist(mtcars)
```

Posteriormente aplicamos `cmdscale()` y almacenamos el resultado en el elemento `fit`

```
fit <- cmdscale(d, k = 2, eig = TRUE)
```

Reducción de la Dimensionalidad

Escalamiento Multidimensional Métrico en R

La salida de *cmdscale()* (siempre y cuando fijemos *eig* = TRUE) será una lista con los siguientes elementos de interés:

- **points:** la nueva matriz de dimensión $n \times k$ con las nuevas variables
- **eig:** el vector de valores propios
- **GOF:** el vector con el coeficiente de Mardia acumulado hasta 1, 2, ..., k.

Reducción de la Dimensionalidad

Escalamiento Multidimensional Métrico en R

Guardamos los nuevos datos:

```
nuevo_dataset <- fit$points  
head(nuevo_dataset, 4)
```

```
##                [,1]      [,2]  
## Mazda RX4      -79.596425    2.132241  
## Mazda RX4 Wag  -79.598570    2.147487  
## Datsun 710      -133.894096   -5.057570  
## Hornet 4 Drive    8.516559   44.985630
```

El argumento GOF nos confirma que la precisión de las nuevas variables es del 99.93% para $k = 2$

```
fit$GOF
```

```
## [1] 0.9993673 0.9993673
```

Reducción de la Dimensionalidad

Escalamiento Multidimensional No Métrico en R

El MDS no métrico en R se lleva a cabo con el paquete MASS, a través de las funciones `sammon()` e `isoMDS()`. La diferencia es que la primera minimiza el STRESS y la segunda el S-STRESS (el denominador del STRESS). Ambos devuelven la misma información.

```
library(MASS)
fit2 <- sammon(d, k = 2, trace = F)
```

Podemos darle a este MDS una matriz de distancias, siempre y cuando todos sus elementos sean no negativos. La función `sammon()` devuelve una lista con dos elementos de interés:

- **points:** la nueva matriz de dimensión $n \times k$ con las nuevas variables.
- **stress:** el valor alcanzado por el STRESS en su última iteración.

Reducción de la Dimensionalidad

Escalamiento Multidimensional No Métrico en R

El MDS no métrico en R se lleva a cabo con el paquete MASS, a través de las funciones `sammon()` e `isoMDS()`. La diferencia es que la primera minimiza el STRESS y la segunda el S-STRESS (el denominador del STRESS). Ambos devuelven la misma información.

```
library(MASS)
fit2 <- sammon(d, k = 2, trace = F)
```

Podemos darle a este MDS una matriz de distancias, siempre y cuando todos sus elementos sean no negativos. La función `sammon()` devuelve una lista con dos elementos de interés:

- **points:** la nueva matriz de dimensión $n \times k$ con las nuevas variables.
- **stress:** el valor alcanzado por el STRESS en su última iteración.

Reducción de la Dimensionalidad

Escalamiento Multidimensional No Métrico en R

Guardamos los nuevos datos:

```
nuevo_dataset2 <- fit2$points  
head(nuevo_dataset, 4)
```

	[,1]	[,2]
Mazda RX4	-79.558234	1.880639
Mazda RX4 wag	-79.440506	2.469755
Datsun 710	-133.854698	-5.152498
Hornet 4 Drive	8.550853	45.164004

El argumento stress nos da el valor que ha alcanzado el STRESS en su última iteración.

```
fit2$stress
```

```
## [1] 1.754727e-05
```

Reducción de la Dimensionalidad

T-distributed Stochastic Neighbor Embedding (T-SNE)

- Método reciente basado en el kernel gaussiano desarrollado por van der Maaten y Hinton (2008).
- Capaz de reflejar comportamientos no lineales en la reducción de dimensiones.
- Ideal para representaciones gráficas, sobre todo acompañadas de análisis de agrupamientos (análisis cluster).

Reducción de la Dimensionalidad

T-distributed Stochastic Neighbor Embedding (T-SNE)

1. Se obtienen las similaridades entre puntos mediante la transformación:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2\right) / 2\sigma_i^2}{\sum_{i \neq j} \exp\left(-\|x_i - x_j\|^2\right) / 2\sigma_i^2}$$

Donde σ_i^2 es un parámetro que depende de la medida de “perplejidad” que se proporciona de antemano. Las similaridades se interpretan como probabilidades condicionadas.

2. Se obtienen las probabilidades conjuntas como:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

Reducción de la Dimensionalidad

T-SNE en R

El algoritmo T-SNE se puede aplicar en R usando el paquete Rtsne, con únicamente una función de interés: Rtsne()

```
install.packages("Rtsne")
```

```
Rtsne(X, dims = 2, initial_dims = 50,  
      perplexity = 30, theta = 0.5, check_duplicates = TRUE, pca = TRUE, partial_pca =  
      FALSE, max_iter = 1000,  
      verbose = getOption("verbose", FALSE), is_distance = FALSE,  
      Y_init = NULL, pca_center = TRUE, pca_scale = FALSE, normalize = TRUE,  
      stop_lying_iter = ifelse(is.null(Y_init), 250L,  
                                0L), mom_switch_iter = ifelse(is.null(Y_init), 250L, 0L), momentum = 0.5,  
      final_momentum = 0.8, eta = 200, exaggeration_factor = 12, num_threads = 1, ...)
```

Reducción de la Dimensionalidad

T-SNE en R

- **dims:** número de dimensiones k al que queremos reducir el dataset original.
- **perplexity:** medida de “perplejidad” (siempre ha de ser mayor que n).
- **theta:** cifra que permite ajustar velocidad del algoritmo a cambio de un peor ajuste (fijar en 0 si se desea el resultado exacto del T-SNE).
- **pca:** booleano que permite reducir primero las dimensiones con ACP antes de comenzar el procedimiento. En ese caso, especificar condiciones en `initial_dims`, `partial_pca`, `pca_center`, `pca_scale`.
- **max_iter:** número de iteraciones para minimizar la divergencia entre p_{ij} y q_{ij} .
- **normalize:** booleano para especificar si se estandariza el dataset.
- **is_distance:** booleano para indicar si el dataset introducido es una matriz de distancias o no.

Reducción de la Dimensionalidad

T-SNE en R

Vamos a reducir el dataset **mtcars** a 2 variables. Lo haremos sin aplicar PCA antes y obteniendo el resultado exacto.

La perplejidad se puede interpretar como una medida suavizada del número de vecinos que utilizaremos para el T-SNE. Como tenemos sólo 32 individuos, fijaremos la perplejidad en 3.

```
library(Rtsne)
t_sne <- Rtsne(mtcars, dims = 2,
               perplexity = 3, theta = 0, pca = FALSE)
```

Reducción de la Dimensionalidad

T-SNE en R

La salida de *Rtsne()* nos devuelve varios argumentos, del cual nos interesa **Y**, que constituye el nuevo dataset con el número de variables que hayamos indicado (en este caso, 2).

```
nuevo_dataset3 <- t_sne$Y  
head(nuevo_dataset3)
```

```
##      [,1]  [,2]  
## [1,] -75.16980    11.80845  
## [2,] -71.94302    14.44016  
## [3,] -38.90554   -78.03795  
## [4,] -45.30226    25.74305  
## [5,]  12.84368    36.95824  
## [6,] -48.98135    24.83617
```

Reducción de la Dimensionalidad

T-SNE en R

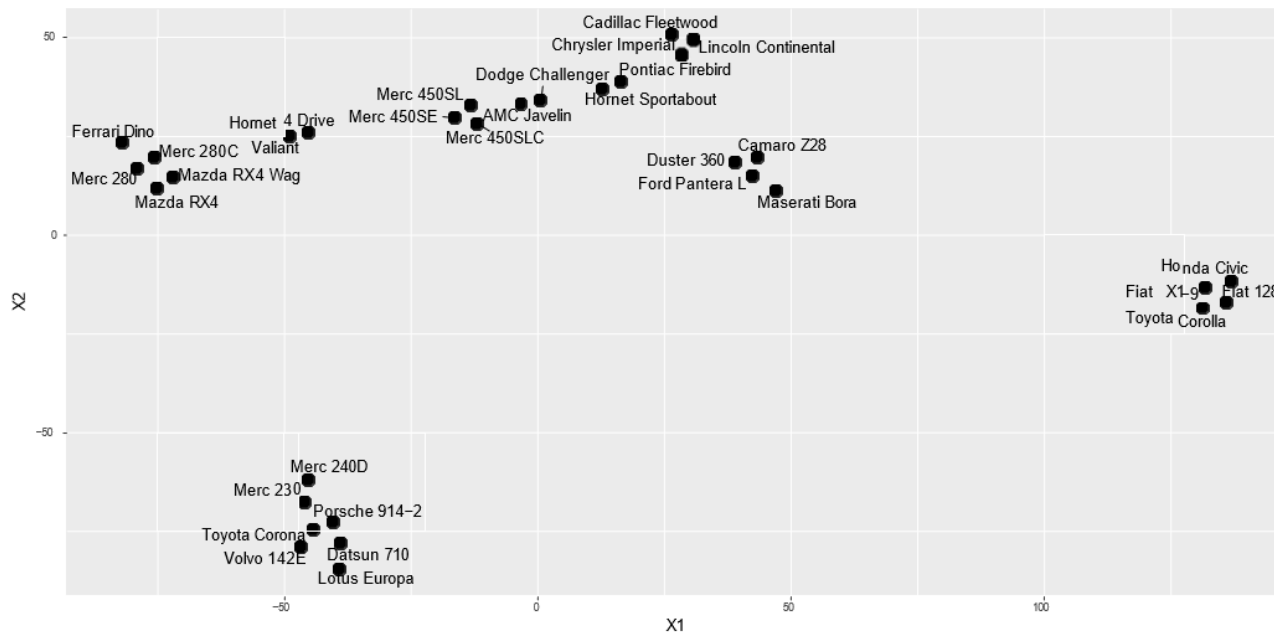
El algoritmo T-SNE tiende a agrupar los individuos en grupos muy segregados entre sí, tal y como se puede ver en nuestro ejemplo si representamos los nuevos datos:

```
ggplot(data.frame(nuevo_dataset3),  
  aes(x = X1, y = X2, label = rownames(mtcars))) +  
  geom_point(size = 3.5) + ggrepel::geom_text_repel()
```

Reducción de la Dimensionalidad

T-SNE en R

```
ggplot(data.frame(nuevo_dataset3), aes(x = X1, y = X2, label = rownames(mtcars)))  
+ geom_point(size = 3.5) + ggrepel::geom_text_repel()
```



Reducción de la Dimensionalidad

T-SNE en R

Alternativamente, a *Rtsne()* le podemos proporcionar una **matriz de distancias**. La función es la misma pero en ese caso debemos especificar el argumento *is_distance = TRUE*.

```
matriz_distancias <- dist(mtcars)
```

```
t_sne <- Rtsne(matriz_distancias, dims = 2,  
               perplexity = 3, theta = 0, pca = FALSE, is_distance  
               = TRUE)
```

Reducción de la Dimensionalidad

Uniform Manifold Approximation and Projection: UMAP

Al igual que T-SNE, UMAP es una técnica de reducción de la dimensionalidad no lineal.

Puede ser utilizada para visualización pero también para reducción de la dimensionalidad más general.

Es más eficiente (escala mejor) que T-SNE.

Intenta preservar tanto la estructura local como global de los datos.

Reducción de la Dimensionalidad

Uniform Manifold Approximation and Projection: UMAP

El algoritmo está basado en tres asunciones sobre los datos:

1. Los datos están uniformemente distribuidos en Riemannian manifold;
2. La métrica de Riemannian es constante localmente (o puede aproximarse como tal).
3. El manifold está conectado localmente.

De estas asunciones es posible modelar el manifold con una estructura topológica difusa. El embedding se encuentra buscando una proyección de los datos de baja dimensionalidad que tenga la estructura topológica difusa equivalente más cercana posible.

<https://arxiv.org/abs/1802.03426>

Reducción de la Dimensionalidad

UMAP en R

El algoritmo UMAP se puede aplicar en R usando el paquete `umap`, con la función `umap()`

```
library(umap)
umap <- umap(mtcars)
```

La salida de `umap()` nos devuelve varios argumentos, del cual nos interesa **layout**, que constituye el nuevo dataset con las nuevas coordenadas.

```
nuevo_dataset4 <- umap$layout
head(nuevo_dataset4)
```

	[,1]	[,2]
Mazda RX4	-1.41129921	2.717781
Mazda RX4 Wag	-1.19673703	2.973634
Datsun 710	-1.17920239	4.376767
Hornet 4 Drive	0.39058785	-2.487032
Hornet Sportabout	1.34320152	-3.921189
Valiant	0.03865413	-1.915573

Reducción de la Dimensionalidad

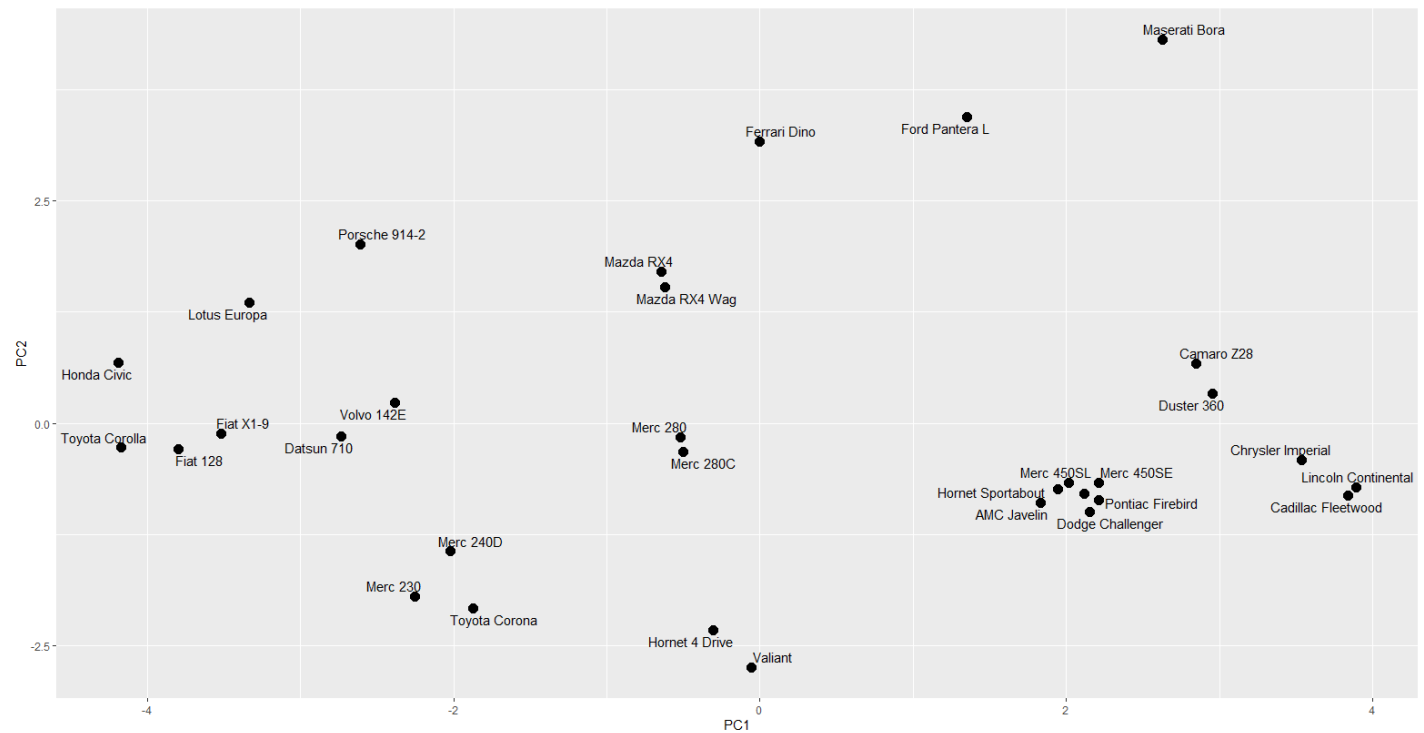
PCA vs T-SNE vs UMAP en R

Ejemplos de ejecución

- `pca <- prcomp(mtcars, scale = TRUE)`
- `ggplot(data.frame(pca$x[,1:2]), aes(x = PC1, y = PC2, label = rownames(mtcars))) + geom_point(size = 3.5) + ggrepel::geom_text_repel()`
- `library(Rtsne)`
- `t_sne <- Rtsne(mtcars, dims=2, perplexity=3, theta=0, pca=TRUE)`
- `ggplot(data.frame(t_sne$Y), aes(x = X1, y = X2, label = rownames(mtcars))) + geom_point(size = 3.5) + ggrepel::geom_text_repel()`
- `library(umap)`
- `umap <- umap(mtcars)`
- `ggplot(data.frame(umap$layout), aes(x = X1, y = X2, label = rownames(mtcars))) + geom_point(size = 3.5) + ggrepel::geom_text_repel()`

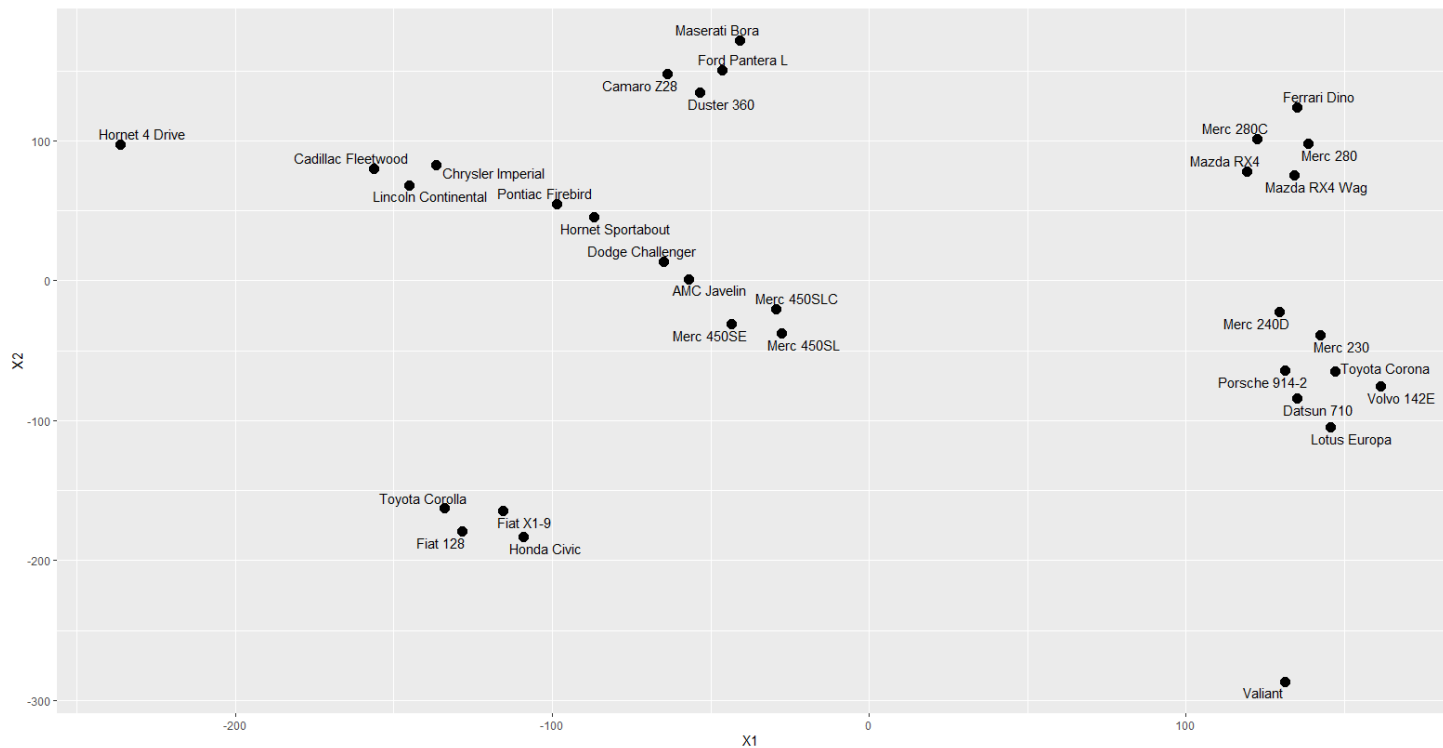
Reducción de la Dimensionalidad

PCA vs T-SNE vs UMAP en R



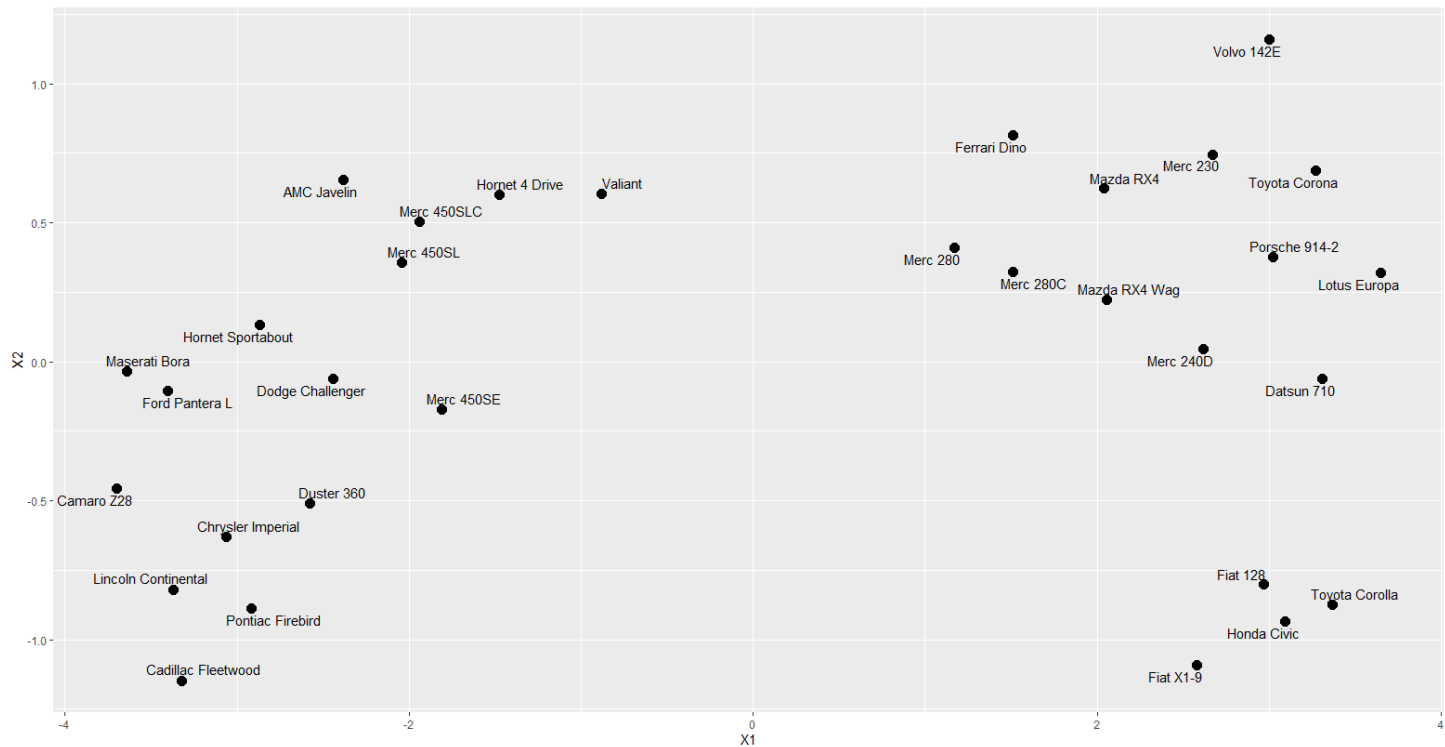
Reducción de la Dimensionalidad

PCA vs T-SNE vs UMAP en R



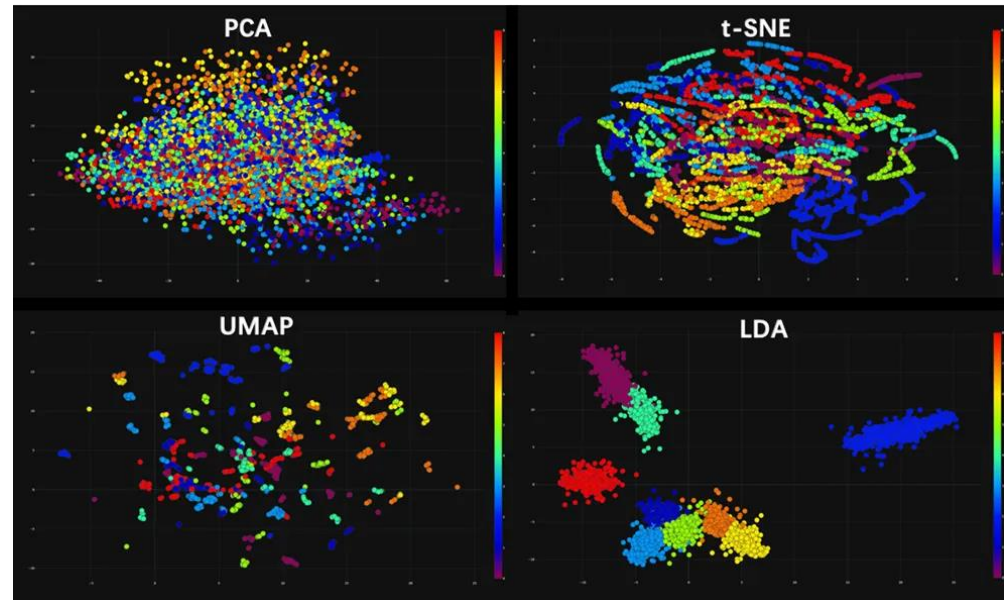
Reducción de la Dimensionalidad

PCA vs T-SNE vs UMAP en R



Reducción de la Dimensionalidad

PCA vs T-SNE vs UMAP

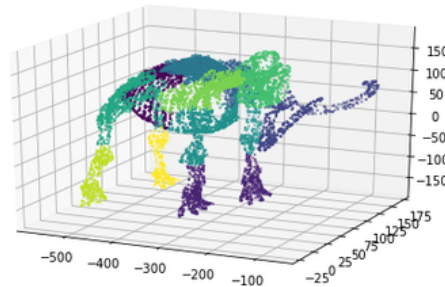


<https://towardsdatascience.com/dimensionality-reduction-for-data-visualization-pca-vs-tsne-vs-umap-be4aa7b1cb29>

Reducción de la Dimensionalidad

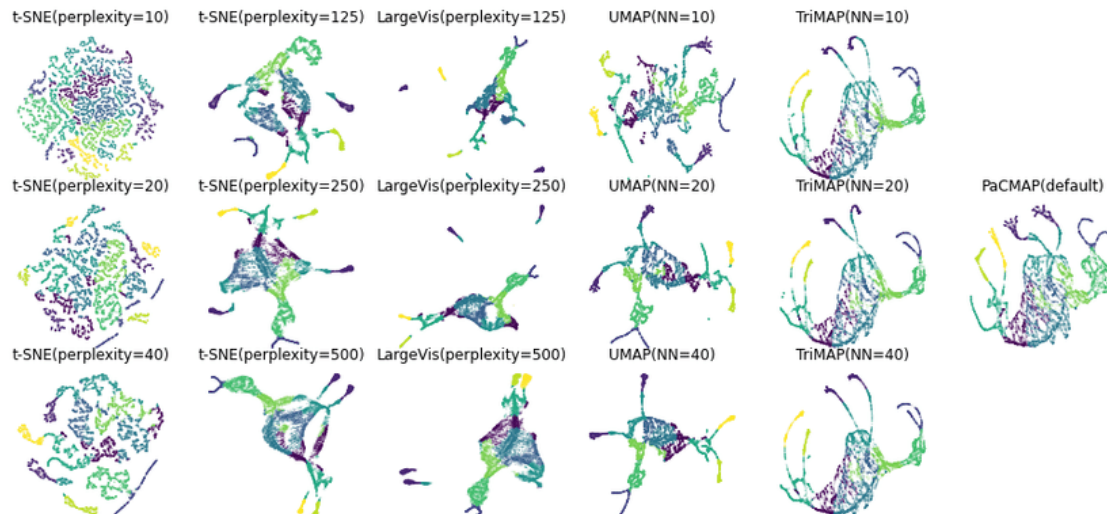
PCA vs T-SNE vs UMAP... TriMAP...PaCMAP...

Original Mammoth



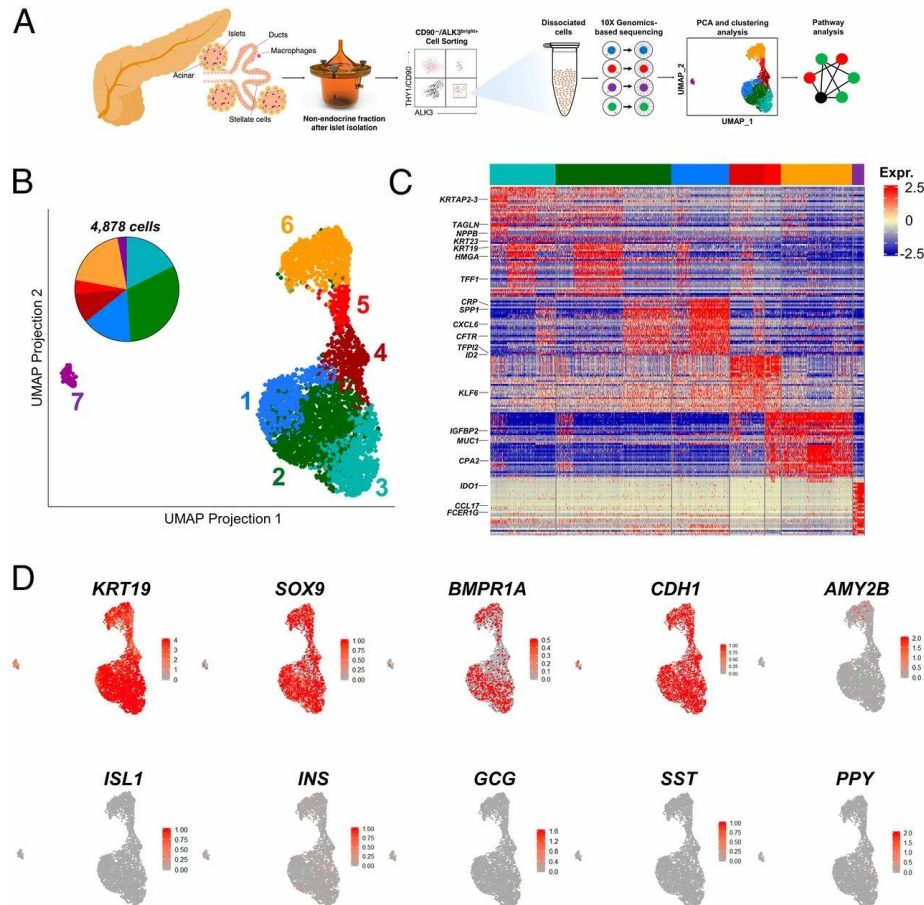
Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMAP, and PaCMAP for Data Visualization

<https://doi.org/10.48550/arXiv.2012.04456>



Reducción de la Dimensionalidad

Aplicación: Single Cell Analysis



Single-cell resolution analysis of
the human pancreatic ductal
progenitor cell niche

<https://doi.org/10.1073/pnas.1918314117>

Reducción de la Dimensionalidad

Selección de variables en modelos lineales

Sea y una variable conocida para los n individuos del dataset \mathbf{X} , queremos construir un modelo lineal multivariante, del tipo:

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k + E, \quad k < p, E \sim N(0, \sigma^2)$$

Asumimos que el número de variables existentes en el modelo óptimo es menor al número total de variables de las que disponemos.

Sin embargo, no sabemos qué variables de nuestro dataset son las que debemos incluir.

Reducción de la Dimensionalidad

Método StepWise

El método StepWise comprende los siguientes pasos:

1. Se comienza con el modelo saturado, con $k = p$.
2. Se evalúa alguna medida de bondad de ajuste (R^2 , AIC, BIC) considerando que quitamos una variable al modelo, para todas las posibilidades.
3. Seleccionamos, si lo hubiera, el modelo con $k - 1$ variables que mayor bondad de ajuste nos proporciona.
4. Volvemos al paso 2, pero esta vez considerando la inclusión de variables descartadas en pasos anteriores.
5. Iterar hasta que ningún otro posible modelo aumenta la bondad de ajuste.

Reducción de la Dimensionalidad

Método StepWise

Existen también otros métodos relacionados, aunque menos sofisticados, con el StepWise:

- Método Forward: se empieza con $k = 0$ y sólo se añaden variables hasta que el modelo no mejore.
- Método Backward: se empieza con $k = p$ y sólo se quitan variables hasta que el modelo no mejore.

La medida de bondad de ajuste que se emplea en R es el *Akaike Information Criterion (AIC)*, que depende de la función de log-verosimilitud:

$$AIC = -2L(x; \beta_0, \beta_1, \dots, \beta_k) + 2(k + 1),$$

donde L es la mencionada función de log-verosimilitud del modelo.

Reducción de la Dimensionalidad

Método StepWise

En el segundo sumando podemos sustituir 2 por $\log(n)$ para obtener el Bayesian Information Criterion (BIC):

$$AIC = -2L(x; \beta_0, \beta_1, \dots, \beta_k) + \log(n)(k + 1),$$

El modelo será mejor cuanto MENOR sea el AIC/BIC.

Reducción de la Dimensionalidad

Método StepWise en R

El método StepWise se puede aplicar en R con la función *step()*. Para ello, debemos proporcionarle un objeto de tipo *lm* o *glm*.

También podemos especificar, a través del comando *direction*, que aplique el método *Forward* (*direction* = "*forward*") o el *Backward* (*direction* = "*backward*").

```
modelo <- lm(y ~ x)
```

```
step(modelo, direction = "both", #Por defecto aplica  
#StepWise ("both")
```

```
trace = 1, #Por defecto saca por pantalla el proceso  
#Fijar en 0 si queremos que no lo saque
```

```
k = 2 #Dejar en 2 para AIC, y en log(n) para BIC  
)
```

Reducción de la Dimensionalidad

Método StepWise en R

Ponemos en práctica el método StepWise en R con la ayuda del dataset **swiss**

- 6 variables: fertilidad (medida estandarizada), % de hombres empleados en agricultura, % de reclutas obteniendo la máxima nota, % de reclutas con educación más allá de primaria, % de católicos y % de mortalidad infantil.
- 47 individuos, cada uno una provincia francófona de Suiza. Datos de 1888.

summary(swiss)

##	Fertility	Agriculture	Examination	Education	Catholic	Infant.Mortality
##	Min. :35.00	Min. : 1.20	Min. : 3.00	Min. : 1.00	Min. : 2.150	Min. :10.80
##	1st Qu.:64.70	1st Qu.:35.90	1st Qu.:12.00	1st Qu.: 6.00	1st Qu.: 5.195	1st Qu.:18.15
##	Median :70.40	Median :54.10	Median :16.00	Median : 8.00	Median : 15.140	Median :20.00
##	Mean :70.14	Mean :50.66	Mean :16.49	Mean :10.98	Mean : 41.144	Mean :19.94
##	3rd Qu.:78.45	3rd Qu.:67.65	3rd Qu.:22.00	3rd Qu.:12.00	3rd Qu.: 93.125	3rd Qu.:21.70
##	Max. :92.50	Max. :89.70	Max. :37.00	Max. :53.00	Max. :100.000	Max. :26.60

Reducción de la Dimensionalidad

Método StepWise en R

Vamos a intentar modelizar la mortalidad infantil según las otras 5 variables disponibles, seleccionando aquellas que proporcionen el mejor ajuste mediante el método StepWise.

Primero declaramos el modelo con todas las variables con la función *lm()* (ya que la variable objetivo es numérica).

```
modelo <- lm(Infant.Mortality ~ . , data = swiss)
```

Reducción de la Dimensionalidad

Método StepWise en R

No parece que todas las variables sean explicativas. Por ejemplo, el coeficiente asociado al % de católicos no aporta prácticamente nada (≈ 0).

`summary(modelo)`

```
##
## Call:
## lm(formula = Infant.Mortality ~ ., data = swiss)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.2512 -1.2860  0.1821  1.6914  6.0937
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.667e+00  5.435e+00   1.595  0.11850
## Fertility    1.510e-01  5.351e-02   2.822  0.00734 **
## Agriculture -1.175e-02  2.812e-02  -0.418  0.67827
## Examination  3.695e-02  9.607e-02   0.385  0.70250
## Education    6.099e-02  8.484e-02   0.719  0.47631
## Catholic     6.711e-05  1.454e-02   0.005  0.99634
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.683 on 41 degrees of freedom
## Multiple R-squared:  0.2439, Adjusted R-squared:  0.1517
## F-statistic: 2.645 on 5 and 41 DF, p-value: 0.03665
```

Reducción de la Dimensionalidad

Método StepWise en R

Introducimos el modelo en la función `step()`. Dejamos el resto de argumentos en los valores por defecto (salvo `trace` si no queremos obtener la salida), y guardamos la salida en un nuevo objeto, `modelo2`.

```
modelo2 <- step(modelo, trace = 0)  
modelo2
```

```
##  
## Call:  
## lm(formula = Infant.Mortality ~ Fertility + Education, data = swiss)  
##  
## Coefficients:  
## (Intercept)      Fertility      Education  
##      8.63758        0.14615        0.09595
```

El nuevo modelo sólo tiene 2 variables.

Reducción de la Dimensionalidad

Método StepWise en R

Introducimos el modelo en la función `step()`. Dejamos el resto de argumentos en los valores por defecto (salvo `trace` si no queremos obtener la salida), y guardamos la salida en un nuevo objeto, `modelo2`.

`summary(modelo2)`

```
##
## Call:
## lm(formula = Infant.Mortality ~ Fertility + Education, data = swiss)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7.6927 -1.4049  0.2218  1.7751  6.1685
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.63758     3.33524   2.590  0.012973 *
## Fertility      0.14615     0.04125   3.543  0.000951 ***
## Education     0.09595     0.05359   1.790  0.080273 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.614 on 44 degrees of freedom
## Multiple R-squared:  0.2296, Adjusted R-squared:  0.1946
## F-statistic: 6.558 on 2 and 44 DF, p-value: 0.003215
```

El nuevo modelo tiene un mayor R^2 (aunque sólo esté optimizando el AIC) y sus dos variables son más influyentes.

Reducción de la Dimensionalidad

Método StepWise en R

Podemos también acceder a los pasos que ha ido dando el modelo, accediendo al elemento anova del nuevo objeto generado por step().

`modelo2$anova`

##	Step	Df	Deviance	Resid.	Df	Resid.	Dev	AIC
## 1		NA		NA	41	295.0662	98.34145	
## 2	- Catholic	1	0.0001533995		42	295.0663	96.34147	
## 3	- Examination	1	1.3199421028		43	296.3863	94.55125	
## 4	- Agriculture	1	4.2499886025		44	300.6363	93.22041	

Vemos que los pasos han sido:

1. Elimina la variable de % de católicos.
2. Elimina la variable sobre las notas de reclutas.
3. Elimina la variable de % de empleo en agricultura.
4. El AIC no puede mejorar más \Rightarrow fin del proceso.

Reducción de la Dimensionalidad

Regresión de mejores subconjuntos

`summary(modelo_subsets)`

```
## Subset selection object
## Call: regsubsets.formula(Infant.Mortality ~ ., data = swiss, nvmax = 5)
## 5 Variables (and intercept)
##           Forced in Forced out
## Fertility      FALSE      FALSE
## Agriculture    FALSE      FALSE
## Examination    FALSE      FALSE
## Education      FALSE      FALSE
## Catholic       FALSE      FALSE
## 1 subsets of each size up to 5
## Selection Algorithm: exhaustive
##           Fertility Agriculture Examination Education Catholic
## 1 ( 1 ) "*"          " "           " "           " "           " "
## 2 ( 1 ) "*"          " "           " "           "*"           " "
## 3 ( 1 ) "*"          "*"           " "           "*"           " "
## 4 ( 1 ) "*"          "*"           "*"           "*"           " "
## 5 ( 1 ) "*"          "*"           "*"           "*"           "*"

```

La salida de *summary()* nos da los mejores modelos de cada tamaño (con 1 variable independiente, con 2, con 3. . .).

Reducción de la Dimensionalidad

Regresión de mejores subconjuntos

Dentro de los mejores, podemos seleccionar el mejor fijándonos en aquel con el menor *BIC*. Podemos recuperar el *BIC* de cada modelo del resumen obtenido en la diapositiva anterior:

```
summary(modelo_subsets)$bic
```

```
## [1] -1.2568831 -0.7118096  2.4691745  6.1095424  9.9596656
```

El menor BIC indicaría cuál es el mejor modelo. En este caso, parece ser el que sólo tiene una variable independiente (Fertility).

Reducción de la Dimensionalidad

Regresión LASSO

Otra alternativa de selección de variables en modelos lineales es la regresión desarrollada por Tibshirani (1996), denominada **Least Absolute Shrinkage and Selection Operator** (LASSO).

Está más enfocada a la predicción que a la inferencia, ya que se centra en optimizar el tradeoff entre sesgo y varianza. Es decir, prefiere que el modelo esté un poco sesgado si eso minimiza la varianza y hace que el error total sea menor que en el caso insesgado.

Reducción de la Dimensionalidad

Regresión LASSO

La regresión LASSO está basada en la regresión Ridge, la cual modificaba la función a optimizar en una regresión lineal clásica añadiendo un término que penaliza tener muchos coeficientes en el modelo:

$$L_{ridge}(\hat{\beta}) = \|y - \hat{\beta}x\|^2 + \lambda\|\hat{\beta}\|^2$$

La regresión LASSO añade la norma en L1 en vez de en L2:

$$L_{lasso}(\hat{\beta}) = \|y - \hat{\beta}x\|^2 + \lambda\|\hat{\beta}\|$$

Este detalle es importante ya que **permite llevar los coeficientes menos importantes a 0**. Es decir, los “hunde”. La implementación dentro de la propia función a optimizar hace que se llegue al mejor conjunto, aunque **no resuelve el problema del sesgo**.

Reducción de la Dimensionalidad

Regresión LASSO en R

La regresión LASSO se puede usar en R con la función *glmnet()* del paquete homónimo, que también permite calcular la *regresión Ridge* e incluso la *regresión Elastic Net*, la cual añade en la función a optimizar tanto la norma en L1 como en L2, con un coeficiente que mide la importancia que le da a cada una.

```
install.packages("glmnet")
```

Reducción de la Dimensionalidad

Regresión LASSO en R

```
glmnet(x, y, family,  
       alpha, lambda, ...)
```

La función *glmnet()* tiene muchos argumentos, pero los más relevantes son:

- **x**: dataset con las **variables independientes (predictoras)** que introduciremos de inicio en el modelo.
- **y**: vector con la **variable dependiente (predicha)** que queremos modelizar.

Reducción de la Dimensionalidad

Regresión LASSO en R

```
glmnet(x, y, family,  
        alpha, lambda, ...)
```

- **family:** para especificar la distribución de y ("*gaussian*" para regresión lineal, "*binomial*" para regresión logística).
- **alpha:** para especificar el coeficiente de la *Elastic Net*
 - Si $\alpha = 0 \Rightarrow$ regresión **Ridge**
 - Si $\alpha = 1 \Rightarrow$ regresión **LASSO**
- **lambda:** coeficiente de regularización. Se puede hallar mediante validación cruzada.

Reducción de la Dimensionalidad

Regresión LASSO en R

Para repetir la modelización de la mortalidad infantil con LASSO primero debemos introducir el dataset con las variables independientes y la variable dependiente aparte:

```
x <- as.matrix(swiss[,1:5])  
y <- swiss[,6]
```

```
library(glmnet)  
modelo_lasso <- glmnet(x, y, family = "gaussian",  
                        alpha = 1)
```

Reducción de la Dimensionalidad

Regresión LASSO en R

La salida de *glmnet()* contiene los coeficientes estimados en todas las iteraciones hasta la convergencia de λ . Nos quedaremos con aquellos de la iteración final especificando $s = 0$ al llamar a *coef()*

```
coef(modelo_lasso, s = 0)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##              1
## (Intercept)  8.76883520
## Fertility    0.15003466
## Agriculture -0.01187388
## Examination  0.03572408
## Education    0.06033145
## Catholic     .
```

Reducción de la Dimensionalidad

Regresión LASSO en R

También podemos acceder al R^2 del modelo accediendo al elemento dev.ratio de la lista. Al igual que antes, tendremos el R^2 de cada iteración así que nos quedaremos con el de la última:

```
tail(modelo_lasso$dev.ratio, 1)  
## [1] 0.2439035
```

De igual manera, podemos acceder al λ de la iteración final accediendo a lambda, el cual nos será de utilidad para la reproducibilidad del modelo:

```
tail(modelo_lasso$lambda, 1)  
## [1] 0.003114871
```

Reducción de la Dimensionalidad

Regresión LASSO en R

Los p-valores e intervalos de confianza del modelo LASSO se pueden obtener usando la función *fixedLassoInf()* del paquete *selectiveInference*. Además de *x* e *y* le debemos especificar el vector de coeficientes *beta* y el *lambda* final.

```
library(selectiveInference)
```

```
modelo_lasso_ci <- fixedLassoInf(x, y,  
  beta = coef(modelo_lasso, s = 0)[-1], lambda =  
  tail(modelo_lasso$lambda, 1))
```

Reducción de la Dimensionalidad

Regresión LASSO en R

Obtenemos los p-valores:

```
modelo_lasso_ci$pv #p-valores  
## [1] 0.05448278 0.57356205 0.58657960 0.33844460
```

Y los intervalos de confianza (podemos fijar la confianza con el argumento alpha en la función fixedLassoInf())

```
modelo_lasso_ci$ci #intervalos de confianza  
##           [,1]      [,2]  
## [1,] -0.005338428 0.2674067  
## [2,] -0.105542406 0.1712012  
## [3,] -0.569993898 0.3283375  
## [4,] -0.179224674 0.2629093
```

Reducción de la Dimensionalidad

Filtros y métricas de importancia: Biblioteca FSelector

Existen otras técnicas para seleccionar las variables más relevantes en un contexto **supervisado** (con una variable objetivo).

Estas técnicas no asumen un modelo lineal, sino que se basan en otras medidas u otros métodos de predicción, como los árboles de decisión.

Tienen la ventaja de adaptarse mejor a datos complejos (no lineales, de gran volumen, etc.)

Reducción de la Dimensionalidad

Filtros y métricas de importancia: Biblioteca FSelector

Sea Y la variable objetivo de nuestro análisis. Algunos de los filtros que podemos usar en R son los siguientes:

CFS: calcula, con respecto a Y , la correlación o la entropía (dependiendo de si las variables son numéricas o no) del resto de variables y selecciona automáticamente las que mayor correlación/entropía tengan.

Chi-Cuadrado: calcula, con respecto a Y , la V de Cramer del resto de variables:

$$V = \sqrt{\frac{X^2}{n \cdot \min\{n - 1, p - 1\}}}$$

Reducción de la Dimensionalidad

Filtros y métricas de importancia: Biblioteca FSelector

Sea Y la variable objetivo de nuestro análisis. Algunos de los filtros que podemos usar en R son los siguientes:

- **OneR:** obtiene las tablas de contingencia de cada variable del dataset con respecto a la variable objetivo, y selecciona aquellas que más asociación tengan con Y .
- **Relief:** elige un número aleatorio de individuos y, usando a los k individuos más cercanos, selecciona las variables cuyos valores estén más relacionados a la variable objetivo.

Reducción de la Dimensionalidad

Filtros en R

Para usar estos filtros en R, tenemos que cargar el paquete *{FSelector}*:

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

Reducción de la Dimensionalidad

Filtros en R

Podemos aplicar estos filtros para seleccionar las variables más relevantes del dataset *swiss* de cara a predecir la mortalidad infantil. Por ejemplo, el filtro CFS lo aplicaríamos con la función *cfs()*; los argumentos son análogos a los de otras funciones como *lm()*, basta con la fórmula y el dataset.

```
cfs(Infant.Mortality ~ ., swiss)
```

```
## [1] "Fertility"
```

El método CFS sólo selecciona la variable *Fertility* como relevante para describir *Infant.Mortality*.

Reducción de la Dimensionalidad

Filtros en R

Para aplicar el algoritmo OneR, lo hacemos mediante la función *oneR()* de la misma manera que con *cfs()*:

```
oneR(Infant.Mortality ~ ., swiss)
```

```
##               attr_importance
## Fertility           0.787234
## Agriculture         0.787234
## Examination         0.787234
## Education           0.787234
## Catholic            0.787234
```

En esta ocasión, nos devuelve una lista con la “importancia” de los atributos.

Reducción de la Dimensionalidad

Filtros en R

Se pueden seleccionar los atributos más importantes a partir de una puntuación de importancia de varias maneras distintas. El paquete *FSelector* nos permite seleccionar a partir de la variable cuya diferencia con la siguiente variable (en orden de importancia) sea la mayor observada.

Reducción de la Dimensionalidad

Filtros en R

Por ejemplo, en esta lista los dos primeros atributos tienen una importancia mucho mayor que los otros tres:

```
ejemplo <- data.frame(importancia = c(10, 9, 5, 4, 3))  
row.names(ejemplo) <- paste0("X", 1:5)  
ejemplo
```

##	importancia
## X1	10
## X2	9
## X3	5
## X4	4
## X5	3

Reducción de la Dimensionalidad

Filtros en R

Lo que podemos hacer es quedarnos únicamente con las variables cuya importancia quede por encima de la variable cuya importancia con la siguiente (en orden decreciente) sea mayor. Las diferencias son:

	##	importancia
• Del primero al segundo = 1 punto.	## X1	10
• Del segundo al tercero = 4 puntos.	## X2	9
• Del tercero al cuarto = 1 punto.	## X3	5
• Del cuarto al quinto = 1 punto.	## X4	4
	## X5	3

Por tanto, seleccionamos sólo el primer y segundo atributos.

Reducción de la Dimensionalidad

Filtros en R

Volviendo al ejemplo de OneR:

```
seleccion_oneR <- oneR(Infant.Mortality ~ ., swiss)  
cutoff.biggest.diff(seleccion_oneR)
```

```
## [1] "Fertility"
```


Reducción de la Dimensionalidad

Filtros en R

```
seleccion_relief
```

```
##               attr_importance
## Fertility      0.020058693
## Agriculture    0.004047097
## Examination    0.016457300
## Education      0.003471842
## Catholic       0.045801661
```

El método Relief también nos devuelve una medida de importancia

```
cutoff.biggest.diff(seleccion_relief)
```

```
## [1] "Catholic"
```

Reducción de la Dimensionalidad

Filtros en R

Por último, el filtro Chi-Cuadrado también se puede aplicar mediante la función `chi.squared()`:

```
chi.squared(Infant.Mortality ~ ., swiss)
```

##	attr_importance
## Fertility	0
## Agriculture	0
## Examination	0
## Education	0
## Catholic	0

... sin embargo, este algoritmo es preferible para el caso en el que las variables sean cualitativas.

Tema 1

- Introducción
- Datos Anómalos
- Imputación Estadística
- Reducción de la Dimensionalidad
- **Ingeniería de variables**

Introducción

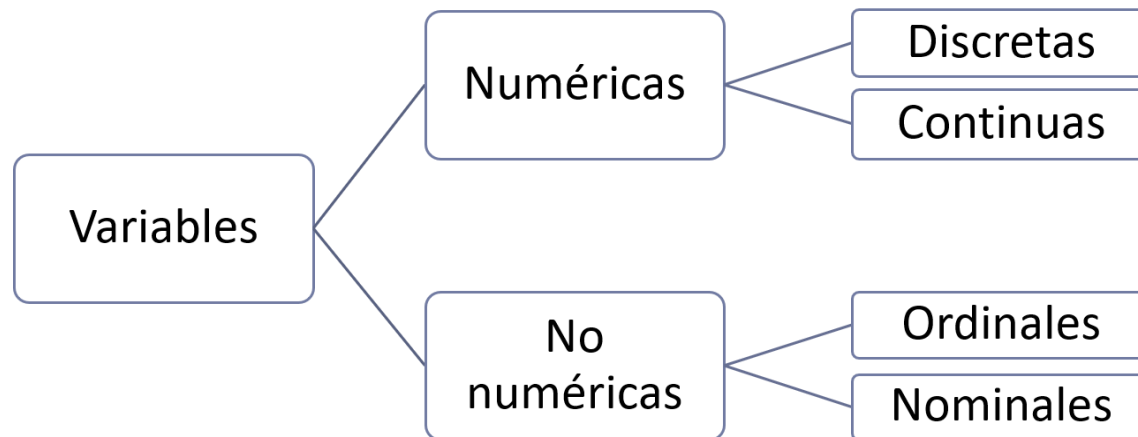
Dentro de los problemas que podíamos resolver con técnicas estadísticas en el preprocesamiento de un conjunto de datos, teníamos:

- Datos ruidosos:
 - Individuos anómalos \Rightarrow métodos de eliminación de outliers.
 - Variables redundantes/irrelevantes \Rightarrow técnicas de reducción de dimensiones.
- Datos incompletos (valores faltantes) \Rightarrow técnicas de imputación estadística.
- **Datos confusos (interpretación poco clara) \Rightarrow transformaciones de variables e ingeniería de variables (feature engineering).**

Introducción

En un dataset, estamos habituados a trabajar con las variables según se han medido en el experimento.

Es habitual que estas variables no sean suficientes para optimizar las técnicas aplicadas en minería de datos.



Introducción

Algunas de las transformaciones de variables más habituales que podemos realizar en minería de datos son:

- Normalización
- Transformación
- Binarización
- Discretización

Normalización

El término *normalización* suele asociarse en estadística a la tipificación utilizada en el cálculo de probabilidades de la Normal. Sin embargo, en minería de datos se consideran muchos tipos de normalizaciones:

- Z-score (tipificación).
- Min-max.
- Rankings.
- Números índices.

Normalización: Z-score

La normalización Z-score se basa en restarle, a cada valor de la variable, su media y dividir el resultado entre su desviación típica:

$$X'_i = \frac{X_i - \bar{X}}{\sigma_X}$$

En ocasiones, podemos encontrarnos esta transformación únicamente como la división de la variable original entre su desviación típica:

$$X'_i = \frac{X_i}{\sigma_X}$$

Normalización: Z-score

Dado que tanto la media como la desviación típica pueden verse altamente influidas por valores atípicos, una solución es sustituirlas por medidas robustas.

Por ejemplo, la media puede reemplazarse por la mediana, y la desviación típica puede sustituirse por la desviación media absoluta (MAD, Median Absolute Deviation):

$$MAD_X = \frac{\sum_{i=1}^n |X_i - \bar{X}|}{n}$$

Normalización: Z-score

Podemos calcular el Z-score para un vector en R directamente con la función *scale()*:

scale(*x*, *center* = *TRUE*, *scale* = *TRUE*)

- *center*: si queremos que reste la media a los valores de X .
- *scale*: si queremos que divida entre la medida de dispersión.
Si *center* = *TRUE*, entonces se divide entre la desviación típica.
Si *center* = *FALSE*, se divide entre la raíz del cuadrado medio:

$$\sqrt{\sum_{i=1}^n X_i^2 / (n - 1)}$$

Normalización: Min-Max

La normalización Min-Max se basa en restar a cada individuo el valor mínimo de la variable, y dividirlo entre el rango (máximo - mínimo):

$$X'_i = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$

El resultado es una variable X' que tomará valores en el intervalo entre 0 y 1.

Esta transformación no tiene una función construida en R, pero es fácil de calcular directamente: $x - \min(x)/(\max(x) - \min(x))$, siendo x un vector.

Normalización: Min-Max

Una normalización similar a la anterior es la de sustituir cada individuo por el número de orden en el que se encuentra en la variable X , tal que:

$$X'_i = \begin{cases} 1, & \text{si } X_i = X_1 \\ 2, & \text{si } X_i = X_2 \\ n, & \text{si } X_i = X_n \end{cases}$$

donde X_i sería el i-ésimo valor más alto de X . Por ejemplo, si $X = (532, 23, 214, 98)$, entonces según esta transformación $X' = (4, 1, 3, 2)$

Normalización: Rankings

Una normalización similar a la anterior es la de sustituir cada individuo por el número de orden en el que se encuentra en la variable X , tal que:

$$X'_i = \begin{cases} 1, & \text{si } X_i = X_1 \\ 2, & \text{si } X_i = X_2 \\ n, & \text{si } X_i = X_n \end{cases}$$

donde X_i sería el i-ésimo valor más alto de X . Por ejemplo, si $X = (532, 23, 214, 98)$, entonces según esta transformación $X' = (4, 1, 3, 2)$

Normalización: Rankings

Podemos hacer esta normalización en R con la función *rank()*

```
rank(x, na.last = TRUE, ties.method)
```

- **na.last:** argumento donde se especifica qué hacer con los valores perdidos.
 - Si na.last = TRUE, se ponen los últimos en el ranking.
 - Si na.last = FALSE, se ponen los primeros.
 - Si na.last = NA, se eliminan del ranking.
 - Si na.last = "keep", se mantienen con valor NA.
- **ties.method:** argumento donde se especifica qué hacer con los empates.

Normalización: Números índices

Es habitual, especialmente en series temporales, trabajar con números índices para poder comparar dos o más variables que se mueven en diferentes rangos de valores y ver cómo ha sido su comportamiento respecto al resto.

Normalización: Números índices

Para una variable X , fijamos una observación que servirá como base, X_0 , y el resto de observaciones equivaldrán al valor de cada una con respecto a la observación base:

$$X'_i = \frac{X_i}{X_0}$$

Dicho número índice suele multiplicarse por 100.

En R, bastaría con dividir cada individuo de la variable por el valor base:

```
x <- x/x[ind_base]
```

donde *ind_base* sería el individuo que utilizaremos de base.

Transformación

En la transformación de variables, ya no nos centramos sólo en cambiar los valores de escala u origen, sino en calcular variables nuevas en las que se modifique su naturaleza por completo.

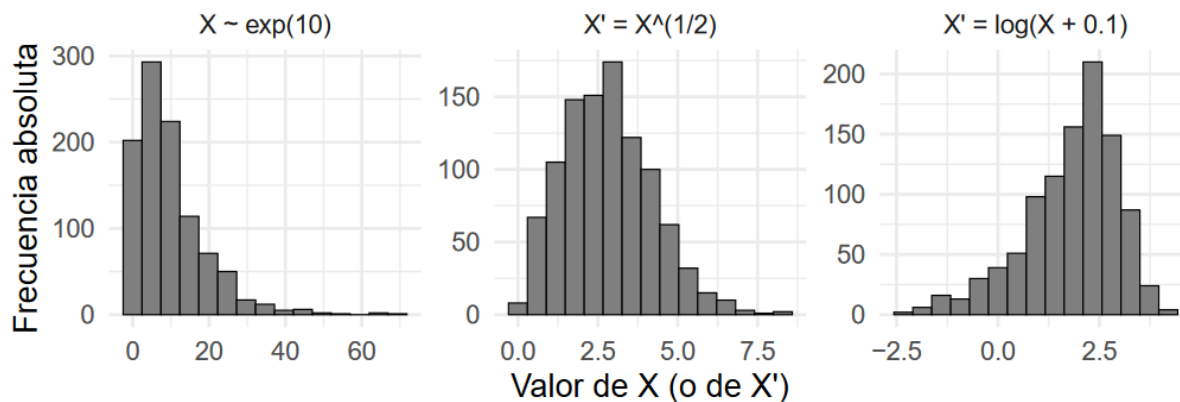
Podemos citar:

- Logaritmos y raíces.
- Relaciones funcionales entre variables.
- Transformación Box-Cox.
- Transformación de ranking

Transformación: Logaritmos y raíces

Unas transformaciones muy simples pero recurrentes, especialmente cuando se desea que una variable tenga una distribución similar a la Normal, son las que se basan en aplicar logaritmos o raíces de cualquier orden:

$$X'_i = \log_k(X_i + \epsilon), \epsilon \rightarrow 0 \quad X'_i = (X_i)^{\frac{1}{k}}$$



Transformación: Relaciones funcionales entre variables

Si sospechamos que dos variables interactúan o guardan algún tipo de relación, podemos construir nuevas variables a través de una función que utilice otras variables del dataset:

$$X'_i = f(X)$$

- El cálculo puede ser de cualquier tipo: combinación lineal, producto, cociente. . .
- El tipo de relación que pueden guardar se debe de conocer de antemano (**conocimiento experto**) para optimizar el procedimiento.

Transformación: Box-Cox

La transformación de Box-Cox es una generalización de cualquier transformación lineal, inversa o cuadrática que podamos hacer sobre una variable, y es útil para convertir una variable numérica en una distribución aproximadamente Normal.

$$\text{Si } X \in \mathbb{R}^+ \Rightarrow X' = \begin{cases} (X^\lambda - 1)/\lambda, & \lambda \neq 0 \\ \log(X), & \lambda = 0 \end{cases}$$

$$\text{Si } X \in \mathbb{R} \Rightarrow X' = \begin{cases} ((X + c)^\lambda - 1)/g\lambda, & \lambda \neq 0 \\ \log(X + c)/g, & \lambda = 0 \end{cases} \quad c \in \mathbb{R}^+$$

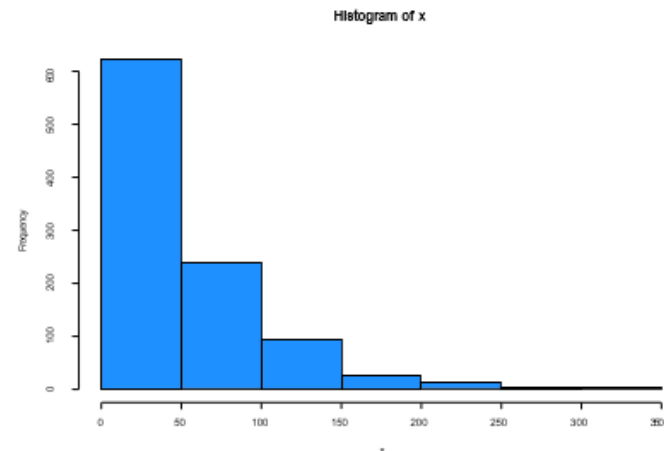
El parámetro λ se determina hallando el valor del intervalo $[-3, 3]$ que proporcione un mejor ajuste a la Normal, y para g suele tomarse la media geométrica de X .

Transformación: Box-Cox en R

La función *boxcox* del paquete *{MASS}* nos permite encontrar el valor de λ que proporcione la mejor aproximación a la Normal, pero hay que proporcionar el vector dentro de una fórmula (si no hay ningún modelo a especificar, entonces $x \sim 1$)

Lo probamos en una variable generada a partir de una *exp(50)*:

```
x <- rexp(1000, 1/50)  
hist(x, col = "dodgerblue")
```



Transformación: Box-Cox en R

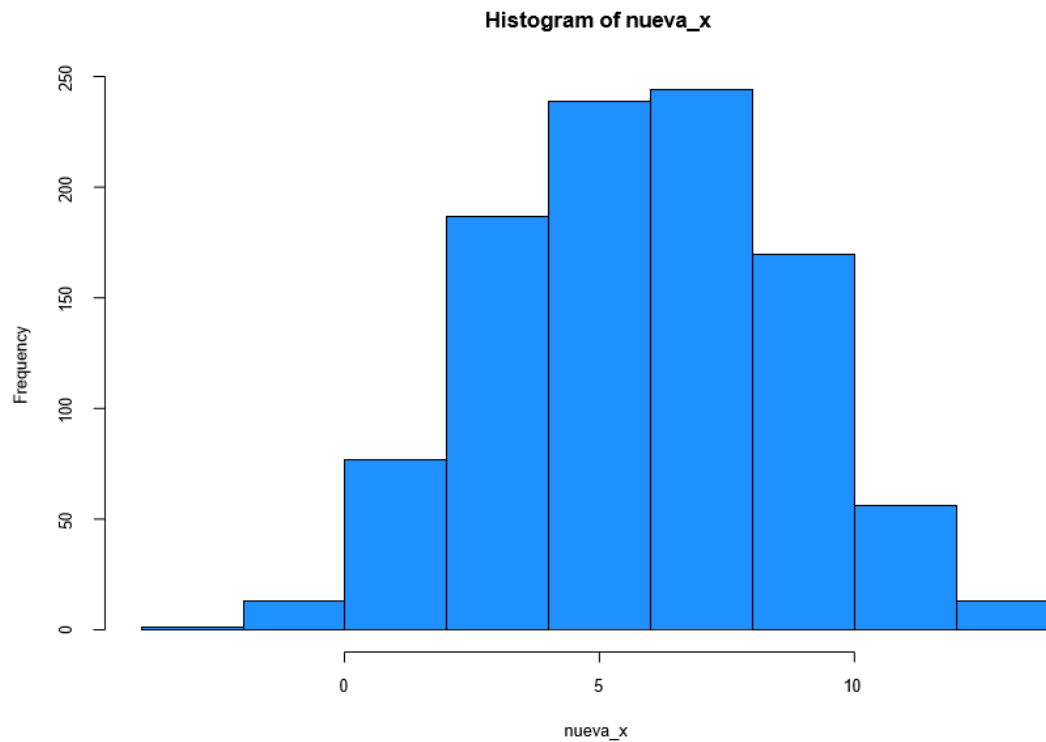
```
library(MASS)  
bc <- boxcox(x ~ 1)
```

En el elemento bc tendremos dos vectores: x, con los valores de λ evaluados, e y, con la bondad del ajuste a la Normal. Nos tendremos que quedar con el valor de x para el cual y sea máximo, así que con la ayuda de which.max():

```
lambda <- bc$x[which.max(bc$y)]  
nueva_x <- (x^lambda - 1) / lambda
```

Transformación: Box-Cox en R

```
hist(nueva_x, col = "dodgerblue")
```



Transformación: Transformación de ranking

Podemos reutilizar la normalización a través del ranking de cada valor para realizar una transformación basada en la función de distribución de la Normal:

$$X'_i = \phi^{-1} \left(\frac{R_i - 3/8}{n + 1/4} \right),$$

donde ϕ es la función de distribución de una Normal estándar y

$$R_i = \begin{cases} 1, & \text{si } X_i = X_1 \\ 2, & \text{si } X_i = X_2 \\ n, & \text{si } X_i = X_n \end{cases}$$

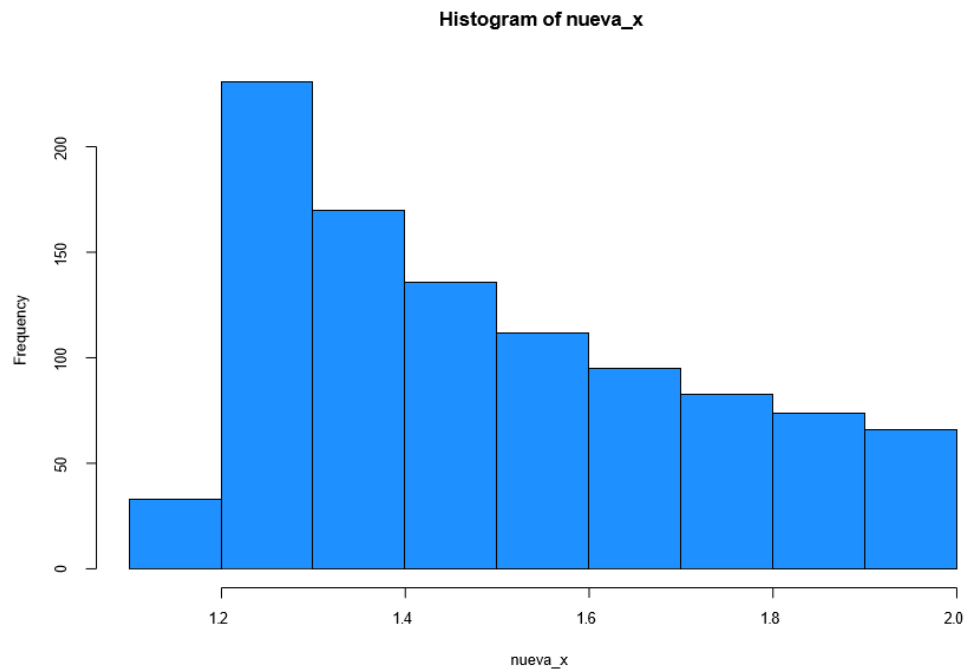
Transformación: Transformación de ranking en R

Para aplicar esta transformación en R nos podemos ayudar de la función *pnorm()*, que es la que devuelve los valores de la función de distribución para una Normal. La volvemos a probar para una $X \sim \text{exp}(50)$, la cual transformamos a través de la fórmula $(r - 3/8)/(n - 1/4)$:

```
x <- rexp(1000, 1/50)
r <- rank(x)
nueva_x <- pnorm((r - 3/8)/(1000 - 1/4))^(-1)
```

Transformación: Transformación de ranking en R

```
hist(nueva_x, col = "dodgerblue")
```



Binarización

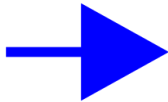
Muchos algoritmos de minería de datos, debido a su funcionamiento interno, tienen problemas a la hora de trabajar con variables categóricas.

Además, cuando la variable objeto de estudio toma más de dos clases (variable **multiclase**), los métodos de predicción en Machine Learning se pueden optimizar si se realiza la predicción para cada clase por separado.

Binarización

Para salvar estos problemas, podemos recurrir a la binarización de variables categóricas.

X ₁	X _i	X _p
			Sí			
			No			
			Sí			
			No			
			No			
			No			
			Sí			



X ₁	X _i _Sí	X _i _No	X _p
			1	0			
			0	1			
			1	0			
			0	1			
			0	1			
			0	1			
			1	0			

Binarización

Sea X una variable, presente en un dataset $\mathbf{X}_{n \times p}$, que puede tomar hasta k valores únicos o clases diferentes (x_1, x_2, \dots, x_k) . La binarización produce k nuevas variables binarias que indican si un individuo toma la clase x_k o no.

$$X \in (x_1, x_2, \dots, x_k) \Rightarrow B_1, B_2, \dots, B_k = 0, 1$$

$$B_{ji} = \begin{cases} 1, & \text{si } X_i = x_j \\ 0, & \text{si } X_i \neq x_j \end{cases}$$

Binarización en R

El paquete *{fastDummies}* de R posee la función *dummy_cols()* que permite obtener las variables binarias (**dummy**) de manera rápida y sencilla.

```
#install.packages("fastDummies")  
library(fastDummies)  
dummy_cols(data, select_columns = NULL,  
            remove_first_dummy = FALSE,  
            remove_most_frequent_dummy = FALSE, ignore_na  
            = FALSE, remove_selected_columns = FALSE)
```

Binarización en R

```
dummy_cols(data, select_columns = NULL,  
            remove_first_dummy = FALSE,  
            remove_most_frequent_dummy = FALSE,  
            ignore_na = FALSE, remove_selected_columns =  
            FALSE)
```

- **ignore_na**: dejar en *FALSE* si queremos que considere los *NA* como posible valor de la variable (y por tanto le creará una variable binaria).
- **remove_selected_columns**: en caso de que queramos eliminar aquellas variables ya binarizadas (por defecto deja en el dataset X y B_1, B_2, \dots, B_k ; si indicamos *TRUE*, dejará sólo B_1, B_2, \dots, B_k).

Binarización en R

El dataset *warpbreaks* almacena datos sobre el número de roturas en 54 telares durante su costura. Tiene tres variables:

- **breaks**: número de roturas en un telar concreto.
- **wool**: tipo de lana utilizada (A o B).
- **tension**: nivel de tensión; baja, media o alta (L, M o H).

`head(warpbreaks)`

##	breaks	wool	tension
## 1	26	A	L
## 2	30	A	L
## 3	54	A	L
## 4	25	A	L
## 5	70	A	L
## 6	52	A	L

Binarización en R

Para aplicar la binarización en este conjunto de datos sobre las dos variables cualitativas que hay, empleamos *dummy_cols()*.

```
datos <- dummy_cols(warpbreaks,  
                    select_columns = c("wool", "tension"))  
head(datos)
```

##	breaks	wool	tension	wool_A	wool_B	tension_L	tension_M	tension_H
## 1	26	A	L	1	0	1	0	0
## 2	30	A	L	1	0	1	0	0
## 3	54	A	L	1	0	1	0	0
## 4	25	A	L	1	0	1	0	0
## 5	70	A	L	1	0	1	0	0
## 6	52	A	L	1	0	1	0	0

Binarización en R

Si además le indicamos `remove_selected_columns = TRUE`, nos dejará únicamente las variables binarias en el dataset:

```
datos <- dummy_cols(warpbreaks,  
                    select_columns = c("wool", "tension"),  
                    remove_selected_columns = TRUE)
```

```
head(datos)
```

##	breaks	wool_A	wool_B	tension_L	tension_M	tension_H
## 1	26	1	0	1	0	0
## 2	30	1	0	1	0	0
## 3	54	1	0	1	0	0
## 4	25	1	0	1	0	0
## 5	70	1	0	1	0	0
## 6	52	1	0	1	0	0

Discretización

En otras ocasiones, puede ser relevante convertir una variable numérica continua en una variable discreta o cualitativa:

- Para poder aplicar otros modelos de predicción sobre ella (p. ej. regresión logística en lugar de lineal).
- Para poder realizar tablas de contingencia y métodos de exploración de datos (p. ej. gráficos).
- Para poder extraer reglas de asociación a partir de ella.

Discretización

Sea X una variable del dataset $\mathbf{X}_{n \times p}$, la discretización consiste en obtener k intervalos discretos a partir de X , de forma que obtengamos la nueva variable

$$X' = \{[d_0, d_1], (d_1, d_2], \dots, (d_{k-1}, d_k]\},$$

donde $d_0 = \min(X)$, $d_k = \max(X)$ y $d_i < d_{i+1}$, $i = 0, 1, \dots, k - 1$. Los valores d_1, d_2, \dots, d_{k-1} serían los puntos de corte.

Discretización

Encontrar el mejor procedimiento de discretización es un asunto muy complejo que depende del objetivo del estudio, además de otros factores.

Los métodos de discretización disponibles en la literatura representan un conjunto muy amplio, de los cuales podemos destacar:

- Intervalos de igual amplitud o frecuencia (o combinación de ambos: método PKID).
- Distancia de Mantaras.
- Algoritmos (CAIM, CACC, Chi2, Ameva. . .).

Discretización en R

La función `cut()` de R nos permite dividir variables continuas en grupos, según los puntos de corte que definamos en el argumento *breaks*. Por ejemplo, para dividir 100 datos procedentes de una $U(0, 1)$ en los intervalos $[0, 0.25]$, $(0.25, 0.5]$, $(0.5, 1]$:

```
x <- runif(1000)
y <- cut(x, breaks = c(0, 0.25, 0.5, 1),
        include.lowest = T) #Si queremos que el 1er intervalo
                             #incluya también el límite inferior
table(y)
```

## y			
##	[0,0.25]	(0.25,0.5]	(0.5,1]
##	244	242	514

Discretización en R

Si queremos aplicar la discretización con intervalos de igual amplitud o frecuencia, podemos usar la función `discretize()` del paquete `{arules}` (el cual también se utiliza para hacer reglas de asociación):

```
# install.packages("arules")
library(arules)
discretize(x, method = "frequency",
           breaks = 3, ...)
```

En ***method*** podemos especificar:

- *frequency*: para intervalos de igual frecuencia.
- *interval*: para intervalos de igual amplitud.
- *fixed*: si queremos que cada valor de la variable sea un intervalo (útil para variables discretas).

Discretización en R

Por último, en el paquete *{discretization}* podemos acceder a una amplia gama de algoritmos de discretización. La mayoría de ellos sólo requieren un vector o un conjunto de datos.

```
# install.packages("discretization")
library(discretization)
ameva(tb) #tb = matriz con las frecuencias
cacc(tb) # observadas de cada valor
caim(tb)
chi2(data)
```