Tipos de análisis

Debido a la diversidad con la que contamos en los datos, existen diferentes formas de realizar el análisis. Sin embargo, retomando el informe anterior (1), la forma más común de clasificarlos es en los dos siguientes grupos:

• Cuantitativo. Es el análisis que se realiza sobre los datos de forma numérica aplicando métodos estadísticos. • Cualitativo. Es el análisis que no se basa en datos numéricos, sino en clasificaciones categóricas textuales, al que también le podemos aplicar diferentes métodos estadísticos.

Un conjunto de datos, puede presentar ambos tipos de características. De hecho, un mismo atributo se puede presentar en ambas formas. Por ejemplo, la valoración de un producto puede ser de forma cualitativa, en las categorías Malo, Normal y Bueno, o también de forma numérica del 1 al 5.

Además de este tipo de división, nosotros vamos a ver cómo podemos usar los métodos estadísticos de forma numérica a lo largo de este tema.

Para añadir valor a este, en el siguiente tema veremos cómo también podemos mostrar esta información estadística de forma gráfica.

Resúmenes estadísticos

Los resúmenes estadísticos son aquellos que se basan en la estadística descriptiva para, como su nombre indica, describir las propiedades de los datos y el conjunto.

Nosotros vamos a cargar el conjunto de datos iris, que es uno de los datasets más comunes y estudiados. Una vez cargado, exploraremos sus propiedades haciendo uso de la estadística descriptiva.

Podemos cargarlos directamente desde el módulo datasets de la librería scikit-learn. Esta librería contiene diferentes datasets de juguete (Boston house prices, Iris, Diabetes, Wine): https://scikit-learn.org/stable/datasets/toy_dataset.html

Además contiene otros datasets del mundo real y datasets que han generado ellos mismos.

La ventaja es que no tenemos que tenerlo cargado en local, la desventaja es que el traspaso a data frame no tan visual y directo como ocurre con el uso de la librería pandas.

In [55]:

```
from sklearn import datasets
dataset = datasets.load_iris(return_X_y=True,as_frame=True)
print(dataset)
```

```
(     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[150 rows x 4 columns], 0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: target, Length: 150, dtype: int32)
```

Por otro lado, Podemos descargar el dataset de https://archive.ics.uci.edu/ml/machine-learning-databases/iris/ y descargar el archivo iris.data. Lo cargamos como hicimos en los temas anteriores.

In [56]:

```
import pandas as pd
properties = ['Logitud sépalo', 'Ancho sépalo', 'Longitud pétalo', 'Ancho pétalo', 'Clase']
dataset = pd.read_csv('iris.data', names=properties)
dataset.head()
```

Out[56]:

	Logitud sépalo	Ancho sépalo	Longitud pétalo	Ancho pétalo	Clase
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Una vez cargado como data frame, Python nos proporciona la función describe(), a modo de resumen sobre la que podemos extraer información genérica de nuestro dataset.

In [57]:

```
dataset.describe()
```

Out[57]:

	Logitud sépalo	Ancho sépalo	Longitud pétalo	Ancho pétalo
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Podemos extraer varias conclusiones de este análisis obtenido para cada columna (propiedad) del dataset iris.

- count: Esta columna nos muestra el número de muestras para esta propiedad. En este caso vemos que disponemos de 150 muestras por característica, pero existen datasets en los que una columna puede tomar valores indefinidos (N/A), inexistentes (None) e incluso que 0 u otro valor acordado represente la falta de datos para una propiedad de una instancia. Es un atributo muy importante ya que podemos ver a simple vista si contamos con pocos datos respecto al tamaño del dataset y, por tanto, no es conveniente utilizar dicha propiedad en el análisis.
- mean: Indica el valor medio de una propiedad considerando todos los valores con el mismo peso. Si el tipo de la propiedad es numerico no ordenado, podemos usar la media, pero para valores ordenados o basados en cadenas, necesitaremos usar la moda.
- std: muestra la dispersión (desviación típica) de los valores. Es decir, la medida en la que se alejan o no de la media. Poemos ver que esta será mayor conforme más distancia exista entre el valor mínimo y el máximo. Debido a que es sensible a la variable, es conveniente normalizar los datos. (No es lo mismo comparar la dispersión entre variables con diferentes magnitudes)
- mínimo / máximo: Valores mínimo y máximo de cada propiedad
- 25% / 50% / 75%: Cuartiles que indican el valor a partir del cual se encuentra el porcentaje indicado de datos

Si usamos un datasets que contenga atributos de tipo caracter: https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer/

In [58]:

```
properties = ['class', 'age', 'menopause', 'tumor-size', 'inv-nodes','node-caps','deg-malig','breast','breast-
dataset_cancer = pd.read_csv('breast-cancer.data', names=properties)
dataset_cancer.head()
```

Out[58]:

	class	age	menopause	tumor-size	inv-nodes	node-caps	deg-malig	breast	breast-quad	irradiat
0	no-recurrence-events	30-39	premeno	30-34	0-2	no	3	left	left_low	no
1	no-recurrence-events	40-49	premeno	20-24	0-2	no	2	right	right_up	no
2	no-recurrence-events	40-49	premeno	20-24	0-2	no	2	left	left_low	no
3	no-recurrence-events	60-69	ge40	15-19	0-2	no	2	right	left_up	no
4	no-recurrence-events	40-49	premeno	0-4	0-2	no	2	right	right_low	no

In [59]:

```
dataset_cancer.describe() #Por defecto solo incluye valores numéricos
```

Out[59]:

	deg-malig
count	286.000000
mean	2.048951
std	0.738217
min	1.000000
25%	2.000000
50%	2.000000
75%	3.000000
max	3.000000

In [60]:

```
dataset_cancer.describe(include="all")
```

Out[60]:

	class	age	menopause	tumor-size	inv-nodes	node-caps	deg-malig	breast	breast-quad	irradiat
count	286	286	286	286	286	286	286.000000	286	286	286
unique	2	6	3	11	7	3	NaN	2	6	2
top	no-recurrence-events	50-59	premeno	30-34	0-2	no	NaN	left	left_low	no
freq	201	96	150	60	213	222	NaN	152	110	218
mean	NaN	NaN	NaN	NaN	NaN	NaN	2.048951	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	0.738217	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	2.000000	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	2.000000	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	3.000000	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	3.000000	NaN	NaN	NaN

Para datos categóricos veremos como para los anteriores no muestra un resultado (NaN), pero incluye otros adicionales relativos a este tipo de dato:

- unique: número de valores categóricos distintos. Por ejemplo en este dataset la clase únicamente puede tomar 2 valores (tener cáncer o no).
- top: el valor más común (moda).
- freq: frecuencia del valor más común (moda)

Análisis univariante.

Ahora vamos a ver respecto al resumen estadístico anterior, como podemos realizar un análisis univariante de cada propiedad, de forma que podamos tomar una decisión sobre cada una de ellas que nos sirva de base para el análisis del conjunto completo.

Vamos a ver las siguientes cuestiones:

- Distribución de frecuencias
- Media/Moda/Mediana
- Desviación típica / Coeficiente de variación
- Rango intercuartílico/Percentiles
- Test de normalidad

Distribución de frecuencias

Vamos a volver a cargar el dataset iris y veremos la frecuencia de un dato categórico, entero y real. Hay que tener en cuenta que cuando es categórico o entero si que podemos contar cuántas veces aparece un valor, pero con valores reales será necesario establecer rangos.

In [61]:

```
dataset['Clase'].value_counts() #Vemos como hay 50 instancias de cada clase
```

Out[61]:

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Clase, dtype: int64
```

In [62]:

```
dataset['Logitud sépalo'].value_counts() #Nos mostraría todos los posibles valores decimales
dataset['Logitud sépalo'].value_counts(bins=5) #bins es el número de intervalos que queremos construir
```

Out[62]:

```
(5.74, 6.46]     42
(5.02, 5.74]     41
(4.295, 5.02]    32
(6.46, 7.18]     24
(7.18, 7.9]      11
Name: Logitud sépalo, dtype: int64
```

¿Que conclusiones podemos obtener analizando la frecuencia?

Respecto a la clase podemos ver si un dataset esta o no balanceado. Es decir, si el número de observaciones que tenemos de una clase es muy inferior al de otras, al aplicar un modelo este las tendrá menos en cuenta, produciendo una desventaja en su clasificación.

Por otro lado, podemos visualizar como se distribuye una determinada propiedad. En este caso la longitud del sépalo tiende a valores cercanos al rango [5.02,6.46], siendo menos común las flores de sépalo de 7 centímetros.

El analisis univariante tiene que ir acompañado del bivariante, donde podríamos verificar si este sépalo mayor se debe a que detro de una determinada clase existe una tendencia a ello.

Media/Moda/Mediana

Otro aspecto relevante a observar es la Media, Moda y Mediana. Podemos coger al igual que la frecuencia las mismas columnas del data frame.

In [63]:

```
print("Media de longitud del sépalo: ",dataset['Logitud sépalo'].mean())
print("Mediana de longitud del sépalo: ",dataset['Logitud sépalo'].median())
print("Moda de longitud del sépalo: ",dataset['Logitud sépalo'].mode()[0])
```

```
Media de longitud del sépalo:  5.843333333333334
Mediana de longitud del sépalo:  5.8
Moda de longitud del sépalo:  5.0
```

Aunque en este caso no se observa tan claramente el efecto de cada uno de ellas, podemos observarlo en un ejemplo sencillo. Imaginemos que tenemos las alturas de una clase, donde uno de ellos es muy alto respecto al resto.

In [64]:

```
altura_alumnos = pd.DataFrame([2.18, 1.72, 1.72, 1.69, 1.68, 1.63, 1.56])
print("Media: ",altura_alumnos[0].mean())
print("Mediana: ",altura_alumnos[0].median())
print("Moda: ",altura_alumnos[0].mode()[0])
```

```
Media:  1.7400000000000002
Mediana:  1.69
Moda:  1.72
```

Podemos observar que aunque la mayoría de alumnos se encuentren enel rango [1.72,1.68], la media se desplaza hasta 1.74 por el peso del alumno más alto, mientras que la mediana se mantiene en 1.69, que podríamos entender como el valor cercano más probable que presentará un alumno nuevo. Por último, la moda es el valor que más se repite, en este caso 1.72.

Desviación típica / Coeficiente de variación

Otro dato importante a tener en cuenta es la desviación típica, que mide la dispersión de los datos respecto la media. Nos permite saber, por ejemplo en el caso anterior, si la altura de la clase es más o menos homogéna. Si volvemos al caso de iris, podemos ver si existe más dispersión en el ancho o el largo del sépalo.

In [65]:

```
print("Desviación típica de longitud del sépalo: ",dataset['Logitud sépalo'].std())
print("Desviación típica de ancho del sépalo: ",dataset['Ancho sépalo'].std())
```

```
Desviación típica de longitud del sépalo:  0.828066127977863
Desviación típica de ancho del sépalo:  0.4335943113621737
```

Observamos que la dispersión es mayor en la longitud que en el ancho. Es decir, si no conocemos las medidas de una nueva flor y le asignasemos la media, el error sería mayor en la longitud que en su anchura.

Pero, ¿Es adecuada esta comparación? Imaginemos que queremos acertar aleatoriamiente la altura y peso de una persona, y nos confundimos en 1 metro, y 3 kilos. Estamos cometiendo un mayor error en el peso si evaluamos la cantidad sin tener en cuenta las unidades pero, a nivel relativo, en la altura el error es mayor porque el rango de estimación es mucho menor que si nos referimos al peso.

El coeficiente de variación es adimensional, y nos permite ver la variación de una muestra. Se obtiene con la división de la desviación típica entre la media. Para la longitud del sépalo y el ancho, vemos que el coeficiente de variación (la distribución de sus valores respecto a la media), es similar. Si el C.V es menor o igual al 30%, significa que la media aritmética es representativa del conjunto de datos.

In [66]:

```
import scipy.stats as ss

ss.variation(dataset['Logitud sépalo'])
print("Coeficiente de variación de longitud del sépalo: ",ss.variation(dataset['Logitud sépalo']))
print("Coeficiente de variación de ancho del sépalo normalizado: ",ss.variation(dataset['Ancho sépalo']))
```

```
Coeficiente de variación de longitud del sépalo:  0.14123809899346393
Coeficiente de variación de ancho del sépalo normalizado:  0.141501827135083
```

In [67]:

```
import numpy as np

altura = np.array([2.18, 1.72, 1.72, 1.69, 1.68, 1.63, 1.56])
peso = np.array([110, 82, 82, 79, 68, 63, 56])
print("Desviación típica altura: ",altura.std())
print("Desviación típica peso: ",peso.std()) #La desviación típica es mucho mayor
print("Coeficiente variación altura: ",ss.variation(altura))
print("Coeficiente variación peso: ",ss.variation(peso))
```

```
Desviación típica altura:  0.18708286933869714
Desviación típica peso:  16.30450044294144
Coeficiente variación altura:  0.10751889042453858
Coeficiente variación peso:  0.21135463537146312
```

La diferencia respecto a la media es mayor en el peso que en la altura.

Rango intercuartílico/Percentiles

También podemos observar rango intercuartilico (IQR), que es la diferencia entre el tercer y el primer cuartil. Un percentil concreto nos indica el valor a partir del cual se encuentra un porcentaje de las observaciones. Por ejemplo si indicamos el 25%, se nos mostrará el valor a partir del cual se encuentran por debajo el 25% de las observaciones.

In [68]:

```
print("Cuartil 25:",dataset['Logitud sépalo'].quantile(0.25))
print("Cuartil 75:",dataset['Logitud sépalo'].quantile(0.75))
```

```
Cuartil 25: 5.1
Cuartil 75: 6.4
```

In [69]:

```
#Rango intercuartílico.
dataset['Logitud sépalo'].quantile(0.75)-dataset['Logitud sépalo'].quantile(0.25)
```

Out[69]:

```
1.3000000000000007
```

Test de normalidad

Dado que la mayoría de los métodos que aplicaremos consideran que la distribución de las muestras es normal, conviene comprobar mediantes test que efectivamente nuestras variables cumplen esta hipótesis.

Existen dos propiedades que nos indican si la distribución de una variable se desvía de una normal. Estos son los estadísticos de asimetría y curtosis.

Un valor de curtosis y/o coeficiente de asimetría entre -1 y 1, es generalmente considerada una ligera desviación de la normalidad, mientras que entre -2 y 2 hay una evidente desviación de la normal pero no extrema.

In [70]:

```
from scipy.stats import kurtosis,skew
print('Curtosis longitud sépalo:', kurtosis(dataset['Logitud sépalo']))
print('Asimetría longitud sépalo:', skew(dataset['Logitud sépalo']))
```

```
Curtosis longitud sépalo: -0.5735679489249765
Asimetría longitud sépalo: 0.3117530585022963
```

Observamos como se desvía ligeramente de la normal, ya que se aleja de 0. Los valores que obtendríamos en una normal serían cercanos a 0.

In [71]:

```
distirbucion_normal = np.random.normal(size= 5000)
print('Curtosis distribución normal:', kurtosis(distirbucion_normal))
print('Asimetría distribución normal:', skew(distirbucion_normal))
```

```
Curtosis distribución normal: -0.10322531158321047
Asimetría distribución normal: -0.07488202669625837
```

Como métodos analíticos de normalidad, usaremos el test de Shapiro-Wilk, aconsejable para muestras pequeñas (<50) por su sensibilidad a las desviaciones, para muestras más grandes podemos usar el test de Kolmogorov-Smirnov, que permite estudiar en función de su media y desviación típica si una muestra procede de una población con una determinada distribución.

Este segundo test no es viable en aquellas muestras en las que la media y varianza poblacional no sean representativas. Por ello, también se añade el test de Jarque Bera que no requiere estimaciones de los parámetros que caracterizan la normal.

En todos ellos, podemos rechazar la hipótesis de que la distribución de nuestra variable es normal si el p-value es menor que un determinado valor, por lo general alfa=0.05. Es decir:

- p<=alfa: rechazar hipótesis, aseguramos que no es normal.
- p>alfa: no podemos rechazar la hipótesis, asumimos normalidad porque es muy probable que sea cierto tras la prueba realizada.

In [72]:

```
from scipy.stats import shapiro
from scipy.stats import kstest
from scipy.stats import jarque_bera

#Test de Shapiro-Wilk
stat, p = shapiro(dataset['Logitud sépalo'])
#Asumimos normalidad por que p-value<0.05
print("p-value del test Shapiro-Wilk: ",p)

stat, p = kstest(dataset['Logitud sépalo'], 'norm')
print("p-value del test Kolmogorov-Smirnov: ",p)

stat, p = jarque_bera(x = dataset['Logitud sépalo'])
print("p-value del test de Jarque-Bera: ",p)
```

```
p-value del test Shapiro-Wilk:  0.010180278681218624
p-value del test Kolmogorov-Smirnov:  0.0
p-value del test de Jarque-Bera:  0.10614621817187797
```

En este caso la longitud del sépalo no sigue una distribución normal, ya que en la mayoría de los test p-value<0.05, es decir, podemos rechazar la hipótesis de que la muestra sea normal y tenemos suficientes evidencias para asumir que la curtosis y asimetría se distancian de la de una normal.

Podemos repetir estos test sobre nuestra muestra normal para ver que ocurre en este caso.

In [73]:

```
#Test de Shapiro-Wilk
stat, p = shapiro(distirbucion_normal)
#Asumimos normalidad por que p-value<0.05
print("p-value del test Shapiro-Wilk: ",p)

stat, p = kstest(distirbucion_normal, 'norm')
print("p-value del test Kolmogorov-Smirnov: ",p)

stat, p = jarque_bera(x = distirbucion_normal)
print("p-value del test de Jarque-Bera: ",p)
```

```
p-value del test Shapiro-Wilk:  0.01939791440963745
p-value del test Kolmogorov-Smirnov:  0.46311438902456825
p-value del test de Jarque-Bera:  0.031862460998596576
```

¿Por qué ocurre esto? Vamos a ver que ocurre cuando generamos una distribución normal con pocas muestras:

In [74]:

```
distirbucion_normal = np.random.normal(size= 150)
print('Curtosis distribución normal:', kurtosis(distirbucion_normal))
print('Asimetría distribución normal:', skew(distirbucion_normal))

#Test de Shapiro-Wilk
stat, p = shapiro(distirbucion_normal)
#Asumimos normalidad por que p-value<0.05
print("p-value del test Shapiro-Wilk: ",p)

stat, p = kstest(distirbucion_normal, 'norm')
print("p-value del test Kolmogorov-Smirnov: ",p)

stat, p = jarque_bera(x = distirbucion_normal)
print("p-value del test de Jarque-Bera: ",p)
```

```
Curtosis distribución normal: -0.09728829129004479
Asimetría distribución normal: -0.3141331172366561
p-value del test Shapiro-Wilk:  0.35962527990341187
p-value del test Kolmogorov-Smirnov:  0.6073010634015855
p-value del test de Jarque-Bera:  0.2827835957654059
```

Observamos que cuando contamos con pocas muestras, puede que la aleatoriedad cause que tengamos una muestra que diste algo de una normal, por lo que es necesario contar con suficientes datos.

Análisis bivariante

El objetivo de realizar un análisis bivariane es observar de forma simultánea el comportamiento de dos variables. A continuación vamos a ver distintos análisis y técnicas que podemos aplicar. La mayoría de ellos pueden extenderse a más de dos variables, convirtiendose en un análisis multivariante.

- Tablas de contingencia

Análisis de relación lineal bivariante:

- Covarianza
- Correlación

1. Las tablas de contingencia están diseñadas para evaluar valores categoricos. Si la mostramos sobre una columna con datos reales nos creará un valor de agrupación para cada uno de los posibles. Por eso, si queremos aplicarla sobre una columna de este tipo, debemos crear previamente rangos que agrupen dichos valores.

In [75]:

```
x = dataset['Logitud sépalo']
bins = np.arange(min(x),max(x),0.5) #Definimos rangos entre el min y el max con step=0.5
rangos = pd.cut(dataset['Logitud sépalo'],bins=bins)
```

In [76]:

```
pd.crosstab(rangos, dataset['Clase'])
```

Out[76]:

Clase	Iris-setosa	Iris-versicolor	Iris-virginica
Logitud sépalo			
(4.3, 4.8]	15	0	0
(4.8, 5.3]	24	5	1
(5.3, 5.8]	10	19	5
(5.8, 6.3]	0	15	13
(6.3, 6.8]	0	9	16
(6.8, 7.3]	0	2	8
(7.3, 7.8]	0	0	6

Observamos que la clase iris-setosa presenta flores de longitud de sépalo pequeños, mientras que en la clase virginica son más grandes.

En este caso, queda bien representada la distribución de cada clase en función de esta propiedad, ya que si nos dan una flor con una longitud de sépalo entre (4.3,4.8] podemos afirmar que es de la clase setosa, por ejemplo.

La mayor dificultad en este caso sería conocer a que clase pertenece una flor con longitud de sépalo entre [5.5,6], donde necesitaríamos más información para poder clasificar correctamente.

1. La covarianza nos permite encontrar si existe cierta relación lineal entre variables. Es decir, cuanto mas aumenta una mas aumenta la otra y viceversa. Por ejemplo, la anchura de la muñeca depende de la altura, de forma que una persona alta tendrá una muñeca de mayor grosor que una persona bajita.

In [77]:

```
matriz_covarianzas = dataset.cov()
matriz_covarianzas
```

Out[77]:

	Logitud sépalo	Ancho sépalo	Longitud pétalo	Ancho pétalo
Logitud sépalo	0.685694	-0.039268	1.273682	0.516904
Ancho sépalo	-0.039268	0.188004	-0.321713	-0.117981
Longitud pétalo	1.273682	-0.321713	3.113179	1.296387
Ancho pétalo	0.516904	-0.117981	1.296387	0.582414

En esta matriz vemos como la longitud del pétalo esta relacionada con su ancho positivamente, es decir, conforme más largo sea el pétalo más ancho será, y vicerversa.

También se observa que la relación entre la longitud del pétalo el ancho del sépalo es inversa, por lo que este tenderá a decrecer conforme aumenta el primero.

Uno de los problemas que presenta esta medida es que su valor depende de la unidades de las propiedades, por lo que tan solo podemos analizar si la relación es positiva o no, ya que definir si están muy relacionadas o no deperá de sus unidades de medida.

1. Por último, vamos a ver la correlación entre variables, que nos muestra la covarianza si estandarizamos las variables (media 0 y varianza 1). Este estadístico tambien se conoce como el coeficiente de correlación de Pearson puede tomar valores desde -1 a 1, en función del signo de la relación, negativa o positiva.

In [78]:

```
matriz_correlacion = dataset.corr()
matriz_correlacion
```

Out[78]:

	Logitud sépalo	Ancho sépalo	Longitud pétalo	Ancho pétalo
Logitud sépalo	1.000000	-0.109369	0.871754	0.817954
Ancho sépalo	-0.109369	1.000000	-0.420516	-0.356544
Longitud pétalo	0.871754	-0.420516	1.000000	0.962757
Ancho pétalo	0.817954	-0.356544	0.962757	1.000000

Observamos como la correlación con la misma variable es 1, el máximo. Además, se mantienen las relaciones que habíamos analizado en la matriz de covarianzas, pero ahora podemos establecer comparaciones, de forma que la relación entre la longitud del pétalo y su ancho es positiva y muy fuerte, mientras que entre el ancho del sépalo y la longitud del sépalo es negativa y débil, casi neutra por ser cercana a 0.0 (sería el punto en el que no existe relación).

Observamos que la distribución del ancho y longitud del sépalo mantienen los valores centrados, con pocos valores en las colas, mientras que para el pétalo presentan una distribución que se aleja de la normal.