universidad de león

Exploración y visualización de datos con Python

Métodos Numéricos y Estadísticos
Grado en Ingeniería Informática / Mecánica

Curso 2022-2023

Eva María Mazcuñán Navarro

Eva María Mazcuñán Navarro
Departamento de Matemáticas
Universidad de León
E-mail: emmazn@unileon.es

Contenidos

Introducción

En esta práctica aprenderás las técnicas básicas para explorar y visualizar un conjunto de datos con Python.

Los pingüinos del archipiélago Palmer

Presentaremos las diferentes técnicas a través de ejemplos trabajando con un conjunto de datos relativos a diferentes características de tres especies de pingüinos del archipiélago Palmer en la Antártida.

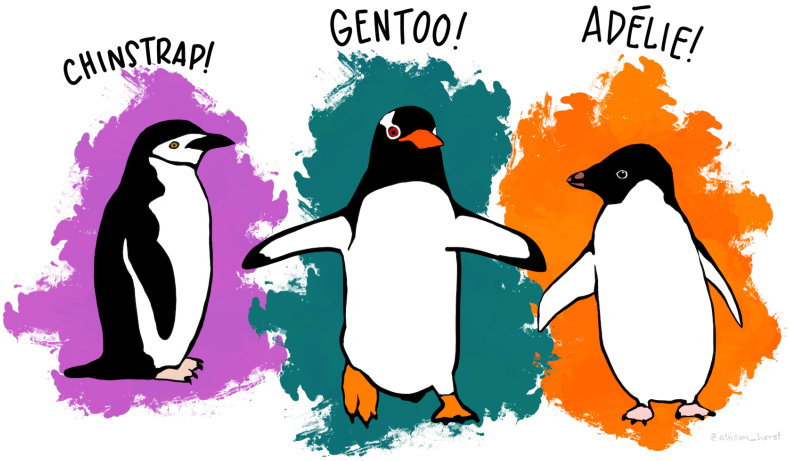


Figura 1: Ilustración de las tres especies de pingüinos del archipiélago Palmer (Artista @allison_horst)

Los datos fueron originalmente publicados en Gorman, Williams y Fraser [1]. Este conjunto de datos se hizo popular a partir de la creación del paquete palmerpenguins de R. Hoy en día los datos de los pingüinos del archipiélago Palmer se usan de forma extendida para ilustrar las técnicas de exploración y visualización de datos no solo en R, sino en muchos otros lenguajes de programación para estadística y ciencia de datos, como Python. Nosotros accederemos a los datos a través de este enlace, que proporciona los datos en formato CSV (*comma separated values*).

Objetivos

Aprenderás en concreto a calcular las medidas descriptivas más representativas de las características de interés y a crear diferentes tipos de gráficos o

visualizaciones.

1. Librerías

Lo primero que necesitamos hacer es importar las librerías de Python que usaremos a lo largo de práctica, que son `pandas` y `seaborn`:

```
import pandas as pd
import seaborn as sns
```

1.1. `pandas`

`pandas` es una librería que permite leer datos almacenados en estructuras similares a una tabla, como las hojas de cálculo o los archivos CSV, y proporciona métodos para explorar y describir esos datos.

Usando esta librería podremos calcular por ejemplo el peso medio de los pingüinos de cada especie. Obtendremos esta tabla:

species	body_mass_g
Adelie	3700.662252
Chinstrap	3733.088235
Gentoo	5076.016260

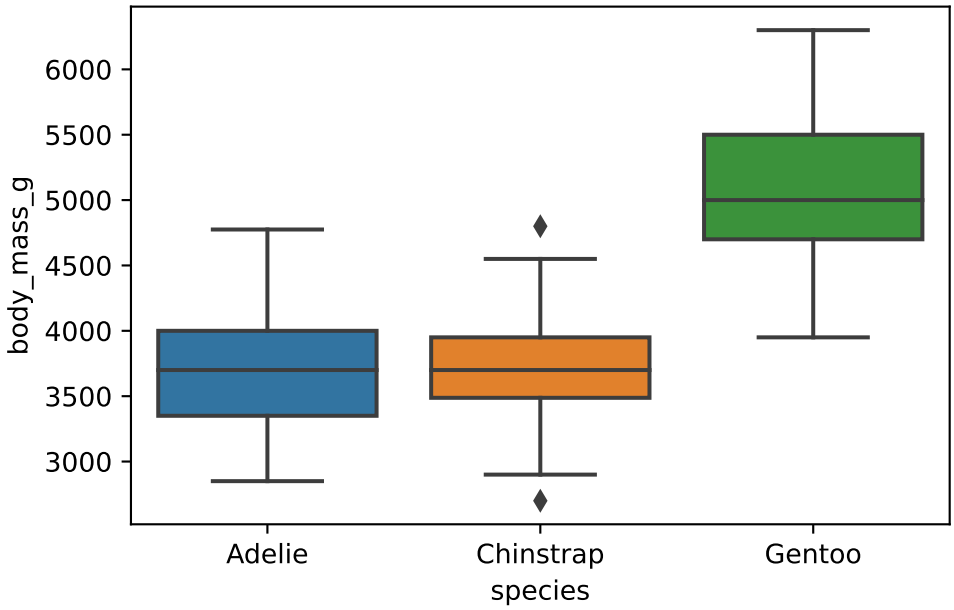
Puedes consultar la documentación oficial de `pandas` aquí.

1.2. `seaborn`

`seaborn` es una librería para visualización de datos. Permite crear gráficos estadísticos muy informativos y visualmente atractivos con pocas líneas de código. Y está diseñado para trabajar con las estructuras de datos creadas con `pandas`.

Usaremos `seaborn` para realizar diferentes tipos de gráficos como diagramas de barras, histogramas, diagramas de caja y bigotes etc.

Aprenderemos por ejemplo a crear el siguiente gráfico, con los diagramas de caja y bigotes para el peso de los pingüinos de cada especie.



Puedes consultar la documentación oficial de `seaborn` aquí.

2. Datos

En esta sección importarás los datos sobre los pingüinos del archipiélago Palmer presentados en la introducción y conocerás la información que contienen.

2.1. Importar los datos

Como se indicó en la introducción, los datos con los que vamos a trabajar están disponibles en la web en un fichero de formato `CSV`.

Ejecuta las instrucciones a continuación para importar el archivo usando la función `read_csv()` y guardar el resultado en una variable de nombre `penguins`:

```
url =
 ↪  "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/penguins.cs
penguins = pd.read_csv(url)
```

El objeto `penguins` que acabas de crear es una **hoja de datos**, representada en `pandas` con la clase `DataFrame`.

```
type(penguins)
```

```
pandas.core.frame.DataFrame
```

En los siguientes apartados aprenderás a realizar una exploración inicial de la hoja de datos `penguins` que acabas de crear para conocer su estructura y la información que contiene.

2.2. Dimensiones

Una hoja de datos es una estructura matricial o tabular que contiene datos organizados por filas y columnas.

Para saber las dimensiones de nuestra hoja de datos `penguins` consulta su propiedad `shape`:

```
penguins.shape
```

```
(344, 7)
```

Vemos que nuestra hoja de datos tiene `344` filas y `7` columnas.

2.3. Primeras y últimas filas

Con las siguientes instrucciones puedes visualizar las cinco primeras y últimas filas de la hoja de datos `penguins` que acabas de crear.

```
penguins.head(5)
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_
0	Adelie	Torgersen	39.1	18.7	181.0	375
1	Adelie	Torgersen	39.5	17.4	186.0	380
2	Adelie	Torgersen	40.3	18.0	195.0	325
3	Adelie	Torgersen	NaN	NaN	NaN	Na
4	Adelie	Torgersen	36.7	19.3	193.0	345

```
penguins.tail(5)
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_
339	Gentoo	Biscoe	NaN	NaN	NaN	Na
340	Gentoo	Biscoe	46.8	14.3	215.0	4850
341	Gentoo	Biscoe	50.4	15.7	222.0	5750
342	Gentoo	Biscoe	45.2	14.8	212.0	5200
343	Gentoo	Biscoe	49.9	16.1	213.0	5400

2.4. Estructura

En nuestra hoja de datos `penguins`:

- Cada columna representa una variable asociada a una propiedad o característica de los pingüinos. Por ejemplo, la primera columna, de nombre `species` indica la especie (Chinstrap, Adélie o Gentoo) de pingüino. En el siguiente apartado se describen las otras seis variables.
- Cada fila se corresponde con un pingüino concreto de los 344 seleccionados en el estudio.
- Cada celda contiene el valor de la característica del pingüino en la correspondiente fila.

Por ejemplo, mirando la primera fila de la hoja de datos

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_
0	Adelie	Torgersen	39.1	18.7	181.0	375

vemos en la primera celda que el primer pingüino del listado es de la especie Adelie.

Unos mismos datos pueden organizarse o presentarse de diferentes maneras en diferentes hojas de datos. Para que sea sencillo trabajar con una hoja de datos es conveniente que haya una relación clara entre su significado y su estructura. Se considera que la hoja de datos está *ordenada* o *limpia* (en inglés se habla de *tidy data*) si está organizada de acuerdo con los siguientes principios:

- Cada **columna** representa una **variable** o característica de interés.
- Cada **fila** representa una **observación**, caso o unidad experimental.
- Cada **celda** contiene un **valor**, el de la variable en la correspondiente columna para la observación en la correspondiente fila.

De acuerdo con la descripción inicial, nuestra hoja de datos cumple con los principios anteriores.

2.5. Variables

Ejecuta la siguiente instrucción:

```
penguins.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

La salida del método `info()` nos da una tabla con información sobre las siete variables de nuestra hoja de datos.

2.5.1. Descripción

En la columna de la tabla de nombre `Column` se lista el nombre de las siete variables en `penguins`. El significado de las variables es el siguiente:

Nombre	Descripción
`species`	Especie de pingüinos (Chinstrap, Adélie o Gentoo)
`island`	Nombre de la isla del archipíelago Palmer (Dream, Torgersen o Biscoe)
`bill_length_mm`	Longitud del pico, en milímetros (ver Figura 2)
`bill_depth_mm`	Anchura del pico, en milímetros (ver Figura 2)
`flipper_length_mm`	Longitud de las alas
`body_mass_g`	Peso en gramos
`sex`	Sexo (MALE o FEMALE)

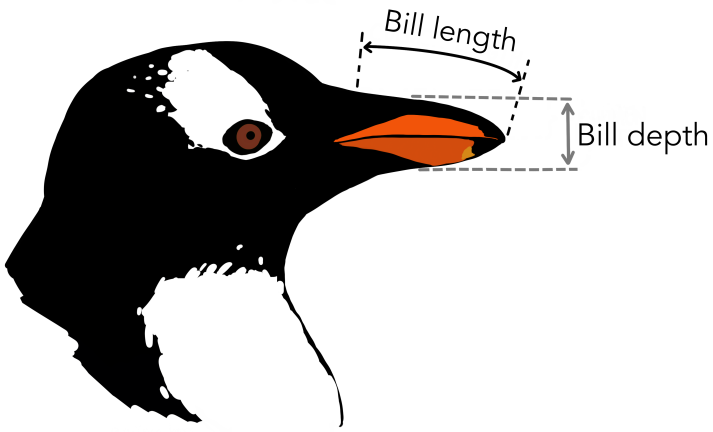


Figura 2: Ilustración de las variables `bill_length_mm` y `bill_depth_mm` (Artista @allison_horst)

Volviendo a mirar la primera fila de nuestra hoja de datos

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass
0	Adelie	Torgersen	39.1	18.7	181.0	375

ahora sabes que el primer pingüino es de la especie Adelie, vive en la isla Torgensen, las dimensiones de su pico son 39.1×18.7 milímetros, sus alas miden 181 milímetros, pesa 3 kilos y 750 gramos, y es un macho.

Problema 1

Describe las características del tercer pingüino del estudio (índice 2).

2.5.2. Valores nulos

Fíjate ahora en la columna `Non-Null Count` de la salida del método `info()`:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

Los valores nulos o perdidos son valores no disponibles, que no han podido registrarse, se representan con el símbolo `NaN` (iniciales de *Not A Number*).

Si vuelves a mirar las cinco primeras filas de la hoja de datos verás que para el cuarto pingüino sólo sabemos que es de la especie Adelie y vive el la isla Torgersen, y se desconocen las otras cinco variables.

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass
3	Adelie	Torgersen	NaN	NaN	NaN	N

Los diferentes métodos de la librería `pandas` contemplan la posibilidad de que las hojas de datos tengan valores nulos y los tratan de un forma predeterminada (por ejemplo, la media los ignora por defecto).

2.5.3. Tipos de variables

Las siete variables de nuestra hoja de datos se dividen en dos clases:

1. `species`, `island` y `sex` son **variables categóricas**. Este tipo de variables representan una característica cualitativa que puede tomar un número finito y fijo de valores, denominados categorías o niveles.

2. Las cuatro restantes, `bill_length_mm`, `bill_depth_mm`, `flipper_length_mm` y `body_mass_g` son **variables numéricas**, que representan características cuantitativas que se describen con valores numéricos (números enteros o reales).

`pandas` asigna un tipo a cada variable de una hoja de datos en función de los valores que presenta, como puede verse en la columna `Dtype` de la salida del método `info()`:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

2.6. Índice

Igual que cada columna (variable) en una hoja de datos tiene un nombre, cada fila (observación) también tiene una etiqueta identificativa. En nuestra hoja de datos cada uno de los 344 pingüinos se identifica con un número entero de la secuencia 0, 1, . . . , 333.

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass
0	Adelie	Torgersen	39.1	18.7	181.0	375
1	Adelie	Torgersen	39.5	17.4	186.0	380
2	Adelie	Torgersen	40.3	18.0	195.0	325

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass
341	Gentoo	Biscoe	50.4	15.7	222.0	575
342	Gentoo	Biscoe	45.2	14.8	212.0	520
343	Gentoo	Biscoe	49.9	16.1	213.0	540

Las etiquetas identificativas de las filas de una hoja de datos forman su **índice**. El índice de una hoja de datos de `pandas` se registra en su propiedad `index`.

```
penguins.index
```

```
RangeIndex(start=0, stop=344, step=1)
```

Si cada pingüino estuviera identificado por un código, podríamos haber indicado esa variable como índice en el momento de la importación de los datos. Cuando no se indica el índice de una hoja de datos de forma explícita, `pandas` asigna una secuencia de números enteros comenzando en 0, como ha ocurrido en nuestro caso.

3. Seleccionar variables

Al estudiar un conjunto de datos, es frecuente tener que seleccionar los datos relevantes para responder a las diferentes cuestiones planteadas.

Si por ejemplo queremos saber cuál es el peso máximo de todos los pingüinos del estudio, seleccionaremos la variable `body_mass_g` y después calcularemos su máximo.

En esta sección aprenderás los métodos para seleccionar variables de una hoja de datos.

3.1. Seleccionar una variable

Utiliza la siguiente instrucción para seleccionar la variable `body_mass_g`:

```
mass = penguins["body_mass_g"]
```

> Para seleccionar una sola variable, usa corchetes `[]` e indica el nombre de la columna de interés.

Ahora podemos aplicar la función `max()` para obtener el peso máximo:

```
mass.max()
```

```
6300.0
```

Vemos que el pingüino más pesado del estudio pesa 6 kilos y 300 gramos.

Podemos realizar las dos operaciones, seleccionar la variable `body_mass_g`, y calcular su máximo con una sola instrucción:

```
penguins["body_mass_g"].max()
```

```
6300.0
```

Problema 2

Calcula el peso medio de todos los pingüinos (función `mean()`).

Problema 3

Calcula el valor mínimo para la longitud de las alas de todos los pingüinos (función `min()`).

3.2. Seleccionar una lista de variables

Para seleccionar las dos variables relativas a las dimensiones del pico, `bill_length_mm` y `bill_depth_mm`, ejecuta la siguiente instrucción:

```
bill = penguins[["bill_length_mm", "bill_depth_mm"]]
```

> **i** Para seleccionar una lista de variables, usa corchetes `[]` adicionales para crear la lista con los nombres de las columnas de interés (los corchetes exteriores indican que se van a seleccionar datos y los interiores crean la lista).

Ahora podemos calcular la media para ambas variables con

```
bill.mean()
```

	0
bill_length_mm	43.92193
bill_depth_mm	17.15117

Problema 4

Calcula el número de observaciones no nulas (función `count()`) para las variables `species` y `body_mass_g` con una sola línea de código.

3.3. `DataFrame` vs `Series`

Ejecutando

```
type(bill)
```

```
pandas.core.frame.DataFrame
```

vemos que el objeto `bill` que hemos creado antes al extraer las dos variables sobre las dimensiones del pico es de tipo `DataFrame`, igual que la hoja de datos original `penguins`.

Pero ejecutando

```
type(mass)
```

```
pandas.core.series.Series
```

vemos que el objeto `mass` que hemos creado antes al extraer la variable `body_mass_g` es de un nuevo tipo, llamado `Series`.

`Series` es el tipo de datos que usa `pandas` para almacenar vectores unidimensionales, representando un conjunto de observaciones de una variable.

Podemos usar el método `head()` para ver los cinco primeros valores de la variable:

```
mass.head(5)
```

	body_mass_g
0	3750.0
1	3800.0
2	3250.0
3	NaN
4	3450.0

Aprecia que `mass` tiene el mismo índice que la hoja de datos `penguins`:

```
mass.index
```

```
RangeIndex(start=0, stop=344, step=1)
```

Podemos ver una hoja de datos (`DataFrame`) como un conjunto de variables (`Series`), dispuestas en columnas, que comparten un índice común.

Problema 5

Averigua si los siguientes objetos son de tipo `Series` o `DataFrame`:

1. `penguins["flipper_length_mm"]`
2. `penguins[["flipper_length_mm"]]`

Generalmente al aplicar un método propio de la clase `Series` a un objeto de clase `DataFrame`, se obtiene el mismo resultado que si se aplicara el método a cada una de las variables de la hoja de datos. Como hicimos antes al calcular simultáneamente la media de las dos variables en la hoja de datos `bill` con `bill.mean()`.

Pero hay métodos propios de `Series` que no pueden aplicarse a `DataFrame`, métodos propios de `DataFrame` que no pueden aplicarse a `Series`, y métodos comunes a `Series` y `DataFrame` pero que requieren diferentes argumentos en

cada caso. Por ejemplo, el método `Series.sort_values()` ordena los valores de una variable de mayor a menor. Puedes comprobarlo con

```
mass.sort_values().head(5)
```

	body_mass_g
190	2700.0
64	2850.0
58	2850.0
116	2900.0
98	2900.0

Sin embargo el método `DataFrame.sort_values()` no puede aplicarse sin argumentos.

Problema 6
Verifica que al ejecutar `bill.sort_values()` se obtiene un error e intenta interpretarlo y corregirlo.

4. Continuará . . .

Bibliografía

[1] KB Gorman, TD Williams y WR Fraser. "Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus Pygoscelis)". en. En: *PLoS ONE* 9.3 (2014). DOI: 10.1371/journal.pone.0090081.

[2] Wes McKinney. "Data Structures for Statistical Computing in Python". En: *Proceedings of the 9th Python in Science Conference*. Ed. por Stéfan van der Walt y Jarrod Millman. 2010, págs. 56-61. DOI: 10.25080/Majora-92bf1922-00a.

[3] The pandas development team. *pandas-dev/pandas: Pandas*. Ver. latest. Feb. de 2020. DOI: 10.5281/zenodo.3509134. URL: 10.5281/zenodo.3509134.

[4] Michael L. Waskom. "seaborn: statistical data visualization". En: *Journal of Open Source Software* 6.60 (2021), pág. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.