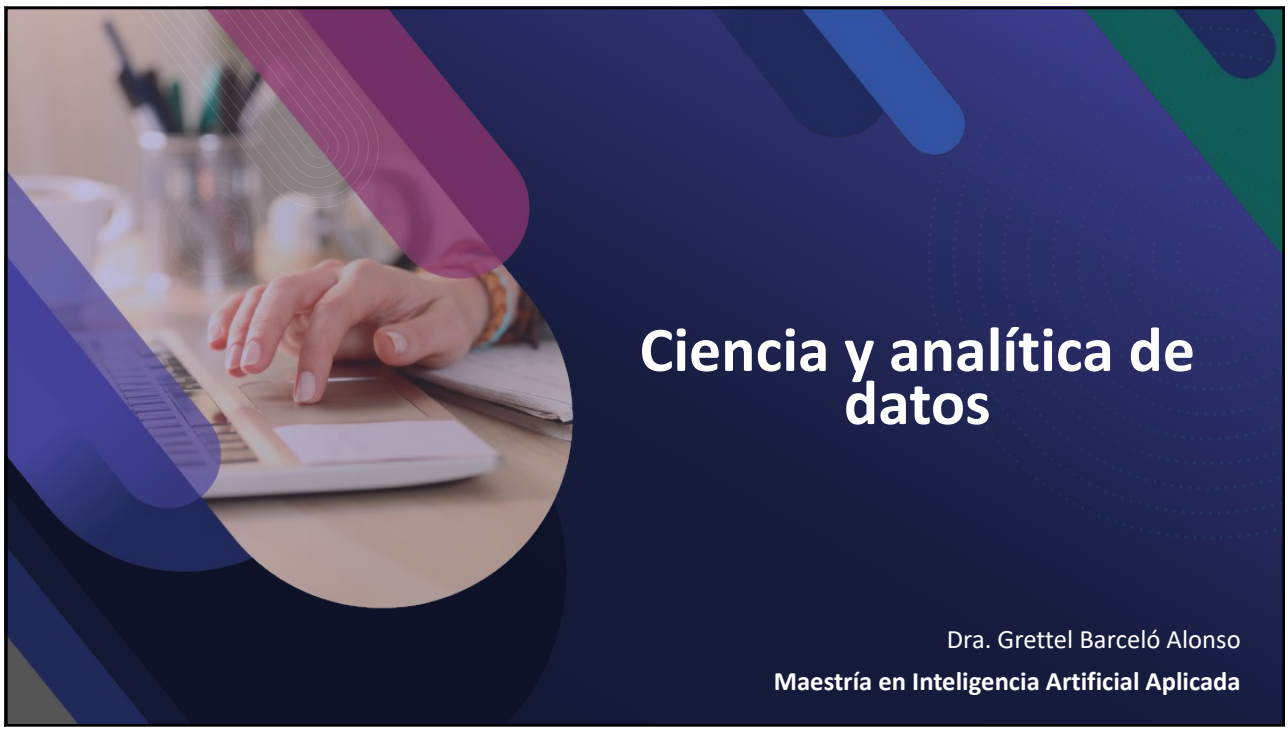

Ciencia y analítica de datos
Dra. Grettel Barceló Alonso
Maestría en Inteligencia Artificial Aplicada

1


Active class
Sesión 2

2

Módulo 3

Análisis, visualización y transformación de datos

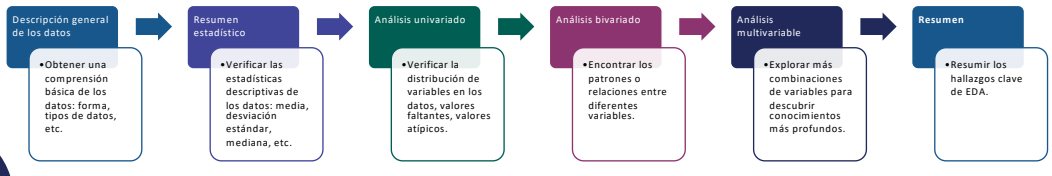
3

Análisis exploratorio de datos

Con EDA (*Exploratory Data Analysis*), se describen los datos utilizando técnicas estadísticas y de visualización para hacer enfoque en sus aspectos más relevantes.

- Primer paso en cualquier análisis.
- Descubrir la estructura subyacente de los datos.
- Inferir la mejor estrategia para la limpieza y el preprocesamiento de los datos.

Pasos de EDA



4

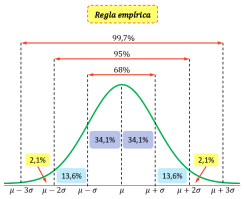
Análisis exploratorio de datos

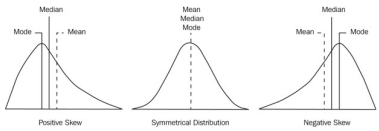
Medidas

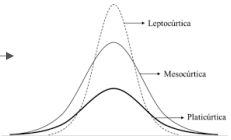
Tendencia central
- Media `mean()`
- Mediana `median()`
- Moda `mode()`

Variabilidad
- Rango `max() - min()`
- Varianza `var()`
- Desviación estándar `std()`

Forma
- Sesgo `skew()`
- Curtosis `kurt()`







Valores faltantes

Cuando no se almacena ningún valor de datos para la variable en una observación `isna()`

Valores atípicos

Observaciones significativamente diferentes del resto.

- Límite arbitrario
- Media y desviación estándar
- Método IQR

5

Visualización en Python

PyViz

- La visualización de datos permite ver la información de forma comprensible y cohesiva, facilitando la identificación de patrones y la comunicación asertiva de las conclusiones o descubrimientos.
- Pero el panorama de visualización de Python puede parecer abrumador al principio.
- Se ha creado PyViz.org, un sitio para ayudar a los usuarios a decidir cuáles son las mejores herramientas de visualización de código abierto de Python para sus propósitos.

https://pyviz.org/overviews/index.html

6

Plataformas

Matplotlib es una de las denominadas bibliotecas núcleo (*core*), sobre la que se construyen varias plataformas de nivel superior.

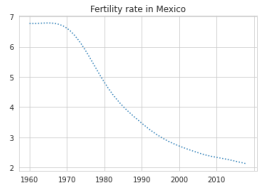
Tiene una API completa y potente, que permite perzonalizar cualquier atributo de la figura

Las plataformas de alto nivel que ocupan Matplotlib, proporcionan una API más simple para cubrir las tareas más comunes de manera concisa y conveniente. Dos de las más usadas son:

- API .plot () de Pandas
- Seaborn

7

Trazado básico



Un gráfico de líneas se utiliza para visualizar información que cambia continuamente con el tiempo.

Para utilizar estas plataformas debes importarlas en los *scripts* de Python

- import matplotlib.pyplot as plt
- import pandas as pd
- import seaborn as sns

Y a través de ellas llamar a los métodos o atributos

Por ejemplo, la función de graficado básica es plot(x,y), que grafica valores de *y* contra *x* como líneas y/o marcadores

- Con Matplotlib: plt.plot(df.index, df[columna])
- Con Pandas: df.plot()
- Con Seaborn: sns.lineplot(x=df.index, y=df[columna])

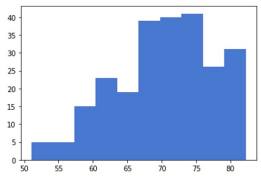
8

Histogramas

- Un histograma es una representación en barras de la distribución de los datos.
- En el eje horizontal se indican los valores de la variable y en el vertical sus frecuencias.
- Las frecuencias se agrupan en clases o bins.
- Utiliza este tipo de diagrama cuando desees observar el grado de homogeneidad o variabilidad de las columnas cuantitativas continuas del dataframe.

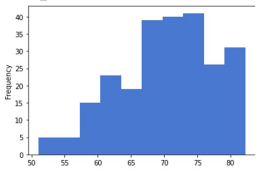
Con Matplotlib

```
plt.hist(lifexp_male['2019'])
```



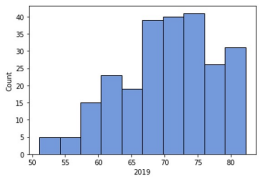
Con Pandas

```
lifexp_male['2019'].plot.hist()
lifexp_male['2019'].plot(kind='hist')
```



Con Seaborn

```
sns.histplot(x=lifexp_male['2019'])
```

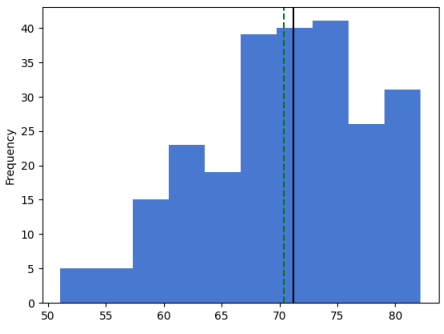


9

Histogramas

Para incluir el promedio y la mediana en un histograma puedes usar la función `axvline()` de Matplotlib:

```
plt.axvline(x=lifexp_male['2019'].mean(), color='green', linestyle='--')
plt.axvline(x=lifexp_male['2019'].median(), color='black', linestyle='-')
```



10

Diagrama de barras

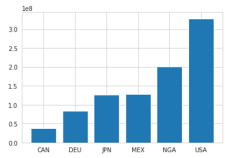
Los gráficos de barras se utilizan para mostrar datos categóricos o cuantitativos discretos, con barras rectangulares de longitudes proporcionales a los valores que representan.

Estos valores pueden ser:

1. El total, promedio u otra medida de resumen de cada categoría
2. El conteo o frecuencia de cada categoría

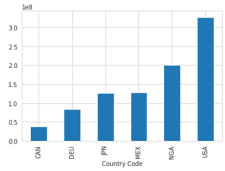
Con Matplotlib

```
plt.bar(population.index, population['2019'])
```



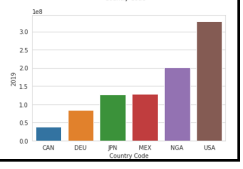
Con Pandas

```
population['2019'].plot(kind='bar')
population ['2019'].plot.bar()
```



Con Seaborn

```
sns.barplot(x=population.index, y=population['2019'])
```



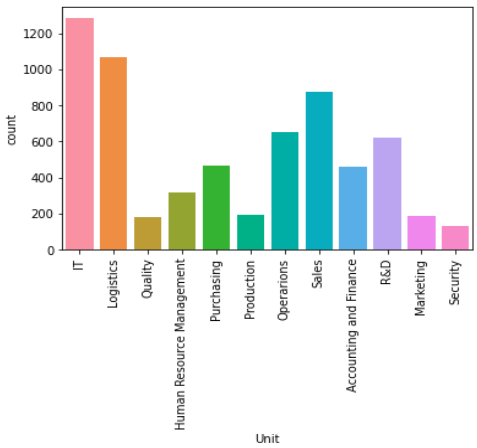
11

Diagrama de barras

Seaborn ofrece además un gráfico de recuento, con variables categóricas, que permite realizar el agrupamiento para *count* de manera automática:

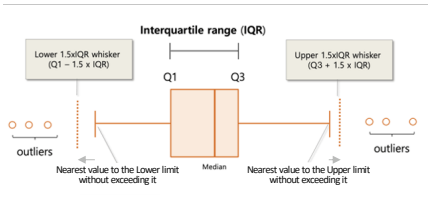
```
sns.countplot(columna)
```

Con pandas y Matplotlib habría que usar groupby() o value_counts() previo al trazado.



12

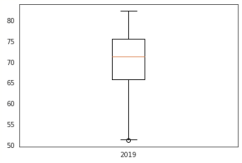
Diagramas de caja y bigote



- Los diagramas de caja y bigote (*boxplot*) se utilizan para mostrar la distribución de datos cuantitativos continuos.
- Ofrecen un panorama de la distribución de dichos valores, a través de sus cuartiles. Para ello, utilizan como representación una caja y segmentos (bigotes) que delimitan los intervalos donde la variable continua concentra la mayoría de las observaciones.

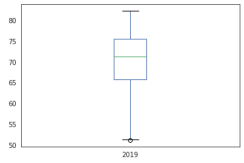
Con Matplotlib

```
plt.boxplot(lifexp_male['2019'].dropna(),
labels=['2019'])
```



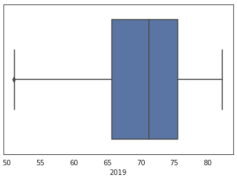
Con Pandas

```
lifexp_male['2019'].plot.box()
lifexp_male['2019'].plot(kind='box')
```



Con Seaborn

```
sns.boxplot(x=lifexp_male['2019'])
```

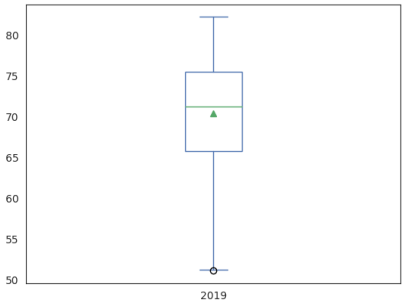


13

Diagramas de caja y bigote

Para añadir el promedio a un boxplot puedes usar el parámetro `showmeans()`

- `plt.boxplot(lifexp_male['2019'], showmeans=True)`
- `lifexp_male['2019'].plot.box(showmeans=True)`
- `sns.boxplot(x=lifexp_male['2019'], showmeans=True)`



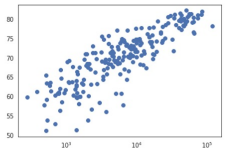
14

Diagrama de dispersión

- Los diagramas de dispersión (*scatter plot*) muestran la relación entre dos variables.
- Utilizan como representación un conjunto de puntos ubicados en coordenadas cartesianas, según los valores de las dos variables.
- De estos puntos se puede notar si las dos aumentan a la vez (correlación positiva), si una aumenta mientras la otra disminuye (correlación negativa) o si no tienen relación alguna (correlación nula).

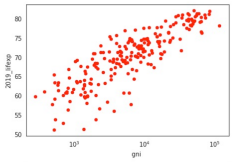
Con Matplotlib

```
plt.scatter(x=population['gni'],y=population['lifexp'])
```



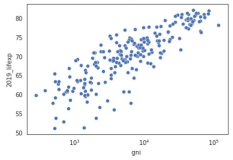
Con Pandas

```
population.plot(kind='scatter',x='gni',y='lifexp',c='red')
```



Con Seaborn

```
sns.scatterplot(x=population['gni'],y=population['lifexp'])
```



15

Diagrama de dispersión

El coeficiente que se calcula por defecto es el de Pearson (medida de dependencia lineal)

$$r = \frac{cov[X,Y]}{\sqrt{var[X]var[Y]}}$$

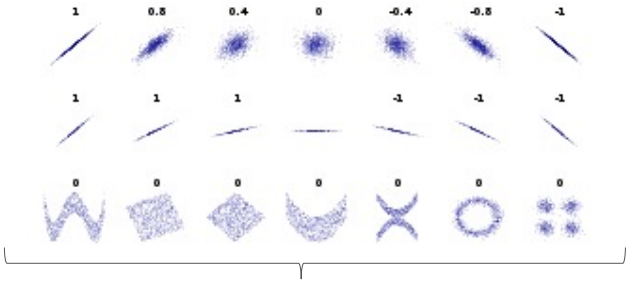
$$r = \frac{\sum(x_i - x)(y_i - y)}{\sqrt{\sum(x_i - x)^2 \sum(y_i - y)^2}}$$

x_i - Valores de la variable *x*

y_i - Valores de la variable *y*

x - Promedio de la variable *x*

y - Promedio de la variable *y*



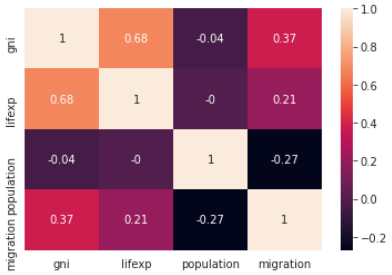
Varios grupos de puntos, con el coeficiente de correlación (*r*) para cada grupo.

Nótese que la correlación refleja la no linealidad y la dirección de la relación lineal. En la figura del centro, la varianza de *y* es nula, por lo que la correlación es indeterminada. Lo mismo sucedería con una línea vertical.

16

Diagramas de calor

- Seaborn además complementa las matrices de dispersión con mapas de calor (*heatmap*), incluyendo en estos últimos la correlación numérica entre el par de variables.
- Puesto que en la diagonal se encuentra relacionada una variable con ella misma, el coeficiente de correlación es 1.

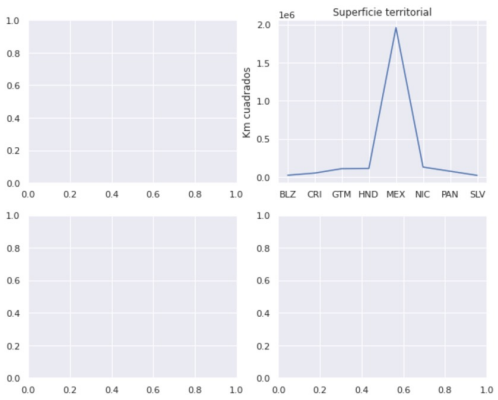



```
sns.heatmap(round(df.corr(numeric_only=True),2),
            annot=True)
```

17

Subgráficas

- Las subgráficas y se ocupan para conjuntar visualizaciones de la misma o diversa naturaleza y sintetizar la información en una única figura.
- Pueden ser colocadas en una dirección o en dos direcciones.
- Para crear la matriz explícitamente, se usa la función subplots() de Matplotlib.
- Dentro de la matriz puede generarse cada gráfica de manera independiente usando cualquiera de las plataformas estudiadas.




```
fig,axs = plt.subplots(2,2,figsize=(10,8))
axs[0,1].plot(centralAmerica['SurfaceArea'])
axs[0,1].set_title('Superficie territorial')
axs[0,1].set_ylabel('Km cuadrados')
```

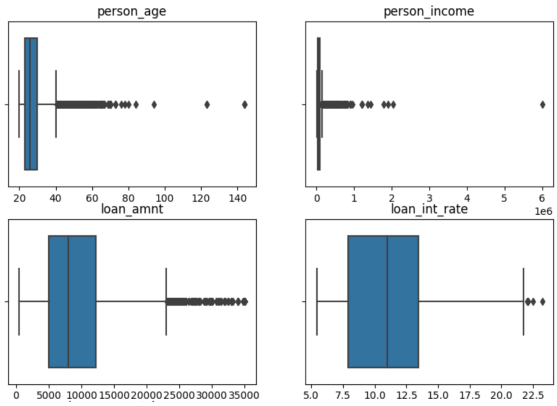
18

Subgráficas

Si todas las subgráficas serán de un mismo tipo, se puede usar un ciclo:

```
fig, axes = plt.subplots(2, 2)
axes = axes.ravel()

for col, ax in zip(df.columns, axes):
    sns.boxplot(x=df[col], ax=ax)
    ax.set(title=f'{col}', xlabel=None)
```



19

Tecnológico de Monterrey

21