09481: Inteligencia Artificial

Profesor del curso: Breyner Posso, Ing. M.Sc.
e-mail: breyner.posso1@u.icesi.edu.co

Programa de Ingeniería de Sistemas.
Departamento TIC.
Facultad de Ingeniería.
Universidad Icesi.
Cali, Colombia.

Agenda

- Análisis exploratorio de datos (EDA).

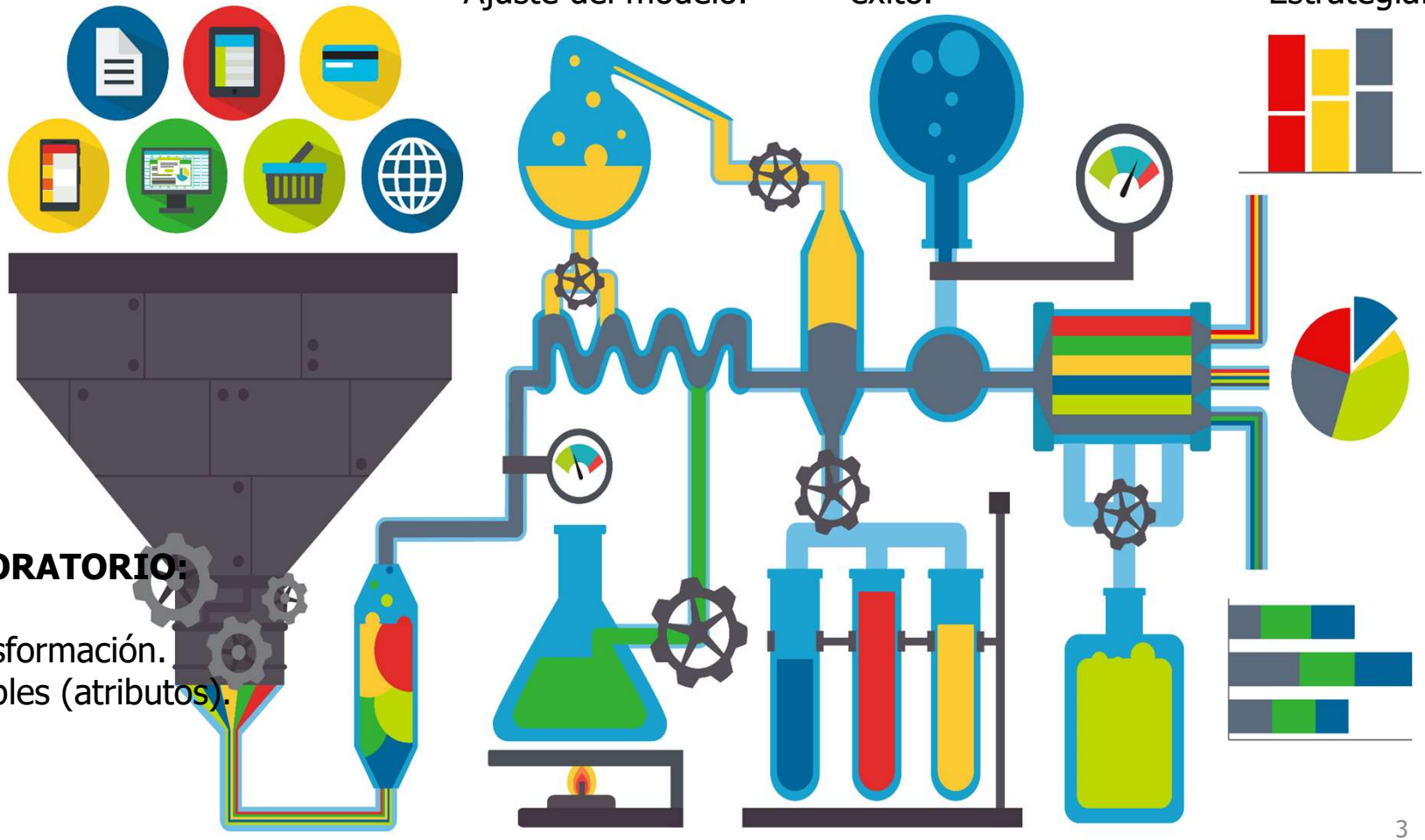
DATOS:
Materia prima.

MODELO:
Implementación del modelo de analítica.
Ajuste del modelo.

EVALUACION:
Viabilidad del negocio.
Validación criterios de éxito.

DESPLIEGUE:
Resultados.
Conocimiento.
Estrategia.

PREGUNTAS:
¿Cómo?
¿Cuáles?
¿Cuándo?
¿Por qué? *



ANALISIS EXPLORATORIO:
Limpieza de datos.
Preparación / transformación.
Selección de variables (atributos).

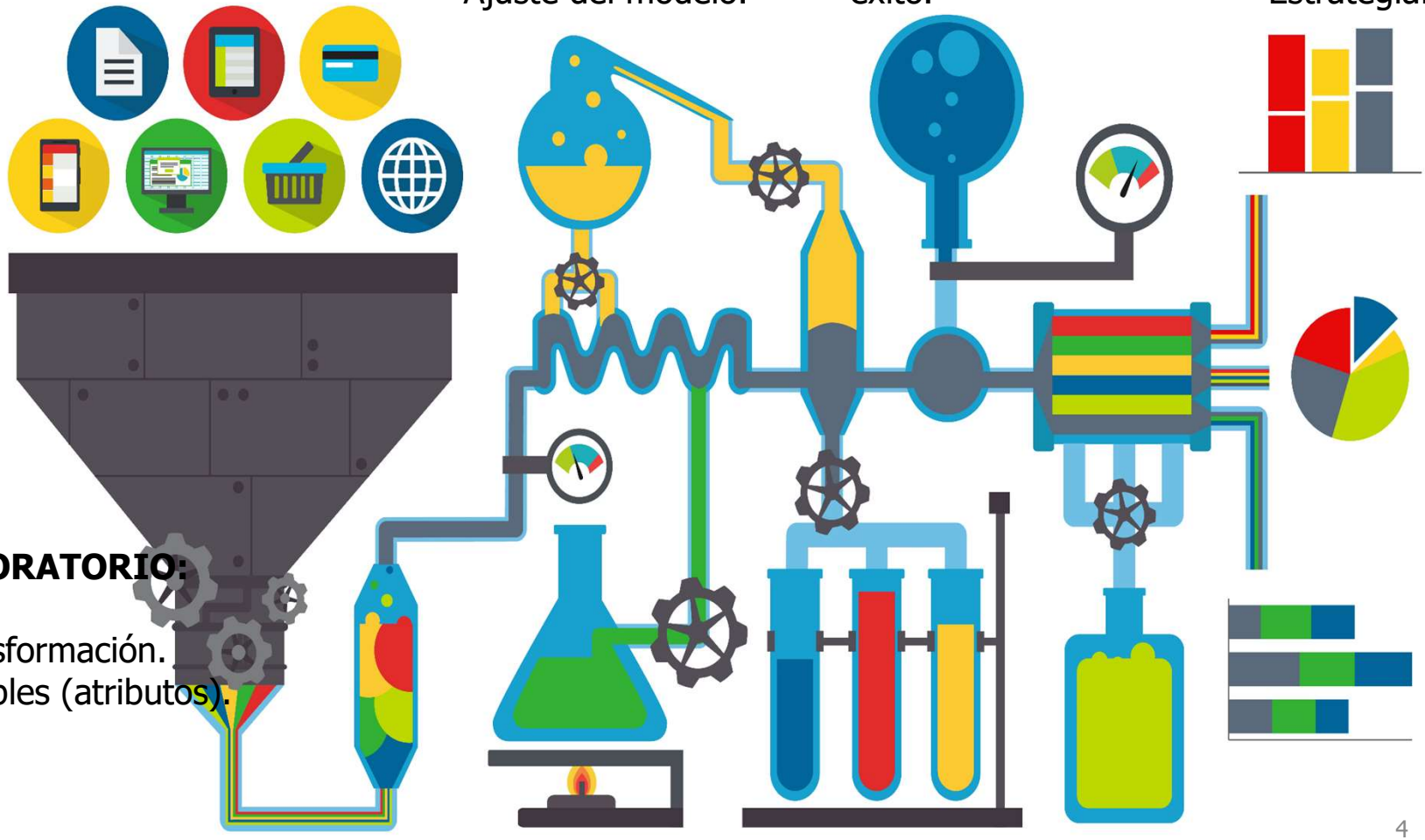
DATOS:
Materia prima.

MODELO:
Implementación del modelo de analítica.
Ajuste del modelo.

EVALUACION:
Viabilidad del negocio.
Validación criterios de éxito.

DESPLIEGUE:
Resultados.
Conocimiento.
Estrategia.

PREGUNTAS:
¿Cómo?
¿Cuáles?
¿Cuándo?
¿Por qué? *



ANALISIS EXPLORATORIO:
Limpieza de datos.
Preparación / transformación.
Selección de variables (atributos).

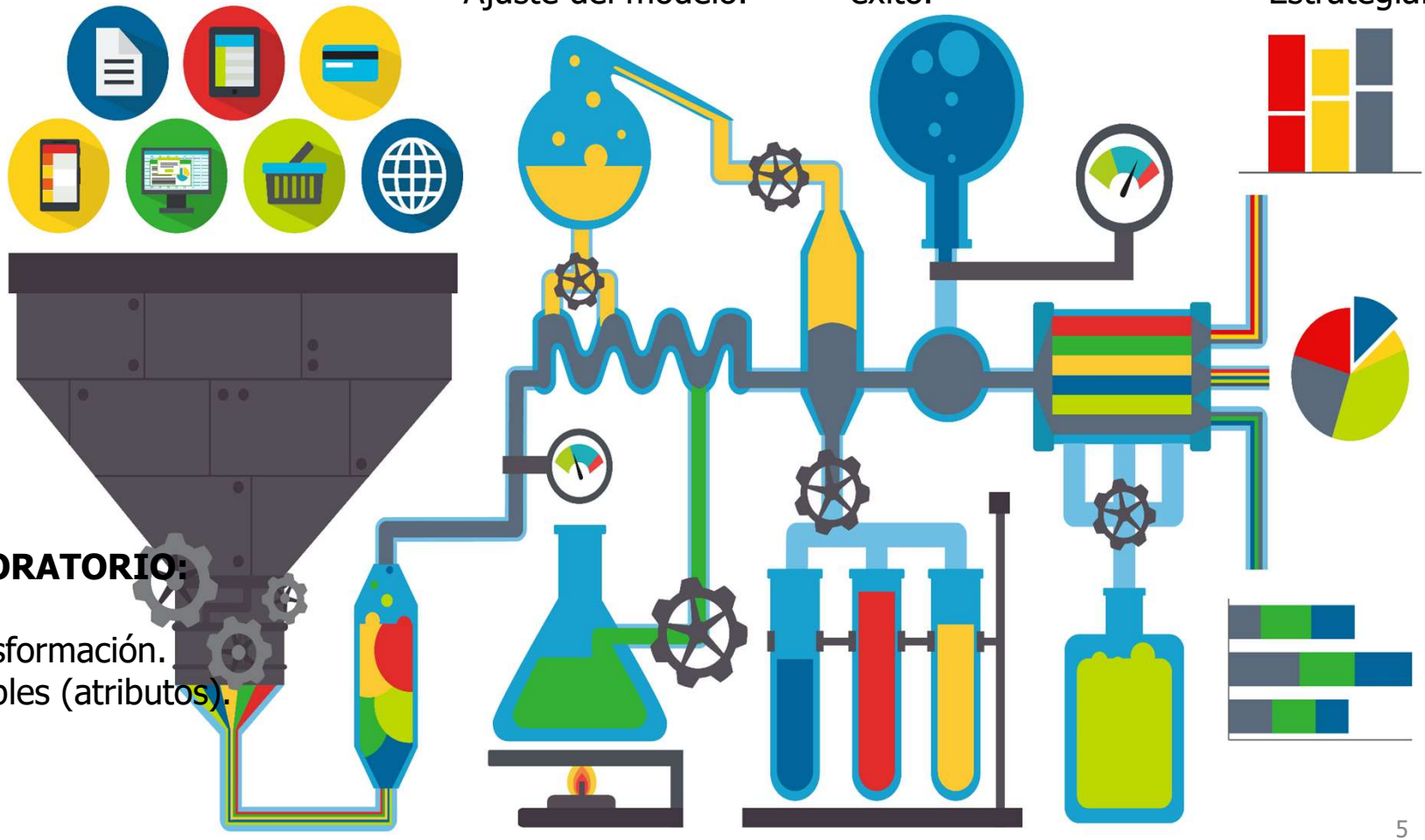
DATOS:
Materia prima.

MODELO:
Implementación del modelo de analítica.
Ajuste del modelo.

EVALUACION:
Viabilidad del negocio.
Validación criterios de éxito.

DESPLIEGUE:
Resultados.
Conocimiento.
Estrategia.

PREGUNTAS:
¿Cómo?
¿Cuáles?
¿Cuándo?
¿Por qué? *



ANALISIS EXPLORATORIO:
Limpieza de datos.
Preparación / transformación.
Selección de variables (atributos).

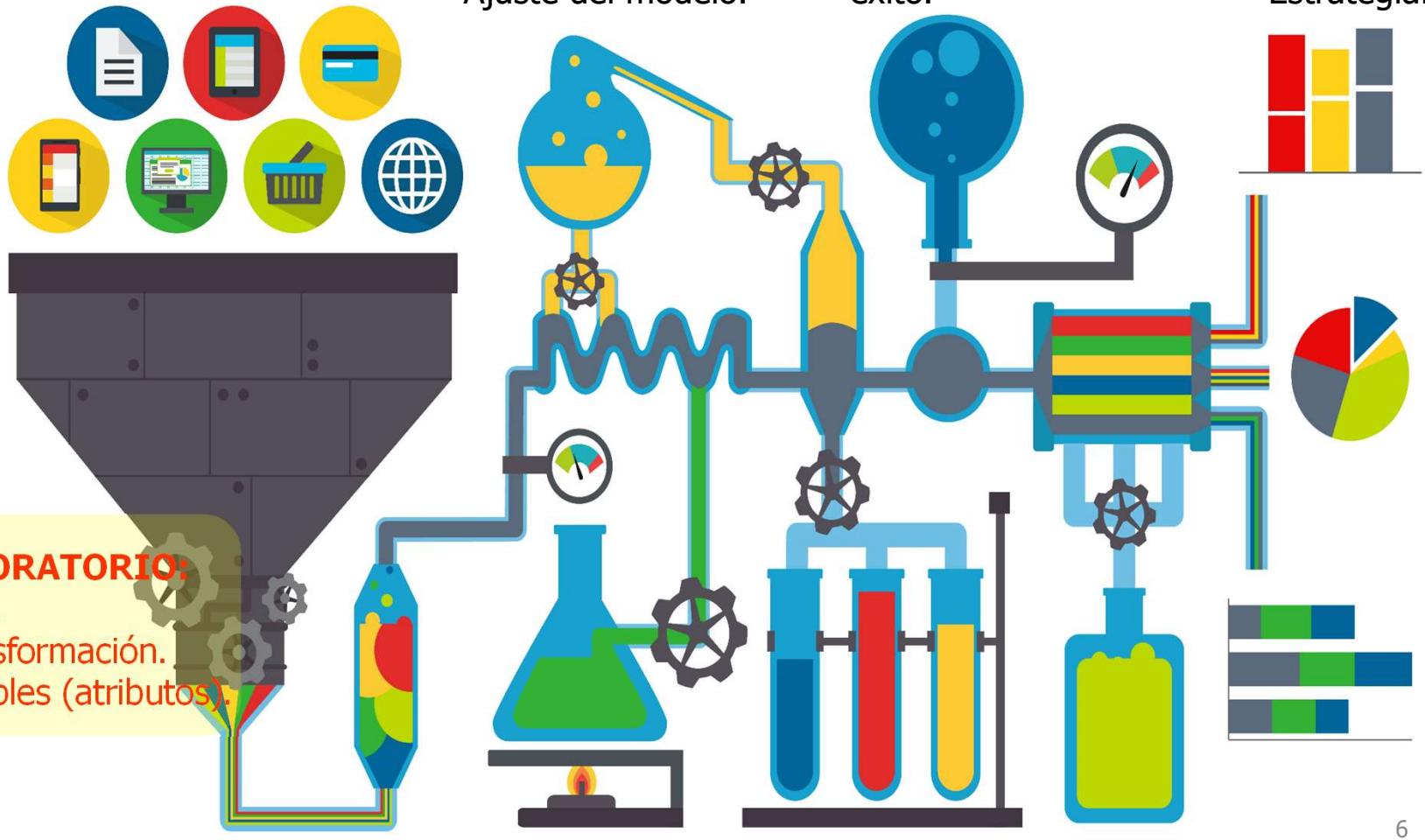
DATOS:
Materia prima.

MODELO:
Implementación del modelo de analítica.
Ajuste del modelo.

EVALUACION:
Viabilidad del negocio.
Validación criterios de éxito.

DESPLIEGUE:
Resultados.
Conocimiento.
Estrategia.

PREGUNTAS:
¿Cómo?
¿Cuáles?
¿Cuándo?
¿Por qué? *



ANALISIS EXPLORATORIO:
Limpieza de datos.
Preparación / transformación.
Selección de variables (atributos).

Key Terms for Data Types

Numeric

Data that are expressed on a numeric scale.

Continuous

Data that can take on any value in an interval. (*Synonyms*: interval, float, numeric)

Discrete

Data that can take on only integer values, such as counts. (*Synonyms*: integer, count)

Categorical

Data that can take on only a specific set of values representing a set of possible categories. (*Synonyms*: enums, enumerated, factors, nominal)

Binary

A special case of categorical data with just two categories of values, e.g., 0/1, true/false. (*Synonyms*: dichotomous, logical, indicator, boolean)

Ordinal

Categorical data that has an explicit ordering. (*Synonym*: ordered factor)

Análisis exploratorio de datos [1/2]

Una vez se tiene un conjunto de datos, es necesario entenderlo para:

- Estimar si puede servir para responder la pregunta de investigación.
- Identificar relaciones entre las variables.
- Identificar patrones y tendencias en los datos.
- Identificar problemas en la calidad de los datos y establecer como lidiar con ellos. Algunos problemas comunes incluyen:
  - Datos faltantes (campos vacíos, nan).
  - Datos anómalos (*outliers*).
  - Datos repetidos (atributos repetidos, observaciones repetidas).
  - Problemas de escala en los valores (e.g.: unos atributos con valores muy grandes y otros con valores muy pequeños).
  - Problemas con los tipos de datos (asignación errónea de enteros, flotantes, cadenas de caracteres, fechas, horas, ubicaciones geográficas, etc..).

Análisis exploratorio de datos [2/2]

Los siguientes comandos de la librería Pandas de Python nos permite entender mejor los datos cuando estos se encuentran en un "***dataframe***" :

- El método ***head*** permite obtener los primeros registros de un *dataframe*.
- El objeto ***dtypes*** indica los tipos de las columnas del *dataframe*.
- El método ***info*** de un *dataframe* permite consultar información como el número de registros (observaciones) y de columnas (atributos) con los tipos de datos correspondientes, el número de registros presentes no nulos, y el tamaño que ocupa el *dataframe* en memoria.
- El método ***describe*** de un *dataframe* permite obtener un resumen de las columnas, con estadísticas descriptivas que permiten entender la distribución de cada variable (mean, std, min, 25%, 50%, 75%, max).

Normalización de atributos de entrada

Sea x un atributo de entrada.

Normalización con rango de salida [0,1]

$x_{max} = \max(x)$

$x_{min} = \min(x)$

Sii:

$(x_{max} - x_{min}) \neq 0$

Entonces:

$$x_{norm} = \left(\frac{x - x_{min}}{x_{max} - x_{min}} \right)$$

$x_{norm} \in [0,1]$

x = {2, 4, 8}

x_{norm} = {0, 1/3, 1}

Normalización con rango de salida [a,b]

$a = \min(x_{norm})$

$b = \max(x_{norm})$

$x_{max} = \max(x)$

$x_{min} = \min(x)$

Sii:

$(x_{max} - x_{min}) \neq 0$

Entonces:

$$x_{norm} = a + \left(\frac{x - x_{min}}{x_{max} - x_{min}} \right)(b - a)$$

$x_{norm} \in [a,b]$

Nota: *a* y *b* se escogen con antelación.

a = 1

b = 10

x = {2, 4, 8}

x_{norm} = {1, 4, 10}

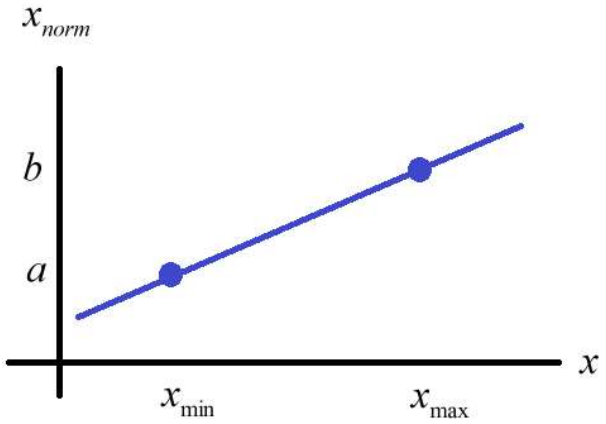


Figura: Representación gráfica del proceso de normalización. Note que el segundo caso de normalización se reduce al primero cuando a=0 y b=1.

Cómo usarlo en Python? https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler

Estandarización de atributos de entrada

Estandarización (*z-score*)

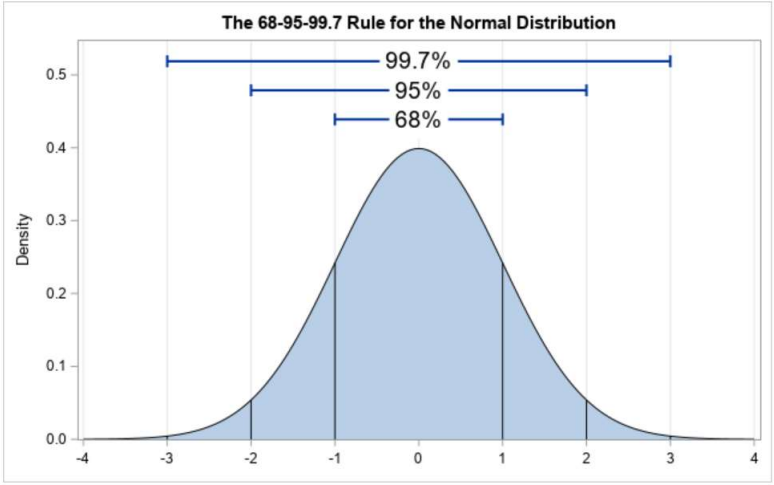
- Se parte de un supuesto de distribución normal.
- x es el valor actual del atributo.
- μ es la media aritmética del atributo x.
- σ es la desviación estándar del atributo x.
- z es la representación estandarizada.

$$z = \frac{x - \mu}{\sigma}$$

Para σ diferente de 0.

Nota: como en la práctica no se conoce ni la media ni la desviación estándar de la *población*, estos valores se estiman a partir de la *muestra* (i.e.: usando las observaciones disponibles del atributo *x*).

Cómo usarlo en Python? https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler

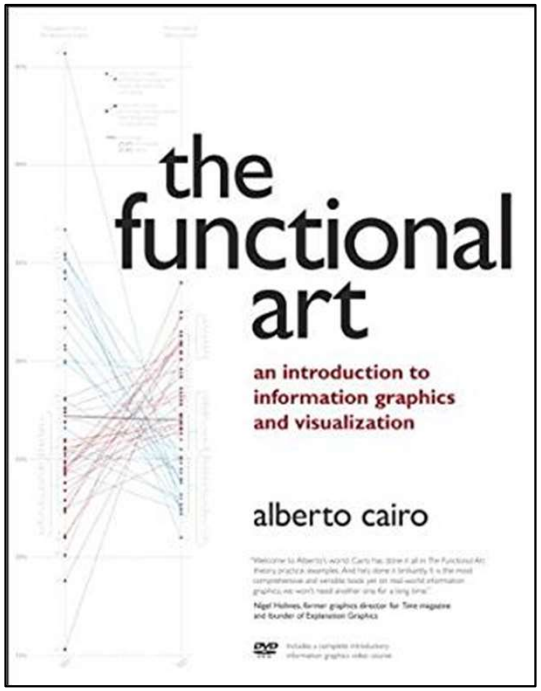


Pregunta 1: Si los datos se distribuyen de forma normal, podríamos detectar los datos anómalos?
Si, podríamos identificar como datos anómalos aquellos que se encuentren a más de ~3 desviaciones estándar de la media, pues P(Z>3) = 0.13499 %, luego P(|Z|>3)=0.26998%.
https://www.wolframalpha.com/input/?i=P%5Bz>3%5D+for+z~normal+distribution+with+mean+0+and+standard+deviation+1

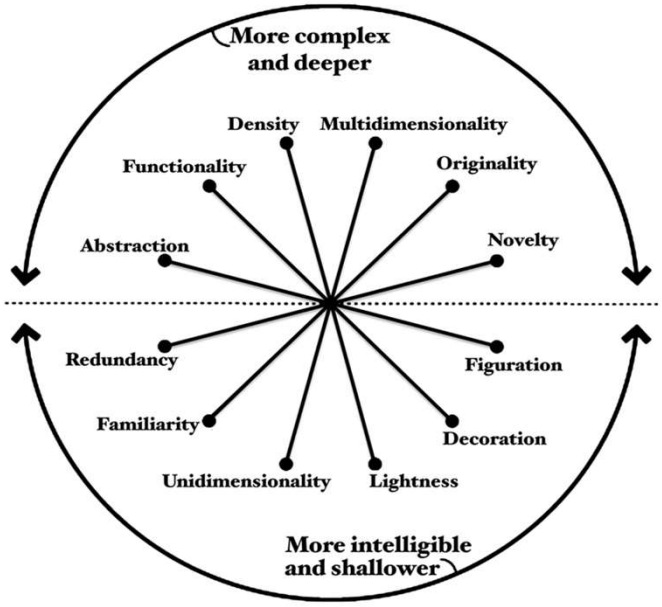
Pregunta 2: Pero cómo sabemos que los datos se distribuyen de forma normal?
Revisando tests estadísticos de normalidad (Shapiro-Wilk, D'Agostino's K-squared, Anderson-Darling, Chi-Square Normality, Lilliefors, Jarque–Bera, Kolmogorov-Smirnov) y ciertas gráficas (histogramas, densidad de kernel, gráficos de dispersión, gráficos de cajas, gráficos P-P, gráficos Q-Q).
Si usa Python, este blog le va a interesar:
https://towardsdatascience.com/normality-tests-in-python-31e04aa4f411

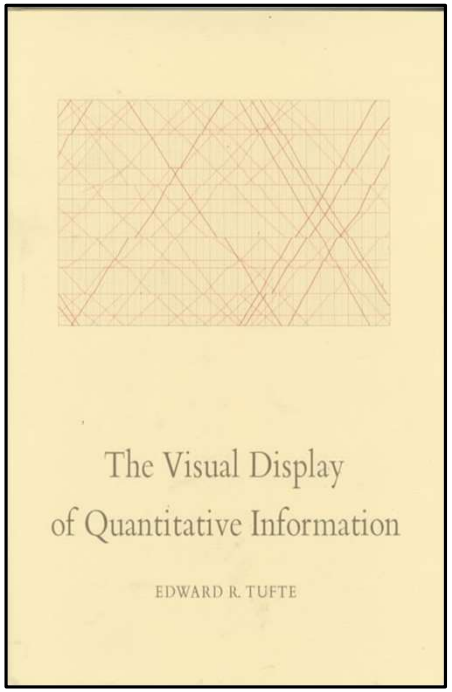
Visualización de datos

La rueda de la visualización



2013





1985

Dimensiones de la rueda de la visualización [1/2]

- Abstracción vs. Figuración:
  ✓Cuadros y gráficos (abstracción) u objetos físicos del mundo real (figuración).
- Funcionalidad vs. Decoración:
  ✓Sin adornos (funcionalidad) o adornos artísticos (decoración).
- Densidad vs. Ligereza:
  ✓ Debe estudiarse en profundidad (densidad) o ser comprensible de un vistazo (ligereza).

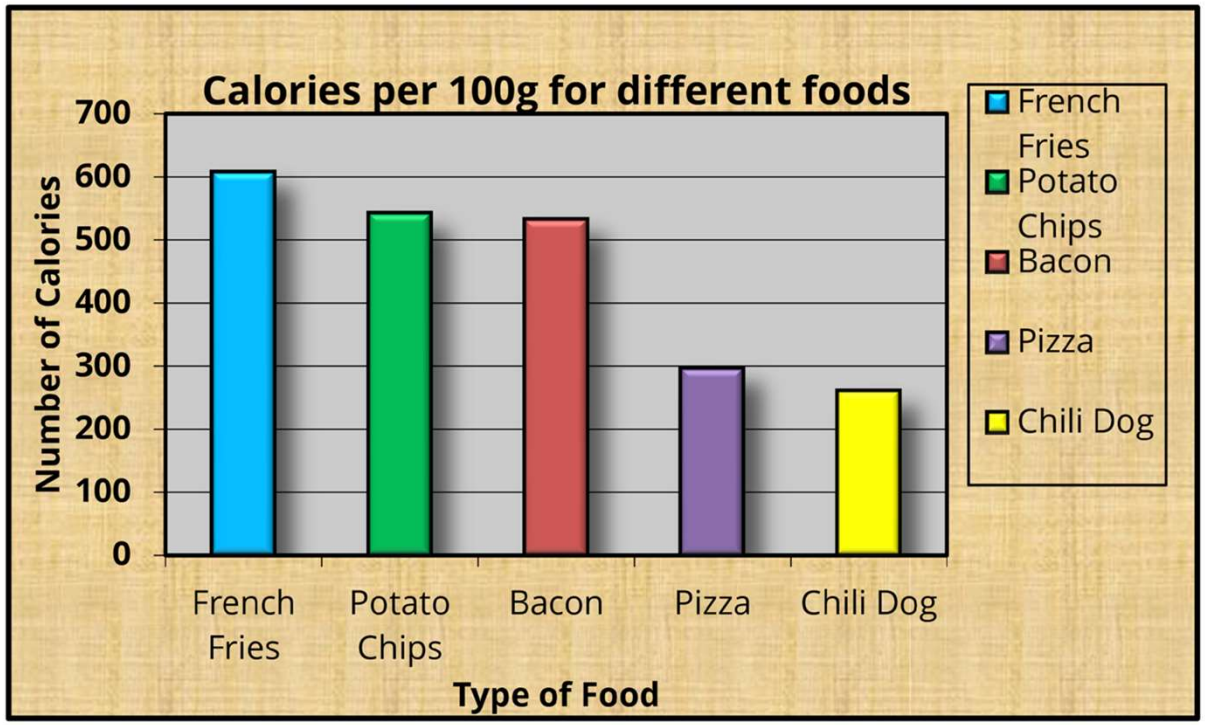
Dimensiones de la rueda de la visualización [2/2]

- Multidimensional vs. Unidimensional:
  - ✓ Diferentes aspectos de los fenómenos (multidimensionales) o elementos únicos o que representen aspecto del fenómeno (unidimensionales).
- Originalidad – Familiaridad:
  - ✓ Nuevos métodos de visualización (originalidad) o métodos de visualización establecidos y bien entendidos (familiaridad).
- Novedad vs. redundancia:
  - ✓ Explicar cada ítem una vez (novedad) o codificar múltiples explicaciones de los mismos fenómenos (redundancia).

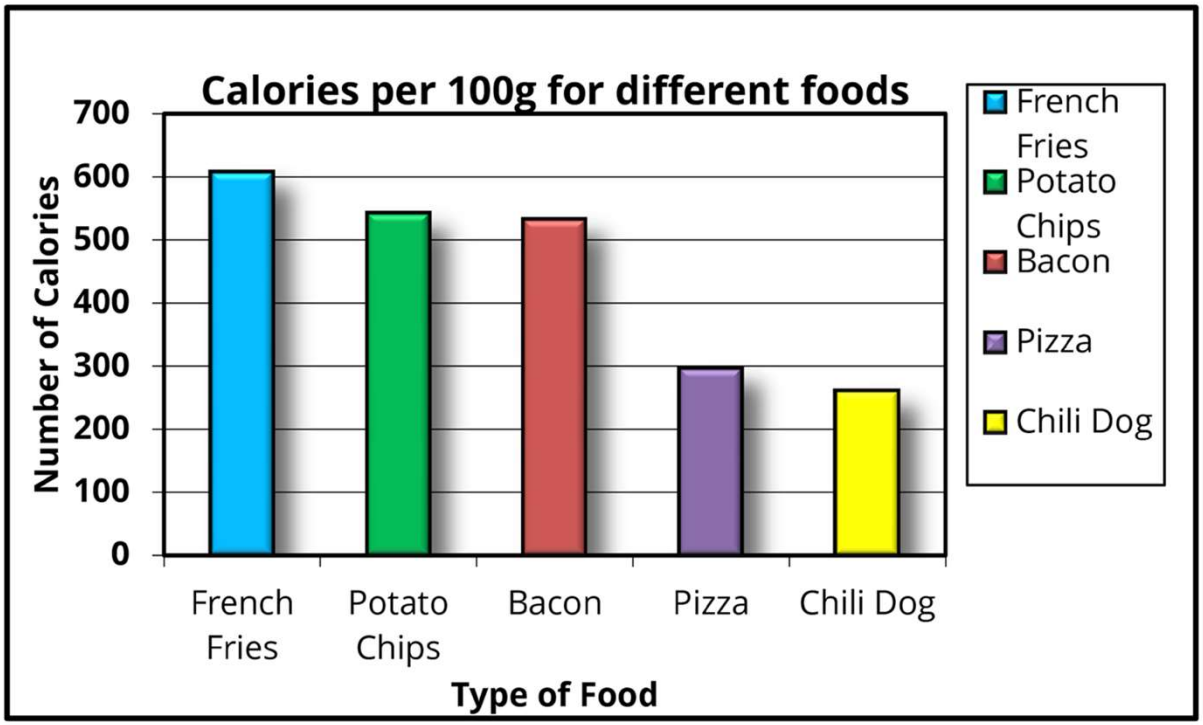
En ocasiones,
eliminar permite mejorar las visualizaciones…

Fuente: Applied Plotting, Charting & Data Representation in Python. Christopher Brooks.

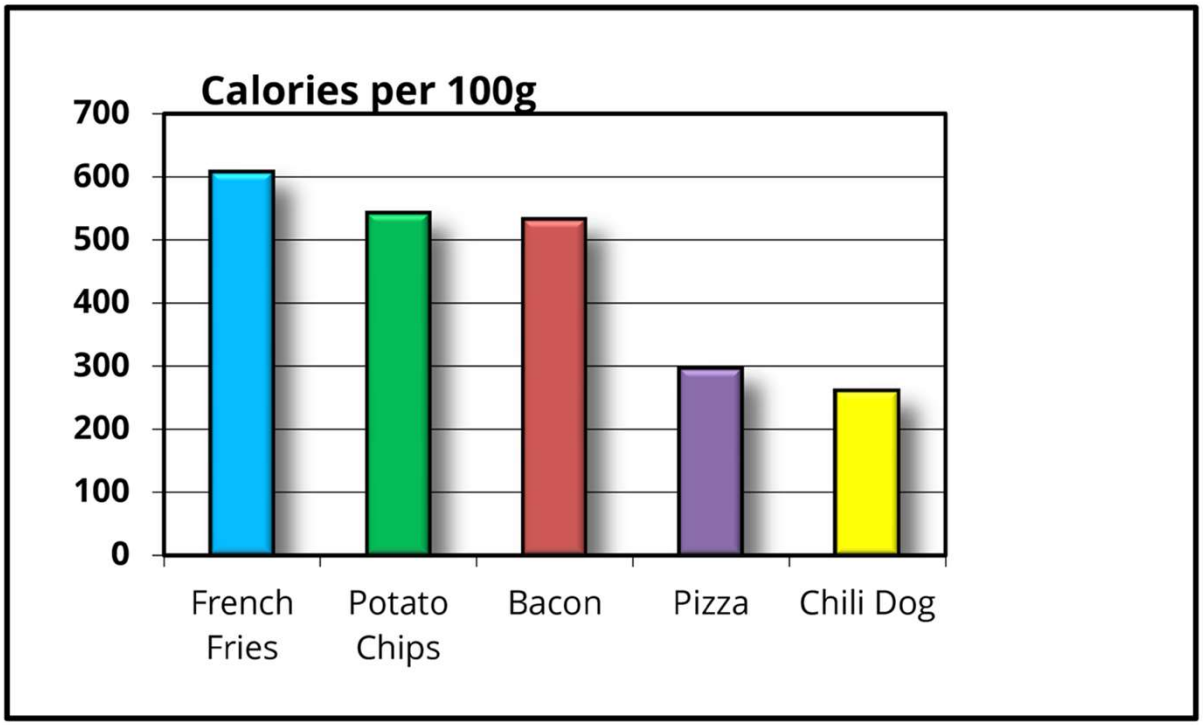
Remove backgrounds



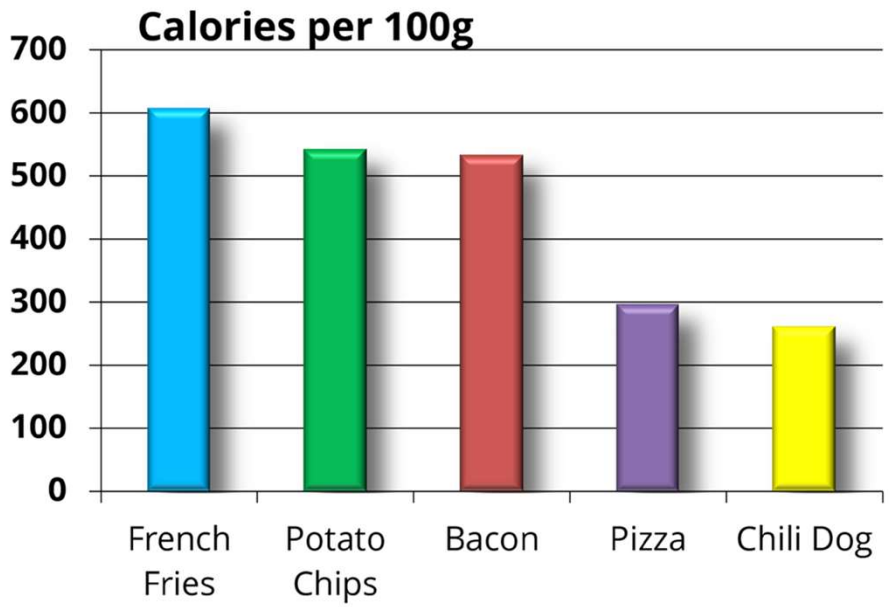
Remove redundant labels



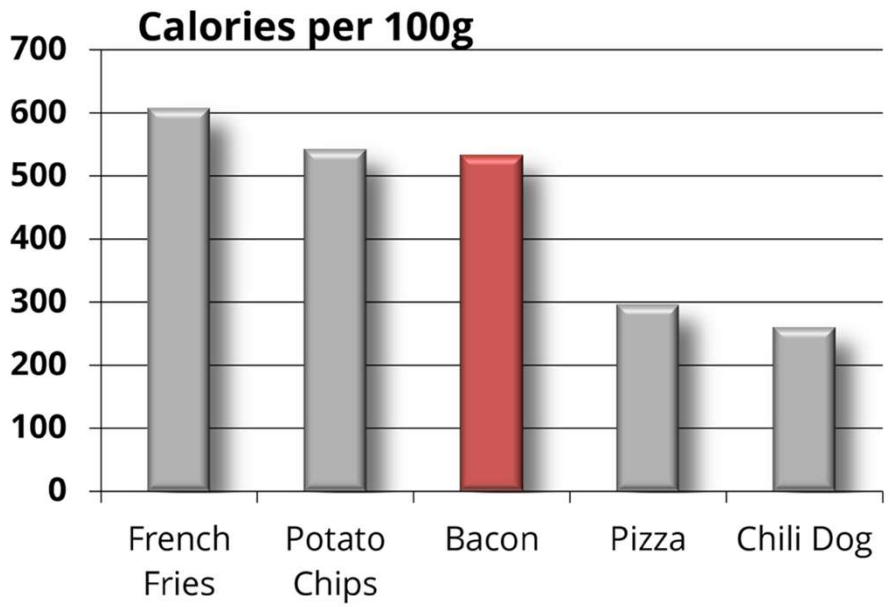
Remove borders



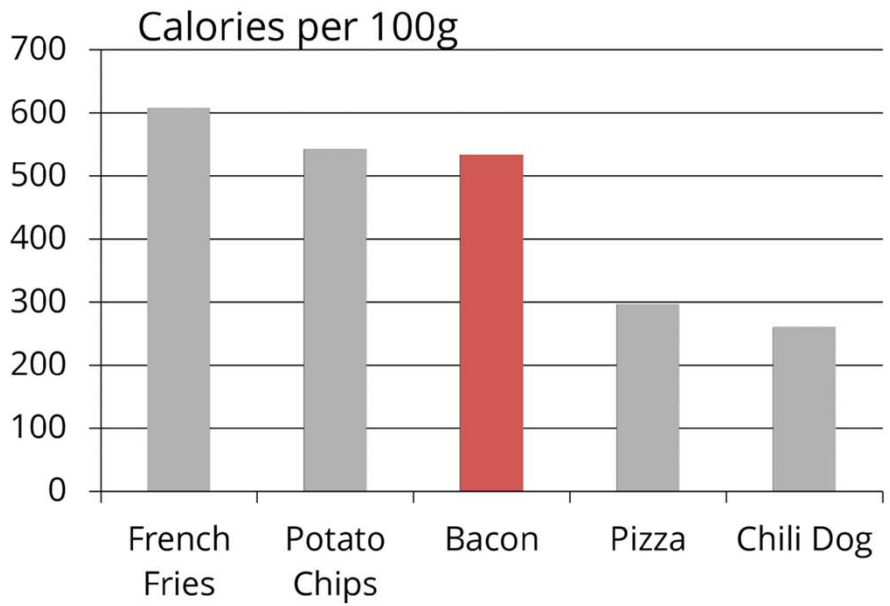
Reduce colors



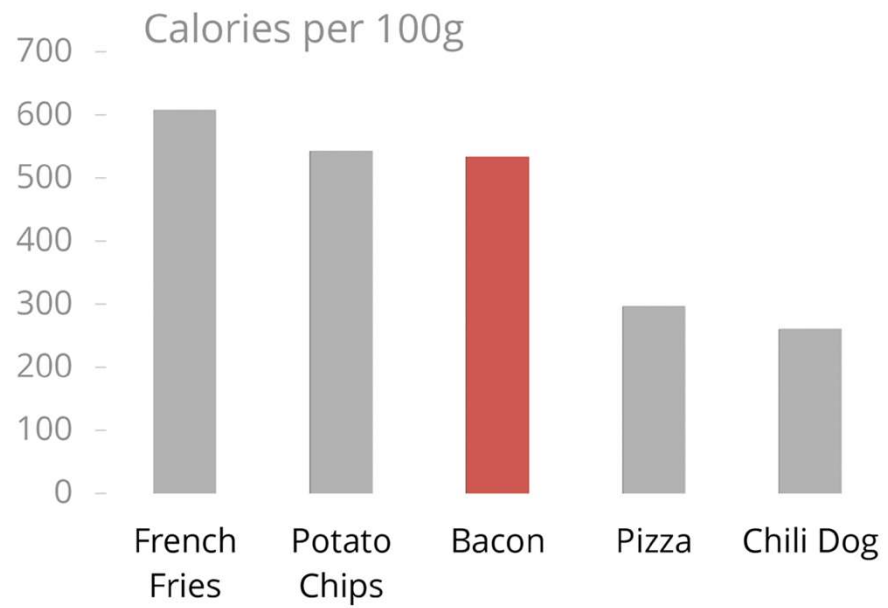
Remove special effects



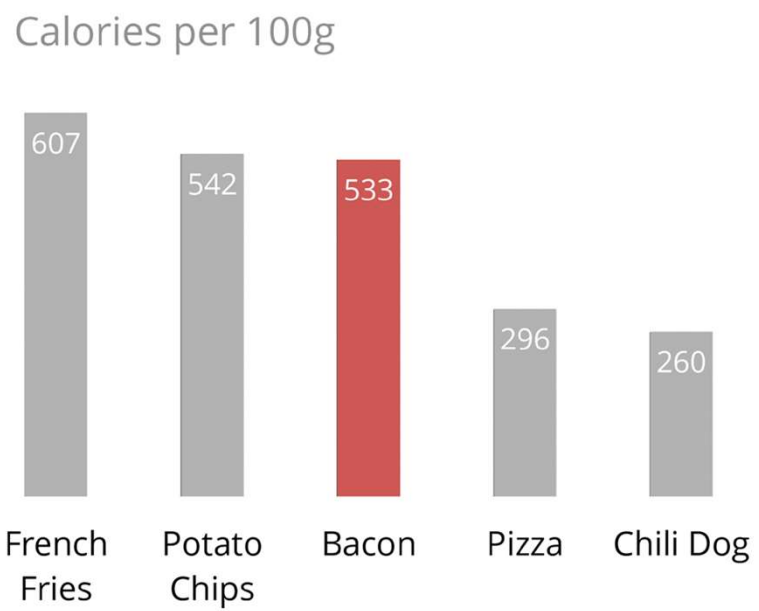
Lighten labels



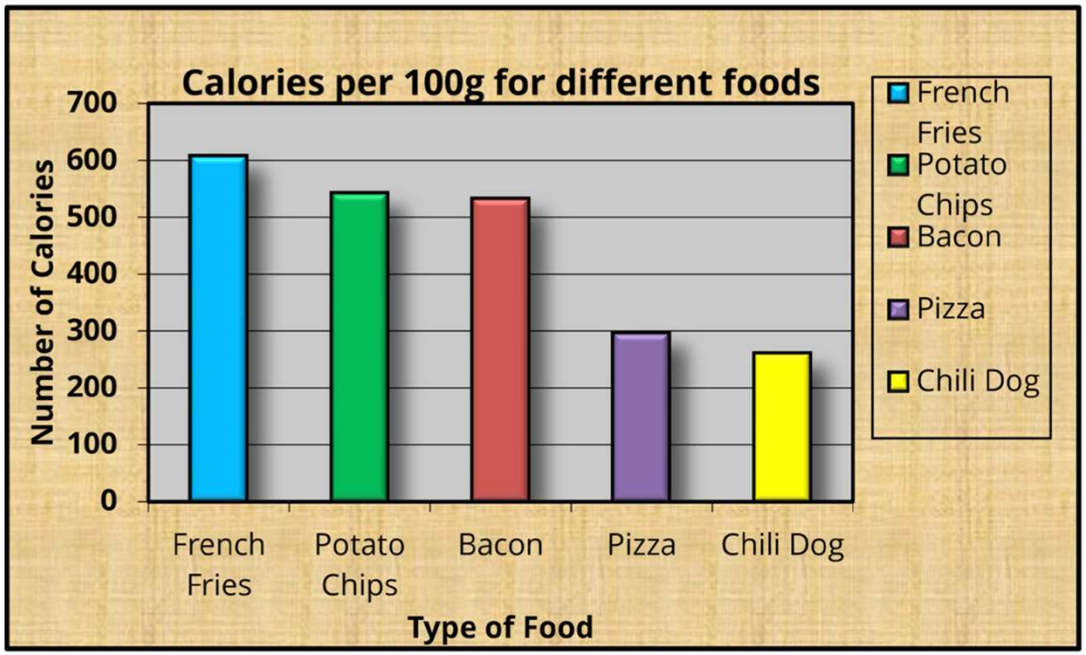
Direct label



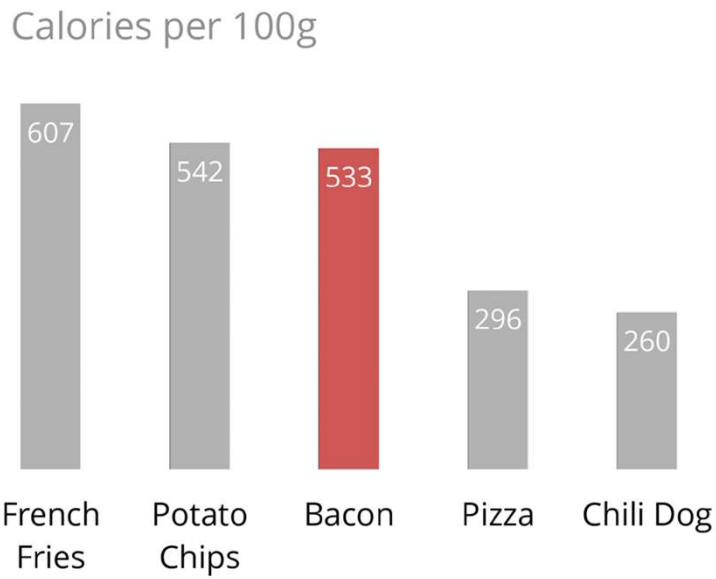
Direct label



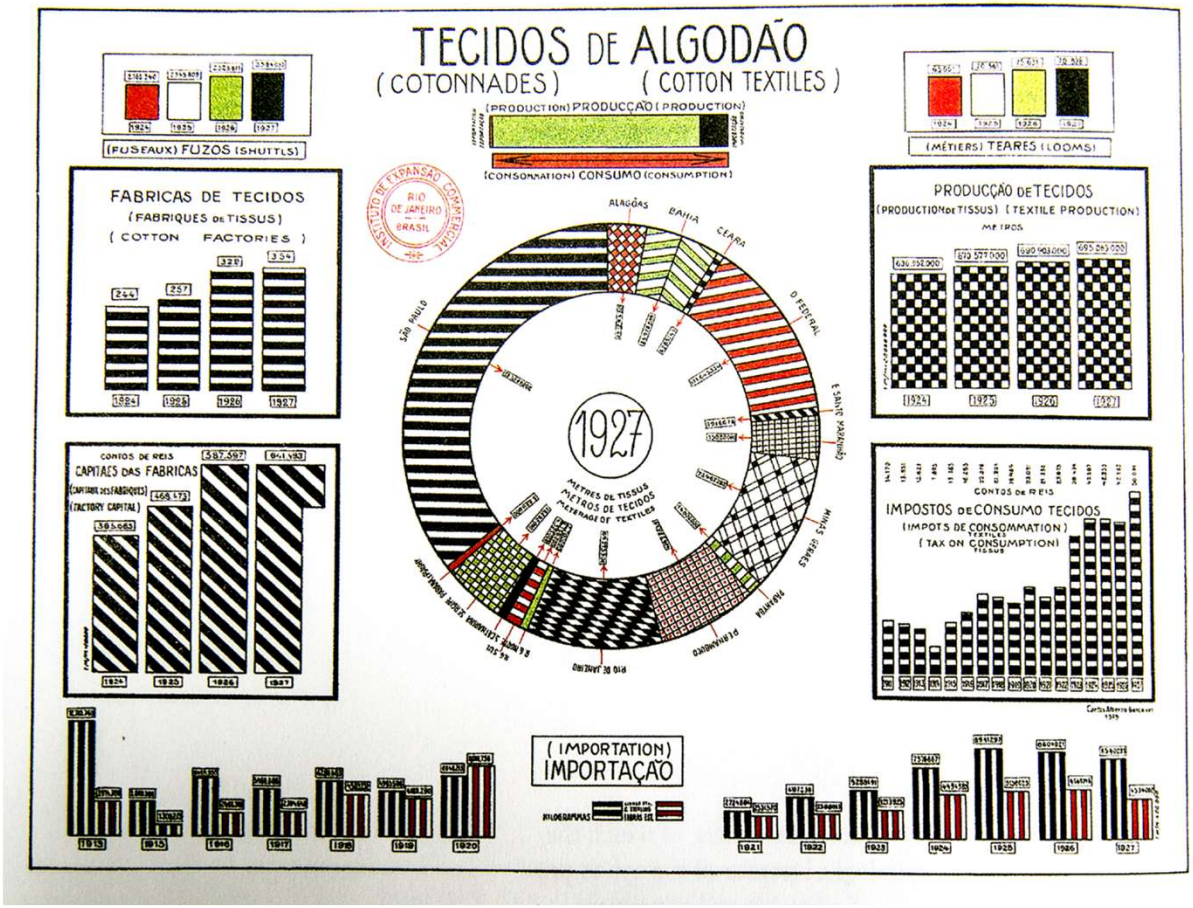
ANTES



DESPUÉS



¿Qué tal les parece esta representación de la información? ¿cómo podrían mejorarla?



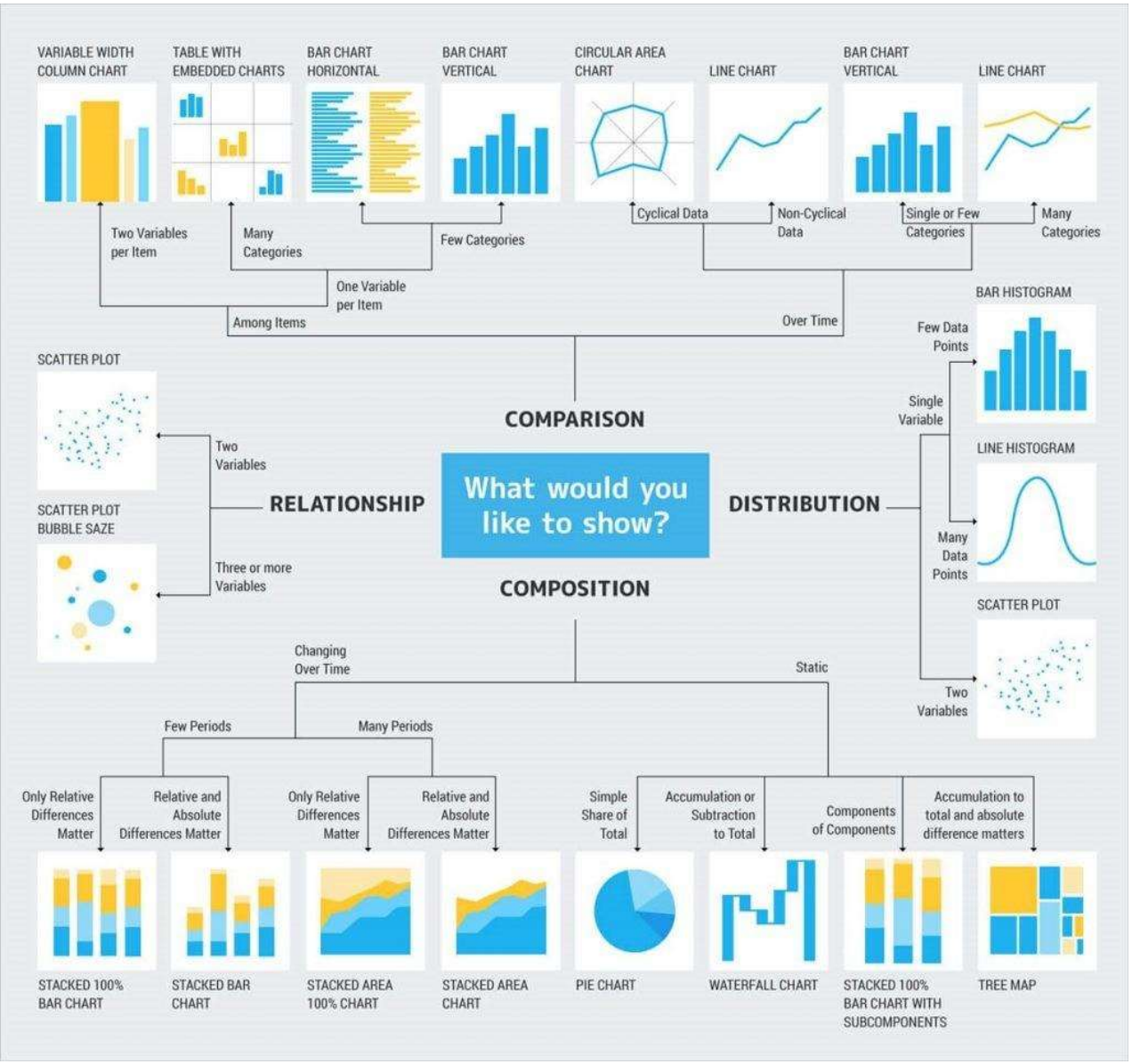
Fuente: Tufte, E. R. (1985). "Instituto de Expansao Commercial, Brasil: 'Graphicos Economicos-Estisticas' (Rio de Janeiro, 1929), p 15." *The Visual Display of Quantitative Information*. Second Edition. Cheshire, CT: Graphics Press. (pp 108).

Características de una buena visualización

1. Veraz*.
2. Funcional.
3. Estética.
4. Informativa y con responsabilidad ética.

*Usted es *responsable de las acciones que toma durante la limpieza, agrupación, y procesamiento de los datos que va a usar para evitar representaciones engañosas de los datos.*

Graficando desde Python



Los gráficos son herramientas poderosas que permiten **identificar y comunicar** conceptos relevantes de los datos.

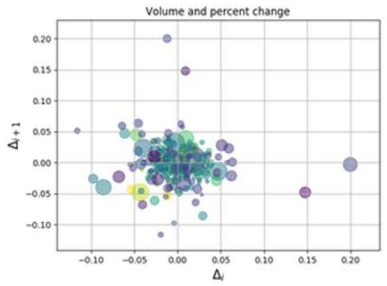
En Python podemos utilizar visualizaciones que nos permiten entender mejor los datos.

- Gráfico de barras.
- Gráfico de líneas.
- Gráfico de densidades.
- Gráfico de dispersión (*scatter plot*).
- Gráfico de caja (*boxplot*).

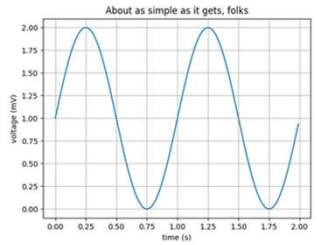
…. Entre otros.

Matplotlib

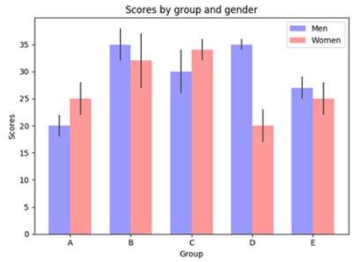
Tipos de visualizaciones



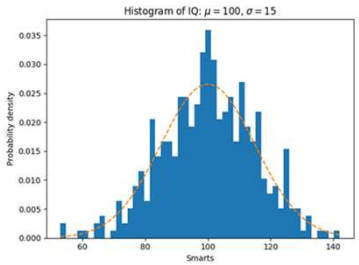
Scatter Plot



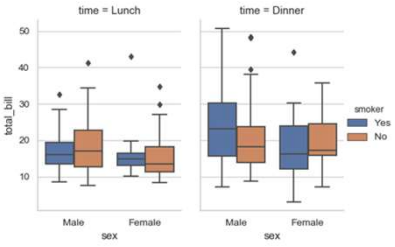
Linear Plot



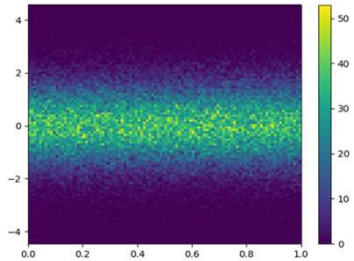
Bar Chart



Histograms



Box Plot

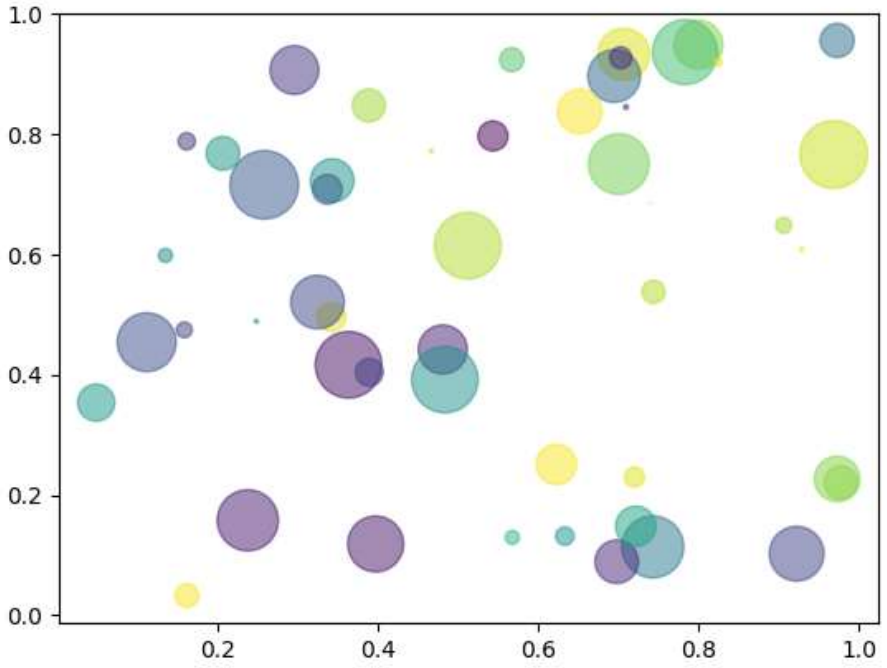


Heat Map

Matplotlib

Scatter Plot

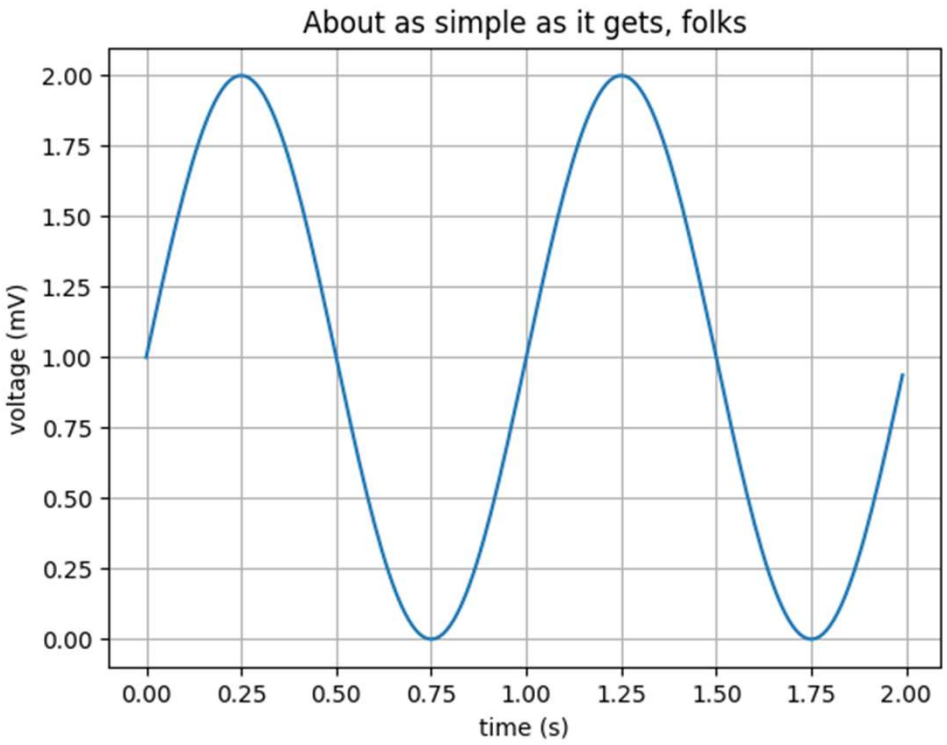
```
import numpy as np
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)


N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2  # 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```



Matplotlib

Linear Plot

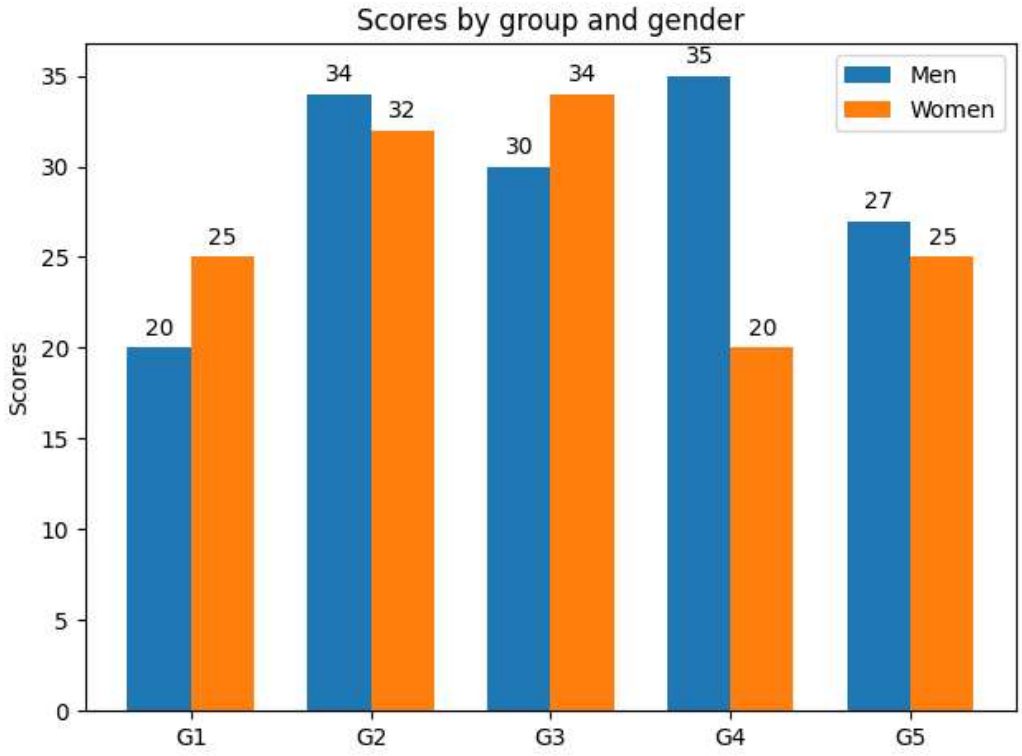
```
fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       title='About as simple as it gets, folks')
ax.grid()

fig.savefig("test.png")
plt.show()
```



Matplotlib

Bar Chart

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np


labels = ['G1', 'G2', 'G3', 'G4', 'G5']
men_means = [20, 34, 30, 35, 27]
women_means = [25, 32, 34, 20, 25]

x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, men_means, width, label='Men')
rects2 = ax.bar(x + width/2, women_means, width, label='Women')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()


def autolabel(rects):
    """Attach a text label above each bar in *rects*, displaying its height."""
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')


autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

plt.show()
```

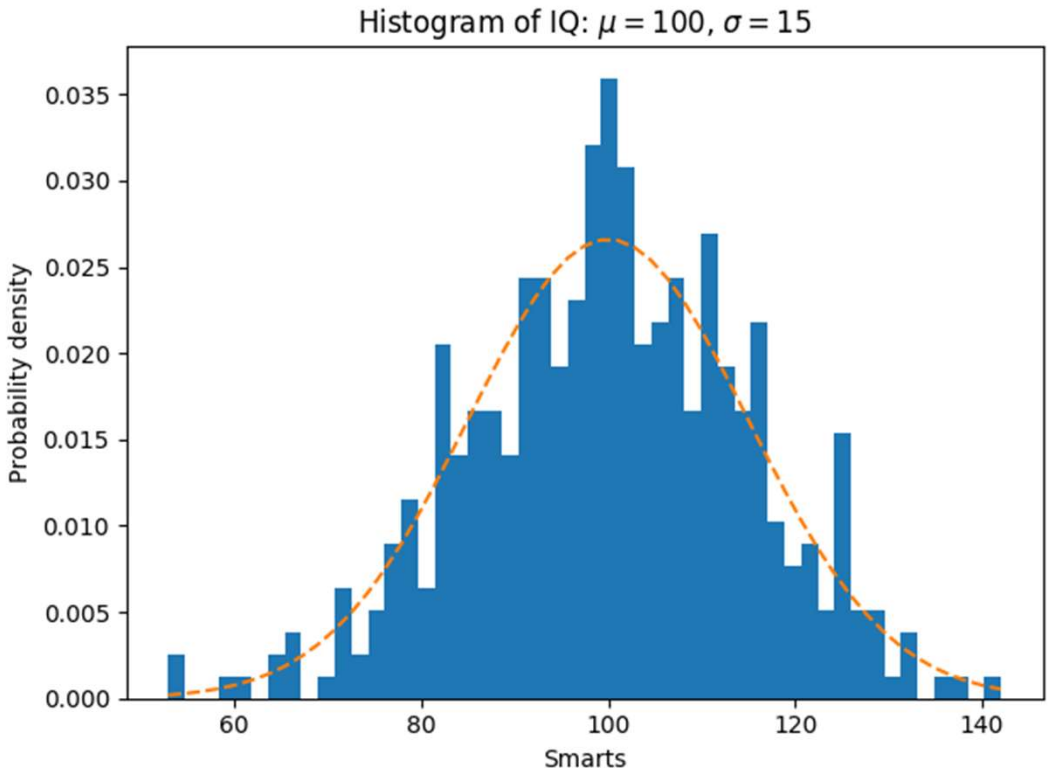


Fuente:
https://matplotlib.org/stable/gallery/lines_bars_and_markers/barchart.html

Matplotlib

Histograms

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(19680801)

# example data
mu = 100  # mean of distribution
sigma = 15  # standard deviation of distribution
x = mu + sigma * np.random.randn(437)

num_bins = 50

fig, ax = plt.subplots()

# the histogram of the data
n, bins, patches = ax.hist(x, num_bins, density=True)

# add a 'best fit' line
y = ((1 / (np.sqrt(2 * np.pi) * sigma)) *
     np.exp(-0.5 * (1 / sigma * (bins - mu))**2))
ax.plot(bins, y, '--')
ax.set_xlabel('Smarts')
ax.set_ylabel('Probability density')
ax.set_title(r'Histogram of IQ: $\mu=100$, $\sigma=15$')

# Tweak spacing to prevent clipping of ylabel
fig.tight_layout()
plt.show()
```

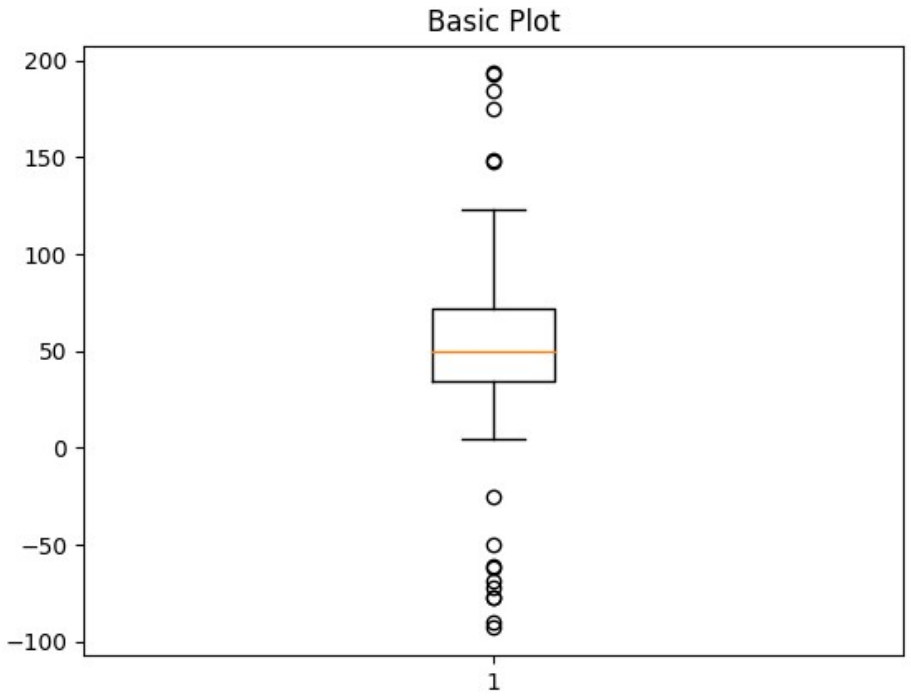


Fuente: https://matplotlib.org/3.3.4/gallery/statistics/histogram_features.html

Matplotlib

Box Plot

```
import numpy as np
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)

# fake up some data
spread = np.random.rand(50) * 100
center = np.ones(25) * 50
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
data = np.concatenate((spread, center, flier_high, flier_low))

fig1, ax1 = plt.subplots()
ax1.set_title('Basic Plot')
ax1.boxplot(data)
```

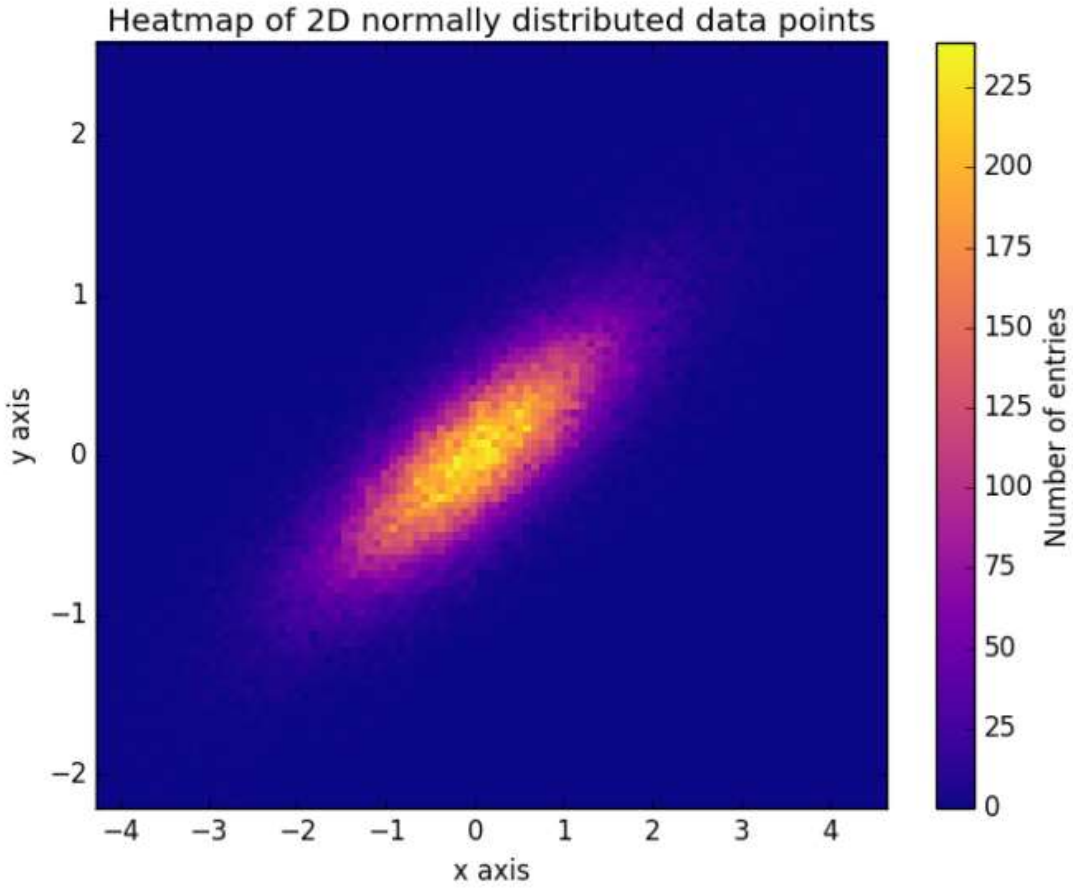


Fuente:
https://matplotlib.org/stable/gallery/pyplots/boxplot_demo_pyplot.html

Matplotlib

Heat map

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

# Define numbers of generated data points and bins per axis.
N_numbers = 100000
N_bins = 100

# set random seed
np.random.seed(0)

# Generate 2D normally distributed numbers.
x, y = np.random.multivariate_normal(
        mean=[0.0, 0.0],      # mean
        cov=[[1.0, 0.4],
             [0.4, 0.25]],    # covariance matrix
        size=N_numbers
        ).T                   # transpose to get columns


# Construct 2D histogram from data using the 'plasma' colormap
plt.hist2d(x, y, bins=N_bins, normed=False, cmap='plasma')

# Plot a colorbar with label.
cb = plt.colorbar()
cb.set_label('Number of entries')

# Add title and labels to plot.
plt.title('Heatmap of 2D normally distributed data points')
plt.xlabel('x axis')
plt.ylabel('y axis')

# Show the plot.
plt.show()
```

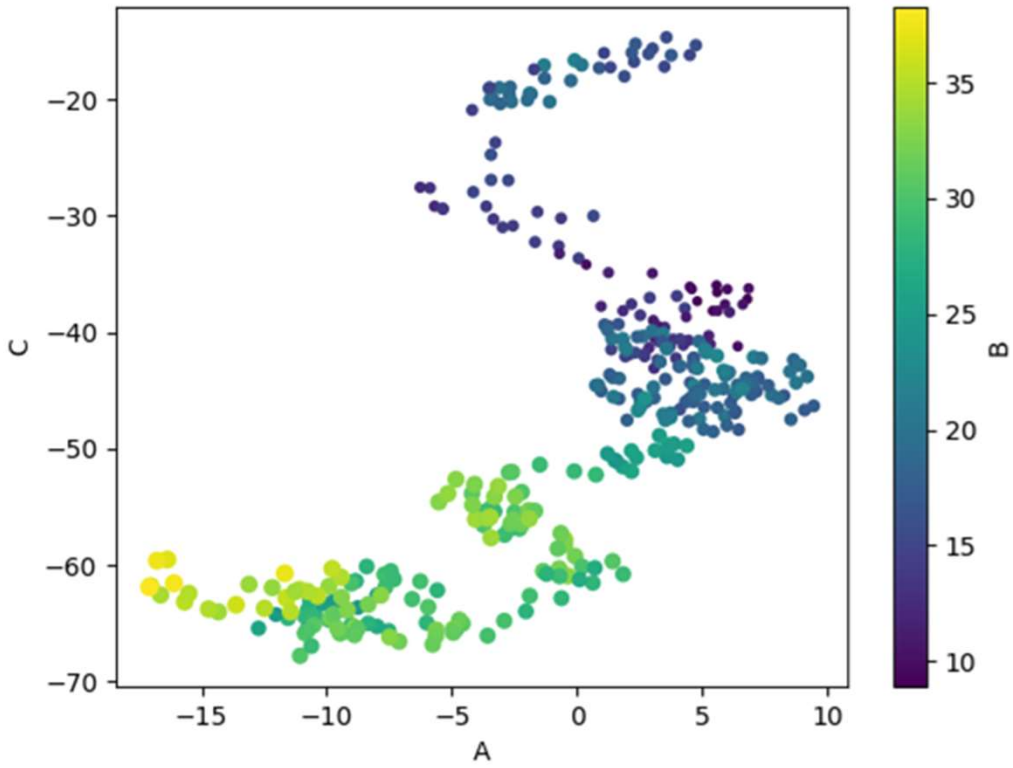


Pandas

Scatterplot by color

```
df.plot.scatter('A', 'C', c='B', s=df['B'], colormap='viridis')
```

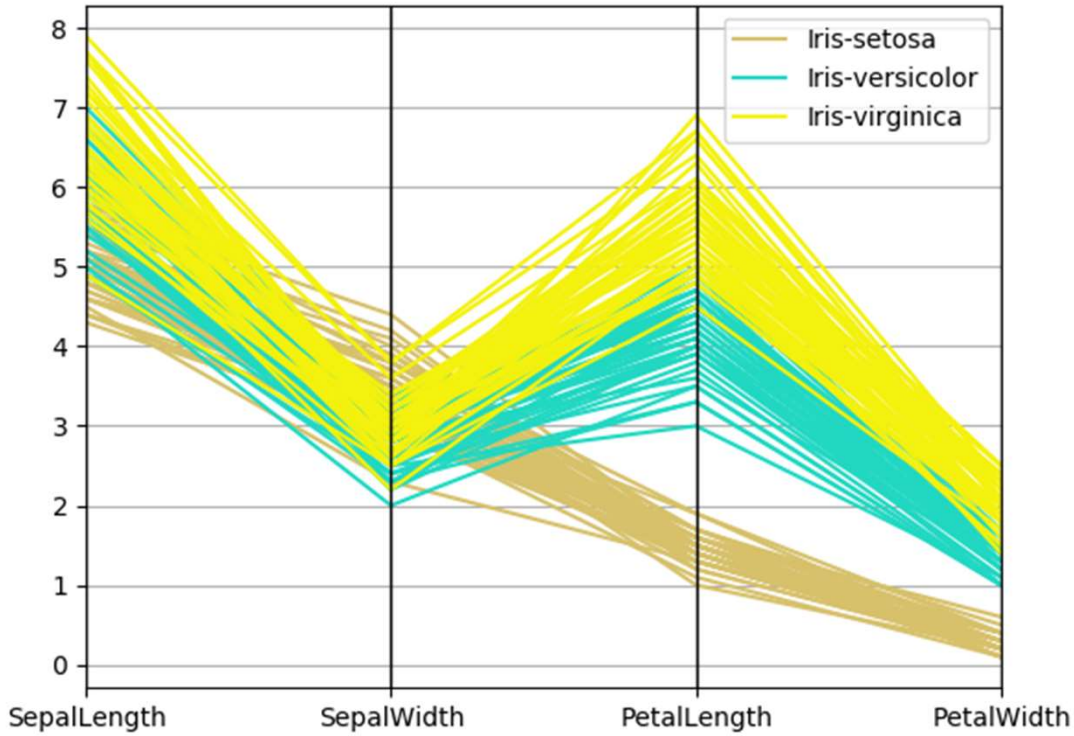
c: color.
s: size.



Pandas

Scatterplot by color

```
plt.figure()
pd.tools.plotting.parallel_coordinates(iris, 'Name');
```

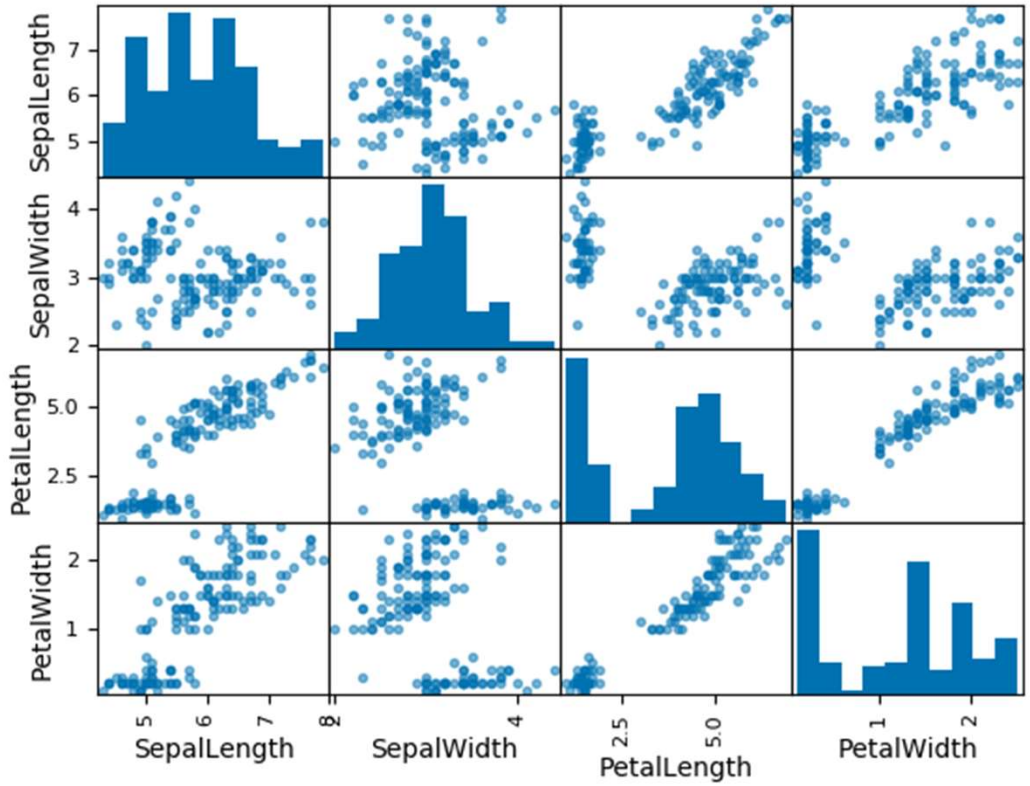


Conjunto de datos "*Iris Species*": https://www.kaggle.com/uciml/iris

Pandas

Scatterplot by color

```
pd.tools.plotting.scatter_matrix(iris);
```

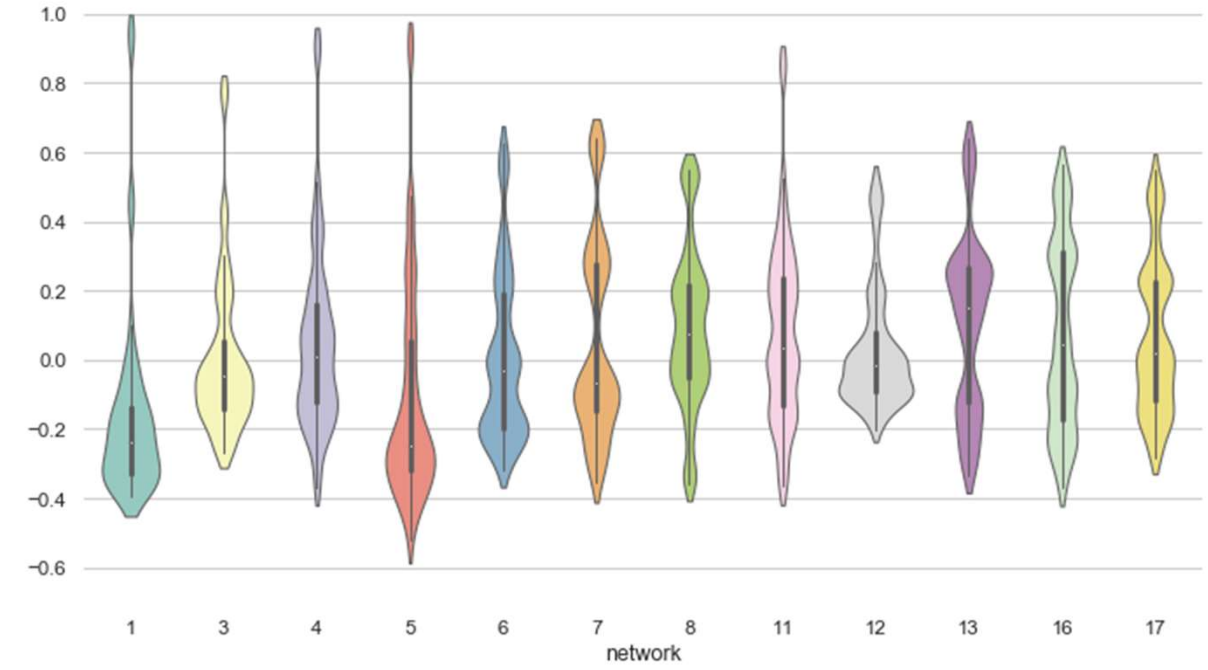


Conjunto de datos "*Iris Species*": https://www.kaggle.com/uciml/iris

Seaborn

Violin Plot

```
# Compute the correlation matrix and average over networks
corr_df = df.corr().groupby(level="network").mean()
corr_df.index = corr_df.index.astype(int)
corr_df = corr_df.sort_index().T

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 6))

# Draw a violinplot with a narrower bandwidth than the default
sns.violinplot(data=corr_df, palette="Set3", bw=.2, cut=1, linewidth=1)

# Finalize the figure
ax.set(ylim=(-.7, 1.05))
sns.despine(left=True, bottom=True)
```

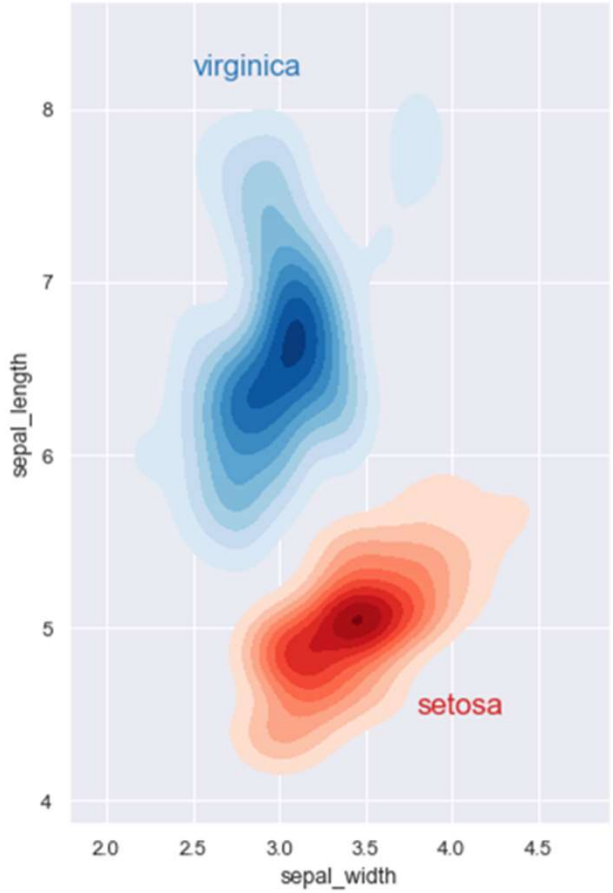


Nota: Los diagramas de violín son similares a los diagramas de caja (*box plots*), excepto que también muestran la densidad de probabilidad de los datos en diferentes valores, generalmente suavizados por un estimador de densidad con un kernel predefinido.

Seaborn

Kde Plot

```
# Draw the two density plots
ax = sns.kdeplot(setosa.sepal_width, setosa.sepal_length,
                 cmap="Reds", shade=True, shade_lowest=False)
ax = sns.kdeplot(virginica.sepal_width, virginica.sepal_length,
                 cmap="Blues", shade=True, shade_lowest=False)

# Add labels to the plot
red = sns.color_palette("Reds")[-2]
blue = sns.color_palette("Blues")[-2]
ax.text(2.5, 8.2, "virginica", size=16, color=blue)
ax.text(3.8, 4.5, "setosa", size=16, color=red)
```

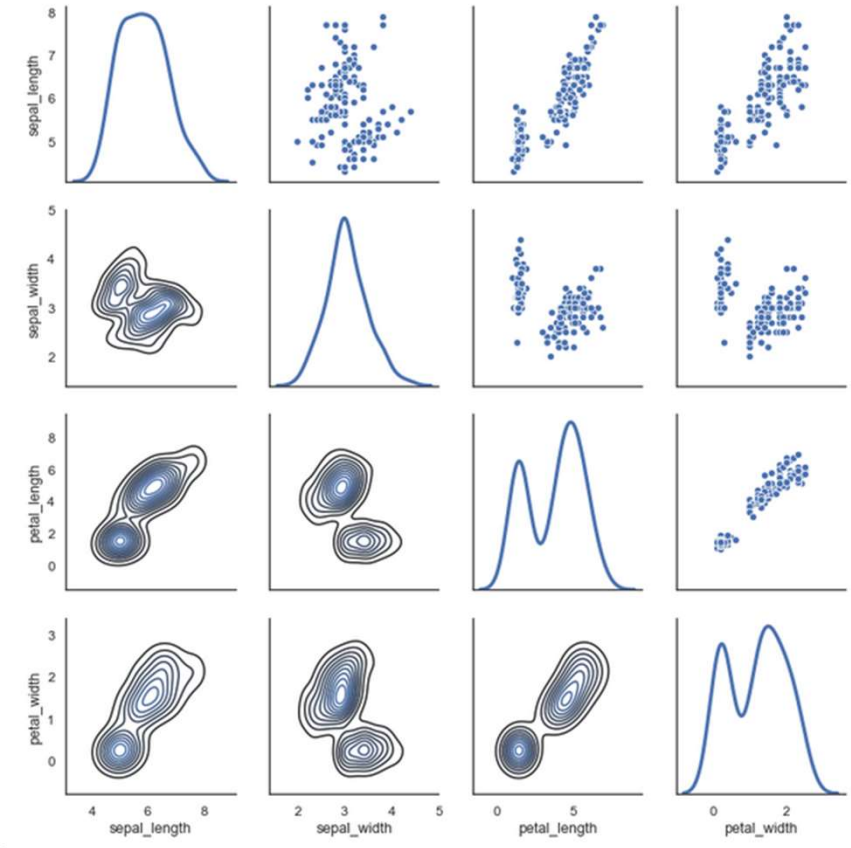


Conjunto de datos "*Iris Species*": https://www.kaggle.com/uciml/iris

Seaborn

PairGrid & Scatterplot matrix

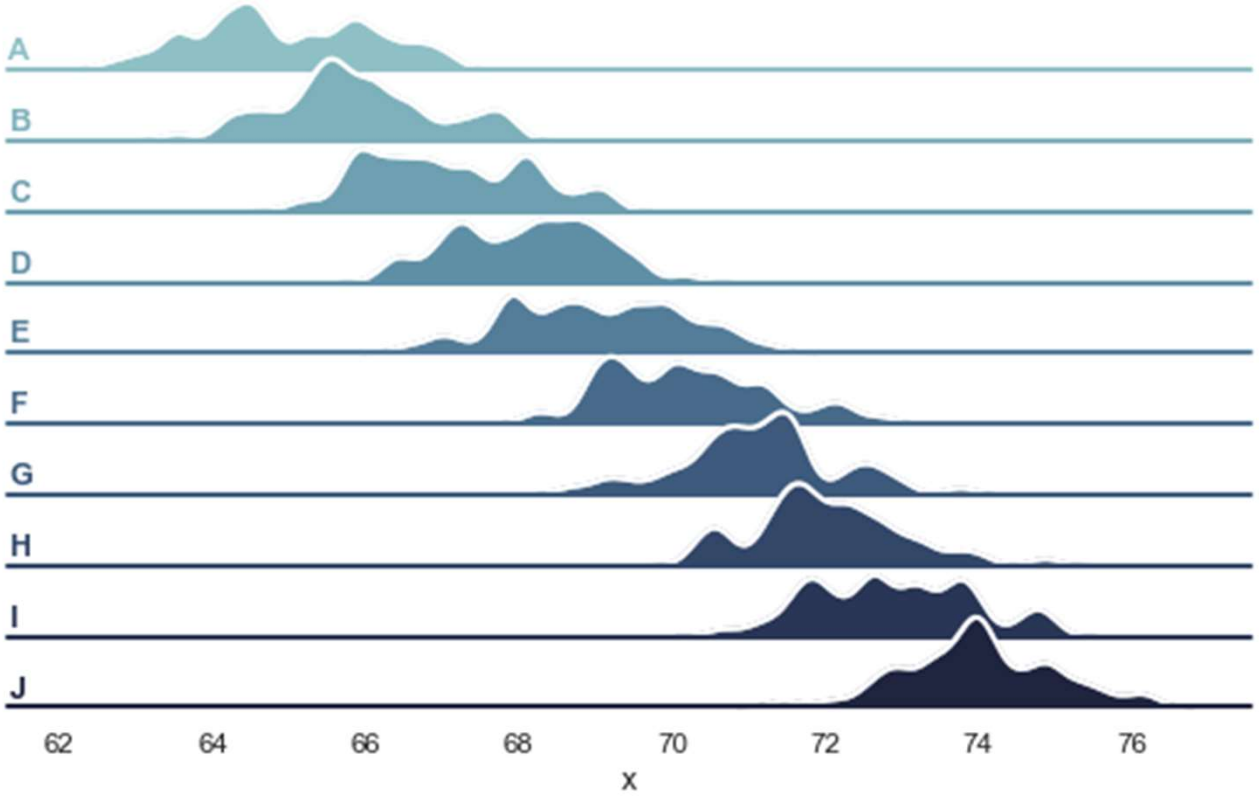
```
sns.set(style="white")

df = sns.load_dataset("iris")

g = sns.PairGrid(df, diag_sharey=False)
g.map_lower(sns.kdeplot)
g.map_upper(sns.scatterplot)
g.map_diag(sns.kdeplot, lw=3)
```



Conjunto de datos "*Iris Species*": https://www.kaggle.com/uciml/iris

Seaborn

Ridge
Plot

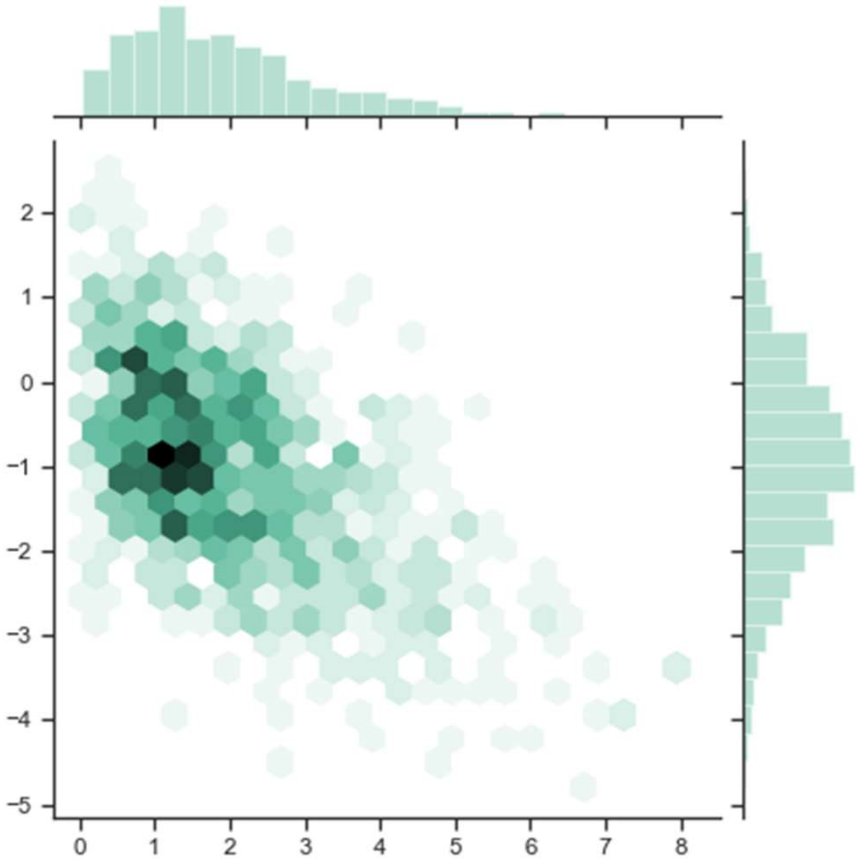


Fuente: https://seaborn.pydata.org/examples/kde_ridgeplot.html

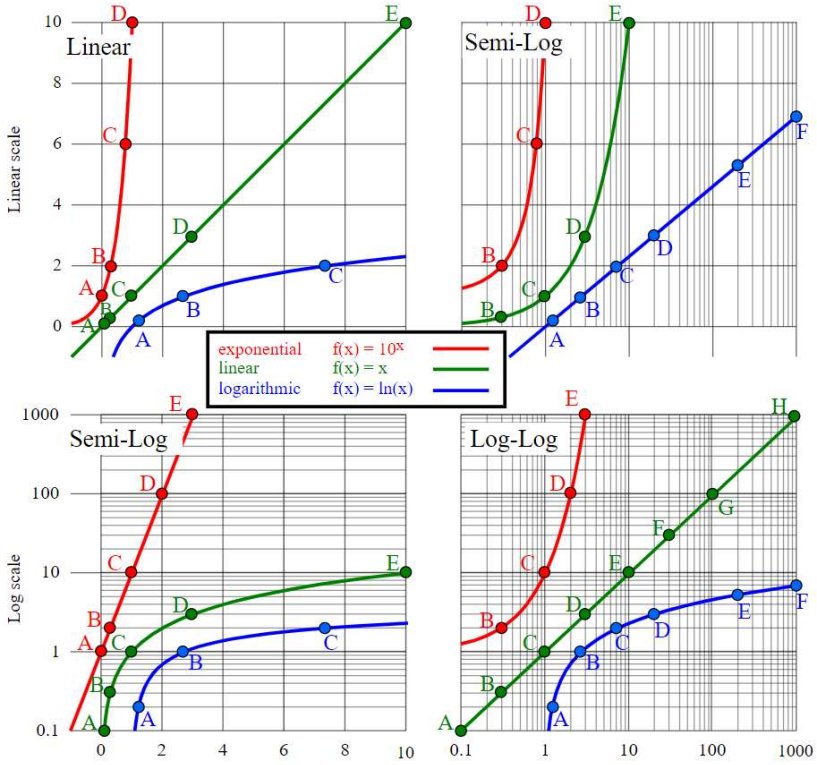
Seaborn

Hexbin Plot with Marginal Distributio

```
import numpy as np
import seaborn as sns
sns.set(style="ticks")

rs = np.random.RandomState(11)
x = rs.gamma(2, size=1000)
y = -.5 * x + rs.normal(size=1000)

sns.jointplot(x, y, kind="hex", color="#4CB391")
```



Comentario sobre escalas lineales, semi-log, y log-log



En Matplotlib puede usar:
plt.plot
plt.semilogx
plt.semilogy
plt.loglog

Fuente: https://en.wikipedia.org/wiki/Logarithmic_scale

Actividad

Revisar estos recursos en línea:

The Infographics Complexity Challenge: Presentation and Exploration.
https://www.peachpit.com/articles/article.aspx?p=1945331&seqNum=3

10 Dashboard Design Errors [and how to avoid them]:
https://www.fusioncharts.com/blog/10-dashboard-design-mistakes/

6 examples of bad dashboard designs:
https://carmel.es/2018/10/26/6-examples-of-bad-dashboard-designs/

Top 50 matplotlib Visualizations – The Master Plots
https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/