

```
In [1]: import numpy as np  
import pandas as pd  
import matplotlib as plt  
import seaborn as sns
```

```
In [2]: # !pip install scipy  
# !pip install statsmodels  
# !pip install pingouin
```

```
In [2]: from scipy import stats  
import statsmodels as sm  
import pingouin as pg
```

Estadística para Ciencia de Datos

1. Estadística Descriptiva

Conceptos básicos

Introducción

La estadística clásica se enfocó casi exclusivamente en la inferencia, un conjunto a veces complejo de procedimientos para sacar conclusiones sobre grandes poblaciones basadas en muestras pequeñas. En 1962, John W. Tukey llamó a una reforma de la estadística en su artículo seminal "The Future of Data Analysis" [Tukey-1962]. Propuso una nueva disciplina científica llamada análisis de datos que incluía la inferencia estadística como un componente más. Tukey presentó gráficos simples (por ejemplo, diagramas de caja, diagramas de dispersión) que, junto con estadísticas resumen (media, mediana, cuantiles, etc.), ayudan a pintar un cuadro de un conjunto de datos.

Tipos Básicos de Datos Estructurados

Hay dos tipos básicos de datos estructurados: **numéricos** y **categóricos**.

Los datos numéricos vienen en dos formas: **continuos**, como la velocidad del viento o la duración del tiempo, y **discretos**, como el conteo de la ocurrencia de un evento.

Los datos categóricos solo toman un conjunto fijo de valores, como el tipo de pantalla de TV (plasma, LCD, LED, etc.) o el nombre de un estado (Alabama, Alaska, etc.).

Los datos binarios son un caso especial de datos categóricos que toman solo uno de dos valores, como 0/1, sí/no, o verdadero/falso.

Otro tipo útil de datos categóricos son los datos ordinales en los que las categorías están ordenadas; un ejemplo de esto es una calificación numérica (1, 2, 3, 4 o 5), los grados en la ESO pueden clasificarse como ordinales.

Términos Clave para Tipos de Datos

- **Numeric (Numérico)**: Datos que se expresan en una escala numérica.
- **Continuous (Continuos)**: Datos que pueden tomar cualquier valor en un intervalo.
(Sinónimos: intervalo, flotante, numérico)
- **Discrete (Discretos)**: Datos que solo pueden tomar valores enteros, como conteos.
(Sinónimos: entero, conteo)
- **Categorical (Categóricos)**: Datos que solo pueden tomar un conjunto específico de valores que representan un conjunto de posibles categorías. (Sinónimos: enums, enumerados, factores, nominales)
- **Binary (Binarios)**: Un caso especial de datos categóricos con solo dos categorías de valores, por ejemplo, 0/1, verdadero/falso. (Sinónimos: dicotómicos, lógicos, indicadores, booleanos)
- **Ordinal (Ordinales)**: Datos categóricos que tienen un orden explícito. (Sinónimo: factor ordenado)

Rectangular Data

Los datos rectangulares son el término general para un array bidimensional con filas que indican registros (casos) y columnas que indican características (features), (variables); el data frame es el formato específico en R y Python.

Los datos no siempre comienzan en esta forma: los datos no estructurados (por ejemplo, texto) deben ser procesados y manipulados para que puedan ser representados como un conjunto de características en los datos rectangulares.

Los datos en bases de datos relacionales deben ser extraídos y puestos en una sola tabla para la mayoría de las tareas de análisis y modelado de datos.

Términos Clave para Datos Rectangulares

- **Data frame**: Los datos rectangulares (como una hoja de cálculo) son la estructura de datos básica para modelos estadísticos y de aprendizaje automático.
- **Feature**: Una columna dentro de una tabla es comúnmente referida como una característica.
 - **Sinónimos**: atributo, input, predictor, variable
- **Outcome**: Muchos proyectos de ciencia de datos involucran predecir un resultado, a menudo un resultado sí/no (en la Tabla 1-1, es "auction was competitive or not"). Las características a veces se utilizan para predecir el resultado en un experimento o estudio.
 - **Sinónimos**: variable dependiente, respuesta, objetivo, output

- **Records:** Una fila dentro de una tabla es comúnmente referida como un registro.

- **Sinónimos:** caso, ejemplo, instancia, observación, patrón, muestra

Tabla 1-1. Formato típico de un data frame

Category	Currency	SellerRating	Duration	EndDay	ClosePrice	OpenPrice	C
Music/Movie/Game	US	3249	5	Mon	0.01	0.01	0
Music/Movie/Game	US	3249	5	Mon	0.01	0.01	0
Automotive	US	3115	7	Tue	0.01	0.01	0
Automotive	US	3115	7	Tue	0.01	0.01	0
Automotive	US	3115	7	Tue	0.01	0.01	0
Automotive	US	3115	7	Tue	0.01	0.01	0
Automotive	US	3115	7	Tue	0.01	0.01	1
Automotive	US	3115	7	Tue	0.01	0.01	1
Automotive	US	3115	7	Tue	0.01	0.01	1
Automotive	US	3115	7	Tue	0.01	0.01	1

En la Tabla 1-1, hay una mezcla de datos medidos o contados (por ejemplo, duración y precio) y datos categóricos (por ejemplo, categoría y moneda). Como se mencionó anteriormente, una forma especial de variable categórica es una variable binaria (sí/no o 0/1), vista en la columna más a la derecha en la Tabla 1-1: una variable indicadora que muestra si una subasta fue competitiva (tuvo múltiples postores) o no. Esta variable indicadora también resulta ser una variable de resultado, cuando el escenario es predecir si una subasta es competitiva o no.



Estructuras de Datos No Rectangulares

Los **datos de series temporales** registran mediciones sucesivas de la misma variable. Es la materia prima para los métodos de pronóstico estadístico y también es un componente clave de los datos producidos por dispositivos: el Internet de las Cosas.

Las **estructuras de datos espaciales**, que se utilizan en el mapeo y análisis de localización, son más complejas y variadas que las estructuras de datos rectangulares.

En la **representación de objetos**, el foco de los datos es un objeto (por ejemplo, una casa) y sus coordenadas espaciales. La vista de campo, por el contrario, se centra en pequeñas unidades de espacio y el valor de una métrica relevante (por ejemplo, el brillo de un píxel).

Las **estructuras de datos de grafos (o redes)** se utilizan para representar relaciones físicas, sociales y abstractas. Por ejemplo, un grafo de una red social, como Facebook o

LinkedIn, puede representar conexiones entre personas en la red. Los centros de distribución conectados por carreteras son un ejemplo de una red física. Las estructuras de grafos son útiles para ciertos tipos de problemas, como la optimización de redes y los sistemas de recomendación.

1.1 Estimaciones de Ubicación (o medidas de tendencia central)

Las variables con datos medidos o contados pueden tener miles de valores distintos. Un paso básico en la exploración de tus datos es obtener un "valor típico" para cada característica (variable): una estimación de dónde se encuentra la mayor parte de los datos (es decir, su tendencia central).

Términos Clave para Estimaciones de Ubicación

- **Media:** La suma de todos los valores dividida por el número de valores.
 - Sinónimo: promedio
- **Media ponderada:** La suma de todos los valores multiplicados por un peso dividida por la suma de los pesos.
 - Sinónimo: promedio ponderado
- **Mediana:** El valor tal que la mitad de los datos está por encima y por debajo.
 - Sinónimo: percentil 50
- **Percentil:** El valor tal que P por ciento de los datos está por debajo.
 - Sinónimo: cuantil
- **Mediana ponderada:** El valor tal que la mitad de la suma de los pesos está por encima y por debajo de los datos ordenados.
- **Media recortada:** El promedio de todos los valores después de eliminar un número fijo de valores extremos.
 - Sinónimo: media truncada
- **Robusto:** No sensible a valores extremos.
 - Sinónimo: resistente
- **Valor atípico:** Un valor de datos que es muy diferente de la mayoría de los datos.
 - Sinónimo: valor extremo

A primera vista, resumir los datos podría parecer bastante trivial: simplemente toma la media de los datos. De hecho, aunque la media es fácil de calcular y expedita de usar, puede que no siempre sea la mejor medida para un valor central. Por esta razón, los estadísticos han desarrollado y promovido varias estimaciones alternativas a la media.

Métricas y Estimaciones

Los estadísticos a menudo usan el término estimación para un valor calculado a partir de los datos a la mano, para hacer una distinción entre lo que vemos en los datos y el estado teórico verdadero o exacto. Los científicos de datos y los analistas de negocios son más propensos a referirse a dicho valor como una métrica.

Media

La estimación más básica de la ubicación es la media, o valor promedio. La media es la suma de todos los valores dividida por el número de valores. Considera el siguiente conjunto de números: {3, 5, 1, 2}. La media es:

$$\text{Media} = \frac{3+5+1+2}{4} = \frac{11}{4} = 2.75$$

Verás que se usa el símbolo \bar{x} (pronunciado "x-barra") para representar la media de una muestra de una población. La fórmula para calcular la media para un conjunto de n valores x_1, x_2, \dots, x_n es:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

Ejemplos

Con SciPy

```
In [4]: import numpy as np
from scipy import stats

# Crear un conjunto de datos de ejemplo
data = np.array([3, 5, 1, 2, 7, 8, 10, 6, 4, 9])
data
```

```
Out[4]: array([ 3,  5,  1,  2,  7,  8, 10,  6,  4,  9])
```

```
In [5]: # Calcular la media usando scipy
mean_scipy = np.mean(data)
print("Media usando SciPy:", mean_scipy)
```

```
Media usando SciPy: 5.5
```

Usando Pandas

```
In [6]: import pandas as pd

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [3, 5, 1, 2, 7, 8, 10, 6, 4, 9]})
df
```

Out[6]:

	values
0	3
1	5
2	1
3	2
4	7
5	8
6	10
7	6
8	4
9	9

```
In [7]: # Calcular la media usando pandas
mean_pandas = df['values'].mean()
print("Media usando Pandas:", mean_pandas)
```

Media usando Pandas: 5.5

Usando Statsmodels

```
In [8]: import statsmodels.api as sm
import pandas as pd

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [3, 5, 1, 2, 7, 8, 10, 6, 4, 9]})

# Agregar una columna de unos para representar el intercepto
df['intercept'] = 1

# Ajustar un modelo de regresión sin variables explicativas, solo el intercepto
model = sm.OLS(df['values'], df['intercept']).fit()

# La media es el valor del intercepto
mean_statsmodels = model.params['intercept']
print("Media usando Statsmodels:", mean_statsmodels)
```

Media usando Statsmodels: 5.500000000000001

Calcula la media de los mismos datos dados arriba pero esta vez usando la librería pingouine

Una variación de la media es una media recortada (trimmed mean), que se calcula eliminando un número fijo de valores ordenados en cada extremo y luego tomando un promedio de los valores restantes. Representando los valores ordenados por x_1, x_2, \dots, x_n donde x_1 es el valor más pequeño y x_n el más grande, la fórmula para calcular la media recortada con p valores más pequeños y más grandes omitidos es:

$$\bar{x}_{\text{recortada}} = \frac{1}{n-2p} \sum_{i=p+1}^{n-p} x_i$$

Una media recortada elimina la influencia de valores extremos. Por ejemplo, en el buceo internacional se eliminan la puntuación más alta y más baja de cinco jueces, y la puntuación final es el promedio de las puntuaciones de los tres jueces restantes. Esto hace que sea difícil para un solo juez manipular la puntuación, quizás para favorecer al concursante de su país. Las medias recortadas se utilizan ampliamente y, en muchos casos, son preferibles a usar la media ordinaria.

Ejemplos con Python

SciPy

```
In [9]: import numpy as np
from scipy.stats import trim_mean

# Crear un conjunto de datos de ejemplo
data = np.array([3, 5, 1, 2, 7, 8, 10, 6, 4, 9])

# Calcular la media recortada usando scipy
trimmed_mean_scipy = trim_mean(data, 0.1) # Recorta el 10% de cada extremo
print("Media recortada usando SciPy:", trimmed_mean_scipy)
```

Media recortada usando SciPy: 5.5

Pandas

```
In [10]: import pandas as pd

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [3, 5, 1, 2, 7, 8, 10, 6, 4, 9]})

# Calcular la media recortada usando pandas
trimmed_df = df['values'].sort_values().iloc[int(0.1*len(df)):int(0.9*len(df))]
trimmed_mean_pandas = trimmed_df.mean()
print("Media recortada usando Pandas:", trimmed_mean_pandas)
```

Media recortada usando Pandas: 5.5

Explique el código de arriba

Statsmodels

No tiene un método directo. Se calcula de forma indirecta.

```
In [11]: import statsmodels.api as sm
import pandas as pd
import numpy as np

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [3, 5, 1, 2, 7, 8, 10, 6, 4, 9]})

# Usar una función personalizada para calcular la media recortada
def trimmed_mean(data, proportiontocut):
    n = len(data)
    k = int(n * proportiontocut)
    trimmed_data = np.sort(data)[k:n-k]
    return np.mean(trimmed_data)
```

```
# Calcular La media recortada usando La función personalizada
trimmed_mean_statsmodels = trimmed_mean(df['values'], 0.1)
print("Media recortada usando una función personalizada con Statsmodels:", trimm)
```

Media recortada usando una función personalizada con Statsmodels: 5.5

Explique el código de arriba

Investigar como se calcula la media recortada con pingouine

Otro tipo de media es una **media ponderada (weighted mean)**, que se calcula multiplicando cada valor de datos x_i por un peso especificado por el usuario w_i y dividiendo su suma por la suma de los pesos. La fórmula para una media ponderada es:

$$\bar{x}_w = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}$$

Hay dos motivaciones principales para usar una media ponderada:

- Algunos valores son intrínsecamente más variables que otros, y las observaciones altamente variables se les da un peso menor. Por ejemplo, si estamos tomando el promedio de múltiples sensores y uno de los sensores es menos preciso, entonces podríamos darle menos peso a los datos de ese sensor.
- Los datos recopilados no representan igualmente a los diferentes grupos que estamos interesados en medir. Por ejemplo, debido a la forma en que se realizó un experimento en línea, es posible que no tengamos un conjunto de datos que refleje con precisión todos los grupos en la base de usuarios. Para corregir eso, podemos dar un mayor peso a los valores de los grupos que estaban subrepresentados.

Ejemplos con Python

Vamos a utilizar un conjunto de datos simple donde cada valor tiene un peso asociado:

```
In [12]: values = np.array([3, 5, 1, 2, 7, 8, 10, 6, 4, 9])
weights = np.array([1, 2, 1, 1, 2, 1, 3, 2, 1, 2])
```

Scipy

```
In [13]: import numpy as np
from scipy.stats import describe

# Crear un conjunto de datos de ejemplo
values = np.array([3, 5, 1, 2, 7, 8, 10, 6, 4, 9])
weights = np.array([1, 2, 1, 1, 2, 1, 3, 2, 1, 2])

# Calcular la media ponderada usando scipy
weighted_mean_scipy = np.average(values, weights=weights)
print("Media ponderada usando SciPy:", weighted_mean_scipy)
```

Media ponderada usando SciPy: 6.375

Pandas

```
In [14]: import pandas as pd

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [3, 5, 1, 2, 7, 8, 10, 6, 4, 9],
                    'weights': [1, 2, 1, 1, 2, 1, 3, 2, 1, 2]})

# Calcular la media ponderada usando pandas
weighted_mean_pandas = np.average(df['values'], weights=df['weights'])
print("Media ponderada usando Pandas:", weighted_mean_pandas)
```

Media ponderada usando Pandas: 6.375

Statsmodels

```
In [15]: import statsmodels.api as sm
import pandas as pd
import numpy as np

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [3, 5, 1, 2, 7, 8, 10, 6, 4, 9],
                    'weights': [1, 2, 1, 1, 2, 1, 3, 2, 1, 2]})

# Añadir una columna de unos para representar el intercepto
df['intercept'] = 1

# Ajustar un modelo de regresión ponderado
model = sm.WLS(df['values'], df[['intercept']], weights=df['weights']).fit()

# La media ponderada es el valor del intercepto
weighted_mean_statsmodels = model.params['intercept']
print("Media ponderada usando Statsmodels:", weighted_mean_statsmodels)
```

Media ponderada usando Statsmodels: 6.3750000000000036

Explicar código de arriba

Investigar cómo se calcula media ponderada usando pingouine

Mediana y Estimaciones Robustas

La mediana es el número central en una lista ordenada de los datos. Si hay un número par de valores de datos, el valor central es uno que no está realmente en el conjunto de datos, sino el promedio de los dos valores que dividen los datos ordenados en mitades superior e inferior. En comparación con la media, que utiliza todas las observaciones, la mediana depende solo de los valores en el centro de los datos ordenados. Aunque esto podría parecer una desventaja, ya que la media es mucho más sensible a los datos, hay muchos casos en los que la mediana es una mejor métrica para la ubicación.

Supongamos que queremos observar los ingresos familiares típicos en vecindarios alrededor del Lago Washington en Seattle. Al comparar el vecindario de Medina con el de Windermere, usar la media produciría resultados muy diferentes porque Bill Gates

vive en Medina. Si usamos la mediana, no importará cuán rico sea Bill Gates, la posición de la observación central seguirá siendo la misma.

Por las mismas razones que se usa una media ponderada, también es posible calcular una mediana ponderada. Al igual que con la mediana, primero ordenamos los datos, aunque cada valor de datos tiene un peso asociado. En lugar del número central, la mediana ponderada es un valor tal que la suma de los pesos es igual para las mitades inferior y superior de la lista ordenada. Al igual que la mediana, la mediana ponderada es robusta frente a valores atípicos.

Ejemplos con Python

SciPy (Numpy)

```
In [16]: import numpy as np

# Crear un conjunto de datos de ejemplo
data = np.array([3, 5, 1, 2, 7, 8, 10, 6, 4, 9])

# Calcular la mediana usando numpy (parte de scipy)
median_scipy = np.median(data)
print("Mediana usando 'SciPy':", median_scipy)
```

Mediana usando 'SciPy': 5.5

Pandas (completar)

```
In [ ]:
```

pingouine (completar)

```
In [ ]:
```

Statsmodels

Statsmodels no tiene una función directa para calcular la mediana. Sin embargo, podemos usar una `regresión cuantílica` para calcularla, ya que la mediana es el cuantil 0.5.

```
In [17]: import statsmodels.api as sm
import pandas as pd
import numpy as np

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [3, 5, 1, 2, 7, 8, 10, 6, 4, 9]})

# Ajustar un modelo de regresión cuantílica para calcular La mediana
quant_reg = sm.QuantReg(df['values'], np.ones(len(df['values'])))
res = quant_reg.fit(q=0.5)

# La mediana es el coeficiente del modelo
median_statsmodels = res.params[0]
print("Mediana usando Statsmodels:", median_statsmodels)
```

Mediana usando Statsmodels: 5.499999999999998

```
C:\Users\juanj\AppData\Local\Temp\ipykernel_11180\768756127.py:13: FutureWarning:  
Series.__getitem__ treating keys as positions is deprecated. In a future version,  
integer keys will always be treated as labels (consistent with DataFrame behavio  
r). To access a value by position, use `ser.iloc[pos]`  
median_statsmodels = res.params[0]
```

Valores Atípicos

La mediana se refiere como una estimación robusta de la ubicación ya que no se ve influenciada por valores atípicos (casos extremos) que podrían sesgar los resultados. Un valor atípico es cualquier valor que está muy distante de los otros valores en un conjunto de datos. La definición exacta de un valor atípico es algo subjetiva, aunque ciertas convenciones se utilizan en varios resúmenes de datos y gráficos. Ser un valor atípico en sí mismo no hace que un valor de datos sea inválido o erróneo. Aun así, los valores atípicos a menudo son el resultado de errores de datos como mezclar datos de diferentes unidades (kilómetros frente a metros) o lecturas incorrectas de un sensor. Cuando los valores atípicos son el resultado de datos incorrectos, la media dará lugar a una mala estimación de la ubicación, mientras que la mediana seguirá siendo válida. En cualquier caso, los valores atípicos deben ser identificados y generalmente merecen una investigación adicional.

Detección de Anomalías

En contraste con el análisis típico de datos, donde los valores atípicos son a veces informativos y otras veces una molestia, en la detección de anomalías los puntos de interés son los valores atípicos, y la mayor parte de los datos sirve principalmente para definir lo "normal" contra lo cual se miden las anomalías.

La mediana no es la única estimación robusta de la ubicación. De hecho, una media recortada se utiliza ampliamente para evitar la influencia de valores atípicos. Por ejemplo, recortar el 10% inferior y superior (una elección común) de los datos proporcionará protección contra valores atípicos en todos los conjuntos de datos, excepto en los más pequeños. La media recortada puede considerarse un compromiso entre la mediana y la media: es robusta frente a valores extremos en los datos, pero utiliza más datos para calcular la estimación de la ubicación.

Ejercicio 1: Estimaciones de Ubicación de la Población y las Tasas de Homicidios

La Tabla 1-2 muestra las primeras filas del conjunto de datos que contiene la población y las tasas de homicidios (en unidades de asesinatos por cada 100,000 personas por año) para cada estado de EE. UU. (Censo de 2010).

Estado	Población	Tasa de Homicidios	Abreviatura
Alabama	4,779,736	5.7	AL
Alaska	710,231	5.6	AK

Estado	Población	Tasa de Homicidios	Abreviatura
Arizona	6,392,017	4.7	AZ
Arkansas	2,915,918	5.6	AR
California	37,253,956	4.4	CA
Colorado	5,029,196	2.8	CO
Connecticut	3,574,097	2.4	CT
Delaware	897,934	5.8	DE

Calcula la media, la media recortada, la media ponderada y la mediana para la población usando Pandas, Scipy y Pingouine (Opcional: Statsmodels). Utilice el fichero `state.csv`

Ejercicio 2. Investigar cómo se interpreta estadísticamente la medida de tendencia central: moda .

Buscar ejemplos concretos donde se aplique la moda. Desarrolle un ejemplo con Pandas, Scipy, Statsmodels y Pingouine

1.2 Estimaciones de Variabilidad o medidas de variación

La ubicación es solo una dimensión para resumir una característica. Una segunda dimensión, la variabilidad, también conocida como dispersión, mide si los valores de los datos están estrechamente agrupados o dispersos. En el corazón de la estadística se encuentra la variabilidad: medirla, reducirla, distinguir la variabilidad aleatoria de la real, identificar las diversas fuentes de variabilidad real y tomar decisiones en su presencia.

Términos Clave para Métricas de Variabilidad

- **Desviaciones (error absoluto):** La diferencia entre los valores observados y la estimación de ubicación.
 - Sinónimos: errores, residuales.
- **Varianza s^2 :** La suma de las desviaciones al cuadrado con respecto a la media, dividida por $n - 1$, donde n es el número de valores de datos.
 - Sinónimo: error cuadrático medio.
- **Desviación estándar:** La raíz cuadrada de la varianza.
- **Desviación absoluta media:** La media de los valores absolutos de las desviaciones con respecto a la media.
 - Sinónimos: norma L1, norma Manhattan.
- **Desviación absoluta mediana desde la mediana:** La mediana de los valores absolutos de las desviaciones con respecto a la mediana.

- **Rango:** La diferencia entre el valor más grande y el más pequeño en un conjunto de datos.
- **Estadísticas de orden:** Métricas basadas en los valores de los datos ordenados de menor a mayor.
 - Sinónimo: rangos.
- **Percentil:** El valor tal que P por ciento de los valores toman este valor o menos y (100 - P) por ciento toman este valor o más.
 - Sinónimo: cuantil.
- **Rango intercuartílico (IQR):** La diferencia entre el percentil 75 y el percentil 25.
 - Sinónimo: IQR.

Así como hay diferentes formas de medir la ubicación (media, mediana, etc.), también hay diferentes formas de medir la variabilidad.

Desviación Estándar y Estimaciones Relacionadas

Las estimaciones de variación más utilizadas se basan en las diferencias, o desviaciones, entre la estimación de la ubicación y los datos observados. Para un conjunto de datos {1, 4, 4}, la media es 3 y la mediana es 4. Las desviaciones con respecto a la media son las diferencias: $1 - 3 = -2$, $4 - 3 = 1$, $4 - 3 = 1$. Estas desviaciones nos dicen cuán dispersos están los datos alrededor del valor central.

Una forma de medir la variabilidad es estimar un valor típico para estas desviaciones. Promediar las desviaciones en sí mismas no nos diría mucho: las desviaciones negativas compensan las positivas. De hecho, la suma de las desviaciones con respecto a la media es precisamente cero. En cambio, un enfoque simple es tomar el promedio de los valores absolutos de las desviaciones con respecto a la media. En el ejemplo anterior, el valor absoluto de las desviaciones es {2, 1, 1}, y su promedio es $(2 + 1 + 1) / 3 = 1.33$. Esto se conoce como desviación absoluta media y se calcula con la fórmula:

$$\text{Desviación absoluta media} = \frac{\sum_{i=1}^n |x_i - \bar{x}|}{n}$$

donde \bar{x} es la media de la muestra.

Ejemplo de Cálculo de la Desviación Absoluta Media.

Supongamos que tenemos los siguientes datos de ejemplo:

$$x = [1, 2, 3, 4, 5, 6, 7, 8, 100]$$

1. Calcular la Media:

$$\text{media} = \frac{1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 100}{9} = \frac{136}{9} \approx 15.11$$

2. Calcular las Desviaciones Absolutas de la Media:

$$|1 - 15.11|, |2 - 15.11|, |3 - 15.11|, |4 - 15.11|, |5 - 15.11|, |6 - 15.11|, |7 - 15.11|$$

$$\approx 14.11, 13.11, 12.11, 11.11, 10.11, 9.11, 8.11, 7.11, 84.89$$

3. Calcular la Media de las Desviaciones Absolutas (MAD):

$$\text{MAD} = \frac{14.11 + 13.11 + 12.11 + 11.11 + 10.11 + 9.11 + 8.11 + 7.11 + 84.89}{9} \approx$$

Interpretación

En este ejemplo, la MAD es aproximadamente 18.31. Esto significa que, en promedio, las desviaciones absolutas de los datos respecto a la media son de 18.31 unidades. La MAD indica la dispersión promedio de los datos alrededor de la media y puede ser influenciada por valores atípicos.

Ejemplos con Python

Con Scipy (Numpy)

```
In [68]: import numpy as np

# Crear un conjunto de datos de ejemplo
data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 100])

# Calcular la desviación absoluta media usando numpy (parte de scipy)
mean = np.mean(data)
mad_scipy = np.mean(np.abs(data - mean))
print("Desviación absoluta media usando SciPy:", mad_scipy)
```

Desviación absoluta media usando SciPy: 18.864197530864196

Con Pandas

```
In [69]: import pandas as pd
import numpy as np

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [1, 2, 3, 4, 5, 6, 7, 8, 100]})

# Calcular la desviación absoluta media usando pandas
mean = df['values'].mean()
mad_pandas = np.mean(np.abs(df['values'] - mean))
print("Desviación absoluta media usando Pandas:", mad_pandas)
```

Desviación absoluta media usando Pandas: 18.864197530864196

Con Statsmodels

```
In [70]: import statsmodels.api as sm
import pandas as pd
import numpy as np

df = pd.DataFrame({'values': [1, 2, 3, 4, 5, 6, 7, 8, 100]})
```

```
# Calcular la media
mean = df['values'].mean()

# Calcular la desviación absoluta media manualmente
mad_mean_statsmodels = np.mean(np.abs(df['values'] - mean))
print("Desviación absoluta media usando Statsmodels (manual):", mad_mean_statsmo
```

Desviación absoluta media usando Statsmodels (manual): 18.864197530864196

Calcular usando pingouine

In []:

Varianza

Las estimaciones más conocidas de la variabilidad son la varianza y la desviación estándar, que se basan en desviaciones al cuadrado. La varianza es un promedio de las desviaciones al cuadrado y la desviación estándar es la raíz cuadrada de la varianza:

$$\text{Varianza} = s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}$$

La varianza es una medida de dispersión que indica cuánto varían los datos respecto a su media. Se calcula como el promedio de las desviaciones al cuadrado de cada valor con respecto a la media.

1. Calcular la Media:

$$\text{media} = \frac{1}{n} \sum_{i=1}^n x_i$$

2. Calcular las Desviaciones al Cuadrado:

$$(x_i - \text{media})^2$$

3. Calcular la Varianza:

$$\text{Varianza} = \frac{1}{n} \sum_{i=1}^n (x_i - \text{media})^2$$

Interpretación de la Varianza

- **Medida de Dispersión:** La varianza mide cuánto se dispersan los valores en un conjunto de datos en relación con la media. Valores de varianza más altos indican mayor dispersión.
- **Sensibilidad a Outliers:** La varianza es sensible a valores atípicos porque las desviaciones al cuadrado pueden aumentar significativamente debido a valores extremos.
- **Unidades Cuadradas:** La varianza se expresa en las mismas unidades que los datos originales, pero al cuadrado. Esto puede dificultar la interpretación directa de la

varianza en comparación con la desviación estándar, que está en las mismas unidades que los datos originales.

Ejemplo Numérico

Supongamos que tenemos los siguientes datos de ejemplo:

$$x = [1, 2, 3, 4, 5, 6, 7, 8, 100]$$

Calcular la Media:

$$\text{media} = \frac{1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 100}{9} = \frac{136}{9} \approx 15.11$$

Calcular las Desviaciones al Cuadrado:

$$(1 - 15.11)^2, (2 - 15.11)^2, (3 - 15.11)^2, (4 - 15.11)^2, (5 - 15.11)^2, (6 - 15.11)^2, (7 - 15.11)^2, (8 - 15.11)^2, (100 - 15.11)^2$$

$$\approx 198.24, 171.14, 145.03, 119.93, 95.82, 72.72, 50.61, 29.51, 7205.45$$

Calcular la Varianza:

$$\text{Varianza} = \frac{198.24 + 171.14 + 145.03 + 119.93 + 95.82 + 72.72 + 50.61 + 29.51 + 7205.45}{9}$$

Ejemplos en Python

Con Pandas

```
In [73]: import pandas as pd

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [1, 2, 3, 4, 5, 6, 7, 8, 100]})

# Calcular la varianza usando pandas
var_pandas = df['values'].var(ddof=0) # ddof=0 para la varianza de la población
print("Varianza usando Pandas:", var_pandas)
```

Varianza usando Pandas: 905.4320987654321

Con SciPy

```
In [74]: from scipy import stats

# Calcular la varianza usando scipy
var_scipy = stats.tvar(data, ddof=0) # ddof=0 para la varianza de la población
print("Varianza usando SciPy:", var_scipy)
```

Varianza usando SciPy: 905.4320987654321

Con Statsmodels (NumPy)

```
In [78]: import statsmodels.api as sm
import numpy as np
import pandas as pd
```

```
# Crear un conjunto de datos de ejemplo
data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 100])
df = pd.DataFrame({'values': data})

# Calcular la media
mean = df['values'].mean()

# Calcular la varianza manualmente usando statsmodels
var_statsmodels = np.mean((df['values'] - mean) ** 2)
print("Varianza usando Statsmodels:", var_statsmodels)
```

Varianza usando Statsmodels: 905.4320987654321

Con pingouine (Completar)

In []:

Desviación estándar o típica.

$$\text{Desviación estándar} = s = \sqrt{\text{Varianza}}$$

La desviación estándar es mucho más fácil de interpretar que la varianza, ya que está en la misma escala que los datos originales. Aún así, con su fórmula más complicada y menos intuitiva, podría parecer peculiar que la desviación estándar sea preferida en estadística sobre la desviación absoluta media. Debe su preeminencia a la teoría estadística: matemáticamente, trabajar con valores al cuadrado es mucho más conveniente que con valores absolutos, especialmente para modelos estadísticos.

Para un conjunto de datos:

$$x = [x_1, x_2, \dots, x_n]:$$

1. Calcular la Media:

$$\text{media} = \frac{1}{n} \sum_{i=1}^n x_i$$

2. Calcular las Desviaciones al Cuadrado:

$$(x_i - \text{media})^2$$

3. Calcular la Varianza:

$$\text{Varianza} = \frac{1}{n} \sum_{i=1}^n (x_i - \text{media})^2$$

4. Calcular la Desviación Estándar:

$$\text{Desviación Estándar} = \sqrt{\text{Varianza}}$$

Interpretación de la Desviación Estándar

- **Medida de Dispersión:** La desviación estándar mide cuánto se dispersan los valores en un conjunto de datos en relación con la media. Valores de desviación estándar más altos indican mayor dispersión.
- **Comparabilidad:** A diferencia de la varianza, la desviación estándar se expresa en las mismas unidades que los datos originales, lo que facilita su interpretación.
- **Sensibilidad a Outliers:** Al igual que la varianza, la desviación estándar es sensible a valores atípicos porque se basa en las desviaciones al cuadrado.

Ejemplo Numérico

Supongamos que tenemos los siguientes datos de ejemplo:

$$x = [1, 2, 3, 4, 5, 6, 7, 8, 100]$$

1. Calcular la Media:

$$\text{media} = \frac{1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 100}{9} = \frac{136}{9} \approx 15.11$$

2. Calcular las Desviaciones al Cuadrado:

$$(1 - 15.11)^2, (2 - 15.11)^2, (3 - 15.11)^2, (4 - 15.11)^2, (5 - 15.11)^2, (6 - 15.11)^2, (7 - 15.11)^2, (8 - 15.11)^2, (100 - 15.11)^2$$

$$\approx 198.24, 171.14, 145.03, 119.93, 95.82, 72.72, 50.61, 29.51, 7205.45$$

3. Calcular la Varianza:

$$\text{Varianza} = \frac{198.24 + 171.14 + 145.03 + 119.93 + 95.82 + 72.72 + 50.61 + 29.51}{9}$$

4. Calcular la Desviación Estándar:

$$\text{Desviación Estándar} = \sqrt{905.27} \approx 30.08$$



SciPy (Numpy)

```
In [80]: import numpy as np
from scipy import stats

# Crear un conjunto de datos de ejemplo
data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 100])

# Calcular la desviación estándar usando numpy (parte de scipy)
std_dev_scipy = np.std(data, ddof=1) # ddof=1 para una muestra
print("Desviación estándar usando SciPy:", std_dev_scipy)
```

Desviación estándar usando SciPy: 31.915687539376478

Pandas

```
In [81]: import pandas as pd

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [1, 2, 3, 4, 5, 6, 7, 8, 100]})
```

```
# Calcular la desviación estándar usando pandas
std_dev_pandas = df['values'].std()
print("Desviación estándar usando Pandas:", std_dev_pandas)
```

Desviación estándar usando Pandas: 31.915687539376478

Statsmodels

```
In [82]: import statsmodels.api as sm
import pandas as pd
import numpy as np

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': [1, 2, 3, 4, 5, 6, 7, 8, 100]})

# Agregar una columna de unos para representar el intercepto
df['intercept'] = 1

# Ajustar un modelo OLS (mínimos cuadrados ordinarios)
model = sm.OLS(df['values'], df['intercept']).fit()

# Calcular la desviación estándar de los residuales
residuals = model.resid
std_dev_statsmodels = np.std(residuals, ddof=1) # ddof=1 para una muestra
print("Desviación estándar usando Statsmodels (de los residuales):", std_dev_st)
```

Desviación estándar usando Statsmodels (de los residuales): 31.915687539376478

Pingouin (completar)

In []:

Desviación absoluta mediana

Ni la varianza, ni la desviación estándar, ni la desviación absoluta media son robustas frente a valores atípicos y extremos . La varianza y la desviación estándar son especialmente sensibles a los valores atípicos.

Una estimación robusta de la variabilidad es la desviación absoluta mediana :

Desviación absoluta mediana = Mediana ($|x_1 - m|, |x_2 - m|, \dots, |x_N - m|$)
donde m es la mediana. Al igual que la mediana, el MAD no se ve influenciado por valores extremos.

Se utiliza para detectar valores atípicos (outliers) en un conjunto de datos . La detección de outliers se basan en establecer un valor umbral. Si los valores superan dicho umbral entonces se consideran outliers (Ver ejemplo 2).

Ejemplo Numérico de Cálculo de la Desviación Absoluta de la Mediana (MAD)

Supongamos que tenemos el siguiente conjunto de datos:

$$x = [3, 1, 5, 3, 6, 7, 2, 9, 150]$$

Paso 1: Calcular la Mediana

Primero, ordenamos los datos en orden ascendente:

$$x_{\text{ordenado}} = [1, 2, 3, 3, 5, 6, 7, 9, 150]$$

La mediana es el valor que divide el conjunto de datos en dos partes iguales. Dado que tenemos un número impar de observaciones (9), la mediana es la observación central:

$$\text{Mediana} = x_5 = 5$$

Paso 2: Calcular las Desviaciones Absolutas respecto a la Mediana

Calculamos la desviación absoluta de cada valor respecto a la mediana:

$$\begin{aligned}|1 - 5| &= 4 \\ |2 - 5| &= 3 \\ |3 - 5| &= 2 \\ |3 - 5| &= 2 \\ |5 - 5| &= 0 \\ |6 - 5| &= 1 \\ |7 - 5| &= 2 \\ |9 - 5| &= 4 \\ |150 - 5| &= 145\end{aligned}$$

Las desviaciones absolutas son:

$$\text{Desviaciones Absolutas} = [4, 3, 2, 2, 0, 1, 2, 4, 145]$$

Paso 3: Calcular la Mediana de las Desviaciones Absolutas

Ordenamos las desviaciones absolutas en orden ascendente:

$$\text{Desviaciones Absolutas Ordenadas} = [0, 1, 2, 2, 2, 3, 4, 4, 145]$$

La mediana de las desviaciones absolutas es la observación central (dado que hay un número impar de observaciones):

$$\text{MAD} = 2$$

Interpretación

La Desviación Absoluta Mediana (MAD) es una medida robusta de la variabilidad de los datos, que no se ve influenciada por valores atípicos. En este ejemplo, la MAD es 2.

Ejemplos en Python

SciPy

```
In [1]: import numpy as np

# Crear un conjunto de datos de ejemplo
data = np.array([3, 1, 5, 3, 6, 7, 2, 9, 150])

# Calcular la mediana
median = np.median(data)

# Calcular la desviación absoluta mediana (MAD) usando numpy
mad_numpy = np.median(np.abs(data - median))
print("Desviación absoluta mediana usando NumPy:", mad_numpy)
```

Desviación absoluta mediana usando NumPy: 2.0

Pandas

```
In [2]: import pandas as pd
import numpy as np

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': data})

# Calcular la mediana
median = df['values'].median()

# Calcular la desviación absoluta mediana (MAD) usando pandas
mad_pandas = np.median(np.abs(df['values'] - median))
print("Desviación absoluta mediana usando Pandas:", mad_pandas)
```

Desviación absoluta mediana usando Pandas: 2.0

```
In [3]: import statsmodels.api as sm
import pandas as pd
import numpy as np

# Crear un DataFrame de ejemplo
df = pd.DataFrame({'values': data})

# Calcular la desviación absoluta mediana (MAD) usando statsmodels sin escala
mad_statsmodels = sm.robust.scale.mad(df['values'], c=1)
print("Desviación absoluta mediana usando Statsmodels:", mad_statsmodels)
```

Desviación absoluta mediana usando Statsmodels: 2.0

Detección de outliers usando MAD (similar a z-score modificado)

Caso simple: Detección de outliers en precios de casas

Dataset:

```
In [34]: import pandas as pd
import numpy as np

data = {
    'precio_casa': [250000, 300000, 150000, 200000, 500000, 230000, 260000, 2700
# Incluye un outlier: 1000000]
```

```

}
df = pd.DataFrame(data)
print(df)

```

	precio_casa
0	250000
1	300000
2	150000
3	200000
4	500000
5	230000
6	260000
7	270000
8	280000
9	220000
10	1000000

1- Calcular la Mediana y la Desviación Absoluta Mediana (MAD)

```

In [35]: # Calcular la mediana
mediana = df['precio_casa'].median()
print(f"Mediana del precio de las casas: {mediana}")

# Calcular la desviación absoluta mediana (MAD)
mad = np.median(np.abs(df['precio_casa'] - mediana))
print(f"Desviación absoluta mediana (MAD): {mad}")

```

Mediana del precio de las casas: 260000.0
 Desviación absoluta mediana (MAD): 40000.0

2-Detectar Outliers usando la MAD

Establecemos un umbral basado en la MAD para detectar outliers. Un umbral común es **3 veces** la MAD.

```

In [36]: # Establecer el umbral para detectar outliers
umbral = 3 * mad
print(f"Umbral para detectar outliers: {umbral}")

# Detectar outliers
outliers = df[np.abs(df['precio_casa'] - mediana) > umbral]
print("Outliers detectados usando MAD:")
print(outliers)

```

Umbral para detectar outliers: 120000.0
 Outliers detectados usando MAD:

	precio_casa
4	500000
10	1000000

Cálculo del Umbral para Detectar Outliers

La fórmula para calcular el umbral es:

$$\text{Umbral} = k \times \text{MAD}$$

donde (k) es un factor de escala (comúnmente 3).

Pasos para Calcular el Umbral y Detectar Outliers

1. Calcular la Mediana ((m)) del Conjunto de Datos:

$$m = \text{mediana}(x_1, x_2, \dots, x_n)$$

2. Calcular las Desviaciones Absolutas respecto a la Mediana:

$$|x_i - m|$$

3. Calcular la Mediana de las Desviaciones Absolutas (MAD):

$$\text{MAD} = \text{mediana}(|x_1 - m|, |x_2 - m|, \dots, |x_n - m|)$$

4. Establecer el Umbral para Detectar Outliers:

$$\text{Umbral} = k \times \text{MAD}$$

5. Identificar Valores que Exceden el Umbral:

Los valores que se consideran outliers son aquellos donde:

$$|x_i - m| > \text{Umbral}$$

Ejemplo numérico para detectar outliers.

Partiendo del mismo conjunto de datos:

$$x = [3, 1, 5, 3, 6, 7, 2, 9, 150]$$

Paso 1: Calcular la Mediana

$$m = x_5 = 5$$

Paso 2: Calcular las Desviaciones Absolutas respecto a la Mediana

Las desviaciones absolutas son:

$$\text{Desviaciones Absolutas} = [4, 3, 2, 2, 0, 1, 2, 4, 145]$$

Paso 3: Calcular la Mediana de las Desviaciones Absolutas (MAD)

Sabemos por el ejemplo anterior que la mediana de las desviaciones absolutas es:

$$\text{MAD} = 2$$

Paso 4: Establecer el Umbral para Detectar Outliers

Usamos (**k = 3**) para calcular el umbral:

$$\text{Umbral} = k \times \text{MAD} = 3 \times 2 = 6$$

Paso 5: Identificar Valores que Exceden el Umbral

Los valores que se consideran outliers son aquellos donde:

$$|x_i - m| > \text{Umbral}$$

Aplicamos esto a nuestras desviaciones absolutas:

$ 1 - 5 = 4$	(No es outlier)
$ 2 - 5 = 3$	(No es outlier)
$ 3 - 5 = 2$	(No es outlier)
$ 3 - 5 = 2$	(No es outlier)
$ 5 - 5 = 0$	(No es outlier)
$ 6 - 5 = 1$	(No es outlier)
$ 7 - 5 = 2$	(No es outlier)
$ 9 - 5 = 4$	(No es outlier)
$ 150 - 5 = 145$	(Es outlier)

El valor que excede el umbral de 6 es 150, por lo que se considera un outlier.

Ajuste del Factor de Escala (k)

El factor de escala (k) es crucial en la detección de outliers. Tradicionalmente, se utiliza (k = 3) porque en una distribución normal aproximadamente el 99.7% de los datos se encuentran dentro de tres desviaciones estándar de la media. Sin embargo, el valor óptimo de (k) puede variar según la naturaleza y distribución de los datos. Es posible que para datos con distribuciones no normales o con una cantidad significativa de ruido, un (k) diferente sea más adecuado.

Caso donde $k = 3$ no es conveniente

Consideremos un conjunto de datos de temperaturas diarias en una ciudad a lo largo de un año, con algunos días extremadamente cálidos o fríos que podrían ser considerados outliers.

```
In [8]: import pandas as pd
import numpy as np

# Crear un DataFrame de ejemplo con temperaturas
np.random.seed(0)
temperaturas = np.random.normal(loc=20, scale=5, size=365).tolist() # Temperatura
temperaturas.extend([40, 42, -10, -12]) # Añadir algunos outliers
data = {
    'temperatura': temperaturas
}
df = pd.DataFrame(data)
df
```

Out[8]:

temperatura	
0	28.820262
1	22.000786
2	24.893690
3	31.204466
4	29.337790
...	...
364	19.921589
365	40.000000
366	42.000000
367	-10.000000
368	-12.000000

369 rows × 1 columns

In [9]:

```
# Calcular la mediana
mediana = df['temperatura'].median()
print(f"Mediana de las temperaturas: {mediana}")

# Calcular la desviación absoluta mediana (MAD)
mad = np.median(np.abs(df['temperatura'] - mediana))
print(f"Desviación absoluta mediana (MAD): {mad}")
```

Mediana de las temperaturas: 19.73716351865227
 Desviación absoluta mediana (MAD): 3.390552282976394

Ajuste del factor de k

Podemos utilizar técnicas exploratorias o basarnos en conocimientos específicos del dominio. En este ejemplo, ajustaremos k empíricamente.

In [10]:

```
# Ajustar el factor k empíricamente
for k in [1, 2, 3, 4, 5]:
    umbral = k * mad
    outliers = df[np.abs(df['temperatura'] - mediana) > umbral]
    print(f"Con k = {k}, el umbral es {umbral}. Outliers detectados:")
    print(outliers)
```

Con $k = 1$, el umbral es 3.390552282976394. Outliers detectados:

	temperatura
0	28.820262
2	24.893690
3	31.204466
4	29.337790
5	15.113611
..	...
362	24.659242
365	40.000000
366	42.000000
367	-10.000000
368	-12.000000

[184 rows x 1 columns]

Con $k = 2$, el umbral es 6.781104565952788. Outliers detectados:

	temperatura
0	28.820262
3	31.204466
4	29.337790
11	27.271368
16	27.470395
..	...
356	10.784652
365	40.000000
366	42.000000
367	-10.000000
368	-12.000000

[67 rows x 1 columns]

Con $k = 3$, el umbral es 10.171656848929182. Outliers detectados:

	temperatura
3	31.204466
20	7.235051
24	31.348773
144	31.915724
183	8.882984
198	30.816180
218	31.296545
271	6.137036
279	30.322464
292	31.519583
327	31.283617
334	6.704139
365	40.000000
366	42.000000
367	-10.000000
368	-12.000000

Con $k = 4$, el umbral es 13.562209131905576. Outliers detectados:

	temperatura
271	6.137036
365	40.000000
366	42.000000
367	-10.000000
368	-12.000000

Con $k = 5$, el umbral es 16.95276141488197. Outliers detectados:

	temperatura
365	40.0
366	42.0

367	-10.0
368	-12.0

Para este conjunto de datos, (**k = 5**) es el valor más adecuado. Detecta las temperaturas inusualmente altas y bajas

Otras técnicas de detección de outliers:

- **Gráficos de Caja (Boxplots)** : Visualizan la dispersión y los outliers en los datos.
- **Histogramas** : Muestran la distribución de los datos y pueden ayudar a identificar si hay colas largas o distribuciones sesgadas.
- **Validación Cruzada** : La validación cruzada puede ayudar a evaluar el impacto de diferentes valores de

k en el rendimiento de un modelo predictivo. Se puede entrenar un modelo con diferentes conjuntos de datos y comparar los resultados.

- **Métodos Estadísticos Robustos** :
 - **Regresión Robust** (e.g., RANSAC, Theil-Sen) : Técnicas de regresión que son menos sensibles a outliers.
 - **Estimadores de M-Estimación** : Se ajustan iterativamente para reducir la influencia de outliers.
- **Análisis de Componentes Principales (PCA)** : El PCA puede ayudar a identificar outliers en datos multivariados reduciendo la dimensionalidad y visualizando las observaciones en el espacio de componentes principales.

Ejercicio 3. Con la misma dataset del ejercicio 2 (`state.csv`) calcule todas las medidas de dispersión dadas en esta en clase (incluyendo el cálculo del umbral para detectar outliers usando la desviación absoluta mediana) sobre la variable Población. Utilizar Pandas, Numpy y SciPy. Interpretar resultados.

Estimaciones Basadas en Percentiles

Un enfoque diferente para estimar la dispersión se basa en observar la extensión de los datos ordenados. Las estadísticas basadas en datos ordenados (clasificados) se denominan **estadísticas de orden**. La medida más básica es el **rango** : la diferencia entre los números más grandes y más pequeños. Los valores mínimos y máximos en sí mismos son útiles y ayudan a identificar valores atípicos, pero el rango es extremadamente sensible a los valores atípicos y no es muy útil como una medida general de dispersión en los datos.

Ejemplo Numérico de Cálculo de Percentiles y Rango Intercuartílico

Supongamos que tenemos los siguientes datos de ejemplo:

$$x = [3, 1, 5, 3, 6, 7, 2, 9]$$

Ordenar los Datos

Primero, ordenamos los datos en orden ascendente:

$$x_{\text{ordenado}} = [1, 2, 3, 3, 5, 6, 7, 9]$$

Calcular los Percentiles

Un percentil es un valor tal que al menos el P por ciento (por ejemplo 25%) de los valores tienen este valor o menos y al menos (100 - P) [100-25=75%] por ciento de los valores tienen este valor o más.

- **Percentil 25 (P25):**

- Hay 8 datos en total.
- La posición del percentil 25 se calcula como: posición = $P \times (n + 1)$

donde:

- P es el percentil expresado como una fracción (por ejemplo, 0.25 para el percentil 25).
- n es el número de datos. Por lo tanto:

$$\begin{aligned} \text{posición} &= 0.25 \times (n + 1) \\ P_{25} &= 0.25 \times (8 + 1) = 0.25 \times 9 = 2.25 \end{aligned}$$

- Interpolamos entre los valores en las posiciones 2 y 3:

$$P_{25} = 2 + 0.25 \times (3 - 2) = 2 + 0.25 \times 1 = 2.25$$

La fórmula de interpolación lineal para calcular un percentil P se puede expresar como:

$$P = x_j + w \times (x_{j+1} - x_j)$$

donde:

- x_j : es el valor en la posición entera inferior.
- x_{j+1} : es el valor en la posición entera superior.
- w : es la parte fraccionaria de la posición del percentil.

El peso w es la parte fraccionaria de la posición:

$$w = \text{posición} - \lfloor \text{posición} \rfloor$$

donde $\lfloor \text{posición} \rfloor$ es la parte entera de la posición.

- **Percentil 50 (P50 o Mediana):**

- La posición del percentil 50 se calcula como $0.50 \times (n + 1)$:

$$P_{50} = 0.50 \times (8 + 1) = 0.50 \times 9 = 4.5$$

- Interpolamos entre los valores en las posiciones 4 y 5:

$$P_{50} = 3 + 0.5 \times (5 - 3) = 3 + 0.5 \times 2 = 4$$

- **Percentil 75 (P75):**

- La posición del percentil 75 se calcula como $0.75 \times (n + 1)$:

$$P_{75} = 0.75 \times (8 + 1) = 0.75 \times 9 = 6.75$$

- Interpolamos entre los valores en las posiciones 6 y 7 :

$$P_{75} = 6 + 0.75 \times (7 - 6) = 6 + 0.75 \times 1 = 6.75$$

Calcular el Rango Intercuartílico (IQR)

El rango intercuartílico es la diferencia entre el percentil 75 y el percentil 25:

$$\text{IQR} = P_{75} - P_{25}$$

$$\text{IQR} = 6.75 - 2.25 = 4.5$$

Interpretación Estadística

- **Percentiles:**

- Los percentiles dividen los datos en partes iguales. Por ejemplo, el percentil 25 (P25) indica que el 25% de los datos son menores o iguales a 2.25. El percentil 50 (P50 o mediana) indica que el 50% de los datos son menores o iguales a 4. El percentil 75 (P75) indica que el 75% de los datos son menores o iguales a 6.75.
- Los percentiles son útiles para entender la distribución de los datos y para identificar valores extremos.

- **Rango:**

- El rango es la diferencia entre el valor máximo y el valor mínimo de los datos. En este ejemplo, el rango es 8. Esto nos da una idea de la amplitud de los datos, pero es muy sensible a los valores atípicos.

- **Rango Intercuartílico (IQR):**

- El IQR es la diferencia entre el percentil 75 y el percentil 25. En este ejemplo, el IQR es 4.5. El IQR es una medida de dispersión que es menos sensible a los valores atípicos que el rango. Proporciona una medida de la variabilidad de los datos en el rango intermedio.

Resumen

- **Percentil 25 (P25):** 2.25
- **Mediana (P50):** 4
- **Percentil 75 (P75):** 6.75
- **Rango:** 8
- **Rango Intercuartílico (IQR):** 4.5

Ejemplos en Python

Pandas

```
In [11]: import pandas as pd

# Crear el DataFrame
data = [3, 1, 5, 3, 6, 7, 2, 9]
df = pd.DataFrame(data, columns=['values'])

# Calcular los percentiles 25, 50 y 75
p25 = df['values'].quantile(0.25)
p50 = df['values'].quantile(0.50)
p75 = df['values'].quantile(0.75)

# Calcular el rango intercuartílico (IQR)
iqr = p75 - p25

print(f"Percentil 25 (P25): {p25}")
print(f"Mediana (P50): {p50}")
print(f"Percentil 75 (P75): {p75}")
print(f"Rango Intercuartílico (IQR): {iqr}")
```

Percentil 25 (P25): 2.75
 Mediana (P50): 4.0
 Percentil 75 (P75): 6.25
 Rango Intercuartílico (IQR): 3.5

SciPy

```
In [12]: import numpy as np
from scipy import stats

# Crear el array de datos
data = np.array([3, 1, 5, 3, 6, 7, 2, 9])

# Calcular los percentiles 25, 50 y 75
p25 = np.percentile(data, 25)
p50 = np.percentile(data, 50)
p75 = np.percentile(data, 75)

# Calcular el rango intercuartílico (IQR)
iqr = stats.iqr(data)

print(f"Percentil 25 (P25): {p25}")
print(f"Mediana (P50): {p50}")
print(f"Percentil 75 (P75): {p75}")
print(f"Rango Intercuartílico (IQR): {iqr}")
```

Percentil 25 (P25): 2.75
 Mediana (P50): 4.0
 Percentil 75 (P75): 6.25
 Rango Intercuartílico (IQR): 3.5

Statsmodels

```
In [13]: import numpy as np
import statsmodels.api as sm

# Crear el array de datos
data = np.array([3, 1, 5, 3, 6, 7, 2, 9])
```

```
# Calcular los percentiles 25, 50 y 75 usando numpy
p25, p50, p75 = np.percentile(data, [25, 50, 75])

# Calcular el rango intercuartílico (IQR) manualmente
iqr = p75 - p25

print(f"Percentil 25 (P25): {p25}")
print(f"Mediana (P50): {p50}")
print(f"Percentil 75 (P75): {p75}")
print(f"Rango Intercuartílico (IQR): {iqr}")
```

Percentil 25 (P25): 2.75
 Mediana (P50): 4.0
 Percentil 75 (P75): 6.25
 Rango Intercuartílico (IQR): 3.5

Características de los Percentiles

Independencia de la Distribución

Los percentiles son una medida no paramétrica, lo que significa que no dependen de la forma de la distribución de los datos. Pueden ser utilizados para describir cualquier conjunto de datos, independientemente de su distribución.

Robustez frente a Valores Atípicos

Los percentiles no son tan sensibles a los valores atípicos como otras medidas de dispersión, como la varianza y la desviación estándar. Esto los hace útiles en distribuciones con outliers o distribuciones sesgadas.

Descriptivos de Posición

Los percentiles proporcionan una descripción clara de la posición de un dato en el contexto del conjunto de datos. Por ejemplo, el percentil 50 (la mediana) indica el punto medio de los datos.

Aplicabilidad en Diferentes Distribuciones

Distribuciones Normales

En una distribución normal, los percentiles tienen interpretaciones específicas debido a la simetría y las propiedades de la campana de Gauss. Por ejemplo, el percentil 50 corresponde a la media, y los percentiles 25 y 75 corresponden a una desviación estándar a cada lado de la media.

Distribuciones Sesgadas

Los percentiles siguen siendo útiles en distribuciones sesgadas porque no dependen de la media y la desviación estándar. En una distribución sesgada, la mediana (percentil 50) es una mejor medida de tendencia central que la media.

Distribuciones Multimodales

En distribuciones con múltiples modos, los percentiles ayudan a describir la posición de los datos en relación con cada modo. Esto es útil para entender la estructura de la distribución.

Distribuciones Asimétricas

Los percentiles son adecuados para distribuciones asimétricas, ya que no se ven afectados por la falta de simetría. La mediana y el rango intercuartílico son especialmente útiles en estos casos.

Ejemplo Práctico

Imaginemos tres conjuntos de datos diferentes: uno con una distribución normal, otro con una distribución sesgada y otro con una distribución multimodal. En cada caso, los percentiles pueden proporcionar información valiosa sobre la distribución de los datos.

Distribución Normal

Datos:

$$x = [3, 5, 7, 9, 11, 13, 15, 17, 19]$$

Percentil 25:

$$P_{25} \approx 7$$

Mediana (P50):

$$P_{50} = 11$$

Percentil 75:

$$P_{75} \approx 15$$

Distribución Sesgada

Datos:

$$x = [1, 2, 3, 4, 5, 6, 7, 8, 20]$$

Percentil 25:

$$P_{25} \approx 3$$

Mediana (P50):

$$P_{50} = 5$$

Percentil 75:

$$P_{75} \approx 7$$

Distribución Multimodal

Datos:

$$x = [1, 1, 2, 2, 9, 9, 10, 10]$$

Percentil 25:

$$P_{25} \approx 2$$

Mediana (P50):

$$P_{50} = 5.5$$

Percentil 75:

$$P_{75} \approx 9.5$$

Interpretación Estadística

Distribución Normal

Los percentiles muestran la simetría de la distribución alrededor de la mediana.

```
In [6]: import numpy as np
import matplotlib.pyplot as plt

# Configuración de la semilla para reproducibilidad
np.random.seed(42)

# Generar datos para una distribución normal
data_normal = np.random.normal(loc=0, scale=1, size=1000)

# Calcular los percentiles 25, 50 y 75
Q1 = np.percentile(data_normal, 25)
Q2 = np.percentile(data_normal, 50) # Mediana
Q3 = np.percentile(data_normal, 75)

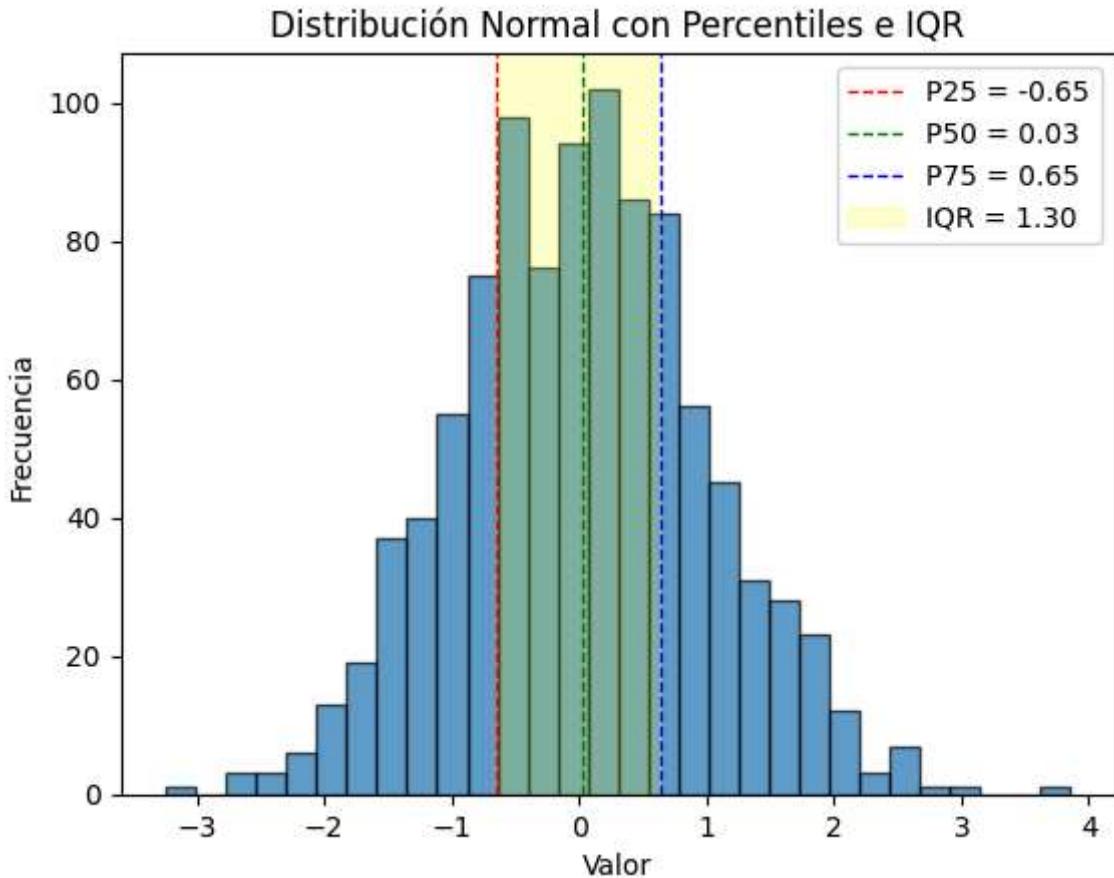
# Calcular el IQR
IQR = Q3 - Q1

# Crear el histograma
plt.hist(data_normal, bins=30, edgecolor='black', alpha=0.7)
plt.axvline(Q1, color='r', linestyle='dashed', linewidth=1, label=f'P25 = {Q1:.2f}')
plt.axvline(Q2, color='g', linestyle='dashed', linewidth=1, label=f'P50 = {Q2:.2f}')
plt.axvline(Q3, color='b', linestyle='dashed', linewidth=1, label=f'P75 = {Q3:.2f}')

# Mostrar el IQR
plt.axvspan(Q1, Q3, alpha=0.2, color='yellow', label=f'IQR = {IQR:.2f}')

# Añadir etiquetas y título
plt.xlabel('Valor')
plt.ylabel('Frecuencia')
plt.title('Distribución Normal con Percentiles e IQR')
plt.legend()
```

```
# Mostrar el gráfico
plt.show()
```



Distribución Sesgada

La mediana es una mejor medida de la tendencia central que la media, y el rango intercuartílico proporciona una buena medida de la dispersión.

```
In [5]: import numpy as np
import matplotlib.pyplot as plt

# Configuración de la semilla para reproducibilidad
np.random.seed(42)

# Generar datos para una distribución muy sesgada (exponencial)
data_skewed = np.random.exponential(scale=2, size=1000)

# Calcular los percentiles 25, 50 y 75
Q1 = np.percentile(data_skewed, 25)
Q2 = np.percentile(data_skewed, 50) # Mediana
Q3 = np.percentile(data_skewed, 75)

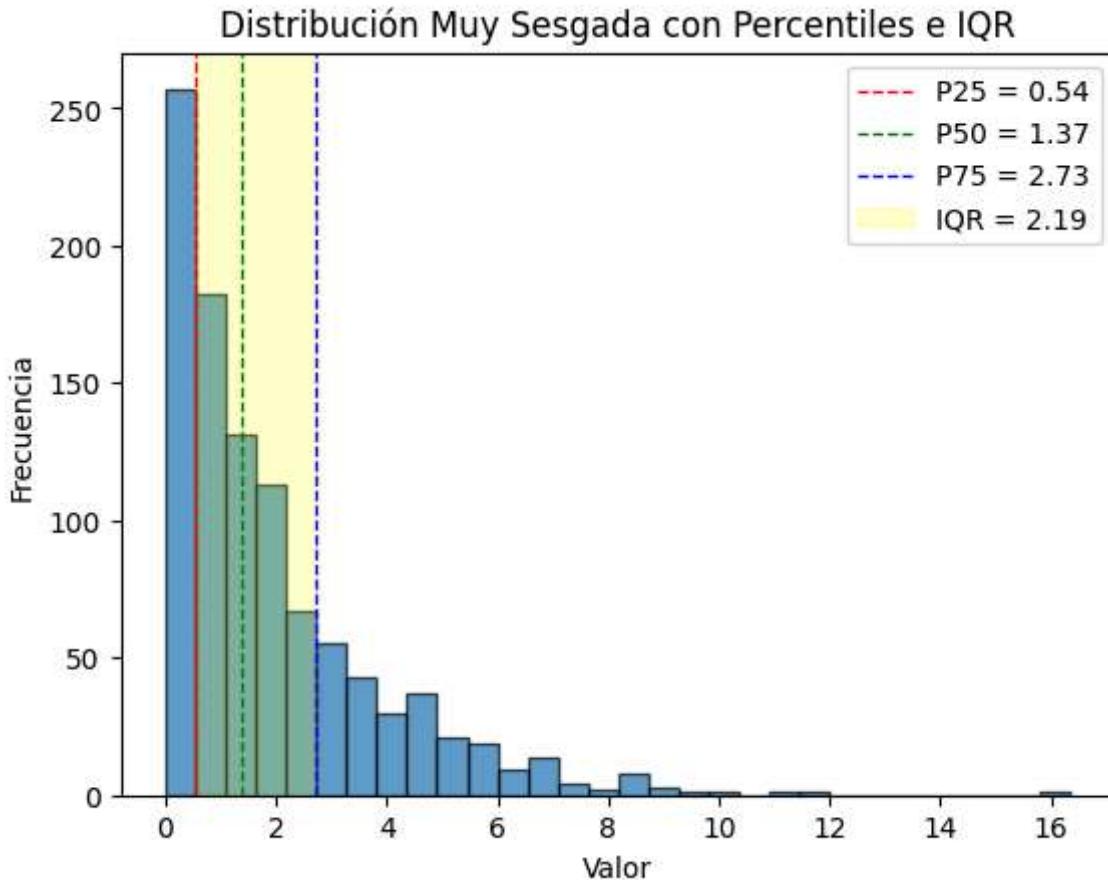
# Calcular el IQR
IQR = Q3 - Q1

# Crear el histograma
plt.hist(data_skewed, bins=30, edgecolor='black', alpha=0.7)
plt.axvline(Q1, color='r', linestyle='dashed', linewidth=1, label=f'P25 = {Q1:.2f}')
plt.axvline(Q2, color='g', linestyle='dashed', linewidth=1, label=f'P50 = {Q2:.2f}')
plt.axvline(Q3, color='b', linestyle='dashed', linewidth=1, label=f'P75 = {Q3:.2f}'
```

```
# Mostrar el IQR
plt.axvspan(Q1, Q3, alpha=0.2, color='yellow', label=f'IQR = {IQR:.2f}')

# Añadir etiquetas y título
plt.xlabel('Valor')
plt.ylabel('Frecuencia')
plt.title('Distribución Muy Sesgada con Percentiles e IQR')
plt.legend()

# Mostrar el gráfico
plt.show()
```



Distribución Multimodal

Los percentiles ayudan a entender cómo se distribuyen los datos en torno a los diferentes modos.

```
In [15]: import numpy as np
import matplotlib.pyplot as plt

# Configuración de la semilla para reproducibilidad
np.random.seed(42)

# Generar datos multimodales
data1 = np.random.normal(loc=0, scale=1, size=500)
data2 = np.random.normal(loc=5, scale=1, size=500)
data3 = np.random.normal(loc=10, scale=1, size=500)

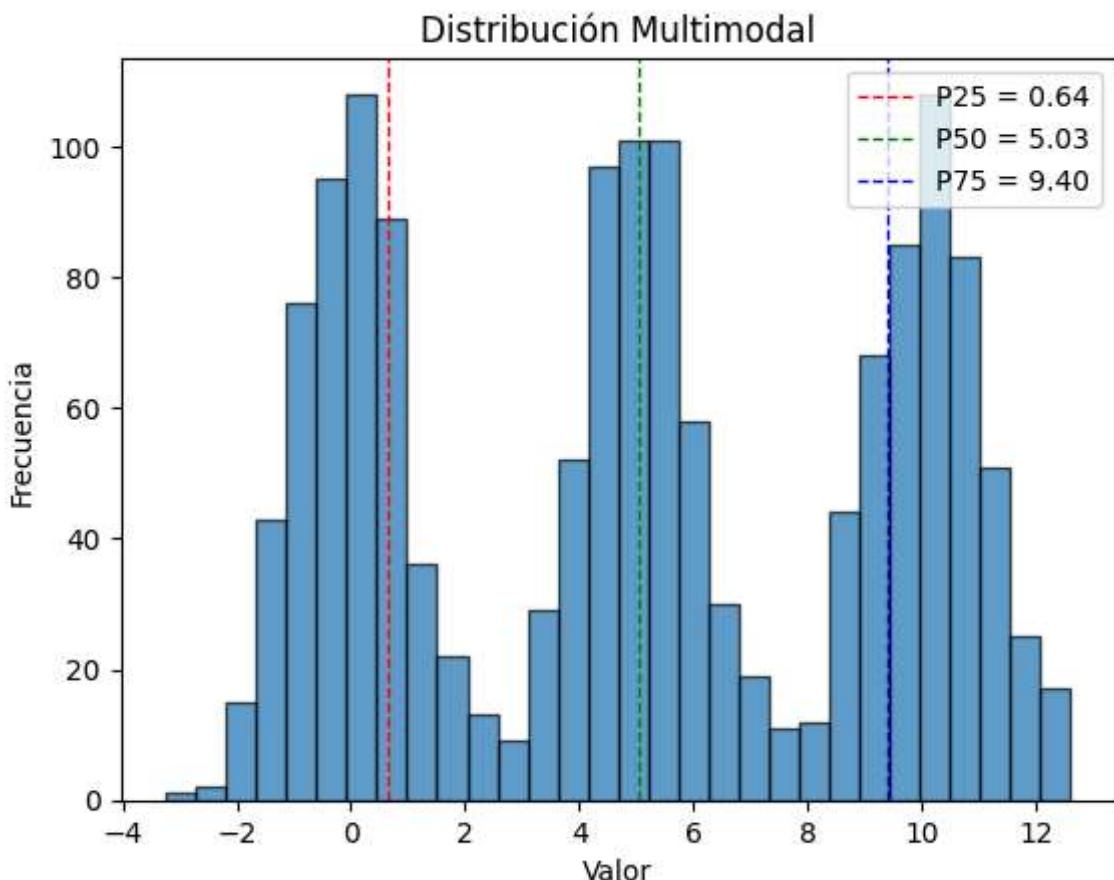
# Combinar los datos
data = np.concatenate([data1, data2, data3])
```

```
# Calcular los percentiles
p25 = np.percentile(data, 25)
p50 = np.percentile(data, 50)
p75 = np.percentile(data, 75)

# Crear el histograma
plt.hist(data, bins=30, edgecolor='black', alpha=0.7)
plt.axvline(p25, color='r', linestyle='dashed', linewidth=1, label=f'P25 = {p25}')
plt.axvline(p50, color='g', linestyle='dashed', linewidth=1, label=f'P50 = {p50}')
plt.axvline(p75, color='b', linestyle='dashed', linewidth=1, label=f'P75 = {p75}')

# Añadir etiquetas y título
plt.xlabel('Valor')
plt.ylabel('Frecuencia')
plt.title('Distribución Multimodal')
plt.legend()

# Mostrar el gráfico
plt.show()
```



Boxplots

In []:

In [7]:

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

# Configuración de la semilla para reproducibilidad
np.random.seed(42)

# Generar datos
```

```

# Distribución Normal
data_normal = np.random.normal(loc=0, scale=1, size=1000)

# Distribución Muy Sesgada (Exponencial)
data_skewed = np.random.exponential(scale=2, size=1000)

# Distribución Multimodal más pronunciada
data1 = np.random.normal(loc=-5, scale=1, size=500)
data2 = np.random.normal(loc=0, scale=1, size=500)
data3 = np.random.normal(loc=5, scale=1, size=500)
data_multimodal = np.concatenate([data1, data2, data3])

# Calcular percentiles y IQR para cada distribución
def calculate_iqr_percentiles(data):
    Q1 = np.percentile(data, 25)
    Q2 = np.percentile(data, 50) # Mediana
    Q3 = np.percentile(data, 75)
    IQR = Q3 - Q1
    return Q1, Q2, Q3, IQR

Q1_normal, Q2_normal, Q3_normal, IQR_normal = calculate_iqr_percentiles(data_normal)
Q1_skewed, Q2_skewed, Q3_skewed, IQR_skewed = calculate_iqr_percentiles(data_skewed)
Q1_multimodal, Q2_multimodal, Q3_multimodal, IQR_multimodal = calculate_iqr_percentiles(data_multimodal)

# Crear Los boxplots
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

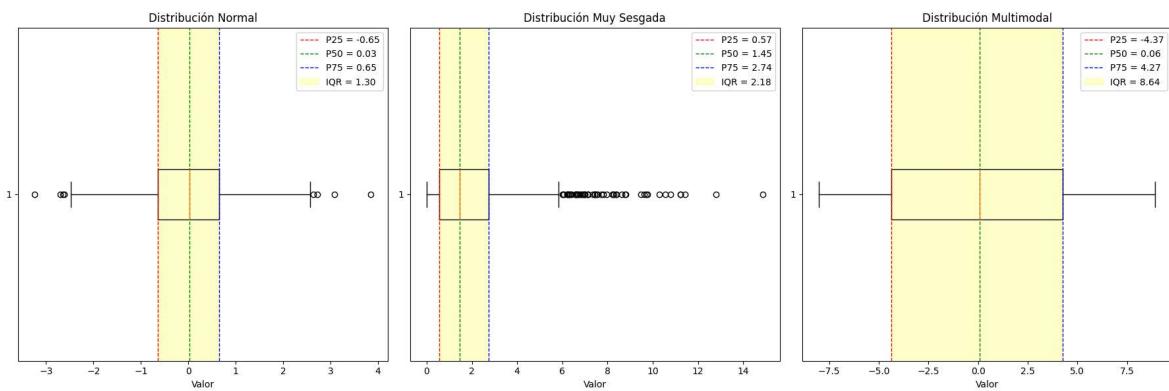
# Boxplot para la Distribución Normal
axs[0].boxplot(data_normal, vert=False)
axs[0].axvline(Q1_normal, color='r', linestyle='dashed', linewidth=1, label=f'P25')
axs[0].axvline(Q2_normal, color='g', linestyle='dashed', linewidth=1, label=f'P50')
axs[0].axvline(Q3_normal, color='b', linestyle='dashed', linewidth=1, label=f'P75')
axs[0].axvspan(Q1_normal, Q3_normal, alpha=0.2, color='yellow', label=f'IQR = {IQR_normal}')
axs[0].set_title('Distribución Normal')
axs[0].set_xlabel('Valor')
axs[0].legend()

# Boxplot para la Distribución Muy Sesgada
axs[1].boxplot(data_skewed, vert=False)
axs[1].axvline(Q1_skewed, color='r', linestyle='dashed', linewidth=1, label=f'P25')
axs[1].axvline(Q2_skewed, color='g', linestyle='dashed', linewidth=1, label=f'P50')
axs[1].axvline(Q3_skewed, color='b', linestyle='dashed', linewidth=1, label=f'P75')
axs[1].axvspan(Q1_skewed, Q3_skewed, alpha=0.2, color='yellow', label=f'IQR = {IQR_skewed}')
axs[1].set_title('Distribución Muy Sesgada')
axs[1].set_xlabel('Valor')
axs[1].legend()

# Boxplot para la Distribución Multimodal
axs[2].boxplot(data_multimodal, vert=False)
axs[2].axvline(Q1_multimodal, color='r', linestyle='dashed', linewidth=1, label=f'P25')
axs[2].axvline(Q2_multimodal, color='g', linestyle='dashed', linewidth=1, label=f'P50')
axs[2].axvline(Q3_multimodal, color='b', linestyle='dashed', linewidth=1, label=f'P75')
axs[2].axvspan(Q1_multimodal, Q3_multimodal, alpha=0.2, color='yellow', label=f'IQR = {IQR_multimodal}')
axs[2].set_title('Distribución Multimodal')
axs[2].set_xlabel('Valor')
axs[2].legend()

# Mostrar Los gráficos
plt.tight_layout()
plt.show()

```



Ejercicio 4

Con la misma dataset de los ejercicios anteriores

(`states.csv`), calcular los percentiles 25, 50 y 75. Calcular tambien el rango y el IQR para la variable población. Calcular/detectar outliers usando MAD y usando el criterio de Tukey (investigar) Realizar dos gráficos. El primero: Un histograma que muestre los percentiles y el IQR. El segundo gráfico un boxplot (diagrama de caja y bigote) que muestre los outliers.

```
In [21]: import pandas as pd
import matplotlib.pyplot as plt
state = pd.read_csv('state.csv')
```

Explorando la Distribución de los Datos

Percentiles y Diagramas de Caja

En la sección anterior, exploramos cómo los percentiles pueden usarse para medir la dispersión de los datos. Los percentiles también son valiosos para resumir la distribución completa. Es común reportar los cuartiles (percentiles 25, 50 y 75) y los deciles (percentiles 10, 20, ..., 90). Los percentiles son especialmente valiosos para resumir los extremos (la gama exterior) de la distribución. La cultura popular ha acuñado el término "one-percenters" para referirse a las personas en el percentil 99 superior de riqueza.

La Tabla 1-4 muestra algunos percentiles de la tasa de asesinatos por estado.

```
state['Murder.Rate'].quantile([0.05, 0.25, 0.5, 0.75, 0.95])
```

	5%	25%	50%	75%	95%
	1.60	2.42	4.00	5.55	6.51

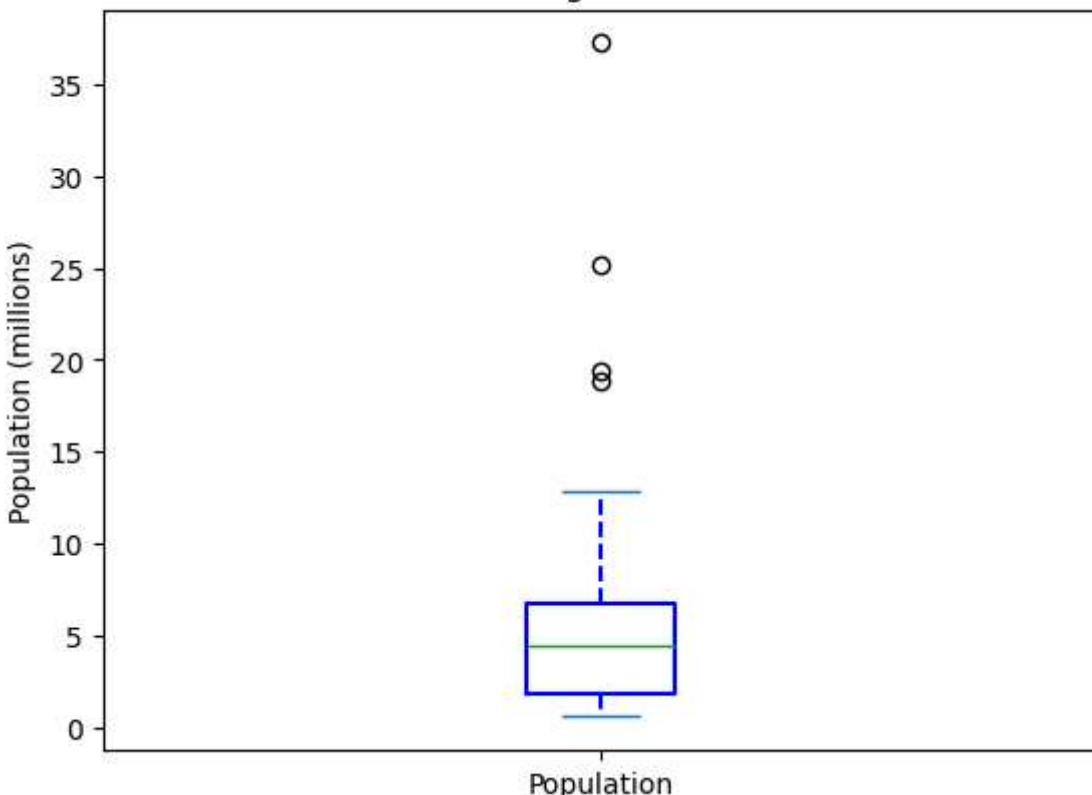
La mediana es de 4 asesinatos por cada 100,000 personas, aunque hay bastante variabilidad: el percentil 5 es solo 1.6 y el percentil 95 es 6.51.

Los diagramas de caja, introducidos por Tukey, se basan en percentiles y ofrecen una forma rápida de visualizar la distribución de los datos. La Figura 1-2 muestra un diagrama de caja de la población por estado.

```
In [22]: import pandas as pd
import matplotlib.pyplot as plt

boxprops = dict(linestyle='-', linewidth=1.5, color='blue')
whiskerprops = dict(linestyle='--', linewidth=1.5, color='blue')
ax = (state['Population'] / 1_000_000).plot.box(boxprops=boxprops, whiskerprops=whiskerprops)
ax.set_ylabel('Population (millions)')
ax.set_title('Fig. 1-2')
plt.show()
```

Fig. 1-2



A partir de este diagrama de caja, podemos ver inmediatamente que la mediana de la población estatal es de aproximadamente 5 millones, la mitad de los estados se encuentran entre aproximadamente 2 millones y 7 millones, y hay algunos outliers con alta población. La parte superior e inferior de la caja son los percentiles 75 y 25, respectivamente. La mediana se muestra mediante la línea horizontal en la caja. Las líneas discontinuas, denominadas "bigotes", se extienden desde la parte superior e inferior de la caja para indicar el rango de la mayor parte de los datos.

Tablas de Frecuencia e Histogramas

Una tabla de frecuencia de una variable divide el rango de la variable en intervalos igualmente espaciados y nos dice cuántos valores caen dentro de cada intervalo. La siguiente serie muestra los intervalos junto al conteo de estados.

```
In [19]: binnedPopulation = pd.cut(state['Population'], 10)
binnedPopulation.value_counts()
```

```
Out[19]: Population
(526935.67, 4232659.0]      24
(4232659.0, 7901692.0]      14
(7901692.0, 11570725.0]      6
(11570725.0, 15239758.0]     2
(15239758.0, 18908791.0]     1
(18908791.0, 22577824.0]     1
(22577824.0, 26246857.0]     1
(33584923.0, 37253956.0]     1
(26246857.0, 29915890.0]     0
(29915890.0, 33584923.0]     0
Name: count, dtype: int64
```

Ejercicio 5. Convertir la serie de arriba en un dataframe con indices enteros

```
In [24]: binnedPopulation = pd.cut(state['Population'], 20)
binnedPopulation.value_counts()
```

```
Out[24]: Population
(526935.67, 2398142.5]      15
(2398142.5, 4232659.0]      9
(4232659.0, 6067175.5]      9
(6067175.5, 7901692.0]      5
(7901692.0, 9736208.5]      4
(9736208.5, 11570725.0]     2
(11570725.0, 13405241.5]    2
(17074274.5, 18908791.0]    1
(18908791.0, 20743307.5]    1
(24412340.5, 26246857.0]    1
(35419439.5, 37253956.0]    1
(13405241.5, 15239758.0]    0
(20743307.5, 22577824.0]    0
(15239758.0, 17074274.5]    0
(26246857.0, 28081373.5]    0
(22577824.0, 24412340.5]    0
(28081373.5, 29915890.0]    0
(29915890.0, 31750406.5]    0
(31750406.5, 33584923.0]    0
(33584923.0, 35419439.5]    0
Name: count, dtype: int64
```

La Tabla 1-5 muestra una tabla de frecuencia de la población por estado:

BinNumber	BinRange	Count	States
1	(526935.67, 4232659.0]	24	Alaska, Arkansas, Connecticut, Delaware, Hawaii, Idaho, Iowa, Kansas, Maine, Mississippi, Montana, Nebraska, Nevada, New Hampshire, New Mexico, North Dakota, Rhode Island, South Dakota, Utah, Vermont, West Virginia, Wyoming, Oklahoma, Oregon
2	(4232659.0, 7901692.0]	14	Alabama, Arizona, Colorado, Indiana, Kentucky, Louisiana, Maryland, Minnesota, Missouri, South Carolina, Tennessee, Washington, Wisconsin, Massachusetts

BinNumber	BinRange	Count	States
3	(7901692.0, 11570725.0]	6	Georgia, Michigan, New Jersey, North Carolina, Ohio, Virginia
4	(11570725.0, 15239758.0]	2	Illinois, Pennsylvania
5	(15239758.0, 18908791.0]	1	Florida
6	(18908791.0, 22577824.0]	1	New York
7	(22577824.0, 26246857.0]	1	Texas
8	(33584923.0, 37253956.0]	1	California
9	(26246857.0, 29915890.0]	0	
10	(29915890.0, 33584923.0]	0	

Ejercicio 6. Utilice Pandas para obtener la tabla 1-5 dada arriba.

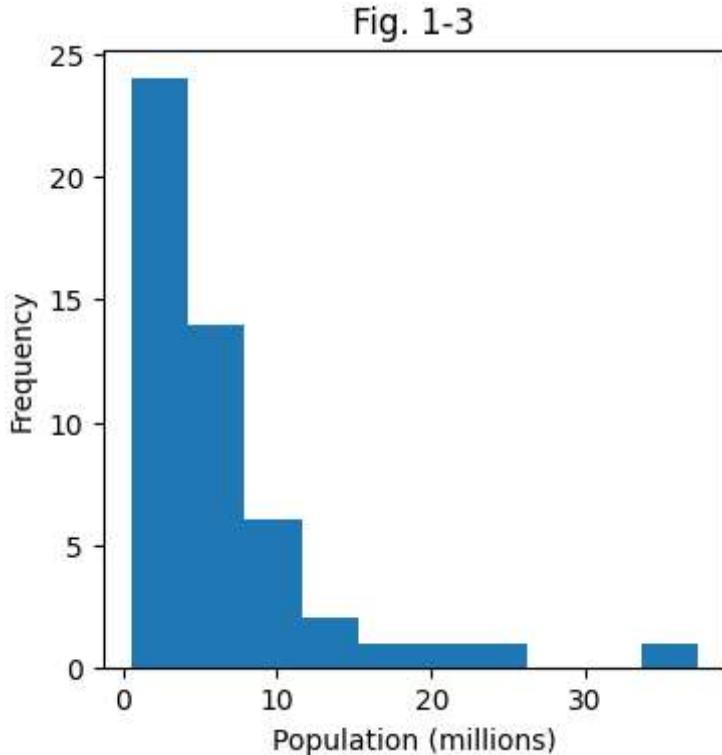
El estado menos poblado es Wyoming, con 563,626 personas, y el más poblado es California, con 37,253,956 personas. Esto nos da un rango de $37,253,956 - 563,626 = 36,690,330$, que debemos dividir en bins de tamaño igual—digamos 10 bins. Con 10 bins de tamaño igual, cada bin tendrá un ancho de 3,669,033, por lo que el primer bin abarcará de 563,626 a 4,232,658. Por el contrario, el bin superior, de 33,584,923 a 37,253,956, tiene solo un estado: California. Los dos bins inmediatamente debajo de California están vacíos, hasta que llegamos a Texas. Es importante incluir los bins vacíos; el hecho de que no haya valores en esos bins es información útil. También puede ser útil experimentar con diferentes tamaños de bins. Si son demasiado grandes, se pueden ocultar características importantes de la distribución. Si son demasiado pequeños, el resultado es demasiado granular y se pierde la capacidad de ver el panorama general.

Tanto las tablas de frecuencia como los percentiles resumen los datos creando bins. En general, los cuartiles y deciles tendrán el mismo conteo en cada bin (bins de conteo igual), pero los tamaños de los bins serán diferentes. La tabla de frecuencia, por el contrario, tendrá diferentes conteos en los bins (bins de tamaño igual), y los tamaños de los bins serán los mismos.

Un histograma es una forma de visualizar una tabla de frecuencia, con bins en el eje x y el conteo de datos en el eje y. Por ejemplo, el bin centrado en 10 millones ($1e+07$) va desde aproximadamente 8 millones hasta 12 millones, y hay seis estados en ese bin. pandas soporta histogramas para data frames con el método `DataFrame.plot.hist`. Use el argumento bins para definir el número de bins. Los diversos métodos de trazado devuelven un objeto de eje que permite ajustar aún más la visualización usando `Matplotlib`:

```
In [50]: ax = (state['Population'] / 1_000_000).plot.hist(figsize=(4, 4))
ax.set_xlabel('Population (millions)')
ax.set_title('Fig. 1-3')
```

Out[50]: Text(0.5, 1.0, 'Fig. 1-3')



El histograma que se muestra en la figura de arriba (Figura 1-3) se trazan de tal manera que:

- Se incluyen bins vacíos en el gráfico.
- Los bins tienen el mismo ancho.
- El número de bins (o, de manera equivalente, el tamaño del bin) depende del usuario.
- Las barras son contiguas—no hay espacio vacío entre las barras, a menos que haya un bin vacío.

Momentos Estadísticos

En la teoría estadística, la ubicación y la variabilidad se refieren a los primeros y segundos momentos de una distribución. Los terceros y cuartos momentos se denominan asimetría (`skewness`) y curtosis (`kurtosis`). La asimetría se refiere a si los datos están sesgados hacia valores más grandes o más pequeños, y la curtosis indica la propensión de los datos a tener valores extremos. Generalmente, no se utilizan métricas para medir la asimetría y la curtosis; en su lugar, se descubren a través de representaciones visuales como las Figuras 1-2 y 1-3.

Gráficas de Densidad y Estimaciones

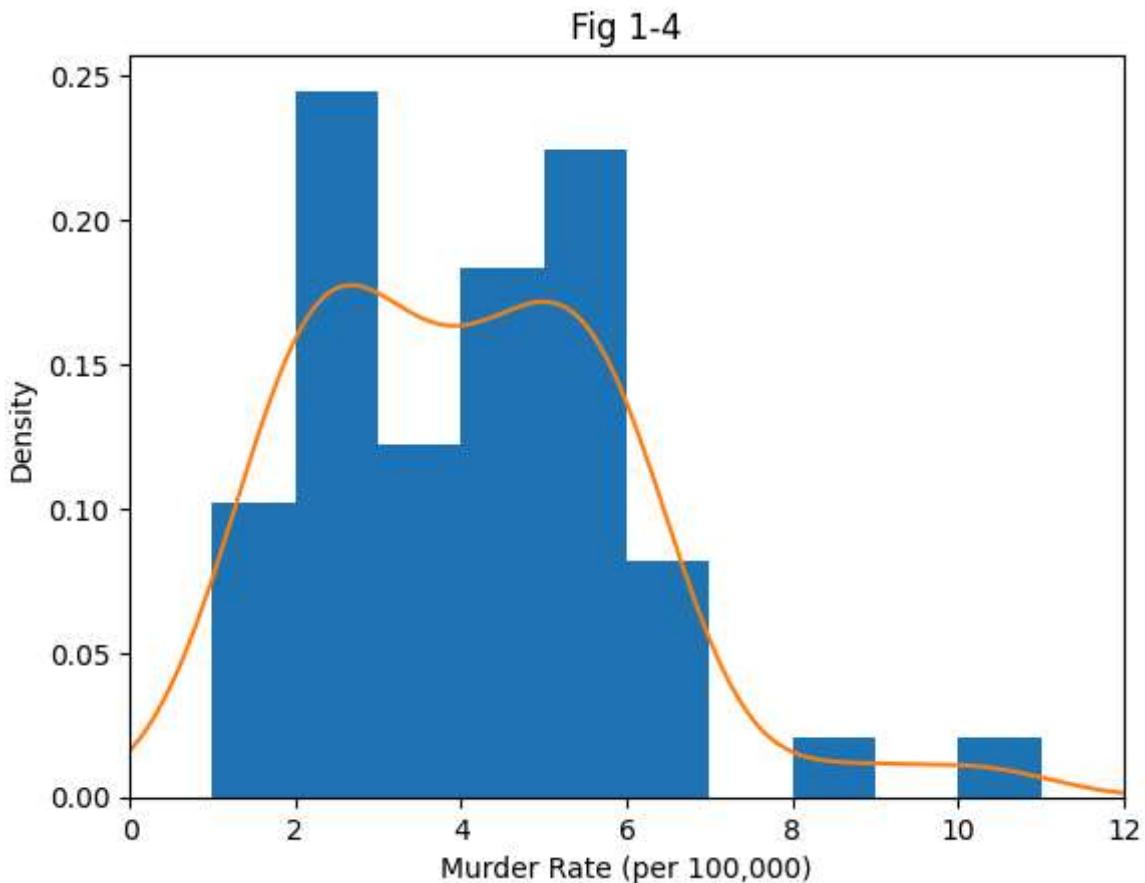
Relacionado con el histograma está la gráfica de densidad, que muestra la distribución de los valores de los datos como una línea continua. Una gráfica de densidad puede considerarse como un histograma suavizado, aunque típicamente se calcula directamente a partir de los datos a través de una estimación de densidad del kernel (KDE). La Figura 1-4 muestra una estimación de densidad superpuesta a un histograma.

Pandas proporciona el método `density` para crear una gráfica de densidad. Use el argumento `bw_method` para controlar la suavidad de la curva de densidad:

```
In [7]: ax = state['Murder.Rate'].plot.hist(density=True, xlim=[0,12], bins=range(1,12))
state['Murder.Rate'].plot.density(ax=ax)
# Las funciones de trazado a menudo toman un argumento opcional de eje (ax),
# Lo que hará que el gráfico se añada al mismo plot (histograma).

ax.set_xlabel('Murder Rate (per 100,000)')
ax.set_title('Fig 1-4')
```

Out[7]: Text(0.5, 1.0, 'Fig 1-4')



Histograma vs. Gráfica de Densidad

Histograma:

- Un histograma es una representación gráfica de la distribución de datos mediante barras.
- En el histograma, el eje `y` representa los conteos de frecuencia, es decir, cuántos datos caen en cada bin.

- Por ejemplo, si tienes un bin que cubre valores de 1 a 2 y hay 5 datos en ese rango, la altura de la barra para ese bin será 5.

Gráfica de Densidad:

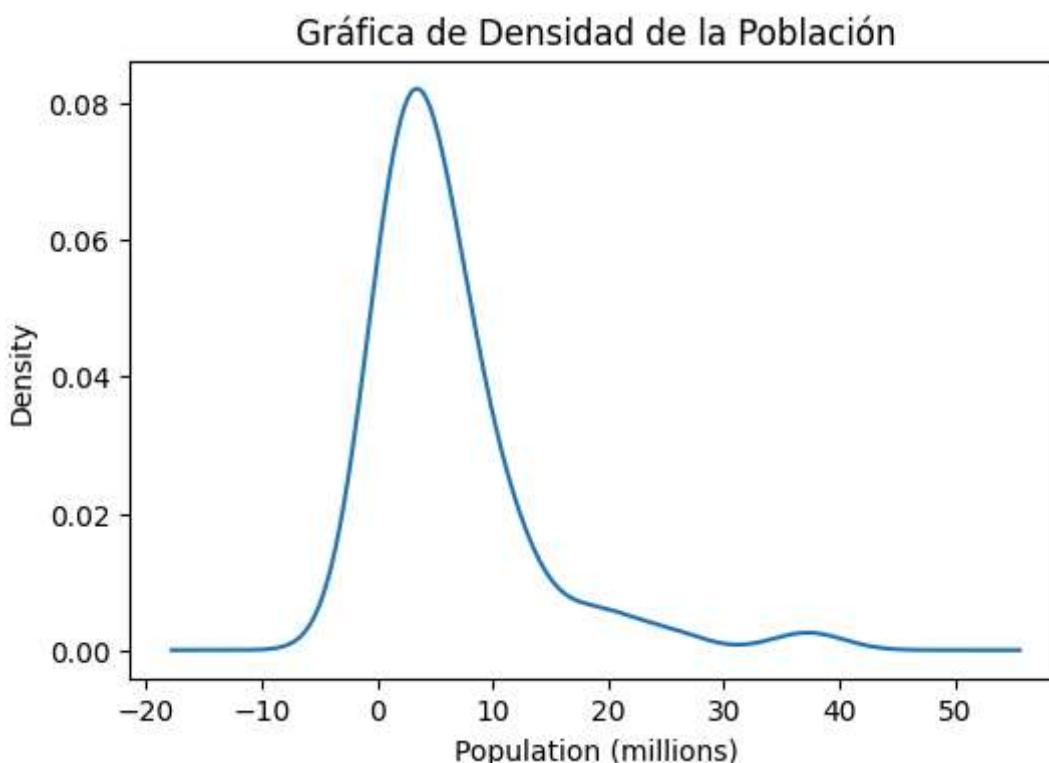
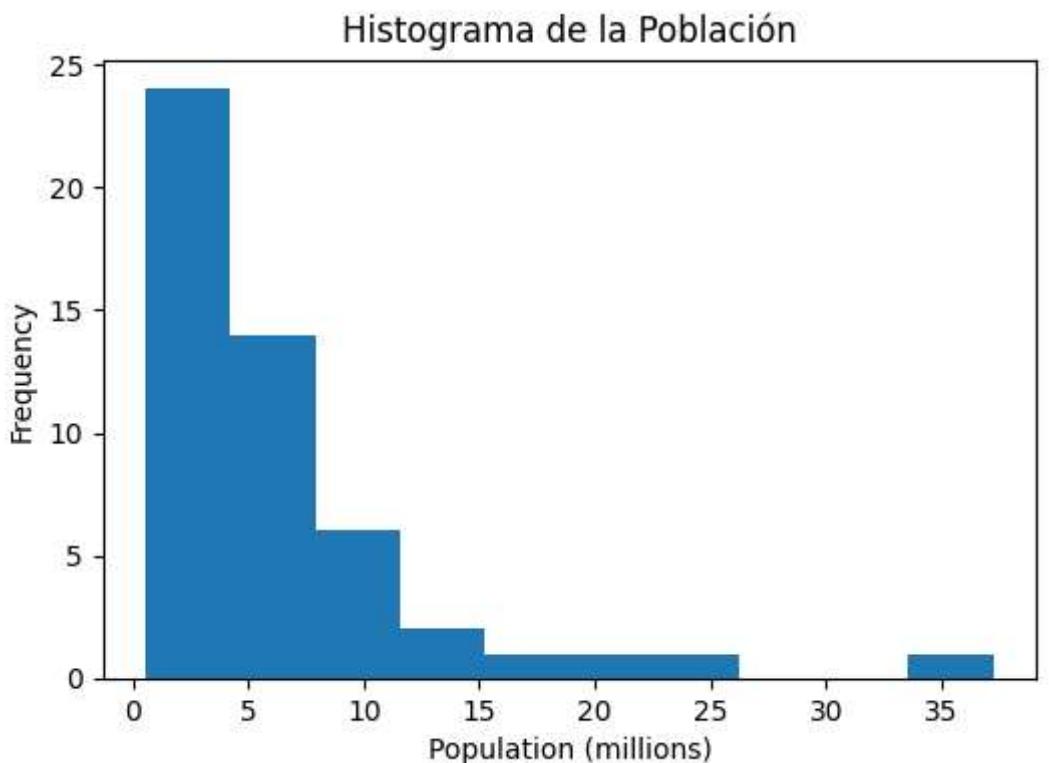
- Una gráfica de densidad es una versión suavizada del histograma, que muestra una línea continua en lugar de barras.
- En una gráfica de densidad, el eje y representa la densidad de probabilidad, no los conteos de frecuencia.
- La densidad de probabilidad es una medida que indica la probabilidad de que una variable aleatoria tome un valor dentro de un intervalo determinado.
- La clave aquí es que el área total bajo la curva de densidad es igual a 1. Esto significa que si sumas todas las áreas bajo la curva, obtendrás 1, lo que representa el 100% de la probabilidad.

```
In [25]: import pandas as pd
import matplotlib.pyplot as plt

# Cargar el archivo CSV
state = pd.read_csv('state.csv')

# Crear el histograma
ax_hist = (state['Population'] / 1_000_000).plot.hist(figsize=(6, 4))
ax_hist.set_xlabel('Population (millions)')
ax_hist.set_ylabel('Frequency')
ax_hist.set_title('Histograma de la Población')
plt.show()

# Crear la gráfica de densidad
ax_density = (state['Population'] / 1_000_000).plot.density(figsize=(6, 4))
ax_density.set_xlabel('Population (millions)')
ax_density.set_ylabel('Density')
ax_density.set_title('Gráfica de Densidad de la Población')
plt.show()
```



Resumen

Un histograma de frecuencias traza los conteos de frecuencia en el eje y y los valores de la variable en el eje x; proporciona una idea de la distribución de los datos de un vistazo.

- Una tabla de frecuencias es una versión tabular de los conteos de frecuencia que se encuentran en un histograma.
- Un diagrama de ca en cambio, a, con la parte superior e inferior de la caja en los percentiles 75 y 25, respectivamente, también proporciona una idea rápida de la

distribución de los datos; a menudo se utiliza en pantallas lado a lado para comparar distribuciones.

- Una gráfica de densidad es una versión suavizada de un histograma; requiere una función para estimar una gráfica basada en los datos (por supuesto, son posibles múltiples estimaciones).

Representación de Datos Binarios y Categóricos

Para los datos categóricos, las proporciones o porcentajes simples cuentan la historia de los datos.

Términos Clave para Explorar Datos Categóricos

Moda

La categoría o valor que ocurre con mayor frecuencia en un conjunto de datos.

Valor Esperado

Cuando las categorías pueden asociarse con un valor numérico, esto proporciona un valor promedio basado en la probabilidad de ocurrencia de una categoría.

Gráficos de Barras

La frecuencia o proporción de cada categoría trazada como barras.

Gráficos de Pastel

La frecuencia o proporción de cada categoría trazada como cuñas en un pastel.

Obtener un resumen de una variable binaria o una variable categórica con algunas categorías es una cuestión bastante sencilla: solo necesitamos averiguar la proporción de 1s, o las proporciones de las categorías importantes. Por ejemplo, la Tabla 1-6 muestra el porcentaje de vuelos retrasados por la causa del retraso en el Aeropuerto de Dallas/Fort Worth desde 2010. Los retrasos se categorizan como debidos a factores bajo el control del transportista, retrasos del sistema de control del tráfico aéreo (ATC), clima, seguridad, o una aeronave entrante tardía.

Tabla 1-6. Porcentaje de retrasos por causa en el Aeropuerto de Dallas/Fort Worth

	Carrier	ATC	Weather	Security	Inbound
	23.02%	30.40%	4.03%	0.12%	42.43%

```
In [30]: dfw = pd.read_csv('dfw_airline.csv')
print(100 * dfw / dfw.values.sum())
```

	Carrier	ATC	Weather	Security	Inbound
0	23.022989	30.400781	4.025214	0.122937	42.4428079

- **Carrier (Transportista):**

- Esta columna muestra el porcentaje de retrasos causados por factores bajo el control de la aerolínea o transportista. Esto puede incluir problemas operacionales, personal, mantenimiento, etc. En este caso, el 23.02% de los retrasos son atribuibles al transportista.

- **ATC (Control de Tráfico Aéreo):**

- Esta columna representa el porcentaje de retrasos causados por el sistema de control de tráfico aéreo. Estos retrasos pueden ocurrir debido a congestión en el espacio aéreo, problemas de gestión del tráfico aéreo, etc. Aquí, el 30.40% de los retrasos son debidos a ATC.

- **Weather (Clima):**

- Esta columna indica el porcentaje de retrasos causados por condiciones meteorológicas adversas, como tormentas, niebla, nieve, etc. En este caso, el 4.03% de los retrasos son atribuibles al clima.

- **Security (Seguridad):**

- Esta columna muestra el porcentaje de retrasos causados por medidas de seguridad, como inspecciones adicionales, procedimientos de seguridad, amenazas de seguridad, etc. En este caso, el 0.12% de los retrasos son debidos a problemas de seguridad.

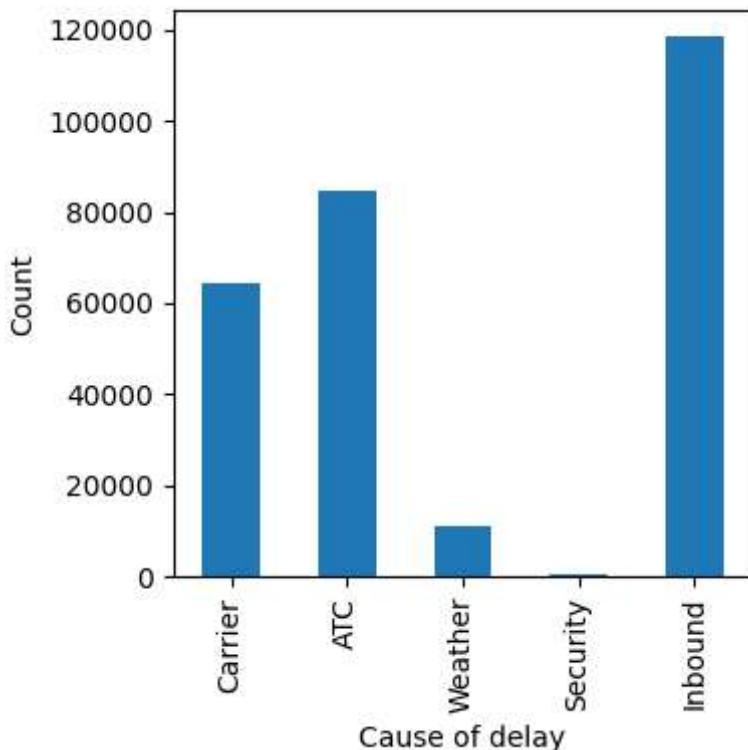
- **Inbound (Aeronave Entrante):**

- Esta columna representa el porcentaje de retrasos causados por la llegada tardía de la aeronave entrante. Esto puede suceder si un vuelo anterior llega tarde, afectando la salida programada del siguiente vuelo. Aquí, el 42.43% de los retrasos son debidos a aeronaves entrantes tardías.

Los gráficos de barras (bar charts), que se ven a menudo en la prensa popular, son una herramienta visual común para mostrar una única variable categórica. Las categorías se enumeran en el eje x y las frecuencias o proporciones en el eje y. Pandas también admite gráficos de barras para data frames:

```
In [49]: ax = dfw.transpose().plot.bar(figsize=(4, 4), legend=False)
ax.set_xlabel('Cause of delay')
ax.set_ylabel('Count')

plt.tight_layout()
plt.show()
ax.set_title('Fig 1-5')
```



Out[49]: Text(0.5, 1.0, 'Fig 1-5')

Tenga en cuenta que un gráfico de barras se asemeja a un histograma; en un gráfico de barras, el eje x representa diferentes categorías de una variable de factor, mientras que en un histograma, el eje x representa valores de una sola variable en una escala numérica. En un histograma, las barras generalmente se muestran tocándose entre sí, con espacios que indican valores que no ocurrieron en los datos. En un gráfico de barras, las barras se muestran separadas entre sí.

Moda

La moda es el valor—o valores en caso de empate—que aparece con mayor frecuencia en los datos. Por ejemplo, la moda de la causa de retraso en el aeropuerto de Dallas/Fort Worth es "Inbound". Como otro ejemplo, en la mayoría de las partes de los Estados Unidos, la moda para la preferencia religiosa sería cristiana. La moda es una estadística resumida simple para datos categóricos y, en general, no se utiliza para datos numéricos.

Valor Esperado

Un tipo especial de datos categóricos es aquel en el que las categorías representan o pueden asignarse a valores discretos en la misma escala. Un comercializador de una nueva tecnología en la nube, por ejemplo, ofrece dos niveles de servicio, uno con un precio de 300\$/mes y otro con un precio de 50\$/mes.

El comercializador ofrece seminarios web gratuitos para generar clientes potenciales, y la empresa calcula que el 5% de los asistentes se suscribirá al servicio de 300 dólares, el 15% se suscribirá al servicio de 50 dólares y el 80% no se suscribirá a nada. Estos datos pueden resumirse, para fines financieros, en un único "valor esperado", que es una forma de media ponderada, en la que los pesos son probabilidades.

Valor Esperado

El valor esperado se calcula de la siguiente manera:

1. Multiplicar cada resultado por su probabilidad de ocurrencia.
2. Sumar estos valores.

En el ejemplo del servicio en la nube, el valor esperado de un asistente a un seminario web es de \$22.50 por mes, calculado de la siguiente manera:

$$EV = 0.05 \times 300 + 0.15 \times 50 + 0.80 \times 0 = 22.5$$

El valor esperado es realmente una forma de media ponderada: añade las ideas de expectativas futuras y pesos de probabilidad, a menudo basados en el juicio subjetivo. El valor esperado es un concepto fundamental en la valoración empresarial y la planificación de inversiones, por ejemplo, el valor esperado de cinco años de beneficios de una nueva adquisición o los ahorros de costos esperados de un nuevo software de gestión de pacientes en una clínica.

El valor esperado es una medida fundamental en probabilidad y estadísticas que se utiliza para predecir el resultado promedio de un conjunto de eventos probabilísticos. Se calcula ponderando cada posible resultado por su probabilidad de ocurrencia y sumando estos valores ponderados.

Pasos para Calcular el Valor Esperado

1. Identificar los Resultados y sus Probabilidades:

- Cada posible resultado de un experimento o evento debe tener una probabilidad asociada.

2. Multiplicar Cada Resultado por su Probabilidad de Ocurrencia:

- Para cada resultado, multiplicamos su valor por la probabilidad de que ocurra.

3. Sumar los Valores Ponderados:

- Sumamos todos los valores ponderados obtenidos en el paso anterior para obtener el valor esperado.

Ejemplo Detallado: Juego de Lotería

Supongamos que estamos analizando un simple juego de lotería. El costo de un boleto es \$10, y hay tres posibles premios:

- Un premio de \$100 con una probabilidad de 1%.
- Un premio de \$50 con una probabilidad de 2%.
- Un premio de \$20 con una probabilidad de 5%.
- No ganar nada con una probabilidad de 92%.

Queremos calcular el valor esperado de las ganancias por boleto.

Paso 1: Identificar los Resultados y sus Probabilidades

- Ganar \$100: Probabilidad = 0.01
- Ganar \$50: Probabilidad = 0.02
- Ganar \$20: Probabilidad = 0.05
- Ganar \$0: Probabilidad = 0.92

Paso 2: Multiplicar Cada Resultado por su Probabilidad de Ocurrencia

- Ganancia ponderada por ganar \$100:

$$100 \times 0.01 = 1$$

- Ganancia ponderada por ganar \$50:

$$50 \times 0.02 = 1$$

- Ganancia ponderada por ganar \$20:

$$20 \times 0.05 = 1$$

- Ganancia ponderada por ganar \$0:

$$0 \times 0.92 = 0$$

Paso 3: Sumar los Valores Ponderados

Sumamos todas las ganancias ponderadas para obtener el valor esperado:

$$EV = 1 + 1 + 1 + 0 = 3$$

Finalmente, para obtener la ganancia neta esperada, restamos el costo del boleto (\$10):

$$EV_{\text{neto}} = 3 - 10 = -7$$

Interpretación del Resultado

El valor esperado neto de **-7 dólares** significa que, en promedio, se espera perder \$7 por cada boleto comprado en este juego de lotería. Aunque es posible ganar dinero, las probabilidades están estructuradas de tal manera que las pérdidas promedio superan las ganancias promedio.

Aplicación del Valor Esperado en Negocios

El valor esperado se utiliza ampliamente en negocios y finanzas para la toma de decisiones. Algunos ejemplos incluyen:

- **Valoración de Proyectos:** Estimar los ingresos esperados de un nuevo proyecto o inversión.
- **Gestión de Riesgos:** Calcular el valor esperado de pérdidas en escenarios de riesgo.

- **Planificación de Presupuestos:** Prever los costos y beneficios futuros esperados para la planificación financiera.

El valor esperado es una herramienta poderosa que ayuda a tomar decisiones informadas basadas en probabilidades y resultados ponderados.

Correlación

El análisis exploratorio de dat EDA os en muchos proyectos de modelado (ya sea en ciencia de datos o en investigación) implica examinar la correlación entre predictor, ees decir correlación y entre predictores y una variable objetivo. Se dice que las variables X e Y (cada una con datos medidos) están positivamente correlacionadas si los valores altos de X van acompañados de valores altos de Y, y los valores bajos de X van acompañados de valores bajos de Y. Si los valores altos de X van acompañados de valores bajos de Y, y viceversa, las variables están negativamente correlacionada.

Términos Clave para la Correlación

Coeficiente de Correlación Una métrica que mide la magnitud en la que las variables numéricas están asociadas entre sí (rango de -1 a $+1$).

El coeficiente de correlación es una métrica que cuantifica la relación entre dos variables numéricas, se usa para determinar tanto la fuerza como la dirección de la relación lineal entre las variables.

Valores del Coeficiente de Correlación

- **+1:** Indica una correlación perfectamente positiva. A medida que una variable aumenta, la otra también aumenta en proporción constante. Todos los puntos en un gráfico de dispersión caerían exactamente en una línea recta ascendente.
- **0:** Indica que no hay correlación lineal entre las variables. Los puntos en un gráfico de dispersión estarían dispersos de manera que no formarían ningún patrón lineal.
- **-1:** Indica una correlación perfectamente negativa. A medida que una variable aumenta, la otra disminuye en proporción constante. Todos los puntos en un gráfico de dispersión caerían exactamente en una línea recta descendente.

Interpretación de la Magnitud del Coeficiente de Correlación

- **0.9 a 1.0 (o -0.9 a -1.0):** Correlación muy fuerte
- **0.7 a 0.9 (o -0.7 a -0.9):** Correlación fuerte
- **0.5 a 0.7 (o -0.5 a -0.7):** Correlación moderada
- **0.3 a 0.5 (o -0.3 a -0.5):** Correlación débil
- **0.0 a 0.3 (o 0.0 a -0.3):** Correlación muy débil o inexistente

Fórmula del Coeficiente de Correlación de Pearson

El coeficiente de correlación más comúnmente usado es el coeficiente de correlación de Pearson, que se calcula usando la siguiente fórmula:

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

donde:

- (r) es el coeficiente de correlación de Pearson.
- (x_i) y (y_i) son los valores individuales de las variables (X) e (Y).
- (\bar{x}) y (\bar{y}) son las medias de las variables (X) e (Y), respectivamente.

Matriz de Correlación Una tabla donde las variables se muestran tanto en las filas como en las columnas, y los valores de las celdas son las correlaciones entre las variables.

La Tabla 1-7, llamada matriz de correlación, muestra la correlación entre los rendimientos diarios de las acciones de telecomunicaciones desde julio de 2012 hasta junio de 2015. En la tabla, se puede ver que Verizon (VZ) y ATT (T) tienen la correlación más alta. Level 3 (LVLT), que es una empresa de infraestructura, tiene la correlación más baja con las demás. Nota la diagonal de 1s (la correlación de una acción consigo misma es 1) y la redundancia de la información por encima y por debajo de la diagonal.

Tabla 1-7. Correlación entre los rendimientos de las acciones de telecomunicaciones

	T	CTL	FTR	VZ	LVLT
T	1.000	0.475	0.328	0.678	0.279
CTL	0.475	1.000	0.420	0.417	0.287
FTR	0.328	0.420	1.000	0.287	0.260
VZ	0.678	0.417	0.287	1.000	0.242
LVLT	0.279	0.287	0.260	0.242	1.000

La Tabla 1-7 muestra la correlación entre los rendimientos diarios de varias acciones de telecomunicaciones. A continuación se explican las siglas de las empresas que aparecen en la tabla:

- **T:** AT&T Inc.
 - AT&T es una de las compañías de telecomunicaciones más grandes del mundo, ofreciendo servicios de telefonía fija y móvil, así como servicios de internet y televisión.
- **CTL:** CenturyLink, Inc.
 - CenturyLink, ahora conocido como Lumen Technologies, es una empresa de telecomunicaciones que proporciona servicios de voz, datos y video en todo el mundo.
- **FTR:** Frontier Communications Corporation
 - Frontier Communications es una empresa de telecomunicaciones que ofrece servicios de voz, banda ancha y video, principalmente en áreas rurales y

suburbanas de los Estados Unidos.

- **VZ:** Verizon Communications Inc.
 - Verizon es una empresa de telecomunicaciones que proporciona servicios de comunicación y tecnología, incluyendo servicios de telefonía móvil y fija, internet y televisión.
- **LVLT:** Level 3 Communications, Inc.
 - Level 3 Communications, ahora parte de CenturyLink (Lumen Technologies), es una empresa de infraestructura que ofrece servicios de red y datos a empresas, gobiernos y otros operadores de telecomunicaciones.

Estas siglas representan algunas de las principales compañías de telecomunicaciones, y la tabla muestra cómo los rendimientos diarios de sus acciones están correlacionados entre sí.

Una tabla de correlaciones como la Tabla 1-7 se traza comúnmente para mostrar visualmente la relación entre múltiples variables. La Figura 1-6 muestra la correlación entre los rendimientos diarios de los principales fondos cotizados en bolsa (ETFs). El siguiente código demuestra esto utilizando el paquete `seaborn.heatmap`.

```
In [34]: import seaborn as sns
sp500_sym = pd.read_csv('sp500_sectors.csv')
sp500_px = pd.read_csv('sp500_data.csv.gz', index_col=0)
```

```
In [35]: # Table 1-7
# Determine telecommunications symbols
telecomSymbols = sp500_sym[sp500_sym['sector'] == 'telecommunications_services']

# Filter data for dates July 2012 through June 2015
telecom = sp500_px.loc[sp500_px.index >= '2012-07-01', telecomSymbols]
telecom.corr()
print(telecom)
```

	T	CTL	FTR	VZ	LVLT
2012-07-02	0.422496	0.140847	0.070879	0.554180	-0.519998
2012-07-03	-0.177448	0.066280	0.070879	-0.025976	-0.049999
2012-07-05	-0.160548	-0.132563	0.055128	-0.051956	-0.180000
2012-07-06	0.342205	0.132563	0.007875	0.140106	-0.359999
2012-07-09	0.136883	0.124279	-0.023626	0.253943	0.180000
...
2015-06-25	0.049342	-1.600000	-0.040000	-0.187790	-0.330002
2015-06-26	-0.256586	0.039999	-0.070000	0.029650	-0.739998
2015-06-29	-0.098685	-0.559999	-0.060000	-0.504063	-1.360000
2015-06-30	-0.503298	-0.420000	-0.070000	-0.523829	0.199997
2015-07-01	-0.019737	0.080000	-0.050000	0.355811	0.139999

[754 rows x 5 columns]

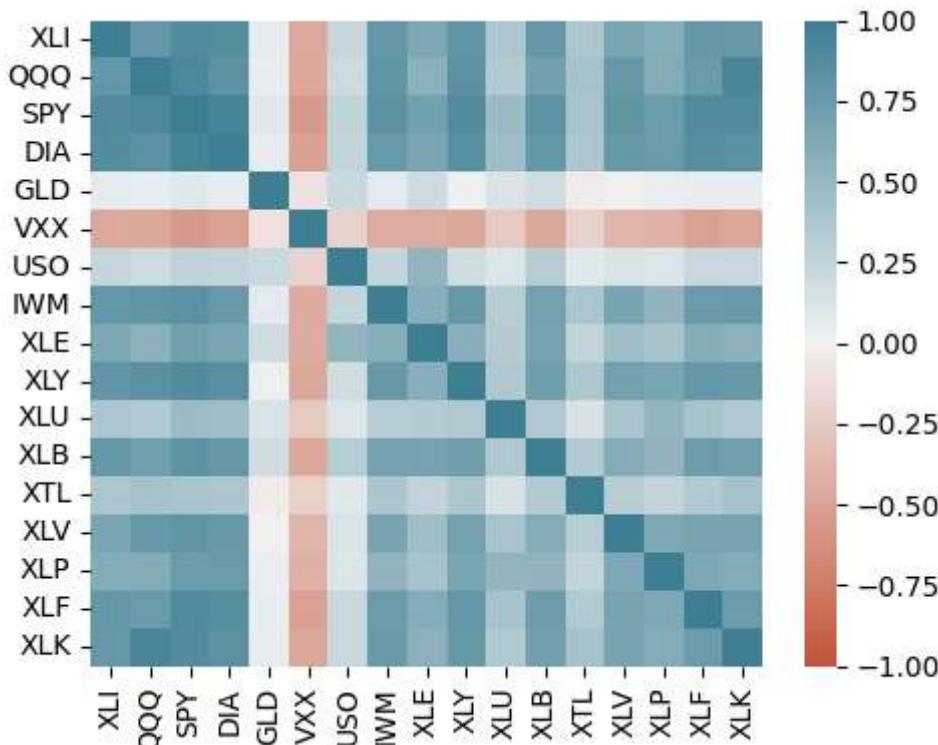
A continuación, nos enfocamos en los fondos negociados en las principales bolsas (sector == 'etf').

```
In [40]: etfs = sp500_px.loc[sp500_px.index > '2012-07-01', sp500_sym[sp500_sym['sector'] == 'etf']]
print(etfs.head())
```

	XLI	QQQ	SPY	DIA	GLD	VXX	USO	\
2012-07-02	-0.376098	0.096313	0.028223	-0.242796	0.419998	-10.40	0.000000	
2012-07-03	0.376099	0.481576	0.874936	0.728405	0.490006	-3.52	0.250000	
2012-07-05	0.150440	0.096313	-0.103487	0.149420	0.239991	6.56	-0.070000	
2012-07-06	-0.141040	-0.491201	0.018819	-0.205449	-0.519989	-8.80	-0.180000	
2012-07-09	0.244465	-0.048160	-0.056445	-0.168094	0.429992	-0.48	0.459999	
	IWM	XLE	XLY	XLU	XLB	XTL	XLK	\
2012-07-02	0.534641	0.028186	0.095759	0.098311	-0.093713	0.019076		
2012-07-03	0.926067	0.995942	0.000000	-0.044686	0.337373	0.000000		
2012-07-05	-0.171848	-0.460387	0.306431	-0.151938	0.103086	0.019072		
2012-07-06	-0.229128	0.206706	0.153214	0.080437	0.018744	-0.429213		
2012-07-09	-0.190939	-0.234892	-0.201098	-0.035751	-0.168687	0.000000		
	XLV	XLP	XLF	XLK				
2012-07-02	-0.009529	0.313499	0.018999	0.075668				
2012-07-03	0.000000	0.129087	0.104492	0.236462				
2012-07-05	-0.142955	-0.073766	-0.142490	0.066211				
2012-07-06	-0.095304	0.119865	0.066495	-0.227003				
2012-07-09	0.352630	-0.064548	0.018999	0.009457				

```
In [41]: fig, ax = plt.subplots(figsize=(5, 4))
ax = sns.heatmap(etfs.corr(), vmin=-1, vmax=1,
                  cmap=sns.diverging_palette(20, 220, as_cmap=True),
                  ax=ax)

plt.tight_layout()
plt.show()
```



Los ETFs para el S&P 500 (SPY) y el índice Dow Jones (DIA) tienen una alta correlación. Del mismo modo, el QQQ y el XLK, compuestos principalmente por empresas tecnológicas, están positivamente correlacionados. Los ETFs defensivos, como los que rastrean los precios del oro (GLD), los precios del petróleo (USO) o la volatilidad del mercado (VXX), tienden a estar débilmente o negativamente correlacionados con los otros ETFs.

In [42]:

```

import numpy as np
from matplotlib.collections import EllipseCollection
from matplotlib.colors import Normalize

def plot_corr_ellipses(data, figsize=None, **kwargs):
    ''' https://stackoverflow.com/a/34558488 '''
    M = np.array(data)
    if not M.ndim == 2:
        raise ValueError('data must be a 2D array')
    fig, ax = plt.subplots(1, 1, figsize=figsize, subplot_kw={'aspect':'equal'})
    ax.set_xlim(-0.5, M.shape[1] - 0.5)
    ax.set_ylim(-0.5, M.shape[0] - 0.5)
    ax.invert_yaxis()

    # xy locations of each ellipse center
    xy = np.indices(M.shape)[::-1].reshape(2, -1).T

    # set the relative sizes of the major/minor axes according to the strength of
    # the positive/negative correlation
    w = np.ones_like(M).ravel() + 0.01
    h = 1 - np.abs(M).ravel() - 0.01
    a = 45 * np.sign(M).ravel()

    ec = EllipseCollection(widths=w, heights=h, angles=a, units='x', offsets=xy,
                           norm=Normalize(vmin=-1, vmax=1),
                           transOffset=ax.transData, array=M.ravel(), **kwargs)
    ax.add_collection(ec)

    # if data is a DataFrame, use the row/column names as tick labels
    if isinstance(data, pd.DataFrame):
        ax.set_xticks(np.arange(M.shape[1]))
        ax.set_xticklabels(data.columns, rotation=90)
        ax.set_yticks(np.arange(M.shape[0]))
        ax.set_yticklabels(data.index)

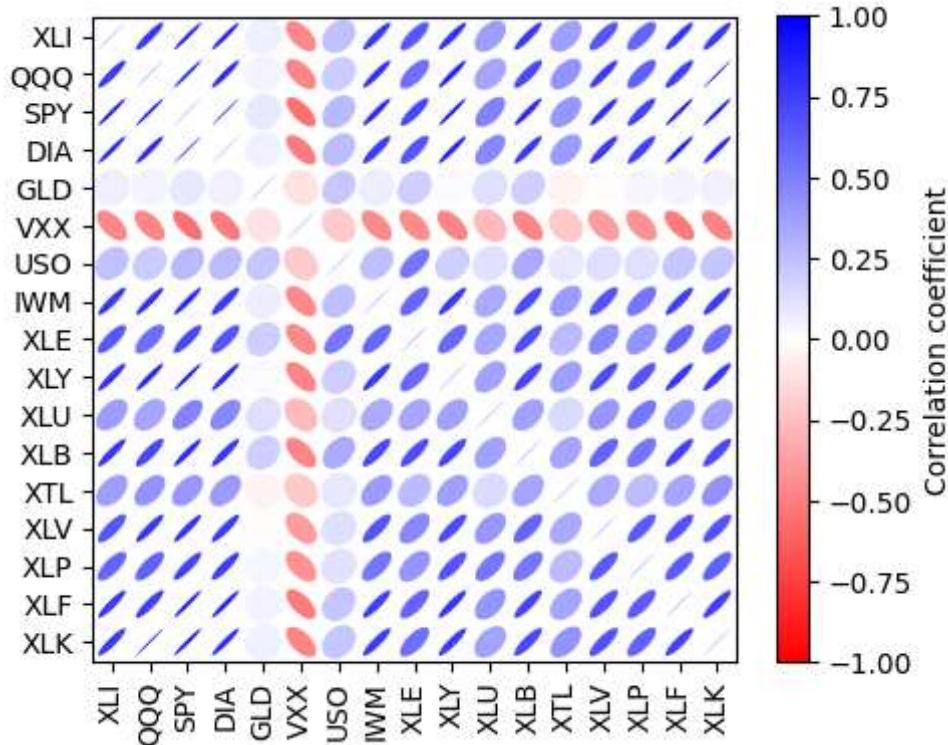
    return ec, ax

m, ax = plot_corr_ellipses(etfs.corr(), figsize=(5, 4), cmap='bwr_r')
cb = fig.colorbar(m, ax=ax)
cb.set_label('Correlation coefficient')

plt.tight_layout()
plt.show()

```

C:\Users\juanj\AppData\Local\Temp\ipykernel_29716\1398409197.py:39: UserWarning:
 Adding colorbar to a different Figure <Figure size 500x400 with 2 Axes> than <Figure size 500x400 with 2 Axes> which fig.colorbar is called on.
 cb = fig.colorbar(m, ax=ax)



La orientación de la elipse indica si dos variables están positivamente correlacionadas (la elipse apunta hacia la parte superior derecha) o negativamente correlacionadas (la elipse apunta hacia la parte superior izquierda). El sombreado y el ancho de la elipse indican la fuerza de la asociación: las elipses más delgadas y oscuras corresponden a relaciones más fuertes.

Al igual que la media y la desviación estándar, el coeficiente de correlación es sensible a los valores atípicos en los datos. Los paquetes de software ofrecen alternativas robustas al coeficiente de correlación clásico. Los métodos en el módulo `sklearn.covariance` de scikit-learn implementan una variedad de enfoques.

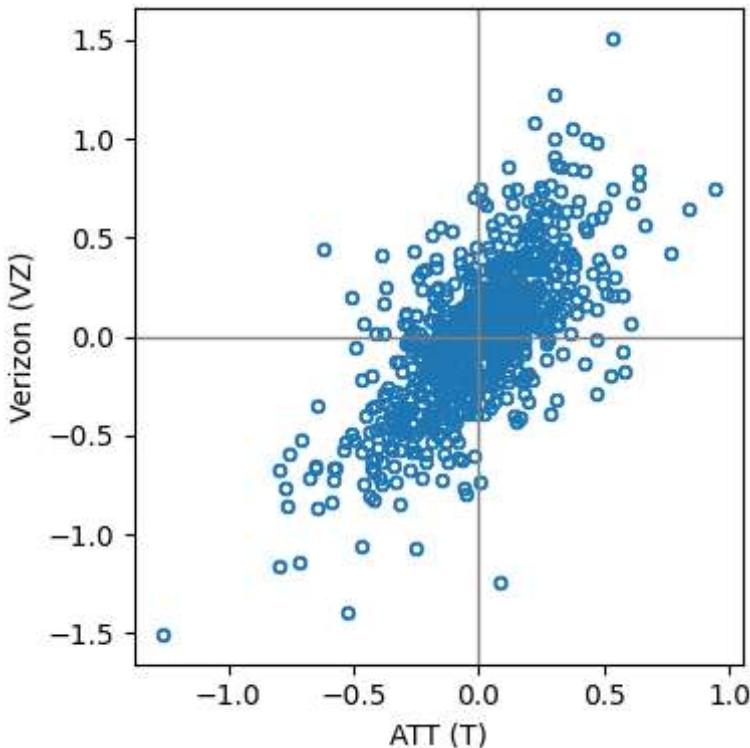
Los estadísticos propusieron otros tipos de coeficientes de correlación, como el rho de Spearman o el tau de Kendall. Estos son coeficientes de correlación basados en el rango de los datos. Los rangos son posiciones de los datos cuando se ordenan de menor a mayor. Dado que trabajan con rangos en lugar de valores, estas estimaciones son robustas frente a valores atípicos y pueden manejar ciertos tipos de no linealidades. Sin embargo, los científicos de datos generalmente pueden ceñirse al coeficiente de correlación de Pearson y sus alternativas robustas para el análisis exploratorio. El atractivo de las estimaciones basadas en rangos es principalmente para conjuntos de datos más pequeños y pruebas de hipótesis específicas.

Gráfico de Dispersion (Scatterplot) Un gráfico en el que el eje x representa el valor de una variable, y el eje y representa el valor de otra variable. Vea la Figura 1-7 para un gráfico de la correlación entre los rendimientos diarios de ATT y Verizon.

Los gráficos de dispersión simples son compatibles con pandas. Especificar el marcador como `\u25EF` usa un círculo abierto para cada punto.

```
In [47]: ax = telecom.plot.scatter(x='T', y='VZ', figsize=(4, 4), marker='$\u25ef$')
ax.set_xlabel('ATT (T)')
ax.set_ylabel('Verizon (VZ)')
ax.axhline(0, color='grey', lw=1)
ax.axvline(0, color='grey', lw=1)

plt.tight_layout()
plt.show()
ax.set_title('Fig 1-7')
```



```
Out[47]: Text(0.5, 1.0, 'Fig 1-7')
```

Exploración de Dos o Más Variables

Estimadores habituales como la media y la varianza analizan variables una a la vez (análisis univariante). El análisis de correlación es un método importante que compara dos variables (análisis bivariante). En esta sección, observamos estimaciones y gráficos adicionales, y más de dos variables (análisis multivariante).

Términos Clave para la Exploración de Dos o Más Variables

- **Tabla de Contingencia:**
 - Un recuento de frecuencias entre dos o más variables categóricas.
- **Hexagonal Binning:**
 - Un gráfico de dos variables numéricas con los registros agrupados en hexágonos.
- **Gráfico de Contorno:**

- Un gráfico que muestra la densidad de dos variables numéricas como un mapa topográfico.

- **Gráfico de Violín:**

- Conocido como *Violin Plot* es un gráfico similar a un boxplot pero mostrando la estimación de densidad.

Al igual que el análisis univariante, el análisis bivariante implica tanto el cálculo de estadísticas resumen como la producción de visualizaciones. El tipo apropiado de análisis bivariante o multivariante depende de la naturaleza de los datos: numéricos versus categóricos.

Ejemplo de tabla de contingencia

Supongamos que tenemos los siguientes datos que muestran el nivel de educación y el estado laboral de un grupo de personas:

Educación	Estado Laboral	Recuento
Secundaria	Empleado	50
Secundaria	Desempleado	10
Secundaria	Estudiante	5
Secundaria	Retirado	5
Preparatoria	Empleado	80
Preparatoria	Desempleado	20
Preparatoria	Estudiante	15
Preparatoria	Retirado	5
Licenciatura	Empleado	100
Licenciatura	Desempleado	15
Licenciatura	Estudiante	10
Licenciatura	Retirado	5
Postgrado	Empleado	60
Postgrado	Desempleado	5
Postgrado	Estudiante	10
Postgrado	Retirado	5

Creamos una tabla de contingencia pivotando la tabla de arriba separando cada categoría del estado laboral en columnas diferentes.

In [19]:

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

```

data = {
    'Educación': [
        'Secundaria', 'Secundaria', 'Secundaria', 'Secundaria',
        'Preparatoria', 'Preparatoria', 'Preparatoria', 'Preparatoria',
        'Licenciatura', 'Licenciatura', 'Licenciatura', 'Licenciatura',
        'Postgrado', 'Postgrado', 'Postgrado', 'Postgrado'
    ],
    'Estado Laboral': [
        'Empleado', 'Desempleado', 'Estudiante', 'Retirado',
        'Empleado', 'Desempleado', 'Estudiante', 'Retirado',
        'Empleado', 'Desempleado', 'Estudiante', 'Retirado',
        'Empleado', 'Desempleado', 'Estudiante', 'Retirado'
    ],
    'Recuento': [
        50, 10, 5, 5,
        80, 20, 15, 5,
        100, 15, 10, 5,
        60, 5, 10, 5
    ]
}

# Creamos el DataFrame
df = pd.DataFrame(data)

```

In [21]:

```

# Crear tabla de contingencia
tabla_contingencia = pd.pivot_table(df, values='Recuento', index='Educación', columns='Estado Laboral')

# Mostrar la tabla de contingencia
tabla_contingencia

```

Out[21]:

	Estado Laboral	Desempleado	Empleado	Estudiante	Retirado	Total
Educación						
Licenciatura	15	100	10	5	130	
Postgrado	5	60	10	5	80	
Preparatoria	20	80	15	5	120	
Secundaria	10	50	5	5	70	
Total	50	290	40	20	400	

Ejercicio 7 Crear un gráfico de barras con las categorías del estado laboral

o bien, podría haberse pivotado tomando como columnas en nivel de educación:

In [22]:

```

# Crear tabla de contingencia con niveles de educación como columnas
tabla_contingencia_1 = pd.pivot_table(df, values='Recuento', index='Estado Laboral', columns='Educación')

# Mostrar la tabla de contingencia
tabla_contingencia_1

```

Out[22]:

	Educación	Licenciatura	Postgrado	Preparatoria	Secundaria	Total
Estado Laboral						
Desempleado	15	5	20	10	50	
Empleado	100	60	80	50	290	
Estudiante	10	10	15	5	40	
Retirado	5	5	5	5	20	
Total	130	80	120	70	400	

Ejercicio 8 Crear un gráfico de barras con las categorías del nivel de educación

Agrupamiento Hexagonal (Hexagonal Binning) y Contornos

Los gráficos de dispersión (Scatterplots) son adecuados cuando hay un número relativamente pequeño de valores de datos. El gráfico de rendimientos de acciones en la Figura 1-7 implica solo unos 750 puntos. Para conjuntos de datos con cientos de miles o millones de registros, un gráfico de dispersión será demasiado denso, por lo que necesitamos una forma diferente de visualizar la relación.

Para ilustrar esto, consideremos el conjunto de datos `kc_tax`, que contiene los valores evaluados de impuestos para propiedades residenciales en el condado de King, Washington.

Para centrarnos en la parte principal de los datos, eliminamos las residencias muy caras y muy pequeñas o grandes utilizando la función `subset`:

En pandas, filtramos el conjunto de datos de la siguiente manera:

```
In [43]: kc_tax = pd.read_csv('kc_tax.csv.gz')
kc_tax0 = kc_tax.loc[(kc_tax.TaxAssessedValue < 750000) &
                     (kc_tax.SqFtTotLiving > 100) &
                     (kc_tax.SqFtTotLiving < 3500), :]
kc_tax0.head()
```

Out[43]:

	TaxAssessedValue	SqFtTotLiving	ZipCode
1	206000.0	1870	98002.0
2	303000.0	1530	98166.0
3	361000.0	2000	98108.0
4	459000.0	3150	98108.0
5	223000.0	1570	98032.0

Significado de Cada Columna en el Conjunto de Datos `kc_tax`

1. TaxAssessedValue:

- **Significado:** Valor Fiscal Evaluado. Representa el valor de la propiedad según la evaluación fiscal realizada por el condado. Este valor es utilizado para calcular los impuestos a la propiedad que el propietario debe pagar.

2. SqFtTotLiving:

- **Significado:** Metros Cuadrados Totales de Vivienda. Indica el área total habitable de la propiedad en pies cuadrados. Incluye todas las áreas habitables dentro de la propiedad, como habitaciones, cocina, baños, etc.

3. ZipCode:

- **Significado:** Código Postal. Contiene el código postal de la ubicación de la propiedad. El código postal es una serie de números que se utilizan para identificar una región específica dentro del condado, facilitando la localización de la propiedad.

Estos datos son típicamente utilizados en análisis inmobiliarios y fiscales para evaluar el mercado de la vivienda, analizar tendencias de precios y calcular impuestos sobre la propiedad.

```
In [44]: print(kc_tax0.shape)
```

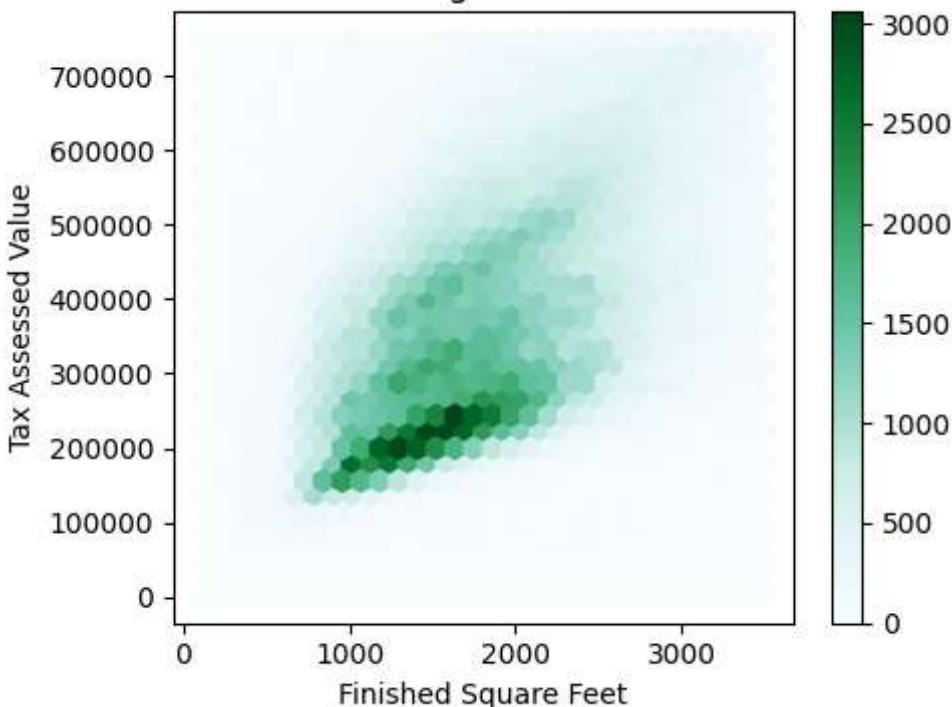
```
(432693, 3)
```

La Figura 1-8 es un gráfico de agrupamiento hexagonal que muestra la relación entre los pies cuadrados terminados y el valor fiscal evaluado para las casas en el condado de King. En lugar de trazar puntos, que aparecerían como una nube oscura monolítica, agrupamos los registros en hexágonos y trazamos los hexágonos con un color que indica el número de registros en ese hexágono. En este gráfico, la relación positiva entre los pies cuadrados y el valor fiscal evaluado es clara.

Una característica interesante es la indicación de bandas adicionales por encima de la banda principal (la más oscura) en la parte inferior, lo que indica casas que tienen la misma superficie en pies cuadrados que las de la banda principal, pero con un valor fiscal evaluado más alto.

```
In [25]: ax = kc_tax0.plot.hexbin(x='SqFtTotLiving', y='TaxAssessedValue',
                               gridsize=30, sharex=False, figsize=(5, 4))
ax.set_xlabel('Finished Square Feet')
ax.set_ylabel('Tax Assessed Value')
ax.set_title('Fig. 1-8')
plt.tight_layout()
plt.show()
```

Fig. 1-8

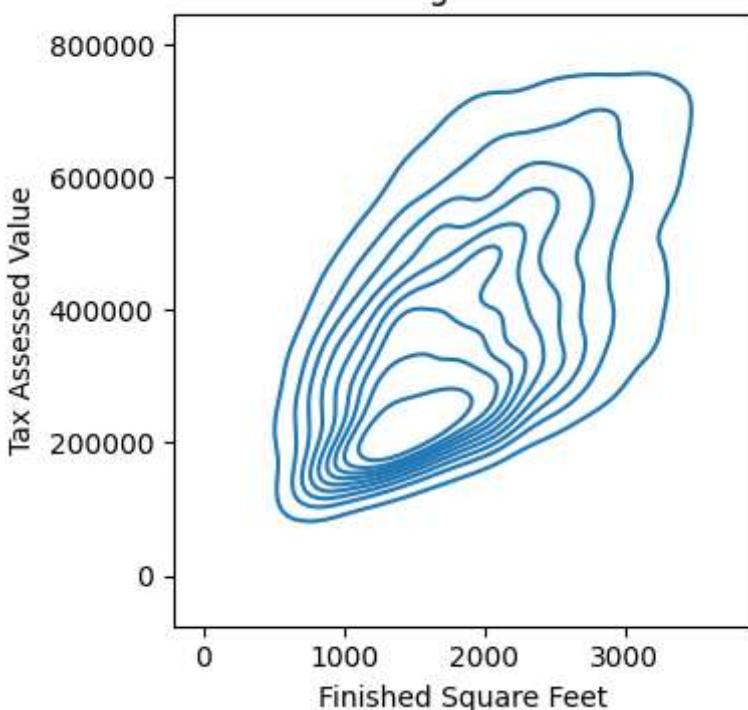


La Figura 1-9 utiliza contornos superpuestos a un gráfico de dispersión para visualizar la relación entre dos variables numéricas. Los contornos son esencialmente un mapa topográfico para dos variables; cada banda de contorno representa una densidad específica de puntos, aumentando a medida que uno se acerca a un "pico". Este gráfico muestra una historia similar a la Figura 1-8 : hay un pico secundario "al norte" del pico principal.

```
In [27]: fig, ax = plt.subplots(figsize=(4, 4))
sns.kdeplot(data=kc_tax0.sample(10000), x='SqFtTotLiving', y='TaxAssessedValue',
ax.set_xlabel('Finished Square Feet')
ax.set_ylabel('Tax Assessed Value')
ax.set_title('Fig. 1-9')

plt.tight_layout()
plt.show()
```

Fig. 1-9



```
In [29]: import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
from scipy.stats import gaussian_kde

# Supongamos que kc_tax0 ya ha sido filtrado según el ejemplo proporcionado previamente

# Tomar una muestra de los datos
sampled_data = kc_tax0.sample(10000)

# Preparar los datos para el gráfico 3D
x = sampled_data['SqFtTotLiving']
y = sampled_data['TaxAssessedValue']

# Calcular el KDE
kde = gaussian_kde([x, y])

# Crear una malla de cuadricula para el gráfico 3D
x_grid, y_grid = np.meshgrid(np.linspace(x.min(), x.max(), 100), np.linspace(y.min(), y.max(), 100))
z_grid = kde(np.vstack([x_grid.ravel(), y_grid.ravel()])).reshape(x_grid.shape)

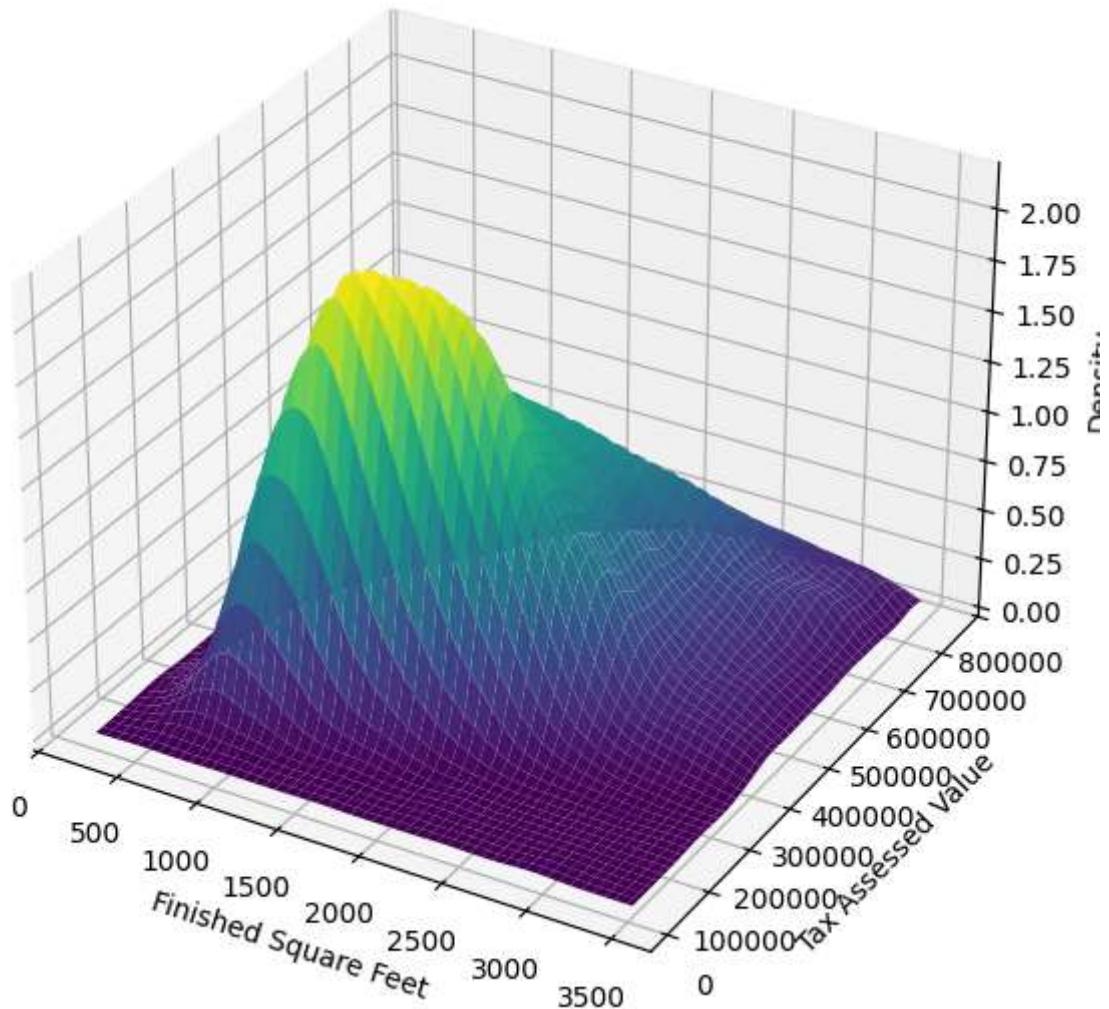
# Crear el gráfico de contornos en 3D
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Dibujar la superficie en 3D
ax.plot_surface(x_grid, y_grid, z_grid, cmap='viridis', edgecolor='none')

# Configurar etiquetas y título
ax.set_xlabel('Finished Square Feet')
ax.set_ylabel('Tax Assessed Value')
ax.set_zlabel('Density')
ax.set_title('Fig. 1-9')
```

```
plt.tight_layout()
plt.show()
```

Fig. 1-9



Análisis con dos variables categóricas

Una forma útil de resumir dos variables categóricas es una tabla de contingencia, una tabla de recuentos por categoría. La Tabla 1-8 muestra la tabla de contingencia entre la calificación de un préstamo personal y el resultado de ese préstamo. Esto se toma de los datos proporcionados por Lending Club, un líder en el negocio de préstamos . La calificación va de A (alta) a G (baja). El resultado es ya sea totalmente pagado, actual, atrasado o cancelado (no se espera recuperar el saldo del préstamo). Esta tabla muestra el recuento y los porcentajes por fila. Los préstamos de alta calificación tienen un porcentaje muy bajo de atrasos/cancelaciones en comparación con los préstamos de menor calificación.

Tabla 1-8. Tabla de contingencia de la calificación y el estado del préstamo

Grade	Charged off	Current	Fully paid	Late	Total
A	1562	50051	20408	469	72490
	0.022	0.690	0.282	0.006	0.161
B	5302	93852	31160	2056	132370
	0.040	0.709	0.235	0.016	0.294
C	6023	88928	23147	2777	120875
	0.050	0.736	0.191	0.023	0.268
D	5007	53281	13681	2308	74277
	0.067	0.717	0.184	0.031	0.165
E	2842	24639	5949	1374	34804
	0.082	0.708	0.171	0.039	0.077
F	1526	8444	2328	606	12904
	0.118	0.654	0.180	0.047	0.029
G	409	1990	643	199	3241
	0.126	0.614	0.198	0.061	0.007
Total	22671	321185	97316	9789	450961

Ejercicio 8 Partiendo del dataset `lc_loans.csv` crear un dataframe que tenga la misma estructura de la `tabla 1-8` dada arriba.

Significado de las Columnas de la Tabla 1-8

1. Grade:

- **Significado:** Calificación del Préstamo. Indica la calificación de riesgo del préstamo, que va de A (alta) a G (baja). Una calificación más alta (A) sugiere un menor riesgo de incumplimiento, mientras que una calificación más baja (G) indica un mayor riesgo.

2. Charged off:

- **Significado:** Cancelado. Muestra el número de préstamos que han sido cancelados. Un préstamo se considera "cancelado" cuando el prestamista ya no espera recuperar el saldo del préstamo debido a que el prestatario no ha cumplido con los pagos.

3. Current:

- **Significado:** Al día. Muestra el número de préstamos que están al día con sus pagos. Estos préstamos están siendo pagados según el cronograma acordado.

4. Fully paid:

- **Significado:** Totalmente Pagado. Muestra el número de préstamos que han sido completamente pagados por los prestatarios. Esto indica que el prestatario

ha cumplido con todas sus obligaciones de pago.

5. Late:

- **Significado:** Atrasado. Muestra el número de préstamos que están atrasados. Un préstamo se considera "atrasado" cuando el prestatario no ha realizado los pagos a tiempo.

6. Total:

- **Significado:** Total. Muestra el total de préstamos en cada calificación. Incluye todos los préstamos que están cancelados, al día, totalmente pagados y atrasados.

Notas sobre las Filas de la Tabla

- Las filas con porcentajes debajo de cada categoría (Charged off, Current, Fully paid, Late) representan los porcentajes correspondientes al número total de préstamos en cada calificación.
- La última fila muestra el total de todos los préstamos en todas las calificaciones, tanto en números absolutos como en porcentajes.

Esta tabla proporciona una visión general de la calidad de los préstamos según su calificación y estado, permitiendo analizar la relación entre la calificación del préstamo y su resultado.

El método `pivot_table` crea la tabla de pivote en Python. El argumento `aggfunc` nos permite obtener los conteos. Calcular los porcentajes es un poco más complicado:

```
crosstab = lc_loans.pivot_table(index='grade', columns='status',
                                 aggfunc=lambda x: len(x), margins=True)
df = crosstab.loc['A':'G', :].copy()
df.loc[:, 'Charged Off':'Late'] = df.loc[:, 'Charged
Off':'Late'].div(df['All'], axis=0)
df['All'] = df['All'] / sum(df['All'])
perc_crosstab = df
```

```
In [35]: lc_loans = pd.read_csv('lc_loans.csv')
crosstab = lc_loans.pivot_table(index='grade', columns='status',
                                 aggfunc=lambda x: len(x), margins=True)
# El argumento `margins` añadirá las sumas de las columnas y las filas.
df = crosstab.loc['A':'G', :].copy()
# Creamos una copia de la tabla de pivote, ignorando las sumas de las columnas.
df.loc[:, 'Charged Off':'Late'] = df.loc[:, 'Charged Off':'Late'].div(df['All'],
# Dividimos las filas por la suma de la fila.
df['All'] = df['All'] / sum(df['All'])
# Dividimos la columna 'All' por su suma.
perc_crosstab = df
perc_crosstab
```

```
C:\Users\juanj\AppData\Local\Temp\ipykernel_16344\4071278522.py:5: FutureWarning:
Setting an item of incompatible dtype is deprecated and will raise in a future error of pandas. Value '[0.0215478  0.04005439  0.04982834  0.06740983  0.08165728  0.1
182579
 0.12619562]' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
    df.loc[:, 'Charged Off':'Late'] = df.loc[:, 'Charged Off':'Late'].div(df['All'], axis=0)
C:\Users\juanj\AppData\Local\Temp\ipykernel_16344\4071278522.py:5: FutureWarning:
Setting an item of incompatible dtype is deprecated and will raise in a future error of pandas. Value '[0.69045386  0.70901262  0.73570217  0.71732838  0.70793587  0.6
5437074
 0.61400802]' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
    df.loc[:, 'Charged Off':'Late'] = df.loc[:, 'Charged Off':'Late'].div(df['All'], axis=0)
C:\Users\juanj\AppData\Local\Temp\ipykernel_16344\4071278522.py:5: FutureWarning:
Setting an item of incompatible dtype is deprecated and will raise in a future error of pandas. Value '[0.28152849  0.23540077  0.19149535  0.18418891  0.17092863  0.1
8040918
 0.19839556]' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
    df.loc[:, 'Charged Off':'Late'] = df.loc[:, 'Charged Off':'Late'].div(df['All'], axis=0)
C:\Users\juanj\AppData\Local\Temp\ipykernel_16344\4071278522.py:5: FutureWarning:
Setting an item of incompatible dtype is deprecated and will raise in a future error of pandas. Value '[0.00646986  0.01553222  0.02297415  0.03107288  0.03947822  0.0
4696218
 0.0614008 ]' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
    df.loc[:, 'Charged Off':'Late'] = df.loc[:, 'Charged Off':'Late'].div(df['All'], axis=0)
```

Out[35]: **status Charged Off Current Fully Paid Late All**

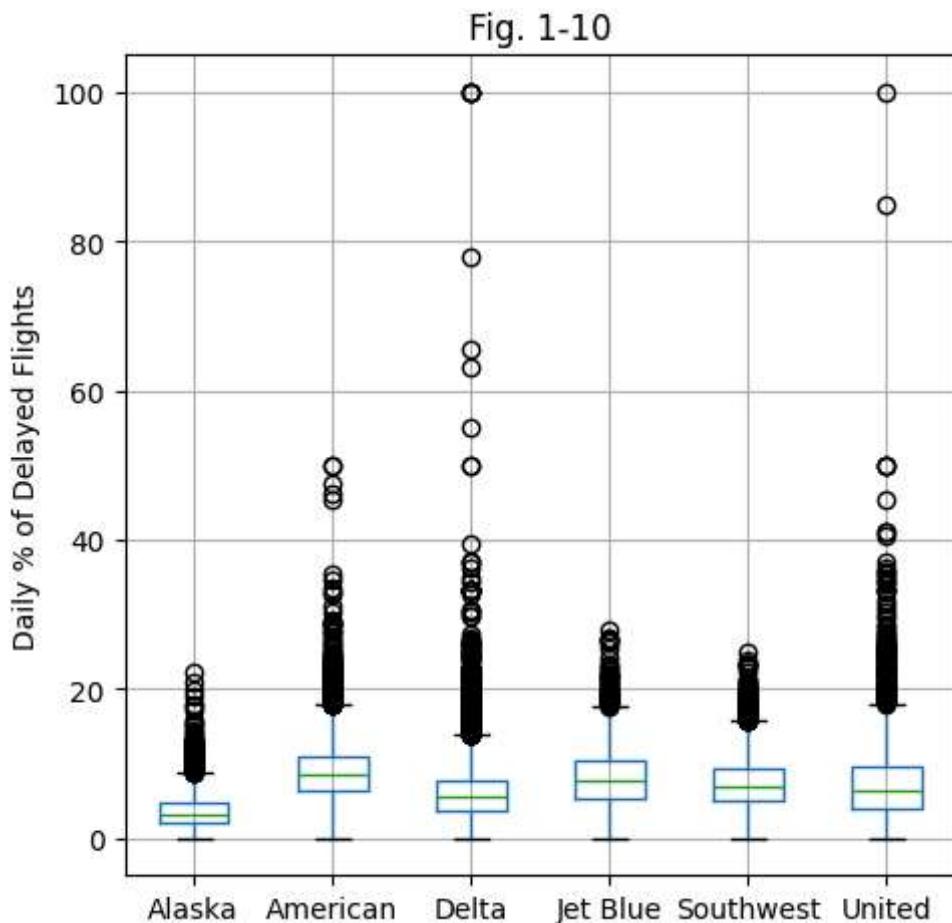
grade					
A	0.021548	0.690454	0.281528	0.006470	0.160746
B	0.040054	0.709013	0.235401	0.015532	0.293529
C	0.049828	0.735702	0.191495	0.022974	0.268039
D	0.067410	0.717328	0.184189	0.031073	0.164708
E	0.081657	0.707936	0.170929	0.039478	0.077177
F	0.118258	0.654371	0.180409	0.046962	0.028614
G	0.126196	0.614008	0.198396	0.061401	0.007187

Datos Categóricos y Numéricos

Los diagramas de caja y bigote (boxplot) son tambien una forma sencilla de comparar visualmente las distribuciones de una variable numérica agrupada según una variable categórica.

Por ejemplo, podríamos querer comparar cómo varía el porcentaje de retrasos de vuelos entre diferentes aerolíneas. La Figura 1-10 muestra el porcentaje de vuelos en un mes que se retrasaron cuando el retraso estaba bajo el control del transportista:

```
In [38]: airline_stats = pd.read_csv('airline_stats.csv')
airline_stats.head()
ax = airline_stats.boxplot(by='airline', column='pct_carrier_delay',
                           figsize=(5, 5))
ax.set_xlabel('')
ax.set_ylabel('Daily % of Delayed Flights')
plt.suptitle('')
ax.set_title('Fig. 1-10')
plt.tight_layout()
plt.show()
```



Alaska se destaca por tener la menor cantidad de retrasos, mientras que American tiene la mayor cantidad de retrasos: el cuartil inferior para American es más alto que el cuartil superior para Alaska. a.

Violin Plot

Un diagrama de violín (violin plot) es una mejora del diagrama de caja tradicional (boxplot) y fue introducido por Hintze y Nelson en 1998. A continuación se detalla su funcionamiento y ventajas:

1. Estimación de Densidad:

- El diagrama de violín incluye una estimación de densidad de la distribución de los datos. La densidad de probabilidad se calcula utilizando técnicas como la estimación de densidad de kernel (KDE).
- La densidad de probabilidad proporciona una idea de qué tan probable es que los datos tomen ciertos valores. En el contexto del diagrama de violín, esta densidad se representa en el eje y.

2. Forma del Diagrama:

- La densidad calculada se refleja y se volteá sobre el eje y, creando dos imágenes especulares una frente a la otra. Esta forma resultante se rellena, creando una figura que se asemeja a un violín, de ahí el nombre del diagrama.
- El eje x representa la variable categórica (por ejemplo, diferentes grupos o categorías), mientras que el eje y muestra la densidad de la variable numérica.

3. Ventajas del Diagrama de Violín:

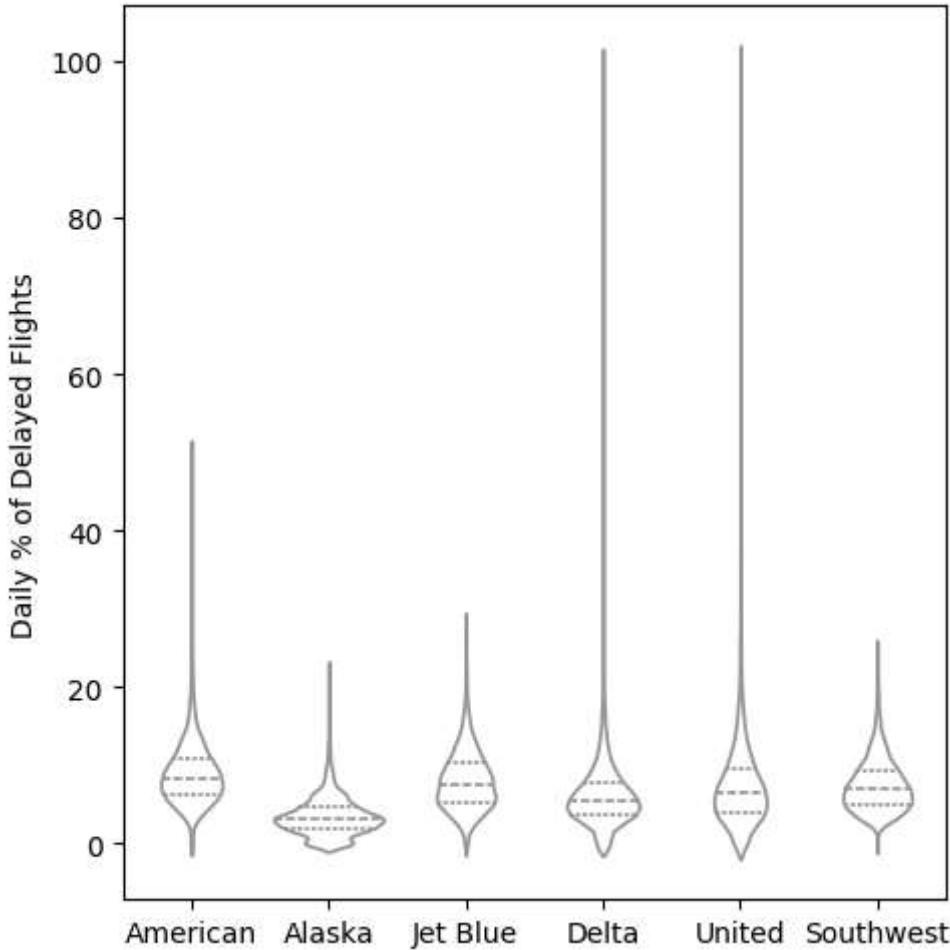
- **Visualización de la Densidad:** A diferencia del diagrama de caja, que muestra solo los estadísticos descriptivos básicos (mediana, cuartiles y valores atípicos), el diagrama de violín permite visualizar la distribución completa de los datos. Esto incluye todos los picos y valles en la densidad de los datos.
- **Detalles Máticos:** El diagrama de violín puede mostrar detalles en la distribución de los datos que no son perceptibles en un diagrama de caja. Por ejemplo, si los datos tienen múltiples modos (picos) o si la distribución es asimétrica, estas características serán visibles en un diagrama de violín pero no en un diagrama de caja.
- **Comparación de Distribuciones:** Al igual que el diagrama de caja, el diagrama de violín es útil para comparar la distribución de una variable numérica entre diferentes grupos categóricos. Sin embargo, proporciona una visión más rica y detallada de cómo varían las distribuciones entre los grupos.

Supongamos que queremos comparar la distribución de los retrasos en vuelos entre diferentes aerolíneas. Un diagrama de violín no solo mostrará los cuartiles y la mediana de los retrasos, sino que también permitirá visualizar si los retrasos tienen una distribución unimodal o multimodal, si hay asimetría y qué tan concentrados están los retrasos en torno a ciertos valores.

El diagrama de violín es una herramienta poderosa para la visualización de datos, ya que combina las ventajas del diagrama de caja con la capacidad de mostrar la densidad de probabilidad de los datos, proporcionando una visión más completa y detallada de la distribución de los datos.

```
In [39]: fig, ax = plt.subplots(figsize=(5, 5))
sns.violinplot(data=airline_stats, x='airline', y='pct_carrier_delay',
                ax=ax, inner='quartile', color='white')
ax.set_xlabel('')
ax.set_ylabel('Daily % of Delayed Flights')
```

```
plt.tight_layout()
plt.show()
```



El gráfico correspondiente se muestra en la Figura 1-11. El diagrama de violín muestra una concentración en la distribución cerca de cero para Alaska y, en menor medida, Delta. Este fenómeno no es tan obvio en el diagrama de caja.

In [41]:

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

# Cargar el archivo CSV
airline_stats = pd.read_csv('airline_stats.csv')

# Crear el gráfico de violín con el diagrama de caja superpuesto
fig, ax = plt.subplots(figsize=(8, 6))

# Diagrama de violín
sns.violinplot(data=airline_stats, x='airline', y='pct_carrier_delay',
                inner=None, palette='Set2', ax=ax)

# Diagrama de caja superpuesto
sns.boxplot(data=airline_stats, x='airline', y='pct_carrier_delay',
            whis=np.inf, color='white', linewidth=1.5, ax=ax)

ax.set_xlabel('')
ax.set_ylabel('Daily % of Delayed Flights')
ax.set_title('Fig. 1-11')
```

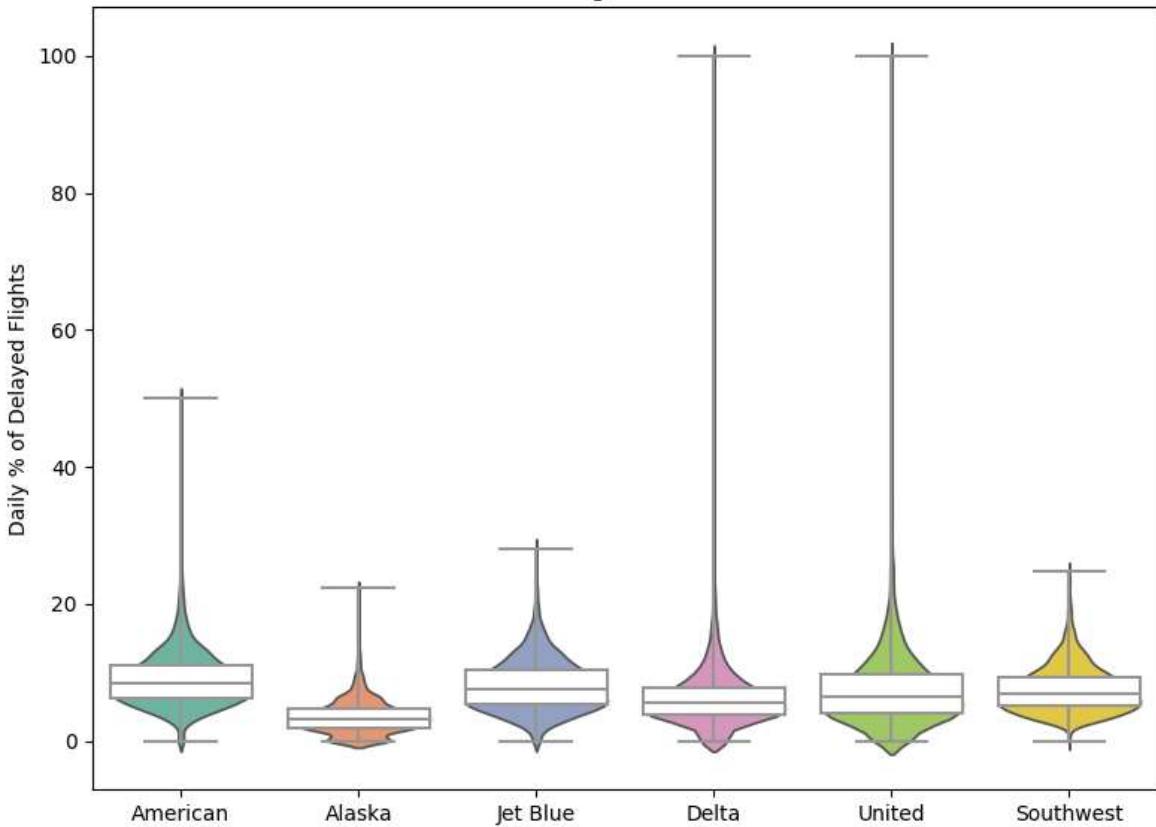
```
plt.tight_layout()
plt.show()
```

C:\Users\juanj\AppData\Local\Temp\ipykernel_16344\3263343933.py:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v 0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.violinplot(data=airline_stats, x='airline', y='pct_carrier_delay',
```

Fig. 1-11



In [40]:

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

```
airline_stats = pd.read_csv('airline_stats.csv')

# violin plot con diferentes colores
fig, ax = plt.subplots(figsize=(5, 5))
sns.violinplot(data=airline_stats, x='airline', y='pct_carrier_delay',
                ax=ax, inner='quartile', palette='Set2')
ax.set_xlabel('')
ax.set_ylabel('Daily % of Delayed Flights')
ax.set_title('Fig. 1-11')

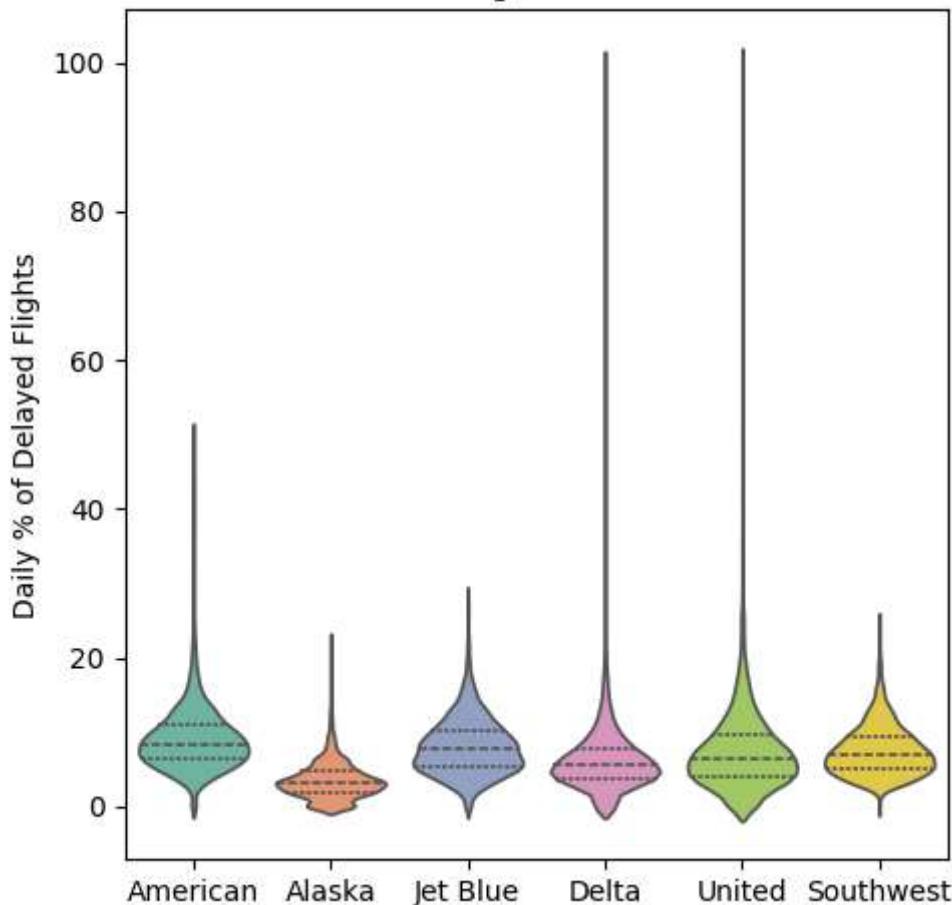
plt.tight_layout()
plt.show()
```

```
C:\Users\juanj\AppData\Local\Temp\ipykernel_16344\1934946127.py:10: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in v 0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.violinplot(data=airline_stats, x='airline', y='pct_carrier_delay',
```

Fig. 1-11



El gráfico de violín muestra la distribución de los porcentajes de retrasos causados por la aerolínea para cada una de las compañías aéreas presentes en el conjunto de datos. Aquí hay algunos puntos clave de interpretación:

- **Forma y Ancho del Violín:** La forma y el ancho del gráfico de violín en diferentes partes indican la densidad de los datos en esas áreas. Las secciones más anchas del violín indican una mayor densidad de valores en ese rango específico de porcentajes de retrasos.
- **Mediana y Cuartiles:** La línea en el centro del violín representa la mediana del porcentaje de retrasos por la aerolínea, mientras que otras las líneas negras delgadas muestran el rango intercuartílico (IQR), es decir, el rango entre el primer cuartil (Q1) y el tercer cuartil (Q3).
- **Comparación entre Aerolíneas:** Comparando los gráficos de violín de las diferentes aerolíneas, se pueden observar diferencias en la distribución de los retrasos. Algunas aerolíneas pueden tener una mayor variabilidad en los retrasos, mientras que otras muestran distribuciones más concentradas.

En general, un diagrama de violín combina aspectos de un diagrama de caja y un gráfico de densidad para proporcionar una visualización detallada de la distribución de datos. Aquí se explica cómo interpretar un diagrama de violín:

Elementos de un Diagrama de Violín

Forma del Violín:

- La forma del diagrama de violín representa la densidad de los datos a diferentes valores de la variable. Las áreas más anchas indican donde hay más datos concentrados, mientras que las áreas más estrechas indican donde hay menos datos.

Parte Central (Diagrama de Caja):

- Dentro del diagrama de violín, generalmente se incluye un diagrama de caja que muestra la mediana (línea central), el rango intercuartílico (IQR, la caja) y, a veces, los valores atípicos (puntos fuera de los "bigotes").
 - **Mediana:** La línea blanca o negra en el centro del violín indica el valor mediano.
 - **Cuartiles:** Las líneas dentro del violín muestran el primer y tercer cuartil, delimitando el rango intercuartílico (IQR).

Densidad Espejada:

- La densidad de los datos se refleja a ambos lados del eje y se volteea horizontalmente. Esto proporciona una imagen simétrica que facilita la comparación visual.

Color:

- Los colores pueden diferenciar categorías o grupos, lo que facilita la comparación entre diferentes grupos en el mismo gráfico.

Interpretación

Distribución de Datos:

- El ancho del violín en diferentes puntos a lo largo del eje y muestra dónde están más concentrados los datos. Una sección más ancha indica una mayor densidad de datos en ese rango.

Mediana y Cuartiles:

- La mediana proporciona el valor central de los datos, mientras que los cuartiles y el rango intercuartílico (IQR) muestran la dispersión de la mitad central de los datos.

Valores Atípicos:

- Si el diagrama de caja interno incluye puntos fuera de los bigotes, estos representan valores atípicos, es decir, valores que están significativamente alejados del resto de los datos.

Comparación entre Grupos:

- Cuando se tienen múltiples violines en el mismo gráfico (para diferentes categorías o grupos), se puede comparar la forma y la dispersión de las distribuciones entre los grupos. Diferencias en la anchura, la forma del violín y la posición de la mediana pueden indicar diferencias significativas en la distribución dePara el caso de nuestro ejemplorolínea puede mostrar lo siguiente:
- **Alaska Airlines:** Tiene un violín estrecho con la mayoría de los datos concentrados cerca de la mediana, indicando poca variabilidad y pocos retrasos.
- **American Airlines:** Tiene un violín más ancho con un rango intercuartílico mayor y más valores atípicos, indicando una mayor variabilidad en los retrasos y una mayor frecuencia de retrasos.

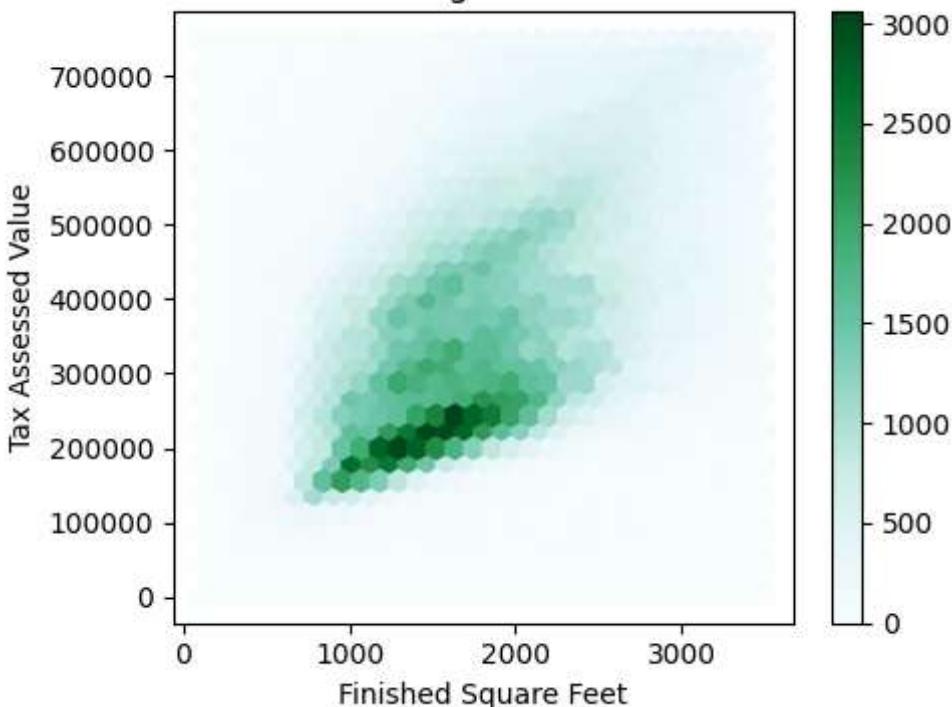
Visualizando Múltiples Variables

Los tipos de gráficos utilizados para comparar dos variables—diagramas de dispersión, binning hexagonal y diagramas de caja—se pueden extender fácilmente a más variables mediante la noción de condicionamiento.

Como ejemplo, volvamos a la **Figura 1-8**, que mostraba la relación entre los pies cuadrados terminados de las casas y sus valores tasados para impuestos. Observamos que parece haber un grupo de casas que tienen un mayor valor tasado por pie cuadrado.

```
In [45]: ax = kc_tax0.plot.hexbin(x='SqFtTotLiving', y='TaxAssessedValue',
                               gridsize=30, sharex=False, figsize=(5, 4))
ax.set_xlabel('Finished Square Feet')
ax.set_ylabel('Tax Assessed Value')
ax.set_title('Fig. 1-8')
plt.tight_layout()
plt.show()
```

Fig. 1-8



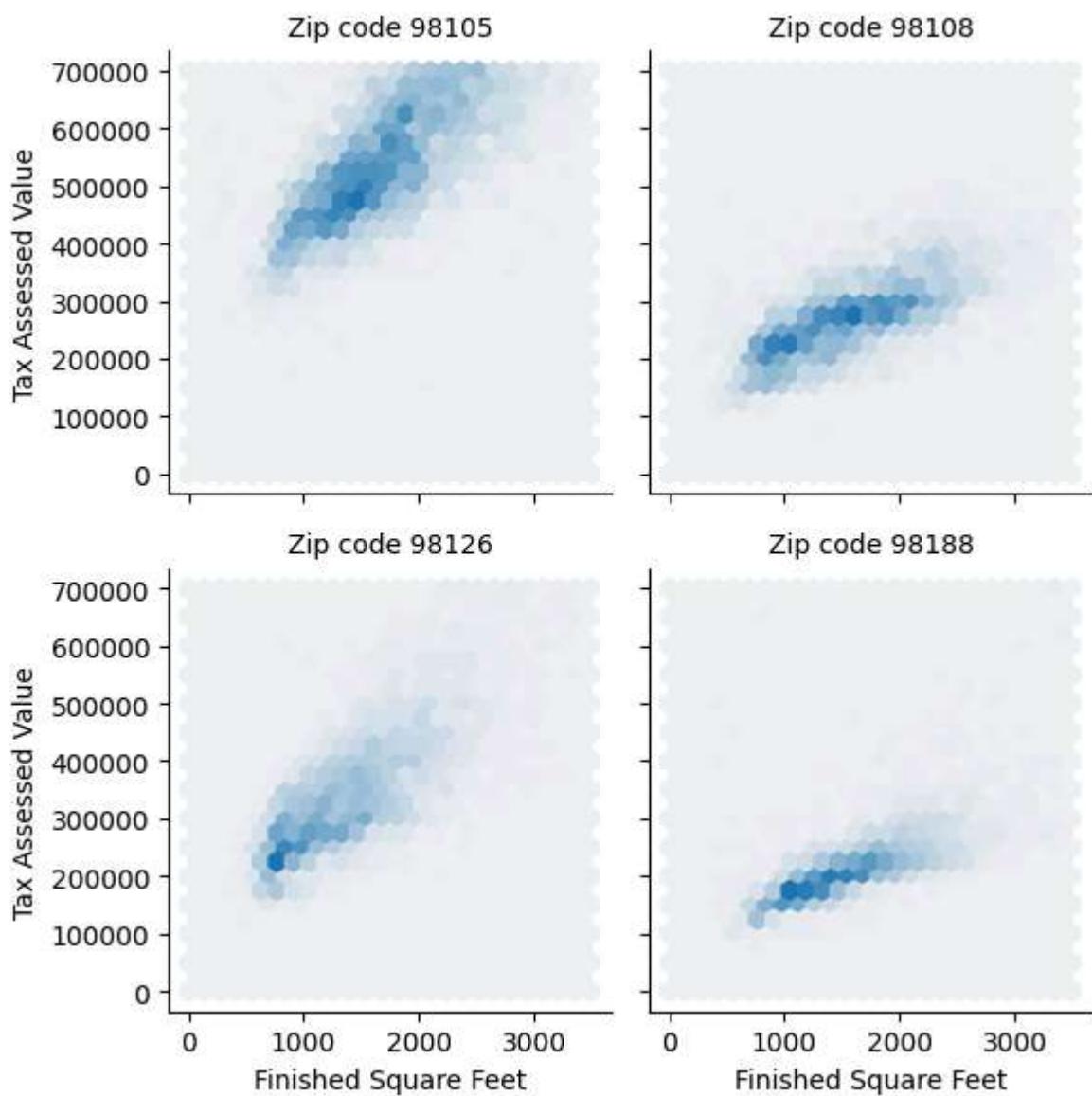
Analizando más a fondo, la Figura 1-12 tiene en cuenta el efecto de la ubicación al graficar los datos para un conjunto de códigos postales. Ahora la imagen es mucho más clara: el valor tasado para impuestos es mucho más alto en algunos códigos postales (98105, 98126) que en otros (98108, 98188). Esta disparidad da lugar a los grupos observados en la Figura 1-8 .

```
In [46]: zip_codes = [98188, 98105, 98108, 98126]
kc_tax_zip = kc_tax0.loc[kc_tax0.ZipCode.isin(zip_codes), :]
kc_tax_zip

def hexbin(x, y, color, **kwargs):
    cmap = sns.light_palette(color, as_cmap=True)
    plt.hexbin(x, y, gridsize=25, cmap=cmap, **kwargs)

g = sns.FacetGrid(kc_tax_zip, col='ZipCode', col_wrap=2)
g.map(hexbin, 'SqFtTotLiving', 'TaxAssessedValue',
      extent=[0, 3500, 0, 700000])
g.set_axis_labels('Finished Square Feet', 'Tax Assessed Value')
g.set_titles('Zip code {col_name:.0f}')

plt.tight_layout()
plt.show()
```



In []: