Resumen del capítulo: Transformación de datos

Agrupación de datos

Hacer una operación `groupby()` cambia el índice de fila de los datos a las claves por las que estamos agrupando. Para agrupar por varias columnas, pasamos una lista al método `groupby()`.

El objeto `DataFrameGroubBy` forma parte de un framework de procesamiento de datos llamado **dividir-aplicar-combinar:**

1. **dividir** los datos en grupos;
2. **aplicar** una función de agregación estadística a cada grupo;
3. **combinar** los resultados para cada grupo.

Ejemplo. Dividimos los datos en grupos con `df.groupby(['platform', 'genre'])`, aplicamos el método `mean()` y combinamos el resultado en un objeto Series con `grp['critic_score'].mean()`.

```
import pandas as pd

df = pd.read_csv('/datasets/vg_sales.csv')
df.dropna(inplace=True)

grp = df.groupby(['platform', 'genre'])
mean_scores = grp['critic_score'].mean()
print(mean_scores)
```

Procesamiento de datos agrupados con agg()

El método agg() usa un diccionario como entrada donde las claves son los nombres de columnas y los valores correspondientes son las funciones de agregación que quieres

aplicarles. Es útil obtener diferentes estadísticas de resumen para diferentes columnas. Incluso podemos aplicar nuestras propias funciones personalizadas con agg().

```
import pandas as pd

df = pd.read_csv('/stats/vg_sales.csv')
df.dropna(inplace=True)

agg_dict = {'critic_score': 'mean', 'jp_sales': 'sum'}

grp = df.groupby(['platform', 'genre'])
print(grp.agg(agg_dict))
```

Agregación de datos con pivot_table()

`pandas` también ofrece tablas dinámicas ("pivot tables") como método alternativo para agrupar y analizar datos. Las tablas dinámicas son una gran herramienta para sintetizar conjuntos de datos y explorar sus diferentes dimensiones. Son muy populares en las aplicaciones de hojas de cálculo como Excel, pero es aún más impresionante crearlas mediante programación con `pandas`.

Los parámetros comunes del método `pivot_table()` son:

- `index=`: la columna cuyos valores se convierten en índices en la tabla dinámica;
- `columns=`: la columna cuyos valores se convierten en columnas en la tabla dinámica;
- `values=`: la columna cuyos valores queremos agregar en la tabla dinámica;
- `aggfunc=`: la función de agregación que queremos aplicar a los valores en cada grupo de filas y columnas.

Ejemplo. Calcula las ventas totales en Europa para cada combinación de género/plataforma.

```
import pandas as pd

df = pd.read_csv('/datasets/vg_sales.csv')
df.dropna(inplace=True)


pivot_data = df.pivot_table(index='genre',
```

```
                            columns='platform',
                            values='eu_sales',
                            aggfunc='sum'
                            )
```

El uso de una tabla dinámica aquí es conveniente porque fácilmente excluimos todas las columnas de df que no nos interesaban para nuestro análisis. Además, puede ser más fácil de leer que el código equivalente basado en `groupby()`.

Cuando se trabaja en tareas, es importante elegir el método de promedio adecuado, ya que eso podría afectar los resultados. En algunos casos, la media aritmética describe los datos con mayor precisión, mientras que en otros puede dar un resultado incorrecto, lo que lleva a la necesidad de calcular la mediana. El método `pivot_table()` acepta diferentes funciones de agregación para el parámetro `aggfunc`, por ejemplo:

- `median`;
- `count` (número de valores);
- `sum`;
- `min`;
- `max`;
- `first` (el primer valor del grupo);
- `last` (el último valor del grupo).

Al llamar a `pivot_table()`, podemos pasar varias funciones al parámetro `aggfunc` a la vez. Por ejemplo, `aggfunc=['median', 'count']` calculará tanto la mediana como el número de valores. Se mostrarán en columnas vecinas en la tabla resultante.

Combinar DataFrames 'verticalmente' con concat()

El método `concat()` se utiliza típicamente para combinar filas de DataFrames separados.

```
import pandas as pd

df = pd.read_csv('/datasets/vg_sales.csv')
```

```
rpgs = df[df['genre'] == 'Role-Playing']
platformers = df[df['genre'] == 'Platform']

df_all_games = pd.concat([rpgs, platformers])
```

¡Dos DataFrames se unen en uno! Recuerda que esto funciona aquí porque ambos DataFrames más pequeños tienen las mismas columnas.

Combinar DataFrames 'horizontalmente' usando `merge()`

El método `merge()` permite combinar dos DataFrames mediante algunas claves.

```
df_1.merge(df_2, on='user_id', how='left')
```

Hay varios tipos de uniones:

- `'inner'`: la conjunción lógica de ambas tablas (se conservan los registros que están presentes en ambos DataFrames).
- `'left'`: todos los valores del DataFrame izquierdo están presentes en el DataFrame fusionado. Los valores del DataFrame derecho solo se conservan para los valores que coinciden con la columna especificada en el DataFrame izquierdo.
- `'right'`: funciona de manera idéntica a una unión izquierda, excepto que el DataFrame combinado conserva todos los valores del DataFrame derecho.
- `'outer'`: todos los valores en la columna especificada se conservan de ambos DataFrames originales, pero el DataFrame fusionado tiene valores ausentes donde no hay ninguna coincidencia.

El modo de unión se establece con el parámetro `how`.

`pandas` agregará automáticamente sufijos a los nombres de las columnas cuando las columnas tengan el mismo nombre en los DataFrames fusionados. Los sufijos por defecto son `_x` y `_y`. Podemos establecer mejores sufijos pasando una lista de cadenas de sufijos al parámetro `suffixes=` en merge():

```
both_pupils = first_pupil_df.merge(second_pupil_df,
                                   on='author',
                                   suffixes=['_1st_student', '_2nd_student']
                                   )
```

Si las columnas utilizadas para fusionarse tienen nombres diferentes.

```
both_pupils = first_pupil_df.merge(second_pupil_df,
                                   left_on='author',
                                   right_on='Author',
                                   )
```

Si se asigna un nombre a una columna de índice, el nombre también se puede pasar al parámetro `on`. También es posible combinar varias columnas a la vez; simplemente pasa una lista de ellas al argumento `on`.

El método `join()` es similar al método `merge()`, se puede considerar como una función de acceso directo para `merge()`.