Resumen del capítulo: Visualización de los datos

Visualización de los datos

La visualización de los datos nos permite ver patrones y relaciones de forma instantánea, lo cual nos ayuda a entender los datos. También nos permite sacar conclusiones, formular nuevas preguntas y comunicar mejor nuestros resultados a otras personas.

Hay dos razones principales para crear visualizaciones como profesional de los datos:

1. Para analizar grandes cantidades de datos revelando sus propiedades visualmente, en otras palabras, *visualizar para analizar*. Al visualizar datos, puedes obtener una percepción que sería imposible si estuvieras leyendo cifras sin procesar.
2. Para comunicar los resultados de tu análisis a tus colegas o clientes de forma comprensible y eficaz.

Elegir el tipo de gráfico correcto

- Gráfico de barras. Los gráficos de barras nos permiten comparar propiedades numéricas (por ejemplo, población) entre categorías (por ejemplo, estados).
- Histograma. Un histograma es un gráfico que muestra la frecuencia con la que aparecen diferentes valores para una variable en tu conjunto de datos. Aunque pueden parecerse a los gráficos de barras, hay diferencias.
- Gráfico de líneas. Son excelentes cuando tienes datos que se conectan cronológicamente y cada punto de tiempo de los datos tiene alguna dependencia con el punto anterior. Cosas como datos de temperatura, datos de tráfico y datos del mercado de valores son buenos candidatos para los gráficos de líneas.
- Gráfico de dispersión. Es simplemente un gráfico en el que se traza un solo punto para cada conjunto de variables, pero los puntos no están conectados por líneas. Los gráficos de dispersión son excelentes para visualizar la relación entre las variables.

Sugerencias de gráficos

Graficar con pandas

`plot()` crea gráficos utilizando los valores en las columnas de DataFrame. Los índices están en el eje X y los valores de las columnas están en el eje Y.

```
df.plot()
```

Podemos cambiar los índices de los ejes, asignando al eje correspondiente los valores de la columna que necesitemos.

```
df.plot(x='column_x', y='column_y')
```

Personalización de gráficos con parámetros plot()

Podemos gestionar el tamaño del gráfico con el parámetro `figsize`. El ancho y el alto en pulgadas se pasan como una tupla:

```
df.plot(figsize=(x_size, y_size))
```

El método `plot()` también tiene el parámetro `style` que se encarga del aspecto de los puntos:

- `'o'`: en lugar de una línea continua, cada valor se marcará como un punto;
- `'x'`: en lugar de una línea continua, cada punto se marcará como una x;
- `'o-'`: mostrará tanto líneas como puntos.

Podemos arreglar los bordes usando los parámetros `xlim` y `ylim`, que aprendiste al estudiar diagramas de caja:

```
df.plot(xlim=(x_min, x_max), ylim=(y_min, y_max))
```

Para mostrar líneas de cuadrícula, establece el parámetro `grid` en `True`:

```
df.plot(grid=True)
```

Uso de varias opciones juntas

```
data.plot(
    x='days',
    y='revenue',
    style='o-',
    xlim=(0, 30),
    ylim=(0, 120000),
    figsize=(4, 5),
    title='A vs B',
    grid=True)
```

Guardar figuras

La función `savefig()` se puede utilizar para guardar la última figura.

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

df = pd.DataFrame({'a':[2, 3, 4, 5], 'b':[4, 9, 16, 25]})

df['b'].plot()
plt.savefig('myplot.png', dpi=300)
```

Graficar con matplotlib y pandas

Probablemente, la combinación más óptima:

- crear un objeto Figure y un objeto Axes con `matplotlib`;
- trazar el núcleo del gráfico con pandas;
- ajustar el gráfico llamando a los métodos de Axes y/o Figure.

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

df = pd.DataFrame({'a':[2, 3, 4, 5], 'b':[4, 9, 16, 25]})

fig, ax = plt.subplots(figsize=(12, 8))

df.plot(x='days', y='revenue', style='o-', ax=ax)

ax.set_xlim(0, 30)
ax.set_ylim(0, 120000)

ax.set_title('A vs B')
ax.grid(True)

fig.savefig('myplot.png', dpi=300)
```

Consulta muchos métodos para el objeto Axes aquí.

Gráficos de barras

A veces usamos gráficos de barras para representar gráficamente datos cuantitativos. Cada barra en dicho gráfico corresponde a un valor: cuanto mayor sea el valor, mayor será la barra. Las diferencias entre los valores son claramente visibles.

En pandas, los gráficos se trazan con el método `plot()`. Se pasan varios tipos de gráficos en el parámetro `kind`. Para trazar un gráfico de barras, indica `'bar'`.

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

df = pd.read_csv('/datasets/west_coast_pop.csv')

df.plot(x='year', kind='bar')
```

Histogramas

Diferencias clave entre gráficos de barras e histogramas:

- Los gráficos de barras se utilizan para comparar valores de variables discretas; los histogramas se utilizan para trazar distribuciones de variables *numéricas continuas*.

- El orden de las barras en los gráficos de barras se puede modificar por estilo o comunicación; el orden de las barras en los histogramas no se puede cambiar.

En un histograma, el eje X representa la variable y tiene el rango de valores que puede tomar la variable. El eje Y representa la frecuencia con la que ocurre cada valor.

En pandas, los histogramas se trazan con el método `hist()`. Se puede aplicar a una lista o una columna de un DataFrame, en cuyo caso la columna se pasa como argumento. El método `hist()` encuentra los valores más altos y más bajos en un conjunto y divide el rango resultante en intervalos o contenedores igualmente espaciados. Luego, el método encuentra la cantidad de valores dentro de cada contenedor y la representa en el gráfico. El número predeterminado de contenedores es 10, que se puede cambiar utilizando el parámetro `bins=`.

Otra forma de trazar un histograma es llamar al método `plot()` con el parámetro `kind='hist'`.

Visualización de un histograma con 16 contenedores y valores mínimos y máximos (min_value y max_value):

```
COPYPYTHON
import matplotlib.pyplot as plt
import pandas as pd

df['age'].hist(bins=16, range=(0, 120)
```

Se pueden mostrar varios histogramas en un solo gráfico.

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

df = pd.read_csv('/datasets/height_weight.csv')

df[df['male'] == 1]['height'].plot(kind='hist', bins=30)
# Incluye un valor alfa para que podamos ver por completo los dos histogramas
df[df['male'] == 0]['height'].plot(kind='hist', bins=30, alpha=0.8)

plt.legend(['Male', 'Female'])
```

Gráficos de líneas

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

df = pd.read_csv('sbux.csv')

df.plot(x='date', y='open')
```

Diagramas de dispersión

Puede haber tantos puntos que muchos de ellos se superponen, por lo que no podemos tener una buena idea de la densidad de puntos en los gráficos anteriores. Se puede arreglar usando el parámetro `alpha=`. Este parámetro controla la transparencia de los puntos y puede tomar cualquier valor entre 0 (transparencia total) y 1 (sin transparencia). De forma predeterminada, se establece en 1 y no hay transparencia.

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

df = pd.read_csv('/datasets/height_weight.csv')

df.plot(x='height', y='weight', kind='scatter', alpha=0.36)
```

Matrices de dispersión

Podemos construir diagramas de dispersión para cada posible par de parámetros. Este conjunto de diagramas por pares se denomina matriz de dispersión.

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

df = pd.read_csv('/datasets/height_weight.csv')

pd.plotting.scatter_matrix(df, figsize=(9, 9))
```