Caso de uso:
¿Sobre qué temas se editan libros en España?
Una visualización de las materias más publicadas a lo largo del siglo XX

En este documento se detalla el proceso de análisis y visualización de datos recogida en esta entrada del Blog del portal de datos abiertos y reutilizables de la BNE: https://datosabiertos.bne.es/blog/visualizacion-temas-sxx

Nos proponemos explorar en esta ocasión, trabajando con estos conjuntos de datos abiertos, cómo han evolucionado los temas de interés en los libros que se publican en España, trabajando con los datos del catálogo bibliográfico de la BNE que recoge las monografías modernas (los libros publicados desde el año 1831).

1. Encuentra el dataset

Desde nuestro repositorio de datos, accede a https://datosabiertos.bne.es/dataset/catalogo-bibliografico-monografias-modernas o emplea el buscador desde la página principal .

Haz clic sobre el formato que prefieras y después en Descargar.

2. Carga tu dataset

Elige el entorno que prefieras para trabajar con el dataset; Anaconda, Jupyter Notebook, Google Colab.., Para realizar nuestro gráfico hemos trabajado desde Google Colab, ya que no requiere de instalación previa y puedes acceder a ello desde Google drive.

Este gráfico se ha realizado mediante el lenguaje de código abierto Python, y se han usado las librerías *pandas* y *numpy* para el tratamiento de los datos y *plotly* para la realización del gráfico.

3. Tratamiento previo de los datos

Antes de poder trabajar con los datos es necesario importar las librerías con las que vamos a trabajar, para ello, ejecuta las siguientes líneas de código en tu terminal:

```
import pandas as pd
import numpy as np
```

Es una práctica recomendable estudiar el dataset y ver qué estructura tienen las variables antes de hacer alguna transformación. Puedes hacer esto mediante la siguiente línea de código;

```
monomodern.info()
```

También conviene comprobar que se ha importado bien la base de datos y qué aspecto tienen los registros. La siguiente línea de código te permite ver los primeros registros del dataset:

```
display(monomodern.head())
```

La base de datos de monografías modernas consta de 1.048.575 registros, y 33 columnas. Cómo nos interesa trabajar sólo con los libros publicados en el siglo XX, hacemos un filtrado de los registros que cumplen dicha condición.

4. Obtención de base de datos que contenga sólo los registros de los libros publicados en el siglo XX

Para poder obtener esta base de datos contamos con dos alternativas: aplicar un filtro en la columna ‘siglo’ y filtrar los registros del siglo XX, o hacer el filtro en la columna

‘fecha_de_publicacion’ para quedarnos con los registros publicados entre el año 1900 y el año 1999.

Para el primer caso, hacemos un subconjunto de los registros de la base de datos que tengan asignado el siglo XX;

```
monomodern_XX = monomodern[monomodern['siglo'] == 'XX']
```

Para el segundo caso, es necesario transformar la variable ‘fecha_de_publicacion’ a formato fecha, ya que para poder aplicar el filtro debe tener ese formato;

```
monomodern['fecha_de_publicacion'] = pd.to_datetime(monomodern['fecha_de_publicacion'], errors='coerce')
```

Una vez cambiado el formato aplicamos el filtro;

```
monomodern_XX = monomodern[(monomodern['fecha_de_publicacion'] >= '1900-01-01') & (monomodern['fecha_de_publicacion'] <= '2000-12-31')]
```

Aunque este método requiera de hacer una transformación inicial, también te permitirá filtrar entre años concretos si lo deseas.

5. Tratamiento de los registros vacíos

Como parte del tratamiento de los datos, identificamos y tratamos los registros vacíos dentro de la columna tema, mediante la siguiente línea de código:

```
print(monomodern_XX['tema'].isna().sum())
```

En la columna ‘tema’ hay un total de 365.842 registros vacíos. En este caso, vamos a eliminarlos;

```
monomodern_XX = monomodern_XX.dropna(subset=['tema'])
```

6. Transformación de la variable ‘tema’

Como hemos visto en la exploración inicial de la base de datos, la columna ‘tema’, es de tipo ‘object’ y contiene uno o más términos, separados por un guión entre ellos, de la siguiente manera;

Tagalos (Pueblo filipino) - Religión - Cristianismo - Filipinas - Historia|2 embne

Previo a poder realizar cualquier procesamiento de texto, es necesario que la variable que vamos a modificar sea tipo *string* por lo que la transformamos;

```
monomodern_XX.loc[:,'tema'] = monomodern_XX['tema'].astype('str')
```

Algunos de los registros cuentan con información relacionada con los encabezamientos del registro bibliográfico que no nos son de interés. Ésta información va detrás del carácter "|"

```
monomodern_XX.loc[:,'tema'] = monomodern_XX['tema'].str.split('|').str[0]
```

En un principio se decidió separar cada uno de los temas en filas distintas y duplicar los registros que fueran necesarios.

```
monomodern_XX_exp = monomodern_XX.assign(tema= monomodern_XX['tema'].str.split('-')).explode('tema').reset_index(drop=True)
```

Miramos ahora cuáles son los temas más populares de las obras publicadas durante el siglo XX

```
monomodern_XX_exp['tema'].value_counts()
```

Lo que nos da como resultado;

tema	count
Historia	35959
Historia	28207
España	26861
España	26740
Libros escolares	21378
...	...
Raguenel, Tiphaine	1
Rosado, Antonio	1
Prohibiciones de disponer	1
Santos Rojas, Sebastián	1
Fundación Centro Etnográfico "Joaquín Díaz"	1

c 127188 rows × 1 columns

Como vemos en los resultados, algunos de los temas están siendo reconocidos como distintos aunque tengan el mismo contenido. Para evitar que esto ocurra hacemos un procesamiento del texto, quitando los espacios;

```
monomodern_XX_exp['tema'] = monomodern_XX_exp['tema'].str.strip()
```

tema	count
España	70201
Historia	69833
Libros escolares	25573
Congresos y asambleas	20463
S.XX	16517
...	...
Osorio, Pepa	1
Lehnert & Landrock	1
Thomas, E. Donnall	1
Ros Marbà, Antoni	1
Bacas, Darío	1

82601 rows × 1 columns

Como se puede ver en los resultados, “Historia” y “España” tenían una presencia mucho mayor que el resto, lo que dificulta la visualización, ya que puede que estén presentes como tema secundario en muchos libros, por lo que optamos por quedarnos con el tema principal de cada libro, exceptuando los registros en los que el tema principal es España, en cuyo caso se mantiene el primer y segundo término. Para llevar a cabo esto se crea la siguiente función;

```
def procesar_tema(tema):
    #Eliminar el texto después del símbolo "|"
    tema = tema.split('|')[0].strip()

    #Separar los términos por el guion " - "
    terminos = [t.strip() for t in tema.split(' - ')]

    #Tratamiento específico si el primer término es "España" o "Historia"
    if terminos[0] in ["España", "Historia"]:
        # Nos quedamos con los dos primeros términos
        return ' - '.join(terminos[:2])
    else:
        # Para los demás casos, solo quedarnos con el primer término
        return terminos[0]

# Aplicar la función a la columna 'tema'
monomodern_XX.loc[:, 'tema'] = monomodern_XX['tema'].apply(procesar_tema)
```

Comprobamos ahora el conteo de los temas;

```
monomodern_XX['tema'].value_counts()
```

tema	count
Lengua española	10858
Matemáticas	7113
España - Historia	5095
Lengua inglesa	4931
Lectura	3760
...	...
Vergilius Vaticanus	1
Oak House School (Barcelona)	1
Poesías griegas helenísticas	1
Casa Asil de Sant Andreu de Palomar	1
Bacas, Darío	1

64868 rows × 1 columns

7. Tratamiento de la variable década

Ya que queremos ver cómo varían los temas más populares del siglo XX a lo largo de las décadas, vamos a realizar también el tratamiento necesario para esta columna.

Primero identificamos los NaN de la columna década de la misma manera que lo hemos hecho con la columna tema;

```
print(monomodern_XX['decada'].isna().sum())
```

Existen 184 registros que no tienen una década asociada dentro de los registros que hemos filtrado en los pasos anteriores. Los descartamos:

```
monomodern_XX = monomodern_XX.dropna(subset=['decada'])
```

Como pudimos ver en el análisis exploratorio, 'decada' es una variable registrada como *float*, un tipo de variable empleada para los números decimales. Como nuestra variable sólo tiene números enteros, la transformamos a integer;

```
monomodern_XX['decada'] = monomodern_XX['decada'].astype(int)
```

8. Creación del gráfico

Una vez hemos hecho la limpieza y el procesamiento necesario de las variables, podemos comenzar a crear nuestro gráfico. Hemos optado por crear un mapa de árbol o *treemap.* Este gráfico suele emplearse con datos jerarquizados, pero también es una manera de representar visualmente el valor de cada una de las categorías.

La biblioteca *plotly* de Python te permite crear este tipo de gráficos. Además, como queremos ver cómo cambian los temas más populares a lo largo de las décadas, podemos aprovechar la funcionalidad que ofrece *plotly* de agregar sliders y construir un mapa de árbol interactivo.

9. Importar librerías

Al igual que en el paso anterior, importa la librería con la que vas a trabajar:

```
import plotly.graph_objects as go
```

10. Creación de los *sliders*

Previo a la realización del gráfico, tenemos que crear los sliders que agregar al gráfico.

Primero creamos una variable con los valores únicos y ordenados de las décadas:

```
decadas = sorted(monomodern_XX['decada'].unique())
```

Y creamos un diccionario vacío, en el cuál almacenaremos los nombres de las décadas con el conteo de registros que tiene asignado;

```
pasos = {}
```

Mediante un bucle *for*, conseguimos la cantidad de registros que tiene cada década y completamos el diccionario creado en el paso anterior:

```
for decada in decadas:
    monomodern_XX_decada = monomodern_XX[monomodern_XX['decada'] == decada]
    total_publicaciones_decada = len(monomodern_XX_decada)
    conteo_categorias = monomodern_XX_decada['tema'].value_counts().head(20)
    pasos[decada] = conteo_categorias
```

11. Generar gráfico

Primero, necesitamos crear la figura para el gráfico inicial, donde incluimos el mapa de árbol:

```
fig = go.Figure()

decada_inicial = decadas[0]
conteo_inicial = pasos[decada_inicial]

fig.add_trace(go.Treemap(
   labels=conteo_inicial.index,
   parents=[""] * len(conteo_inicial),
   values=conteo_inicial.values,
   name=f'Década {decada_inicial}',
   textinfo="label+value"
))
```

La función *Treemap* de *plotly*, que te permite crear mapas de árbol, viene asociada con algunos parámetros. Los que hemos empleado han sido;

- Labels: indica las categorías que se van a mostrar
- Parents: indica la jerarquía. En nuestro caso queda vacío porque no se tiene en cuenta
- Values: indica el valor de cada categoría
- Name: el nombre que se le da a cada uno de los frames que se muestran.
- Textifo: para añadir texto explicativo dentro de la representación gráfica. En este caso mostramos el nombre de la categoría y el conteo (label+value)

Puedes consultar los parámetros de la función en el siguiente enlace; https://plotly.com/python-api-reference/generated/plotly.graph_objects.Treemap.html

Una vez hemos creado el gráfico inicial, creamos uno para cada década. Primero creamos la lista vacía *slider_steps*, y mediante un bucle *for* llenamos la lista para cada una de las décadas:

```
sliders_steps = []

for decada in decadas:
   conteo = pasos[decada]

   # Creamos un slider para cada década
   sliders_steps.append({
      'label': f'{decada}',
```

```
      'method': 'update',
      'args': [{
        'labels': [conteo.index.tolist()],
        'values': [conteo.values.tolist()],
      }]
    })
```

Una vez creados, actualizamos el gráfico con el slider

```
fig.update_layout(
    sliders=[{
      'active': 0,
      'currentvalue': {'prefix': 'Década: '},
      'pad': {'t': 50},
      'steps': sliders_steps
    }]
)
```

De manera opcional, puedes personalizar el gráfico cambiando el nombre o modificando los colores.

```
fig.update_layout(
    title="20 categorías más populares de cada década",
    treemapcolorway = ["royalblue", "lightyellow"],
)
```

Finalmente, ejecutamos esta línea para mostrar el gráfico;

```
fig.show()
```

12. Personalización del gráfico

Para generar nuestro gráfico final modificamos algunos de los parámetros del *treemap*, como los colores, título, tipo y tamaño de letra…

Para modificar el color, *plotly* ofrece tres posibilidades:

- La presentada en el apartado anterior; dentro de la función *update_layout*, utilizando *treemapcolorway*

Una nota sobre herramientas de visualización

Existen muchas herramientas que te permiten realizar visualizaciones con datos de manera gratuita.

Algunos de ellos requieren de ciertos conocimientos de programación, aunque existe mucha documentación disponible online ;

- Python: además de **Plotly** (https://plotly.com/python/) que te permite hacer gráficos , algunas de las librerías de Python más conocidas que te permiten realizar gráficos son **Matplotlib** (https://matplotlib.org/) y **Seaborn** (https://seaborn.pydata.org/)
- R/RStudio: las librerías más comúnmente usadas son **ggplot2** (https://r-graph-gallery.com/ggplot2-package.html), **lattice** (https://cran.r-project.org/web/packages/lattice/lattice.pdf)**. Plotly** (https://plotly.com/r/) también está disponible en R para crear gráficos interactivos.
- Javascript: la librería d3.js (https://d3js.org/).

También gratuito es **Raw graphs** (https://www.rawgraphs.io/), que está construido sobre d3.js. Permite copiar y pegar tus propios datos o subirlos a la plataforma. No requiere de conocimientos de programación previos, aunque es conveniente que los datos que introduzcas hayan sido previamente procesados.