Tema 4.2 Otras funcionalidades de Pandas:

Tomando como ejemplo el código anterior en donde se trabajó con el DataFrame "datos" veremos algunas funcionalidades mas:

Habiendo corrido el código de la sección anterior, ejecutar en la Terminal de Python o acceder al siguiente link:

https://colab.research.google.com/drive/16z885Vh6NpOVzgCMoYc_435EhVPKemys?usp=sharing#scrollTo=5xdB5K-ltlxg

Algunos ejemplos de funcionalidades:

```
#los siguientes ejemplos devuelven valores de distintos tipos (types) de datos
datos.values
datos.size
datos.index
datos.columns
datos.Fecha
datos["Fecha"]
datos.CaniaAzucar
datos["CaniaAzucar"]
datos.Maiz
datos["Maiz"]
datos[["Fecha","Maiz"]] #Accedo a dos columnas a la vez
datos[1][0:12]
datos.astype("int")
datos.head #(cabeza)
datos.tail #(cola)
datos.T #transpuesta
datos.sort_values
datos.sort_values(by=["Maiz"])
datos.sort_index()
#IMPORTANTE
#ninguno de los ejemplos anteriores modifica el DataFrame
# si queremos actualizar el dataframe datos se hace:
datos = datos.__algo__()
```

Uso de Pandas con matplotlib y numpy para mostrar datos:

Si en Episodio 2 pudiste desarrollar todo el programa, habrás dado una introducción del uso de estas bibliotecas, sino será el momento de empezar desde cero.

Recurso: TUTORIAL (clase) y EJEMPLOS interesantes para uso de matplotlib y numpy

Importar matplotlib:

```
import matplotlib.pyplot as plt
```

Importa matplotlib como plt, quiere decir que haremos uso de la biblioteca matplotlib llamándola plt. Es como ponerle un apodo y podemos elegirlo. Se le acostumbra llamar plt, así es más fácil escribirlo cuando necesitemos utilizarlo después.

Para graficar, plt tiene el método plot():

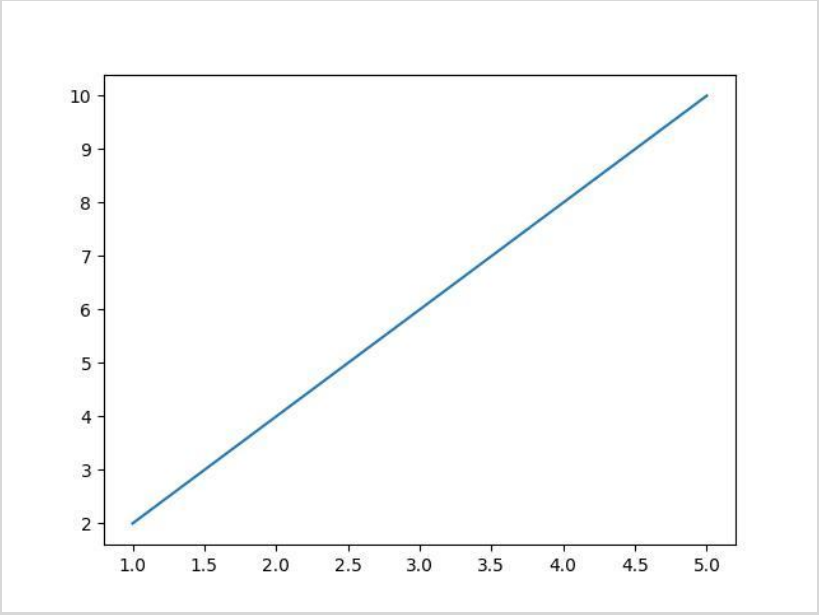
```
plt.plot(lista) #hace una gráfica a partir de los datos que contiene la lista
plt.show() #muestra la gráfica
```

El método plot() de la librería Matplotlib se utiliza para crear gráficos de línea. El método acepta varias entradas, como listas o matrices de datos, y crea una visualización de línea a partir de ellas. El gráfico se puede personalizar aún más agregando etiquetas de eje, títulos y leyendas.

Por ejemplo, si tienes dos listas de datos 'x' e 'y', puedes crear un gráfico de línea simple utilizando el método plot() de la siguiente manera:

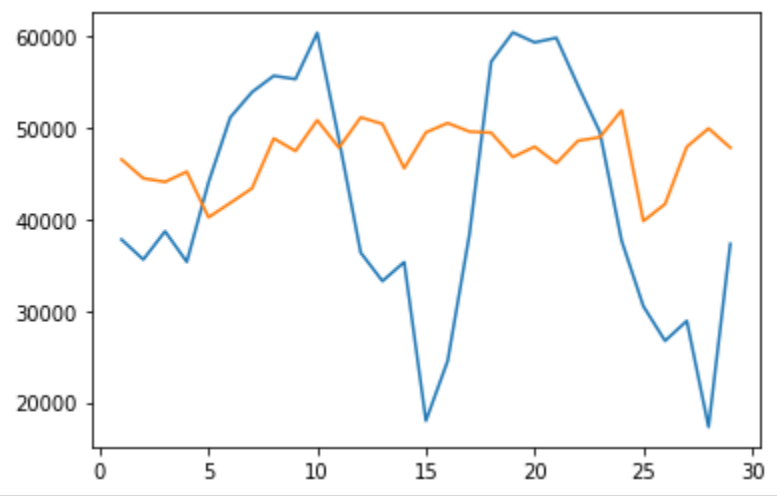
```
import matplotlib.pyplot as plt


x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]


plt.plot(x, y)
plt.show()
```

Show:



Para el ejemplo del dataset de la producción de bioetanol:

```
import pandas as pd

import matplotlib.pyplot as plt

url="https://datos.magyp.gob.ar/dataset/18e03919-2828-4e21-b7ab-fe340b411ea/resource/edeffcb4-1ed7-4eb0-ab7e-3e85b8854afb/download/serie_de_tiempo_produccion_por_insumo_bioetanol.csv"

datos = pd.read_csv(url,header=0, names=['Fecha','CaniaAzucar','Maiz'])

print(datos.sum()) #hace sumatoria de todas las columnas (las fechas no las puede sumar porque siguen siendo str)

print("\n")

print(f"Total de bioetanol producido con caña de azucar: {datos['CaniaAzucar'].sum()} m3")

print(f"Total de bioetanol producido con maíz: {datos['Maiz'].sum()} m3")

plt.plot(datos["CaniaAzucar"])

plt.plot(datos["Maiz"])

plt.show()
```

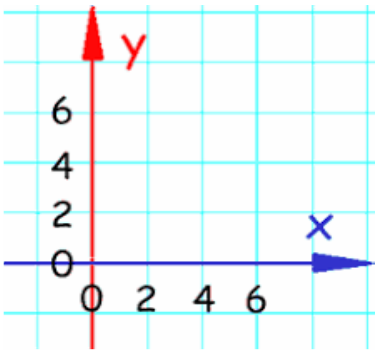


Si bien obtuvimos resultados, la gráfica está incompleta, falta colocarle las referencias, nombres a los ejes y escalar si es necesario.

Colocarle nombres a los ejes curvas:

```
plt.plot(datos["CaniaAzucar"])
plt.plot(datos["Maiz"])
```

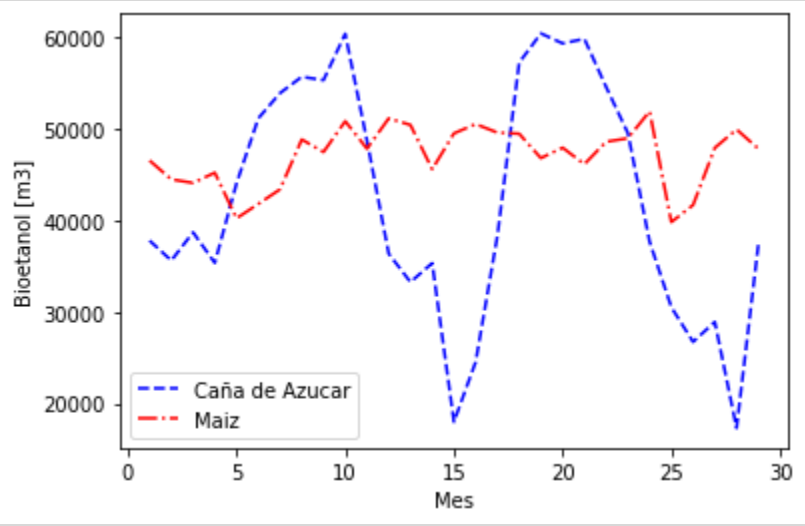
```
#nombre de los ejes
plt.ylabel('Bioetanol [m3]')


plt.xlabel('Mes')
plt.legend()
plt.show()
```



Cambiamos color y le colocamos una leyenda de descripción a cada curva:

```
plt.plot(datos["CaniaAzucar"], 'g--', label='Caña de Azucar')
plt.plot(datos["Maiz"], 'r-.', label='Maiz')
plt.ylabel('Bioetanol [m3]')
plt.xlabel('Mes')
plt.legend()
plt.show()
```

Resultados:

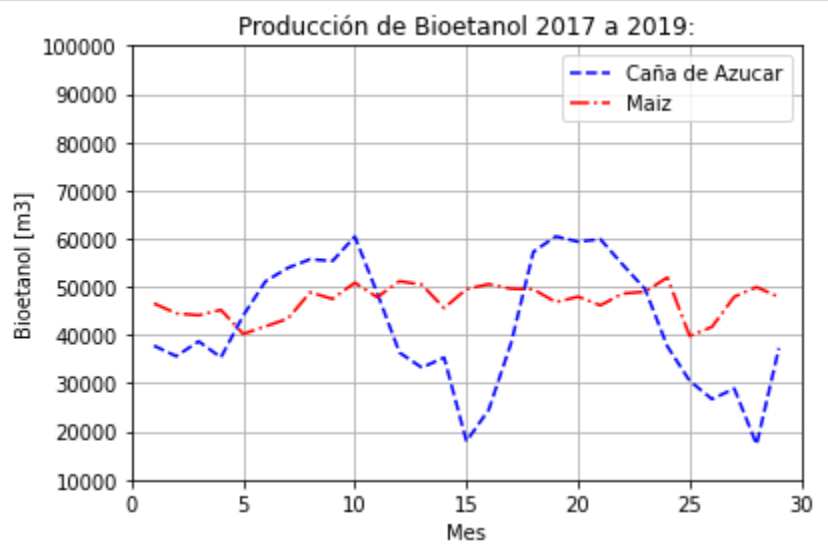


Lista de funcionalidades:

- plt.figure(figsize = (w, h)) elegimos el tamaño de la figura.
- plt.subplots(nrows, ncols) creamos un conjunto de subplots para ser usados más adelante.
- plt.xlabel('xlabel') y plt.ylabel('ylabel') colocamos nombre a los ejes.
- plt.axis([xmin, xmax, ymin, ymax])establecemos los límites de los ejes.
- plt.title('title') mostramos un título sobre el gráfico.
- plt.grid() activamos grilla cuadriculada.

```
import pandas as pd

import matplotlib.pyplot as plt

url="https://datos.magyp.gob.ar/dataset/18e03919-2828-4e21-b7ab-fe340b411ea/resource/edeffcb4-1ed7-4eb0-ab7e-3e85b8854afb/download/serie_de_tiempo_produccion_por_insumo_bioetanol.csv"

datos = pd.read_csv(url,header=0, names=['Fecha','CaniaAzucar','Maiz'])

print(datos.sum()) #hace sumatoria de todas las columnasc(las fechas nolas puede sumar porque siguen siendo str)

print("\n")

print(f"Total de bioetanol producido con caña de azucar:

{datos['CaniaAzucar'].sum()} m3")

print(f"Total de bioetanol producido con maíz: {datos['Maiz'].sum()} m3")

plt.plot(datos["CaniaAzucar"], 'b--', label='Caña de Azucar')

plt.plot(datos["Maiz"], 'r-.', label='Maiz')

plt.ylabel('Bioetanol [m3]')

plt.xlabel('Mes')

plt.legend()

plt.axis([0, 30, 10000, 100000])

plt.grid()

plt.show()
```

Resultado:



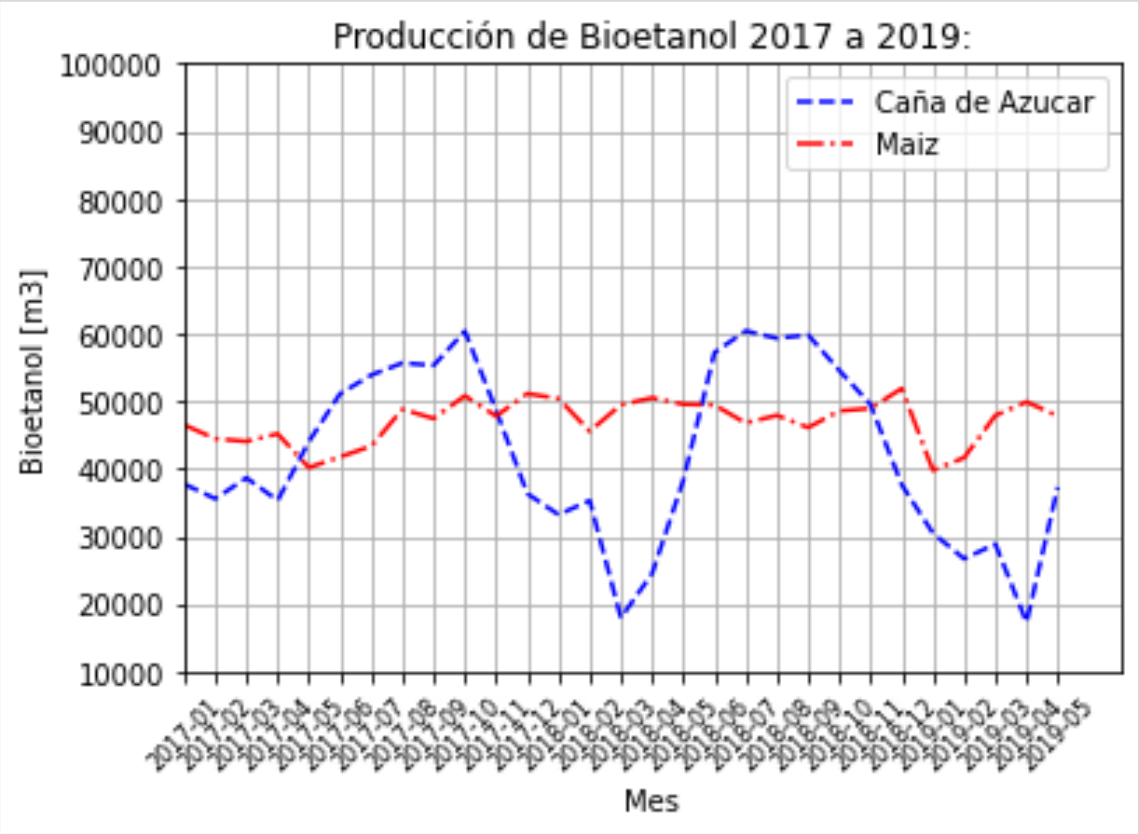
Se logran observar resultados que nos permiten analizar cómo fue la producción del bioetanol durante los años 2017 a 2019. Si bien los datos pueden apreciarse bien, habría que expresar de mejor manera los meses, ya que van del 1 al 29 y podríamos especificarlo por el nombre y año del mes.

Imprimimos en función de la fecha, en X van las fechas en Y van los valores de producción.

Graficamos las fechas en X. Usamos el método .xticks() para establecer la fecha en un ángulo de 45° y cambiarle la fuente para que se aprecie mejor.

```
plt.plot(datos["Fecha"],datos["CaniaAzucar"], 'b--', label='Caña de Azucar')
plt.plot(datos["Fecha"],datos["Maiz"], 'r-.', label='Maiz')
plt.ylabel('Bioetanol [m3]')
plt.xlabel('Mes')
plt.xticks(fontsize=8,rotation=45)
```

Resultado:



Ahora vamos a calcular la producción anual de bioetanol con caña de azúcar y maíz (para los 3 años). Elaboramos un gráfico de barras (usar plt.bar() en vez de plt.plot() y comparar ambos resultados).

Para ello, primero tendremos que manipular los datos del DataFrame datos para obtener la producción anual. Podemos hacerlo usando el método groupby() de Pandas para agrupar los datos por año y luego sumar las producciones mensuales de bioetanol para cada insumo. Aquí te muestro cómo hacerlo:

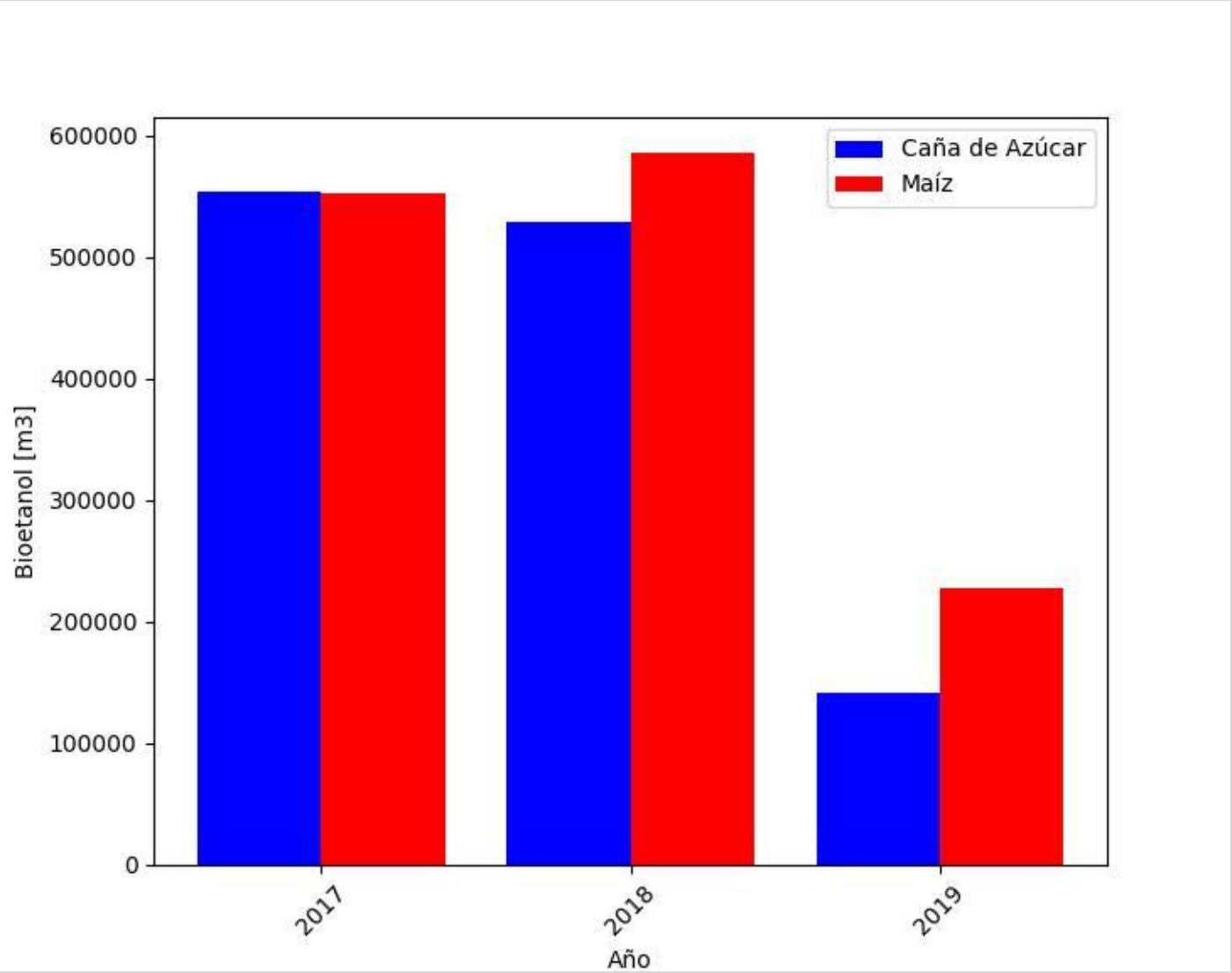
```
# Agregamos una columna 'Año' al DataFrame
datos['Año'] = pd.DatetimeIndex(datos['Fecha']).year


# Agrupamos los datos por año y sumamos las producciones mensuales para cada insumo
produccion_anual = datos.groupby('Año').sum()[['CaniaAzucar', 'Maiz']]


# Mostramos la producción anual para cada insumo
print('Producción anual de bioetanol:')
print(produccion_anual)
```

La salida del código anterior mostrará la producción anual de bioetanol con caña de azúcar y maíz para cada uno de los tres años del conjunto de datos. Para elaborar un gráfico de barras de la producción anual, puedes utilizar la función plt.bar() de Matplotlib de la siguiente manera:

```
# Creamos una figura y un eje de barras
fig, ax = plt.subplots()

# Creamos las barras para la producción anual de cada insumo
ax.bar(produccion_anual.index, produccion_anual['CaniaAzucar'], width=0.4, color='b', label='Caña de Azúcar')
ax.bar(produccion_anual.index + 0.4, produccion_anual['Maiz'], width=0.4, color='r', label='Maíz')

# Configuramos el eje X
ax.set_xticks(produccion_anual.index + 0.2)
ax.set_xticklabels(produccion_anual.index)
ax.tick_params(axis='x', labelrotation=45)

# Configuramos la leyenda y las etiquetas de los ejes
ax.legend()
ax.set_xlabel('Año')
ax.set_ylabel('Bioetanol [m3]')

# Mostramos el gráfico de barras
plt.show()
```

El código anterior creará un gráfico de barras que muestra la producción anual de bioetanol con caña de azúcar y maíz para cada uno de los tres años. Cada barra representa la producción anual de bioetanol para un insumo determinado y está etiquetada con el año correspondiente. La función plt.bar() crea las barras y la función ax.set_xticklabels() configura las etiquetas del eje X para mostrar los años correspondientes. La función ax.tick_params() se utiliza para configurar la rotación de las etiquetas del eje X. La leyenda y las etiquetas de los ejes se configuran utilizando las funciones ax.legend(), ax.set_xlabel(), y ax.set_ylabel().

La función subplots() de la librería Matplotlib se utiliza para crear múltiples gráficos en una sola figura. Al llamar a esta función, se crean dos objetos: una figura y un arreglo de subtramas. El arreglo de subtramas puede ser utilizado para crear y organizar los gráficos. La función acepta varios parámetros para controlar el diseño y los detalles de los gráficos, como el número de filas y columnas de subtramas, los espacios entre ellas, entre otros.



Separar fecha en días, meses y años con Pandas:

Pandas cuenta con una funcionalidad para separar la información de tiempo, separa años de meses y días u horas de minutos y segundos.

El Datetime Index.time de Pandas genera un objeto Index que contiene los valores de tiempo presentes en cada una de las entradas del objeto DatetimeIndex que se le pasó. En otras palabras, separa los valores de tiempo de la columna que le pasemos como entrada y devuelve una columna de salida.

IMPORTANDO: import datetime

Ejemplo:

```
datos['Anios'] = pd.DatetimeIndex(datos['Fecha']).year
datos['Mes'] = pd.DatetimeIndex(datos['Fecha']).month
print(datos) #ahora tiene dos columnas nuevas al final
```

Resultado:

```
In [167]: datos
Out[167]:
       Fecha  CaniaAzucar   Maiz  Anios  Mes
1    2017-01        37794  46551   2017    1
2    2017-02        35607  44489   2017    2
3    2017-03        38696  44100   2017    3
4    2017-04        35355  45218   2017    4
5    2017-05        44053  40222   2017    5
6    2017-06        51159  41800   2017    6
7    2017-07        53924  43396   2017    7
8    2017-08        55711  48852   2017    8
9    2017-09        55345  47483   2017    9
10   2017-10        60415  50850   2017   10
11   2017-11        48724  47841   2017   11
12   2017-12        36361  51160   2017   12
13   2018-01        33251  50443   2018    1
14   2018-02        35328  45593   2018    2
15   2018-03        17968  49519   2018    3
16   2018-04        24522  50537   2018    4
17   2018-05        38336  49606   2018    5
18   2018-06        57254  49484   2018    6
19   2018-07        60456  46815   2018    7
```

Tema 4.3.1 Comparando datasets:

Al poder ordenar la información en datasets es posible poder comparar u obtener conclusiones a partir de hacer el uso de dos datasets o más a la vez. Entonces si tengo una base de datos con la producción de melones en Mendoza en función del tiempo en meses y otro dataset de lluvias en Mendoza en función del tiempo en meses, podría obtener la correlación que existe entre la lluvia y la producción de melones e inferir nuevos conocimientos.

Se procede a realizar la comparación entre la producción de Bioetanol y la venta de Bioetanol a partir de los siguientes datasets:

Producción de bioetanol: ProduccionBioetanol2017-2019

Venta de bioetanol: VentaBioetanol2017-2019

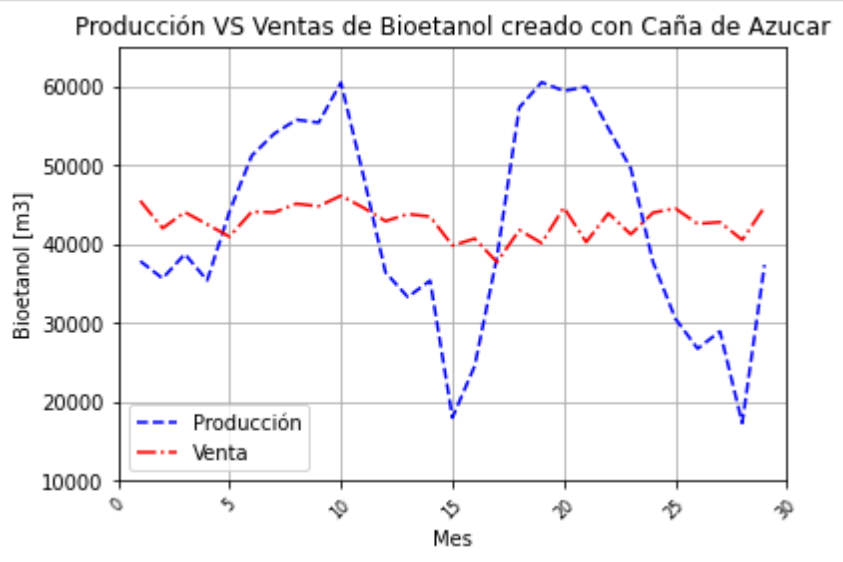
El siguiente código realiza la importación de los dos datasets mostrados anteriormente, se procede a filtrar y eliminar la información que no sirve (acondicionamiento del DataFrame) y luego se procede comparar los dos dataset para obtener nuevas conclusiones/resultados. Ejecutar el siguiente código:

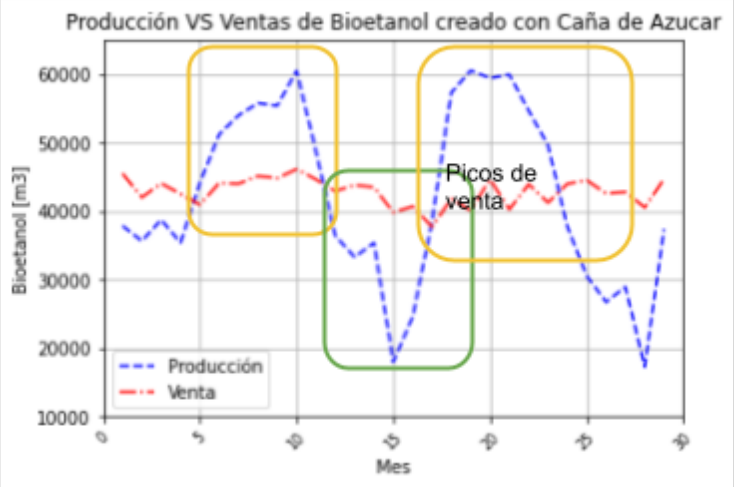
```
import pandas as pd

import datetime

import matplotlib.pyplot as plt

import matplotlib.dates as mdates

urlProducc="https://datos.magyp.gob.ar/dataset/18e03919-2828-4e21-b7ab-
4fe340b411ea/resource/edeffcb4-1ed7-4eb0-ab7e-
3e85b8854afb/download/serie_de_tiempo_produccion_por_insumo_bioetanol.csv"

urlVentas="https://datos.magyp.gob.ar/dataset/b3819c0c-ce32-4c2b-884e-
984d3e10bdc0/resource/a49ccbda-c6f4-4fe9-8ebf-8c1c664f28dc/download/ventas-internas-de-
bioetanol-por-tipo-de-insumo-.csv"

datos = pd.read_csv(urlProducc,header=0, names=['Fecha','CaniaAzucar','Maiz'])

ventasBioetanol = pd.read_csv(urlVentas,header=0,

names=['id_pais','nom_pais','anio','mes','cod_unimed','nom_unimed','CaniaAzucar','Maiz','total'])

# ORDENAMOS DATOS DE LA PRODUCCION DE BIOETANOL

datos['Anios'] = pd.DatetimeIndex(datos['Fecha']).year

datos['Mes'] = pd.DatetimeIndex(datos['Fecha']).month

# ORDENAMOS DATOS DE LA VENTA DE BIOETANOL

#Hay columnas que no son de interés, puedo dejarlas o eliminarlas.

#NO es necesario eliminarlas pero te enseñamos cómo
```

```
#axis=0 es para filas - axis = 1 es para columnas
#.drop('NombreColumna',axis=1)
#drop('NumeroFila',axis=0)
ventasBioetanol = ventasBioetanol.drop('id_pais',axis=1) #elimino columna id_pais
ventasBioetanol = ventasBioetanol.drop('nom_pais',axis=1) #elimino columna nombre_pais
ventasBioetanol = ventasBioetanol.drop('cod_unimed',axis=1) #elimino columna cod_unimed
ventasBioetanol = ventasBioetanol.drop('nom_unimed',axis=1) #elimino columna nom_unimed
plt.title('Producción VS Ventas de Bioetanol creado con Caña de Azucar')
plt.plot(datos["CaniaAzucar"], 'b--', label='Producción')
plt.plot(ventasBioetanol["CaniaAzucar"], 'r-.', label='Venta')
plt.ylabel('Bioetanol [m3]')
plt.xlabel('Mes')
plt.xticks(fontsize=8,rotation=45)
plt.legend() #muestra leyendas
plt.axis([0, 30, 10000, 65000])
plt.grid()
plt.show()
```

Resultados:

Es importante tener en cuenta la fecha de los datos, así comparamos datos de la misma fecha. Debemos prestar atención a la cantidad de filas y a cómo están ordenados los datos para poder realizar la comparación, en este caso los datos estaban ordenados de la misma manera y en el mismo rango de tiempo.





En amarillo se logra ver que al mismo tiempo que aumentan la producción de bioetanol con caña de azúcar, aumenta la venta de bioetanol y se logran ver picos de venta.

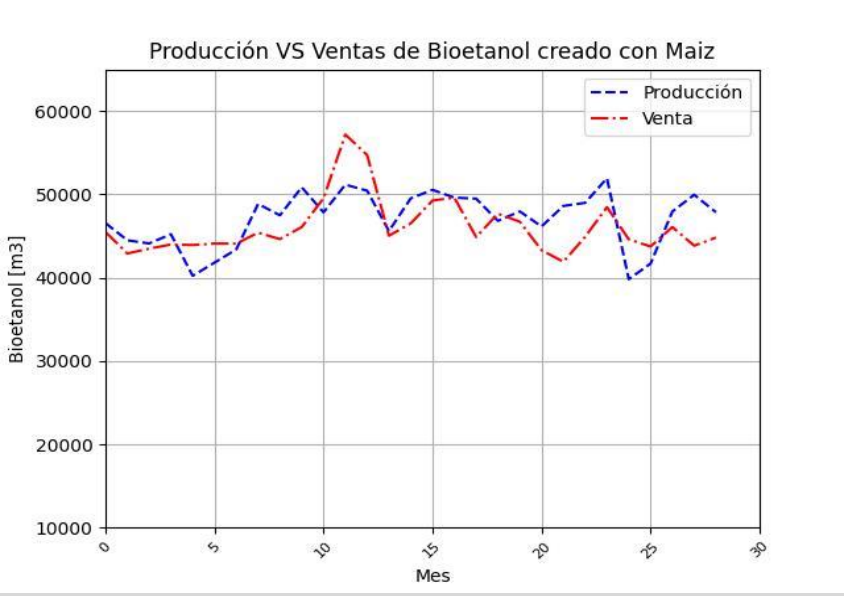
En verde se observa que al disminuir la producción de bioetanol las ventas caen hasta un mes después, luego se ve una recuperación.

Actividad: Realizar el análisis de la producción y venta de maíz destinado al bioetanol.

```
# Comparando datasets: Bioetanol con maíz: Producción y ventas


import pandas as pd

import datetime

import matplotlib.pyplot as plt

import matplotlib.dates as mdates

urlProducc="https://datos.magyp.gob.ar/dataset/18e03919-2828-4e21-b7ab-4fe340b411ea/resource/edeffcb4-1ed7-4eb0-ab7e-3e85b8854afb/download/serie_de_tiempo_produccion_por_insumo_bioetanol.csv"

urlVentas="https://datos.magyp.gob.ar/dataset/b3819c0c-ce32-4c2b-884e-984d3e10bdc0/resource/a49ccbda-c6f4-4fe9-8ebf-8c1c664f28dc/download/ventas-internas-de-bioetanol-por-tipo-de-insumo-.csv"

datos = pd.read_csv(urlProducc,header=0, names=['Fecha','CaniaAzucar','Maiz'])

ventasBioetanol = pd.read_csv(urlVentas,header=0,names=['id_pais','nom_pais','anio','mes','cod_unimed','nom_unimed','CaniaAzucar','Maiz','total'])

# ORDENAMOS DATOS DE LA PRODUCCION DE BIOETANOL

datos['Anios'] = pd.DatetimeIndex(datos['Fecha']).year

datos['Mes'] = pd.DatetimeIndex(datos['Fecha']).month
```

```
# ORDENAMOS DATOS DE LA VENTA DE BIOETANOL
#Hay columnas que no son de interés, puedo dejarlas o eliminarlas.
#NO es necesario eliminarlas pero te enseñamos cómo
#axis=0 es para filas - axis = 1 es para columnas
#.drop('NombreColumna',axis=1)
#drop('NumeroFila',axis=0)
ventasBioetanol = ventasBioetanol.drop('id_pais',axis=1) #elimino columna id_pais
ventasBioetanol = ventasBioetanol.drop('nom_pais',axis=1) #elimino columna nombre_pais
ventasBioetanol = ventasBioetanol.drop('cod_unimed',axis=1) #elimino columna cod_unimed
ventasBioetanol = ventasBioetanol.drop('nom_unimed',axis=1) #elimino columna nom_unimed
plt.title('Producción VS Ventas de Bioetanol creado con Maiz')
plt.plot(datos["Maiz"], 'b--', label='Producción')
plt.plot(ventasBioetanol["Maiz"], 'r-.', label='Venta')
plt.ylabel('Bioetanol [m3]')
plt.xlabel('Mes')
plt.xticks(fontsize=8,rotation=45)
plt.legend() #muestra leyendas
plt.axis([0, 30, 10000, 65000])
plt.grid()
plt.show()
```

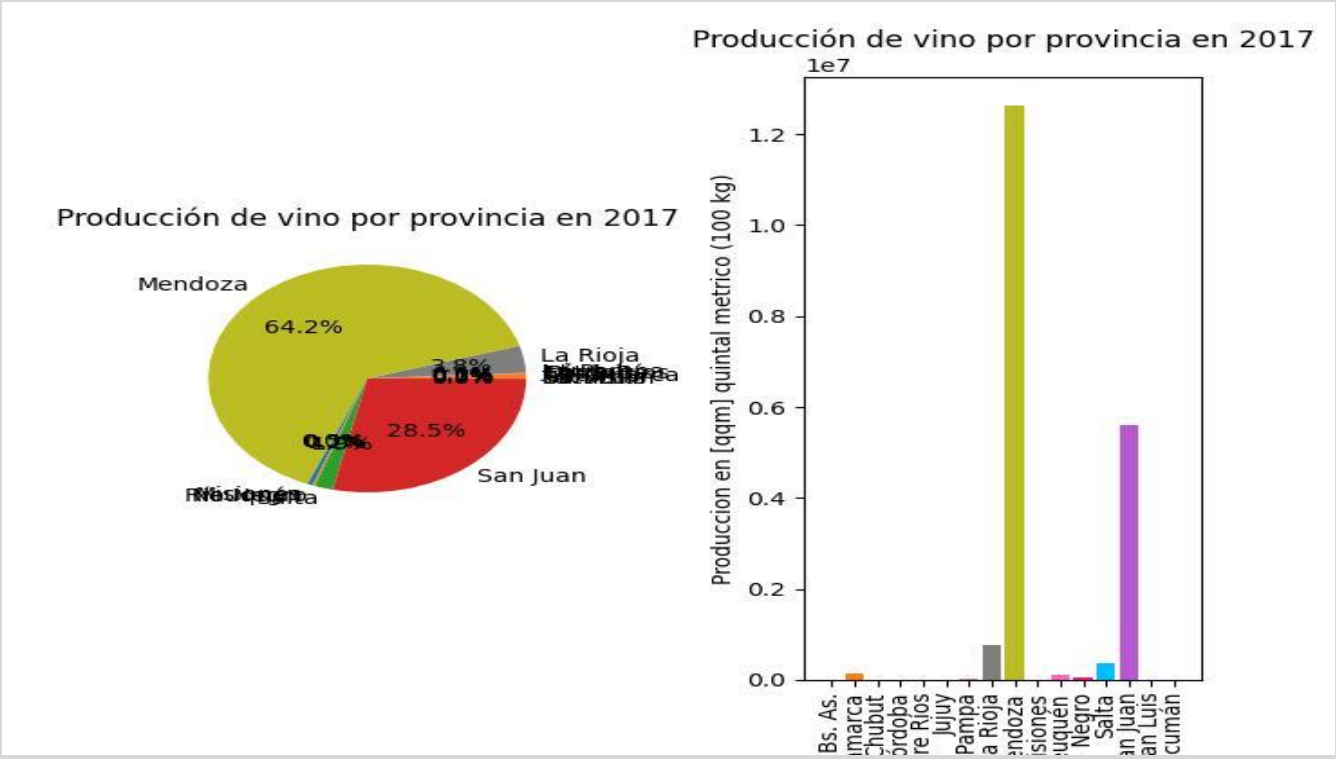
Resultado:



Actividad opcional: Realizar el análisis del siguiente DataSet: INV - Producción de uvas mediante un gráfico de torta por año, comparando el porcentaje de producción de cada provincia con respecto al país. Tener en cuenta que la suma de la producción de todas las provincias (por año) representa el 100% o el total producido en ese año.

```
# Actividad: Producción de vino por provincia (2017):

import pandas as pd

import matplotlib.pyplot as plt

urlProducc="https://datos.magyp.gob.ar/dataset/e217791c-e898-4204-8a62-63273cf81ef7/resource/2a928aef-7f9e-4335-bf3d-a1c5eab49037/download/serie-tiempo-produccion-de-uvas-de-argentina-2012-2017.csv"


datos=pd.read_csv(urlProducc, header=0, names=['Fecha', 'produccion_bs_as', 'produccion_catamarca', 'produccion_chubut', 'produccion_cordoba', 'produccion_entre_rios', 'produccion_jujuy', 'produccion_la_pampa', 'produccion_la_rioja', 'produccion_mendoza', 'produccion_misiones', 'produccion_neuquen', 'produccion_rio_negro', 'produccion_salta', 'produccion_san_juan', 'produccion_san_luis', 'produccion_tucuman'])


datos = datos.fillna(0)


# se filtran los datos del año 2017

datos_2017 = datos.loc[datos['Fecha'].astype(str).str.contains('2017')]


# se crea una lista con los nombres de las provincias

provincias = ['Bs. As.', 'Catamarca', 'Chubut', 'Córdoba', 'Entre Ríos', 'Jujuy', 'La Pampa', 'La Rioja', 'Mendoza', 'Misiones', 'Neuquén', 'Río Negro', 'Salta', 'San Juan', 'San Luis', 'Tucumán']


# se calcula la producción total por provincia en 2017

produccion_por_provincia = datos_2017.sum()[1:].tolist()


# se muestra el gráfico de torta

plt.subplot(1, 2, 1)

plt.title('Producción de vino por provincia en 2017')
```

```
plt.pie(produccion_por_provincia, labels=provincias, autopct='%1.1f%%')


# se muestra el gráfico de barras

plt.subplot(1, 2, 2)

plt.title('Producción de vino por provincia en 2017')

plt.bar(provincias, produccion_por_provincia, color=['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf', '#ff69b4', '#ff1493', '#00bfff', '#ba55d3', '#90ee90', '#ffd700'])

plt.xticks(rotation=90)


# se agrega la columna con los nombres de las provincias

plt.subplots_adjust(wspace=0.6)

plt.subplots_adjust(bottom=0.1, top=0.9)

plt.ylabel('Produccion en [qqm] quintal metrico  (100 kg)')

plt.xlabel('Provincias')

ax2 = plt.subplot


plt.show()
```



En este otro ejemplo mostramos la suma de todos los años:

```
''' # Producción de vino por provincia correspondiente a la suma de todos los años:

import pandas as pd

import matplotlib.pyplot as plt


urlProducc="https://datos.magyp.gob.ar/dataset/e217791c-e898-4204-8a62-63273cf81ef7/resource/2a928aef-7f9e-4335-bf3d-a1c5eab49037/download/serie-tiempo-produccion-de-uvas-de-argentina-2012-2017.csv"


datos =pd.read_csv(urlProducc, header=0, names=['Fecha', 'produccion_bs_as', 'produccion_catamarca', 'produccion_chubut', 'produccion_cordoba', 'produccion_entre_rios', 'produccion_jujuy', 'produccion_la_pampa', 'produccion_la_rioja', 'produccion_mendoza', 'produccion_misiones', 'produccion_neuquen', 'produccion_rio_negro', 'produccion_salta', 'produccion_san_juan', 'produccion_san_luis', 'produccion_tucuman'])


# se eliminan las filas con valores nulos

datos = datos.dropna()


# se crea una lista con los nombres de las provincias

provincias = ['Bs. As.', 'Catamarca', 'Chubut', 'Córdoba', 'Entre Ríos', 'Jujuy', 'La Pampa', 'La Rioja', 'Mendoza', 'Misiones', 'Neuquén', 'Río Negro', 'Salta', 'San Juan', 'San Luis', 'Tucumán']


# se calcula la producción total por provincia

produccion_por_provincia = datos.sum()[1:].tolist()


# se muestra el gráfico de torta

plt.subplot(1, 2, 1)

plt.title('Producción de vino por provincia')

plt.pie(produccion_por_provincia, labels=provincias, autopct='%1.1f%%')
```

```
# se muestra el gráfico de barras

plt.subplot(1, 2, 2)

plt.title('Producción de vino por provincia')

plt.bar(provincias, produccion_por_provincia, color=['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf', '#ff69b4', '#ff1493', '#00bfff', '#ba55d3', '#90ee90', '#ffd700'])

plt.xticks(rotation=90)


# se agrega la columna con los nombres de las provincias

plt.subplots_adjust(wspace=0.6)

plt.subplots_adjust(bottom=0.1, top=0.9)

ax2 = plt.subplot(1, 2, 2)

ax2.set_ylabel('Producción')

ax2.yaxis.set_label_position("right")

ax2.set_xticklabels(provincias)

plt.show()
'''
```

Actividad básica - prioritaria: De a dos o tres, buscar un dataset externo que les sea de interés, analizar los datos y compartir resultados con el curso completo.

Actividad complementaria - Opcional: Realizar el análisis comparativo entre los siguientes datasets:

- SENASA - Existencias porcinas

- SENASA - Existencias equinas

- SENASA - Existencias de bovinos

- SENASA - Existencias caprinas

- SENASA - Existencias ovinas

- Investigar, analizar y sacar conclusiones.

Actividad básica - prioritaria: Analizar la info del formulario del ejercicio 9 con las gráficas y conclusiones correspondientes.

Actividad prioritaria: Exponer resultados ante tus compañeros. Discutir al respecto.