


2 Análisis de precios de un producto

Objetivo. Mediante el uso de las bibliotecas Numpy, Pandas y Matplotlib, realizar un análisis de datos de los precios de un producto y con ello definir el mejor precio de venta para un producto de una nueva compañía. Usando una visualización efectiva, convencer a la junta de socios de que ese es el mejor precio de acuerdo con la competencia y la información que se tiene.

[HeCompA - Precios-Producto](#) by [Luis M. de la Cruz](#) is licensed under [Attribution-ShareAlike 4.0 International](#) 

Trabajo realizado con el apoyo del Programa UNAM-DGAPA-PAPIME PE101922

- El avance de la tecnología nos provee de herramientas sofisticadas para contar historias con datos.
- Hoy en día es fácil poner datos en una hoja de cálculo, elegir un conjunto de ellos, escoger algunas opciones y presionar un botón para obtener un gráfico.
- Cuando hacemos esto, es posible que nuestra historia brille ¡por su ausencia!
- ¡Las herramientas de software no conocen la historia que quiero contar!
- Tener la habilidad de contar historias con datos y visualizaciones efectivas, es cada vez más importante con el aumento exponencial de los datos.
- Nos permite tomar decisiones a partir de la información presentada.
- Una visualización efectiva es muy importante, por que los gráficos son posiblemente lo único que verá la audiencia.
- Visualización: combinación de gráficos, imágenes, animaciones y tablas para comunicar algo a alguien.

2.1 Ejemplo: Precio al menudeo de un producto

(Con base en: “[Storytelling with data](#)”, Cole Nussbaumer Knaflic, John Wiley & Sons, Inc., Hoboken, New Jersey, 2015. Capítulo 8.)

- Una *startup* ha creado un producto y está analizando cómo ponerle el mejor precio de venta.
- Dos consideraciones importantes en toda startup son:
 - ¿cómo es que los competidores tasan sus productos? y
 - ¿cómo es que este precio ha cambiado con los años?

La siguiente tabla muestra los precios de productos de diferentes compañías competidoras desde 2013 y hasta 2019.

| | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 |
|-------------------|------|------|------|------|------|------|------|
| Producto A | 395 | 420 | 430 | 390 | 300 | 275 | 260 |
| Producto B | 370 | 400 | 405 | 380 | 295 | 255 | 245 |
| Producto C | | | 100 | 180 | 200 | 240 | 182 |
| Producto D | | | | 160 | 265 | 215 | 210 |
| Producto E | | | | | | 100 | 205 |

Misión

Definir el mejor precio de venta para el producto de la nueva compañía y convencer a la junta de socios de que ese es el mejor precio de acuerdo con la competencia y la información que se tiene.

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

# Parámetros para el estilo de las gráficas
params = {'figure.figsize' : (10,5),
          'xtick.labelsize': 16,
          'ytick.labelsize': 16,
          'axes.labelsize' : 20,
          'axes.titlesize' : 20,
          'legend.fontsize': 14,
          'grid.color'      : 'darkgray',
          'grid.linewidth' : 0.5,
          'grid.linestyle' : '--',
          'font.family': 'DejaVu Serif',
          }
plt.rcParams.update(params)
```

```
precios = pd.read_excel("Libro1.xlsx", index_col=0)
precios
```

| | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 |
|--------|-------|-------|-------|-------|-------|------|------|
| Prod A | 395.0 | 420.0 | 430.0 | 390.0 | 300.0 | 275 | 260 |
| Prod B | 370.0 | 400.0 | 405.0 | 380.0 | 295.0 | 255 | 245 |
| Prod C | NaN | NaN | 100.0 | 180.0 | 200.0 | 240 | 182 |
| Prod D | NaN | NaN | NaN | 160.0 | 265.0 | 215 | 210 |
| Prod E | NaN | NaN | NaN | NaN | NaN | 100 | 205 |

2.1.1 Una visualización típica

```
# Visualización 1: barras y colores por año para cada producto

# Utilizamos la función plot() de DataFrame
precios.plot(kind='bar', rot=0)

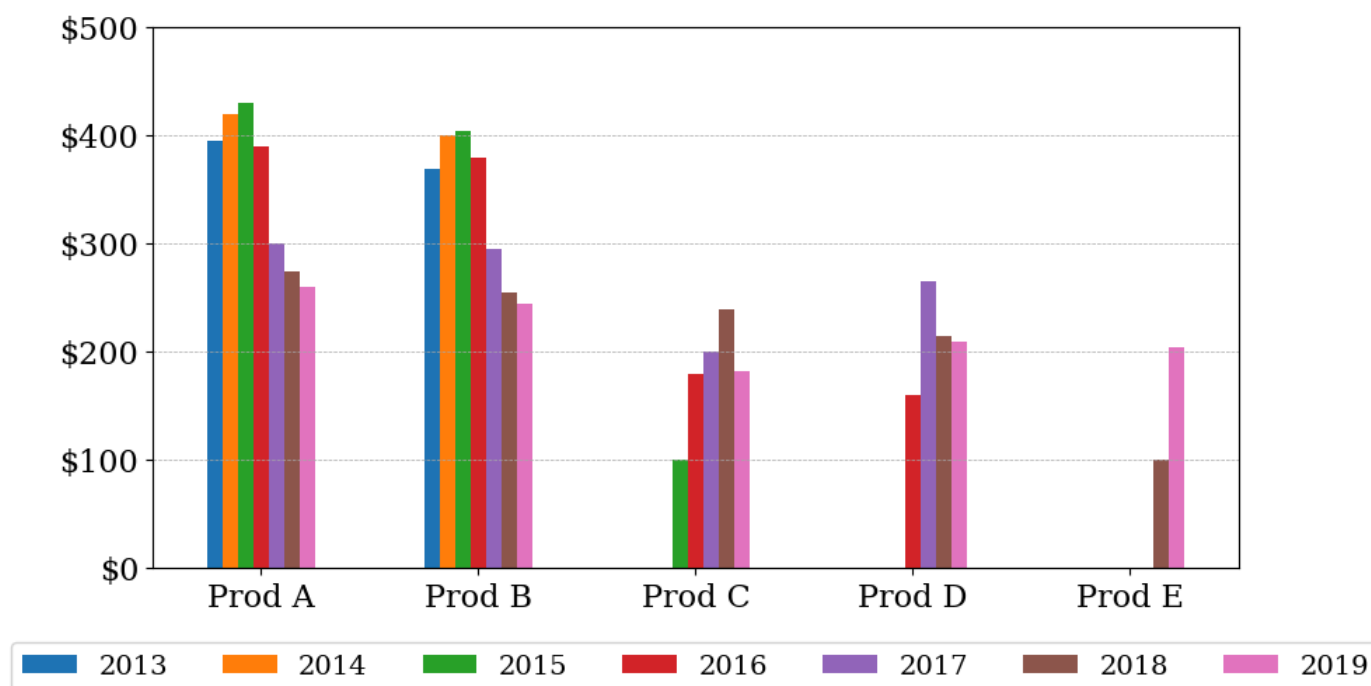
# Definimos las marcas en el eje y
plt.yticks(ticks=[0,100,200,300,400,500],
           labels=['\$', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Personalizamos la gráfica con título, subtítulo, leyenda y rejilla.
plt.title('Los precios han bajado desde la introducción del producto C en 2015',
plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.legend(loc='lower center', bbox_to_anchor=(0.5, -0.25), ncol=7)
plt.grid(axis='y')

# Salvamos la gráfica en una imagen y la mostramos
plt.savefig('vis_inicial.png', bbox_inches='tight', dpi=150)
plt.show()
```

Precio promedio por año

Los precios han bajado desde la introducción del producto C en 2015



Observaciones * Los productos A y B han estado a la venta desde 2013 a 2019. * El producto C comenzó su venta en 2015, el D en 2016 y el E en 2018. * El precio de A y B bajó en 2016, un año después de que entró al mercado el producto C en 2015. Esta observación es importante.

2.2 Construcción de la historia

2.2.1 El contexto: ¿Quién? ¿Qué? ¿Cómo?

- **¿Quién?** VP (Vice President of product), el que conoce todas las cuestiones técnicas del producto, el que hace la primera decisión para poner el precio. Es a quién se presentará el análisis de los datos.
- **¿Qué?** Análisis de cómo el precio de los competidores ha cambiado con el tiempo y recomendar un rango de precios.
- **¿Cómo?** Mostrar un precio promedio al menudeo de los productos A, B, C, D y E para diferentes años.

2.2.1.1 La idea central

- Articular un punto de vista único para transmitir lo que está en juego en un solo enunciado.
- Para nuestro ejemplo diríamos algo como lo siguiente:
 - “Con base en el análisis de precios del mercado y su cambio en el tiempo, para ser competitivos, se recomienda introducir nuestro producto a un precio al menudeo en el rango $P - Q$.”

2.2.1.2 La historia en 3 minutos

- Si solo tuvieras tres minutos para contar tu historia con palabras, ¿cómo lo harías?
- Si eres capaz de hacer esto, significa que tienes muy claro lo que deseas contar.
 - Intenta hacer esto con tu historia.
 - Grábate y escúchate varias veces.
 - Repítelo hasta que sientas que lo has logrado.
- Tanto la idea central como la historia en 3 minutos, serán de utilidad para la presentación de tu historia.

2.2.1.3 El color

- El color puede ayudar a discernir entre valores de datos; pero también puede confundir.
- En esta visualización, los colores distraen del objetivo principal: “mostrar el cambio en los precios a lo largo del tiempo”.
- Entonces, la primera mejora es eliminar el color, vea la siguiente visualización.

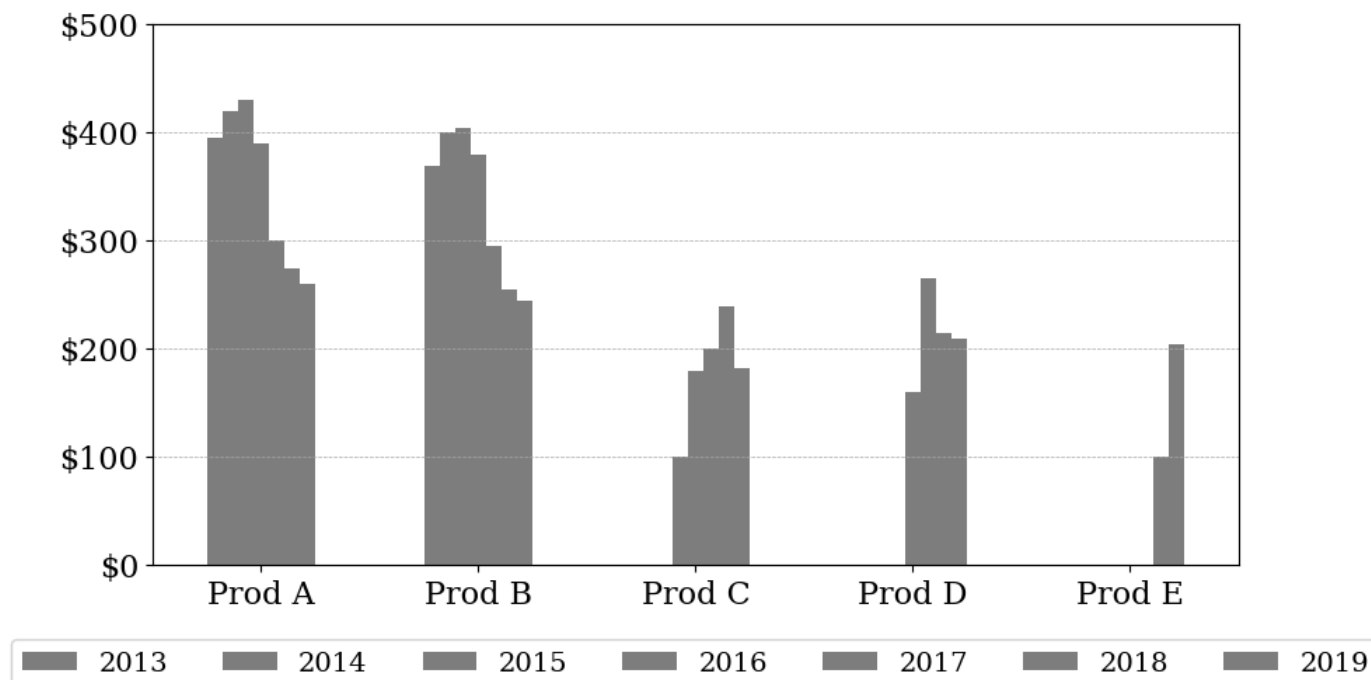
```
# Visualización 2: eliminamos el color

# En la función plot() de DataFrame definimos el color
precios.plot(kind='bar', rot=0, color='gray')

plt.yticks(ticks=[0,100,200,300,400,500], labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])
plt.title('Los precios han bajado desde la introducción del producto C en 2015',
plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.legend(loc='lower center', bbox_to_anchor=(0.5, -0.25), ncol=7)
plt.grid(axis='y')
plt.show()
```

Precio promedio por año

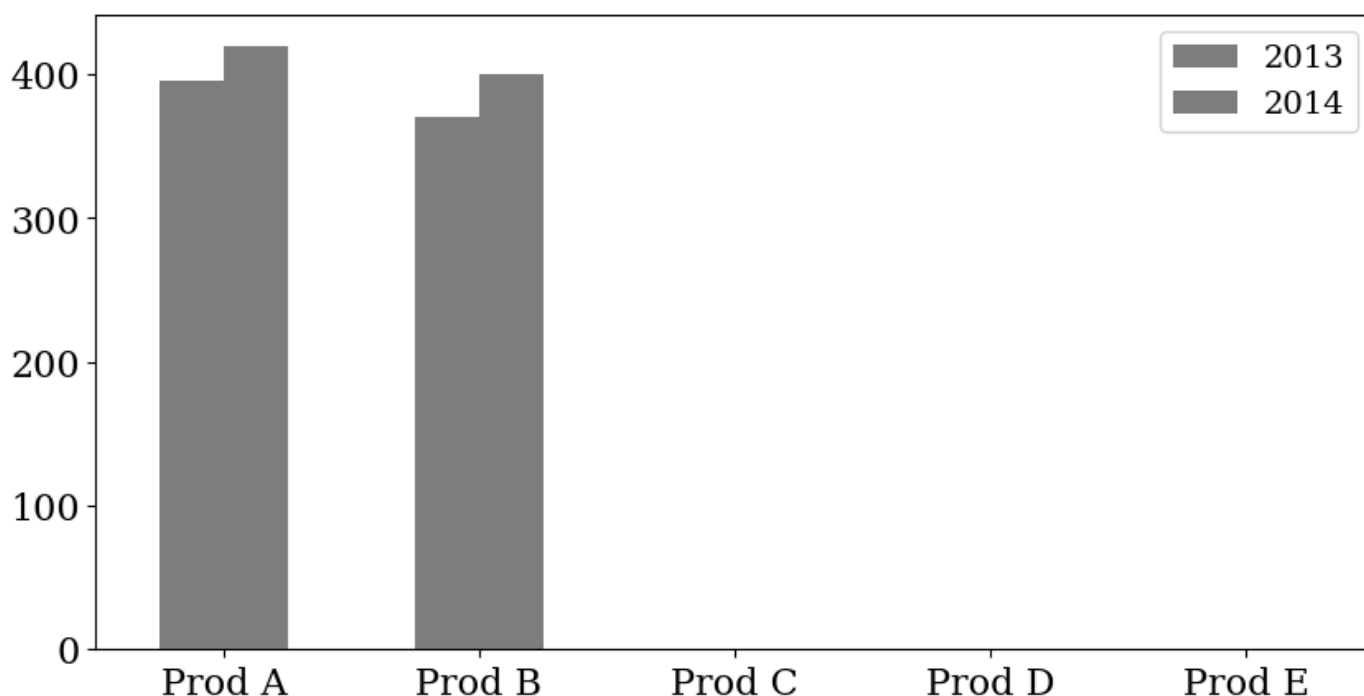
Los precios han bajado desde la introducción del producto C en 2015

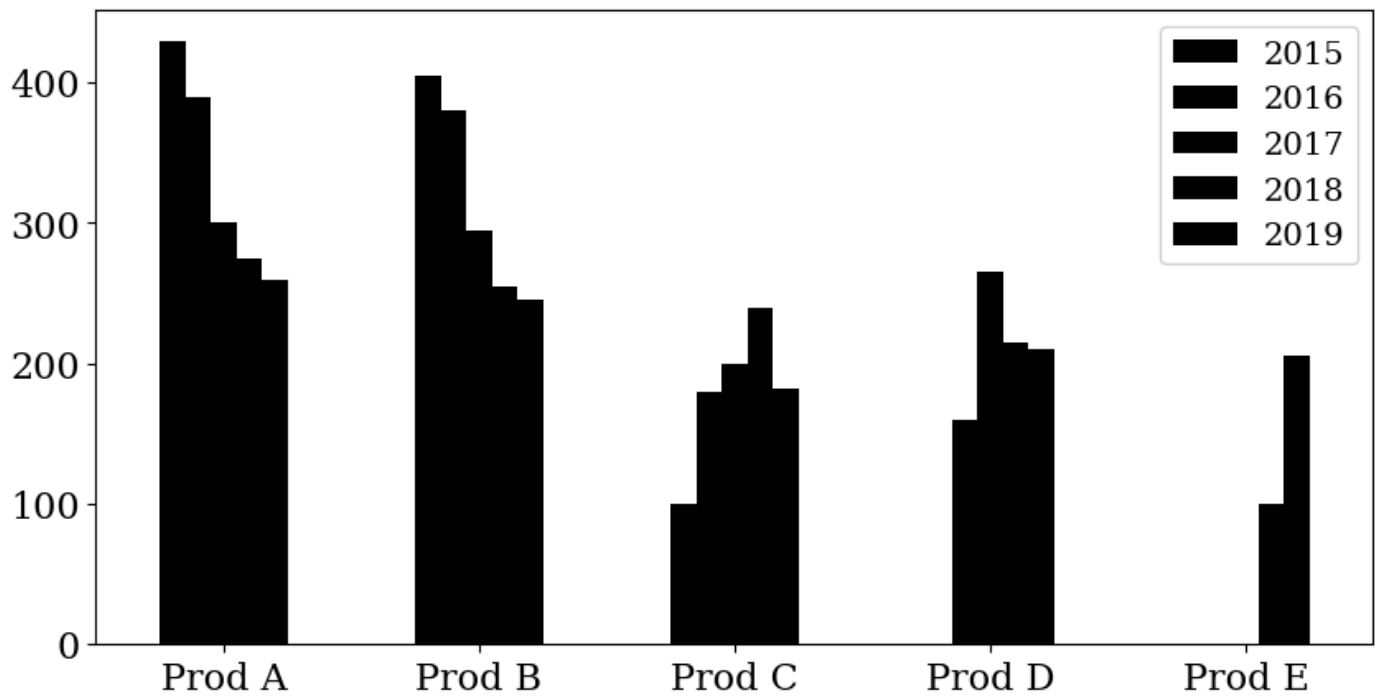


Observaciones - Efectivamente, el color ya no es un distractor, sin embargo, no se distingue la información por años. - De acuerdo con el texto de la primera visualización, vamos a resaltar la información a partir del año 2015, que fue cuando se introdujo el producto C y los precios empezaron a bajar.

Gráfica de barras gris y negro

```
precios.plot(y = [2013, 2014], kind='bar', rot=0, color='gray')
precios.plot(y = [2015, 2016, 2017, 2018, 2019], kind='bar', rot=0, color='k')
```





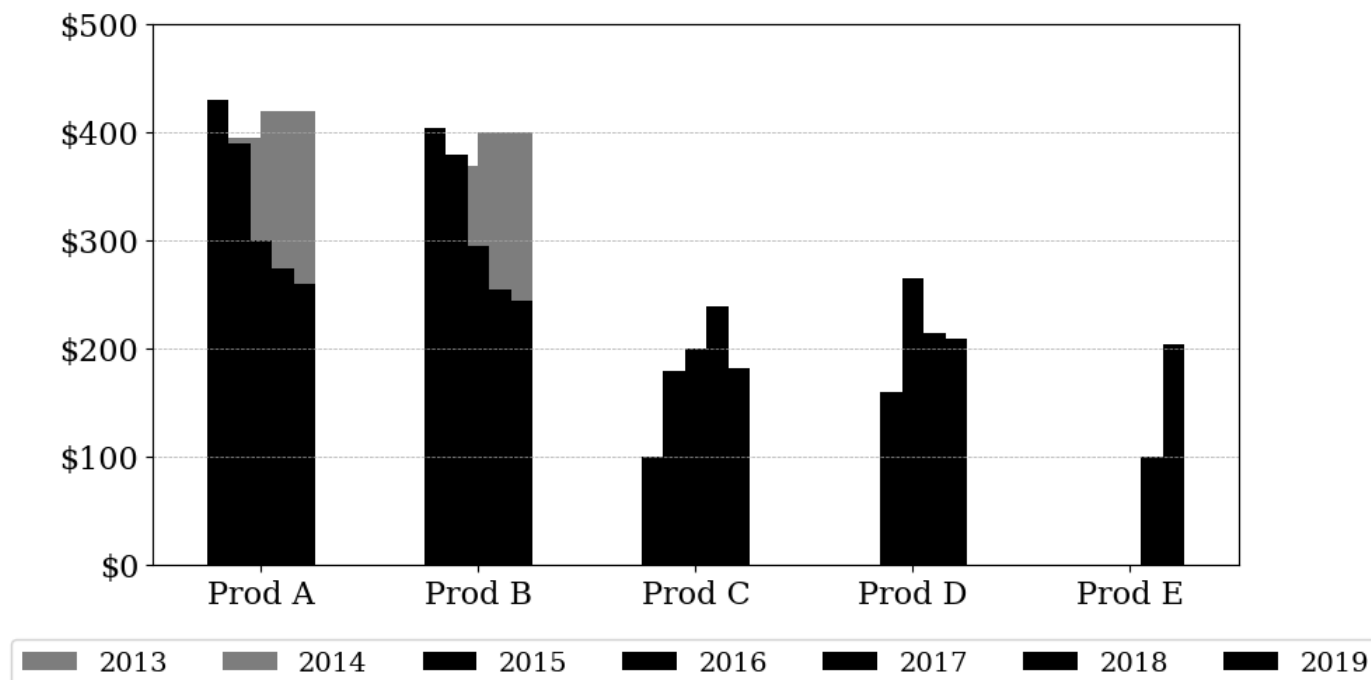
Gráfica de barras gris y negro juntas

```
# Juntamos las dos gráficas en una sola
ax1 = precios.plot(y = [2013, 2014], kind='bar', rot=0, color='gray')
precios.plot(y = [2015, 2016, 2017, 2018, 2019], kind='bar', rot=0, color='k', ax=
ax1)

plt.yticks(ticks=[0,100,200,300,400,500], labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])
plt.title('Los precios han bajado desde la introducción del producto C en 2015',
fontsize=16)
plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.legend(loc='lower center', bbox_to_anchor=(0.5, -0.25), ncol=7)
plt.grid(axis='y')
plt.show()
```

Precio promedio por año

Los precios han bajado desde la introducción del producto C en 2015



Gráfica de barras gris y negras juntas, recorridas

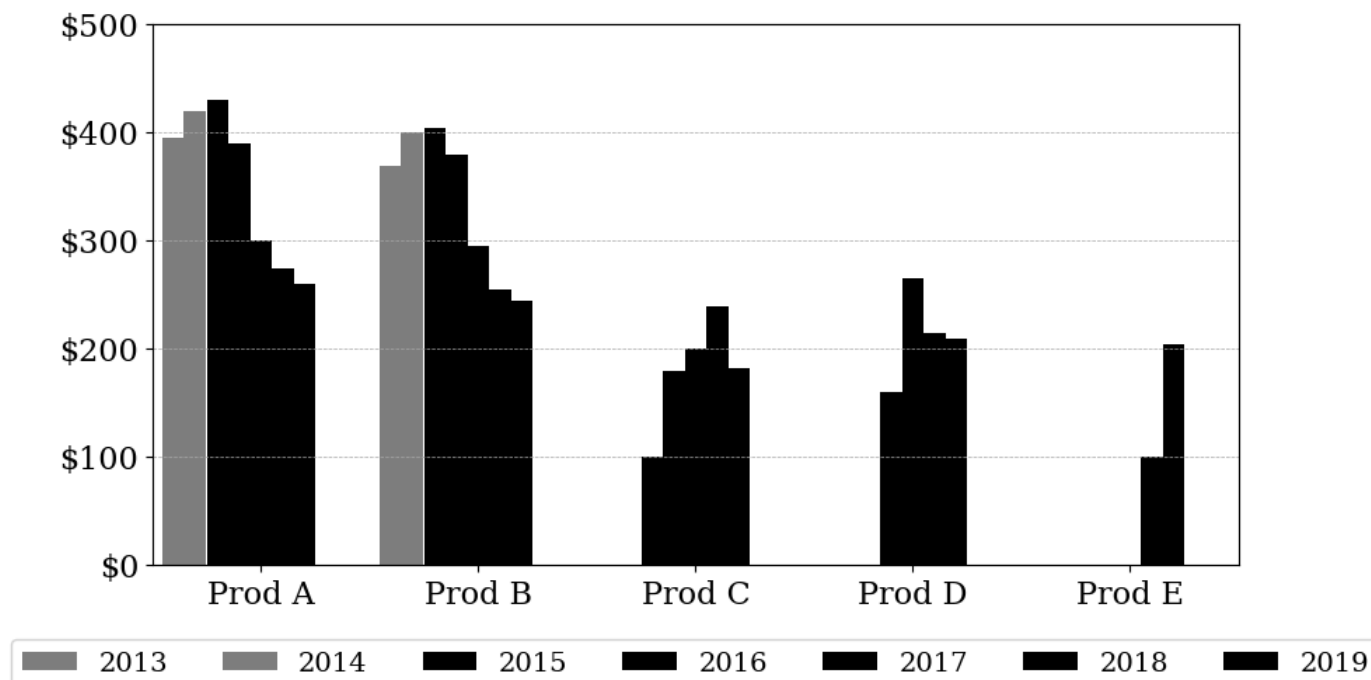
```
# Visualización 3: resaltamos tendencias

# Juntamos las dos gráficas en una sola y las colocamos una junto a otra
ax1 = precios.plot(y = [2013, 2014], kind='bar', rot=0, color='gray',
                    width=0.2, position=2.28)
precios.plot(y = [2015, 2016, 2017, 2018, 2019], kind='bar', rot=0, color='k',
             ax = ax1)

plt.yticks(ticks=[0,100,200,300,400,500], labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])
plt.title('Los precios han bajado desde la introducción del producto C en 2015',
          fontweight='bold', color='green', y=1.05)
plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.legend(loc='lower center', bbox_to_anchor=(0.5, -0.25), ncol=7)
plt.grid(axis='y')
plt.show()
```

Precio promedio por año

Los precios han bajado desde la introducción del producto C en 2015



Observaciones - Se nota claramente como los precios de A y de B van en declive después de 2015. - Pero no pasa lo mismo con los productos D y E, que fueron lanzados en años posteriores a C. - Por lo tanto, el texto en azul de la gráfica, no es correcto. Ese texto debe ser corregido en la visualización final.

2.2.2 Eligiendo la estrategia de visualización

- Parece que la forma en que se está visualizando la información no es la más adecuada.
- Se desea mostrar cómo cambia un precio a lo largo del tiempo y tratar de encontrar una tendencia.
- Es posible que usar líneas sea lo más adecuado.
- Adicionalmente, las líneas eliminan el efecto de escalera que se ve en las barras.

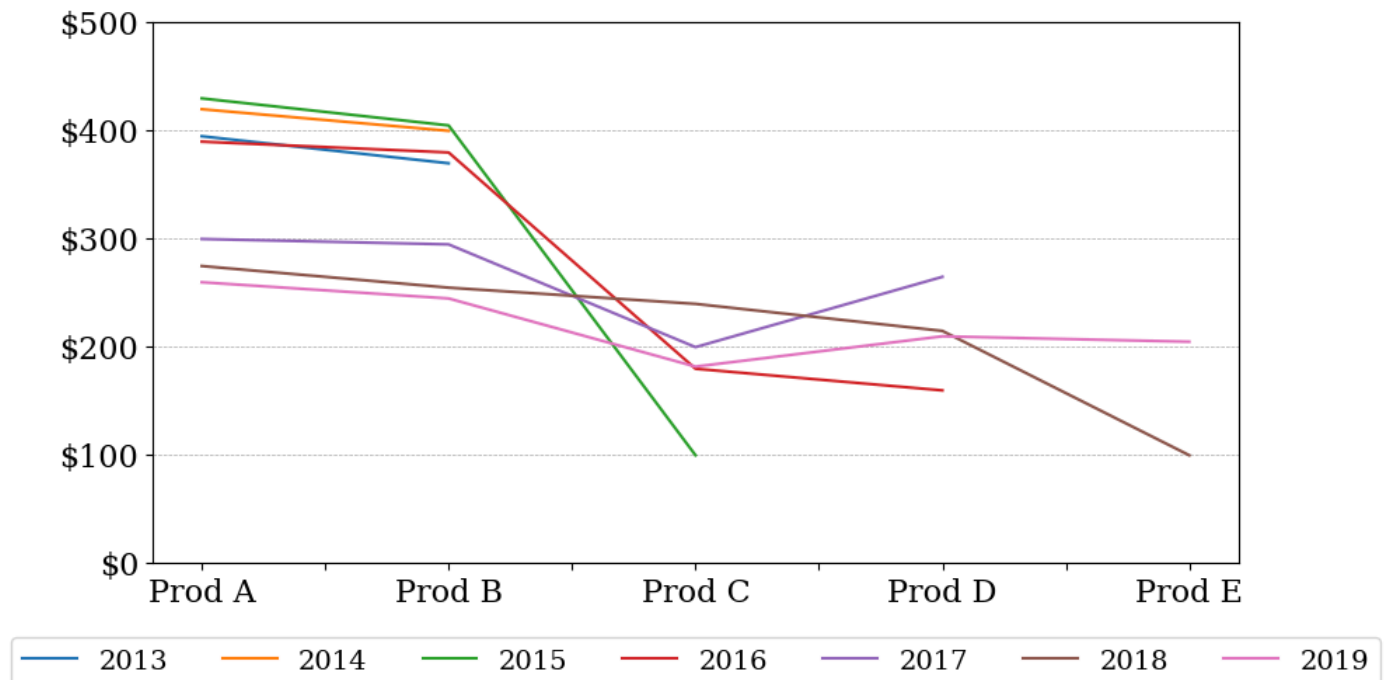
Gráfica de líneas con DataFrame

```
precios.plot(kind='line', rot=0)

plt.yticks(ticks=[0,100,200,300,400,500],
           labels=['\ $0', '\ $100', '\ $200', '\ $300', '\ $400', '\ $500'])
plt.title('Los precios han bajado desde la introducción del producto C en 2015',
plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.legend(loc='lower center', bbox_to_anchor=(0.5, -0.25), ncol=7)
plt.grid(axis='y')
plt.show()
```


Precio promedio por año

Los precios han bajado desde la introducción del producto C en 2015



Observa que las líneas, como se graficaron antes, no dan mucha información y por le contrario se ve todo enmarañado.

Vamos a usar Matplotlib en lo que sigue para tener acceso a más funcionalidades.

2.2.2.1 Líneas por cada producto

Primero arreglamos los datos por producto (renglones del DataFrame)

```
A = np.array(precios.iloc[0])
B = np.array(precios.iloc[1])
C = np.array(precios.iloc[2])
D = np.array(precios.iloc[3])
E = np.array(precios.iloc[4])
print(A)
print(B)
print(C)
print(D)
print(E)
```

```
[395. 420. 430. 390. 300. 275. 260.]
[370. 400. 405. 380. 295. 255. 245.]
[ nan  nan 100. 180. 200. 240. 182.]
[ nan  nan  nan 160. 265. 215. 210.]
[ nan  nan  nan  nan  nan 100. 205.]
```

```
# Arreglo para usarse en el eje x
x = np.array([i for i in range(7)])
```

```
print('\nx: ',x)
```

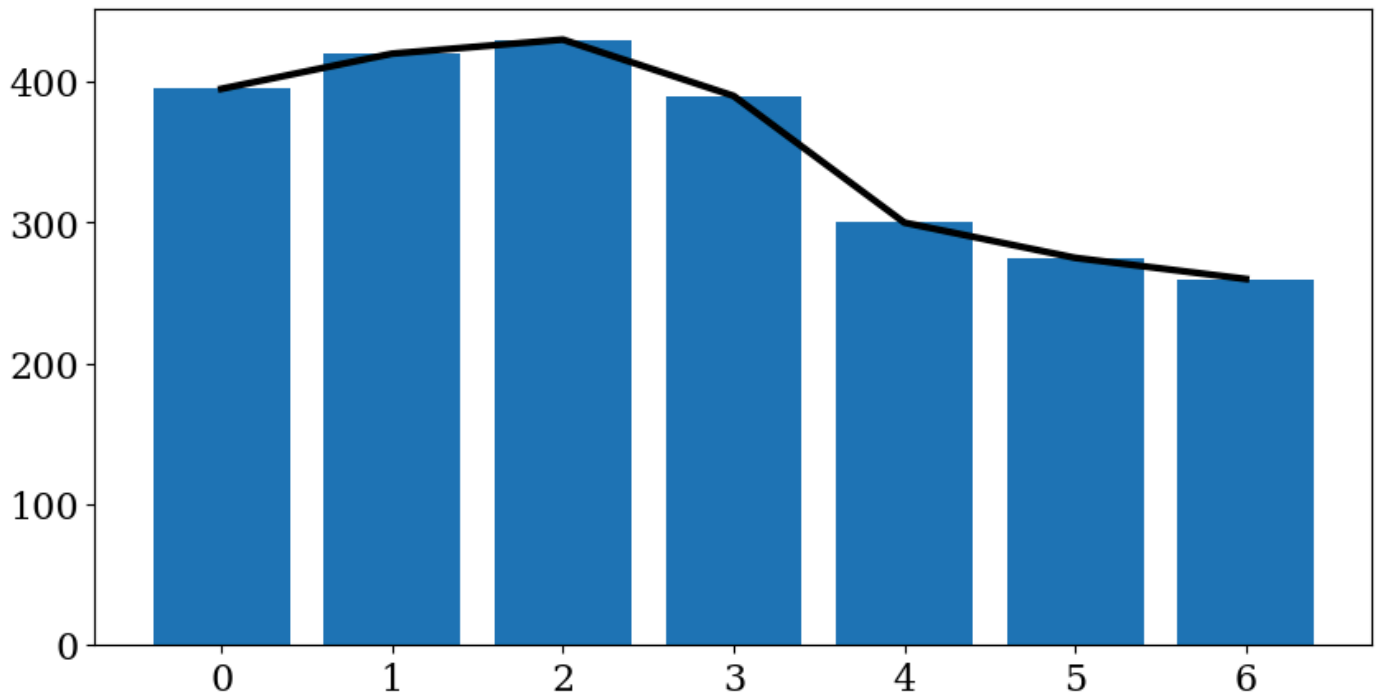
x: [0 1 2 3 4 5 6]

Graficamos barras y líneas

```
# Un primer intento
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.bar(x, A)
ax.plot(x, A, lw=3, c='k')

plt.show()
```



```
# Visualización 4: usamos barras y líneas
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

offset = 10 # Este número nos ayudará a recorrer las gráficas

# Producto A
ax.bar(x, A, color='silver')
ax.plot(x, A, lw=3, c='k')
ax.text(-0.5,410,'2013', fontsize=10, color='k', rotation='vertical')
ax.text(5.75,270,'2019', fontsize=10, color='k', rotation='vertical')

# Producto B
```

```
ax.bar(x+offset, B, color='silver')
ax.plot(x+offset, B, lw=3, c='k')

# Producto C
ax.bar(x+2*offset-2, C, color='silver')
ax.plot(x+2*offset-2, C, lw=3, c='k')

# Producto D
ax.bar(x+3*offset-3, D, color='silver')
ax.plot(x+3*offset-3, D, lw=3, c='k')

# Producto E
ax.bar(x+4*offset-5, E, color='silver')
ax.plot(x+4*offset-5, E, lw=3, c='k')

# Etiquetas de los ejes
ax.set_ylabel('Precio del producto', fontsize=15)
ax.set_xlabel('Año', fontsize=15)

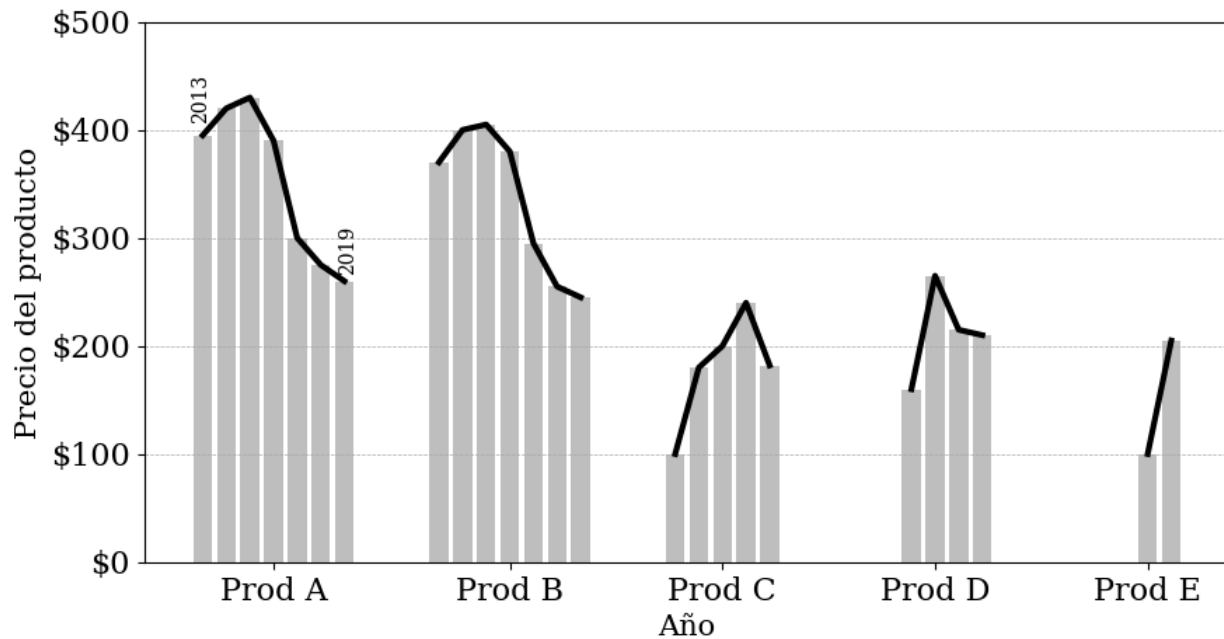
# Marcas sobre los ejes
ax.set_xticks(ticks=[3, offset+3, 2*offset+2, 3*offset+1, 4*offset],
              labels=['Prod A', 'Prod B', 'Prod C', 'Prod D', 'Prod E'])
ax.set_yticks(ticks=[0, 100, 200, 300, 400, 500],
              labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla en el eje y
ax.grid(axis='y')

plt.title('Los precios han bajado desde la introducción del producto C en 2015',
plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.show()
```

Precio promedio por año

Los precios han bajado desde la introducción del producto C en 2015



Observaciones

- Se usa el mismo diseño (ejes, límites, título, ...)
- Esta visualización permite ver con más claridad lo que sucede con el precio de cada producto a lo largo del tiempo.
- Aunque es difícil comparar entre productos.
- Si graficamos en un mismo eje x todos los productos obtenemos algo mejor.

2.2.2.2 Líneas en un mismo eje.

```
x2 = np.arange(2013, 2020, 1)
x2
```

```
array([2013, 2014, 2015, 2016, 2017, 2018, 2019])
```

```
# Visualización 5: usamos solo líneas
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2, A, lw=3, c='k')

# Producto B
ax.plot(x2, B, lw=3, c='k')

# Producto C
ax.plot(x2, C, lw=3, c='k')
```

```

# Producto D
ax.plot(x2, D, lw=3, c='k')

# Producto E
ax.plot(x2, E, lw=3, c='k')

# Etiquetas de los ejes
ax.set_ylabel('Precio del producto', fontsize=15)
ax.set_xlabel('Año', fontsize=15)

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
               labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

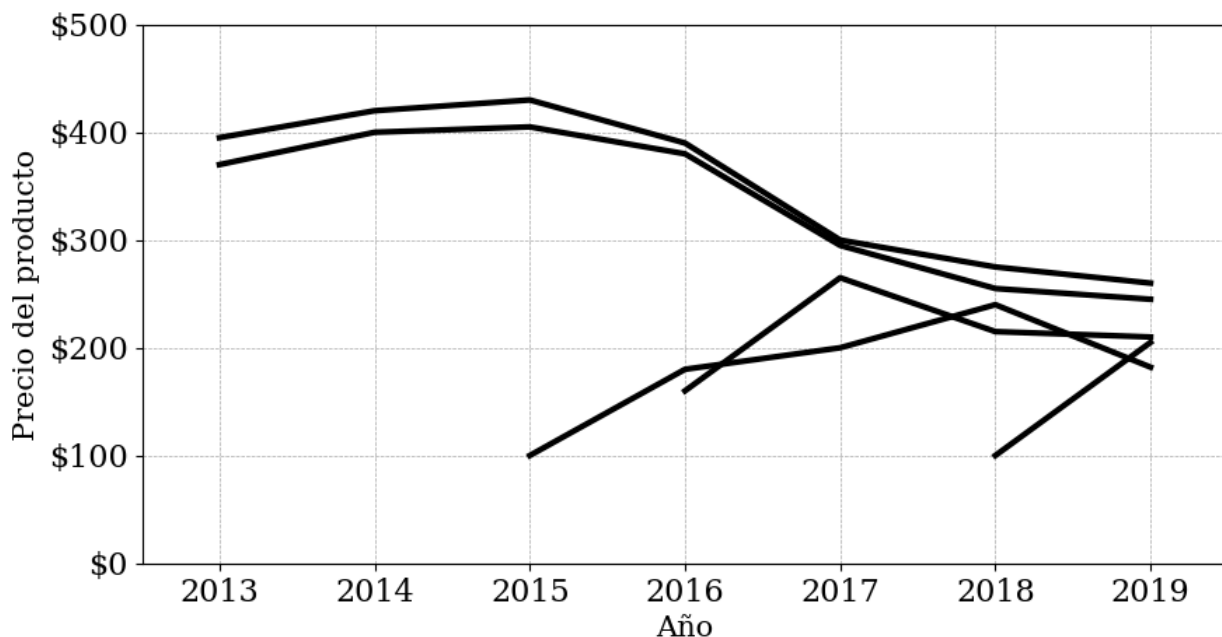
# Rejilla
ax.grid()

plt.title('Los precios han bajado desde la introducción del producto C en 2015',
plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.show()

```

Precio promedio por año

Los precios han bajado desde la introducción del producto C en 2015



Observaciones - Obsérvese que se reduce el desorden y se evita la repetición de etiquetas en las gráficas. - Quizá ahora podamos agregar color (el cual habíamos eliminado antes) para identificar cada producto.

2.2.2.3 Líneas con color

```
# Visualización 6: usamos líneas con color
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2, A, lw=3, label='Prod A')

# Producto B
ax.plot(x2, B, lw=3, label='Prod B')

# Producto C
ax.plot(x2, C, lw=3, label='Prod C')

# Producto D
ax.plot(x2, D, lw=3, label='Prod D')

# Producto E
ax.plot(x2, E, lw=3, label='Prod E')

# Etiquetas de los ejes
ax.set_ylabel('Precio del producto', fontsize=15)
ax.set_xlabel('Año', fontsize=15)

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

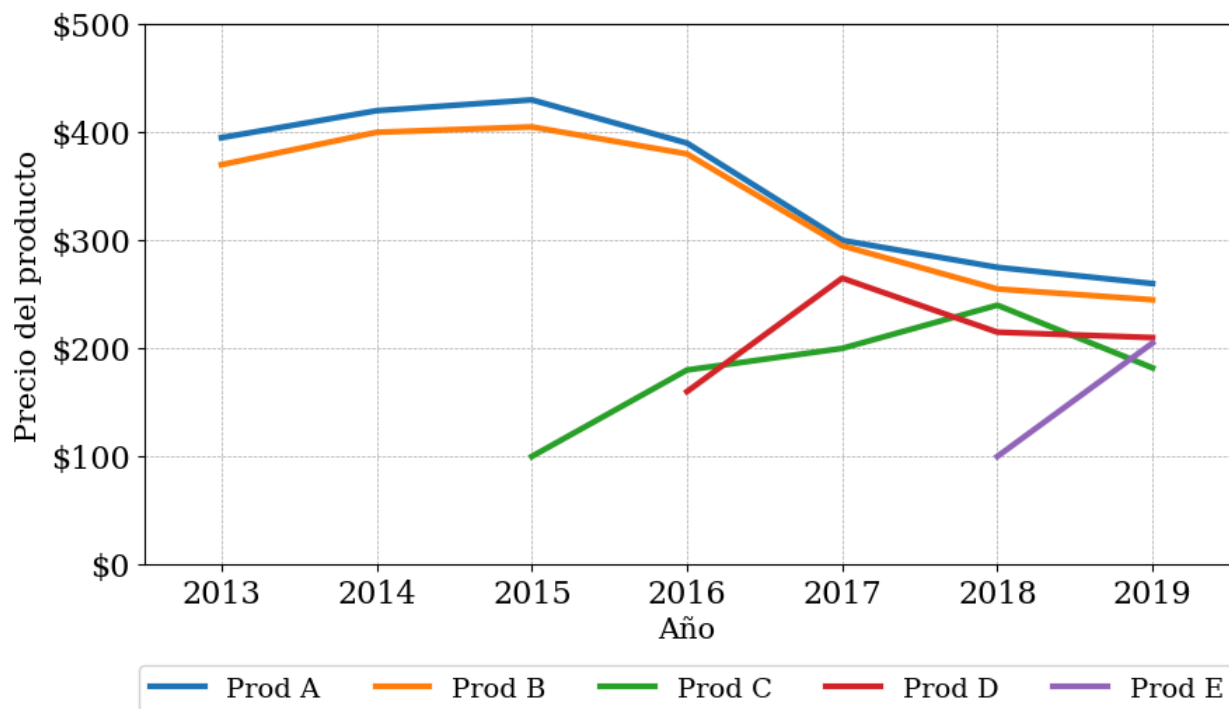
# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
               labels=['\ $0', '\ $100', '\ $200', '\ $300', '\ $400', '\ $500'])

# Rejilla
ax.grid()

plt.title('Los precios han bajado desde la introducción del producto C en 2015',
plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.legend(loc='lower center', bbox_to_anchor=(0.5, -0.30), ncol=7)
plt.show()
```

Precio promedio por año

Los precios han bajado desde la introducción del producto C en 2015



2.2.3 Eliminando el desorden

- Significa eliminar lo que no aporta información. En nuestro ejemplo podemos hacer lo siguiente:
 - Quitar protagonismo al título, debe estar, pero no debe distraer del objetivo principal; en este caso no necesita estar en texto resaltado (bold).
 - Eliminar los bordes de la gráfica y la rejilla.
 - Quitar protagonismo a los ejes y sus etiquetas haciéndolas más tenues.
 - En este caso no es necesario poner la etiqueta a cada eje, pues de la información se deduce de que se trata.
 - Eliminar el color otra vez; se puede usar de manera estratégica; más adelante se verá cómo.
 - Etiquetar las líneas directamente. Esto evita el trabajo visual de la audiencia: ya no tiene que ver primero la leyenda, luego buscar en el gráfico la curva que corresponda, y esto varias veces hasta entender lo que se muestra.

```
# Visualización 7: eliminamos elementos que solo distraen
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2, A, lw=3, c='darkgray')

# Producto B
ax.plot(x2, B, lw=3, c='darkgray')

# Producto C
```

```
ax.plot(x2, C, lw=3, c='darkgray')

# Producto D
ax.plot(x2, D, lw=3, c='darkgray')

# Producto E
ax.plot(x2, E, lw=3, c='darkgray')

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
               labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla
ax.grid()

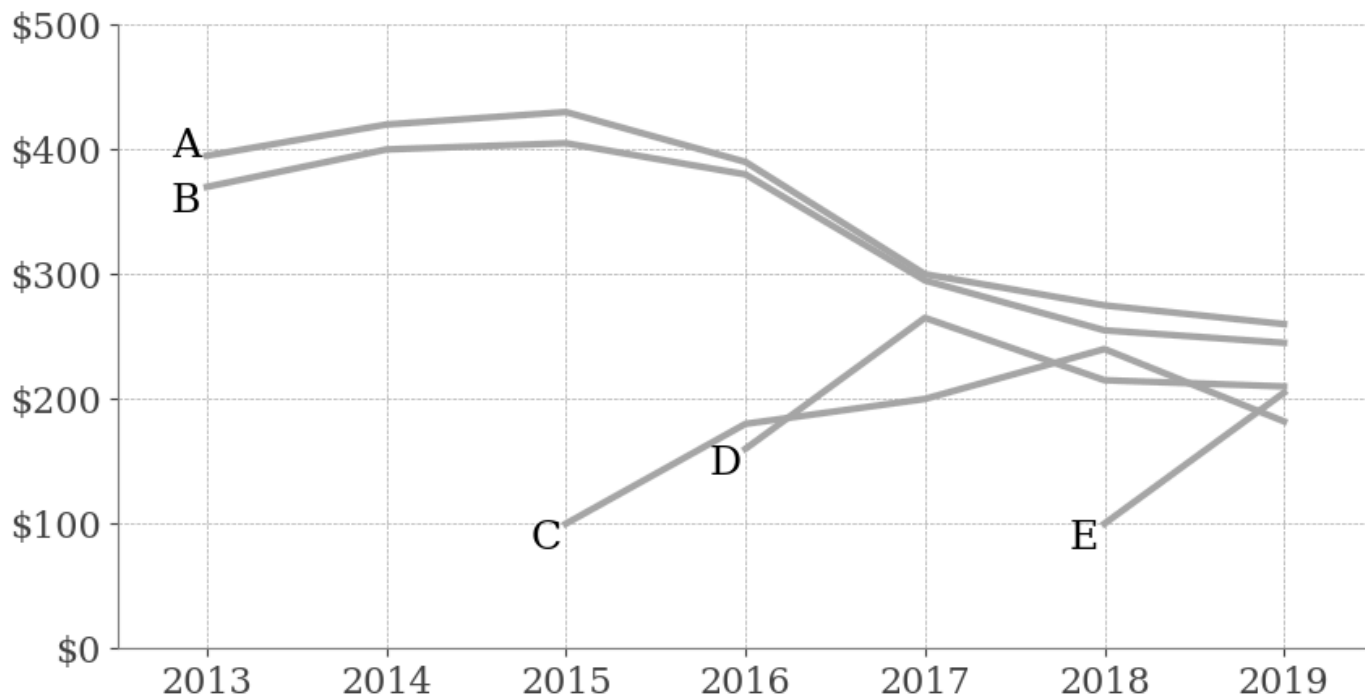
# Etiquetado de cada línea
ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)
ax.text(x = x2[2]-0.20, y = C[2]-20, s = 'C', fontsize = 18)
ax.text(x = x2[3]-0.20, y = D[3]-20, s = 'D', fontsize = 18)
ax.text(x = x2[5]-0.20, y = E[5]-20, s = 'E', fontsize = 18)

# Eliminación de algunas líneas del recuadro
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')

# Color de los ticks
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='gray')

plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.show()
```


Precio promedio por año



Observaciones. - Para el objetivo planteado, esta gráfica muestra claramente la tendencia de los precios de los productos A, B, C, D y E a lo largo del tiempo. - Hemos eliminado información que es irrelevante para ese objetivo. - Esta visualización se puede usar como un lienzo para posteriormente resaltar algunas cosas de interés.

2.2.4 Enfocar la atención: *preattentive features*

- Finalmente llegamos a un punto interesante: enfocar la atención de la audiencia en puntos relevantes mediante el uso estratégico de algunas *preattentive features*.
- Consideremos el texto de la visualización inicial:
 - “Los precios han bajado desde la introducción del producto C in 2010.”
 - Este texto se cambiará por: “Después del lanzamiento del producto C en 2015, el precio promedio al menudeo de los productos existentes ha disminuido.” Este último es más correcto que el primero.
- ¿Cómo se puede demostrar la validez de este último texto usando *preattentive features*?

2.2.4.1 Color y marcadores

```
# Visualización 8: enfocamos la atención en el descenso
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2, A, lw=3, c='darkgray')
ax.plot(x2[2:], A[2:], lw=3, color='royalblue')

# Producto B
ax.plot(x2, B, lw=3, c='darkgray')
```

```
plt.plot(x2[2:], B[2:], lw=3, color='royalblue')

# Producto C
ax.plot(x2, C, lw=3, c='darkgray')
ax.scatter(x2[2], C[2], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Producto D
ax.plot(x2, D, lw=3, c='darkgray')

# Producto E
ax.plot(x2, E, lw=3, c='darkgray')

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
               labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla
ax.grid()

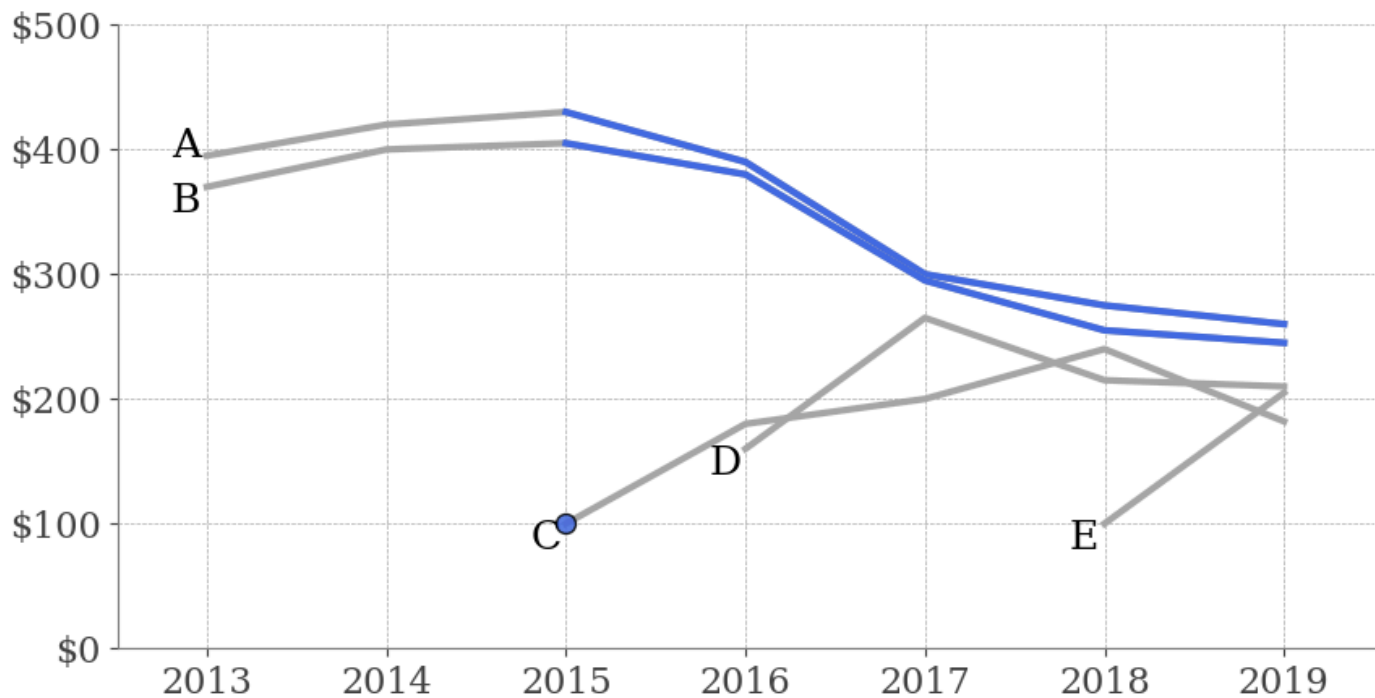
# Etiquetado de cada línea
ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)
ax.text(x = x2[2]-0.20, y = C[2]-20, s = 'C', fontsize = 18)
ax.text(x = x2[3]-0.20, y = D[3]-20, s = 'D', fontsize = 18)
ax.text(x = x2[5]-0.20, y = E[5]-20, s = 'E', fontsize = 18)

# Eliminación de algunas líneas del recuadro
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')

# Color de los ticks
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='#444444')

plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.show()
```

Precio promedio por año



Observaciones.

- En esta última visualización estamos enfocando la atención del público usando un color sobre las gráficas de los productos A y B: se resalta el descenso del precio.
- Adicionalmente, se agrega un marcador con el mismo color en el inicio de la curva del producto C, para indicar cuando se introdujo.
- Se ve claramente el inicio de C y qué pasó con A y B posterior a eso.
- Se usa el color de manera consistente.

2.2.4.2 Subida y bajada del precio

```
# Visualización 9: enfocamos la atención en el ascenso y descenso
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2[2:], A[2:], lw=3, color='royalblue')
ax.plot(x2[0:3], A[0:3], lw=3, color='firebrick')

# Producto B
ax.plot(x2[2:], B[2:], lw=3, color='royalblue')
ax.plot(x2[0:3], B[0:3], lw=3, color='firebrick')

# Producto C
ax.plot(x2[5:], C[5:], lw=3, color='royalblue')
ax.plot(x2[2:6], C[2:6], lw=3, color='firebrick')
```

```
# Producto D
ax.plot(x2[4:], D[4:], lw=3, color='royalblue')
ax.plot(x2[3:5], D[3:5], lw=3, color='firebrick')

# Producto E
ax.plot(x2, E, lw=3, c='firebrick')

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
               labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla
ax.grid()

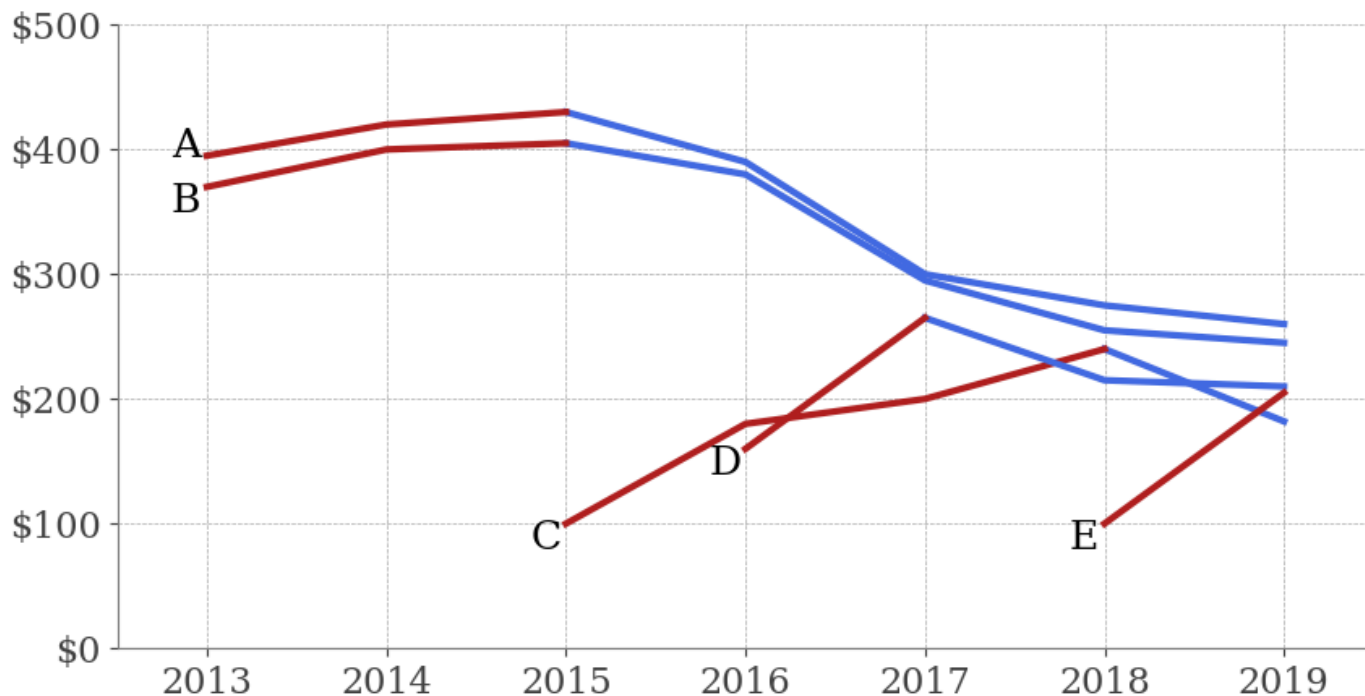
# Etiquetado de cada línea
ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)
ax.text(x = x2[2]-0.20, y = C[2]-20, s = 'C', fontsize = 18)
ax.text(x = x2[3]-0.20, y = D[3]-20, s = 'D', fontsize = 18)
ax.text(x = x2[5]-0.20, y = E[5]-20, s = 'E', fontsize = 18)

# Eliminación de algunas líneas del recuadro
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')

# Color de los ticks
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='gray')

plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.show()
```

Precio promedio por año



Observaciones.

En esta gráfica se muestra que usando la misma estrategia, se puede resaltar el hecho de que: con el lanzamiento de un nuevo producto, es típico ver un ascenso inicial (parte roja) del precio promedio al menudeo, seguido de un descenso (parte azul).

2.2.4.3 Precio final

```
# Visualización 10: marcamos los precios finales
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2, A, lw=3, c='darkgray')
ax.scatter(x2[-1], A[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='red')

# Producto B
ax.plot(x2, B, lw=3, c='darkgray')
ax.scatter(x2[-1], B[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='red')

# Producto C
ax.plot(x2, C, lw=3, c='darkgray')
ax.scatter(x2[-1], C[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='red')

# Producto D
ax.plot(x2, D, lw=3, c='darkgray')
ax.scatter(x2[-1], D[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='red')
```

```
# Producto E
ax.plot(x2, E, lw=3, c='silver')
ax.scatter(x2[-1], E[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
               labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla
ax.grid()

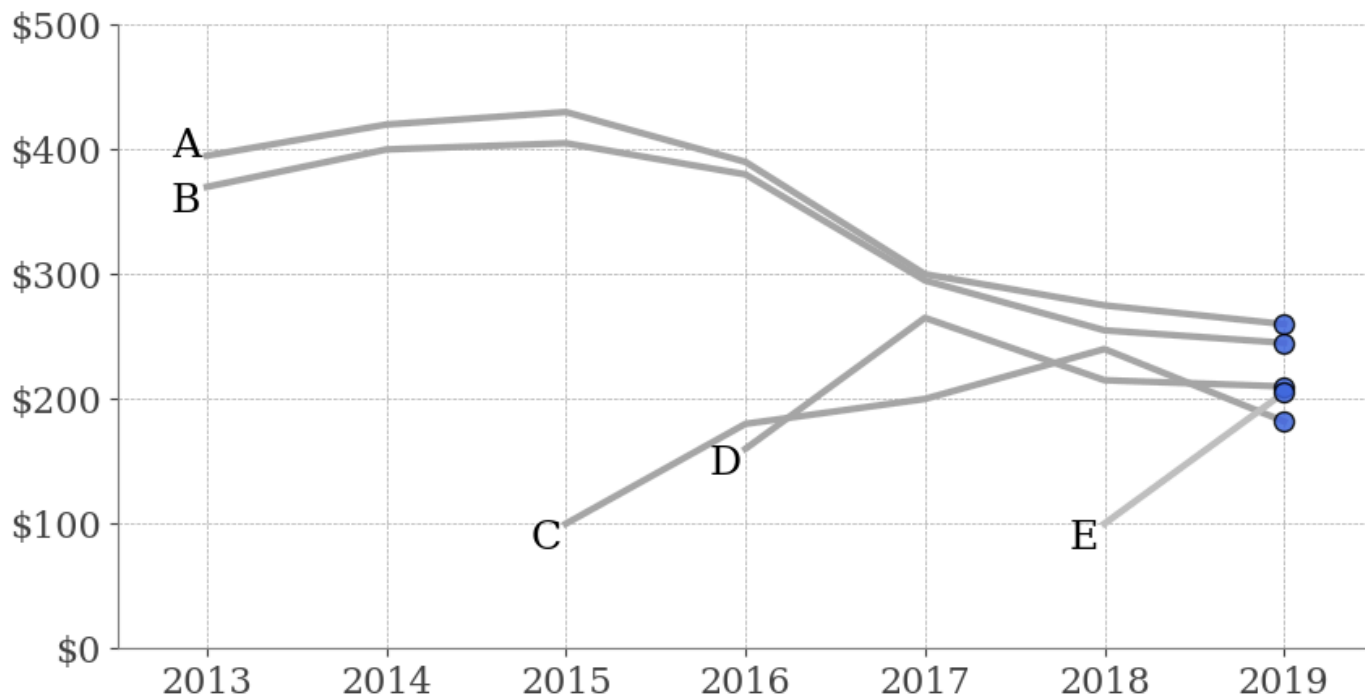
# Etiquetado de cada línea
ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)
ax.text(x = x2[2]-0.20, y = C[2]-20, s = 'C', fontsize = 18)
ax.text(x = x2[3]-0.20, y = D[3]-20, s = 'D', fontsize = 18)
ax.text(x = x2[5]-0.20, y = E[5]-20, s = 'E', fontsize = 18)

# Eliminación de algunas líneas del recuadro
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')

# Color de los ticks
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='#444444')

plt.suptitle('Precio promedio por año', fontsize=24, y=1.05)
plt.show()
```

Precio promedio por año



Observación

- En esta gráfica se muestra que: en 2019, los precios de todos los productos al menudeo convergen al intervalo de $[180, 260]$ con un promedio de 220.
- Quizá sea necesario agregar estos números en el gráfico para mayor claridad.
- Se usan marcadores y colores para llevar nuestra atención hacia los precios finales.

2.2.5 Pensar como diseñador

- Durante todo el proceso hemos estado pensando como diseñadores, pensando en los colores, las líneas, los marcadores y toda la decoración de la gráfica, y cómo resaltar lo que es importante.
- Se puede hacer un poco más:
 - Agregar textos simples a los ejes. Se debe tener cuidado con el uso de mayúsculas, solo poner mayúscula la primera letra de un título.
 - Alinear cada elemento, en este caso, los títulos.
- Además, vamos a agregar una región donde se recomienda debe estar el precio de lanzamiento de nuestro producto.

```
# Visualización 11: marcamos el rango de precios de introducción
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2, A, lw=3, c='darkgray')
ax.scatter(x2[-1], A[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color
```

```
# Producto B
ax.plot(x2, B, lw=3, c='darkgray')
ax.scatter(x2[-1], B[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color

# Producto C
ax.plot(x2, C, lw=3, c='darkgray')
ax.scatter(x2[-1], C[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color

# Producto D
ax.plot(x2, D, lw=3, c='darkgray')
ax.scatter(x2[-1], D[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color

# Producto E
ax.plot(x2, E, lw=3, c='darkgray')
ax.scatter(x2[-1], E[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color

# Promedio final
precios_finales = np.array([A[-1], B[-1], C[-1], D[-1], E[-1],])
promedio_final = np.mean(precios_finales)
ax.scatter(x2[-1], promedio_final, marker='<', alpha=0.85, ec = 'k', s=75,
           zorder=5, color='orange', label='Promedio')

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
              labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla
ax.grid()

# Etiquetado de cada línea
ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)
ax.text(x = x2[2]-0.20, y = C[2]-20, s = 'C', fontsize = 18)
ax.text(x = x2[3]-0.20, y = D[3]-20, s = 'D', fontsize = 18)
ax.text(x = x2[5]-0.20, y = E[5]-20, s = 'E', fontsize = 18)

# Eliminación de algunas líneas del recuadro
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')

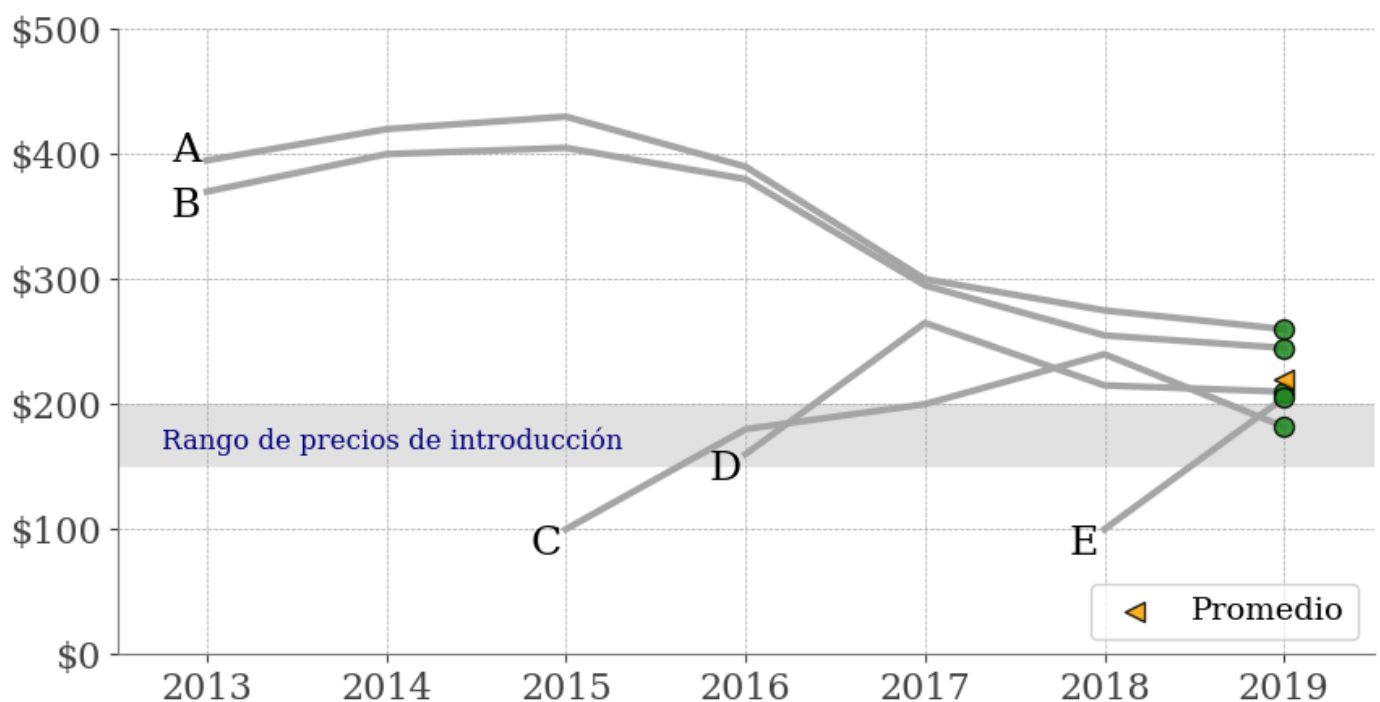
# Color de los ticks
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='#444444')
```



```
# Recuadro para indicar la región del precio final
left, bottom, width, height = (2012, 150, 8, 50)
rect = plt.Rectangle((left, bottom), width, height,
                    facecolor="black", alpha=0.1)
ax.add_patch(rect)
ax.text(x = x2[0]-0.25, y = 165, s = 'Rango de precios de introducción', fontsize=12)

plt.suptitle('Precio promedio por año', fontsize=24, x =0.275, y=1.05)
plt.legend(loc='lower right')
plt.savefig('vis_final.png',bbox_inches='tight', dpi=150)
plt.show()
```

Precio promedio por año





5 Quizz 2. Mínimos Cuadrados

Objetivo.

- Entender en qué consiste el ajuste de rectas por el método de mínimos cuadrados.
- Realizar una implementación de las matemáticas requeridas en este método.
- Probar la implementación con datos generados de manera sintética.

Mínimos cuadrados by [Luis M. de la Cruz](#) is licensed under [Attribution-ShareAlike 4.0 International](#)

Trabajo realizado con el apoyo del Programa UNAM-DGAPA-PAPIME PE101922

6 Introducción.

El **método de mínimos cuadrados** se utiliza para ajustar rectas a series de datos presentados en la forma (x, y) como en la siguiente tabla:

| | | | | |
|-------|-------|-------|---------|-------|
| x_0 | x_1 | x_2 | \dots | x_N |
| y_0 | y_1 | y_2 | \dots | y_N |

Este tipo de datos pueden provenir de diferentes fuentes como estudios geofísicos en campo, estudios experimentales en laboratorio, obtención de datos mediante encuestas, observación de fenómenos naturales, etc. La idea es estudiar como una variable depende de la otra.

Una posibilidad es que la variación sea lineal es decir: $y = mx + b$.

Sin embargo, en la mayoría de los casos no es posible obtener una recta exacta que pase por todos los puntos.

Para ello, el método de mínimos cuadrados nos proporciona con una metodología para obtener la mejor recta que represente a todos los puntos del conjunto de datos.

7 Conjuntos de datos.

Para esta actividad vamos a generar de manera sintética 4 conjuntos de datos. Usaremos la función `np.random.rand()` de numpy.

```
import numpy as np
import matplotlib.pyplot as plt
from macti.evaluation import *
```

```
quizz = Quizz('q2', 'HeCompA', 'local')
```

```
# Total de datos de cada conjunto
N = 20

# Fijamos la semilla para obtener los mismos valores siempre
np.random.seed(0)

# Conjunto 0
x0 = np.linspace(0,10,N)
y0 = x0 + np.random.randn(N)*2

# Conjunto 1
x1 = np.random.randn(len(x0))*10
y1 = np.random.randn(len(x0))*10

# Conjunto 2
x2 = np.arange(0,N)*100
y2 = -x2 + np.random.randn(N)*500

# Conjunto 3
x3 = np.array([i for i in range(N)])
y3 = np.random.rand(N)*0.5-0.5
```

```
print('\nArreglo x0 :\n{}'.format(x0))
print('Arreglo y0 :\n{}'.format(y0))

print('\nArreglo x1 :\n{}'.format(x1))
print('Arreglo y1 :\n{}'.format(y1))

print('\nArreglo x2 :\n{}'.format(x2))
print('Arreglo y2 :\n{}'.format(y2))

print('\nArreglo x3 :\n{}'.format(x3))
print('Arreglo y3 :\n{}'.format(y3))
```

Arreglo x0 :

```
[ 0.          0.52631579  1.05263158  1.57894737  2.10526316  2.63157895
  3.15789474  3.68421053  4.21052632  4.73684211  5.26315789  5.78947368
  6.31578947  6.84210526  7.36842105  7.89473684  8.42105263  8.94736842
  9.47368421 10.         ]
```

Arreglo y0 :

```
[ 3.52810469  1.32663021  3.01010755  6.06073377  5.84037914  0.67702319
  5.05807157  3.38149611  4.00408861  5.55803911  5.55124504  8.6980207
  7.83786492  7.0854553  8.25614752  8.5620855  11.40921078  8.53705189
 10.09981961  8.29180852]
```

Arreglo x1 :

```
[-25.52989816  6.53618595  8.64436199 -7.4216502  22.69754624
 -14.54365675  0.45758517 -1.8718385  15.32779214  14.6935877]
```

```
1.54947426  3.7816252  -8.87785748 -19.80796468  -3.47912149
1.56348969 12.30290681 12.02379849  -3.87326817  -3.02302751]
```

Arreglo y1 :

```
[-10.48552965 -14.20017937 -17.06270191  19.50775395  -5.09652182
 -4.38074302 -12.5279536   7.77490356 -16.13897848  -2.1274028
 -8.95466561  3.86902498  -5.10805138 -11.80632184  -0.28182228
 4.28331871  0.66517222  3.02471898  -6.34322094  -3.62741166]
```

Arreglo x2 :

```
[  0 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300
1400 1500 1600 1700 1800 1900]
```

Arreglo y2 :

```
[-336.23022389 -279.77658077 -606.57314102 -1163.14130117
 -311.28692887 -700.8904681  -1415.09917348  -468.60887224
 -1253.64918219 -874.0273021  -635.45471891 -1035.50854462
 -630.29965773 -1917.41291018 -1198.82917941 -1842.40504547
 -2035.39857459 -1989.42483238 -1955.77626606 -1871.91732889]
```

Arreglo x3 :

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

Arreglo y3 :

```
[-0.08552999 -0.49765226 -0.16109173 -0.36499601 -0.13240299 -0.01890573
 -0.37562343 -0.21192133 -0.20397903 -0.21387405 -0.38845918 -0.02362549
 -0.27643731 -0.07679566 -0.15026036 -0.35128152 -0.09310109 -0.30174713
 -0.0594484  -0.20936356]
```

8 Análisis exploratorio.

```
plt.figure(figsize=(10,5))

ax0 = plt.subplot(221)
ax0.scatter(x0, y0, fc = 'C0', ec='k', alpha=0.75)
ax0.set_xlabel('$x_0$')
ax0.set_ylabel('$y_0$')
ax0.grid()

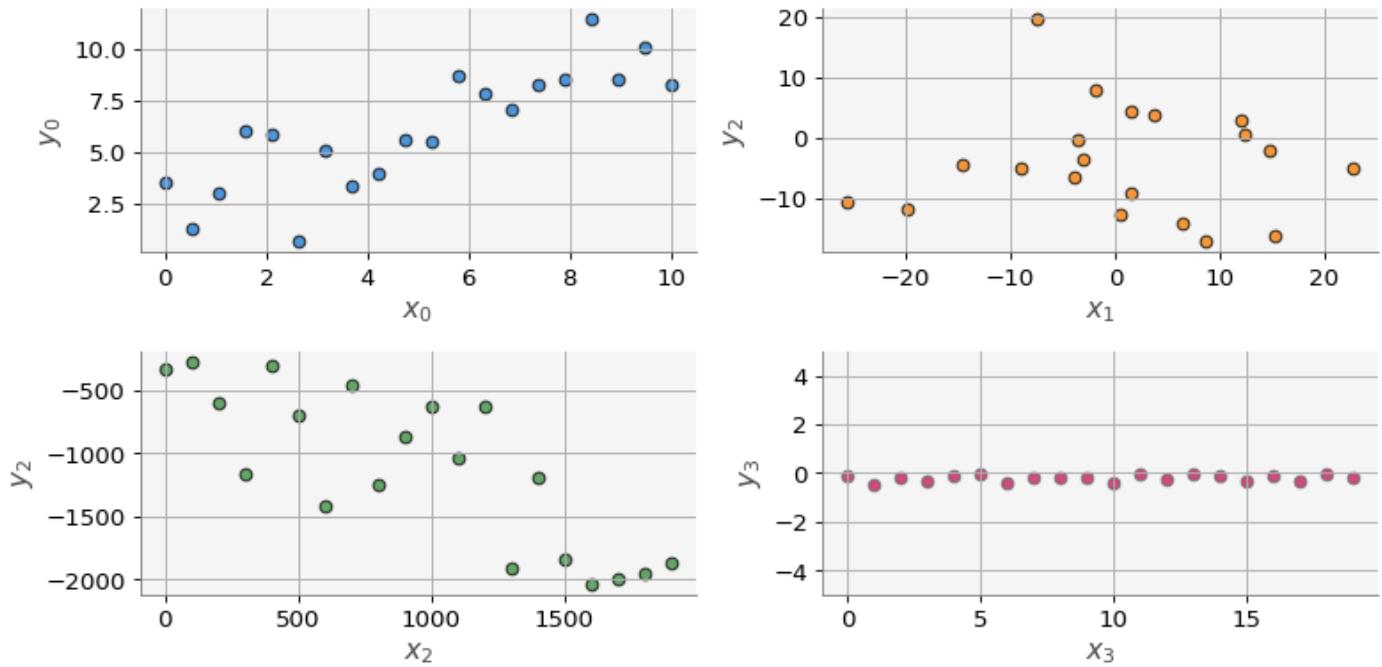
ax1 = plt.subplot(222)
ax1.scatter(x1, y1, fc = 'C1', ec='k', alpha=0.75)
ax1.set_xlabel('$x_1$')
ax1.set_ylabel('$y_2$')
ax1.grid()

ax2 = plt.subplot(223)
ax2.scatter(x2, y2, fc = 'C2', ec='k', alpha=0.75)
ax2.set_xlabel('$x_2$')
ax2.set_ylabel('$y_2$')
ax2.grid()

ax3 = plt.subplot(224)
```

```
ax3.scatter(x3, y3, fc = 'C3', ec='dimgrey', alpha=0.75)
ax3.set_xlabel('$x_3$')
ax3.set_ylabel('$y_3$')
ax3.set_ylim(-5,5)
ax3.grid()

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



9 Valores de m y b óptimos.

Los valores de la pendiente y de la ordenada al origen de la recta que mejor aproxima a un conjunto de datos están dados por las siguientes fórmulas:

$$m = \frac{\sum_{i=1}^N x_i(y_i - \bar{y})}{\sum_{i=1}^N x_i(x_i - \bar{x})} \quad (1)$$

$$b = \bar{y} - m\bar{x} \quad (2)$$

donde $\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i$ y $\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$ representan el valor medio de los datos.

Podemos usar los conjuntos de datos definidos antes para calcular m y b y con ello obtener la recta que ajusta los datos.

9.1 Ejercicio 1.

Calcular la media del conjunto de datos x_0, y_0 . Almacenar las medias en las variables X y Y .

```
#### BEGIN SOLUTION

#### FileAnswer definition
# Se crea un objeto para guardar las respuestas, solo una vez.
file_answer = FileAnswer()
#file_answer.verb = 0

# Media de x
X = 0.0
for xi in x0:
    X += xi
X /= N

# Media de y
Y = 0.0
for yi in y0:
    Y += yi
Y /= N

file_answer.write('1', X, 'Revisa la definición de la media y tu implementación p
file_answer.write('2', Y, 'Revisa la definición de la media y tu implementación p

# END SOLUTION

print('Media de x: {}'.format(X))
print('Media de y: {}'.format(Y))
```

Media de x: 5.0

Media de y: 6.138669185891269

```
quizz.eval_numeric('1', X)
quizz.eval_numeric('2', Y)
```

1 | Tu resultado es correcto.

2 | Tu resultado es correcto.

9.2 Ejercicio 2.

Calcular la pendiente m del conjunto de datos x_0, y_0 usando la fórmula (1). Almacenar la pendiente en la variable `m`.

```
#### BEGIN SOLUTION
# Cálculo de m
Sxx = 0
Sxy = 0
for xi, yi in zip(x0, y0):
    Sxy += xi * (yi - Y)
    Sxx += xi * (xi - X)
m = Sxy / Sxx

file_answer.write('3', m, 'Revisa la fórmula para el cálculo de la pendiente y tu

# END SOLUTION

print('Pendiente m: {}'.format(m))
```

Pendiente m: 0.7725548289646033

```
quizz.eval_numeric('3', m)
```

3 | Tu resultado es correcto.

9.3 Ejercicio 3.

Calcular la ordenada al origen b del conjunto de datos x_0, y_0 usando la fórmula (2). Almacenar la pendiente en la variable `b`.

```
# Cálculo de b
# b = ...
#### BEGIN SOLUTION
# Cálculo de b
b = Y - m * X

file_answer.write('4', b, 'Revisa la fórmula para el cálculo de la ordenada al or

#### END SOLUTION

print('Ordenada al origen b: {}'.format(b))
```

Ordenada al origen b: 2.2758950410682526

```
quizz.eval_numeric('4', b)
```

4 | Tu resultado es correcto.

9.4 Ejercicio 4.

Construir una recta usando los valores de m y b calculados en los ejercicios anteriores como sigue: *

Construir un arreglo para x , de nombre `xr0`, usando alguna función de `numpy` (el arreglo debe ser de 10 elementos):

```
xr0 = np. ...
```

- Evaluar $y = mx + b$ usando el arreglo `xr0` para generar el arreglo `yr0`:

```
yr0 = ...
```

```
# Construcción de las rectas
# xr0 = np....
# yr0 = ...

#### BEGIN SOLUTION
# Construcción de la recta
xr0 = np.linspace(x0.min(), x0.max(), 10)
yr0 = m * xr0 + b

file_answer.write('5', xr0, 'Recuerda que el arreglo debe tener 10 elementos, che
file_answer.write('6', yr0, 'Solo tienes que implementar la fórmula de la recta u

#### END SOLUTION

print(xr0)
print(yr0)
```

```
[ 0.          1.11111111  2.22222222  3.33333333  4.44444444  5.55555556
 6.66666667  7.77777778  8.88888889 10.          ]
[ 2.27589504  3.1342893  3.99268355  4.8510778  5.70947206  6.56786631
 7.42626057  8.28465482  9.14304908 10.00144333]
```



```
quizz.eval_numeric('5', xr0)
quizz.eval_numeric('6', yr0)
```

 5 | Tu resultado es correcto.

 6 | Tu resultado es correcto.

Si realizaste correctamente los ejercicios 1 — 4, entonces la siguiente celda de código graficará los cuatro conjuntos de datos y en la primera gráfica se verá la línea recta que construiste.

NOTA. En esta graficación estamos usando la biblioteca `macti.visual`.

```
import macti.visual as mvis

axp = [dict(xlabel='$x_0$', ylabel='$y_0$'),
       dict(xlabel='$x_1$', ylabel='$y_1$'),
       dict(xlabel='$x_2$', ylabel='$y_2$'),
       dict(xlabel='$x_3$', ylabel='$y_3$', ylim=(-5,5)),]

vis = mvis.Plotter(2,2,fig_par=dict(figsize=(10,5)), axis_par=axp)

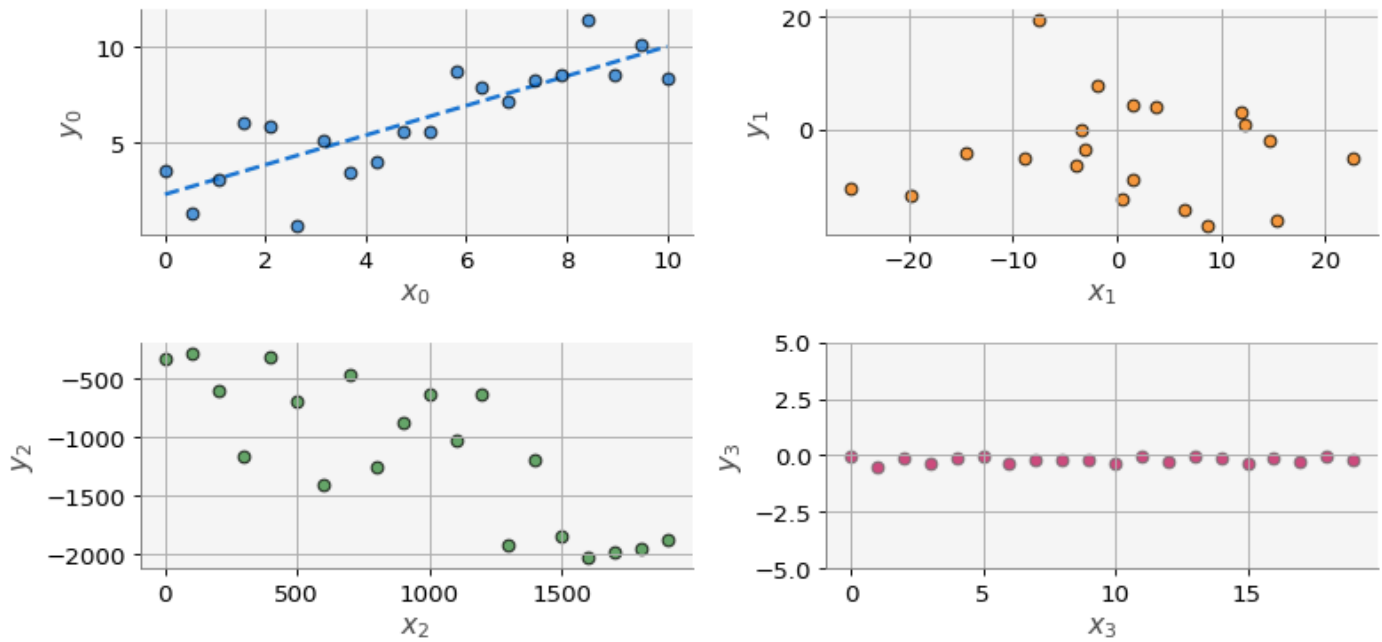
vis.plot(1, xr0, yr0, c='C0', lw=2, ls = '--')
vis.scatter(1, x0, y0, fc='C0', ec='k', alpha=0.75, zorder=5)

vis.scatter(2, x1, y1, fc='C1', ec='k', alpha=0.75)

vis.scatter(3, x2, y2, fc='C2', ec='k', alpha=0.75)

vis.scatter(4, x3, y3, fc='C3', ec='dimgrey', alpha=0.75)

vis.tight_layout()
vis.grid()
vis.show()
```



9.5 Ejercicio 5.

Construir la función `media(x)` para calcular la media de un conjunto de datos. La función debe recibir como entrada el arreglo con los datos, `x`, y regresa la media de los mismos. Probar la función para los datos `x0` y `y0` como sigue:

```
X0 = media(x0)
Y0 = media(y0)
```

```
# Función media()
# def media (x) :
# ...
```

```
#### BEGIN SOLUTION
```

```
def media(x):
    xm = 0.0
    for xi in x:
        xm += xi
    return xm / N
```

```
X0 = media(x0)
Y0 = media(y0)
```

```
file_answer.write('7', X0, 'Revisa la definición de la media y la implementación
file_answer.write('8', Y0, 'Revisa la definición de la media y la implementación
```

```
#### END SOLUTION
```

```
print('Media de x0 : {}'.format(X0))
print('Media de y0 : {}'.format(Y0))
```

Media de x0 : 5.0

Media de y0 : 6.138669185891269

```
quizz.eval_numeric('7', X0)
quizz.eval_numeric('8', Y0)
```

7 | Tu resultado es correcto.

8 | Tu resultado es correcto.

9.6 Ejercicio 6.

Construir la función `mincua(x, y)` para calcular m y b de un conjunto de datos. La función recibe como entrada los arreglos de datos, x , y , y regresa m y b . Calcular m y b para los conjuntos de datos:

- (x_1, y_1) , almacenar m y b en las variables `m1` y `b1` respectivamente.
- (x_2, y_2) , almacenar m y b en las variables `m2` y `b2` respectivamente.
- (x_3, y_3) , almacenar m y b en las variables `m3` y `b3` respectivamente.

```
# Función mincua()
# def mincua (x) :
# ...
#

#### BEGIN SOLUTION
def mincua(x, y):
    # Cálculo de la media
    X = media(x)
    Y = media(y)

    # Cálculo de m
    Sxx = 0
    Sxy = 0
    for xi, yi in zip(x, y):
        Sxy += xi * (yi - Y)
        Sxx += xi * (xi - X)
    m = Sxy / Sxx
```

```
# Cálculo de b
b = Y - m * X

return m, b
#### END SOLUTION
```

```
# Cálculo de m y b
# m1, b1 = ...
# ...

#### BEGIN SOLUTION
m1, b1 = mincua(x1, y1)
m2, b2 = mincua(x2, y2)
m3, b3 = mincua(x3, y3)

file_answer.write('9' , [m1, b1], 'Checa que ejecutaste la función mincua(x,y) co
file_answer.write('10', [m2, b2], 'Checa que ejecutaste la función mincua(x,y) co
file_answer.write('11', [m3, b3], 'Checa que ejecutaste la función mincua(x,y) co

#### END SOLUTION

print('m1 = {}, \t b1 = {}'.format(m1, b1))
print('m2 = {}, \t b2 = {}'.format(m2, b2))
print('m3 = {}, \t b3 = {}'.format(m3, b3))
```

```
m1 = -0.02002336877662938,    b1 = -3.9396674988126574
m2 = -0.86095919430528,      b2 = -308.1742770138992
m3 = 0.003809973693633117,    b3 = -0.24601956385565651
```

```
quizz.eval_numeric('9', [m1, b1])
quizz.eval_numeric('10', [m2, b2])
quizz.eval_numeric('11', [m3, b3])
```

9 | Tu resultado es correcto.

10 | Tu resultado es correcto.

11 | Tu resultado es correcto.

9.7 Ejercicio 7.

Construir las rectas de cada conjunto de datos como sigue:

- Usando m_1 y b_1 construir los arreglos de coordenadas xr_1 y yr_1 .
- Usando m_2 y b_2 construir los arreglos de coordenadas xr_2 y yr_2 .
- Usando m_3 y b_3 construir los arreglos de coordenadas xr_3 y yr_3 .

```
# Construcción de las rectas
# xr1 = np.linspace( ... )
# yr1 = ...
# ...
#### BEGIN SOLUTION
# Construcción de las rectas
xr1 = np.linspace(x1.min(), x1.max(), 10)
yr1 = m1 * xr1 + b1

xr2 = np.linspace(x2.min(), x2.max(), 10)
yr2 = m2 * xr2 + b2

xr3 = np.linspace(x3.min(), x3.max(), 10)
yr3 = m3 * xr3 + b3

file_answer.write('12', xr1, 'El arreglo xr1 no está bien construido.')
file_answer.write('13', yr1, 'El arreglo yr1 no está bien evaluado. Checa la fórmula.')
file_answer.write('14', xr2, 'El arreglo xr2 no está bien construido.')
file_answer.write('15', yr2, 'El arreglo yr2 no está bien evaluado. Checa la fórmula.')
file_answer.write('16', xr3, 'El arreglo xr3 no está bien construido.')
file_answer.write('17', yr3, 'El arreglo yr3 no está bien evaluado. Checa la fórmula.')

file_answer.to_file('q2')
#### END SOLUTION
```

El directorio `:/home/jovyan/HeCompA/.ans/03_AnalisisNumerico/` ya existe
Respuestas y retroalimentación almacenadas.

```
print(xr1, yr1, sep = '\n', end='\n\n')
print(xr2, yr2, sep = '\n', end='\n\n')
print(xr3, yr3, sep = '\n', end='\n\n')
```

```
[-25.52989816 -20.17129323 -14.81268829 -9.45408336 -4.09547843
 1.26312651  6.62173144 11.98033637 17.33894131 22.69754624]
[-3.42847293 -3.53577026 -3.64306758 -3.7503649 -3.85766222 -3.96495955
-4.07225687 -4.17955419 -4.28685151 -4.39414884]

[ 0.          211.11111111 422.22222222 633.33333333 844.44444444
1055.55555556 1266.66666667 1477.77777778 1688.88888889 1900.          ]
[-308.17427701 -489.93232915 -671.69038128 -853.44843341
-1035.20648554 -1216.96453767 -1398.7225898 -1580.48064193
-1762.23869406 -1943.99674619]

[ 0.          2.11111111 4.22222222 6.33333333 8.44444444 10.55555556
```

```
12.66666667 14.77777778 16.88888889 19.          ]
[-0.24601956 -0.23797629 -0.22993301 -0.22188973 -0.21384645 -0.20580317
-0.1977599 -0.18971662 -0.18167334 -0.17363006]
```

```
quizz.eval_numeric('12', xr1)
quizz.eval_numeric('13', yr1)
quizz.eval_numeric('14', xr2)
quizz.eval_numeric('15', yr2)
quizz.eval_numeric('16', xr3)
quizz.eval_numeric('17', yr3)
```

 12 | Tu resultado es correcto.

 13 | Tu resultado es correcto.

 14 | Tu resultado es correcto.

 15 | Tu resultado es correcto.

 16 | Tu resultado es correcto.

 17 | Tu resultado es correcto.

Si caculaste todo correctamente, entonces la siguiente celda de código graficará los cuatro conjuntos de datos junto con las líneas rectas que construiste.

```
axp = [dict(xlabel='$x_0$', ylabel='$y_0$'),
       dict(xlabel='$x_1$', ylabel='$y_1$'),
       dict(xlabel='$x_2$', ylabel='$y_2$'),
       dict(xlabel='$x_3$', ylabel='$y_3$', ylim=(-5,5))],

vis = mvis.Plotter(2,2,fig_par=dict(figsize=(10,5)), axis_par=axp)

vis.plot(1, xr0, yr0, c='C0', lw=2, ls = '--')
vis.scatter(1, x0, y0, fc='C0', ec='k', alpha=0.75)

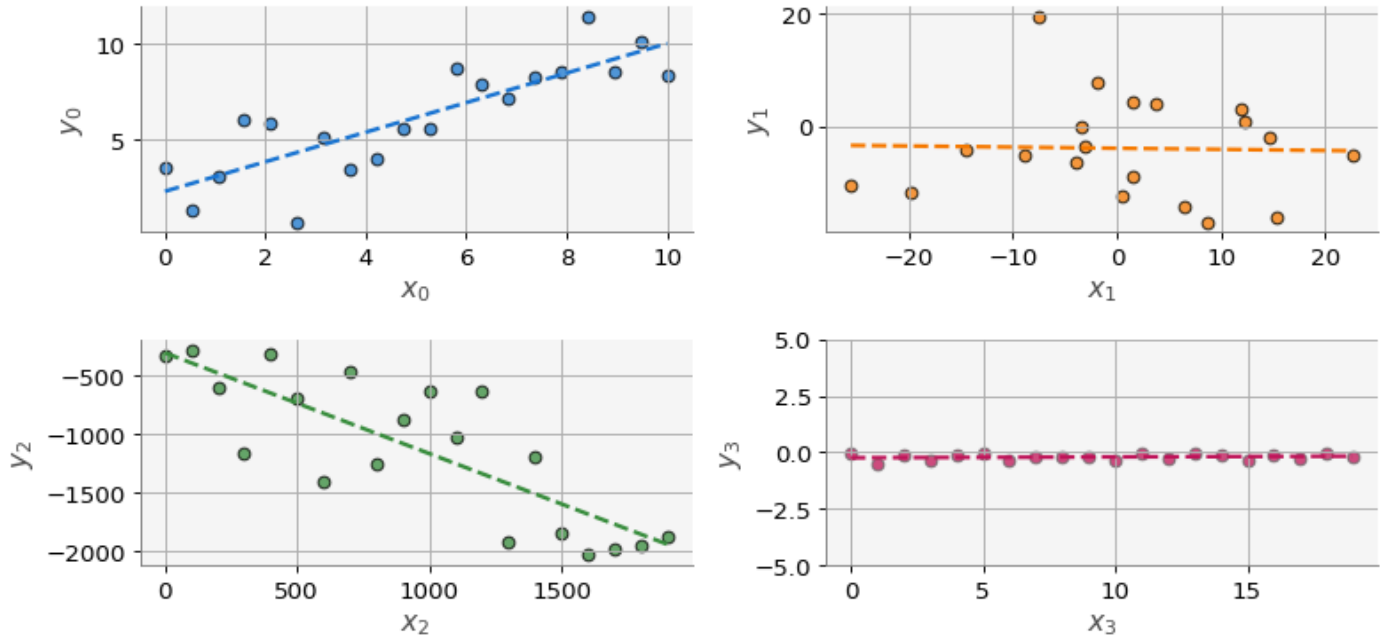
vis.plot(2, xr1, yr1, c='C1', lw=2, ls = '--')
vis.scatter(2, x1, y1, fc='C1', ec='k', alpha=0.75)

vis.plot(3, xr2, yr2, c='C2', lw=2, ls = '--')
```

```
vis.scatter(3, x2, y2, fc='C2', ec='k', alpha=0.75)

vis.plot(4, xr3, yr3, c='C3', lw=2, ls = '--')
vis.scatter(4, x3, y3, fc='C3', ec='dimgrey', alpha=0.75)

vis.tight_layout()
vis.grid()
vis.show()
```



10 Apéndice A: Deducción del método.

Dado el conjunto de datos:

| | | | | |
|-------|-------|-------|---------|-------|
| x_0 | x_1 | x_2 | \dots | x_N |
| y_0 | y_1 | y_2 | \dots | y_N |

lo que en principio desearíamos es que se cumpliera que:

$$y_i = mx_i + b \text{ para } i = 0, \dots, N$$

que es equivalente a

$$0 = mx_i + b - y_i \text{ para } i = 0, \dots, N, \quad (\text{A.1})$$

Pero la ecuación (A.1) no se cumple en general, de tal manera que lo que se pide es que las desviaciones de cada punto con respecto de la recta sean pequeñas.

En el caso de este método, la desviación se define como la diferencia del valor y_i con respecto de la recta elevada al cuadrado, es decir: $(mx_i + b - y_i)^2$. Y para calcular la desviación global se suman todas las diferencias, por

lo que obtenemos:

$$f(m, b) = \sum_{i=1}^N (mx_i + b - y_i)^2$$

Observa que del lado derecho hemos puesto $f(m, b)$ es decir, una función que depende de la pendiente m y de la ordenada al origen b .

El valor de la función f (la desviación global) depende de m y b ; entonces para encontrar los valores de m y b más adecuados, debemos minimizar f con respecto a esas variables.

Recordando nuestras clases de cálculo, sabemos que para minimizar una función, debemos calcular su derivada, igualarla a cero y resolver para encontrar los puntos críticos (máximos y mínimos). En este caso, debemos derivar con respecto a m y con respecto a b , y luego resolver un sistema de dos ecuaciones. Veamos como:

$$\frac{\partial f}{\partial m} = \frac{\partial}{\partial m} \left(\sum_{i=1}^N (mx_i + b - y_i)^2 \right) = 2 \sum_{i=1}^N x_i (mx_i + b - y_i) = 0 \quad (A.2)$$

$$\frac{\partial f}{\partial b} = \frac{\partial}{\partial b} \left(\sum_{i=1}^N (mx_i + b - y_i)^2 \right) = 2 \sum_{i=1}^N (mx_i + b - y_i) = 0 \quad (A.3)$$

De la ecuación (A.3) tenemos que:

$$m \sum_{i=1}^N x_i + \sum_{i=1}^N b - \sum_{i=1}^N y_i = 0$$

y despejando b obtenemos:

$$b = \underbrace{\frac{1}{N} \sum_{i=1}^N y_i}_{\bar{y}} - m \underbrace{\frac{1}{N} \sum_{i=1}^N x_i}_{\bar{x}} = \bar{y} - m\bar{x} \quad (A.4)$$

Ahora sustituimos (A.4) en (A.2) y obtenemos:

$$\sum_{i=1}^N x_i (mx_i + \bar{y} - m\bar{x} - y_i) = 0$$

Ahora despejamos m :

$$m = \frac{\sum_{i=1}^N x_i (y_i - \bar{y})}{\sum_{i=1}^N x_i (x_i - \bar{x})} \quad (A.5)$$

Las ecuaciones (A.4) y (A.5) proporcionan los valores de m y b de un punto crítico de la función $f(m, b)$. Falta demostrar que ese punto crítico es un mínimo. Para ello se deben calcular las derivadas segundas $\left(\frac{\partial}{\partial^2 m}, \frac{\partial}{\partial^2 b} \right)$,

$\frac{\partial}{\partial m \partial b}$) y ver que se cumplen los criterios necesarios.



1 El cuarteto de Anscombe

Objetivo. Visualizar el cuarteto de Anscombe usando herramientas de matplotlib, pandas y numpy, para demostrar las conclusiones del artículo del autor.

Anscombe, F. J. (1973). "Graphs in Statistical Analysis". The American Statistician. 27 (1): 17–21.
doi:10.2307/2682899. JSTOR 2682899.

[HeCompA - Anscombe](#) by [Luis M. de la Cruz](#) is licensed under [Attribution-ShareAlike 4.0 International](#)

Trabajo realizado con el apoyo del Programa UNAM-DGAPA-PAPIME PE101922

1.1 ¿Porqué Visualizar?

El proceso de transformar datos crudos en imágenes ayuda a mejorar las interpretaciones de grandes conjuntos de datos y eso permite obtener una perspectiva que podría pasarse por alto si se usarán solamente métodos estadísticos.

1.2 Ejemplo: Anscombe's quartet

Referencia en wikipedia: [Anscombe's quartet](#).

Consiste de cuatro conjuntos de datos que tienen las mismas propiedades estadísticas:

| Propiedad | Valor |
|------------------------------|---------------------|
| Media \bar{x} | 9 |
| Media \bar{y} | 7.50 |
| Varianza muestral s_x^2 | 11 |
| Varianza muestral s_y^2 | 4.125 |
| Correlación entre x y y | 0.816 |
| Regresión lineal | $y = 3.00 + 0.500x$ |
| Coef. de determinación R^2 | 0.67 |

Cada conjunto consiste de once puntos (x, y) y fueron contruidos por el estadístico F. J. Anscombe.

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import macti.visual
```

1.2.1 Paso 1.

Leer el archivo con la información y ponerla en un DataFrame

```
data = pd.read_csv('AnscombeQuartet.txt', sep='\t', header=None)
data
```

1.2.2 Paso 2.

Limpieza y organización de la información.

```
header = pd.MultiIndex.from_product([['Dataset 1', 'Dataset 2',
                                     'Dataset 3', 'Dataset 4'],
                                    ['x', 'y']],
                                    names=['dat', 'val'])

data.columns = header
data.index = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
data
```

1.2.3 Paso 3.

Calculo de algunas propiedades estadísticas.

```
data.mean(axis=0) # Calculamos la media de todos los conjuntos de datos
```

```
data.var(axis=0) # Calculamos la varianza muestral de todos los conjuntos de datos
```

```
data['Dataset 1'].corr() # Correlación
                        # Cambiar el número del dataset
```

1.2.4 Paso 4.

Realizamos una regresión lineal para ajustar una recta a los datos.

```
# Convertir los valores en un arreglo columna de numpy
X = data.iloc[:, 0].values.reshape(-1, 1)
X
```

La siguiente celda define una función para realizar el cálculo de la regresión lineal.

```
def regresionLineal(data, i):
    X = data.iloc[:, i].values.reshape(-1, 1) # Vector columna X
    Y = data.iloc[:, i+1].values.reshape(-1, 1) # Vector columna Y

    # if i == 4:
    #     X = np.delete(X, 2).reshape(-1, 1)
    #     Y = np.delete(Y, 2).reshape(-1, 1)
```

```

linear_regressor = LinearRegression() # Objeto para la regresión lineal
linear_regressor.fit(X, Y)            # Se realiza la regresión lineal
R2 = linear_regressor.score(X,Y)      # Coeficiente de determinación
m = linear_regressor.coef_            # Coeficientes de la regresión
b = linear_regressor.intercept_       # lineal y = mx + b

X_pred = np.arange(0,21,1)
X_pred.shape = (-1,1) # vector columna
Y_pred = linear_regressor.predict(X_pred) # Se realiza la predicción

return X, Y, X_pred, Y_pred, R2, m[0][0], b[0]

```

```

# Cálculo de la regresión para el Dataset 1
X, Y, X_pred, Y_pred, R2, m, b = regresionLineal(data, 4)
print('R2 = {:.3f} \t m = {:.3f} \t b = {:.3f}'.format(R2, m, b))

```

```

# Cálculo de la regresión para todos los Dataset's
for i in range(0,7,2):
    X, Y, X_pred, Y_pred, R2, m, b = regresionLineal(data, i)
    print('Dataset {:} : R2 = {:.3f} \t m = {:.3f} \t b = {:.3f}'.format(i//2+

```

¿Qué se puede decir de estos resultados?

1.2.5 Paso 5.

Graficación de los resultados.

```

import matplotlib.pyplot as plt
params = {'legend.fontsize': 16,
          'axes.labelsize':16,
          'axes.titlesize':16,
          'xtick.labelsize':16,
          'ytick.labelsize':16}
plt.rcParams.update(params)

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14,8))

num = 1
for a in axes:
    for ax in a:
        X, Y, X_pred, Y_pred, R2, m, b = regresionLineal(data, (num - 1)*2)
        ax.scatter(X, Y, marker = 'o', c='orange', s=75, edgecolor='red')
        leyenda = 'R2 : {:.3f}, m : {:.3f}, b : {:.3f}'.format(R2, m, b)
        ax.plot(X_pred, Y_pred, lw=2.0, label=leyenda)
        ax.set_xlim(2,20)
        ax.set_aspect(aspect=1.0)
        ax.set_xlabel('x')
        ax.set_ylabel('y')

```

```
ax.set_title('Dataset {}'.format(num))
ax.legend()
num += 1
plt.tight_layout()
#plt.savefig('anscombe.png', dpi=300)
```

¿Qué puede decir de estos gráficos

- Gráfica del Dataset 1: relación lineal simple entre dos variables correlacionadas.
- Gráfica del Dataset 2: se observa una relación entre x y y pero no parece ser lineal.
- Gráfica del Dataset 3: relación lineal pero la regresión obtenida se ve afectada por el dato extremo que influye en el resultado final y altera el coeficiente de correlación de 1 a 0.816.
- Gráfica del Dataset 4: se muestra como un valor atípico es suficiente para producir un coeficiente de correlación alto, aún cuando la relación entre las variables no es lineal.

Este cuarteto es usado todavía en la actualidad para ilustrar la importancia de graficar los datos antes de realizar cualquier análisis estadístico. También se muestra el efecto de los valores atípicos.

La intención fue cambiar la impresión de que **“los cálculos numéricos son exactos, pero los gráficos aproximados”**.

[Edward Tufte](#) usó el cuarteto en la primera página del primer capítulo de su libro [The Visual Display of Quantitative Information](#), para enfatizar la importancia de mirar los datos antes de analizarlos.



4 Análisis de la tasa de fertilidad total mundial

This notebook by Luis M. de la Cruz Salas is licensed under [Attribution-NonCommercial-NoDerivatives 4.0 International](#)

Para lograr una excelente visualización de datos se debe tener: **mucha curiosidad** e interés en **muchos temas variados** que pueden parecer poco relacionados entre ellos: matemáticas, demografía, epidemiología, economía, deportes, ventas por internet, historia, psicología y muchos etcéteras.



Durante el proceso de esta visualización la vida tenderá a convertirse en un **caos intelectual**, pero **sistemático y emocionante**.

La **visualización de datos** es muy importante, pues es posible que sea **lo único que vean tus interlocutores**: cliente, colega, jefe, tutor, jurado, lectores de un periódico, público en una conferencia, ... y probablemente a ellos no les interese mucho los datos numéricos o los algoritmos usados en su análisis.

El público a quien deseas transmitir tus hallazgos olvidarán muy pronto los números; **pero si introduces tus hallazgos mediante un relato que cuente la historia de los datos, es posible que ellos se lleven un buen sabor de boca y recuerden la información recibida por mucho más tiempo e incluso tomen acciones.**

4.1 Objetivo

- Obtener información de <http://data.un.org/> de las siguientes variables:
 - Tasa de fertilidad total (TFR)
 - Ingreso per cápita (GDP)
 - Nivel de educación (EDU)
 - Esperanza de vida al nacer (LEB)
- Construir una visualización que permita comparar el cambio en el Total Fertility Rate (TFR) en función del tiempo para varios países.
- Generar un data set para el índice de desarrollo humano (HDI) de acuerdo con:
https://en.wikipedia.org/wiki/Human_Development_Index

[HeCompA - Anscombe](#) by [Luis M. de la Cruz](#) is licensed under [Attribution-ShareAlike 4.0 International](#)   

Trabajo realizado con el apoyo del Programa UNAM-DGAPA-PAPIME PE101922

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from math import ceil
```

```
# Esta biblioteca solo funciona en la plataforma www.macti.unam.mx
```

```
# Si no estas en esa plataforma, solo comenta la línea que sigue.
import macti.visual
```

5 Conjuntos de datos.

Los datos ya están en el directorio de esta notebook. Solo los leemos.

```
TFR = pd.read_csv('UNdata_Export_20230621_205538168.zip')
# GPD per capita
GDP = pd.read_csv('UNdata_Export_20230624_011417717.zip')
# Life expectancy at birth
LEB = pd.read_csv('UNdata_Export_20230624_013747471.zip')
# Gross enrolment ratio. Tertiary education
EDU = pd.read_csv('UNdata_Export_20230624_014454110.zip')
```

```
EDU.head()
```

| | Reference Area | Time Period | Sex | Age group | Units of measurement | Observation Value |
|---|----------------|-------------|-------------|----------------|----------------------|-------------------|
| 0 | Afghanistan | 2014 | Female | Not applicable | Percent | 3.67329 |
| 1 | Afghanistan | 2014 | Male | Not applicable | Percent | 13.28657 |
| 2 | Afghanistan | 2014 | All genders | Not applicable | Percent | 8.66280 |
| 3 | Afghanistan | 2011 | Female | Not applicable | Percent | 1.89233 |
| 4 | Afghanistan | 2011 | All genders | Not applicable | Percent | 3.75598 |

```
print(TFR.columns)
print(GDP.columns)
print(LEB.columns)
print(EDU.columns)
```

```
Index(['Country or Area', 'Year(s)', 'Variant', 'Value'], dtype='object')
Index(['Country or Area', 'Year', 'Item', 'Value'], dtype='object')
Index(['Country or Area', 'Year(s)', 'Variant', 'Value'], dtype='object')
Index(['Reference Area', 'Time Period', 'Sex', 'Age group',
      'Units of measurement', 'Observation Value'],
      dtype='object')
```

Agrupamos los datos por países y regiones geográficas.

```
TFR_group = TFR.groupby('Country or Area')
GDP_group = GDP.groupby('Country or Area')
LEB_group = LEB.groupby('Country or Area')
EDU_group = EDU.groupby('Reference Area')
```

Determinamos los países que se tienen en cada conjunto de datos

```
TFR_countries = TFR_group.groups.keys()
GDP_countries = GDP_group.groups.keys()
LEB_countries = LEB_group.groups.keys()
EDU_countries = EDU_group.groups.keys()
```

```
print(len(TFR_countries))
print(len(GDP_countries))
print(len(LEB_countries))
print(len(EDU_countries))
```

284

220

284

220

```
print(TFR_countries)
```

```
dict_keys(['Afghanistan', 'Africa', 'Albania', 'Algeria', 'American Samoa', 'Andorra',
'Angola', 'Anguilla', 'Antigua and Barbuda', 'Argentina', 'Armenia', 'Aruba', 'Asia',
'Australia', 'Australia/New Zealand', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bermuda', 'Bhutan',
'Bolivia (Plurinational State of)', 'Bonaire, Sint Eustatius and Saba', 'Bosnia and
Herzegovina', 'Botswana', 'Brazil', 'British Virgin Islands', 'Brunei Darussalam',
'Bulgaria', 'Burkina Faso', 'Burundi', 'Cabo Verde', 'Cambodia', 'Cameroon', 'Canada',
'Caribbean', 'Cayman Islands', 'Central African Republic', 'Central America', 'Central
Asia', 'Central and Southern Asia', 'Chad', 'Chile', 'China', 'China, Hong Kong SAR',
'China, Macao SAR', 'Colombia', 'Comoros', 'Congo', 'Cook Islands', 'Costa Rica',
'Croatia', 'Cuba', 'Curaçao', 'Cyprus', 'Czechia', 'Côte d'Ivoire', 'Dem. People's
Republic of Korea', 'Democratic Republic of the Congo', 'Denmark', 'Djibouti',
'Dominica', 'Dominican Republic', 'Eastern Africa', 'Eastern Asia', 'Eastern Europe',
'Eastern and South-Eastern Asia', 'Ecuador', 'Egypt', 'El Salvador', 'Equatorial Guinea',
'Eritrea', 'Estonia', 'Eswatini', 'Ethiopia', 'Europe', 'Europe and Northern America',
'Falkland Islands (Malvinas)', 'Faroe Islands', 'Fiji', 'Finland', 'France', 'French
Guiana', 'French Polynesia', 'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana',
'Gibraltar', 'Greece', 'Greenland', 'Grenada', 'Guadeloupe', 'Guam', 'Guatemala',
'Guernsey', 'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti', 'High-income countries', 'Holy
See', 'Honduras', 'Hungary', 'Iceland', 'India', 'Indonesia', 'Iran (Islamic Republic
of)', 'Iraq', 'Ireland', 'Isle of Man', 'Israel', 'Italy', 'Jamaica', 'Japan', 'Jersey',
'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati', 'Kosovo (under UNSC res. 1244)', 'Kuwait',
'Kyrgyzstan', 'Land-locked Developing Countries (LLDC)', 'Lao People's Democratic
Republic', 'Latin America and the Caribbean', 'Latvia', 'Least developed countries',
'Lebanon', 'Lesotho', 'Less developed regions', 'Less developed regions, excluding
China', 'Less developed regions, excluding least developed countries', 'Liberia',
'Libya', 'Liechtenstein', 'Lithuania', 'Low-income countries', 'Lower-middle-income
countries', 'Luxembourg', 'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali',
'Malta', 'Marshall Islands', 'Martinique', 'Mauritania', 'Mauritius', 'Mayotte',
'Melanesia', 'Mexico', 'Micronesia', 'Micronesia (Fed. States of)', 'Middle Africa',
'Middle-income countries', 'Monaco', 'Mongolia', 'Montenegro', 'Montserrat', 'More
developed regions', 'Morocco', 'Mozambique', 'Myanmar', 'Namibia', 'Nauru', 'Nepal',
```



```
'Netherlands', 'New Caledonia', 'New Zealand', 'Nicaragua', 'Niger', 'Nigeria', 'Niue',
'No income group available', 'North Macedonia', 'Northern Africa', 'Northern Africa and
Western Asia', 'Northern America', 'Northern Europe', 'Northern Mariana Islands',
'Norway', 'Oceania', 'Oceania (excluding Australia and New Zealand)', 'Oman', 'Other non-
specified areas', 'Pakistan', 'Palau', 'Panama', 'Papua New Guinea', 'Paraguay', 'Peru',
'Philippines', 'Poland', 'Polynesia', 'Portugal', 'Puerto Rico', 'Qatar', 'Republic of
Korea', 'Republic of Moldova', 'Romania', 'Russian Federation', 'Rwanda', 'RÃ©union',
'Saint BarthÃ©lemy', 'Saint Helena', 'Saint Kitts and Nevis', 'Saint Lucia', 'Saint
Martin (French part)', 'Saint Pierre and Miquelon', 'Saint Vincent and the Grenadines',
'Samoa', 'San Marino', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
'Seychelles', 'Sierra Leone', 'Singapore', 'Sint Maarten (Dutch part)', 'Slovakia',
'Slovenia', 'Small Island Developing States (SIDS)', 'Solomon Islands', 'Somalia', 'South
Africa', 'South America', 'South Sudan', 'South-Eastern Asia', 'Southern Africa',
'Southern Asia', 'Southern Europe', 'Spain', 'Sri Lanka', 'State of Palestine', 'Sub-
Saharan Africa', 'Sudan', 'Suriname', 'Sweden', 'Switzerland', 'Syrian Arab Republic',
'Tajikistan', 'Thailand', 'Timor-Leste', 'Togo', 'Tokelau', 'Tonga', 'Trinidad and
Tobago', 'Tunisia', 'Turkmenistan', 'Turks and Caicos Islands', 'Tuvalu', 'TÃ¼rkiye',
'Uganda', 'Ukraine', 'United Arab Emirates', 'United Kingdom', 'United Republic of
Tanzania', 'United States Virgin Islands', 'United States of America', 'Upper-middle-
income countries', 'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela (Bolivarian Republic
of)', 'Viet Nam', 'Wallis and Futuna Islands', 'Western Africa', 'Western Asia', 'Western
Europe', 'Western Sahara', 'World', 'Yemen', 'Zambia', 'Zimbabwe']])
```

Determinamos los paÃ­ses que son comunes a todos los conjuntos de datos.

```
filtra_GDP = lambda p: p in GDP_countries
filtra_LEB = lambda p: p in LEB_countries
filtra_EDU = lambda p: p in EDU_countries
```

```
len(list(filter(filtra_EDU, list(filter(filtra_GDP, list(filter(filtra_LEB, TFR_c
```

181

```
countries = list(filter(filtra_EDU, list(filter(filtra_GDP, list(filter(filtra_LEB,
```

```
print(countries)
```

```
['Afghanistan', 'Albania', 'Algeria', 'Angola', 'Anguilla', 'Antigua and Barbuda',
'Argentina', 'Armenia', 'Aruba', 'Australia', 'Austria', 'Azerbaijan', 'Bahamas',
'Bahrain', 'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bermuda',
'Bhutan', 'Bosnia and Herzegovina', 'Botswana', 'Brazil', 'British Virgin Islands',
'Brunei Darussalam', 'Bulgaria', 'Burkina Faso', 'Burundi', 'Cambodia', 'Cameroon',
'Canada', 'Central African Republic', 'Chad', 'Chile', 'Colombia', 'Comoros', 'Congo',
'Cook Islands', 'Costa Rica', 'Croatia', 'Cuba', 'Cyprus', 'Democratic Republic of the
Congo', 'Denmark', 'Djibouti', 'Dominica', 'Dominican Republic', 'Ecuador', 'Egypt', 'El
Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia', 'Ethiopia', 'Fiji', 'Finland',
'France', 'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana', 'Greece', 'Grenada',
'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras', 'Hungary',
```

```
'Iceland', 'India', 'Indonesia', 'Iraq', 'Ireland', 'Israel', 'Italy', 'Jamaica',
'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati', 'Kuwait', 'Kyrgyzstan', 'Lao
People's Democratic Republic', 'Latvia', 'Lebanon', 'Lesotho', 'Liberia',
'Liechtenstein', 'Lithuania', 'Luxembourg', 'Madagascar', 'Malawi', 'Malaysia',
'Maldives', 'Mali', 'Malta', 'Marshall Islands', 'Mauritania', 'Mauritius', 'Mexico',
'Monaco', 'Mongolia', 'Montenegro', 'Montserrat', 'Morocco', 'Mozambique', 'Myanmar',
'Namibia', 'Nauru', 'Nepal', 'Netherlands', 'New Zealand', 'Nicaragua', 'Niger',
'Nigeria', 'Norway', 'Oman', 'Pakistan', 'Palau', 'Panama', 'Papua New Guinea',
'Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal', 'Puerto Rico', 'Qatar',
'Republic of Korea', 'Romania', 'Russian Federation', 'Rwanda', 'Saint Kitts and Nevis',
'Saint Lucia', 'Saint Vincent and the Grenadines', 'Samoa', 'San Marino', 'Sao Tome and
Principe', 'Saudi Arabia', 'Senegal', 'Serbia', 'Seychelles', 'Sierra Leone', 'Slovakia',
'Slovenia', 'Solomon Islands', 'Somalia', 'South Africa', 'Spain', 'Sri Lanka', 'Sudan',
'Suriname', 'Sweden', 'Switzerland', 'Syrian Arab Republic', 'Tajikistan', 'Thailand',
'Timor-Leste', 'Togo', 'Tonga', 'Trinidad and Tobago', 'Tunisia', 'Turkmenistan',
'Tuvalu', 'Uganda', 'Ukraine', 'United Arab Emirates', 'Uruguay', 'Uzbekistan',
'Vanuatu', 'Venezuela (Bolivarian Republic of)', 'Viet Nam', 'Yemen', 'Zambia',
'Zimbabwe']
```

Determinamos los años para el caso del TFR.

```
years = list(TFR_group.get_group('Afghanistan')['Year(s)'])
print(years)
```

```
[2101, 2100, 2099, 2098, 2097, 2096, 2095, 2094, 2093, 2092, 2091, 2090, 2089, 2088,
2087, 2086, 2085, 2084, 2083, 2082, 2081, 2080, 2079, 2078, 2077, 2076, 2075, 2074, 2073,
2072, 2071, 2070, 2069, 2068, 2067, 2066, 2065, 2064, 2063, 2062, 2061, 2060, 2059, 2058,
2057, 2056, 2055, 2054, 2053, 2052, 2051, 2050, 2049, 2048, 2047, 2046, 2045, 2044, 2043,
2042, 2041, 2040, 2039, 2038, 2037, 2036, 2035, 2034, 2033, 2032, 2031, 2030, 2029, 2028,
2027, 2026, 2025, 2024, 2023, 2022, 2021, 2020, 2019, 2018, 2017, 2016, 2015, 2014, 2013,
2012, 2011, 2010, 2009, 2008, 2007, 2006, 2005, 2004, 2003, 2002, 2001, 2000, 1999, 1998,
1997, 1996, 1995, 1994, 1993, 1992, 1991, 1990, 1989, 1988, 1987, 1986, 1985, 1984, 1983,
1982, 1981, 1980, 1979, 1978, 1977, 1976, 1975, 1974, 1973, 1972, 1971, 1970, 1969, 1968,
1967, 1966, 1965, 1964, 1963, 1962, 1961, 1960, 1959, 1958, 1957, 1956, 1955, 1954, 1953,
1952, 1951, 1950]
```

5.1 Visualización exploratoria.

```
ax1 = plt.subplot(221)
TFR.plot(x='Year(s)', y='Value', kind='scatter', marker='.', fc='C0', alpha=0.5,
ax1.set_title('TFR')

ax2 = plt.subplot(222)
GDP.plot(x='Year', y='Value', kind='scatter', marker='.', fc='C1', alpha=0.5, ax=
ax2.set_title('GDP')

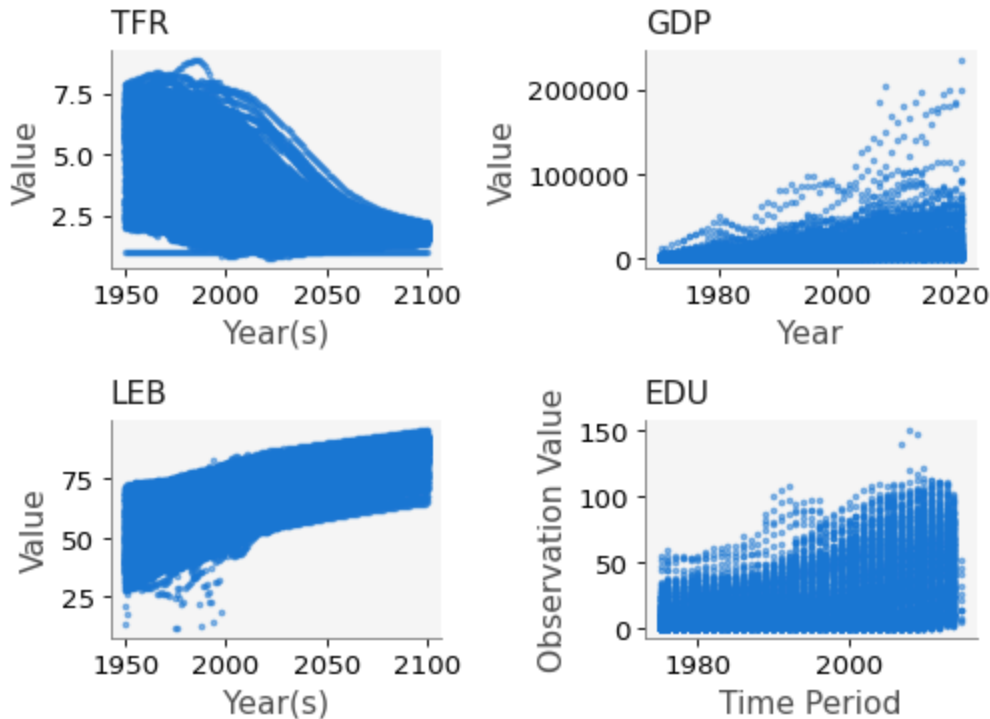
ax3 = plt.subplot(223)
LEB.plot(x='Year(s)', y='Value', kind='scatter', marker='.', fc='C2', alpha=0.5,
ax3.set_title('LEB')
```

```

ax4 = plt.subplot(224)
EDU.plot(x='Time Period', y='Observation Value', kind='scatter', marker='.', fc='
ax4.set_title('EDU')

plt.tight_layout()
plt.savefig('TFR_01.png', dpi=300)
plt.show()

```



5.2 Visualización del TFR.

Las siguientes funciones son de utilidad para la visualización final.

```

from math import ceil

def maxmin(data, time, value, country):
    """
    Calcula el valor máximo y el mínimo de todos los países.

    Parameters
    -----
    data : DataFrame
        Dataframe con la información de los países.

    time: str
        Nombre de la columna con la información de los años.

    value: str
        Nombre de la columna con la información de los datos.

```

```

country: str
    Nombre de la columna con los nombres de los países.

Returns
-----
p_max, y_max, p_min, y_min, yticks
"""
# Se obtiene el valor máximo del time
x_max = data[time].max()
x_min = data[time].min()

# Se obtiene el valor máximo del value
y_max = data[value].max()

# Extrae el nombre del país con el valor máximo
p_max = data[data[value] == y_max].iloc[0][country]

# Se obtiene el valor mínimo
y_min = data[value].min()

# Extrae el nombre del país con el valor mínimo
p_min = data[data[value] == y_min].iloc[0][country]

return p_max, y_max, p_min, y_min, x_min, x_max

def set_canvas(data, time, value, country, figsize, xstep=10, ystep = 1):
    """
    Genera un lienzo para crear las gráficas sobre él.

    Parameters
    -----
    data : DataFrame
        Dataframe con la información de los países.

    time: str
        Nombre de la columna con la información de los años.

    value: str
        Nombre de la columna con la información de los datos.

    country: str
        Nombre de la columna con los nombres de los países.

    figsize: tuple
        Tamaño de la figura.

    xstep, ystep: int
        Paso de los ticks en los ejes horizontal y vertical, respectivamente.

    Returns
    """

```

```

-----
fig, ax:
    Figura y ejes donde se hará la gráfica.
"""

p_max, y_max, p_min, y_min, x_min, x_max = maxmin(data, time, value, country)
print('Máximo = {}, \t País : {}'.format(y_max, p_max))
print('Mínimo = {}, \t País : {}'.format(y_min, p_min))

# Se generan los yticks
yticks = [i for i in range(0, ceil(y_max)+1, ystep)]

fig = plt.figure(figsize=figsize)
ax = plt.gca()

if not data.empty:
    data.plot(x=time, y=value, color='lightgray', rot = 70, xlabel='', lw = 0.5)

    ax.spines.top.set_visible(False)
    ax.spines.bottom.set_visible(False)
    ax.spines.left.set_visible(False)
    ax.spines.right.set_visible(False)
    ax.set_ylim(y_min, y_max)
    ax.set_yticks(yticks)
    ax.set_xticks([a for a in range(x_min, x_max+1, xstep)])
    ax.grid(lw=0.5, color='gainsboro')

    return fig, ax

def plot_country(ax, country, time='Year(s)', value = 'Value', color='gray', label='')
    """
    Dibuja la curva de un país.

    Parameters
    -----
    ax : Axis
        Ejes donde se hará el gráfico.

    time: str
        Nombre de la columna con la información de los años.

    value: str
        Nombre de la columna con la información de los datos.

    country: str
        Nombre de la columna con los nombres de los países.

    color: str
        Color de la curva.

    label: str
        Etiqueta para la curva.

```

```

maxim:
    Límite para graficar una línea continua, a partir de este
    límite se dibuja la línea punteada.
.....
x = country[time][country[time]>=maxim-1]
y = country[value][country[time]>=maxim-1]
ax.plot(x, y, c=color, ls = '--', lw = 0.75)

x = country[time][country[time]<maxim]
y = country[value][country[time]<maxim]
ax.plot(x, y, c=color, ls = '-', lw = 2.0, label=label)

```

```
TFR_group.get_group('Yemen').dropna()
```

| | Country or Area | Year(s) | Variant | Value |
|-------|-----------------|---------|---------|--------|
| 43017 | Yemen | 2100 | Medium | 1.8205 |
| 43018 | Yemen | 2099 | Medium | 1.8224 |
| 43019 | Yemen | 2098 | Medium | 1.8230 |
| 43020 | Yemen | 2097 | Medium | 1.8345 |
| 43021 | Yemen | 2096 | Medium | 1.8387 |
| ... | ... | ... | ... | ... |
| 43163 | Yemen | 1954 | Medium | 7.8979 |
| 43164 | Yemen | 1953 | Medium | 7.8988 |
| 43165 | Yemen | 1952 | Medium | 7.8836 |
| 43166 | Yemen | 1951 | Medium | 7.8959 |
| 43167 | Yemen | 1950 | Medium | 7.9156 |

151 rows × 4 columns

```

fig, ax = set_canvas(TFR, 'Year(s)', 'Value', 'Country or Area', (10,7))

ax.plot([years[-1], years[0]], [2.1, 2.1], c='dimgray', ls = 'solid', lw=0.75)
ax.plot([years[-1], years[0]], [2.1, 2.1], c='dimgray', ls = 'solid', alpha=0.25, lw=2)

ax.set_title('Promedio de número de hijos por mujer', loc='left', color='gray', fontstyle='italic')
ax.set_title('fuente: http://data.un.org', loc='right', color='gray', fontstyle='italic')
plt.suptitle('Evolución del TFR (Total Fertility Rate)', y = 0.96, color='black', fontstyle='italic')
ax.annotate('Nivel de \n reemplazo \n promedio = 2.1',
            xy=(2090, 2.095), xytext=(2090, 4.0),
            bbox=dict(boxstyle='round', facecolor='whitesmoke', edgecolor='gray'),
            arrowprops=dict(arrowstyle='->', facecolor='dimgray', edgecolor='dimgray',
                            fontsize=8, color='black', horizontalalignment='center'))

```

```

c_to_plot = [('Niger', 'Nigeria', 'darkred'),
              ('Central African Republic', 'República Centroafricana', 'red'),
              ('Burundi', 'Burundi', 'orangered'),
              ('Malawi', 'Malawi', 'darkorange'),
              ('Yemen', 'Yemen', 'limegreen'),
              ('Mexico', 'México', 'forestgreen'),
              ('Cuba', 'Cuba', 'dodgerblue'),
              ('Liechtenstein', 'Liechtenstein', 'royalblue'),
              ('Japan', 'Japón', 'mediumblue'),
              ('World', 'Promedio Mundial', 'black'),
              ('Mongolia', 'Mongolia', 'darkorange'),
              ('India', 'India', 'limegreen'),
              ('United States of America', 'USA', 'deepskyblue'),
              ('China', 'China', 'mediumblue'),
              ('Spain', 'España', 'purple'),
              ('Republic of Korea', 'Corea del sur', 'crimson'),
              ('Holy See', 'Ciudad del Vaticano', 'olivedrab'),
              ]

for c in c_to_plot:
    c_tfr = TFR_group.get_group(c[0]).dropna()
    plot_country(ax, c_tfr, color=c[2])
    ytext = c_tfr['Value'][c_tfr['Year(s)'] == 2020].values[0]
    ytext_i = c_tfr['Value'][c_tfr['Year(s)'] == 1950].values[0]

    if c[0] == 'Mexico':
        ytext_i -= 0.1
    if c[0] == 'Malawi':
        ytext_i -= 0.03

    plt.text(1945, ytext_i, "{:.2f}".format(ytext_i), color = c[2], fontsize=8, f

xy_x = 2020
xytext_x = 2040

if c[0] == 'World':
    yoff = 0.6
elif c[0] == 'Mongolia':
    yoff = 0.6
elif c[0] == 'India':
    yoff = 0.4
elif c[0] == 'United States of America':
    yoff = -0.2
elif c[0] == 'Republic of Korea':
    yoff = -0.7
elif c[0] == 'Mexico':
    yoff = -0.05
elif c[0] == 'Malawi':
    yoff = 0.6
elif c[0] == 'Yemen':
    yoff = 0.0

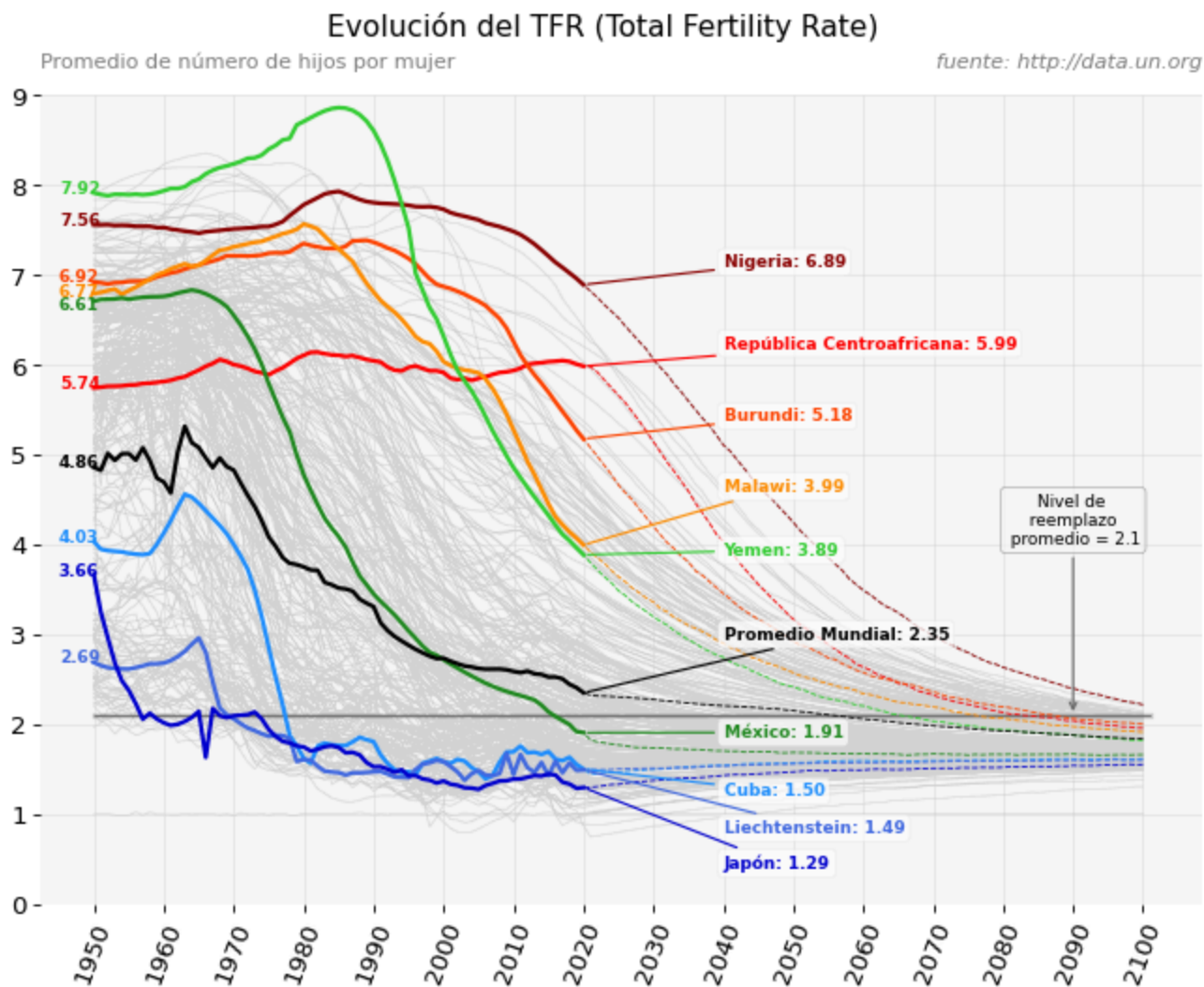
```

```
elif c[0] == 'Japan':
    yoff = -0.9
elif c[0] == 'Cuba':
    yoff = -0.3
elif c[0] == 'Liechtenstein':
    yoff = -0.7
elif c[0] == 'Holy See':
    xy_x = 1980
    xytext_x = 1965
    yoff = -0.5
else:
    yoff = 0.2

plt.annotate(c[1]+" {:.2f}".format(ytext), xy = (xy_x, ytext), xytext = (xytext_x, ytext),
             color = c[2], fontsize=8, fontweight='bold',
             bbox=dict(boxstyle='round', fc='white', ec='gainsboro', alpha=0.5),
             arrowprops=dict(arrowstyle='-', color=c[2]))

plt.savefig('TFR.png', dpi=300)
plt.show()
```

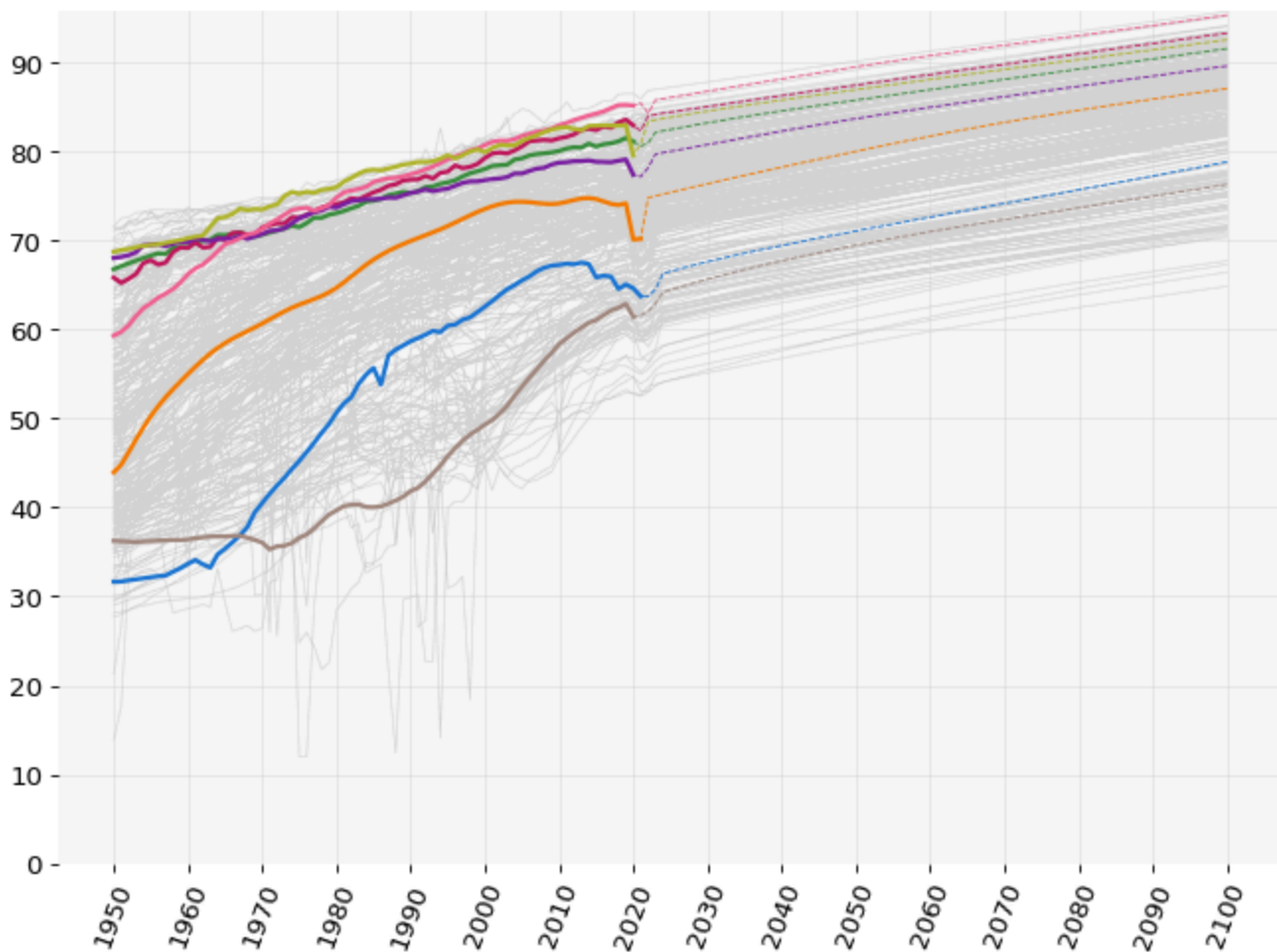
Máximo = 8.8637, País : Yemen
Mínimo = 0.7455, País : China, Hong Kong SAR



5.3 Exploración de los otros datos.

```
fig, ax = set_canvas(LEB, 'Year(s)', 'Value', 'Country or Area', (10,7), ystep=10)
plot_country(ax, LEB_group.get_group('Yemen').dropna(), color='C0', label='Yemen')
plot_country(ax, LEB_group.get_group('Mexico').dropna(), color='C1', label='México')
plot_country(ax, LEB_group.get_group('Germany').dropna(), color='C2', label='Alemania')
plot_country(ax, LEB_group.get_group('Holy See').dropna(), color='C3', label='El Vaticano')
plot_country(ax, LEB_group.get_group('United States of America').dropna(), color='C4', label='Estados Unidos')
plot_country(ax, LEB_group.get_group('Niger').dropna(), color='C5', label='Nigeria')
plot_country(ax, LEB_group.get_group('China, Hong Kong SAR').dropna(), color='C6', label='China')
plot_country(ax, LEB_group.get_group('San Marino').dropna(), color='C8', label='San Marino')
```

Máximo = 95.7552, País : Monaco
 Mínimo = 11.9951, País : Cambodia

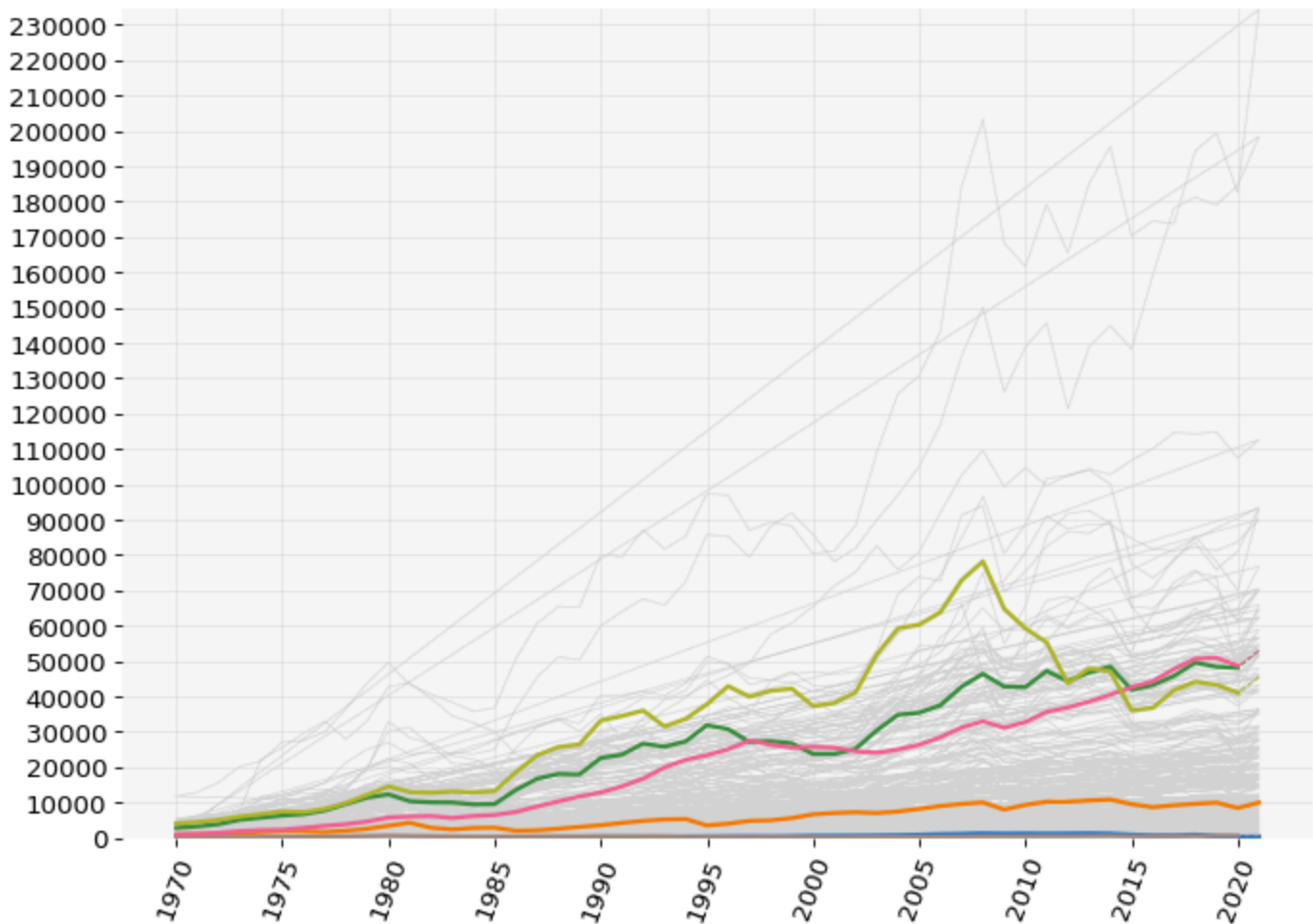


```
maxmin(GDP, 'Year', 'Value', 'Country or Area')
```

```
('Monaco', 234317.084818276, 'Viet Nam', 34.1125600868106, 1970, 2021)
```

```
fig, ax = set_canvas(GDP, 'Year', 'Value', 'Country or Area', (10,7), xstep=5, ystep=10)
plot_country(ax, GDP_group.get_group('Yemen').dropna(), time = 'Year', color='C0')
plot_country(ax, GDP_group.get_group('Mexico').dropna(), time = 'Year', color='C1')
plot_country(ax, GDP_group.get_group('Germany').dropna(), time = 'Year', color='C2')
#plot_country(ax, GDP_group.get_group('Holy See').dropna(), time = 'Year', color='C3')
#plot_country(ax, GDP_group.get_group('United States of America').dropna(), time = 'Year', color='C4')
plot_country(ax, GDP_group.get_group('Niger').dropna(), time = 'Year', color='C5')
plot_country(ax, GDP_group.get_group('China, Hong Kong SAR').dropna(), time = 'Year', color='C6')
plot_country(ax, GDP_group.get_group('San Marino').dropna(), time = 'Year', color='C7')
```

Máximo = 234317.084818276, País : Monaco
 Mínimo = 34.1125600868106, País : Viet Nam



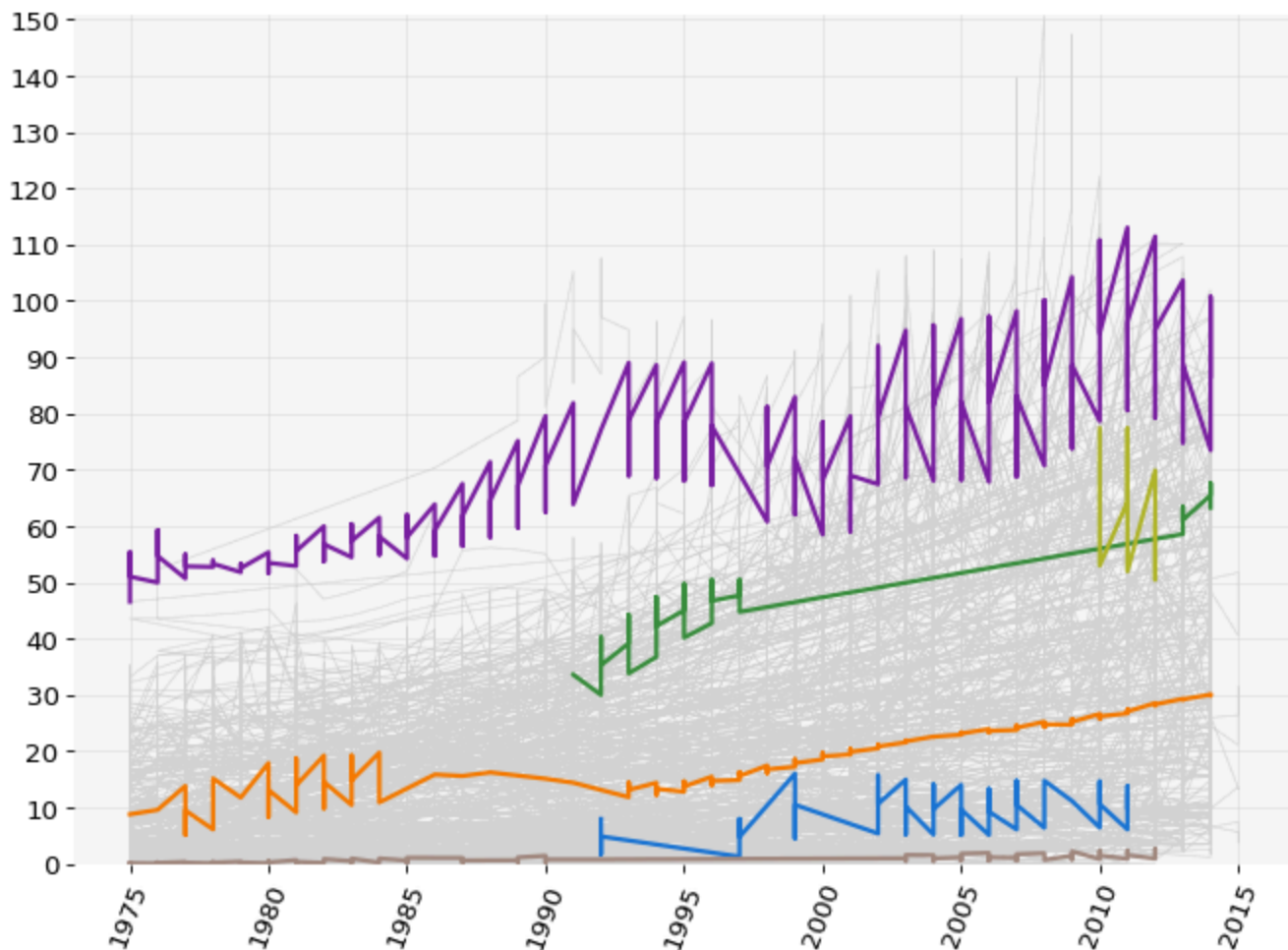
```
maxmin(EDU, 'Time Period', 'Observation Value', 'Reference Area')
```

```
('Cuba', 150.70758, 'Anguilla', 0.0, 1975, 2015)
```

```
fig, ax = set_canvas(EDU, 'Time Period', 'Observation Value', 'Reference Area', (
    plot_country(ax, EDU_group.get_group('Yemen').dropna(), time = 'Time Period', val
    plot_country(ax, EDU_group.get_group('Mexico').dropna(), time = 'Time Period', va
    plot_country(ax, EDU_group.get_group('Germany').dropna(), time = 'Time Period', v
    #plot_country(ax, EDU_group.get_group('Holy See').dropna(), time = 'Time Period',
    plot_country(ax, EDU_group.get_group('United States of America').dropna(), time =
    plot_country(ax, EDU_group.get_group('Niger').dropna(), time = 'Time Period', val
    #plot_country(ax, EDU_group.get_group('China, Hong Kong SAR').dropna(), time = 'T
    plot_country(ax, EDU_group.get_group('San Marino').dropna(), time = 'Time Period'
```

Máximo = 150.70758, País : Cuba

Mínimo = 0.0, País : Anguilla



5.4 Construcción del HDI

Se extrae información para el año 2010.

```
countries = list(filter(filtra_EDU, list(filter(filtra_GDP, list(filter(filtra_LE
tfr = pd.Series(dtype=float)
gdp = pd.Series(dtype=float)
edu = pd.Series(dtype=float)
leb = pd.Series(dtype=float)
final_countries = []
año = 2010

países_to_remove = ['Malaysia', 'British Virgin Islands', 'Cook Islands', 'Monaco']
for p in países_to_remove:
    countries.remove(p)

for i, c in enumerate(countries):

    c_g = EDU_group.get_group(c).groupby('Sex').get_group('Female')
    if año in c_g['Time Period'].values:
        final_countries.append(c)
```

```

edu = pd.concat([edu, c_g[c_g['Time Period'] == año]['Observation Value']]

c_g = GDP_group.get_group(c)
gdp = pd.concat([gdp, c_g[c_g['Year'] == año]['Value']])

c_g = TFR_group.get_group(c)
tfr = pd.concat([tfr, c_g[c_g['Year(s)'] == año]['Value']])

c_g = LEB_group.get_group(c)
leb = pd.concat([leb, c_g[c_g['Year(s)'] == año]['Value']])

```

```
print(len(tfr), len(gdp), len(edu), len(leb))
```

118 118 118 118

Se construye el dataframe con todas las variables para los países donde se tiene información.

```

hdi = pd.DataFrame()
hdi['País'] = final_countries
hdi['TFR'] = list(tfr)
hdi['GDP'] = list(gdp)
hdi['EDU'] = list(edu)
hdi['LEB'] = list(leb)
hdi

```

| | País | TFR | GDP | EDU | LEB |
|-----|---------------------|--------|--------------|----------|---------|
| 0 | Albania | 1.6564 | 4052.927488 | 52.38337 | 77.9359 |
| 1 | Algeria | 2.8434 | 4493.635433 | 35.33614 | 73.8081 |
| 2 | Antigua and Barbuda | 1.7854 | 13038.258604 | 22.67580 | 76.8195 |
| 3 | Argentina | 2.3462 | 10023.206688 | 89.15845 | 75.7208 |
| 4 | Armenia | 1.5010 | 3462.399904 | 63.13266 | 73.1597 |
| ... | ... | ... | ... | ... | ... |
| 113 | Uruguay | 2.0110 | 11567.565434 | 80.32340 | 76.8580 |
| 114 | Uzbekistan | 2.4407 | 1789.707894 | 7.55595 | 69.2354 |
| 115 | Viet Nam | 1.8949 | 1631.765100 | 22.71635 | 73.5126 |
| 116 | Yemen | 4.8553 | 1179.830933 | 6.40428 | 67.2800 |
| 117 | Zimbabwe | 4.0248 | 826.676632 | 5.23100 | 50.6523 |

118 rows × 5 columns

Se agrega una columna para el HDI.

```
hdi['HDI'] = np.cbrt(hdi.TFR * hdi.EDU * hdi.LEB)
```

```
hdi['HDI'] = hdi.HDI / hdi.HDI.max()
```

```
hdi
```

| | País | TFR | GDP | EDU | LEB | HDI |
|-----|---------------------|--------|--------------|----------|---------|----------|
| 0 | Albania | 1.6564 | 4052.927488 | 52.38337 | 77.9359 | 0.719815 |
| 1 | Algeria | 2.8434 | 4493.635433 | 35.33614 | 73.8081 | 0.742293 |
| 2 | Antigua and Barbuda | 1.7854 | 13038.258604 | 22.67580 | 76.8195 | 0.555621 |
| 3 | Argentina | 2.3462 | 10023.206688 | 89.15845 | 75.7208 | 0.955952 |
| 4 | Armenia | 1.5010 | 3462.399904 | 63.13266 | 73.1597 | 0.725812 |
| ... | ... | ... | ... | ... | ... | ... |
| 113 | Uruguay | 2.0110 | 11567.565434 | 80.32340 | 76.8580 | 0.881393 |
| 114 | Uzbekistan | 2.4407 | 1789.707894 | 7.55595 | 69.2354 | 0.412952 |
| 115 | Viet Nam | 1.8949 | 1631.765100 | 22.71635 | 73.5126 | 0.558836 |
| 116 | Yemen | 4.8553 | 1179.830933 | 6.40428 | 67.2800 | 0.486832 |
| 117 | Zimbabwe | 4.0248 | 826.676632 | 5.23100 | 50.6523 | 0.388894 |

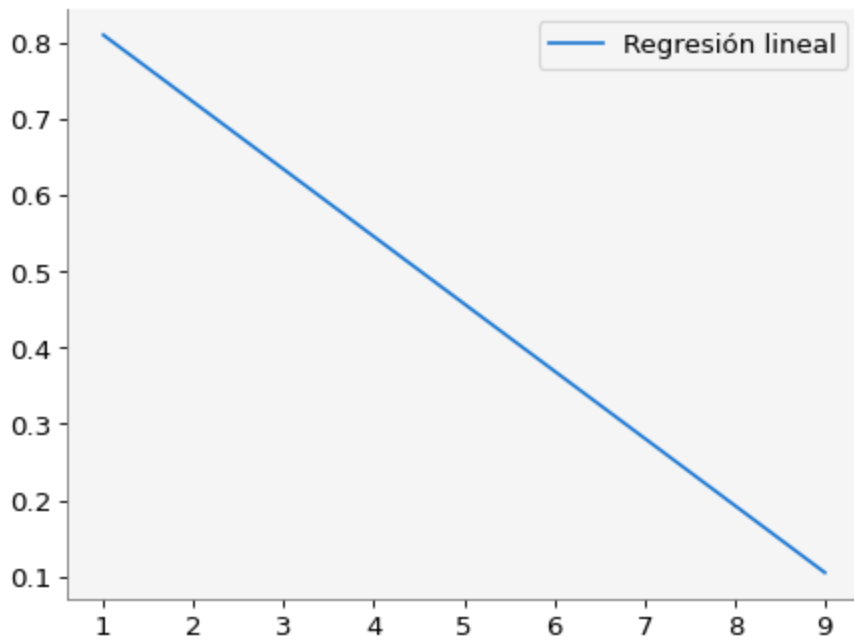
118 rows × 6 columns

Se realiza una regresión lineal del HDI vs TFR.

```
lres = scipy.stats.linregress(hdi.TFR, hdi.HDI)
print(lres.slope, lres.intercept, lres.rvalue**2)
x = np.linspace(1,9,100)
y = lres.slope * x + lres.intercept

plt.plot(x, y, label='Regresión lineal')
plt.legend()
plt.show()
```

-0.08804520907984292 0.8975361617317406 0.5010566535168526



Se grafica todo juntos

```
hdi.sort_values('EDU', inplace = True)

lista_paises = [('Niger', 'Nigeria', 'darkred'),
                ('Central African Republic', 'República Centroafricana', 'red'),
                ('Burundi', 'Burundi', 'orangered'),
                ('Malawi', 'Malawi', 'darkorange'),
                ('Yemen', 'Yemen', 'limegreen'),
                ('Mexico', 'México', 'forestgreen'),
                ('Cuba', 'Cuba', 'dodgerblue'),
                ('Liechtenstein', 'Liechtenstein', 'royalblue'),
                ('Japan', 'Japón', 'mediumblue')
                ]

paises_to_plot = [hdi[hdi['País'] == p[0]] for p in lista_paises]

plt.figure(figsize=(12,8))

ax = plt.subplot(111)
niv_edu = np.array(hdi['EDU']*10.0)
#gdp_val = np.array(np.log(hdi['GDP']))
gdp_val = np.array(hdi['GDP'])

scatter = ax.scatter(hdi['TFR'], hdi['HDI'], marker='o', s=niv_edu, c=gdp_val, ec

ha = 'right'
for pa, cl in zip(paises_to_plot, lista_paises):
    if cl[0] == 'Yemen':
        xt, yt = 6., 0.55
    elif cl[0] == 'Niger':
        xt, yt = 8, 0.4
```

```

elif cl[0] == 'India':
    xt, yt = 3.0, 0.2
elif cl[0] == 'Yemen':
    xt, yt = 4.5, 1.0
elif cl[0] == 'Mexico':
    xt, yt = 2.25, .25
elif cl[0] == 'Liechtenstein':
    xt, yt = 0.75, 0.35
    ha = 'left'
elif cl[0] == 'Burundi':
    xt, yt = 6.75, 0.45
elif cl[0] == 'Central African Republic':
    xt, yt = 7, 0.1
elif cl[0] == 'Malawi':
    xt, yt = 4, 0.1
elif cl[0] == 'Republic of Korea':
    xt, yt = 0.65, 1.05
elif cl[0] == 'Cuba':
    xt, yt = 3, 1.05
else:
    xt, yt = 0.75, 1.0

niv_edu = np.array(pa['EDU']*10.0)
gdp_val = np.array(np.log(pa['GDP']))
ax.scatter(pa['TFR'], pa['HDI'], marker='o', s = niv_edu, ec=cl[2], facecolor=cl[2])

plt.annotate(cl[1], xy = (pa['TFR'], pa['HDI']), xytext = (xt, yt), zorder=20,
             color = cl[2], fontsize=10, fontweight='normal', ha=ha,
             bbox=dict(boxstyle='round', fc='white', ec='gainsboro', alpha=0.5),
             arrowprops=dict(arrowstyle='->', color=cl[2]))

ax.plot(x,y,'k--', alpha=0.5, lw=3, label='Ajuste: slope = {:.3f}, intercept = {:.3f}'.format(slope, intercept))
ax.set_xlabel('TFR')
ax.set_ylabel('HDI')
ax.set_xticks([i for i in range(1,10)], labels=[i for i in range(1,10)], fontsize=10)
ax.set_yticks([0,0.2,0.4,0.6,0.8,1.0], labels=[0,0.2,0.4,0.6,0.8,1.0], fontsize=10)
ax.set_ylim(-0.1, 1.1)
ax.spines['bottom'].set_visible(False)
ax.grid(lw=0.5)
ax.set_title('Efecto del TFR, GDP y EDU en el HDI', loc='left', color='gray', fontstyle='italic')
ax.set_title('fuente: http://data.un.org', loc='right', color='gray', fontstyle='italic')
plt.suptitle('Índice de desarrollo humano (HDI) en 2010', y = 0.96, color='black')
texto = """
Cada círculo representa a un país. El nivel de
educación (EDU) en las mujeres se representa
como el tamaño de cada círculo.
"""

plt.text(5.5,0.9,texto, fontsize=10,bbox=dict(boxstyle='round', fc='white', ec='gainsboro', alpha=0.5))
plt.colorbar(mappable=scatter, label='GDP [USD $]', ticks=[1000, 20000, 40000, 60000])
plt.legend(loc='lower left', frameon=True)

```



```
plt.savefig('HDI.pdf', dpi = 300)
```

```
/opt/conda/lib/python3.11/site-packages/matplotlib/text.py:1461: FutureWarning: Calling
float on a single element Series is deprecated and will raise a TypeError in the future.
Use float(ser.iloc[0]) instead
```

```
x = float(self.convert_xunits(x))
```

```
/opt/conda/lib/python3.11/site-packages/matplotlib/text.py:1463: FutureWarning: Calling
float on a single element Series is deprecated and will raise a TypeError in the future.
Use float(ser.iloc[0]) instead
```

```
y = float(self.convert_yunits(y))
```

```
/opt/conda/lib/python3.11/site-packages/matplotlib/text.py:1461: FutureWarning: Calling
float on a single element Series is deprecated and will raise a TypeError in the future.
Use float(ser.iloc[0]) instead
```

```
x = float(self.convert_xunits(x))
```

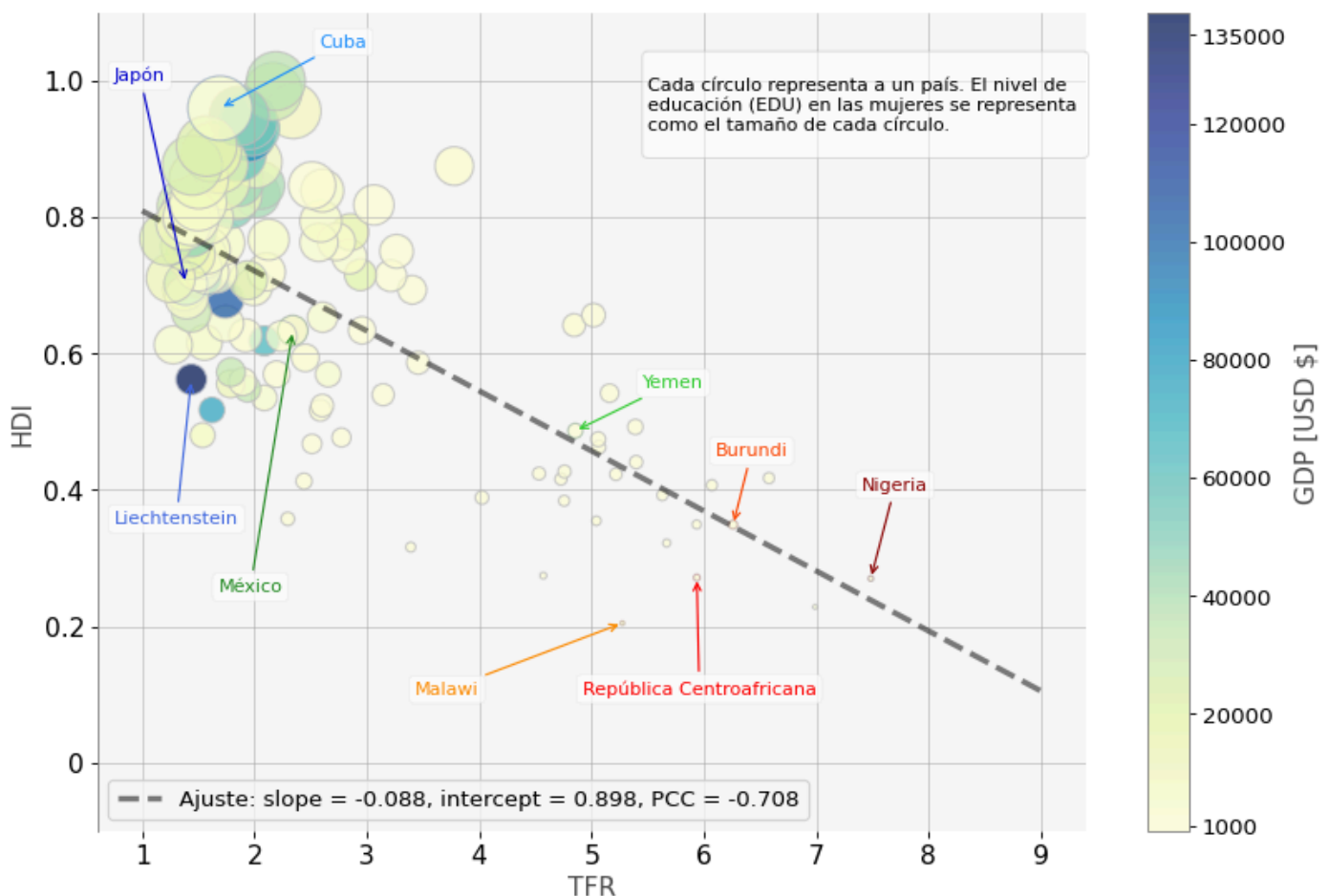
```
/opt/conda/lib/python3.11/site-packages/matplotlib/text.py:1463: FutureWarning: Calling
float on a single element Series is deprecated and will raise a TypeError in the future.
Use float(ser.iloc[0]) instead
```

```
y = float(self.convert_yunits(y))
```

Índice de desarrollo humano (HDI) en 2010

Efecto del TFR, GDP y EDU en el HDI

fuelle: <http://data.un.org>





3 Análisis de precios de un producto: graficas para presentación.

HeCompA - Precios-Producto by [Luis M. de la Cruz](#) is licensed under

[Attribution-ShareAlike 4.0 International](#)

Trabajo realizado con el apoyo del Programa UNAM-DGAPA-PAPIME PE101922

3.0.1 Contar la historia final

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

# Parámetros para el estilo de las gráficas
params = {'figure.figsize' : (10,5),
          'xtick.labelsize': 16,
          'ytick.labelsize': 16,
          'axes.labelsize' : 20,
          'axes.titlesize' : 20,
          'legend.fontsize': 14,
          'grid.color'      : 'darkgray',
          'grid.linewidth' : 0.5,
          'grid.linestyle' : '--',
          'font.family': 'DejaVu Serif',
          }

plt.rcParams.update(params)
```

```
precios = pd.read_excel("Libro1.xlsx", index_col=0)
precios
```

| | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 |
|--------|-------|-------|-------|-------|-------|------|------|
| Prod A | 395.0 | 420.0 | 430.0 | 390.0 | 300.0 | 275 | 260 |
| Prod B | 370.0 | 400.0 | 405.0 | 380.0 | 295.0 | 255 | 245 |
| Prod C | NaN | NaN | 100.0 | 180.0 | 200.0 | 240 | 182 |
| Prod D | NaN | NaN | NaN | 160.0 | 265.0 | 215 | 210 |
| Prod E | NaN | NaN | NaN | NaN | NaN | 100 | 205 |

```
A = np.array(precios.iloc[0])
B = np.array(precios.iloc[1])
C = np.array(precios.iloc[2])
D = np.array(precios.iloc[3])
```

```
E = np.array(precios.iloc[4])
print(A)
print(B)
print(C)
print(D)
print(E)
```

```
[395. 420. 430. 390. 300. 275. 260.]
[370. 400. 405. 380. 295. 255. 245.]
[ nan  nan 100. 180. 200. 240. 182.]
[ nan  nan  nan 160. 265. 215. 210.]
[ nan  nan  nan  nan  nan 100. 205.]
```

```
# Arreglo para usarse en el eje x
x2 = np.arange(2013,2020,1)
x2
```

```
array([2013, 2014, 2015, 2016, 2017, 2018, 2019])
```

```
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.scatter(x2[0], A[0], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Producto B
ax.scatter(x2[0], B[0], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Límites en el eje y
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes (ojo: ya no hacen falta los xticks)
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
               labels=['\ $0', '\ $100', '\ $200', '\ $300', '\ $400', '\ $500'])

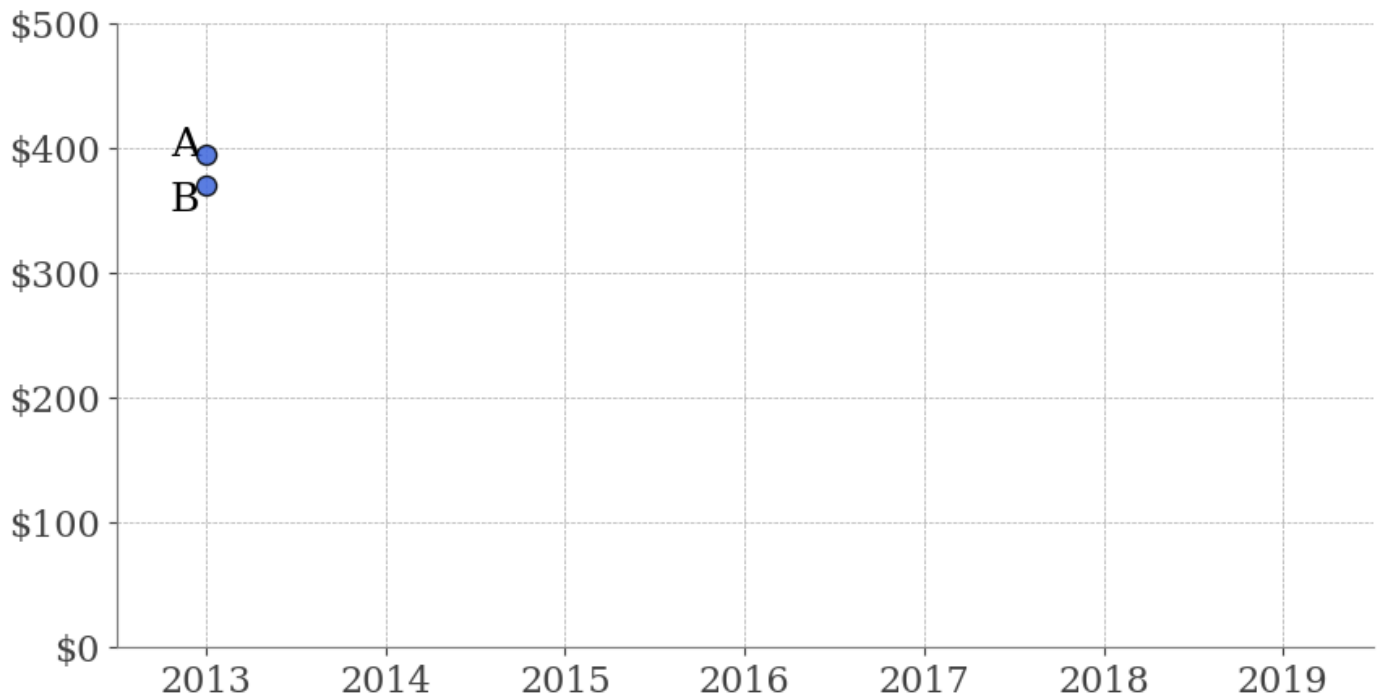
# Rejilla
ax.grid()

ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='#444444')
```

```
plt.suptitle('Precio promedio por año', fontsize=24, x =0.275, y=1.05)
plt.savefig('vis1.png',bbox_inches='tight', dpi=150)
plt.show()
```

Precio promedio por año



```
# Visualización 12: usamos líneas con color
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2, A, lw=3, c='darkgray')
ax.scatter(x2[0], A[0], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='darkgray')
ax.scatter(x2[-1], A[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='darkgray')

# Producto B
ax.plot(x2, B, lw=3, c='darkgray')
ax.scatter(x2[0], B[0], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='darkgray')
ax.scatter(x2[-1], B[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='darkgray')

# Límites en el eje y
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes (ojo: ya no hacen falta los xticks)
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
```

```
labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])
```

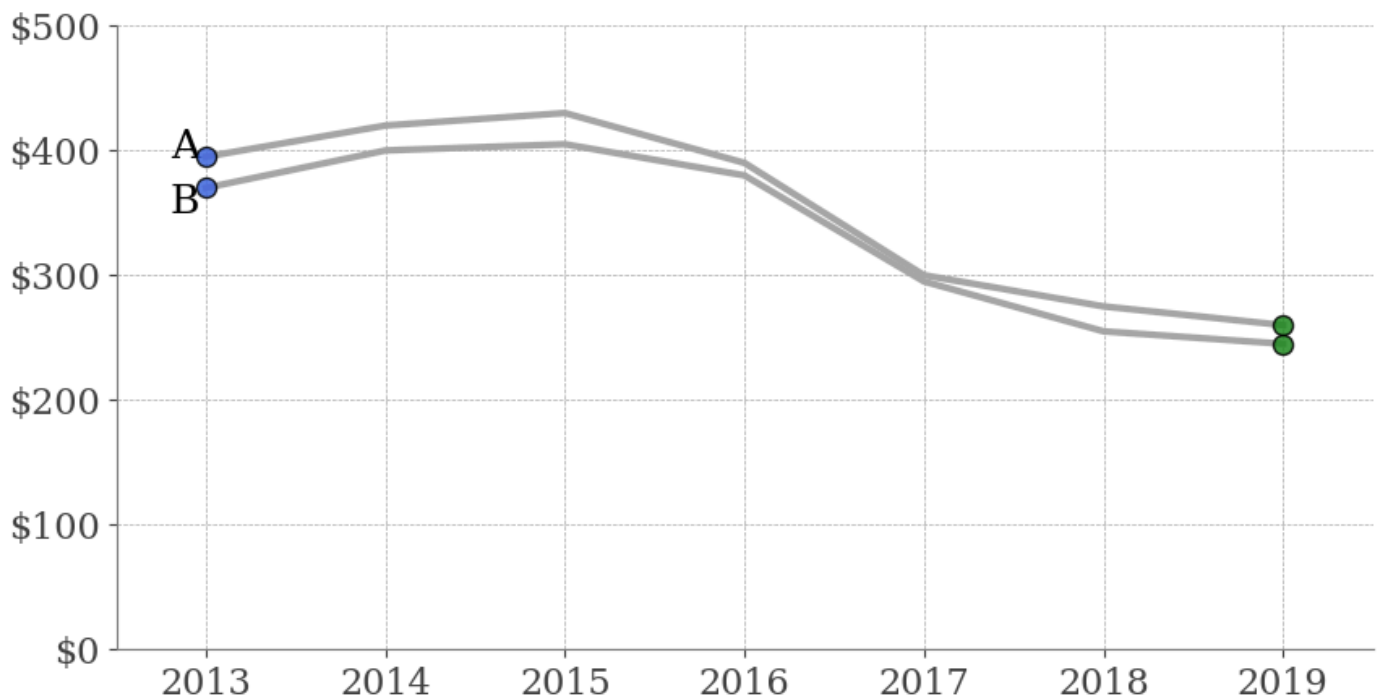
```
# Rejilla
ax.grid()

ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='#444444')

plt.suptitle('Precio promedio por año', fontsize=24, x =0.275, y=1.05)
plt.savefig('vis2.png',bbox_inches='tight', dpi=150)
plt.show()
```

Precio promedio por año



```
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2, A, lw=3, c='darkgray')
ax.scatter(x2[0], A[0], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='
```

```

# Producto B
ax.plot(x2, B, lw=3, c='darkgray')
ax.scatter(x2[0], B[0], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Producto C
ax.scatter(x2[2], C[2], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Producto D
ax.scatter(x2[3], D[3], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Producto E
ax.scatter(x2[5], E[5], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
               labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla
ax.grid()

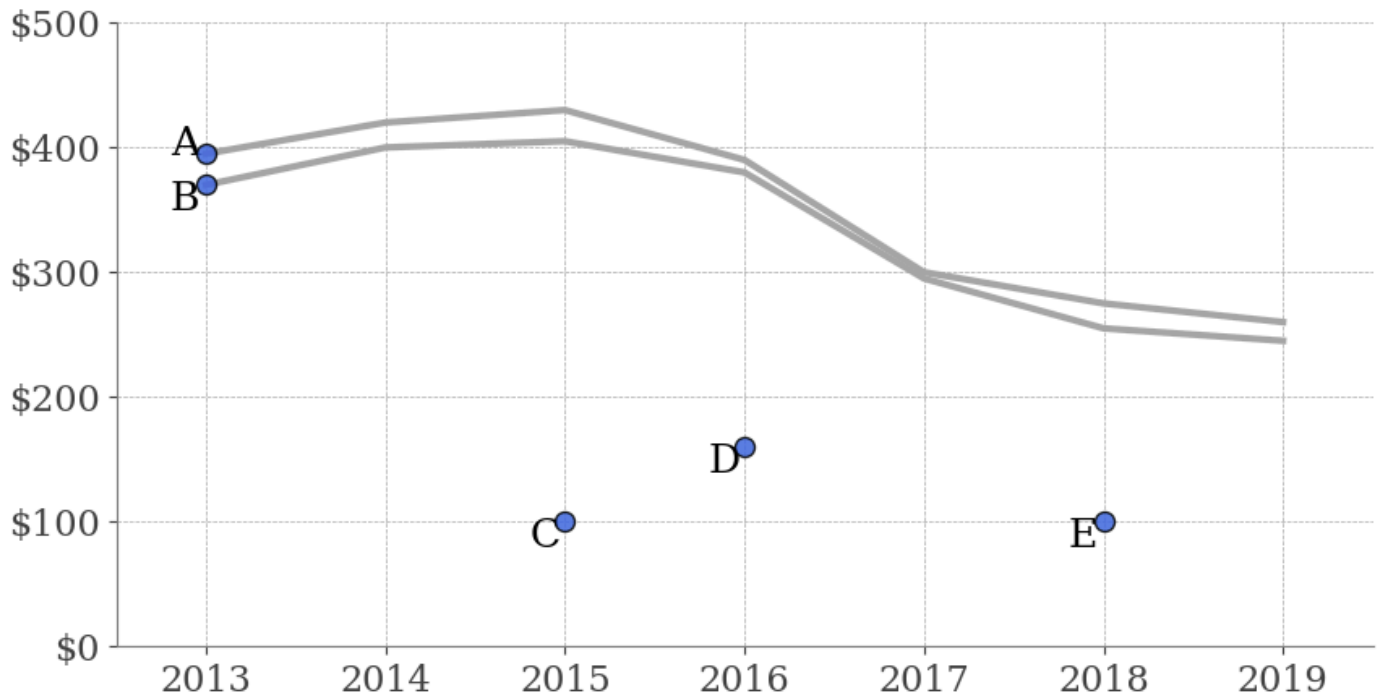
ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)
ax.text(x = x2[2]-0.20, y = C[2]-20, s = 'C', fontsize = 18)
ax.text(x = x2[3]-0.20, y = D[3]-20, s = 'D', fontsize = 18)
ax.text(x = x2[5]-0.20, y = E[5]-20, s = 'E', fontsize = 18)

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='#444444')

plt.suptitle('Precio promedio por año', fontsize=24, x =0.275, y=1.05)
plt.savefig('vis3.png',bbox_inches='tight', dpi=150)
plt.show()

```

Precio promedio por año



```
# Visualización 8: usamos líneas con color
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2[0:3], A[0:3], lw=3, color='firebrick')
ax.scatter(x2[0], A[0], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='')

# Producto B
ax.plot(x2[0:3], B[0:3], lw=3, color='firebrick')
ax.scatter(x2[0], B[0], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='')

# Producto C
ax.plot(x2[2:6], C[2:6], lw=3, color='firebrick')
ax.scatter(x2[2], C[2], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='')

# Producto D
ax.plot(x2[3:5], D[3:5], lw=3, color='firebrick')
ax.scatter(x2[3], D[3], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='')

# Producto E
ax.plot(x2, E, lw=3, c='firebrick')
ax.scatter(x2[5], E[5], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='')

# Límites en los ejes
ax.set_ylim(0,500)
```



```

ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
               labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla
ax.grid()

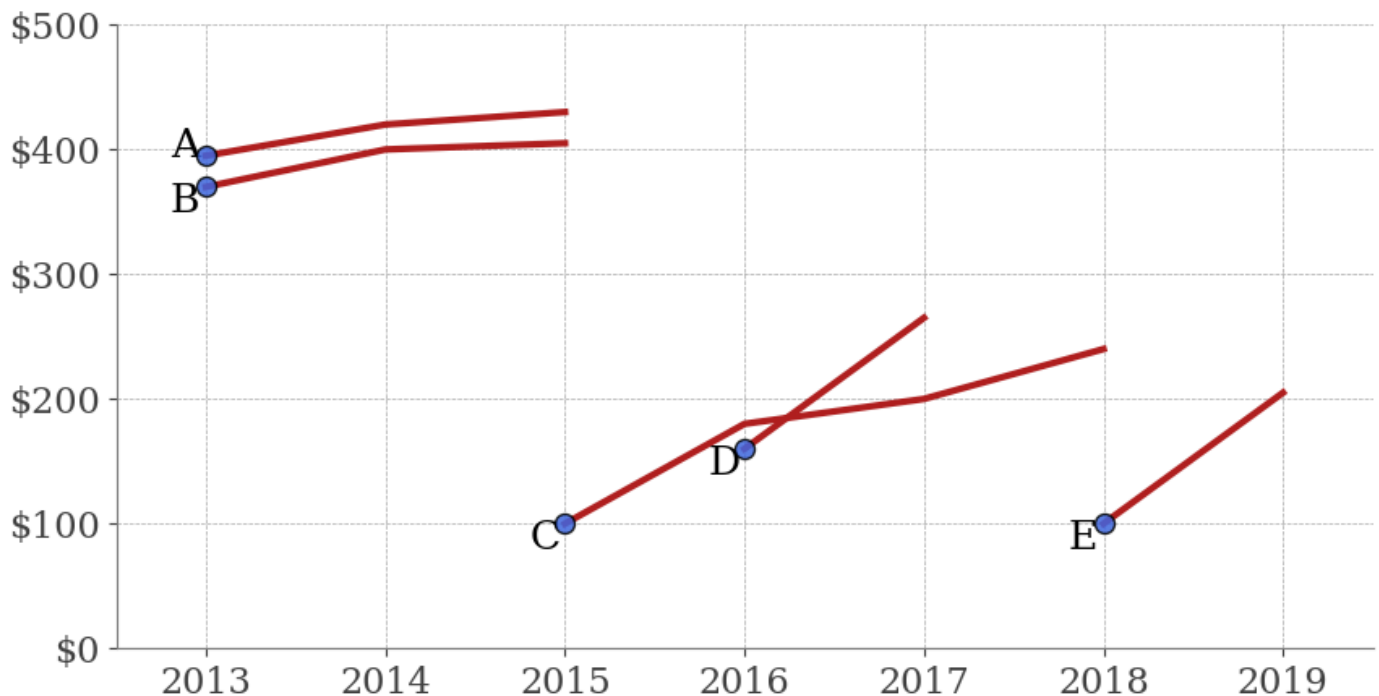
ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)
ax.text(x = x2[2]-0.20, y = C[2]-20, s = 'C', fontsize = 18)
ax.text(x = x2[3]-0.20, y = D[3]-20, s = 'D', fontsize = 18)
ax.text(x = x2[5]-0.20, y = E[5]-20, s = 'E', fontsize = 18)

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='#444444')

plt.suptitle('Precio promedio por año', fontsize=24, x =0.275, y=1.05)
plt.savefig('vis4.png',bbox_inches='tight', dpi=150)
plt.show()

```

Precio promedio por año



```

fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2[2:], A[2:], lw=3, color='royalblue')
ax.plot(x2[0:3], A[0:3], lw=3, color='firebrick')
ax.scatter(x2[0], A[0], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Producto B
ax.plot(x2[2:], B[2:], lw=3, color='royalblue')
ax.plot(x2[0:3], B[0:3], lw=3, color='firebrick')
ax.scatter(x2[0], B[0], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Producto C
ax.plot(x2[5:], C[5:], lw=3, color='royalblue')
ax.plot(x2[2:6], C[2:6], lw=3, color='firebrick')
ax.scatter(x2[2], C[2], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Producto D
ax.plot(x2[4:], D[4:], lw=3, color='royalblue')
ax.plot(x2[3:5], D[3:5], lw=3, color='firebrick')
ax.scatter(x2[3], D[3], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Producto E
ax.plot(x2, E, lw=3, c='firebrick')
ax.scatter(x2[5], E[5], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
               labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla
ax.grid()

ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)
ax.text(x = x2[2]-0.20, y = C[2]-20, s = 'C', fontsize = 18)
ax.text(x = x2[3]-0.20, y = D[3]-20, s = 'D', fontsize = 18)
ax.text(x = x2[5]-0.20, y = E[5]-20, s = 'E', fontsize = 18)

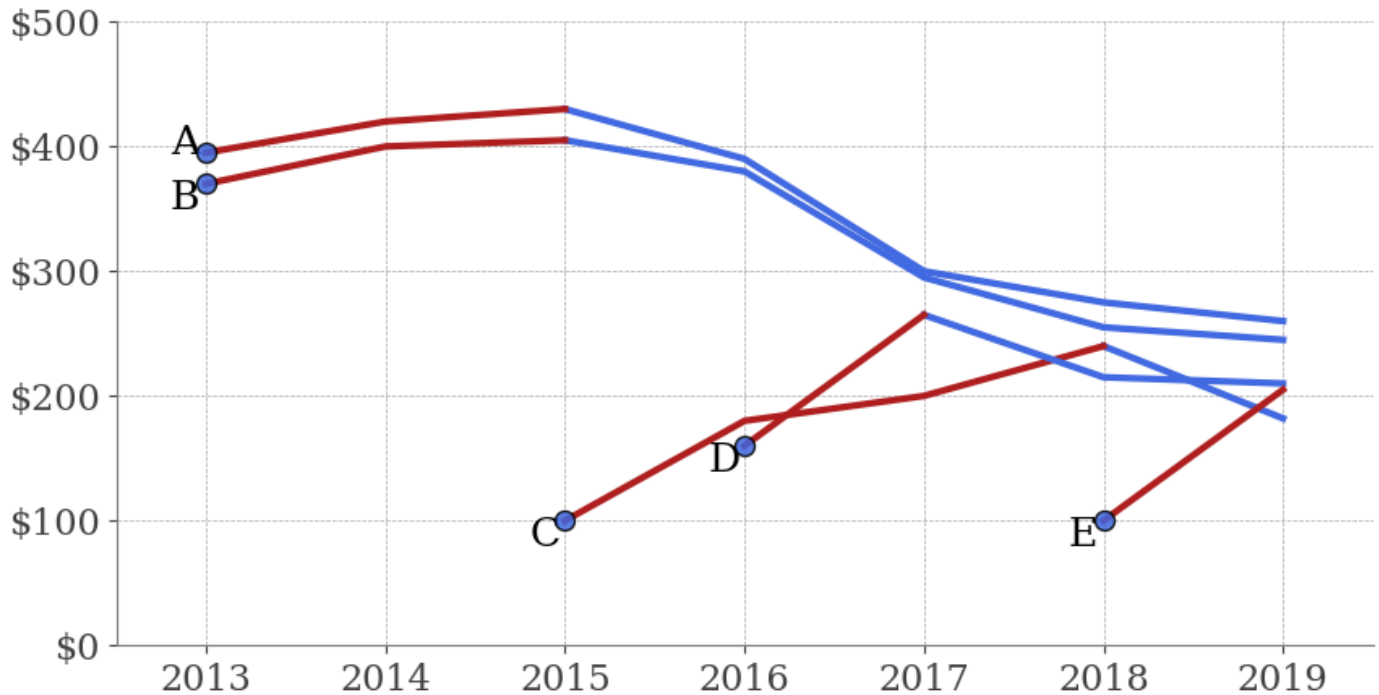
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')
ax.tick_params(axis='x', colors='#444444')

```

```
ax.tick_params(axis='y', colors='#444444')

plt.suptitle('Precio promedio por año', fontsize=24, x =0.275, y=1.05)
plt.savefig('vis5.png',bbox_inches='tight', dpi=150)
plt.show()
```

Precio promedio por año



```
# Visualización 11: marcamos el rango de precios de introducción
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2, A, lw=3, c='darkgray')
ax.scatter(x2[-1], A[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='darkgray')

# Producto B
ax.plot(x2, B, lw=3, c='darkgray')
ax.scatter(x2[-1], B[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='darkgray')

# Producto C
ax.plot(x2, C, lw=3, c='darkgray')
ax.scatter(x2[-1], C[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='darkgray')

# Producto D
ax.plot(x2, D, lw=3, c='darkgray')
ax.scatter(x2[-1], D[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='darkgray')
```

```

# Producto E
ax.plot(x2, E, lw=3, c='darkgray')
ax.scatter(x2[-1], E[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color

# Promedio final
precios_finales = np.array([A[-1], B[-1], C[-1], D[-1], E[-1],])
promedio_final = np.mean(precios_finales)
ax.scatter(x2[-1], promedio_final, marker='<', alpha=0.85, ec = 'k', s=75,
           zorder=5, color='orange', label='Promedio')

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
              labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla
ax.grid()

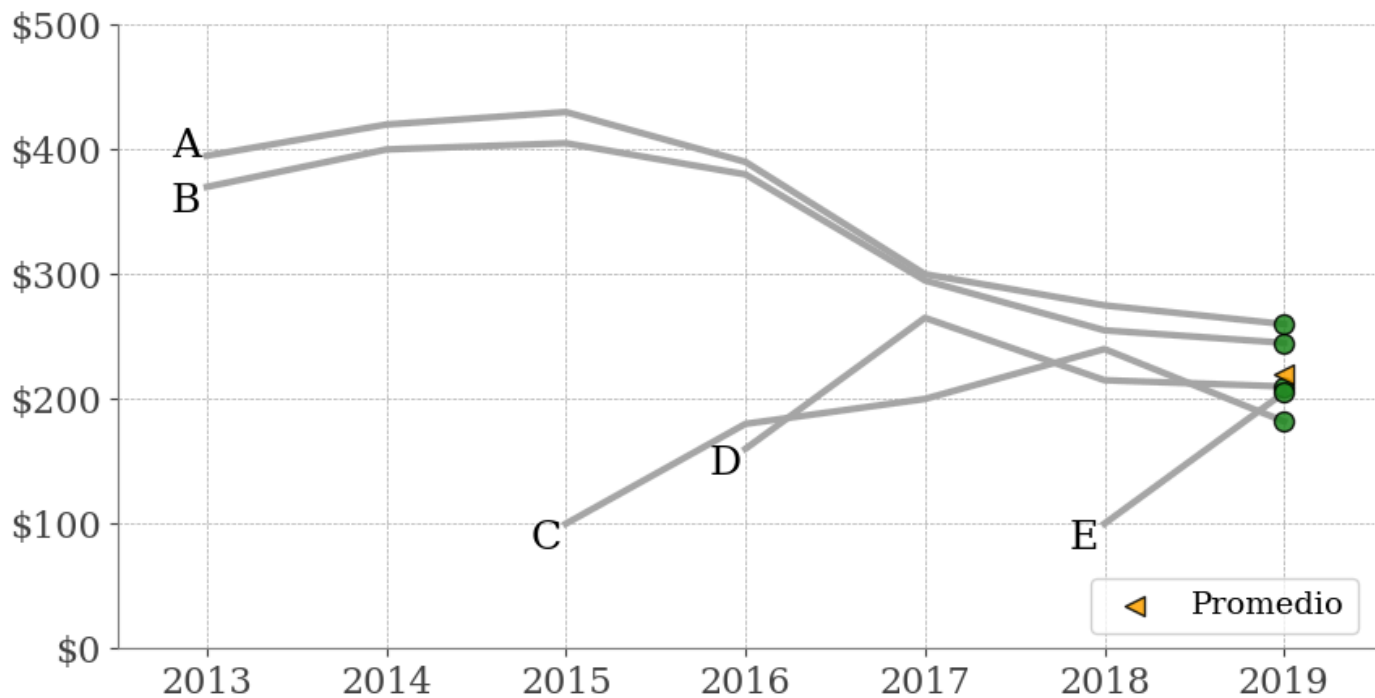
ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)
ax.text(x = x2[2]-0.20, y = C[2]-20, s = 'C', fontsize = 18)
ax.text(x = x2[3]-0.20, y = D[3]-20, s = 'D', fontsize = 18)
ax.text(x = x2[5]-0.20, y = E[5]-20, s = 'E', fontsize = 18)

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='#444444')

plt.suptitle('Precio promedio por año', fontsize=24, x =0.275, y=1.05)
plt.legend(loc='lower right')
plt.savefig('vis6.png',bbox_inches='tight', dpi=150)
plt.show()

```

Precio promedio por año



```
# Visualización 11: marcamos el rango de precios de introducción
fig = plt.figure() # Se define una figura
ax = fig.gca()     # Se obtienen los ejes de la figura

# Producto A
ax.plot(x2, A, lw=3, c='darkgray')
ax.scatter(x2[-1], A[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='green')

# Producto B
ax.plot(x2, B, lw=3, c='darkgray')
ax.scatter(x2[-1], B[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='green')

# Producto C
ax.plot(x2, C, lw=3, c='darkgray')
ax.scatter(x2[-1], C[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='green')

# Producto D
ax.plot(x2, D, lw=3, c='darkgray')
ax.scatter(x2[-1], D[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='green')

# Producto E
ax.plot(x2, E, lw=3, c='darkgray')
ax.scatter(x2[-1], E[-1], marker='o', alpha=0.85, ec = 'k', s=75, zorder=5, color='green')

# Promedio final
precios_finales = np.array([A[-1], B[-1], C[-1], D[-1], E[-1],])
```

```

promedio_final = np.mean(precios_finales)
ax.scatter(x2[-1], promedio_final, marker='<', alpha=0.85, ec = 'k', s=75,
           zorder=5, color='orange', label='Promedio')

# Límites en los ejes
ax.set_ylim(0,500)
ax.set_xlim(2012.5,2019.5)

# Marcas sobre los ejes
ax.set_xticks(ticks=[i for i in range(2013,2020)])
ax.set_yticks(ticks=[0,100,200,300,400,500],
              labels=['\$0', '\$100', '\$200', '\$300', '\$400', '\$500'])

# Rejilla
ax.grid()

# Etiquetado de cada línea
ax.text(x = x2[0]-0.20, y = A[0], s = 'A', fontsize = 18)
ax.text(x = x2[0]-0.20, y = B[0]-20, s = 'B', fontsize = 18)
ax.text(x = x2[2]-0.20, y = C[2]-20, s = 'C', fontsize = 18)
ax.text(x = x2[3]-0.20, y = D[3]-20, s = 'D', fontsize = 18)
ax.text(x = x2[5]-0.20, y = E[5]-20, s = 'E', fontsize = 18)

# Eliminación de algunas líneas del recuadro
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_color('gray')
ax.spines['bottom'].set_color('gray')

# Color de los ticks
ax.tick_params(axis='x', colors='#444444')
ax.tick_params(axis='y', colors='#444444')

# Recuadro para indicar la región del precio final
left, bottom, width, height = (2012, 150, 8, 50)
rect = plt.Rectangle((left, bottom), width, height,
                    facecolor="black", alpha=0.1)
ax.add_patch(rect)
ax.text(x = x2[0]-0.25, y = 165, s = 'Rango de precios de introducción', fontsize=14)

plt.suptitle('Precio promedio por año', fontsize=24, x =0.275, y=1.05)
plt.legend(loc='lower right')
plt.savefig('vis_final.png',bbox_inches='tight', dpi=150)
plt.show()

```

Precio promedio por año

