

Módulo 7  
Clase 1

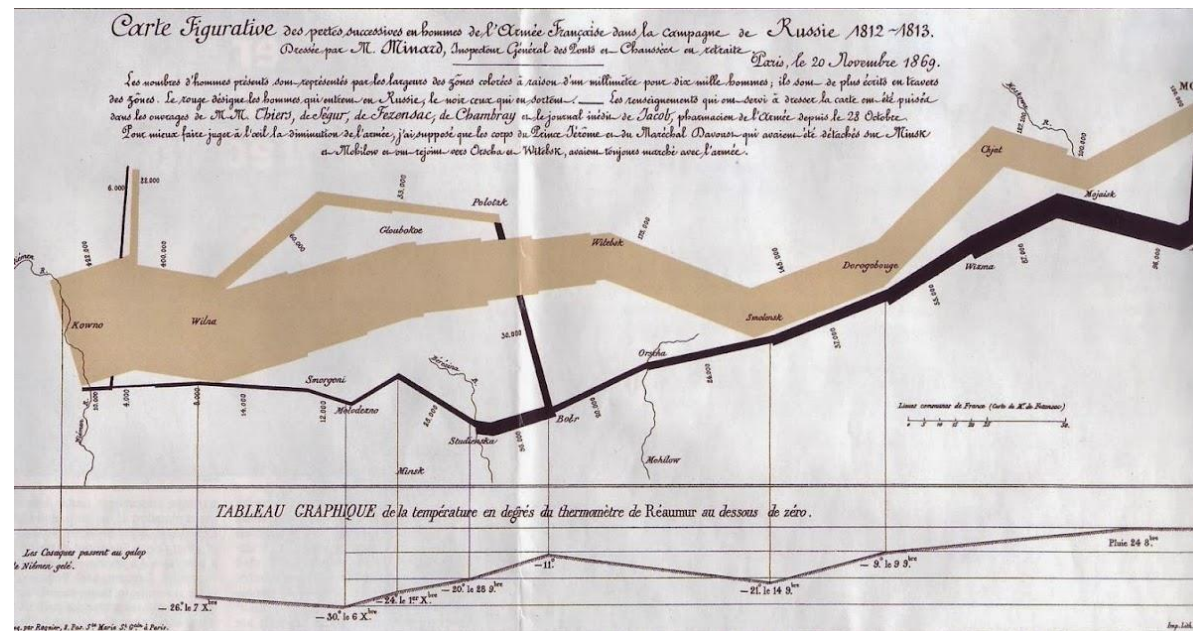
# Análisis Exploratorio de Datos

# Visualización de Datos



# Visualización de Datos.

Entre los que dibujan datos en periódicos, nadie discute la condición fundadora de la infografía del magnífico trabajo de Charles Minard sobre la campaña de Napoleón a Rusia, entre los años 1812 y 1813. En esta infografía, publicado en un diario de la época, se ilustra cómo 442 mil hombres ingresan a Rusia (color ocre) y sólo 10.000 regresan. Esta fue la peor campaña de Napoleón.



Nótese cómo en esta infografía se mezclan elementos visuales para narrar una historia centrada en base a los datos.

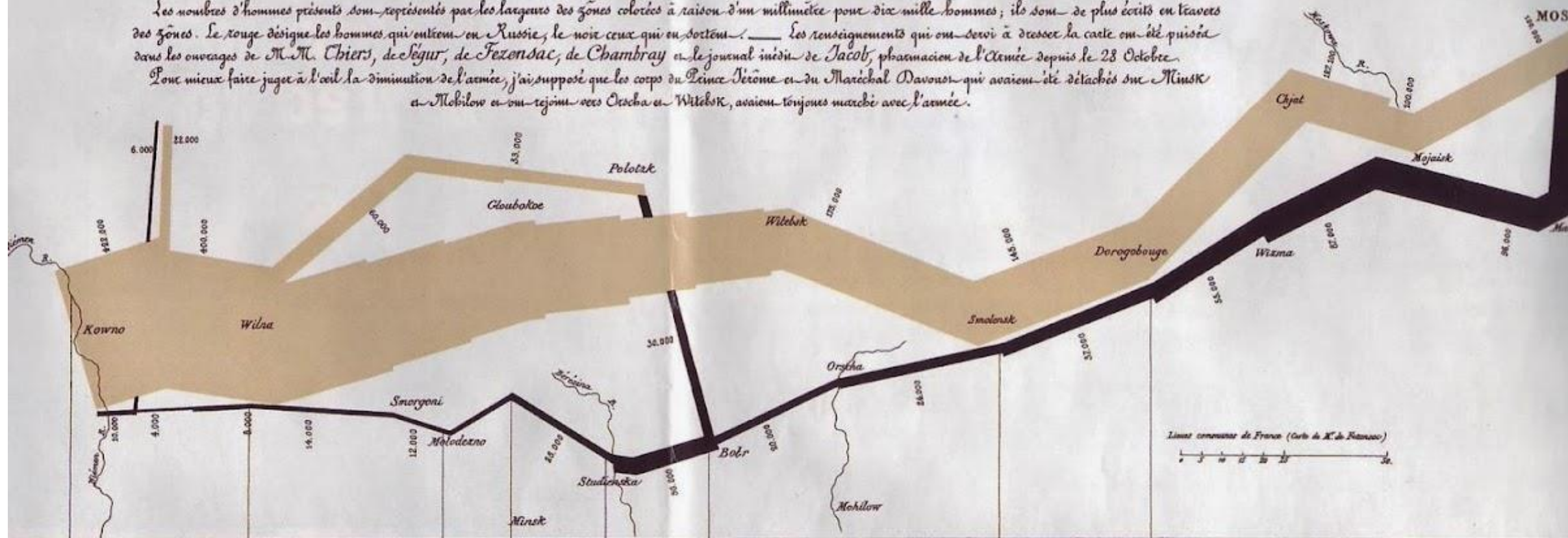
# Carte Figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812-1813.

Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite

Paris, le 20 Novembre 1869.

Les nombres d'hommes présents sont représentés par les largeurs des zones colorées à raison d'un millimètre pour dix mille hommes; ils sont de plus écrits en travers des zones. Le rouge désigne les hommes qui entrent en Russie, le noir ceux qui en sortent. — Les renseignements qui ont servi à dresser la carte ont été puisés dans les ouvrages de M.M. Thiers, de Ségur, de Fozzard, de Chambray et le journal inédit de Jacob, pharmacien de l'Armée depuis le 28 Octobre.

Pour mieux faire juger à l'œil la diminution de l'armée, j'ai supposé que les corps du Prince Jérôme et du Maréchal Davout qui avaient été détachés sur Minsk et Mohilow et qui rejoignent vers Orscha et Wiltsch, avaient toujours marché avec l'armée.

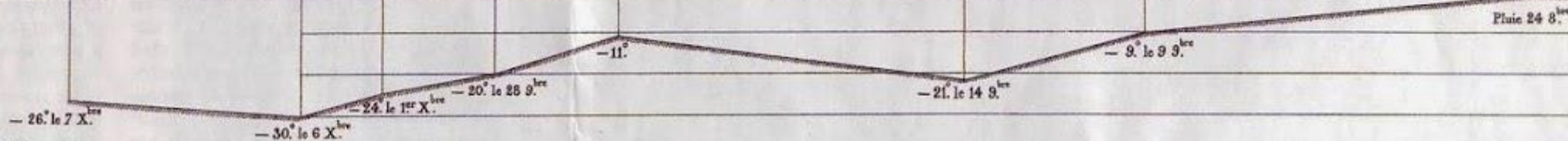


Lignes connues de France (Carte de M. de Bismarck)



## TABLEAU GRAPHIQUE de la température en degrés du thermomètre de Réaumur au dessous de zéro.

Les Cosaques passent au galop le Niémen gelé.



eg. par Rognier, 8. Par. 5<sup>me</sup> Marie 5<sup>me</sup> 0<sup>me</sup> à Paris.

Imp. Lab. Eng.



# Visualización de Datos

- La visualización de datos es el proceso de representar datos e información de una manera visual, gráfica o diagramática, con el objetivo de facilitar la comprensión, el análisis y la comunicación de patrones, tendencias, relaciones y otros insights relevantes que puedan extraerse de ellos.
- La visualización de datos puede ser utilizada en una variedad de contextos, desde la investigación científica hasta los negocios, la educación y el periodismo, y puede ser realizada mediante diferentes herramientas y técnicas, como gráficos, mapas, diagramas de flujo, infografías, entre otros.
- La visualización de datos se considera una herramienta poderosa en la toma de decisiones, ya que puede ayudar a identificar patrones y tendencias que de otra manera serían difíciles de detectar, y a comunicar de manera más efectiva información compleja a una audiencia no especializada.

# Librerías de Visualización

Python cuenta con varias librerías para visualización. Las más utilizadas son las siguientes:

- **Matplotlib:** para graficas sencillas con bars, pies, lines, scatter plots, etc.
- **Seaborn:** para visualización estadística, para crear mapas de calor o de alguna manera resumiendo los datos y aún desea mostrar la distribución de los datos.
- **Plotly y Bokeh:** para visualización interactiva. Si los datos son tan complejos (o no puede ver la información de sus datos), se recomienda utilizar Plotly y Bokeh para crear visualizaciones interactivas que permitan a los usuarios explorar los datos mismos.



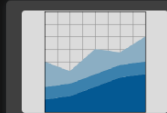
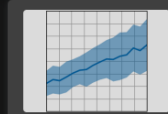
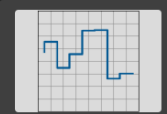
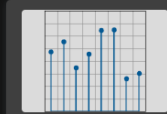
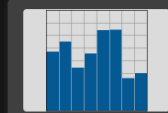
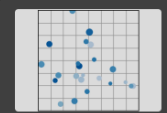
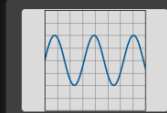
Librería Matplotlib

# Librería Matplotlib

- Es una librería para análisis visual
- Muy popular en proyectos de ciencia de datos y se complementa muy bien con Pandas y Numpy
- Posee buen performance en su ejecución
- Viene en la distribución Anaconda

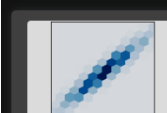
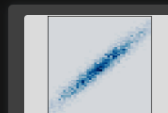
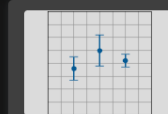
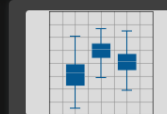
## Basic

Basic plot types, usually y versus x.



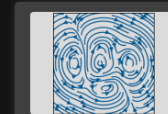
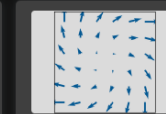
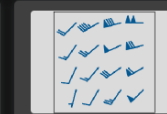
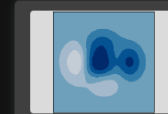
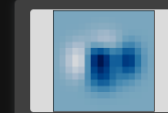
## Statistics plots

Plots for statistical analysis.



## Plots of arrays and fields

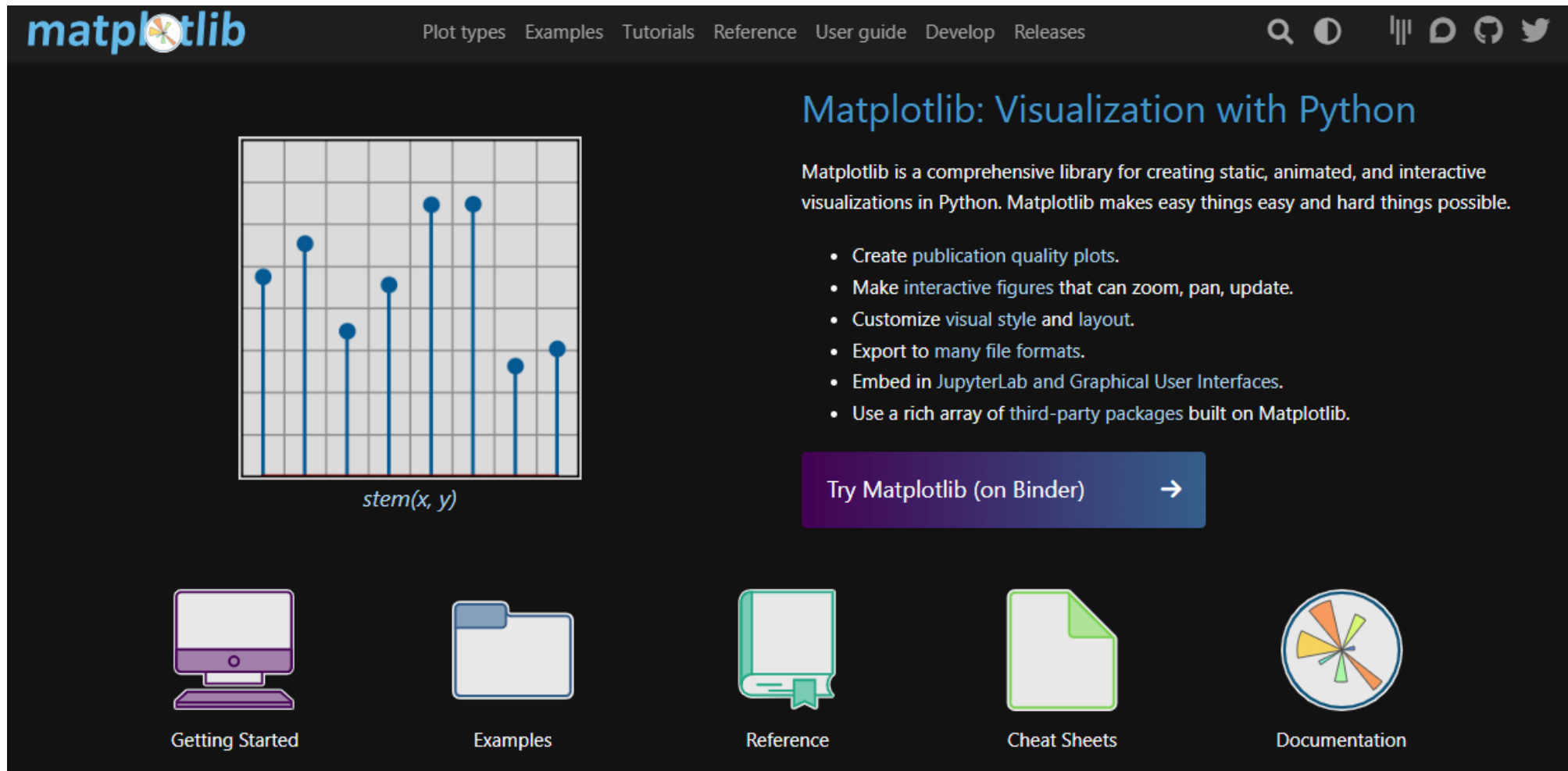
Plotting for arrays of data  $z(x, y)$  and fields  $u(x, y)$ ,  $v(x, y)$ .





# Web Matplotlib

<https://matplotlib.org>



The screenshot shows the Matplotlib website homepage. At the top is the Matplotlib logo and a navigation bar with links: Plot types, Examples, Tutorials, Reference, User guide, Develop, and Releases. On the right of the navigation bar are icons for search, a moon, a list, a document, and social media links for GitHub and Twitter. The main content area features a large stem plot on the left with the label  $stem(x, y)$  below it. To the right of the plot is the heading "Matplotlib: Visualization with Python" followed by a paragraph describing the library as a comprehensive tool for creating static, animated, and interactive visualizations in Python. Below this is a bulleted list of features. A purple button labeled "Try Matplotlib (on Binder)" with a right arrow is positioned below the list. At the bottom of the page is a row of five icons with corresponding labels: a computer icon for "Getting Started", a folder icon for "Examples", a book icon for "Reference", a document icon for "Cheat Sheets", and a circular icon with colored segments for "Documentation".

**matplotlib**


Plot types Examples Tutorials Reference User guide Develop Releases


## Matplotlib: Visualization with Python


Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.


- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.


[Try Matplotlib \(on Binder\)](#) →

 Getting Started

 Examples

 Reference

 Cheat Sheets

 Documentation

# Librería Matplotlib

- La librería Matplotlib, puede ser utilizada para programar los gráficos y visualizaciones que necesitemos. Esta librería tiene dos mecanismos para ser programada:
- Mediante la utilización de **funciones del módulo pyplot**  
Mediante el modelo de **orientación a objetos**
- El módulo pyplot, ofrece mayor simpleza en la programación, puesto que utiliza funciones prefabricadas para facilitar la programación de un gráfico. Sin embargo, si es requerido un mayor nivel de personalización, ahí se hace necesario conocer el modelo de orientación a objetos que provee la librería.

En esta presentación conoceremos ambos enfoques.

# Set de Datos

- Para este ejemplo, utilizaremos el set de datos de contagios totales de covid-19 por grupo etario, disponible en el repositorio del Ministerio de Ciencias.



	A	B	C	D
1	Grupo de edad	Sexo	Fecha	Contagiados
2	00 - 04 años	M	2020-03-25	4
3	05 - 09 años	M	2020-03-25	2
4	10 - 14 años	M	2020-03-25	7
5	15 - 19 años	M	2020-03-25	8
6	20 - 24 años	M	2020-03-25	25
7	25 - 29 años	M	2020-03-25	61
8	30 - 34 años	M	2020-03-25	88
9	35 - 39 años	M	2020-03-25	72
10	40 - 44 años	M	2020-03-25	62
11	45 - 49 años	M	2020-03-25	47
12	50 - 54 años	M	2020-03-25	28
13	55 - 59 años	M	2020-03-25	30
14	60 - 64 años	M	2020-03-25	18
15	65 - 69 años	M	2020-03-25	14
16	70 - 74 años	M	2020-03-25	16
17	75 - 79 años	M	2020-03-25	8
18	80 y más años	M	2020-03-25	6
19	00 - 04 años	F	2020-03-25	6
20	05 - 09 años	F	2020-03-25	4
21	10 - 14 años	F	2020-03-25	2
22	15 - 19 años	F	2020-03-25	12
23	20 - 24 años	F	2020-03-25	43
24	25 - 29 años	F	2020-03-25	65
25	30 - 34 años	F	2020-03-25	80
26	35 - 39 años	F	2020-03-25	79



# Importando la librería

```
In [2]: import matplotlib.pyplot as plt
```

```
In [3]: %matplotlib inline
```

Variable de ambiente para desplegar en línea los gráficos en el entorno Jupyter

# Obtención de los datos

➤ Para trabajar en este ejemplo, utilizaremos las cifras del COVID-19 en Chile por rango etario.

```
df = pd.read_csv('datos-covit-etareo.csv')  
df
```

	Grupo de edad	Sexo	Fecha	Contagiados
0	00 - 04 años	M	2020-03-25	4
1	05 - 09 años	M	2020-03-25	2
2	10 - 14 años	M	2020-03-25	7
3	15 - 19 años	M	2020-03-25	8
4	20 - 24 años	M	2020-03-25	25
...	...	...	...	...
1593	60 - 64 años	F	2020-06-01	2407
1594	65 - 69 años	F	2020-06-01	1547
1595	70 - 74 años	F	2020-06-01	1178
1596	75 - 79 años	F	2020-06-01	880
1597	80 y más años	F	2020-06-01	1525

1598 rows x 4 columns

# Obtención de los datos

➤ Ahora realizamos algunas adecuaciones a nuestro DataFrame para ejemplificar de mejor manera.

```
df2 = df.groupby('Fecha').sum()  
df2.reset_index(inplace=True)  
df2.drop('Fecha', axis=1, inplace=True)  
df2.head(10)
```

Contagiados	
0	1012
1	1252
2	1434
3	1723
4	1906
5	2088
6	2373
7	2744
8	2938
9	3398



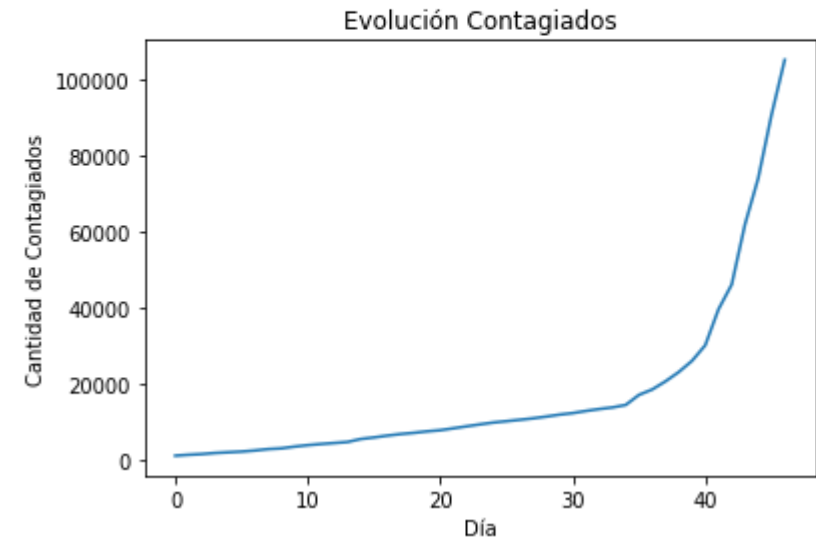
# Nuestro primer gráfico

➤ Nuestro primer gráfico lo construiremos utilizando las funciones pyplot de matplotlib.



La instrucción **plt.show()** puede ser omitida en los notebooks jupyter, en otro entorno sí debe incluirse

```
plt.plot(df2)
plt.xlabel('Día')
plt.ylabel('Cantidad de Contagiados')
plt.title('Evolución Contagiados')
plt.show()
```



# Creando sub gráficos



Nota: df\_m y df\_f son datasets que previamente han sido filtrados y agrupados convenientemente.

Cant. filas

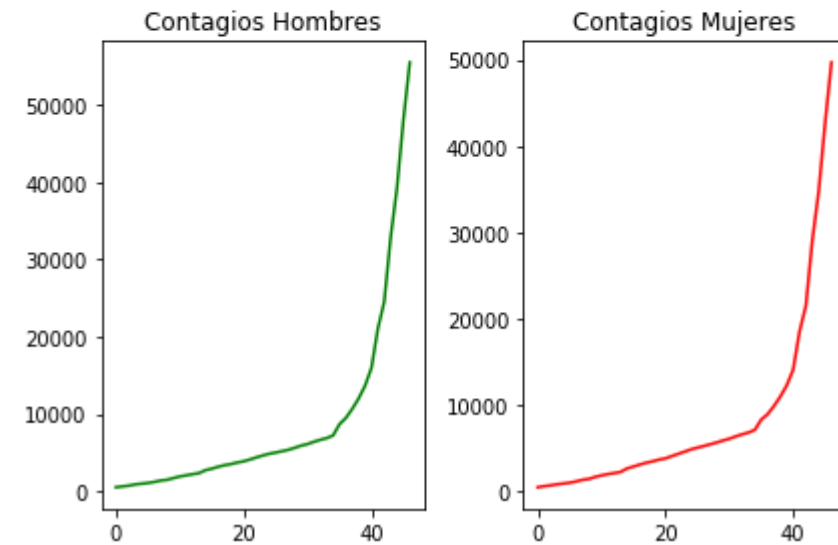
Cant. cols

Índice de la posición del gráfico en la grilla

```
plt.subplot(1,2,1)
plt.plot(df_m, 'g')
plt.title('Contagios Hombres')

plt.subplot(1,2,2)
plt.plot(df_f, 'r')
plt.title('Contagios Mujeres')

plt.tight_layout()
```



# Tamaño de la Figura y DPI

Matplotlib permite especificar la relación de aspecto, el DPI y el tamaño de la figura cuando se crea el objeto Figure. Puede usar los argumentos de las palabras clave `figsize` y `dpi`. No es necesario poner las dos.



**figsize** es una tupla del ancho y alto de la figura en pulgadas



**dpi** es el punto por pulgada (pixel por pulgada).

```
# se cambia el tamaño de la figura y el numero de puntos por pulgada  
plt.figure(figsize=(8,4), dpi=100)
```

```
plt.plot(x, y) # se grafica una linea de color azul
```

```
plt.xlabel('Nombre del eje X') # definir el nombre del eje X
```

```
plt.ylabel('Nombre del eje Y') # definir el nombre del eje Y
```

```
plt.title('Titulo de la grafica'); # definir el titulo de la grafica
```

```
# agrego ; al final del ultimo comando para solo mostrar la grafica  
# plt.show() no es necesario en jupyter notebook
```





# Escala Logarítmica



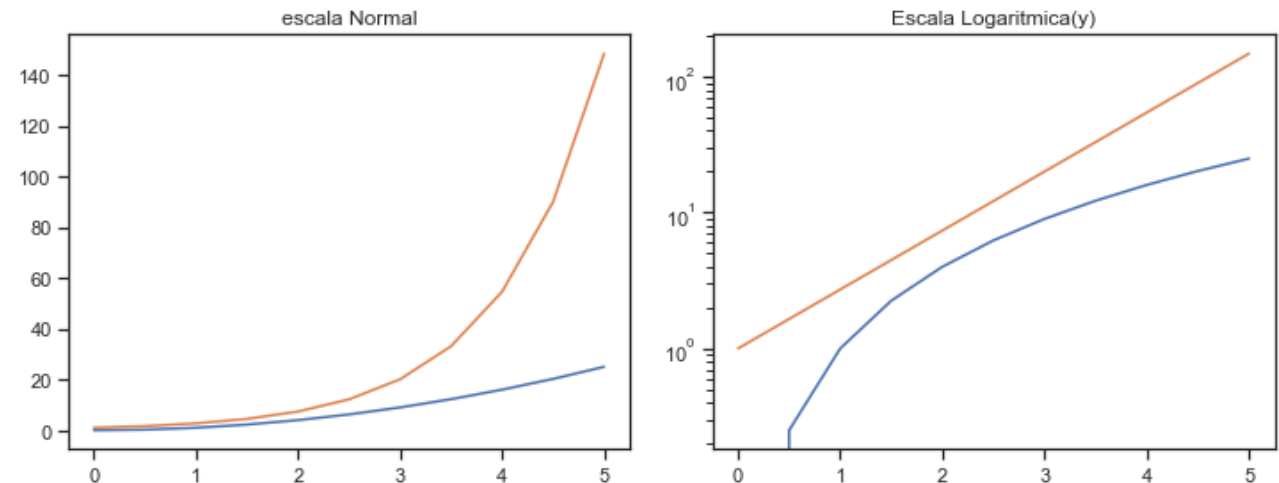
Mediante la función **yscale()** se puede definir que el eje y tenga una escala logarítmica.

```
plt.figure(figsize=(10,4))
```

```
plt.subplot(1,2,1)  
plt.plot(x, x**2, x, np.exp(x))  
plt.title("escala Normal")
```

```
plt.subplot(1,2,2)  
plt.plot(x, x**2, x, np.exp(x))  
plt.yscale("log")  
plt.title("Escala Logaritmica(y)");
```

```
plt.tight_layout() # para que no se superpongan las graficas
```



# Anotaciones de Texto

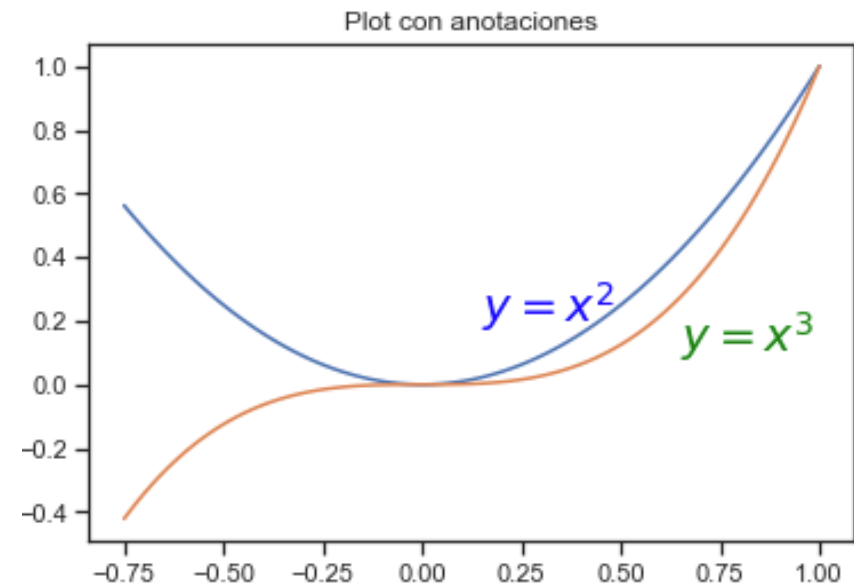


Anotar texto en figuras matplotlib se puede hacer usando la función `text`. Es compatible con el formato LaTeX al igual que los textos y títulos de la etiqueta del eje:

```
# Datos para graficar
xx = np.linspace(-0.75, 1., 100)

plt.plot(xx, xx**2, xx, xx**3)
plt.title("Plot con anotaciones")

# Anotacion 1
plt.text(0.15, 0.2, r"$y=x^2$", fontsize=20, color="blue")
#Anotacion 2
plt.text(0.65, 0.1, r"$y=x^3$", fontsize=20, color="green");
```



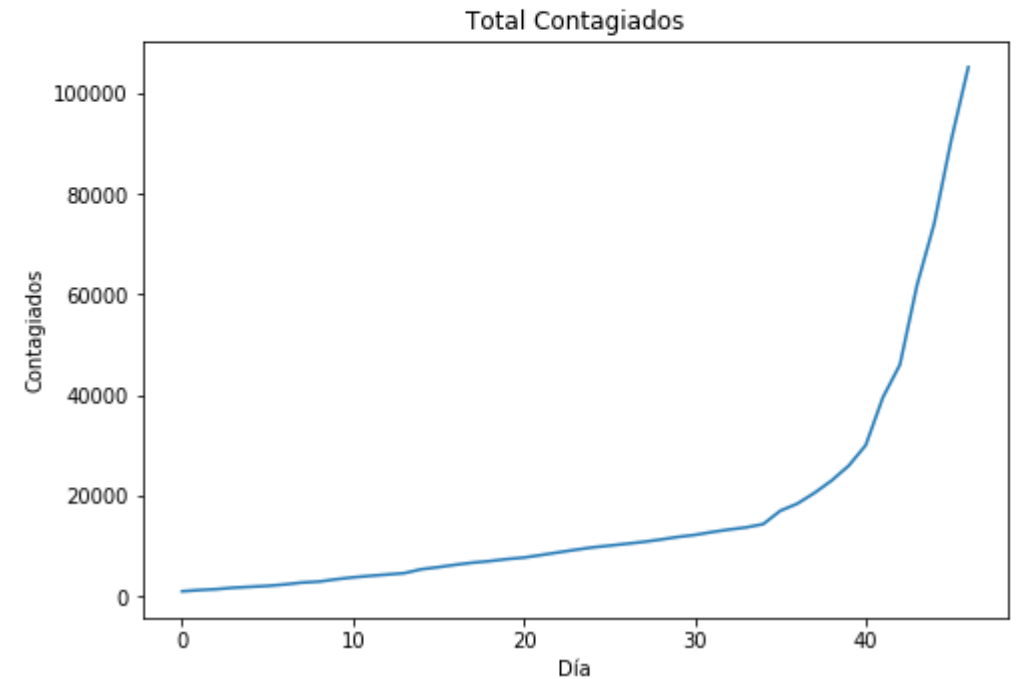
# Utilizando Orientación a Objetos



Ahora crearemos un gráfico utilizando orientación a objetos

```
fig = plt.figure()
axes = fig.add_axes([0,0,1,1])
axes.plot(df2)
axes.set_xlabel('Día')
axes.set_ylabel('Contagiados')
axes.set_title('Total Contagiados')
```

```
Text(0.5, 1.0, 'Total Contagiados')
```



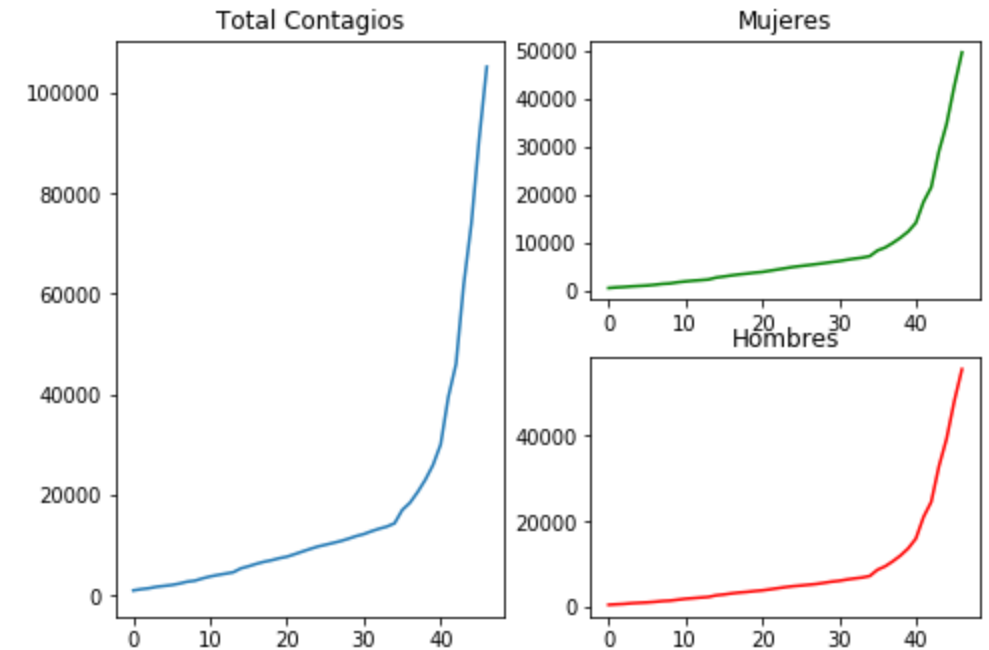


# Ubicando la posición de los ejes

➤ Nótese lo que sucede cuando modificamos los valores de los ejes.

```
fig = plt.figure()
axes1 = fig.add_axes([0, 0, 0.45, 1])
axes2 = fig.add_axes([0.55, 0, 0.45, 0.45])
axes3 = fig.add_axes([0.55, 0.55, 0.45, 0.45])
axes1.plot(df2)
axes2.plot(df_m, 'r')
axes3.plot(df_f, 'g')
axes1.set_title('Total Contagios')
axes2.set_title('Hombres')
axes3.set_title('Mujeres')
```

Text(0.5, 1.0, 'Mujeres')

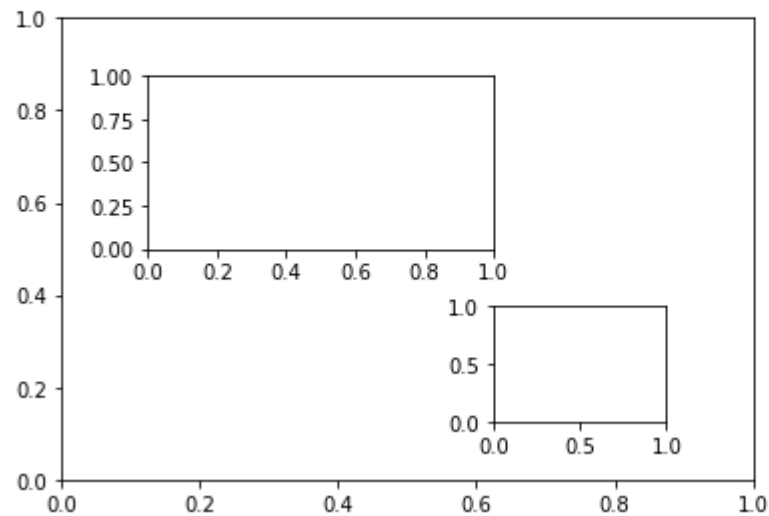


# Ubicando la posición de los ejes

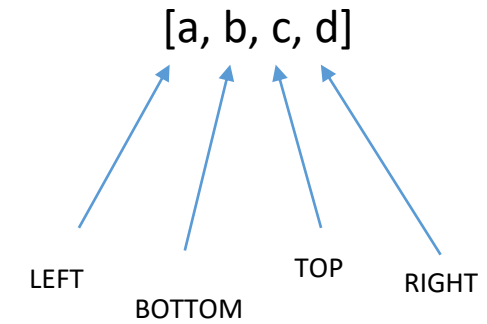
➤ Nótese lo que sucede cuando modificamos los valores de los ejes.

```
In [42]: fig = plt.figure()  
axes1 = fig.add_axes([0.1,0.1,0.8,0.8])  
axes2 = fig.add_axes([0.2,0.5,0.4,0.3])  
axes3 = fig.add_axes([0.6,0.2,0.2,0.2])
```

← Crea canvas vacío



Definen la ubicación relativa del gráfico en el canvas.

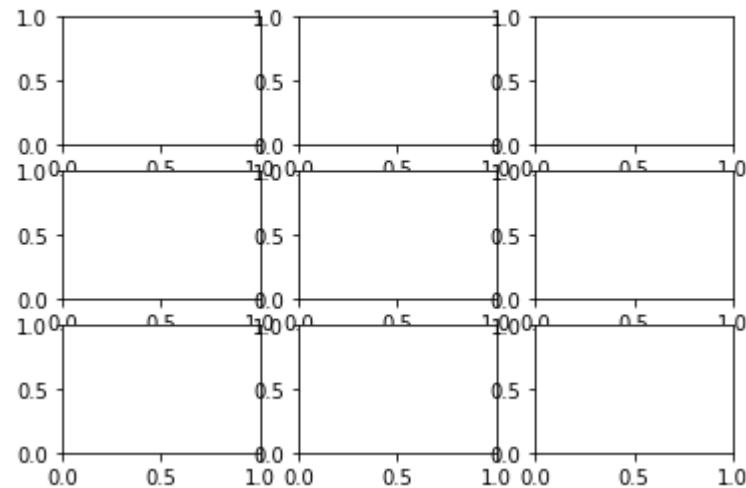


Los valores van entre 0 y 1 (0% y 100%)

# Subplots

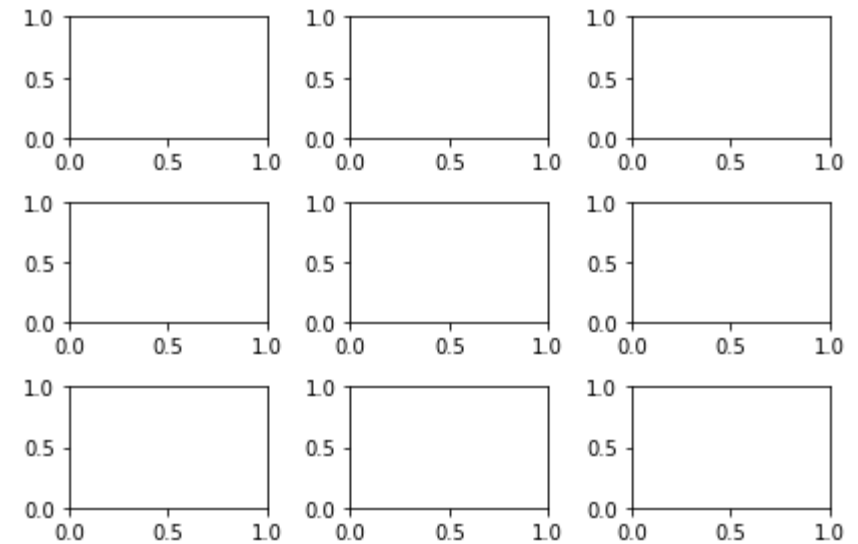
- Volvamos a crear subgráficos, ahora utilizando orientación a objetos.

```
fig, axes = plt.subplots(nrows=3, ncols=3)
```



- Para organizar el espacio entre cada gráfico se usa el método **tight\_layout()**

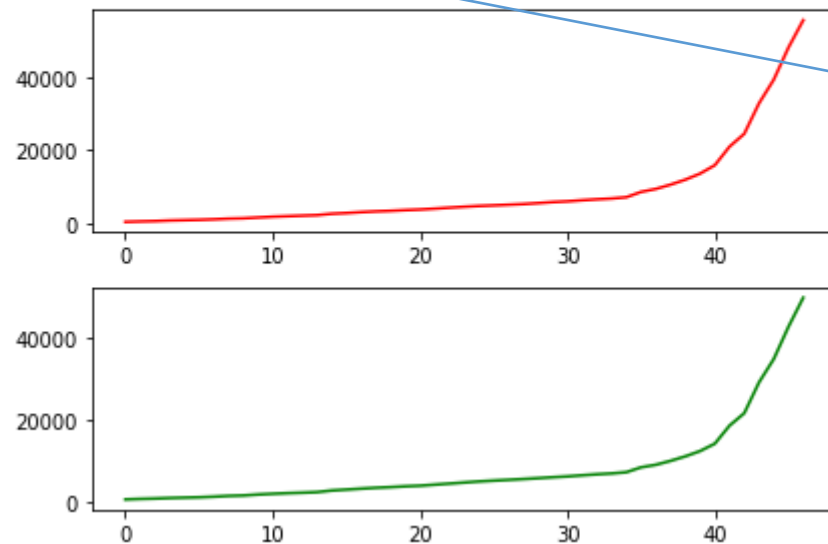
```
fig, axes = plt.subplots(nrows=3, ncols=3)  
fig.tight_layout()
```



# Subplots

- Nótese la forma empaquetada que retorna el método subplots que llena una tupla de datos (fig y axes).

```
fig, axes = plt.subplots(nrows=2, ncols=1)
axes[0].plot(df_m, 'r')
axes[1].plot(df_f, 'g')
fig.tight_layout()
```



Axes es un arreglo de ejes, por lo tanto podemos referenciar cada elemento con la notación []



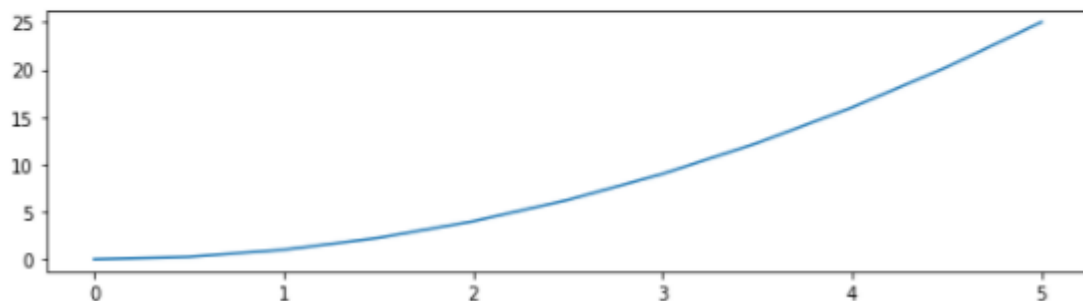
# Tamaño del gráfico



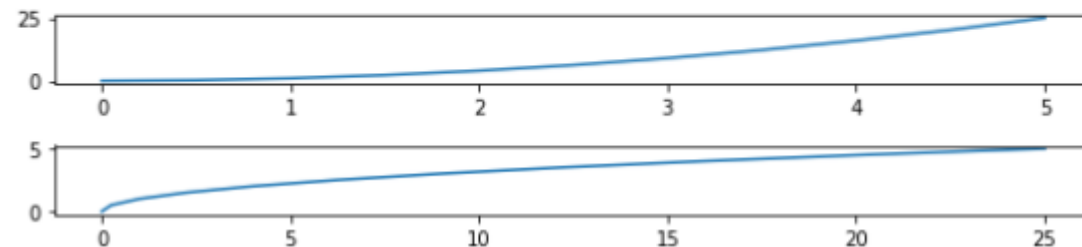
Con el parámetro `figsize`, que recibe una tupla, podemos especificar las dimensiones de la figura (pulgadas).

```
fig = plt.figure(figsize=(8,2))  
axe = fig.add_axes([0,0,1,1])  
axe.plot(x,y)
```

[<matplotlib.lines.Line2D at 0x169764e1780>]



```
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(8,2))  
axes[0].plot(x,y)  
axes[1].plot(y,x)  
plt.tight_layout()
```



Lo mismo se puede especificar al realizar subgráficos.

# Gráficos con más de una serie

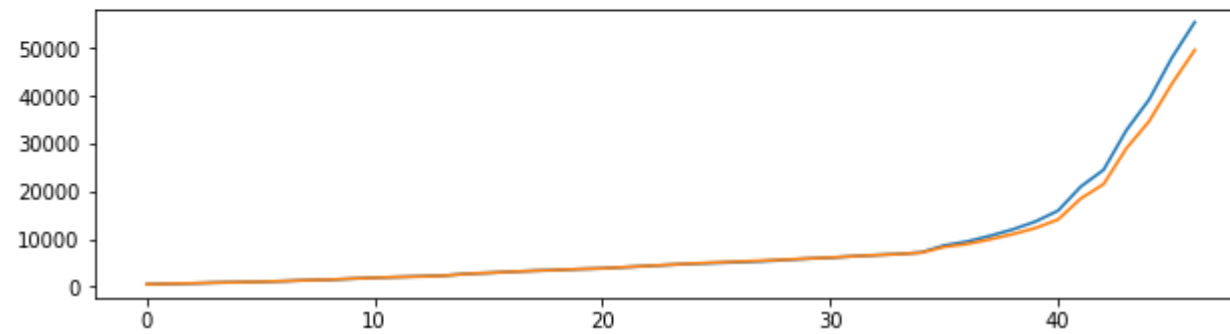


Nótese la forma empaquetada que retorna el método `subplots` que llena una tupla de datos (fig y axes).

Ploteamos en el mismo  
juego de ejes

```
fig = plt.figure(figsize=(8,2))  
axe = fig.add_axes([0,0,1,1])  
axe.plot(df_m)  
axe.plot(df_f)
```

```
[<matplotlib.lines.Line2D at 0x1b7ef3d8488>]
```



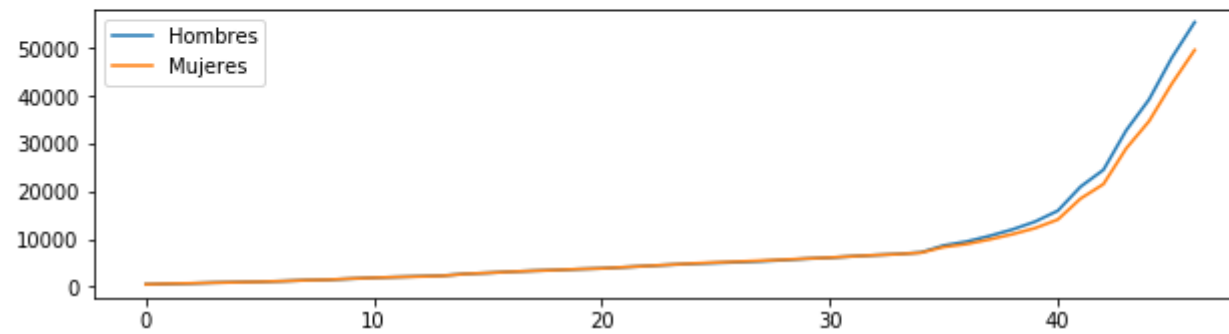
# Agregar leyendas



Podemos crear un gráfico con múltiples líneas y agregar un cuadro de leyendas

```
fig = plt.figure(figsize=(8,2))  
axe = fig.add_axes([0,0,1,1])  
axe.plot(df_m, label='Hombres')  
axe.plot(df_f, label='Mujeres')  
axe.legend(loc=0)
```

<matplotlib.legend.Legend at 0x1b7f117d688>



# Guardar un gráfico



Se pueden especificar distintas extensiones (png, jpg, gif) y se puede especificar la resolución (dpi) de la imagen.

```
fig.savefig('mi_figura.png', dpi=200)
```



 mi\_figura.png



# Apariencia

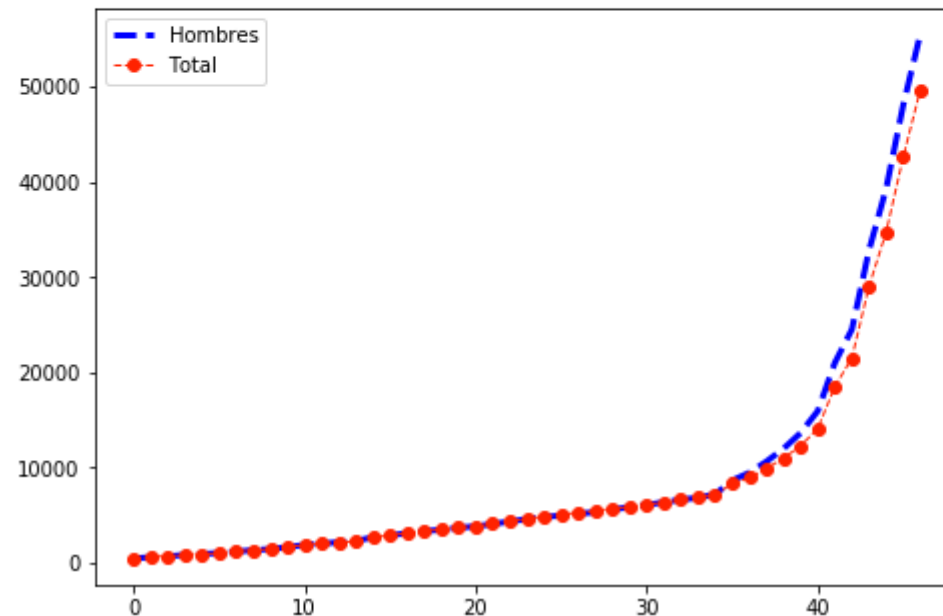


Podemos variar el color del gráfico, el ancho y el estilo de la línea.

red, green, orange, blue, purple, black...  
también puede ser el código RGB

```
fig = plt.figure()
axe = fig.add_axes([0,0,1,1])
axe.plot(df_m, label='Hombres', color='blue', linewidth=3, linestyle='dashed')
axe.plot(df_f, label='Total', color='#FF2200', lw=1, ls='--', marker='o') #o, s, +, 1
axe.legend(loc=0)
# otros parámetros
# markersize, markerfacecolor, markeredgewidth, markeredgewidth
```

<matplotlib.legend.Legend at 0x1b7f120b808>



o, s, +, 1

Otros parámetros:  
**Markersize**  
**Markerfacecolor**  
**Markeredgewidth**  
**Markeredgewidth**

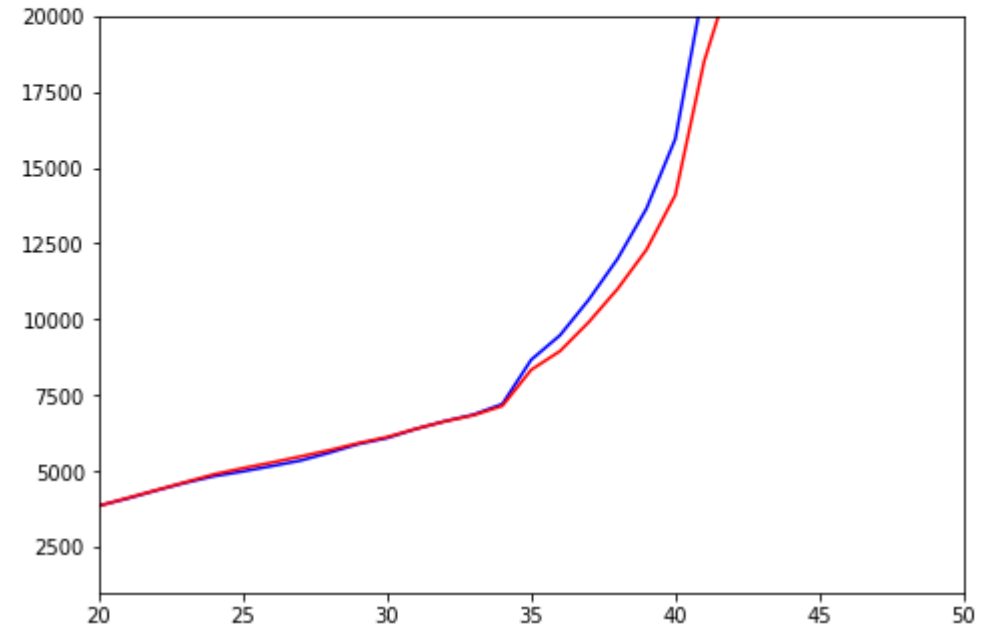
# Límites

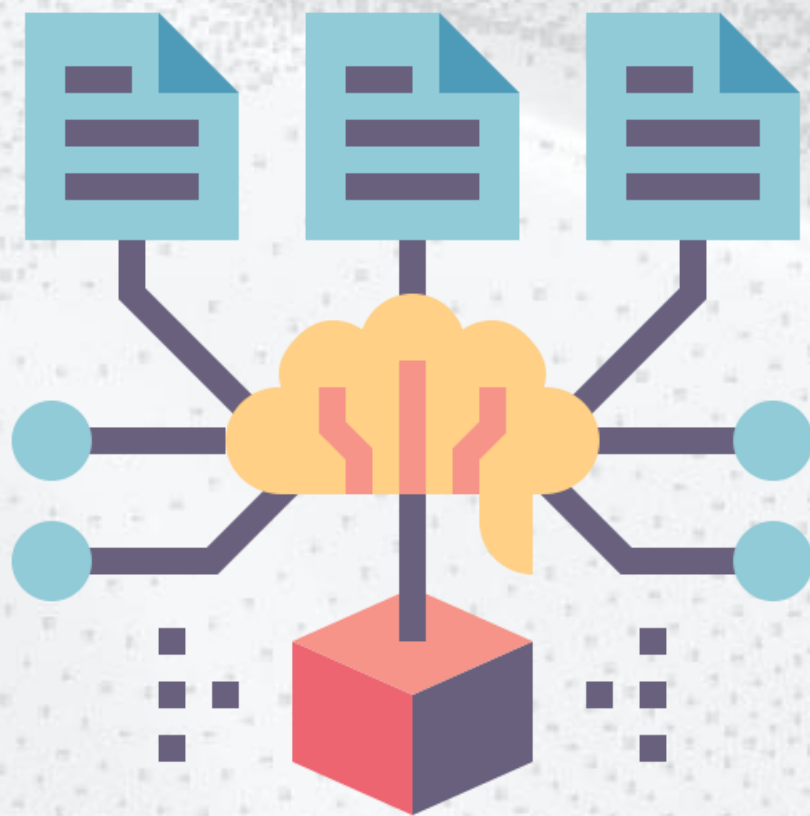


Podemos setear el rango de valores de los ejes con el cual limitar su despliegue.

```
fig = plt.figure()
axe = fig.add_axes([0,0,1,1])
axe.plot(df_m, label='Hombres', color='blue')
axe.plot(df_f, label='Total', color='red')
axe.set_xlim([20,50])
axe.set_ylim([1000,20000])
```

(1000, 20000)





Tipos de gráfico

# Otros tipos de gráfico



Para ilustrar algunos ejemplos, vamos a recurrir al dataset de Salarios de San Francisco:

```
df = pd.read_csv('Salaries.csv')
df.head(2)
```

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	567595.43	567595.43	2011	NaN	San Francisco	NaN
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909.28	538909.28	2011	NaN	San Francisco	NaN

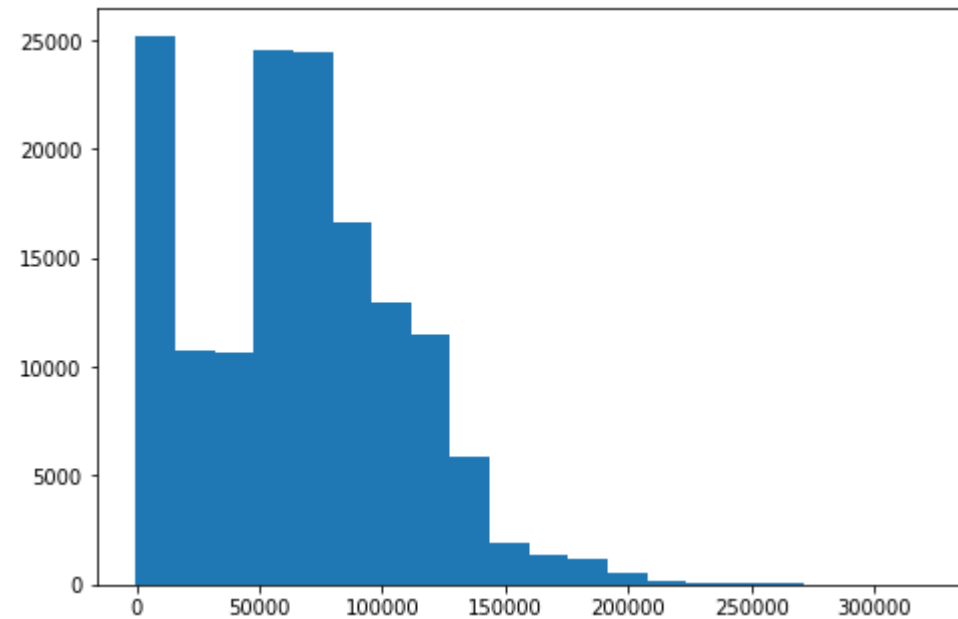


# Histograma



El siguiente es un histograma, que muestra la frecuencia de los valores en cada intervalo.

```
fig = plt.figure()  
axe = fig.add_axes([0,0,1,1])  
axe.hist(df['BasePay'].dropna(), bins=20)
```

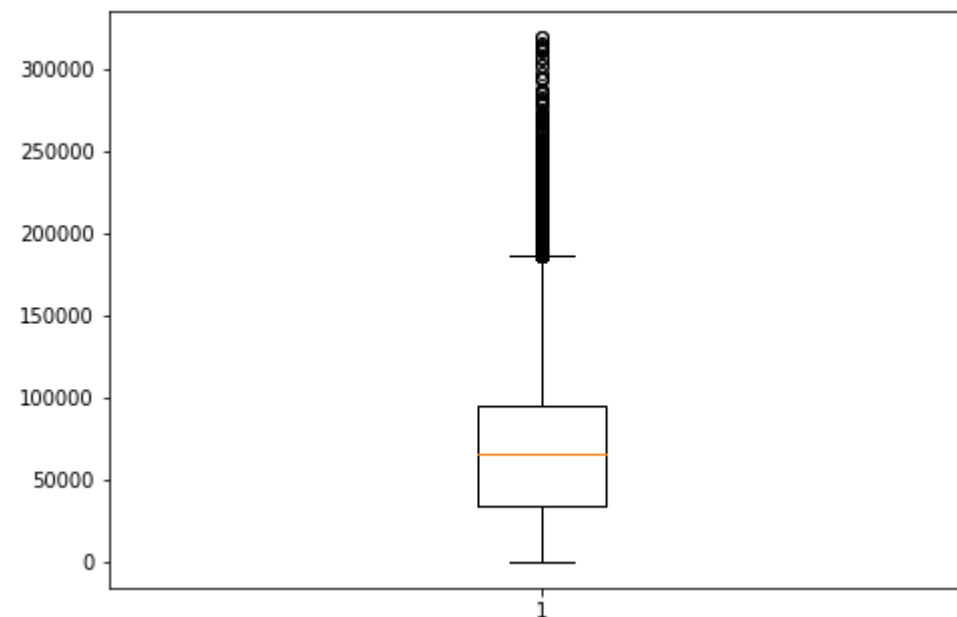


# Diagrama de caja



El siguiente es un diagrama de caja, que muestra el mínimo, máximo, percentil 25, 50 y 75.

```
fig = plt.figure()
axe = fig.add_axes([0,0,1,1])
axe.boxplot(df['BasePay'].dropna())
```



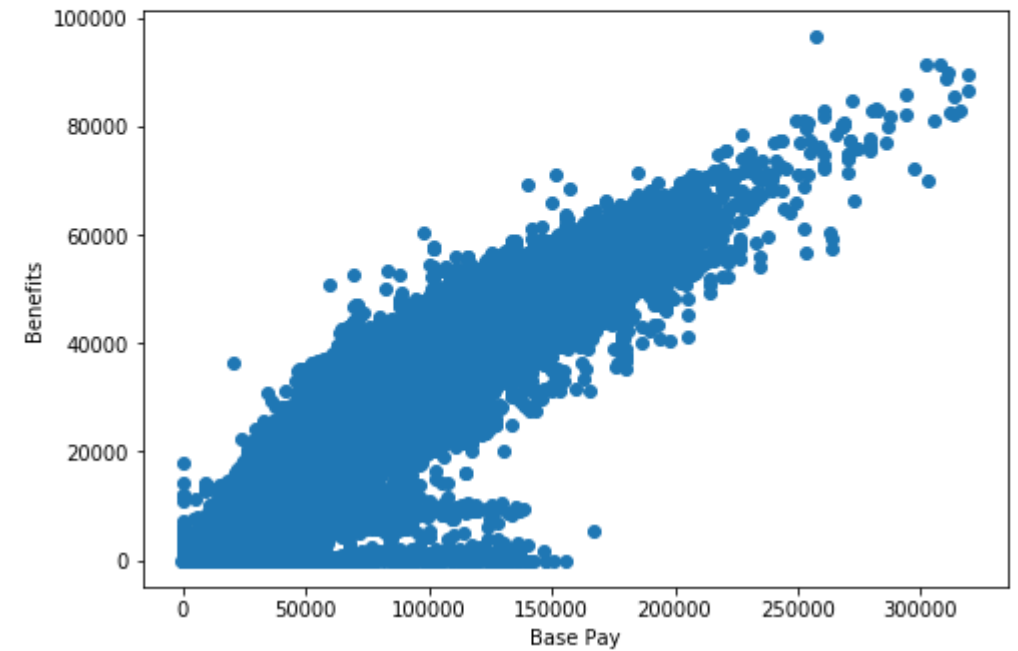
# Diagrama de dispersión



El siguiente es un diagrama de dispersión (scatterplot)

```
fig = plt.figure()
axe = fig.add_axes([0,0,1,1])
axe.scatter(df['BasePay'],df['Benefits'])
axe.set_xlabel('Base Pay')
axe.set_ylabel('Benefits')
```

```
Text(0, 0.5, 'Benefits')
```



# Diagrama de Barras

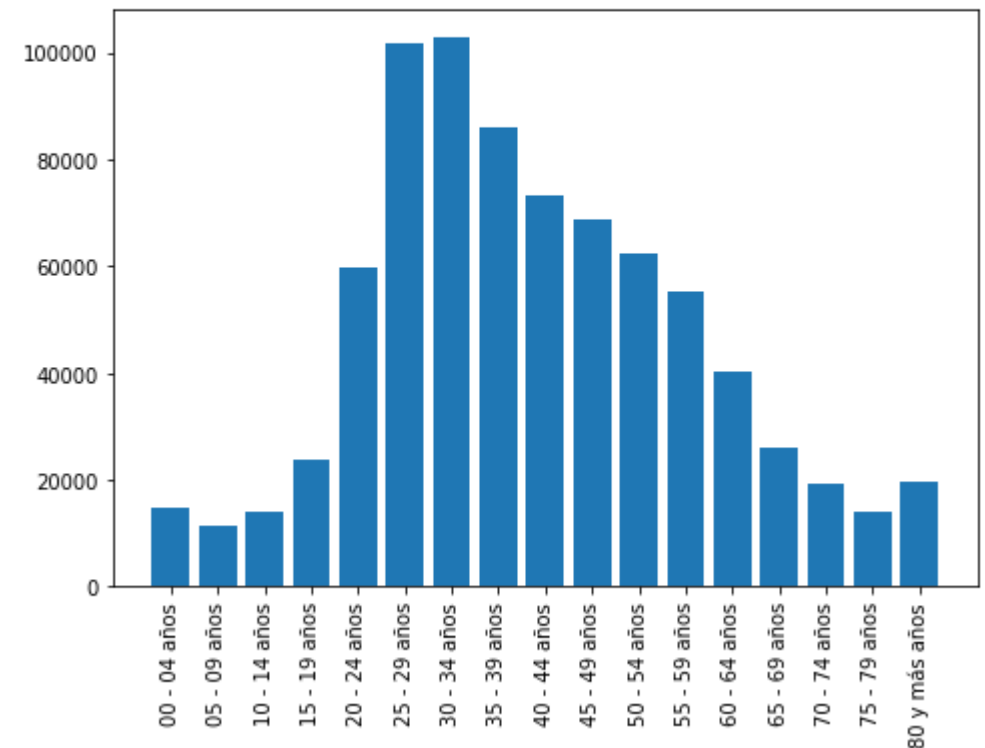
- Volvamos a nuestro ejemplo de Covid en Chile y construyamos un diagrama de barras.

Un poco de wrangling para este ejemplo:

```
df3 = df.groupby('Grupo de edad').sum()
df3.reset_index(inplace=True)
df3.head(2)
```

	Grupo de edad	Contagiados
0	00 - 04 años	14898
1	05 - 09 años	11246

```
fig = plt.figure()
axe = fig.add_axes([0,0,1,1])
axe.bar(df3['Grupo de edad'].values, df3['Contagiados'].values)
axe.tick_params(axis='x', labelrotation=90)
```





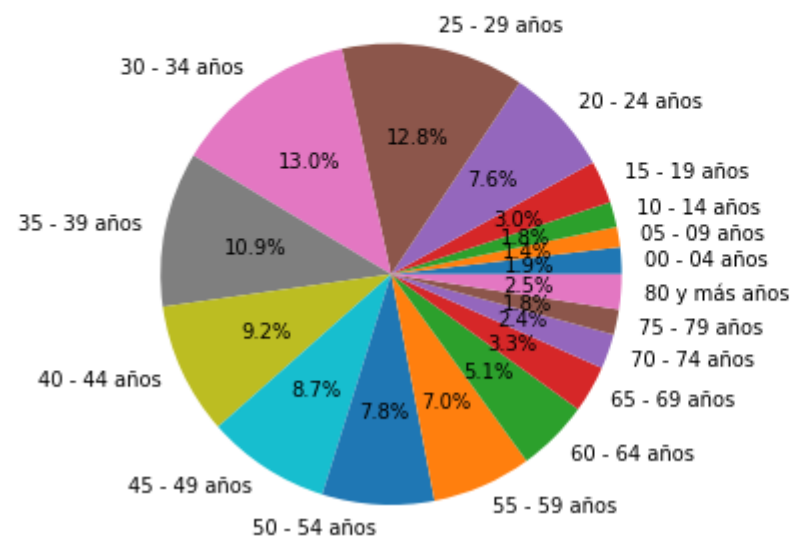
# Diagrama de torta

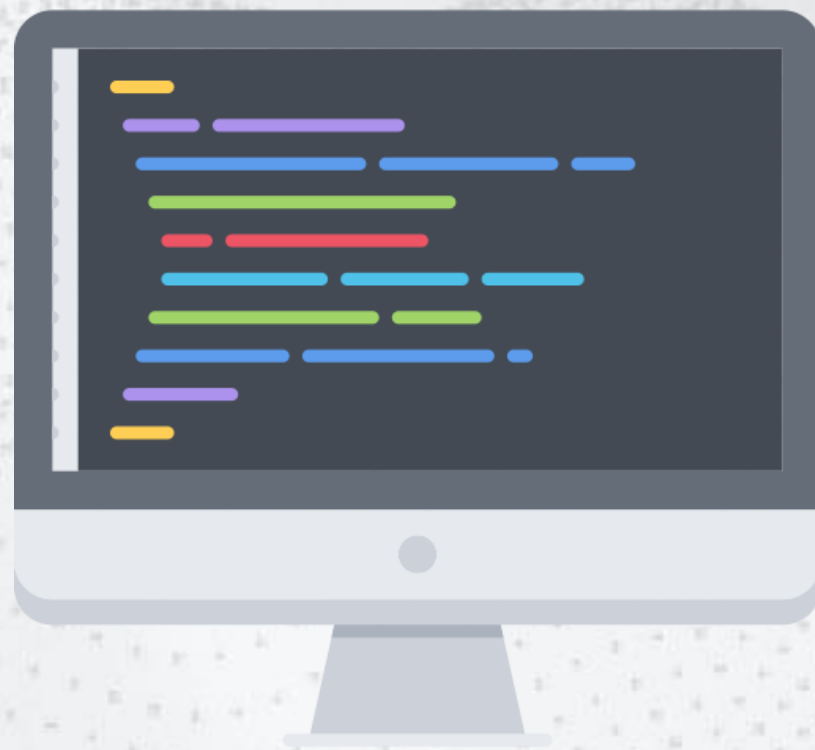
## Construyamos un diagrama de Torta

```
df4 = df.groupby(['Grupo de edad']).sum()
df4.reset_index(inplace=True)
df4.head(2)
```

	Grupo de edad	Contagiados
0	00 - 04 años	14898
1	05 - 09 años	11246

```
fig = plt.figure()
axe = fig.add_axes([0,0,1,1])
axe.pie(df4['Contagiados'].values, labels=df4['Grupo de edad'].values, autopct='%1.1f%%')
axe.tick_params(axis='x', labelrotation=90)
```





Estilos

# Estilos disponibles



Un estilo es una combinación de colores y formas que puede ajustarse un gráfico. A continuación, obtenemos el listado de estilos disponibles en el entorno.

Con esto, podemos fijar una apariencia pre-establecida a todos los gráficos que construyamos con la librería matplotlib.

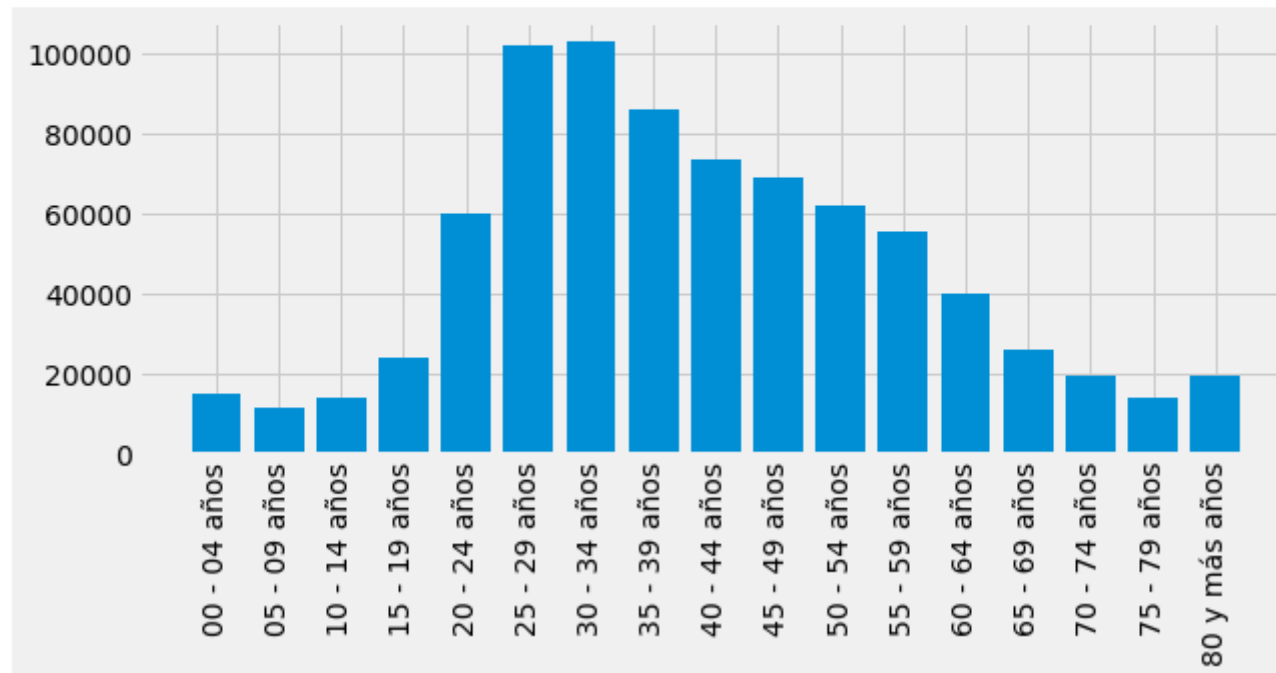
```
plt.style.available
```

```
['bmh',  
 'classic',  
 'dark_background',  
 'fast',  
 'fivethirtyeight',  
 'ggplot',  
 'grayscale',  
 'seaborn-bright',  
 'seaborn-colorblind',  
 'seaborn-dark-palette',  
 'seaborn-dark',  
 'seaborn-darkgrid',  
 'seaborn-deep',  
 'seaborn-muted',  
 'seaborn-notebook',  
 'seaborn-paper',  
 'seaborn-pastel',  
 'seaborn-poster',  
 'seaborn-talk',  
 'seaborn-ticks',  
 'seaborn-white',  
 'seaborn-whitegrid',
```

# Estilos disponibles

```
plt.style.use('fivethirtyeight')
```

```
fig = plt.figure(figsize=(8,3))  
axe = fig.add_axes([0,0,1,1])  
axe.bar(df4['Grupo de edad'].values, df4['Contagiados'].values)  
axe.tick_params(axis='x', labelrotation=90)
```

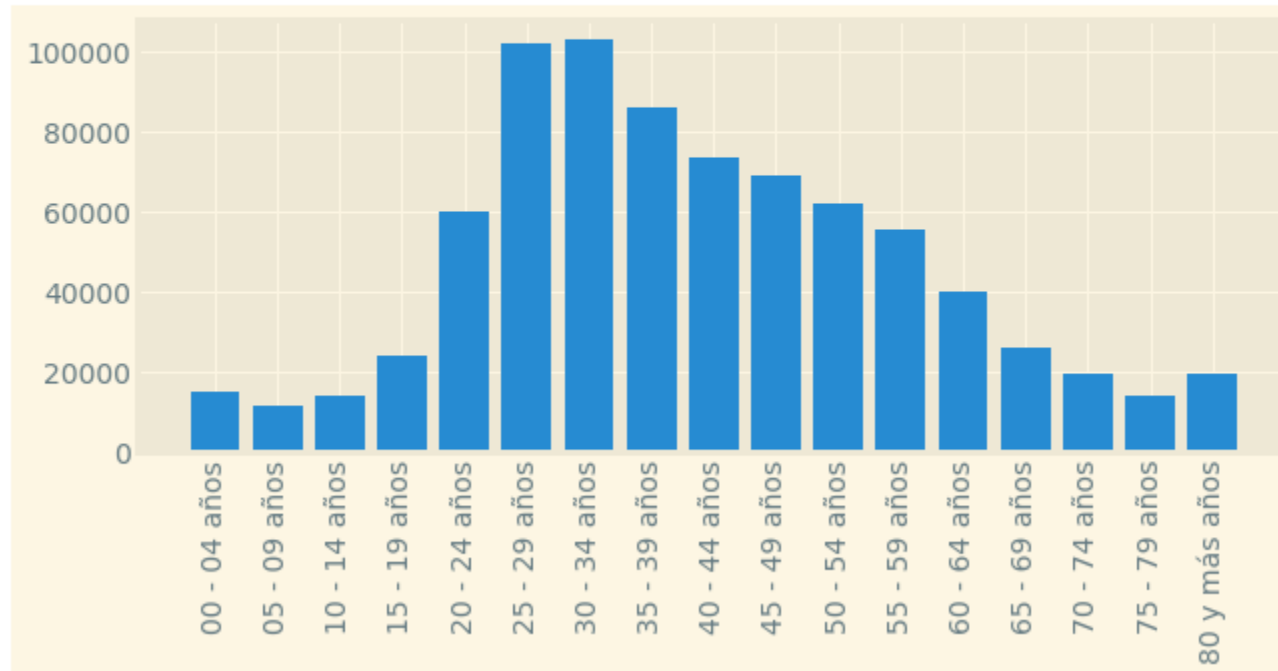




# Estilos disponibles

```
plt.style.use('Solarize_Light2')
```

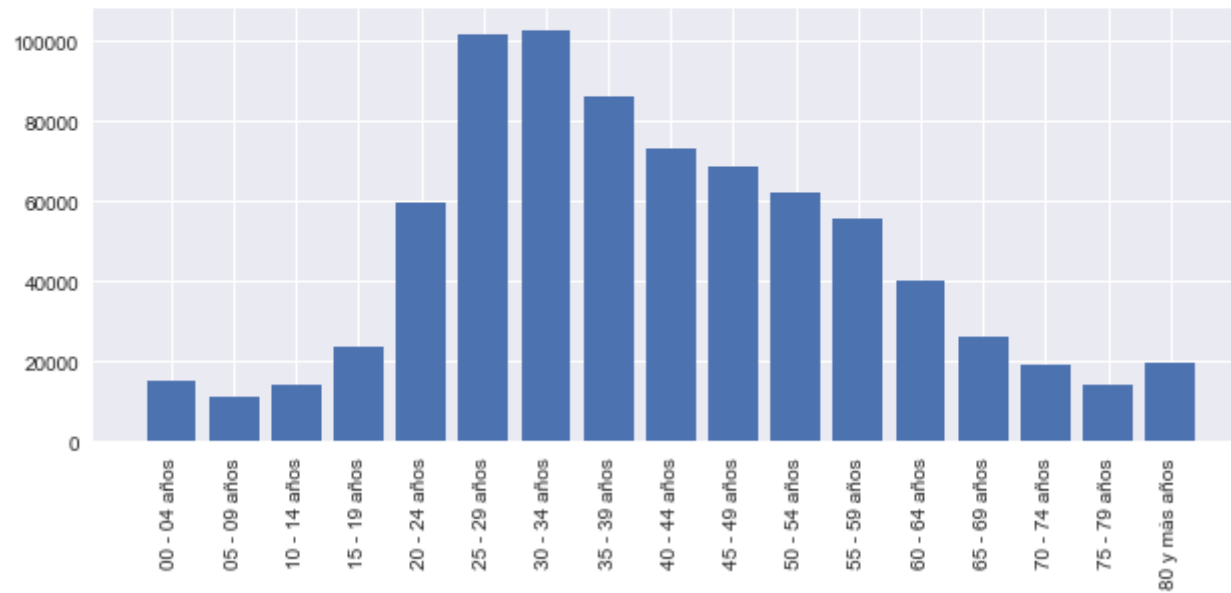
```
fig = plt.figure(figsize=(8,3))  
axe = fig.add_axes([0,0,1,1])  
axe.bar(df4['Grupo de edad'].values, df4['Contagiados'].values)  
axe.tick_params(axis='x', labelrotation=90)
```



# Estilos disponibles

```
plt.style.use('seaborn')
```

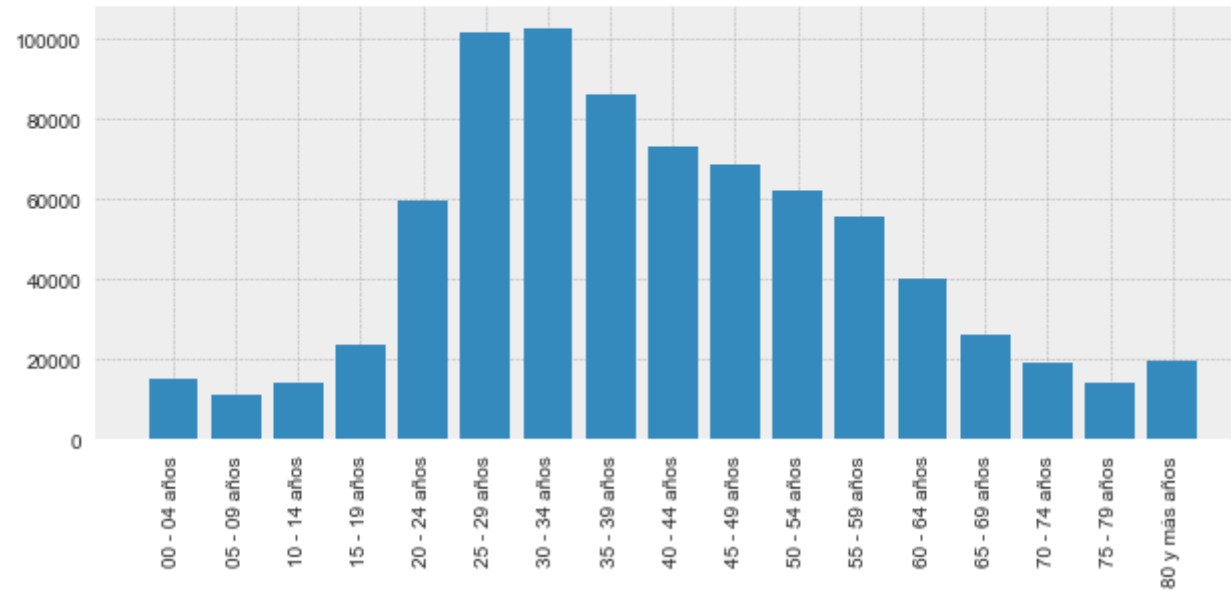
```
fig = plt.figure(figsize=(8,3))  
axe = fig.add_axes([0,0,1,1])  
axe.bar(df4['Grupo de edad'].values, df4['Contagiados'].values)  
axe.tick_params(axis='x', labelrotation=90)
```



# Estilos disponibles

```
plt.style.use('bmh')
```

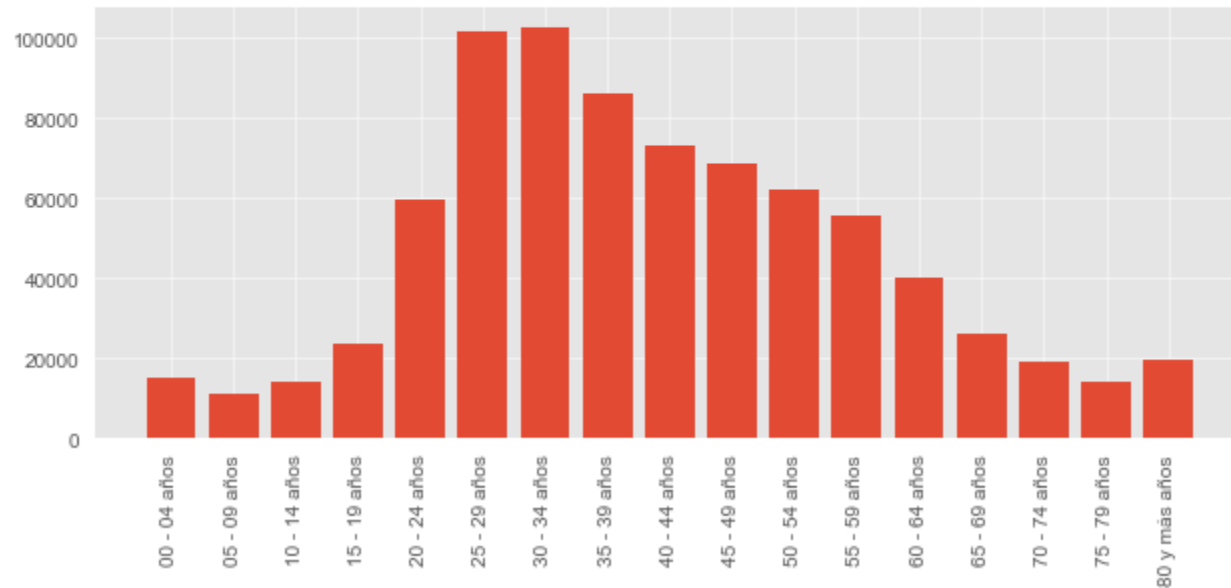
```
fig = plt.figure(figsize=(8,3))  
axe = fig.add_axes([0,0,1,1])  
axe.bar(df4['Grupo de edad'].values, df4['Contagiados'].values)  
axe.tick_params(axis='x', labelrotation=90)
```



# Estilos disponibles

```
plt.style.use('ggplot')
```

```
fig = plt.figure(figsize=(8,3))  
axe = fig.add_axes([0,0,1,1])  
axe.bar(df4['Grupo de edad'].values, df4['Contagiados'].values)  
axe.tick_params(axis='x', labelrotation=90)
```







Gracias