Matplotlib

November 8, 2017

1 Matplotlib

Matplotlib se trata de un biblioteca de visualización 2D de Python. Matplotlib nos permite generar gráficos de calidad en una gran variedad de formatos. El primer paso para utlizar la librería es importarla.

```
In [14]: #Importamos pyplot de matplotlib
         import matplotlib.pyplot as plt
```

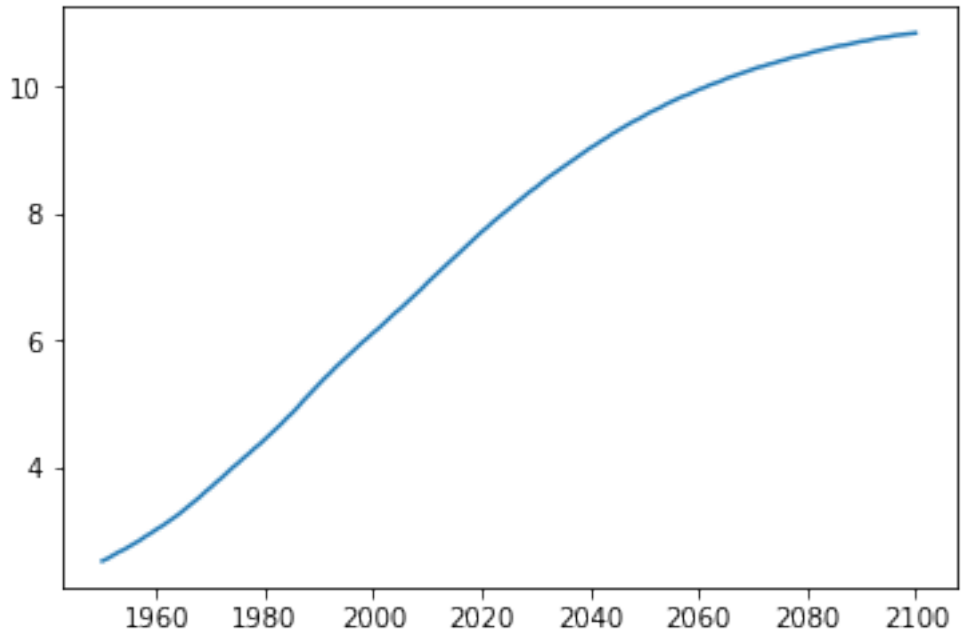
Como toda librería de visualización el gráfico más básico que podemos mostrar es el gráfico de tipo línea, para realizar este tipo de gráfico basta con ejecutar el comando plt.plot(x,y), donde x e y son los datos que queremos representar en cada uno de los ejes.

```
In [3]: #Nos creamos dos listas numéricas que representan año y población mundial en billones
        #de habitantes
        year = [1950,1951,1952,1953,1954,1955,1956,1957,1958,1959,1960,1961,1962,
         1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,
         1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,
         1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,
         2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,
         2015,2016,2017,2018,2019,2020,2021,2022,2023,2024,2025,2026,2027,
         2028,2029,2030,2031,2032,2033,2034,2035,2036,2037,2038,2039,2040,
         2041,2042,2043,2044,2045,2046,2047,2048,2049,2050,2051,2052,2053,
         2054,2055,2056,2057,2058,2059,2060,2061,2062,2063,2064,2065,2066,
         2067,2068,2069,2070,2071,2072,2073,2074,2075,2076,2077,2078,2079,
         2080,2081,2082,2083,2084,2085,2086,2087,2088,2089,2090,2091,2092,
         2093,2094,2095,2096,2097,2098,2099,2100]

        poblacion = [2.53,2.57,2.62,2.67,2.71,2.76,2.81,2.86,2.92,2.97,
         3.03,3.08,3.14,3.2,3.26,3.33,3.4,3.47,3.54,3.62,3.69,3.77,3.84,
         3.92,4.0,4.07,4.15,4.22,4.3,4.37,4.45,4.53,4.61,4.69,4.78,4.86,
         4.95,5.05,5.14,5.23,5.32,5.41,5.49,5.58,5.66,5.74,5.82,5.9,5.98,
         6.05,6.13,6.2,6.28,6.36,6.44,6.51,6.59,6.67,6.75,6.83,6.92,7.0,
         7.08,7.16,7.24,7.32,7.4,7.48,7.56,7.64,7.72,7.79,7.87,7.94,8.01,
         8.08,8.15,8.22,8.29,8.36,8.42,8.49,8.56,8.62,8.68,8.74,8.8,8.86,
         8.92,8.98,9.04,9.09,9.15,9.2,9.26,9.31,9.36,9.41,9.46,9.5,9.55,
         9.6,9.64,9.68,9.73,9.77,9.81,9.85,9.88,9.92,9.96,9.99,10.03,
```

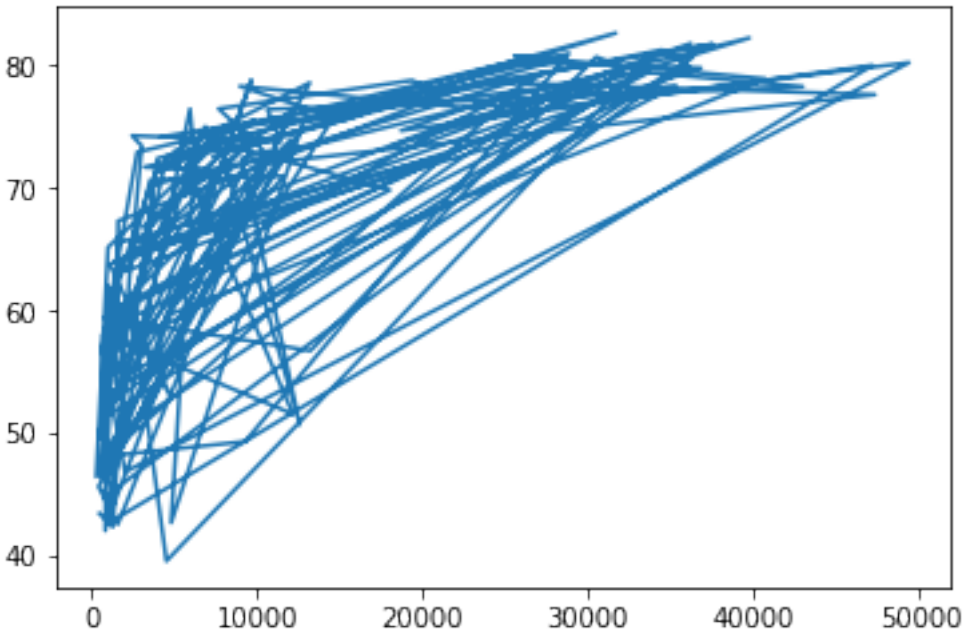
```
10.06,10.09,10.13,10.16,10.19,10.22,10.25,10.28,10.31,10.33,
10.36,10.38,10.41,10.43,10.46,10.48,10.5,10.52,10.55,10.57,
10.59,10.61,10.63,10.65,10.66,10.68,10.7,10.72,10.73,10.75,
10.77,10.78,10.79,10.81,10.82,10.83,10.84,10.85]

#Realizamos una representación gráfica de los datos
plt.plot(year, poblacion)
plt.show()
```



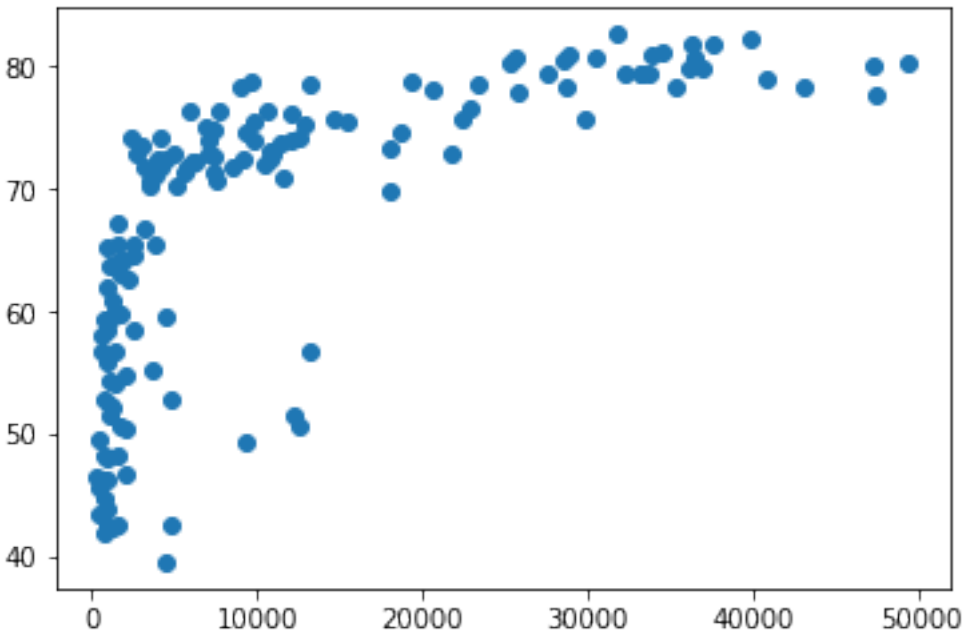
A continuación vamos a trabajar con el fichero de datos gapminder.csv, este fichero contiene información sobre la esperanza de vida de las personas, renta per capita, población para distintos países del mundo para el año 2007.

```
In [21]: #Cargamos el dataset
         import pandas as pd
         gap_minder = pd.read_csv('gapminder.csv')
         #Nos quedamos con las columnas life_exp y gdp_cap
         esperanza_vida = gap_minder['life_exp']
         renta = gap_minder['gdp_cap']
         #Nos creamos un plot
         plt.plot(renta, esperanza_vida)
         plt.show()
```



En este caso podemos ver que este tipo de visualización no nos aclara nada respecto a nuestros datos. Cuando queremos ver si existe una correlación entre dos variables, la mejor visualización es la que se conoce como scatter plots. Python dispone del comando plt.scatter(x,y) que nos permite realizar este tipo de visualizaciones.

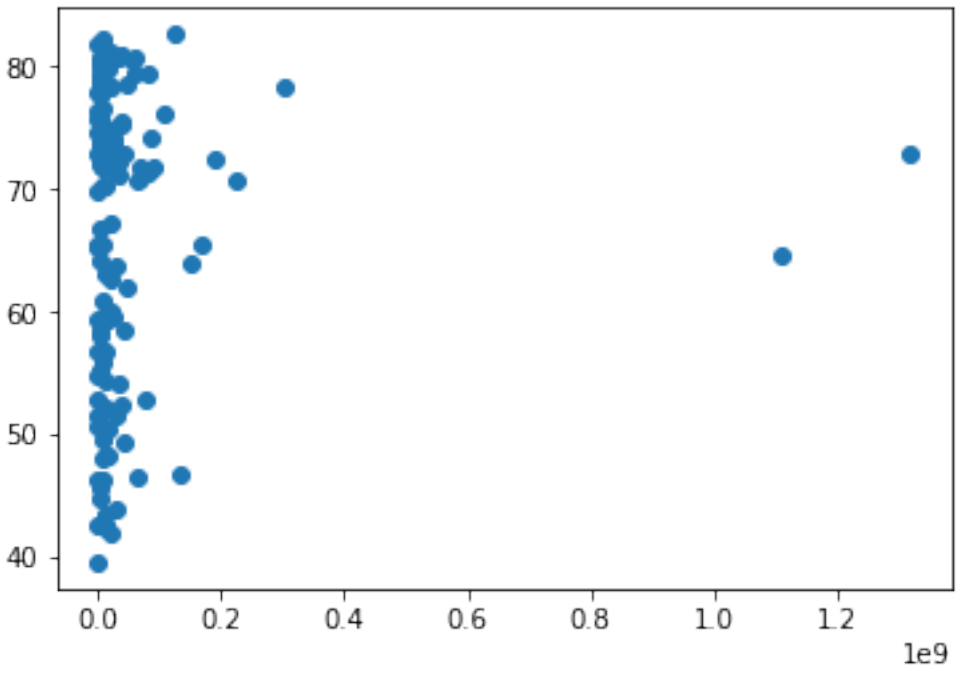
```
In [11]: #Nos creamos nuestra primera visualización de tipo scatter
         plt.scatter(renta, esperanza_vida)
         plt.show()
```



Este gráfico nos muestra de una forma más clara los datos. Podemos ver que los países con una elevada esperanza de vida (mayor a 80 años) se tratan de países con una elevada renta. Por otro lado podemos ver ciertos puntos con comportamientos fuero de lo normal, por ejemplo se pueden observar situaciones de países con una renta cercana a los 15000 y con una esperanza de vida por debajo de los 60 años. También podemos ver países con una renta similar pero con diferencias considerables en su esperanza de vida, lo que nos indica que existen otros factores (como era de esperar) que influyen en la esperanza de vida de un país. Finalmente podemos decir que inicialmente se aprecia una tendencia lineal constante hasta cerca de los 70 años, a partir de los 70 podemos ver como este comportamiento cambia, la gráfica se adapta a una tendencia logarítmica.

Como acabamos elegir el gráfico adecuado a la hora de visualizar es fundamental para poder realizar un análises previo de estos, una correcta visualización nos ayudará a comprender mejor nuestros y datos y por lo tanto nos ayudará a sacar un mayor partido a estos. £Existe algún de relación entre la población y la esperanza de vida?

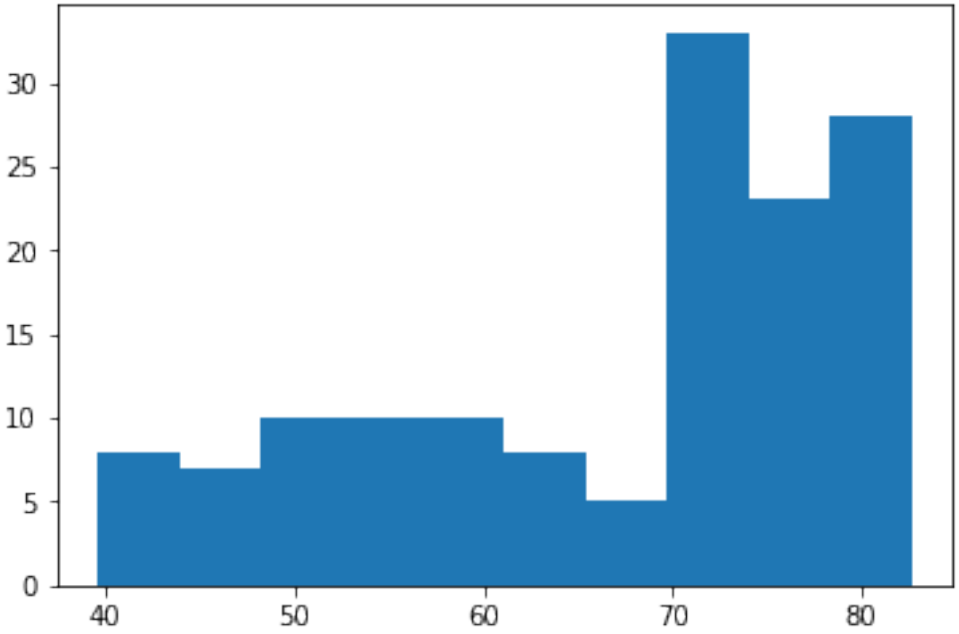
```
In [26]: #Nos quedamos con la población de las personas
         poblacion = gap_minder['population']
         #Nos creamos nuestro scatter
         plt.scatter(poblacion, esperanza_vida)
         plt.show()
```



Podemos ver como en este caso la relación existente entre la población de un país y su esperanza de vida es totalmente nula.

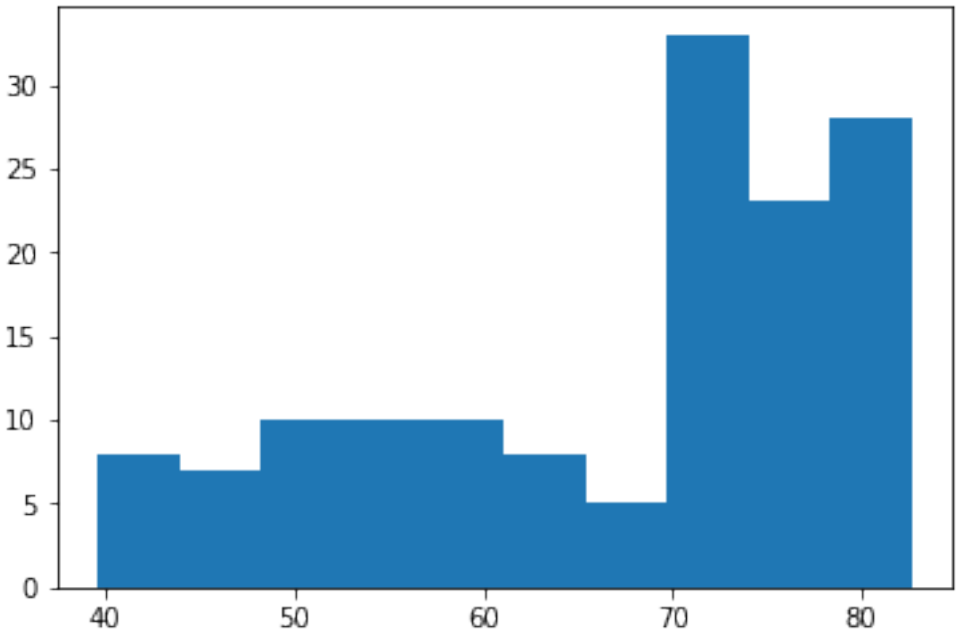
Otro de los gráficos más útiles a la hora de analizar datos son los histogramas. Python dispone del comando plt.hist() que nos permite realizar el histograma de una variable.

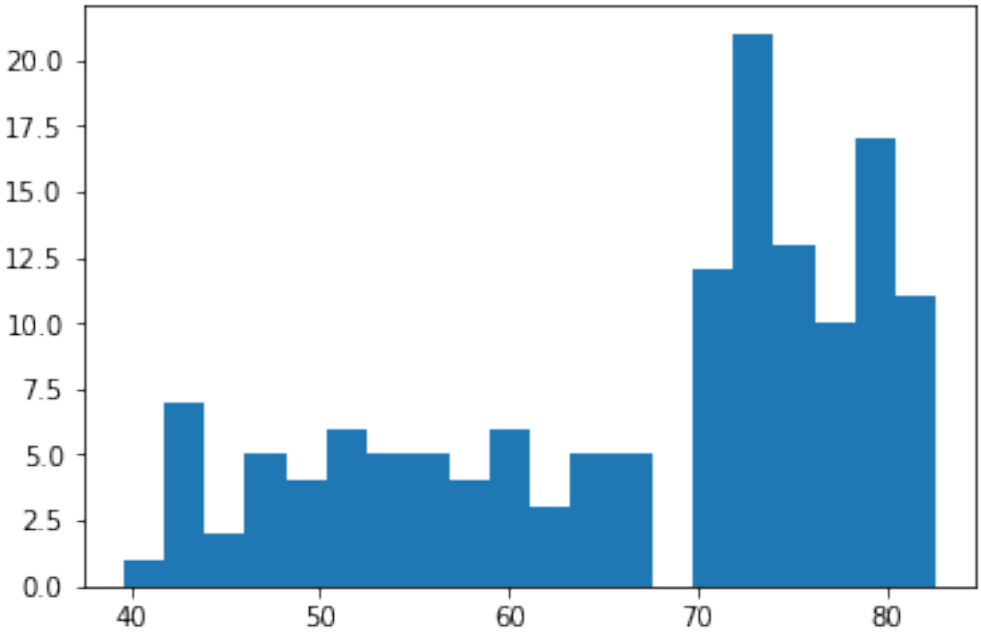
```
In [15]: #Filtramos y obtenemos la esperanza de vida de los países para el año 2007
         gap_minder2007 = gap_minder[gap_minder.year == 2007]
         gap_minder2007_lexp = gap_minder2007['life_exp']
         #Representamos mediante un histograma la esperanza de vida en el año 2007
         plt.hist(gap_minder2007_lexp)
         plt.show()
```



Uno de los parámetros más importantes de un histograma es el número de bins que deseamos representar, es decir, el periodo que vamos usar para juntar la información, por defecto toma el valor de 10, pero esto puede modicarse dándole el valor que estimemos oportuno. A continuación vamos hacer dos plots para ello vamos hacer uso del comando plt.clf() que es trata de un comando que lo hace es limpiar un plot para poder hacer otro.

```
In [16]: #Mostramos la esperanza de vida en el año 2007 con 10 bins
         plt.hist(gap_minder2007_lexp)
         plt.show()
         #Hacemos un cleaning
         plt.clf()
         #Mostramos la esperanza de vida con 20 bins
         plt.hist(gap_minder2007_lexp, bins = 20)
         plt.show()
         #Hacemos un cleaning
         plt.clf()
```

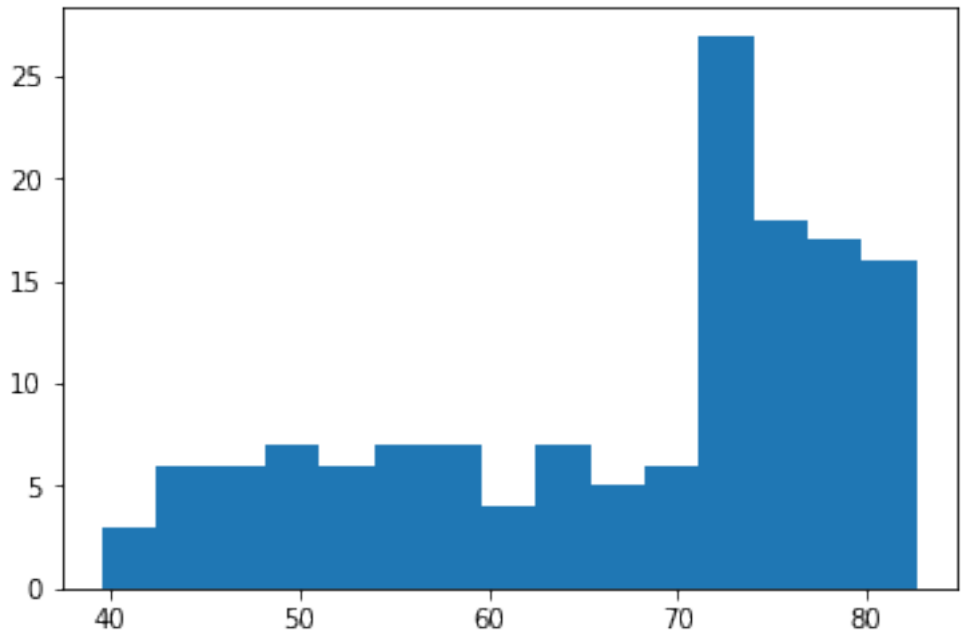


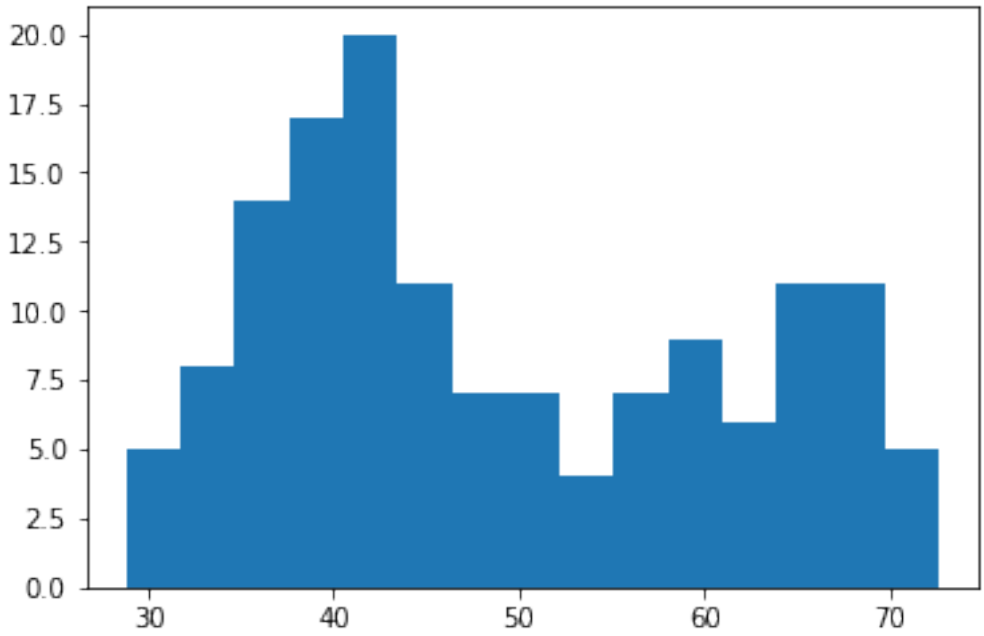


Una de las utilidades de los histogramas es comparar, por ejemplo podemos comparar la distrbución de la esperanza de vida entre los años 2007 y 1950.

```
In [20]: #Nos creamos una lista con la esperanza de vida para el año 1950
         gap_minder1950_lexp = [28.8,55.23,43.08,30.02,62.48,69.12,66.8,50.94,
```

```
 37.48,68.0,38.22,40.41,53.82,47.62,50.92,59.6,
 31.98,39.03,39.42,38.52,68.75,35.46,38.09,54.74,
 44.0,50.64,40.72,39.14,42.11,57.21,40.48,61.21,
 59.42,66.87,70.78,34.81,45.93,48.36,41.89,45.26,
 34.48,35.93,34.08,66.55,67.41,37.0,30.0,67.5,43.15,
 65.86,42.02,33.61,32.5,37.58,41.91,60.96,64.03,72.49,
 37.37,37.47,44.87,45.32,66.91,65.39,65.94,58.53,63.03,
 43.16,42.27,50.06,47.45,55.56,55.93,42.14,38.48,42.72,
 36.68,36.26,48.46,33.68,40.54,50.99,50.79,42.24,59.16,
 42.87,31.29,36.32,41.72,36.16,72.13,69.39,42.31,37.44,
 36.32,72.67,37.58,43.44,55.19,62.65,43.9,47.75,61.31,
 59.82,64.28,52.72,61.05,40.0,46.47,39.88,37.28,58.0,
 30.33,60.4,64.36,65.57,32.98,45.01,64.94,57.59,38.64,
 41.41,71.86,69.62,45.88,58.5,41.22,50.85,38.6,59.1,
 44.6,43.58,39.98,69.18,68.44,66.07,55.09,40.41,43.16,
 32.55,42.04,48.45]
#Mostramos la esperanza de vida en 2007
plt.hist(gap_minder2007_lexp, bins = 15)
plt.show()
#Hacemos un cleaning
plt.clf()
#Mostramos la esperanza de vida en 1950
plt.hist(gap_minder1950_lexp, bins = 15)
plt.show()
#Hacemos un cleaning
plt.clf()
```

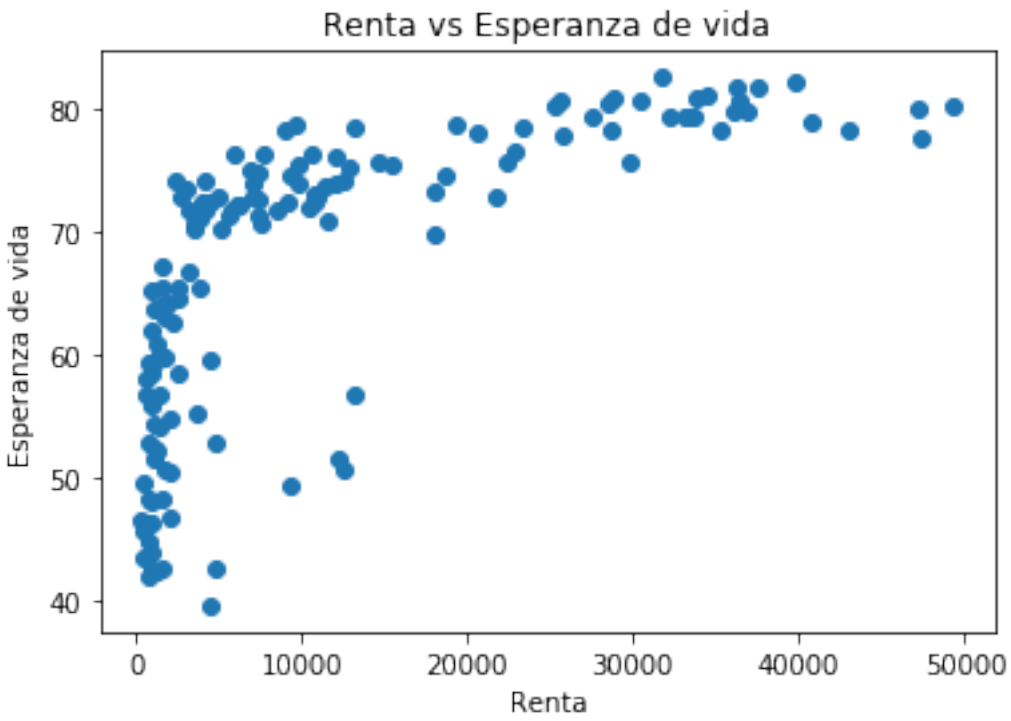




Podemos ver como el histograma muestra las diferencias de forma clara entre ambos datos. Mientras que en el año 2007 la esperanza de vida se concentra en su mayoría entre los 65 y los 85, en 1950 está entre los 30 y los 55.

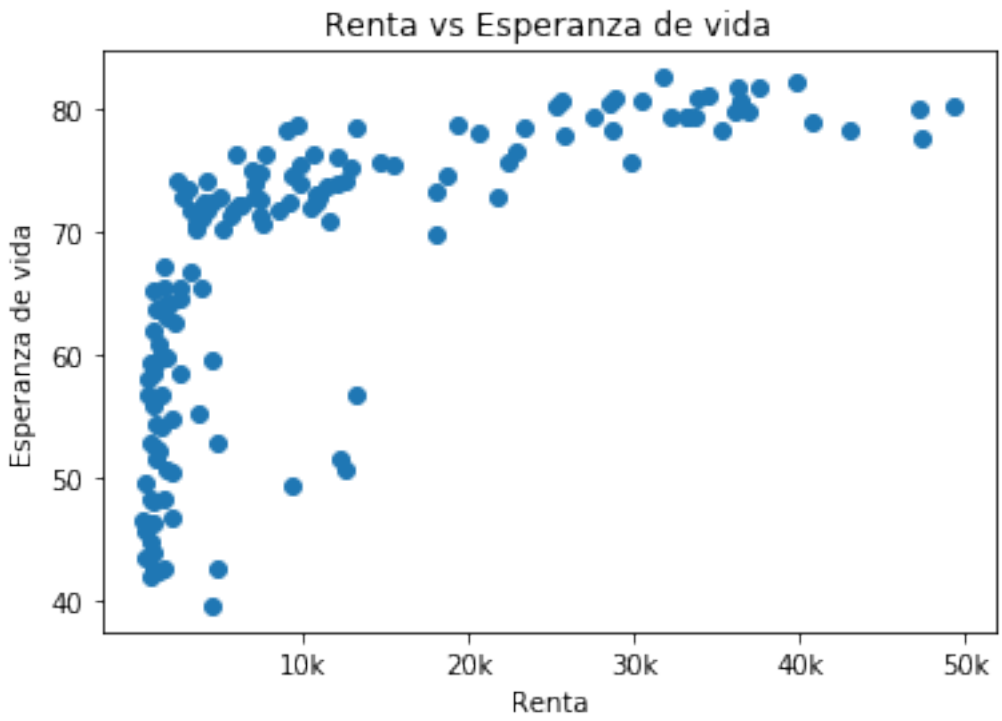
Python nos permite añadir una serie de elementos a nuestro gráficos de forma que estos quedan mucho más claros a la vista de las personas. Por ejemplo nos permite asignar un nombre a cada uno de los ejes y añadir un título al gráfico. Para ello contamos con plt.xlabel, plt.ylabel y plt.title.

```
In [24]: #Realizamos un gráfico de tipo scatter entre la renta per capita y la esperanza de vida
         plt.scatter(renta, esperanza_vida)
         #Le añadimos nombre al ejex, ejey y le ponemos un título
         plt.xlabel('Renta')
         plt.ylabel('Esperanza de vida')
         plt.title('Renta vs Esperanza de vida')
         plt.show()
```



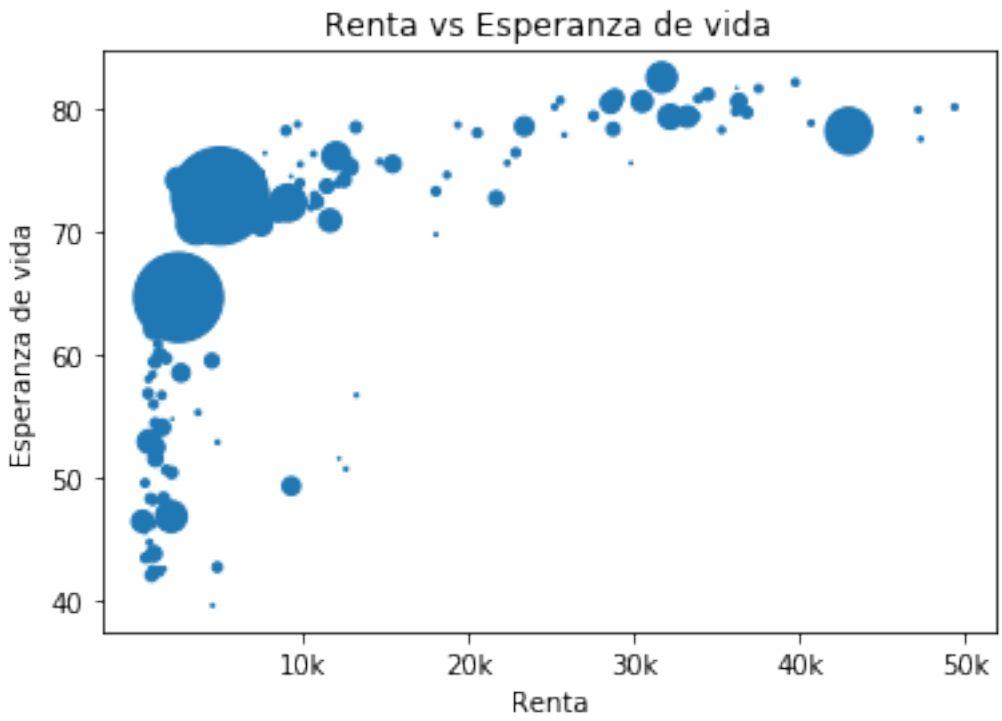
Otro elemento que podemos añadir a nuestro gráfico son los denominados ticks, que nos permiten cambiar el nombre a los propios valores que toma un eje. Por ejemplo podemos realizar el siguiente cambio para que nuestro gráfico visualmente sea más claro.

```
In [25]: #Realizamos un gráfico de tipo scatter entre la renta per capita y la esperanza de vida
         plt.scatter(renta, esperanza_vida)
         #Le añadimos nombre al ejex, ejey y le ponemos un título
         plt.xlabel('Renta')
         plt.ylabel('Esperanza de vida')
         plt.title('Renta vs Esperanza de vida')
         #Añadimos ticks a nuestro grafo
         tick_val = [10000,20000, 30000, 40000, 50000]
         tick_lab = ['10k', '20k', '30k', '40k', '50k']
         plt.xticks(tick_val, tick_lab)
         plt.show()
```



Otro factor que podemos modificar en un scatter plot es el tamaño de los puntos, por ejemplo podemos poner el tamaño de los puntos en función de la población del país, es decir, cuanto mayor sea el punto más población tendrá el país, de esta forma agregamos información bastante útil. Para esto se dispone del parámetro s dentro del gráfico de tipo scatter.

```
In [30]: import numpy as np
         #Nos creamos un array con el valor de la población para poder normalizar
         poblacion_np = np.array(poblacion)
         poblacion_np = poblacion_np / 1000000
         #Realizamos un gráfico de tipo scatter entre la renta per capita y la esperanza de vida
         plt.scatter(renta, esperanza_vida, s = poblacion_np)
         #Le añadimos nombre al ejex, ejey y le ponemos un título
         plt.xlabel('Renta')
         plt.ylabel('Esperanza de vida')
         plt.title('Renta vs Esperanza de vida')
         #Añadimos ticks a nuestro grafo
         tick_val = [10000,20000, 30000, 40000, 50000]
         tick_lab = ['10k', '20k', '30k', '40k', '50k']
         plt.xticks(tick_val, tick_lab)
         plt.show()
```

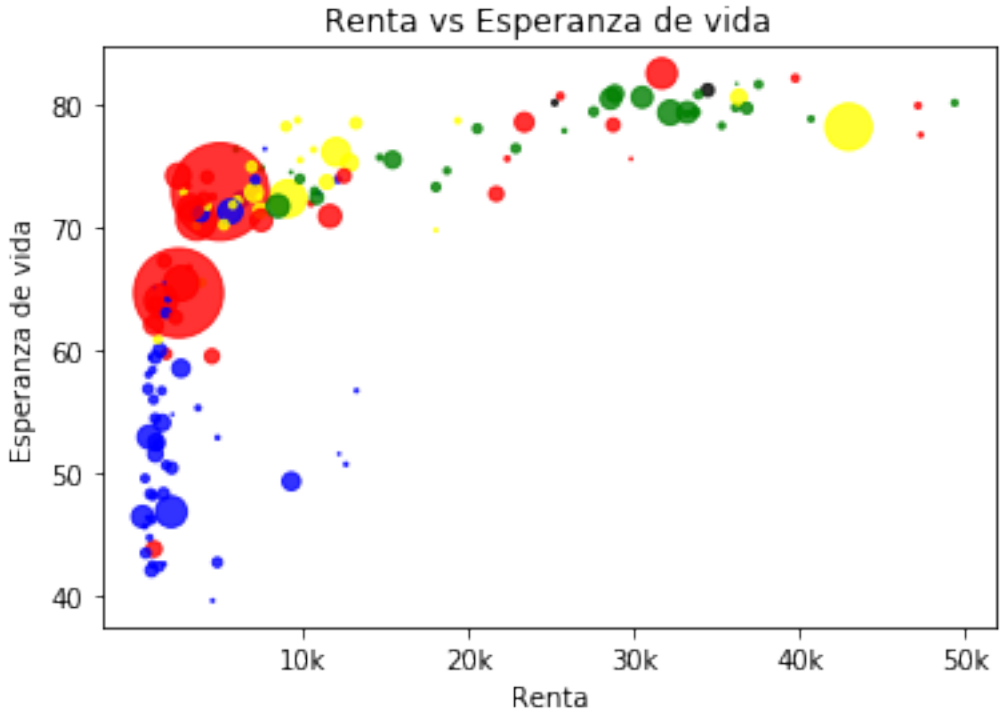


Otra de las cosas que podemos variar es el color de los puntos en función de una determinada variable, por ejemplo, podemos asignar un color a cada uno de los continentes. Para ello contamos con el parámetro c dentro de scatte, también contamos con el parámetro alpha que toma un valor entre 0 y 1, que nos indica el nivel de transparencia, siendo 1 totalmente transparente y 0 totalmente opaco.

```
In [33]: #Nos creamos el diccionario para asignarle un color a cada continente
         dict = {
             'Asia':'red',
             'Europe':'green',
             'Africa':'blue',
             'Americas':'yellow',
             'Oceania':'black'
         }

         #Obtenemos las columna cont de gp
         cont = gap_minder['cont']
         #Ahora reemplazamos cada continente por su color mapeando la columna cont con nuestro d
         cont_color = cont.replace(dict)
         #Nos creamos un array con el valor de la población para poder normalizar
         poblacion_np = np.array(poblacion)
         poblacion_np = poblacion_np / 1000000
         #Realizamos un gráfico de tipo scatter entre la renta per capita y la esperanza de vida
         plt.scatter(renta, esperanza_vida, s = poblacion_np, c = cont_color, alpha = 0.8)
         #Le añadimos nombre al ejex, ejey y le ponemos un título
```

```
plt.xlabel('Renta')
plt.ylabel('Esperanza de vida')
plt.title('Renta vs Esperanza de vida')
#Añadimos ticks a nuestro grafo
tick_val = [10000,20000, 30000, 40000, 50000]
tick_lab = ['10k', '20k', '30k', '40k', '50k']
plt.xticks(tick_val, tick_lab)
plt.show()
```



Podemos ver finalmente como este scatter, nos permite analizar nuestros datos de una forma más clara. Podemoss observar como la mayoría de los países africanos se encuentra en una zona donde tienen una renta pequeña además de una menor esperanza de vida, esto se comporte con países asiáticos donde se tiene una esperanza de vida inferior a los 70 años con una renta inferior a 10000 y además una gran densidad de población. En el lado opuesto tenemos los países europeos donde la gran mayoría tienen una población pequeño y además una elevada esperanda de vida y una renta elevada.

Finalmente Python también nos permite añadir texto a una determinada zona de nuestro grafo, para ello debemos conocer exactamente el luegar en el cual se encuentra un determinado punto, esto puede realizarse haciendo uso de plt.text().

```
In [36]: #Nos creamos el diccionario para asignarle un color a cada continente
         dict = {
             'Asia':'red',
             'Europe':'green',
             'Africa':'blue',
```

```
    'Americas':'yellow',
    'Oceania':'black'
}

#Obtenemos las columna cont de gp
cont = gap_minder['cont']
#Ahora reemplazamos cada continente por su color mapeando la columna cont con nuestro d
cont_color = cont.replace(dict)
#Nos creamos un array con el valor de la población para poder normalizar
poblacion_np = np.array(poblacion)
poblacion_np = poblacion_np / 1000000
#Realizamos un gráfico de tipo scatter entre la renta per capita y la esperanza de vida
plt.scatter(renta, esperanza_vida, s = poblacion_np, c = cont_color, alpha = 0.8)
#Le añadimos nombre al ejex, ejey y le ponemos un título
plt.xlabel('Renta')
plt.ylabel('Esperanza de vida')
plt.title('Renta vs Esperanza de vida')
#Añadimos ticks a nuestro grafo
tick_val = [10000,20000, 30000, 40000, 50000]
tick_lab = ['10k', '20k', '30k', '40k', '50k']
plt.xticks(tick_val, tick_lab)
#Añadimos los nombres a determinados países
plt.text(1550, 71, 'India')
plt.text(28800, 80, 'Spain')
plt.show()
```