AnalisisPoblacionFacEco

October 30, 2021

1 Práctica 2: Análisis de población vs factores de desarrollo

C03 : Visualización Científica y Narrativas

RAUGM 2021: Geociencias e inclusión

Luis Miguel de la Cruz Salas

Realizar un análisis complementario del FR (Fertility Rate) mundial comparando con otros factores de desarrollo.

Usaremos la misma información de la práctica 1, así como los scripts que se desarrollaron en ella para continuar con esta práctica 2.

1.1 Importar las bibliotecas

Incluimos las bibliotecas necesarias para la lectura de datos y para la visualización.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

En esta práctica usaremos un estilo de fondo obscuro con texto y gráficos claros. Para ello haremos lo siguiente:

1.2 Lectura de datos de FR

Usamos la misma información de la práctica 1.

[2]:
```
FR = pd.read_csv('../01/UNdata_Export_20211021_200853345.zip')

# Se agrupa por país
paises = FR.groupby('Country or Area')
```

[3]:
```
FR
```

[3]:
```
     Country or Area    Year(s) Variant  Value
0        Afghanistan  2015-2020  Medium  4.555
```

```
1         Afghanistan  2010-2015  Medium  5.447
2         Afghanistan  2005-2010  Medium  6.478
3         Afghanistan  2000-2005  Medium  7.182
4         Afghanistan  1995-2000  Medium  7.654
…                 …          …       …       …
4041          Zimbabwe  1970-1975  Medium  7.400
4042          Zimbabwe  1965-1970  Medium  7.400
4043          Zimbabwe  1960-1965  Medium  7.300
4044          Zimbabwe  1955-1960  Medium  7.000
4045          Zimbabwe  1950-1955  Medium  6.800

[4046 rows x 4 columns]
```

1.3 Funciones para graficación

Ponemos todos los scripts de la práctica 1, con algunas modificacines, en funciones como sigue:

[4]:
```python
from math import ceil

def maxminTicks(FR):
    """
    Calcula el máximo y el mínimo de todos los países y los yticks.

    Parameters
    ----------
    paises : DataFrameGroupBy
        Dataframe generado por GroupBy con la información de los países.

    Returns
    -------
    p_max, y_max, p_min, y_min, yticks
        El país con el máximo valor, el valor máximo, la lista para los yticks,
        el país con el valor mínimo y el valor mínimo.
    """
    # Se obtiene el valor máximo
    y_max = FR['Value'].max()

    # Extrae el nombre del país con el valor máximo
    p_max = FR[FR['Value'] == y_max].iloc[0][0]

    # Se obtiene el valor mínimo
    y_min = FR['Value'].min()

    # Extrae el nombre del país con el valor mínimo
    p_min = FR[FR['Value'] == y_min].iloc[0][0]

    # Se generan los yticks
```

```
    yticks = [i for i in range(0,ceil(y_max)+1)]

    return p_max, y_max, p_min, y_min, yticks
```

[5]:
```
def inicializaGrafica(y_maximo, yticks):
    """
    Inicializa algunos parámetros de la figura (el canvas).

    Parameters
    ----------
    y_maximo : int
        Valor máximo para el eje y.

    yticks : list
        Lista de valores para los ticks en el eje y.
    """
    fig = plt.figure(figsize=(10,10))
    plt.xticks(rotation=70, fontsize=10)
    plt.xlim(-2,14,-1)
    plt.gca().invert_xaxis()
    plt.ylim(0,y_maximo)
    plt.yticks(yticks)
    plt.grid(ls='--', lw=0.5)

    # Información adicional y títulos
    plt.title('Promedio de número de hijos por mujer', loc='left', fontsize=12)
    plt.title('fuente: http://data.un.org', loc='right', fontstyle='italic',␣
↪fontsize=10)
    plt.suptitle('Evolución del FR (Fertility Rate)', y = 0.94, fontsize=14)

    # Se eliminan las líneas del marco de la gráfica
    ejes = plt.gca()
    ejes.spines['right'].set_visible(False)
    ejes.spines['top'].set_visible(False)
    ejes.spines['left'].set_visible(False)
    ejes.spines['bottom'].set_visible(False)

    # Modificamos algunos parámetros de los ticks en el eje y
    ejes.tick_params(axis='y', width=1, length=25)

    # Realizamos algunas anotaciones sobre el gráfico base
    plt.annotate('Nivel de \n reemplazo: \n promedio = 2.1',
                 xy=(11.5, 2.095), xytext=(11.5, 1.0),
                 bbox=dict(boxstyle='round', facecolor='gray',␣
↪edgecolor='black', alpha=0.1, linewidth=0.75),
                 arrowprops=dict(arrowstyle='->', facecolor='black',␣
↪edgecolor='black'),
```

```
                fontsize=10, color='black', horizontalalignment='center')

    plt.text(2, 8.25, 'Cada línea representa la \n evolución del promedio  \n␣
↪de hijos por mujer en un país',
             transform=plt.gca().transData, horizontalalignment='center',␣
↪color='black',
             bbox=dict(boxstyle='round', facecolor='gray', edgecolor='black',␣
↪alpha=0.1, linewidth=0.75))
```

[6]:
```
def graficaFR(paises, parametros={}):
    """
    Realiza la gráfica de todos los países.

    Parameters
    ----------
    paises : DataFrameGroupBy
        Dataframe generado por GroupBy con la información de los países.

    parametros : dict
        Parámetros para generar la gráfica.
    """
    for p in paises.groups.keys():
        pais = paises.get_group(p)
        plt.plot(pais['Year(s)'], pais['Value'], **parametros)

    # Al final de todas las gráficas ponemos la del nivel de reemplazo
    plt.plot([-1,14],[2.1,2.1], 'k--', lw=1.0, zorder=1000)
```

[7]:
```
def graficaFR_Pais(paises, p, parametros={}, par_glow = None):
    """
    Realiza la gráfica de un solo país con realce para fondo negro.

    Parameters
    ----------
    paises : DataFrameGroupBy
        Dataframe generado por GroupBy con la información de los países.

    parametros : dict
        Parámetros para generar la gráfica.

    par_glow : dict
        Diccionario con los parámetros para resaltar las curvas con un "halo"
        a su alrededor. Cuando par_glow = None (defalt) no se hace nada. En otro
        caso es conveniente pasar la transparencia y el color. Esta
        curva se dibuja por detrás de la curva principal, con un ancho mayor y␣
↪el
        color definido
```

```
        Ejemplo:
                par_glow = {'alpha':0.4, 'c':'yellow'}
    """
    pais = paises.get_group(p)

    if par_glow:
    # Se grafica una curva con una línea 3 veces más ancha que la original
    # y transparente para resaltarla.
        plt.plot(pais['Year(s)'], pais['Value'], lw=parametros['lw']*3,␣
↪**par_glow)

    # Se grafica la curva del país con los parámetros necesarios.
    line = plt.plot(pais['Year(s)'], pais['Value'], **parametros)

    # Ponemos un texto al final de la curva para mostrar el
    # nombre del país y el valor final de fertilidad
    pais_val = pais['Value'].iloc[0]
    plt.text(x = 0, y = pais_val,
             s = ' {} {:1.2f}'.format(p, pais_val),
             c = line[0].get_color(), weight = 'bold')

    # Ponemos el valor inicial de fertilidad al principio de la curva.
    pais_val = pais['Value'].iloc[-1]
    plt.text(x = 13.75, y = pais_val,
             s = '{:1.2f} '.format(pais_val),
             c = line[0].get_color(), weight = 'bold')
```

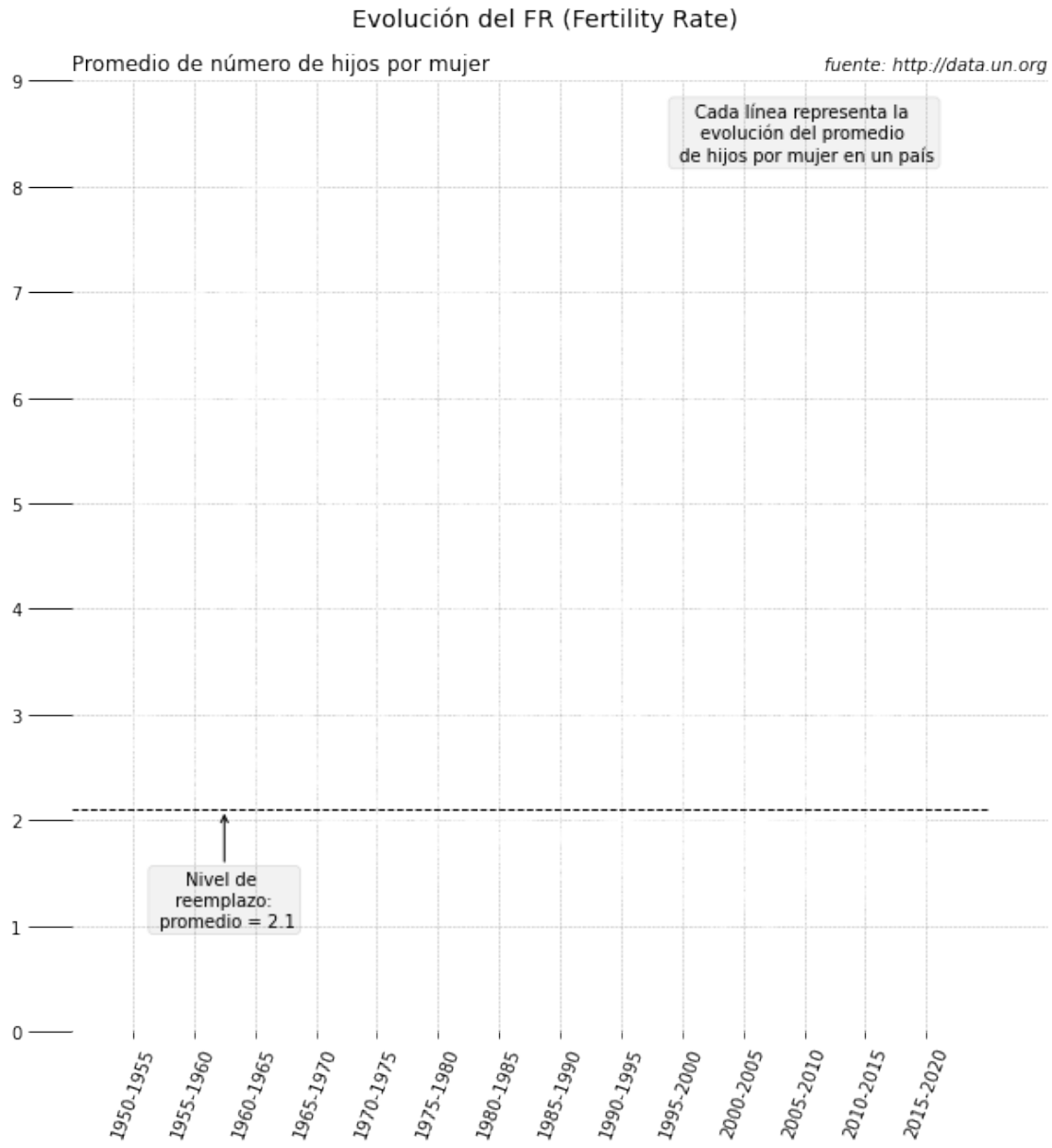
1.4 Visualización del *canvas*

Calculamos el máximo en el eje y y los `yticks` y posteriormente hacemos la gráfica base.

[8]:
```
p_max, y_max, p_min, y_min, yticks = maxminTicks(FR)
print('Máximo = {}, \t País : {}'.format(y_max, p_max))
print('Mínimo = {}, \t País : {}'.format(y_min, p_min))
print('yticks : {}'.format(yticks))
```

```
Máximo = 8.8,    País : Yemen
Mínimo = 0.85,   País : China, Macao SAR
yticks : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[9]:
```
# Hacemos la gráfica base
inicializaGrafica(y_max, yticks)
graficaFR(paises, {'lw':0.5, 'c':'#FFFFFF'})
```



Este es ahora nuestro lienzo listo para agregar información.

1.5 Selección de países con diferente GDP.

Definimos un conjunto de países a ser graficados y sus colores correspondientes. Esta elección se basa en el GDP (Gross Domestic Product), véase la siguiente figura:

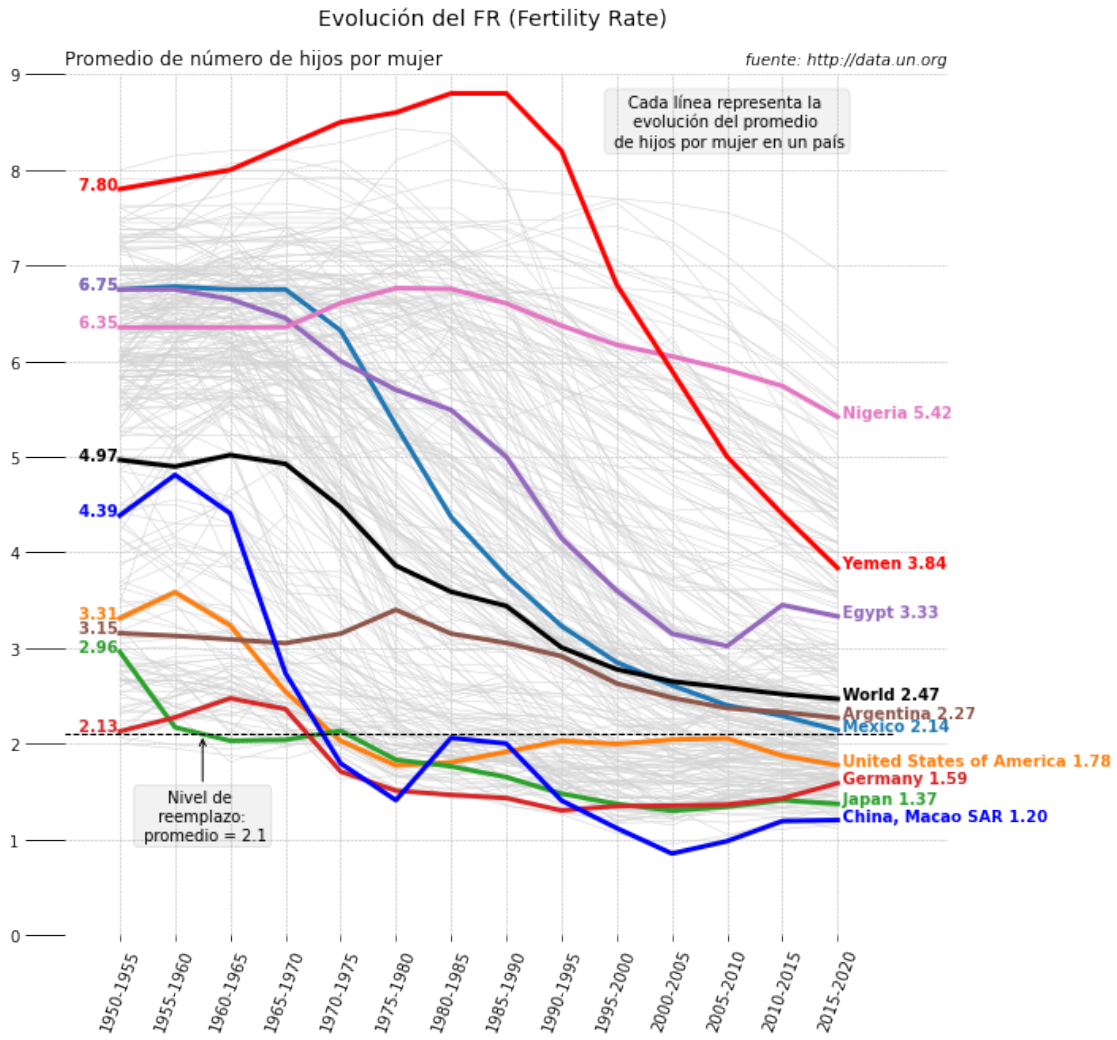
De acuerdo con la gráfica anterior, vamos a elegir países con diferente GDP y diferente zona geográfica, en este caso serán: México, USA, Japón, Alemania, Egipto, Argentina, Nigeria. Además agregamos a los países que hayan tenido el máximo y el mínimo de FR en el rango de tiempo estudiado.

[10]:
```
paises_colores = {
    'Mexico'                    : 'C0',
    'United States of America' : 'C1',
    'Japan'                     : 'C2',
    'Germany'                   : 'C3',
    'Egypt'                     : 'C4',
    'Argentina'                 : 'C5',
    'Nigeria'                   : 'C6',
    p_max                       : 'red',      # País con el valor máximo
    p_min                       : 'blue',     # País con el valor mínimo
    'World'                     : 'black'     # Datos promedio mundiales
}
```

1.5.1 Visualización de los países

Realizamos las gráficas para estos países.

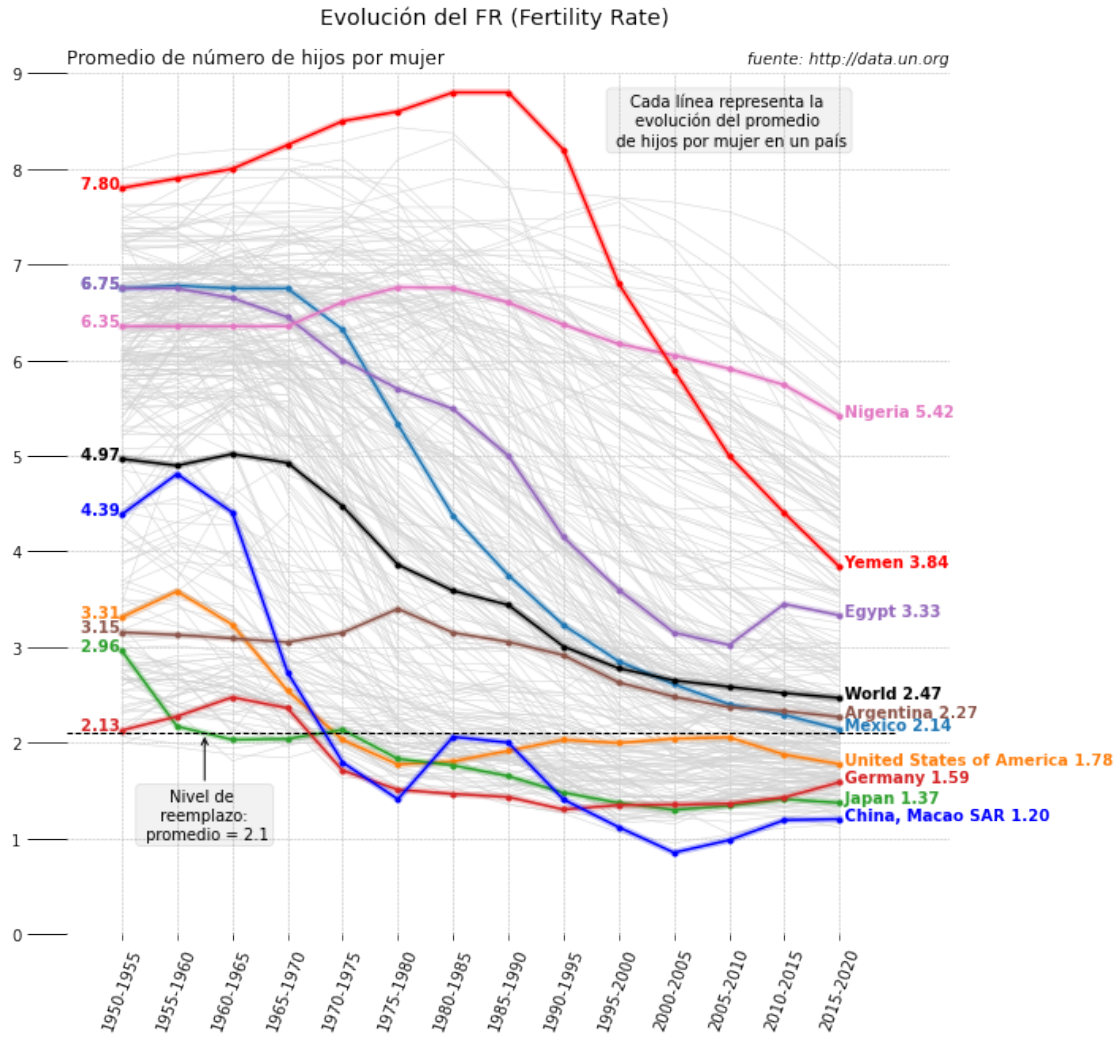
[11]:
```
# Hacemos la gráfica base
inicializaGrafica(y_max, yticks)
graficaFR(paises, {'lw':0.5, 'c':'lightgrey'})

# Hacemos la gráfica para los países definidos antes para
# hacer la comparación entre ellos.
for p, c in paises_colores.items():
    par = {'lw':3.0, 'c':c}
    graficaFR_Pais(paises, p, par)

plt.savefig('FR01.pdf')
```



Si puso atención, en la definición de la función `graficaFR_Pais()` hay un parámetro para resaltar las gráficas. Usaremos ese parámetro como sigue:

[12]:
```
# Hacemos la gráfica base
inicializaGrafica(y_max, yticks)
graficaFR(paises, {'lw':0.5, 'c':'lightgrey'})

# Hacemos la gráfica para los países definidos antes para hacer la comparación␣
↪entre ellos.
for p, c in paises_colores.items():
    par = {'lw':1.5, 'c':c, 'marker':'.'}
    par_glow = {'alpha':0.25, 'c':c}          # Parámetros para poner "brillo" a␣
↪las curvas
    graficaFR_Pais(paises, p, par, par_glow) # Pasamos los parámetros a la␣
↪función
```

```
plt.savefig('FR02.pdf')
```



En esta visualización intentamos graficar países de los diferentes continentes y con un GDP (Gross Domestic Product, en miles de millones de dolares) diferente: USA (20,544), Japon (4,971), Alemania (3,948), Mexico (1,221), Argentina (519), Egipto (250), Macao SAR (55), Yemen (27), Nigeria (9). (Información tomada de https://data.worldbank.org/country). Observamos que no necesariamente entre menos GDP mayor fertilidad, véase por ejemplo el caso de Macao SAR. Sin embargo, en general si se observa que los países con mayor GDP tienden a bajar el número de hijos por mujer por debajo del NR. Los tres países que en esta visualización se ven com mayor fertilidad (Egipto, Yemen y Nigeria) tienen un GPD bajo, comparado con los otros países, pero todos han reducido su porcentaje de fertilidad. Nigeria es un caso particular, pues solo ha bajado de 6.35 en 1955 a 5.42 en 2020.

1.6 Datos de GDP por país

Vamos ahora a complementar esta visualización con información del GPD per capita en cada país.

Obtenemos la información de los ingresos por persona en cada país de http://data.un.org haciendo la búsqueda de *per capita income* como se muestra en la siguiente figura:

Observación : los datos son solo para el año 2010, que es el año en el que vamos a realizar la comparación. Por esta razón tomaremos los datos de la década 2005-2010 para el caso del FR (esta información se encuentra en el lugar 2 del arreglo de cada país, véanse las gráficas anteriores y recuérdese que el eje x fue invertido).

[13]:
```
# Leemos el archivo que obtiene en un DataFrame
per_capita_income = pd.read_csv('UNdata_Export_20211023_002155840.zip')
pd.set_option('display.max_rows', None) # Para poder ver todo el dataframe
per_capita_income
```

[13]:

	Country or Area	Year(s)	Value	\
0	Afghanistan	2010	1300	
1	Albania	2010	8480	
2	Algeria	2010	7970	
3	Angola	2010	5000	
4	Antigua and Barbuda	2010	18840	
5	Armenia	2010	7510	
6	Australia	2010	37580	
7	Austria	2010	40260	
8	Azerbaijan	2010	9160	
9	Bahamas	2010	28660	
10	Bahrain	2010	18910	
11	Bangladesh	2010	1760	
12	Barbados	2010	24950	
13	Belarus	2010	13470	
14	Belgium	2010	38450	
15	Belize	2010	7090	
16	Benin	2010	1440	
17	Bhutan	2010	5230	
18	Bolivia (Plurinational State of)	2010	4470	
19	Bosnia and Herzegovina	2010	8850	
20	Botswana	2010	13710	
21	Brazil	2010	10890	
22	Bulgaria	2010	13700	
23	Burkina Faso	2010	1320	
24	Burundi	2010	520	
25	Cabo Verde	2010	4230	
26	Cambodia	2010	2030	
27	Cameroon	2010	2130	
28	Canada	2010	38400	
29	Central African Republic	2010	960	
30	Chad	2010	1480	

31	Chile	2010	17360
32	China	2010	7470
33	Colombia	2010	8910
34	Comoros	2010	1160
35	Congo	2010	3100
36	Costa Rica	2010	11270
37	Côte d'Ivoire	2010	1850
38	Croatia	2010	18070
39	Cyprus	2010	30300
40	Czech Republic	2010	23400
41	Democratic Republic of the Congo	2010	340
42	Denmark	2010	41540
43	Dominica	2010	12070
44	Dominican Republic	2010	8850
45	Ecuador	2010	8330
46	Egypt	2010	6300
47	El Salvador	2010	6380
48	Equatorial Guinea	2010	15600
49	Eritrea	2010	490
50	Estonia	2010	19040
51	Ethiopia	2010	960
52	Fiji	2010	4440
53	Finland	2010	36550
54	France	2010	34970
55	Gabon	2010	12460
56	Gambia	2010	1880
57	Georgia	2010	4910
58	Germany	2010	38450
59	Ghana	2010	1610
60	Greece	2010	26770
61	Grenada	2010	9880
62	Guatemala	2010	4600
63	Guinea	2010	900
64	Guinea-Bissau	2010	1140
65	Guyana	2010	2970
66	Haiti	2010	1110
67	Honduras	2010	3730
68	Hungary	2010	19720
69	Iceland	2010	29270
70	India	2010	3400
71	Indonesia	2010	4140
72	Iraq	2010	3640
73	Ireland	2010	34830
74	Israel	2010	26270
75	Italy	2010	31930
76	Japan	2010	34570
77	Jordan	2010	5750

78	Kazakhstan	2010	10440
79	Kenya	2010	1610
80	Kiribati	2010	3470
81	Kuwait	2010	47770
82	Kyrgyzstan	2010	2070
83	Lao People's Democratic Republic	2010	2310
84	Latvia	2010	16830
85	Lebanon	2010	13340
86	Lesotho	2010	2100
87	Liberia	2010	470
88	Lithuania	2010	18350
89	Luxembourg	2010	57690
90	Madagascar	2010	930
91	Malawi	2010	710
92	Malaysia	2010	14680
93	Maldives	2010	6900
94	Mali	2010	1150
95	Malta	2010	24390
96	Mauritania	2010	2280
97	Mauritius	2010	13670
98	Mexico	2010	14600
99	Micronesia (Federated States of)	2010	3630
100	Mongolia	2010	3680
101	Montenegro	2010	12820
102	Morocco	2010	4580
103	Mozambique	2010	880
104	Namibia	2010	6400
105	Nepal	2010	1340
106	Netherlands	2010	40900
107	New Zealand	2010	29020
108	Nicaragua	2010	3480
109	Niger	2010	700
110	Nigeria	2010	2180
111	Norway	2010	58130
112	Oman	2010	25320
113	Pakistan	2010	2650
114	Palau	2010	14640
115	Panama	2010	13930
116	Papua New Guinea	2010	2380
117	Paraguay	2010	5430
118	Peru	2010	8690
119	Philippines	2010	3920
120	Poland	2010	19220
121	Portugal	2010	24670
122	Qatar	2010	77640
123	Republic of Korea	2010	28650
124	Republic of Moldova	2010	3330

125	Romania	2010	15260
126	Russian Federation	2010	20110
127	Rwanda	2010	1150
128	Saint Kitts and Nevis	2010	15630
129	Saint Lucia	2010	10680
130	Saint Vincent and the Grenadines	2010	10130
131	Samoa	2010	4080
132	Sao Tome and Principe	2010	1700
133	Saudi Arabia	2010	27720
134	Senegal	2010	1830
135	Serbia	2010	10830
136	Seychelles	2010	21600
137	Sierra Leone	2010	1080
138	Singapore	2010	57280
139	Slovakia	2010	21770
140	Slovenia	2010	26120
141	Solomon Islands	2010	2230
142	South Africa	2010	10260
143	Spain	2010	31070
144	Sri Lanka	2010	5000
145	Sudan	2010	1800
146	Suriname	2010	7760
147	Swaziland	2010	4880
148	Sweden	2010	40130
149	Switzerland	2010	51760
150	Syrian Arab Republic	2010	4800
151	Tajikistan	2010	1890
152	Thailand	2010	8410
153	The former Yugoslav Republic of Macedonia	2010	11220
154	Timor-Leste	2010	5310
155	Togo	2010	830
156	Tonga	2010	4620
157	Trinidad and Tobago	2010	24440
158	Tunisia	2010	8870
159	Turkey	2010	15810
160	Turkmenistan	2010	7390
161	Uganda	2010	1140
162	Ukraine	2010	6540
163	United Arab Emirates	2010	41550
164	United Kingdom	2010	36020
165	United Republic of Tanzania	2010	1400
166	United States of America	2010	48880
167	Uruguay	2010	13290
168	Uzbekistan	2010	3120
169	Vanuatu	2010	4270
170	Venezuela (Bolivarian Republic of)	2010	11790
171	Viet Nam	2010	3210

```
172                                  Yemen     2010   2590
173                                 Zambia     2010   1380

     Value Footnotes
0                NaN
1                NaN
2                NaN
3                NaN
4                NaN
5                NaN
6                NaN
7                NaN
8                NaN
9                NaN
10               NaN
11               NaN
12               NaN
13               NaN
14               NaN
15               NaN
16               NaN
17               NaN
18               NaN
19               NaN
20               NaN
21               NaN
22               NaN
23               NaN
24               NaN
25               NaN
26               NaN
27               NaN
28               NaN
29               NaN
30               NaN
31               NaN
32               NaN
33               NaN
34               NaN
35               NaN
36               NaN
37               NaN
38               NaN
39               NaN
40               NaN
41               NaN
42               NaN
```

```
43              NaN
44              NaN
45              NaN
46              NaN
47              NaN
48              NaN
49              NaN
50              NaN
51              NaN
52              NaN
53              NaN
54              NaN
55              NaN
56              NaN
57              NaN
58              NaN
59              NaN
60              NaN
61              NaN
62              NaN
63              NaN
64              NaN
65              NaN
66              NaN
67              NaN
68              NaN
69              NaN
70              NaN
71              NaN
72              NaN
73              NaN
74              NaN
75              NaN
76              NaN
77              NaN
78              NaN
79              NaN
80              NaN
81              NaN
82              NaN
83              NaN
84              NaN
85              NaN
86              NaN
87              NaN
88              NaN
89              NaN
```

90	NaN
91	NaN
92	NaN
93	NaN
94	NaN
95	NaN
96	NaN
97	NaN
98	NaN
99	NaN
100	NaN
101	NaN
102	NaN
103	NaN
104	NaN
105	NaN
106	NaN
107	NaN
108	NaN
109	NaN
110	NaN
111	NaN
112	NaN
113	NaN
114	NaN
115	NaN
116	NaN
117	NaN
118	NaN
119	NaN
120	NaN
121	NaN
122	NaN
123	NaN
124	NaN
125	NaN
126	NaN
127	NaN
128	NaN
129	NaN
130	NaN
131	NaN
132	NaN
133	NaN
134	NaN
135	NaN
136	NaN

```
137             NaN
138             NaN
139             NaN
140             NaN
141             NaN
142             NaN
143             NaN
144             NaN
145             NaN
146             NaN
147             NaN
148             NaN
149             NaN
150             NaN
151             NaN
152             NaN
153             NaN
154             NaN
155             NaN
156             NaN
157             NaN
158             NaN
159             NaN
160             NaN
161             NaN
162             NaN
163             NaN
164             NaN
165             NaN
166             NaN
167             NaN
168             NaN
169             NaN
170             NaN
171             NaN
172             NaN
173             NaN
```

[14]:
```
pd.set_option('display.max_rows', 15) # Regreso a un número limitado de␣
↪renglones por despliegue de dataframe
```

1.6.1 Tratamiento de los datos.

Observamos que el número de países listados en los dos dataframes, el de fertilidad y el de ingresos, es diferente, así que necesitamos ajustarlos para poder hacer una comparación entre ellos.

[15]:
```
# Se agrupa por país la información de los ingresos
# (para acceder más fácil y de manera similar a como se
# hace con la información del FR)
ingreso_pais = per_capita_income.groupby('Country or Area')
```

[16]:
```
# Checamos la longitud de cada agrupación para ver si coinciden
print(len(paises.groups.keys()), len(per_capita_income))
```

```
286 174
```

[17]:
```
# Se obtiene la lista de países a comparar:
lista_paises_final = []

# Hacemos el recorrido usando el DataFrame de mayor longitud (paises)
for p in paises.groups.keys():
    try:                              # Aseguramos que se tiene la misma info
        ingreso_pais.get_group(p)     # en cada DataFrame.
        lista_paises_final.append(p) # Si se tuvo éxito entonces se agrega el␣
 ↪país a la lista final.
    except KeyError: # Captura de la excepción de tipo KeyError
        continue     # Lo único que hacemos es saltarnos el país que no está
                     # en ambos DataFrames y continuar con el siguiente.

print('Lista final de países en ambos Dataframes : ', len(lista_paises_final),␣
 ↪'\n')
print(lista_paises_final)
```

```
Lista final de países en ambos Dataframes :  169

['Afghanistan', 'Albania', 'Algeria', 'Angola', 'Antigua and Barbuda',
'Armenia', 'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
'Bolivia (Plurinational State of)', 'Bosnia and Herzegovina', 'Botswana',
'Brazil', 'Bulgaria', 'Burkina Faso', 'Burundi', 'Cabo Verde', 'Cambodia',
'Cameroon', 'Canada', 'Central African Republic', 'Chad', 'Chile', 'China',
'Colombia', 'Comoros', 'Congo', 'Costa Rica', 'Croatia', 'Cyprus', 'Democratic
Republic of the Congo', 'Denmark', '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', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati',
'Kuwait', 'Kyrgyzstan', "Lao People's Democratic Republic", 'Latvia', 'Lebanon',
'Lesotho', 'Liberia', 'Lithuania', 'Luxembourg', 'Madagascar', 'Malawi',
'Malaysia', 'Maldives', 'Mali', 'Malta', 'Mauritania', 'Mauritius', 'Mexico',
'Mongolia', 'Montenegro', 'Morocco', 'Mozambique', 'Namibia', 'Nepal',
'Netherlands', 'New Zealand', 'Nicaragua', 'Niger', 'Nigeria', 'Norway', 'Oman',
```

```
'Pakistan', 'Palau', 'Panama', 'Papua New Guinea', 'Paraguay', 'Peru',
'Philippines', 'Poland', 'Portugal', 'Qatar', 'Republic of Korea', 'Republic of
Moldova', 'Romania', 'Russian Federation', 'Rwanda', 'Saint Kitts and Nevis',
'Saint Lucia', 'Saint Vincent and the Grenadines', 'Samoa', 'Sao Tome and
Principe', 'Saudi Arabia', 'Senegal', 'Serbia', 'Seychelles', 'Sierra Leone',
'Singapore', 'Slovakia', 'Slovenia', 'Solomon Islands', 'South Africa', 'Spain',
'Sri Lanka', 'Sudan', 'Suriname', 'Sweden', 'Switzerland', 'Syrian Arab
Republic', 'Tajikistan', 'Thailand', 'Timor-Leste', 'Togo', 'Tonga', 'Trinidad
and Tobago', 'Tunisia', 'Turkey', 'Turkmenistan', 'Uganda', 'Ukraine', 'United
Arab Emirates', 'United Kingdom', 'United Republic of Tanzania', 'United States
of America', 'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela (Bolivarian Republic
of)', 'Viet Nam', 'Yemen', 'Zambia']
```

[18]:
```
# Revisamos la información del FR uno de los países
# para identificar en qué lugar está la info de 2010
paises.get_group('Mexico')
```

[18]:

	Country or Area	Year(s)	Variant	Value
2254	Mexico	2015-2020	Medium	2.14
2255	Mexico	2010-2015	Medium	2.29
2256	Mexico	2005-2010	Medium	2.40
2257	Mexico	2000-2005	Medium	2.61
2258	Mexico	1995-2000	Medium	2.85
2259	Mexico	1990-1995	Medium	3.23
2260	Mexico	1985-1990	Medium	3.75
2261	Mexico	1980-1985	Medium	4.37
2262	Mexico	1975-1980	Medium	5.33
2263	Mexico	1970-1975	Medium	6.32
2264	Mexico	1965-1970	Medium	6.75
2265	Mexico	1960-1965	Medium	6.75
2266	Mexico	1955-1960	Medium	6.78
2267	Mexico	1950-1955	Medium	6.75

[19]:
```
ingreso_pais.get_group('Mexico')
```

[19]:

	Country or Area	Year(s)	Value	Value Footnotes
98	Mexico	2010	14600	NaN

[20]:
```
# Ahora creamos dos listas, una para el FR y otra para el PCI (Per Capita␣
↪Income).
l_FR = []
l_PCI = []
for p in lista_paises_final:
    l_FR.append(paises.get_group(p)['Value'].iloc[2])  # El lugar 2 corresponde␣
↪a los datos para 2010
    l_PCI.append(ingreso_pais.get_group(p)['Value'].iloc[0]) # Solo se tiene el␣
↪dato para 2010
```

```
print('\nFR (tamaño: {}) \nDatos :\n{}'.format(len(l_FR), l_FR))
print('\nPCI (tamaño: {}) \nDatos :\n{}'.format(len(l_PCI), l_PCI))
```

```
FR (tamaño: 169)
Datos :
[6.478, 1.64, 2.724, 6.35, 2.0, 1.72, 1.952, 1.399, 1.83, 1.91, 2.25, 2.482,
1.75, 1.442, 1.821, 2.84, 5.495, 2.55, 3.4, 1.312, 3.035, 1.86, 1.519, 6.079,
6.39, 2.714, 3.081, 5.25, 1.636, 5.3, 6.853, 1.898, 1.62, 2.1, 4.9, 4.8, 1.94,
1.522, 1.48, 6.633, 1.854, nan, 2.57, 2.69, 3.02, 2.395, 5.401, 4.8, 1.662,
5.45, 2.75, 1.84, 1.977, 4.2, 5.65, 1.8, 1.362, 4.365, 1.416, 2.302, 3.621,
5.535, 5.2, 2.726, 3.62, 3.244, 1.329, 2.131, 2.796, 2.5, 4.399, 2.003, 2.927,
1.436, 1.339, 3.8, 2.541, 4.65, 3.88, 2.4, 2.78, 3.4, 1.493, 1.9, 3.373, 5.2,
1.417, 1.621, 4.83, 5.73, 2.216, 2.288, 6.7, 1.393, 5.07, 1.698, 2.4, 2.397,
1.82, 2.53, 5.537, 3.608, 2.813, 1.746, 2.142, 2.68, 7.55, 5.91, 1.924, 2.9,
4.168, nan, 2.632, 4.133, 2.892, 2.68, 3.3, 1.374, 1.372, 2.23, 1.173, 1.27,
1.51, 1.456, 4.85, nan, 1.6, 2.132, 4.468, 4.85, 3.225, 5.1, 1.583, 2.3, 5.565,
1.257, 1.317, 1.435, 4.4, 2.625, 1.454, 2.279, 5.0, 2.667, 1.891, 1.469, 3.7,
3.609, 1.556, 5.3, 5.036, 4.034, 1.8, 2.024, 2.202, 2.65, 6.38, 1.383, 1.97,
1.862, 5.579, 2.055, 2.03, 2.49, 4.2, 2.547, 1.928, 5.0, 5.6]

PCI (tamaño: 169)
Datos :
[1300, 8480, 7970, 5000, 18840, 7510, 37580, 40260, 9160, 28660, 18910, 1760,
24950, 13470, 38450, 7090, 1440, 5230, 4470, 8850, 13710, 10890, 13700, 1320,
520, 4230, 2030, 2130, 38400, 960, 1480, 17360, 7470, 8910, 1160, 3100, 11270,
18070, 30300, 340, 41540, 12070, 8850, 8330, 6300, 6380, 15600, 490, 19040, 960,
4440, 36550, 34970, 12460, 1880, 4910, 38450, 1610, 26770, 9880, 4600, 900,
1140, 2970, 1110, 3730, 19720, 29270, 3400, 4140, 3640, 34830, 26270, 31930,
34570, 5750, 10440, 1610, 3470, 47770, 2070, 2310, 16830, 13340, 2100, 470,
18350, 57690, 930, 710, 14680, 6900, 1150, 24390, 2280, 13670, 14600, 3680,
12820, 4580, 880, 6400, 1340, 40900, 29020, 3480, 700, 2180, 58130, 25320, 2650,
14640, 13930, 2380, 5430, 8690, 3920, 19220, 24670, 77640, 28650, 3330, 15260,
20110, 1150, 15630, 10680, 10130, 4080, 1700, 27720, 1830, 10830, 21600, 1080,
57280, 21770, 26120, 2230, 10260, 31070, 5000, 1800, 7760, 40130, 51760, 4800,
1890, 8410, 5310, 830, 4620, 24440, 8870, 15810, 7390, 1140, 6540, 41550, 36020,
1400, 48880, 13290, 3120, 4270, 11790, 3210, 2590, 1380]
```

Obsérvese que se tienen `nan` en algunos lugares de la lista de FR, lo que significa que no se tiene información completa para algunos países. Tomaremos esto en cuenta más adelante.

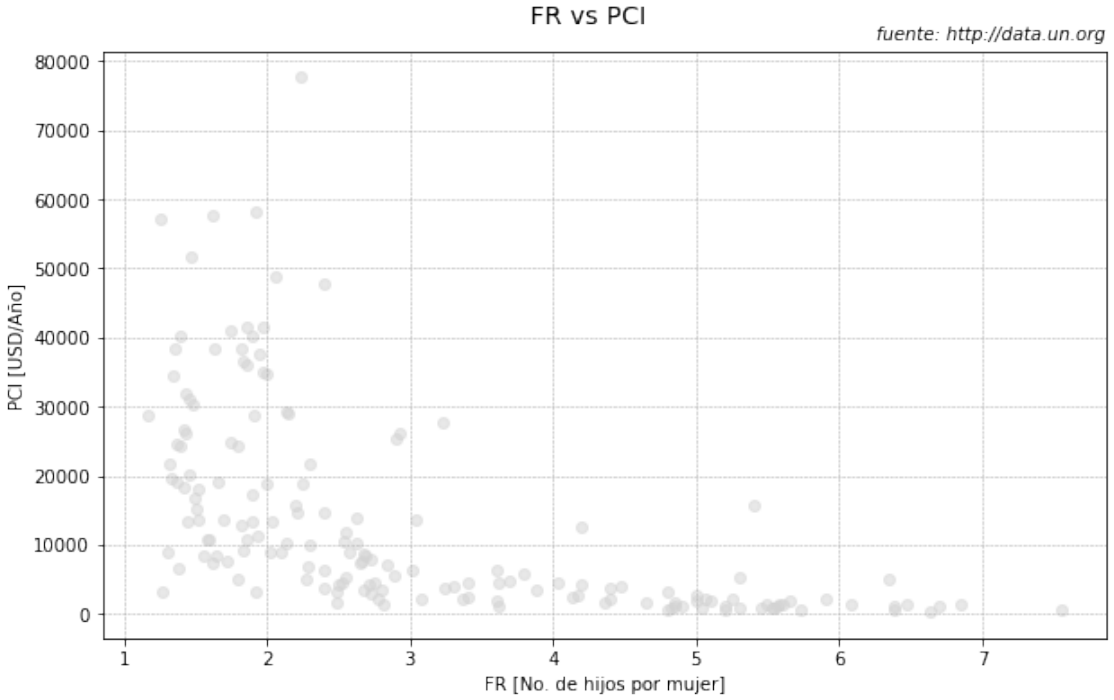
1.6.2 Visualización FR vs PCI

Ahora haremos la gráfica de esta información usando puntos en el plano.

[21]:
```
fig = plt.figure(figsize=(10,6)) # Cambiamos el tamaño de la figura
```

```
# Graficamos el PCI (eje $y$) en función de la FR (eje $x$)
# Usamos un color gris tenue y con transparencia. En este caso, este
# será nuestro lienzo base para lo demás.
plt.scatter(l_FR, l_PCI, marker='o', color='lightgray', alpha=0.5)

# Información adicional y títulos
plt.title('fuente: http://data.un.org', loc='right', fontstyle='italic',␣
 ↪fontsize=10)
plt.suptitle(' FR vs PCI ', y = 0.94, fontsize=14)

plt.grid(ls='--', lw=0.5, c='#AAAAAA')
plt.xlabel('FR [No. de hijos por mujer]')
plt.ylabel('PCI [USD/Año]')
```

[21]:
```
Text(0, 0.5, 'PCI [USD/Año]')
```



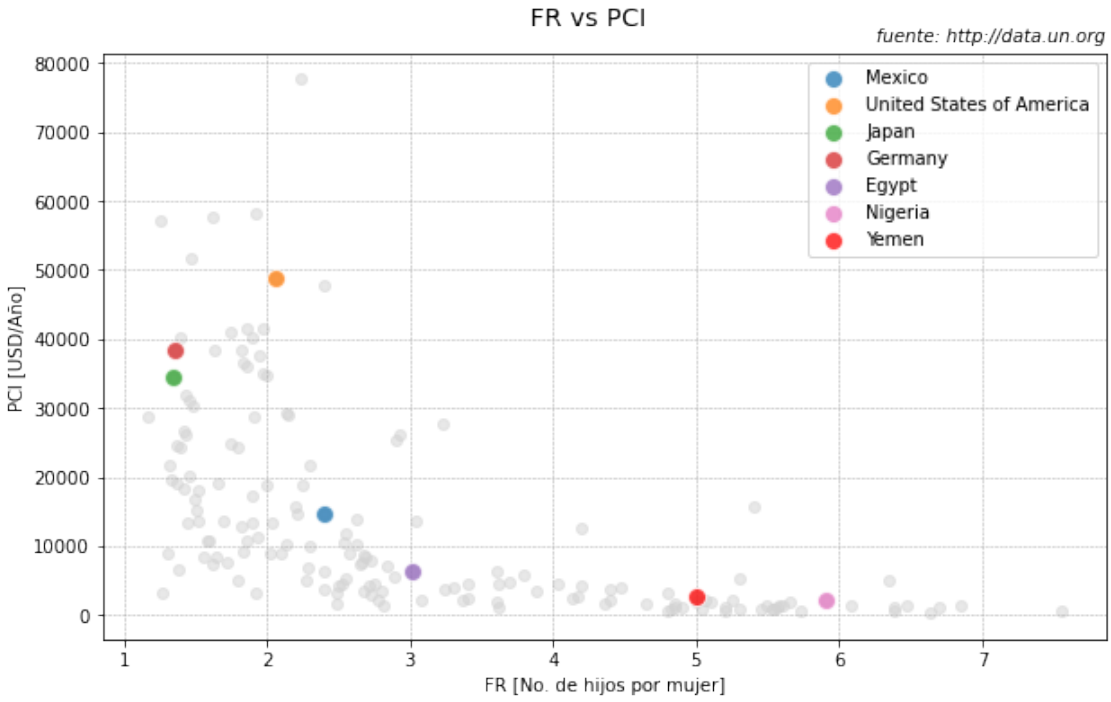
Observamos en la visualización anterior que los puntos se aglomeran en ciertas zonas del espacio. Se puede ver un comportamiento decreciente conforme aumenta el FR.

Vamos a completar la visualización anterior, resaltando los países que elegimos en el paso 5 (probablemente algunos no aparezcan debido a que ajustamos la información).

[22]:
```
fig = plt.figure(figsize=(10,6)) # Cambiamos el tamaño de la figura

# Graficamos el ingreso (eje $y$) en función de la fertilidad (eje $x$)
```

```
# Usamos un color gris tenue y con transparencia. En este caso, esta
# será nuestro lienzo base para lo demás.
plt.scatter(l_FR, l_PCI, marker='o', color='lightgray', alpha=0.5)

# Información adicional y títulos
plt.title('fuente: http://data.un.org', loc='right', fontstyle='italic',␣
 ↪fontsize=10)
plt.suptitle(' FR vs PCI ', y = 0.94, fontsize=14)

plt.grid(ls='--', lw=0.5, c='#AAAAAA')
plt.xlabel('FR [No. de hijos por mujer]')
plt.ylabel('PCI [USD/Año]')

# Gráfica de los países elegidos en el paso 5 usando puntos de colores y␣
 ↪resaltados.
for p, c in paises_colores.items():
    try:
        ip = lista_paises_final.index(p)
        plt.scatter(l_FR[ip], l_PCI[ip],
                    marker='o', s=100,
                    facecolor=c, edgecolor='w', alpha=0.75,
                    zorder=1000, label=p)
    except ValueError:
        continue

plt.legend()
```

[22]: <matplotlib.legend.Legend at 0x7fe81e7b1640>



1.6.3 Ajuste de los datos.

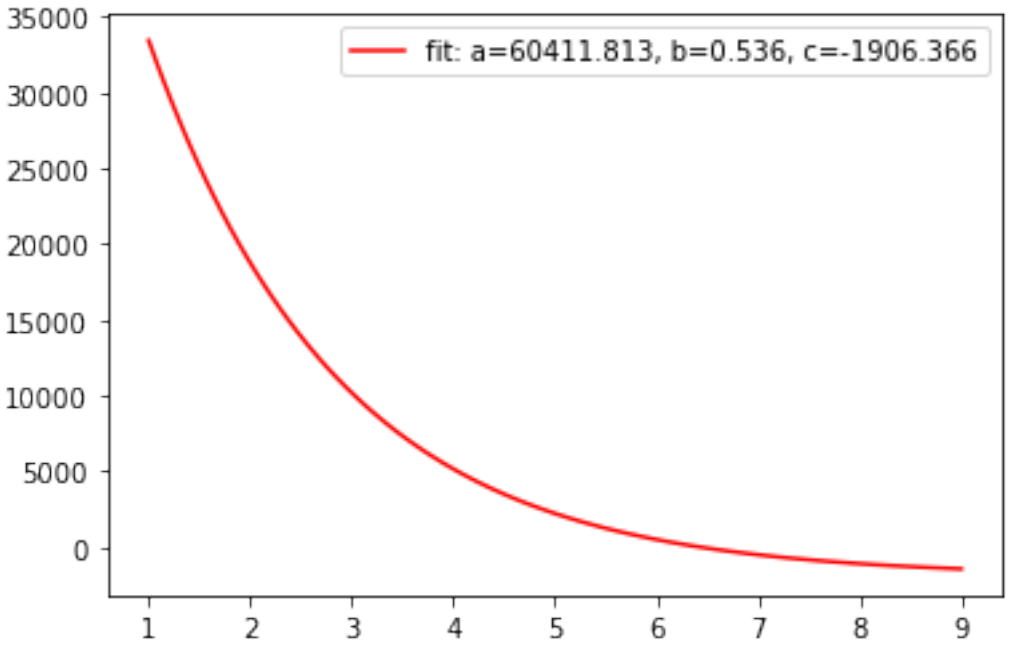
Dada la tendencia que se observa en esta gráfica, vamos a intentar ajustar una curva a los datos.

[23]:
```
# Usaremos en este caso la función curve_fit.
from scipy.optimize import curve_fit

# La función que queremos ajustar: exponencial decreciente.
def func(x, a, b, c):
    return a * np.exp(-b * x) + c

# Vamos a limpiar los datos, pues existen algunos 'nan' en ellos,
# particularmente en la FR.
xdata = []
ydata = []
for lf, li in zip(l_FR, l_PCI):
    if ~np.isnan(lf):
        xdata.append(lf)
        ydata.append(li)

# Con las listas xdata y ydata ya podemos hacer el ajuste
popt, pcov = curve_fit(func, xdata, ydata)
perr = np.sqrt(np.diag(pcov))
```

```
<ipython-input-23-d8950f1d24e3>:6: RuntimeWarning: overflow encountered in exp
  return a * np.exp(-b * x) + c
<ipython-input-23-d8950f1d24e3>:6: RuntimeWarning: overflow encountered in
multiply
  return a * np.exp(-b * x) + c
```

[24]:
```
# Graficamos el ajuste
x = np.linspace(1,9,100)
y = func(x, *popt)
plt.plot(x, y, 'r-',label='fit: a={:5.3f}, b={:5.3}, c={:5.3f}'.format(popt[0],␣
 ↪popt[1], popt[2]))
plt.legend()
```

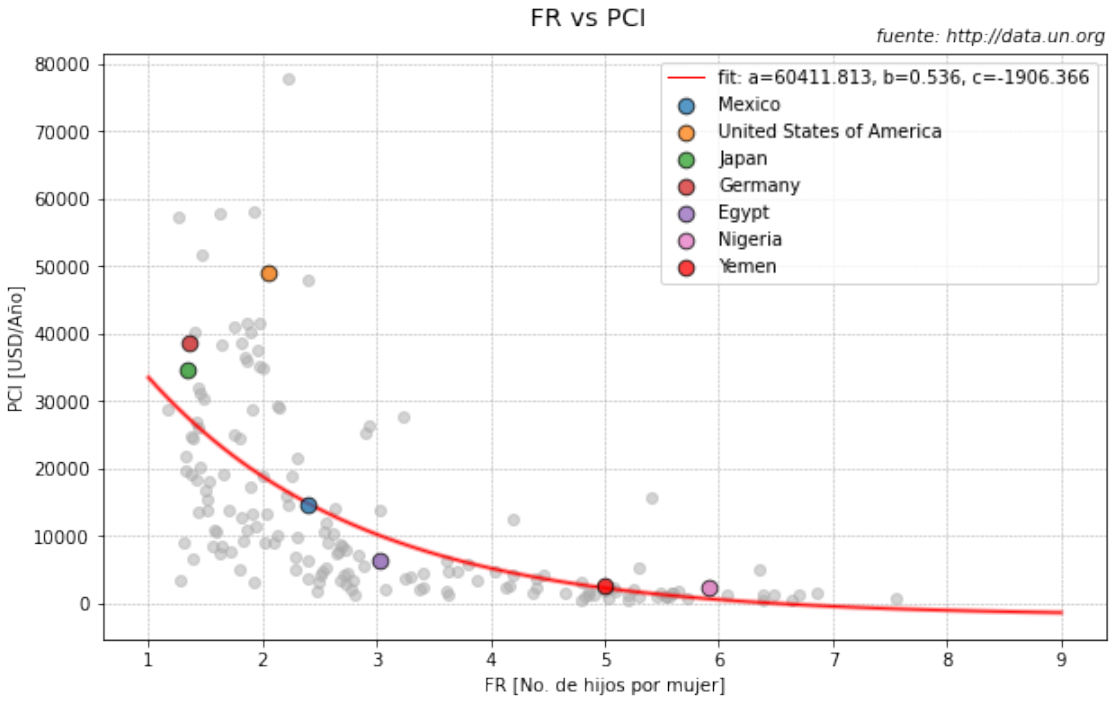
[24]:
```
<matplotlib.legend.Legend at 0x7fe81eefeca0>
```



Finalmente graficamos todo junto:

[25]:
```
fig = plt.figure(figsize=(10,6)) # Cambiamos el tamaño de la figura

# Graficamos el ingreso (eje $y$) en función de la fertilidad (eje $x$)
# Usamos un color gris tenue y con transparencia. En este caso, esta
# será nuestro lienzo base para lo demás.
plt.scatter(l_FR, l_PCI, marker='o', color='#ABABAB', alpha=0.5)

# Información adicional y títulos
```

```
plt.title('fuente: http://data.un.org', loc='right', fontstyle='italic',␣
 ↪fontsize=10)
plt.suptitle(' FR vs PCI ', y = 0.94, fontsize=14)

plt.grid(ls='--', lw=0.5, c='#AAAAAA')
plt.xlabel('FR [No. de hijos por mujer]')
plt.ylabel('PCI [USD/Año]')

# Gráfica de los países elegido en el paso 5 usando puntos de colores y␣
 ↪resaltados.
for p, c in paises_colores.items():
    try:
        ip = lista_paises_final.index(p)
        plt.scatter(l_FR[ip], l_PCI[ip],
                    marker='o', s=75,
                    facecolor=c, edgecolor='black', alpha=0.75,
                    zorder=1000, label=p)
    except ValueError:
        continue

# Gráfica del ajuste resaltada!
plt.plot(x, y, 'r-', lw=3.0, alpha=0.4)
plt.plot(x, y, 'r-', lw=1.0, label='fit: a={:5.3f}, b={:5.3}, c={:5.3f}'.
 ↪format(popt[0], popt[1], popt[2]))

plt.legend()
plt.savefig('FR03_ajuste.pdf')
```



¿Que podría mejorar de esta visualización? (colores, dimensiones, anotaciones, leyendas, ajuste, etc.)

1.7 Datos del nivel en educación.

Vamos ahora realizar una visualización similar pero con información del nivel de educación en cada país.

Obtenemos la información de http://data.un.org haciendo la búsqueda que se muestra en la siguiente figura:

Observación : igual que en el caso del ingreso, los datos son solo para el año 2010, que es el año en el que vamos a realizar la comparación.

[26]:
```
# Leemos el archivo generado
educacion = pd.read_csv('UNdata_Export_20211023_005512887.zip')
educacion
```

[26]:

	Reference Area	Time Period	Sex	Age group	\
0	Albania	2010	Female	Not applicable	
1	Albania	2010	Male	Not applicable	
2	Albania	2010	All genders	Not applicable	
3	Algeria	2010	Female	Not applicable	
4	Algeria	2010	All genders	Not applicable	
..	…	…	…	…	
454	Yemen	2010	All genders	Not applicable	

```
455          Yemen         2010       Female  Not applicable
456       Zimbabwe         2010  All genders  Not applicable
457       Zimbabwe         2010         Male  Not applicable
458       Zimbabwe         2010       Female  Not applicable

    Units of measurement  Observation Value
0                Percent           52.38337
1                Percent           37.49551
2                Percent           44.54065
3                Percent           35.33614
4                Percent           29.83456
..                   …                  …
454              Percent           10.57348
455              Percent            6.40428
456              Percent            5.90560
457              Percent            6.60183
458              Percent            5.23100

[459 rows x 6 columns]
```

Observamos que la información viene para 'Female', 'Male' y 'All genders'.

[27]:
```
# Organizamos la información por país y por sexo:
educacion_pais = educacion.groupby(['Reference Area', 'Sex'])

# Podemos por ejemplo revisar la información para México:
gender = 'Female' # Este parámetro permite elegir el sexo para la comparación.
educacion_pais.get_group(('Mexico',gender))
```

[27]:
```
    Reference Area  Time Period     Sex       Age group Units of measurement  \
289         Mexico         2010  Female  Not applicable              Percent

     Observation Value
289           26.61178
```

1.7.1 Tratamiento de los datos

Igual que antes, debemos asegurarnos que tenemos el mismo número de datos.

[28]:
```
lista_paises_final2 = []
for p in paises.groups.keys():
    try:
        e = educacion_pais.get_group((p, gender))
        lista_paises_final2.append(p)
    except KeyError:
        continue
```

```
print('Lista final de países en ambos Dataframes : ', len(lista_paises_final2),␣
↪'\n')
print(lista_paises_final2)
```

```
Lista final de países en ambos Dataframes :  127

['Albania', 'Algeria', 'Antigua and Barbuda', 'Argentina', 'Armenia', 'Aruba',
'Australia', 'Austria', 'Azerbaijan', 'Barbados', 'Belarus', 'Belgium',
'Belize', 'Benin', 'Bermuda', 'Bhutan', 'Bosnia and Herzegovina', 'Brunei
Darussalam', 'Bulgaria', 'Burkina Faso', 'Burundi', 'Cambodia', 'Cameroon',
'Central African Republic', 'Chad', 'Chile', 'China', 'Colombia', 'Comoros',
'Cook Islands', 'Croatia', 'Cuba', 'Cyprus', "Côte d'Ivoire", 'Denmark',
'Djibouti', 'Egypt', 'El Salvador', 'Eritrea', 'Estonia', 'Ethiopia', 'Finland',
'France', 'Gambia', 'Georgia', 'Gibraltar', 'Greece', 'Guinea', 'Guyana',
'Honduras', 'Hungary', 'Iceland', 'India', 'Indonesia', 'Ireland', 'Italy',
'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kyrgyzstan', "Lao People's
Democratic Republic", 'Latvia', 'Lebanon', 'Liberia', 'Liechtenstein',
'Lithuania', 'Luxembourg', 'Madagascar', 'Malawi', 'Mali', 'Malta',
'Mauritania', 'Mauritius', 'Mexico', 'Monaco', 'Mongolia', 'Montenegro',
'Morocco', 'Mozambique', 'Nepal', 'Netherlands', 'New Zealand', 'Niger',
'Norway', 'Oman', 'Panama', 'Paraguay', 'Peru', 'Philippines', 'Poland',
'Portugal', 'Puerto Rico', 'Qatar', 'Republic of Korea', 'Romania', 'Rwanda',
'Saint Lucia', 'San Marino', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal',
'Serbia', 'Seychelles', 'Slovakia', 'Slovenia', 'Spain', 'Sri Lanka', 'Sudan',
'Sweden', 'Switzerland', 'Syrian Arab Republic', 'Tajikistan', 'Thailand',
'Timor-Leste', 'Tunisia', 'Turkey', 'Uganda', 'Ukraine', 'United Arab Emirates',
'United Republic of Tanzania', 'United States of America', 'Uruguay',
'Uzbekistan', 'Viet Nam', 'Yemen', 'Zimbabwe']
```

[29]:
```
# Ahora creamos dos listas, una para el FR y otra para el nivel educativo
l_FR2 = []
l_ED = []
for p in lista_paises_final2:
    l_FR2.append(paises.get_group(p)['Value'].iloc[2])  # El lugar 2␣
↪corresponde a los datos para 2010

    e = educacion_pais.get_group((p, gender))
    l_ED.append(e['Observation Value'].iloc[0]) # Solo se tiene el dato para␣
↪2010

print('\nFR (tamaño: {}) \nDatos :\n{}'.format(len(l_FR2), l_FR2))
print('\nEducación (tamaño: {}) \nDatos :\n{}'.format(len(l_ED), l_ED))
```

```
FR (tamaño: 127)
Datos :
[1.64, 2.724, 2.0, 2.37, 1.72, 1.76, 1.952, 1.399, 1.83, 1.75, 1.442, 1.821,
```

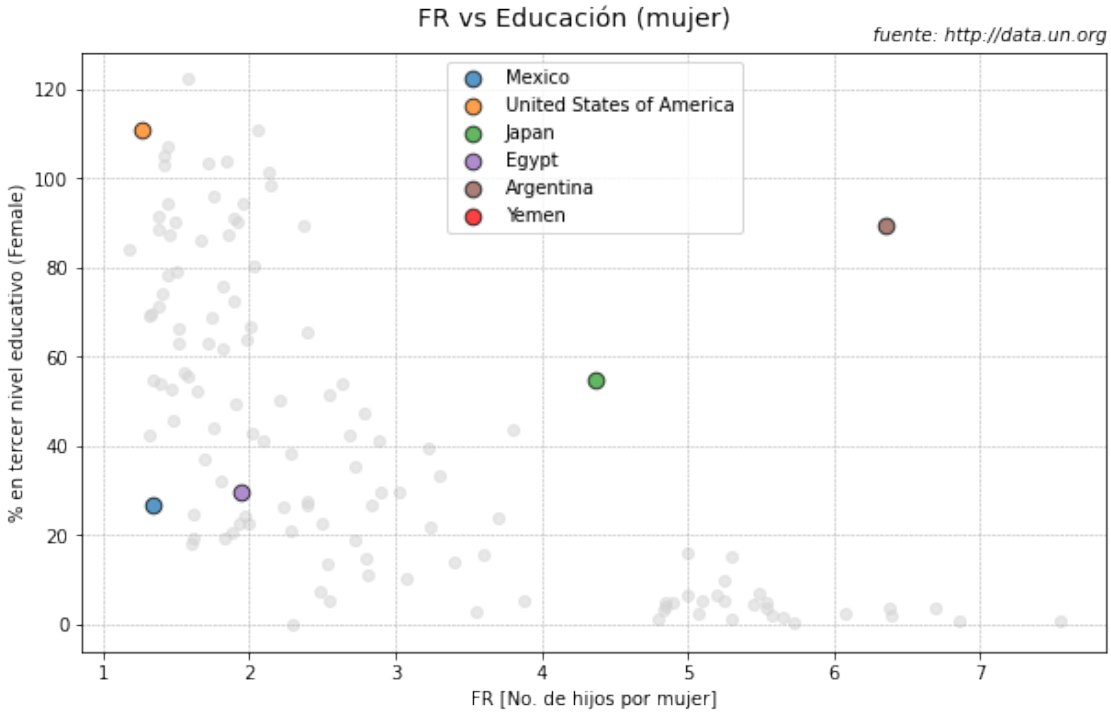
```
2.84, 5.495, nan, 2.55, 1.312, 1.884, 1.519, 6.079, 6.39, 3.081, 5.25, 5.3,
6.853, 1.898, 1.62, 2.1, 4.9, nan, 1.522, 1.577, 1.48, 5.25, 1.854, 3.55, 3.02,
2.395, 4.8, 1.662, 5.45, 1.84, 1.977, 5.65, 1.8, nan, 1.416, 5.535, 2.726,
3.244, 1.329, 2.131, 2.796, 2.5, 2.003, 1.436, 2.28, 1.339, 3.8, 2.541, 2.78,
3.4, 1.493, 1.9, 5.2, nan, 1.417, 1.621, 4.83, 5.73, 6.7, 1.393, 5.07, 1.698,
2.4, nan, 2.397, 1.82, 2.53, 5.537, 2.813, 1.746, 2.142, 7.55, 1.924, 2.9,
2.632, 2.892, 2.68, 3.3, 1.374, 1.372, 1.72, 2.23, 1.173, 1.51, 4.85, 1.6, nan,
4.85, 3.225, 5.1, 1.583, 2.3, 1.317, 1.435, 1.454, 2.279, 5.0, 1.891, 1.469,
3.7, 3.609, 1.556, 5.3, 2.024, 2.202, 6.38, 1.383, 1.97, 5.579, 2.055, 2.03,
2.49, 1.928, 5.0, 3.885]

Educación (tamaño: 127)
Datos :
[52.38337, 35.33614, 22.6758, 89.15845, 63.13266, 43.94988, 94.4554, 74.21418,
19.15638, 95.85448, 94.43531, 75.79072, 26.86975, 6.88722, 40.60865, 5.28042,
42.45824, 20.63689, 66.38689, 2.30799, 2.16036, 10.4775, 9.89285, 1.24086,
0.62324, 72.3256, 24.82943, 41.28762, 4.85552, 0.0, 62.82417, 122.09149,
45.62976, 5.48327, 87.4787, 2.8065, 29.51936, 27.62104, 1.28404, 85.84674,
4.40732, 103.71226, 63.94209, 1.76123, 32.27263, 0.0, 104.93733, 5.07742,
18.92756, 22.00731, 69.67089, 101.31812, 14.96807, 22.55804, 66.54445, 78.14745,
38.30558, 54.72148, 43.48516, 51.52325, 47.51425, 14.16958, 90.27941, 49.31282,
6.44251, 27.35761, 103.05674, 19.23368, 3.43914, 0.52294, 3.55572, 54.0586,
2.54282, 36.91036, 26.61178, 0.0, 65.26063, 61.84135, 13.67459, 3.59284,
11.05068, 68.86876, 98.58846, 0.81802, 90.37135, 29.62032, 53.84087, 41.2427,
42.50251, 33.14009, 88.32686, 71.18919, 103.19948, 26.24801, 83.86655, 78.87271,
4.87987, 17.96875, 77.30496, 4.22886, 39.57034, 5.48741, 55.62779, 0.0,
69.33533, 106.96577, 87.35101, 20.87279, 16.22916, 90.8397, 52.51459, 24.06156,
15.64289, 56.33325, 15.15506, 42.62535, 50.19507, 3.52858, 91.50879, 24.31404,
1.87433, 110.74234, 80.3234, 7.55595, 22.71635, 6.40428, 5.231]
```

1.7.2 Visualización FR vs Educación

[30]:
```
fig = plt.figure(figsize=(10,6)) # Cambiamos el tamaño de la figura

# Graficamos el nivel de eduación (eje $y$) en función de la fertilidad (eje␣
↪$x$)
plt.scatter(l_FR2, l_ED, marker='o', color='lightgray', alpha=0.5)

# Información adicional y títulos
plt.title('fuente: http://data.un.org', loc='right', fontstyle='italic',␣
↪fontsize=10)
plt.suptitle('FR vs Educación (mujer) ', y = 0.94, fontsize=14)

plt.grid(ls='--', lw=0.5, c='#AAAAAA')
plt.xlabel('FR [No. de hijos por mujer]')
plt.ylabel('% en tercer nivel educativo ({})'.format(gender))
```

```
# Gráfica de los países elegido en el paso 5 usando puntos de colores y␣
 ↪resaltados.
for p, c in paises_colores.items():
    try:
        ip = lista_paises_final2.index(p)
        plt.scatter(l_FR[ip], l_ED[ip],
                    marker='o', s=75,
                    facecolor=c, edgecolor='k', alpha=0.75,
                    zorder=1000, label=p)
    except ValueError:
        continue

plt.legend(loc='upper center')
```

[30]: <matplotlib.legend.Legend at 0x7fe81e7a8580>



- ¿Qué se puede decir de esta visualización?
- ¿Cómo se podría mejorar? (colores, dimensiones, anotaciones, leyendas, etc)
- ¿Podría realizar un ajuste de una curva a estos datos?
- ¿Cree que los datos están correctos?

1.8 Visualización final FR vs PCI vs ED.

Combinar las tres variables: FR, ingresos, educación, en una sola visualización.

1.8.1 Tratamiento de los datos.

[31]:
```
# Creamos una lista de países que tengan la información de las tres variables.
lista_paises_final3 = []
for p in lista_paises_final:
    if p in lista_paises_final2:
        lista_paises_final3.append(p)

print(len(lista_paises_final3), lista_paises_final3)
```

```
112 ['Albania', 'Algeria', 'Antigua and Barbuda', 'Armenia', 'Australia',
'Austria', 'Azerbaijan', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin',
'Bhutan', 'Bosnia and Herzegovina', 'Bulgaria', 'Burkina Faso', 'Burundi',
'Cambodia', 'Cameroon', 'Central African Republic', 'Chad', 'Chile', 'China',
'Colombia', 'Comoros', 'Croatia', 'Cyprus', 'Denmark', 'Egypt', 'El Salvador',
'Eritrea', 'Estonia', 'Ethiopia', 'Finland', 'France', 'Gambia', 'Georgia',
'Greece', 'Guinea', 'Guyana', 'Honduras', 'Hungary', 'Iceland', 'India',
'Indonesia', 'Ireland', 'Italy', 'Japan', 'Jordan', 'Kazakhstan', 'Kyrgyzstan',
"Lao People's Democratic Republic", 'Latvia', 'Lebanon', 'Liberia', 'Lithuania',
'Luxembourg', 'Madagascar', 'Malawi', 'Mali', 'Malta', 'Mauritania',
'Mauritius', 'Mexico', 'Mongolia', 'Montenegro', 'Morocco', 'Mozambique',
'Nepal', 'Netherlands', 'New Zealand', 'Niger', 'Norway', 'Oman', 'Panama',
'Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal', 'Qatar', 'Republic of
Korea', 'Romania', 'Rwanda', 'Saint Lucia', 'Sao Tome and Principe', 'Saudi
Arabia', 'Senegal', 'Serbia', 'Seychelles', 'Slovakia', 'Slovenia', 'Spain',
'Sri Lanka', 'Sudan', 'Sweden', 'Switzerland', 'Syrian Arab Republic',
'Tajikistan', 'Thailand', 'Timor-Leste', 'Tunisia', 'Turkey', 'Uganda',
'Ukraine', 'United Arab Emirates', 'United Republic of Tanzania', 'United States
of America', 'Uruguay', 'Uzbekistan', 'Viet Nam', 'Yemen']
```

[32]:
```
# Ahora creamos tres listas para las variables FR, PCI y educación, para los
# países de la lista anterior.
lf = []
li = []
le = []
for p in lista_paises_final3:
    lf.append(paises.get_group(p)['Value'].iloc[2])  # El lugar 2 corresponde a␣
 ↪los datos para 2010
    li.append(ingreso_pais.get_group(p)['Value'].iloc[0]) # Solo se tiene el␣
 ↪dato para 2010
    e = educacion_pais.get_group((p, gender))
    le.append(e['Observation Value'].iloc[0]) # Solo se tiene el dato para 2010

print('\nFR (tamaño: {}) \nDatos :\n{}'.format(len(lf), lf))
print('\nPCI (tamaño: {}) \nDatos :\n{}'.format(len(li), li))
print('\nEducación (tamaño: {}) \nDatos :\n{}'.format(len(le), le))
```

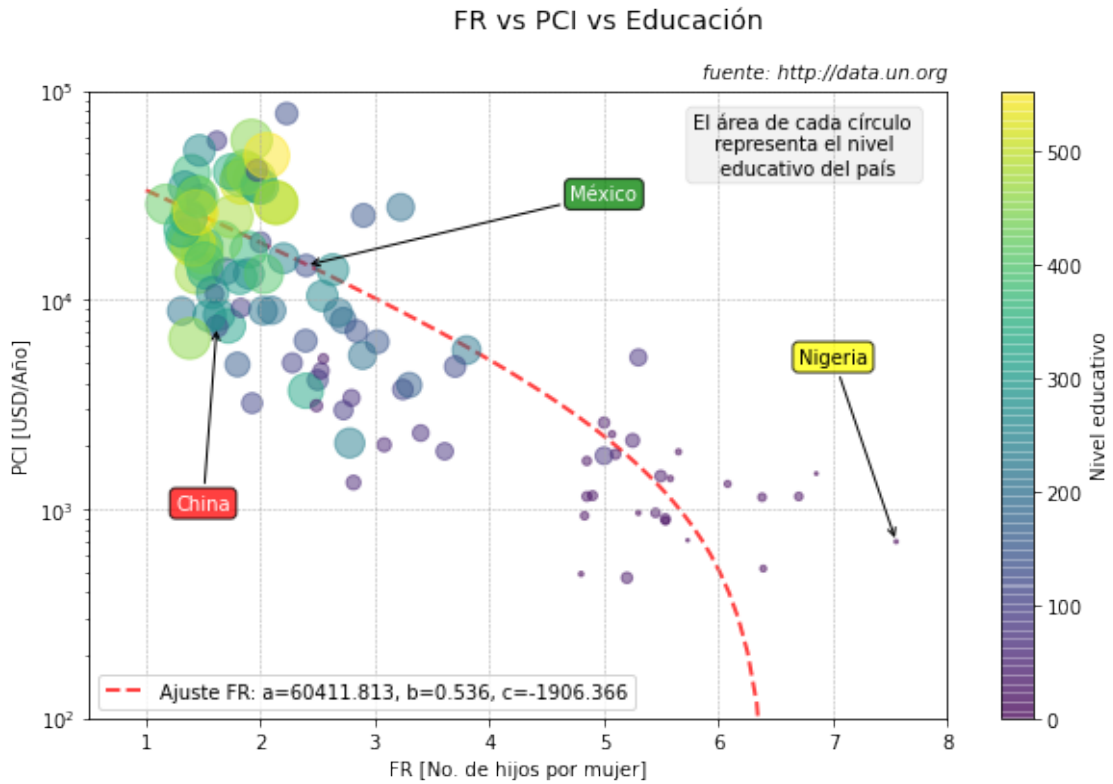
```
FR (tamaño: 112)
Datos :
[1.64, 2.724, 2.0, 1.72, 1.952, 1.399, 1.83, 1.75, 1.442, 1.821, 2.84, 5.495,
2.55, 1.312, 1.519, 6.079, 6.39, 3.081, 5.25, 5.3, 6.853, 1.898, 1.62, 2.1, 4.9,
1.522, 1.48, 1.854, 3.02, 2.395, 4.8, 1.662, 5.45, 1.84, 1.977, 5.65, 1.8,
1.416, 5.535, 2.726, 3.244, 1.329, 2.131, 2.796, 2.5, 2.003, 1.436, 1.339, 3.8,
2.541, 2.78, 3.4, 1.493, 1.9, 5.2, 1.417, 1.621, 4.83, 5.73, 6.7, 1.393, 5.07,
1.698, 2.4, 2.397, 1.82, 2.53, 5.537, 2.813, 1.746, 2.142, 7.55, 1.924, 2.9,
2.632, 2.892, 2.68, 3.3, 1.374, 1.372, 2.23, 1.173, 1.51, 4.85, 1.6, 4.85,
3.225, 5.1, 1.583, 2.3, 1.317, 1.435, 1.454, 2.279, 5.0, 1.891, 1.469, 3.7,
3.609, 1.556, 5.3, 2.024, 2.202, 6.38, 1.383, 1.97, 5.579, 2.055, 2.03, 2.49,
1.928, 5.0]

PCI (tamaño: 112)
Datos :
[8480, 7970, 18840, 7510, 37580, 40260, 9160, 24950, 13470, 38450, 7090, 1440,
5230, 8850, 13700, 1320, 520, 2030, 2130, 960, 1480, 17360, 7470, 8910, 1160,
18070, 30300, 41540, 6300, 6380, 490, 19040, 960, 36550, 34970, 1880, 4910,
26770, 900, 2970, 3730, 19720, 29270, 3400, 4140, 34830, 31930, 34570, 5750,
10440, 2070, 2310, 16830, 13340, 470, 18350, 57690, 930, 710, 1150, 24390, 2280,
13670, 14600, 3680, 12820, 4580, 880, 1340, 40900, 29020, 700, 58130, 25320,
13930, 5430, 8690, 3920, 19220, 24670, 77640, 28650, 15260, 1150, 10680, 1700,
27720, 1830, 10830, 21600, 21770, 26120, 31070, 5000, 1800, 40130, 51760, 4800,
1890, 8410, 5310, 8870, 15810, 1140, 6540, 41550, 1400, 48880, 13290, 3120,
3210, 2590]

Educación (tamaño: 112)
Datos :
[52.38337, 35.33614, 22.6758, 63.13266, 94.4554, 74.21418, 19.15638, 95.85448,
94.43531, 75.79072, 26.86975, 6.88722, 5.28042, 42.45824, 66.38689, 2.30799,
2.16036, 10.4775, 9.89285, 1.24086, 0.62324, 72.3256, 24.82943, 41.28762,
4.85552, 62.82417, 45.62976, 87.4787, 29.51936, 27.62104, 1.28404, 85.84674,
4.40732, 103.71226, 63.94209, 1.76123, 32.27263, 104.93733, 5.07742, 18.92756,
22.00731, 69.67089, 101.31812, 14.96807, 22.55804, 66.54445, 78.14745, 54.72148,
43.48516, 51.52325, 47.51425, 14.16958, 90.27941, 49.31282, 6.44251, 103.05674,
19.23368, 3.43914, 0.52294, 3.55572, 54.0586, 2.54282, 36.91036, 26.61178,
65.26063, 61.84135, 13.67459, 3.59284, 11.05068, 68.86876, 98.58846, 0.81802,
90.37135, 29.62032, 53.84087, 41.2427, 42.50251, 33.14009, 88.32686, 71.18919,
26.24801, 83.86655, 78.87271, 4.87987, 17.96875, 4.22886, 39.57034, 5.48741,
55.62779, 0.0, 69.33533, 106.96577, 87.35101, 20.87279, 16.22916, 90.8397,
52.51459, 24.06156, 15.64289, 56.33325, 15.15506, 42.62535, 50.19507, 3.52858,
91.50879, 24.31404, 1.87433, 110.74234, 80.3234, 7.55595, 22.71635, 6.40428]
```

Hacemos la visualización usando la variable eduación como un valor para el área de los círculos que se van a graficar.

[33]:
```
fig = plt.figure(figsize=(10,6)) # Cambiamos el tamaño de la figura
```

```
# Eje x: FR
# Eje y: PCI
# Area y color: Educación

se = np.array(le) * 5
max_se = np.max(se)
color = [int(c) for c in se]
scatter = plt.scatter(lf, li, marker='o', c=se, alpha=0.5, s=se, cmap="viridis")
plt.colorbar(mappable=scatter, label='Nivel educativo')

plt.title('fuente: http://data.un.org', loc='right', fontstyle='italic',␣
↪fontsize=10)
plt.suptitle(' FR vs PCI vs Educación ', y = 0.98, fontsize=14)

plt.grid(ls='--', lw=0.5, c='#AAAAAA')
plt.xlabel('FR [No. de hijos por mujer]')
plt.ylabel('PCI [USD/Año]')

# Gráficamos el ajuste entre fertilidad e ingresos
plt.plot(x, y, '--', c='red', lw=2.0, zorder=0, alpha=0.75,
         label='Ajuste FR: a={:5.3f}, b={:5.3}, c={:5.3f}'.format(popt[0],␣
↪popt[1], popt[2]))

# Identificamos algunos países en el gráfico
ipm = lista_paises_final3.index('Mexico')
plt.annotate('México', xy=(lf[ipm], li[ipm]), xytext=(5, 30000),
             bbox=dict(boxstyle='round', facecolor='green', edgecolor='black',␣
↪alpha=0.75, linewidth=1.1),
             arrowprops=dict(arrowstyle='->', facecolor='black',␣
↪edgecolor='black'),
             fontsize=10, color='white', horizontalalignment='center')

ipn = lista_paises_final3.index('Niger')
plt.annotate('Nigeria', xy=(lf[ipn], li[ipn]), xytext=(7, 5000),
             bbox=dict(boxstyle='round', facecolor='yellow', edgecolor='black',␣
↪alpha=0.75, linewidth=1.1),
             arrowprops=dict(arrowstyle='->', facecolor='black',␣
↪edgecolor='black'),
             fontsize=10, color='black', horizontalalignment='center')

ipj = lista_paises_final3.index('China')
plt.annotate('China', xy=(lf[ipj], li[ipj]), xytext=(1.5, 1000),
             bbox=dict(boxstyle='round', facecolor='red', edgecolor='black',␣
↪alpha=0.75, linewidth=1.1),
             arrowprops=dict(arrowstyle='->', facecolor='black',␣
↪edgecolor='black'),
```

```
            fontsize=10, color='white', horizontalalignment='center')

# Agregamos un poco más de información
plt.text(6.75, 40000, 'El área de cada círculo \n representa el nivel \n␣
 ↪educativo del país',
            transform=plt.gca().transData,
            horizontalalignment='center', color='black',
            bbox=dict(boxstyle='round', facecolor='gray', edgecolor='black',␣
 ↪alpha=0.1, linewidth=0.75))

# Usamos escala semilogarítmica
plt.yscale('log')
plt.ylim(1e2,1e5)
plt.xlim(0.5,8)
plt.legend(loc='lower left', fontsize=10)
plt.savefig('FR04_fert_ing_educ.pdf')
```



Ejercicio: Use otra escala para los ejes y modifique el mapa de color para los círculos.

[ ]: