AnalisisPoblacion

October 30, 2021

1 Práctica 1: Análisis de la población mundial

C03 : Visualización Científica y Narrativas

RAUGM 2021: Geociencias e inclusión

This notebook by Luis M. de la Cruz Salas is licensed under Attribution-NonCommercial-NoDerivatives 4.0 International

- Extraer información importante de un conjunto de datos y resaltarla mediante algunas técnicas simples.
- Construir una visualización que permita comparar el cambio en el Fertility Rate en función del tiempo.
- Con esta visualización tratar de responder las siguientes preguntas:
  - ¿Los países ricos están recuperando su FR?
  - ¿Países en desarrollo como China y Brasil están estabilizando sus poblaciones?
  - ¿Cómo ha sido la evolución de la población en México?
  - ¿y en otros países?
- Usar Numpy, Pandas y Matplotlib.

1.1 Importar las bibliotecas

Incluir 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
```

1.2 Lectura del archivo

Obtener la información de http://data.un.org/ con la búsqueda 'fertility rate', descargar el archivo y leerlo en un DataFrame.

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

Observe que se puede leer directamente el archivo comprimido. Si descomprime el archivo ZIP obtendrá el archivo en formato CSV el cual también puede leerse como arriba solo cambiando la extensión.

Veamos que contiene el DataFrame:

[3]:
```
FR # Despliega la parte inicial y final el DataFrame.
```

[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]
```

Observe que el número total de renglones es de **4046** y se tienen **4** columnas de datos.

Podemos desplegar solamente el principio o solamente el final:

[4]:
```
FR.head() # Despliega los primeros renglones del DataFrame
```

[4]:
```
  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
```

[5]:
```
FR.tail() # Despliega los últimos renglones del DataFrame
```

[5]:
```
     Country or Area    Year(s) Variant  Value
4041        Zimbabwe  1970-1975  Medium    7.4
4042        Zimbabwe  1965-1970  Medium    7.4
4043        Zimbabwe  1960-1965  Medium    7.3
4044        Zimbabwe  1955-1960  Medium    7.0
4045        Zimbabwe  1950-1955  Medium    6.8
```

1.3 Agrupar por países

Vamos a organizar el DataFrame por países para que su análisis sea más sencillo.

[6]:
```
FR.index # Nos da el rango total de renglones
```

[6]:
```
RangeIndex(start=0, stop=4046, step=1)
```

Observamos que tenemos 4096 rengones, lo anterior es porque se tiene información para varios rangos de años por país. Para agrupar la información por país hacemos lo siguiente:

[7]:
```
# Primero se agrupa por país
paises = FR.groupby('Country or Area')
```

[8]:
```
print(type(paises)) # paises es un objeto de tipo DataFrameGroupBy
```

```
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>
```

1.3.1 Exploración de la agrupación por países

Vamos a ver que tiene el grupo **países** transformándolo a un DataFrame:

[9]:
```
dummy = pd.DataFrame(paises) # Casting
dummy.index
```

[9]:
```
RangeIndex(start=0, stop=286, step=1)
```

[10]:
```
dummy
```

[10]:
```
                           0  \
0                Afghanistan
1                     Africa
2                    Albania
3                    Algeria
4             American Samoa
..                       …
281                     World
282  World Bank income groups
283                     Yemen
284                    Zambia
285                  Zimbabwe

                                                    1
0       Country or Area    Year(s) Variant  Value
0…
1       Country or Area    Year(s) Variant  Value
1…
2       Country or Area    Year(s) Variant  Value
2…
3       Country or Area    Year(s) Variant  Value
4…
4       Country or Area    Year(s) Variant  Value
5…
..                                                …
281       Country or Area    Year(s) Variant  Value…
282                  Country or Area    Year(s) Vari…
```

```
283       Country or Area    Year(s) Variant  Value…
284       Country or Area    Year(s) Variant  Value…
285       Country or Area    Year(s) Variant  Value…

[286 rows x 2 columns]
```

El DataFrame `dummy` contiene dos columnas: la 0 y la 1. La 0 nos da el nombre del país y la 1 la información del país.

Vamos a explorar este DataFrame:

[11]:
```
dummy[0] # Imprime la columa correspondiente. Cambie el 0 por 1 y vea lo que␣
↪sucede
```

[11]:
```
0                    Afghanistan
1                         Africa
2                        Albania
3                        Algeria
4                 American Samoa
                  …
281                         World
282    World Bank income groups
283                         Yemen
284                        Zambia
285                      Zimbabwe
Name: 0, Length: 286, dtype: object
```

[12]:
```
print(type(dummy[0]), type(dummy[1])) # Veamos el tipo de las columas
```

```
<class 'pandas.core.series.Series'> <class 'pandas.core.series.Series'>
```

[13]:
```
dummy.iloc[0] # Con la función iloc accedemos a un renglón particular usando un␣
↪entero
```

[13]:
```
0                                          Afghanistan
1       Country or Area    Year(s) Variant  Value
0…
Name: 0, dtype: object
```

[14]:
```
dummy.iloc[0][0] # Columna correspondiente del renglón, cambie el segundo 0 por␣
↪1 y vea el resultado
```

[14]:
```
'Afghanistan'
```

[15]:
```
print(type(dummy.iloc[0][0]), type(dummy.iloc[0][1])) # tipos de las columnas
```

```
<class 'str'> <class 'pandas.core.frame.DataFrame'>
```

Para acceder a la información de un país podemos hacer lo siguiente:

[16]:
```
dummy[dummy[0] == 'Mexico']
```

[16]:

	0	1
158	Mexico	Country or Area Year(s) Variant Value…

Ejercicio: Imprima la información del FR de México

[17]:
```
dummy.iloc[158][1]
```

[17]:

	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

1.3.2 Usando `get_group()`

Otra manera de extraer los datos de cada país es usando la función `get_group()`.

Por ejemplo, vamos a obtener la información de dos países con un PIB por arriba de México, por ejemplo España y Suecia.

[18]:
```
# Después se obtienen los datos de España y Suecia
spa = paises.get_group('Spain')
swe = paises.get_group('Sweden')
```

[19]:
```
print(type(spa), type(swe))
```

```
<class 'pandas.core.frame.DataFrame'> <class 'pandas.core.frame.DataFrame'>
```

[20]:
```
spa
```

[20]:

	Country or Area	Year(s)	Variant	Value
3402	Spain	2015-2020	Medium	1.330
3403	Spain	2010-2015	Medium	1.325
3404	Spain	2005-2010	Medium	1.454
3405	Spain	2000-2005	Medium	1.277
3406	Spain	1995-2000	Medium	1.187

```
3407           Spain  1990-1995  Medium  1.280
3408           Spain  1985-1990  Medium  1.460
3409           Spain  1980-1985  Medium  1.880
3410           Spain  1975-1980  Medium  2.550
3411           Spain  1970-1975  Medium  2.850
3412           Spain  1965-1970  Medium  2.840
3413           Spain  1960-1965  Medium  2.810
3414           Spain  1955-1960  Medium  2.700
3415           Spain  1950-1955  Medium  2.530
```

[21]:
```
swe
```

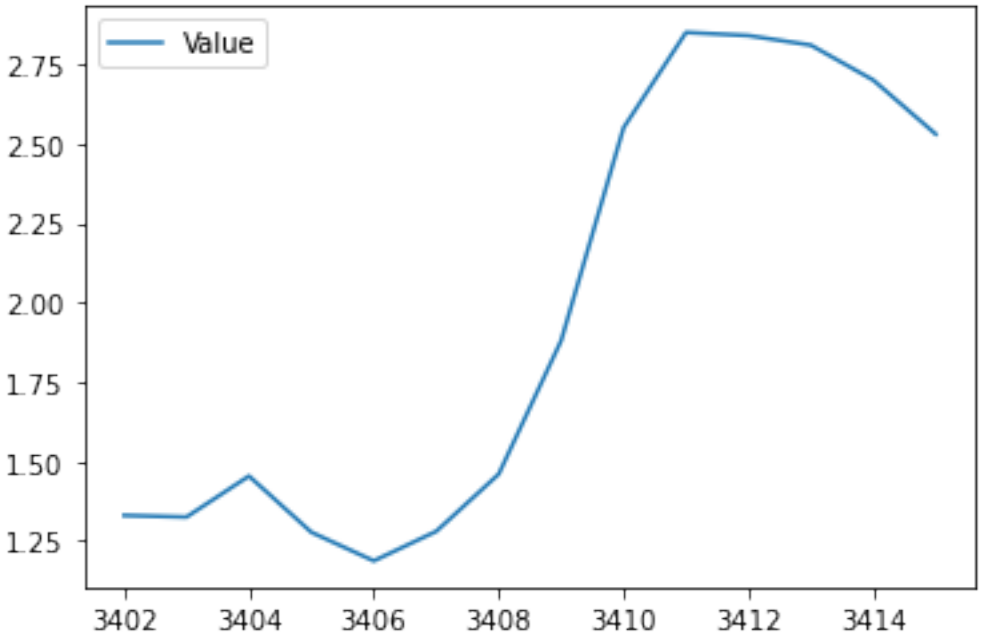
[21]:
```
     Country or Area    Year(s) Variant  Value
3500          Sweden  2015-2020  Medium  1.850
3501          Sweden  2010-2015  Medium  1.902
3502          Sweden  2005-2010  Medium  1.891
3503          Sweden  2000-2005  Medium  1.670
3504          Sweden  1995-2000  Medium  1.559
3505          Sweden  1990-1995  Medium  2.006
3506          Sweden  1985-1990  Medium  1.909
3507          Sweden  1980-1985  Medium  1.644
3508          Sweden  1975-1980  Medium  1.665
3509          Sweden  1970-1975  Medium  1.906
3510          Sweden  1965-1970  Medium  2.170
3511          Sweden  1960-1965  Medium  2.310
3512          Sweden  1955-1960  Medium  2.247
3513          Sweden  1950-1955  Medium  2.237
```

1.4 Graficación usando .plot de DataFrame

Pandas provee de la función plot para hacer un gráfico rápido del DataFrame. Veamos como funciona:

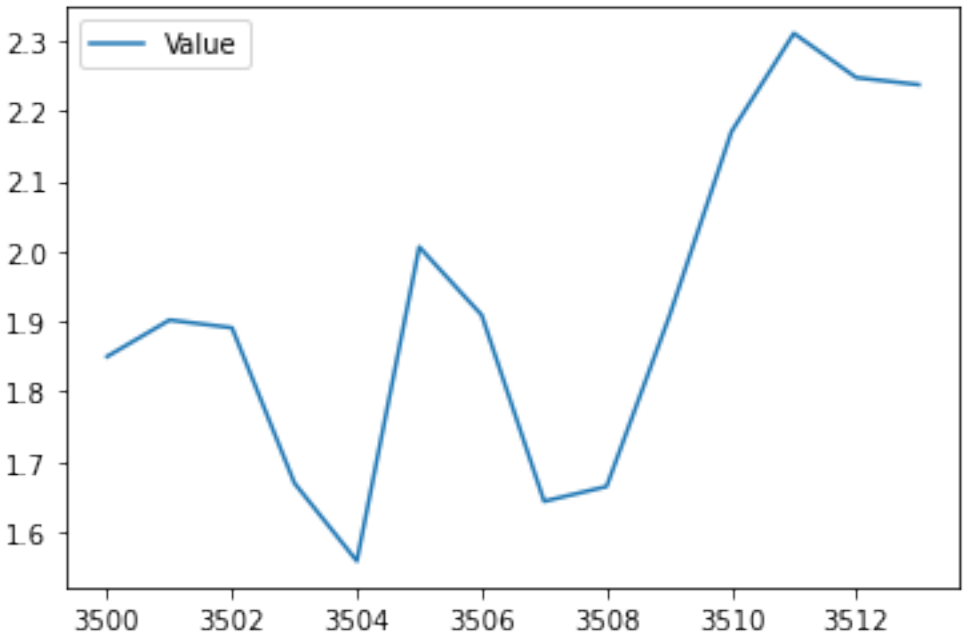
[22]:
```
spa.plot()
```

[22]:
```
<AxesSubplot:>
```



[23]:
```
swe.plot()
```

[23]: <AxesSubplot:>



Muchos de los tipos de datos que ofrecen bibliotecas con Pandas, Xarray y otras contienen una

función plot que realiza gráficas de manera automática. Casi todas están basadas en Matplotlib. Cada una de esas funciones tiene sus propias funcionalidades y parámentros que pueden modificarse para mejorar las gráficas resultantes.

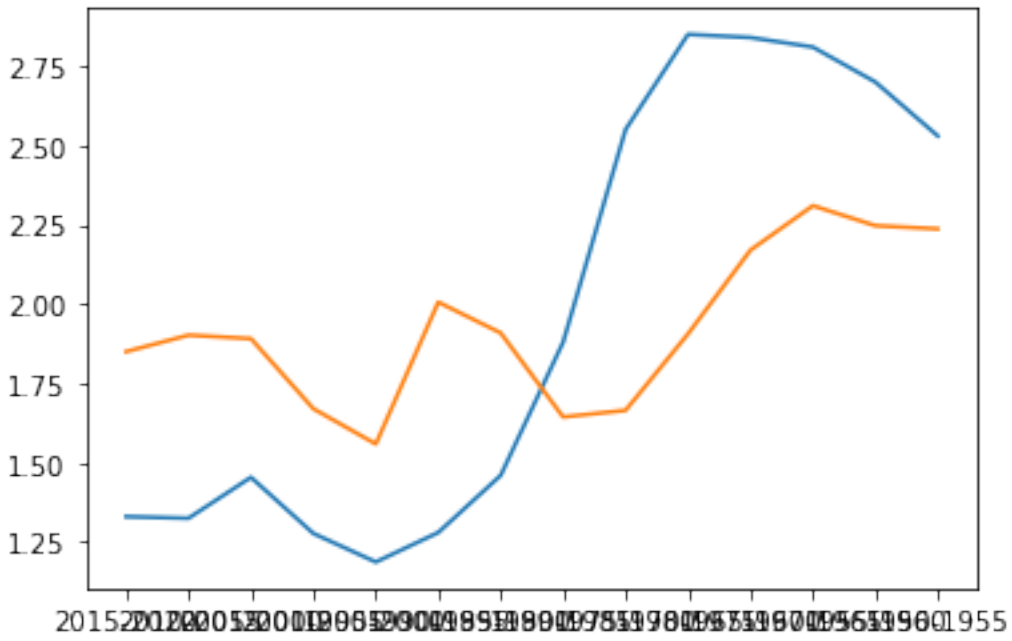
1.5 Visualización con Matlplotlib.

En esta práctica usaremos Matplotlib como primer intento de lograr una buena visualización.

Para realizar las gráficas España y Suecia podemos hacer lo siguiente:

[24]:
```
# Se usa la función plot() que proviene de Matplotlib
plt.plot(spa['Year(s)'], spa['Value'])
plt.plot(swe['Year(s)'], swe['Value'])
```

[24]:
```
[<matplotlib.lines.Line2D at 0x7f06aaad46a0>]
```



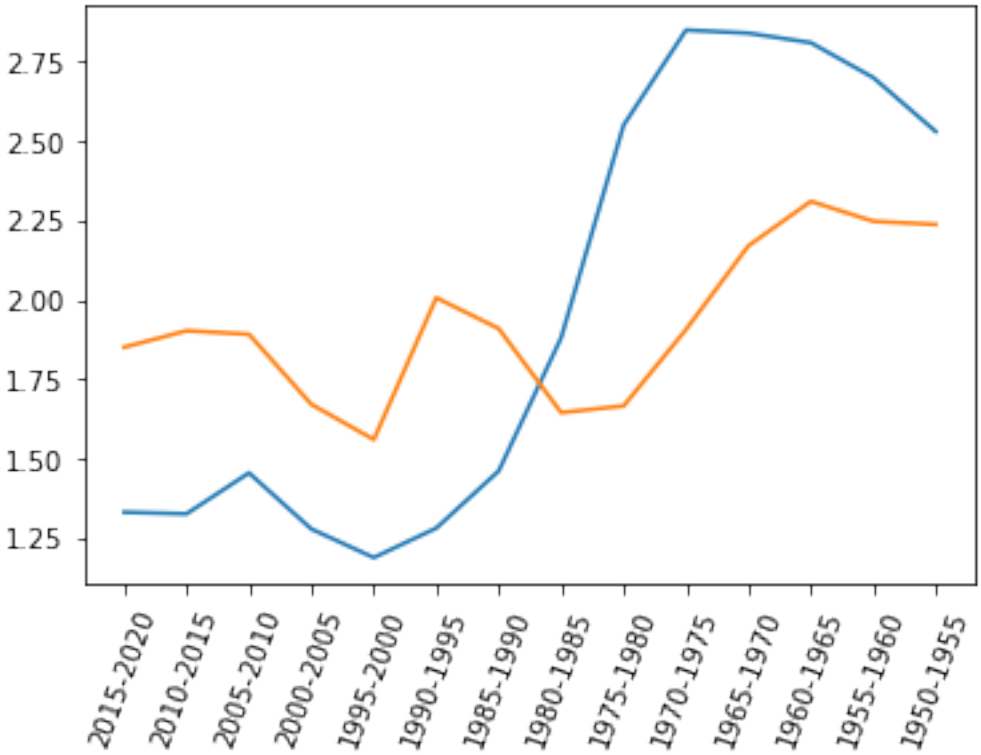
Como se puede observar ambas gráficas aparecen en una sola figura. Pero tenemos muchos problemas para entenderla, empezando con las etiquetas en el eje x (`xtics` y `xlabels`) que aparecen amontonadas.

1.5.1 Rotación de las etiquetas en el eje x.

Rotar los `xtics` para apreciarlos mejor.

[25]:
```
plt.xticks(rotation=70) # Rotación de 70 grados con respecto al eje x, en␣
 ↪sentido horario.
plt.plot(spa['Year(s)'], spa['Value'])
plt.plot(swe['Year(s)'], swe['Value'])
```

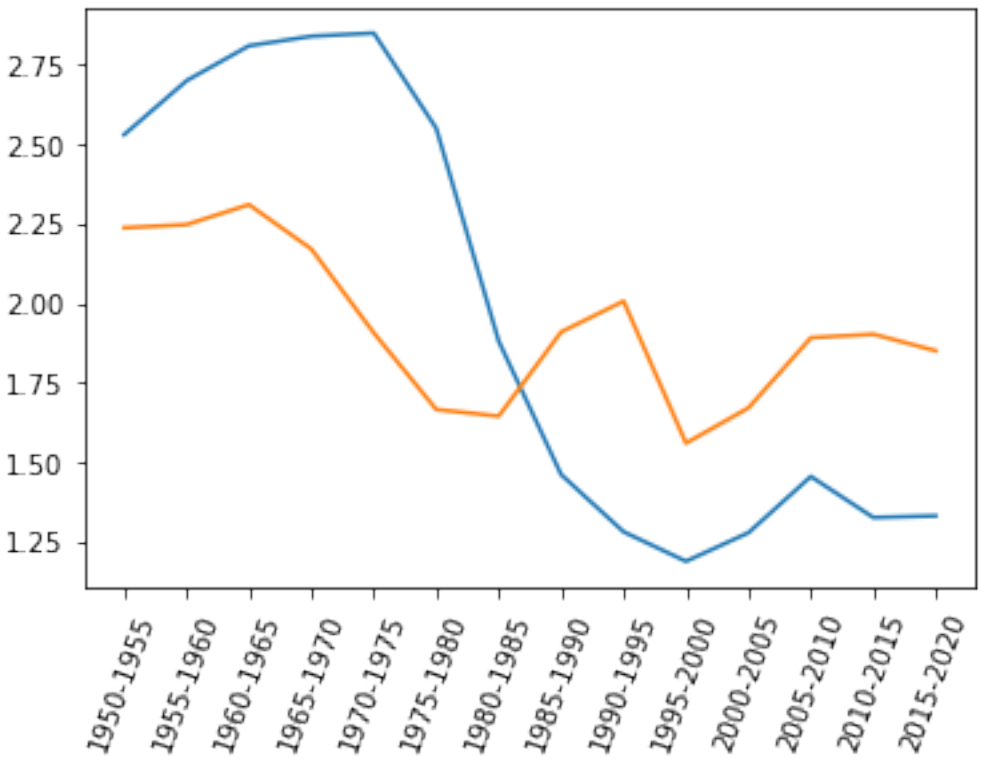
[25]: [<matplotlib.lines.Line2D at 0x7f06aab06130>]



Se observa en el eje x la información de mayor (2015-2020) a menor (1950-1955).

1.5.2 Invertir los datos del eje x

[26]:
```
plt.xticks(rotation=70)
plt.plot(spa['Year(s)'], spa['Value'])
plt.plot(swe['Year(s)'], swe['Value'])
plt.gca().invert_xaxis() # La función gca() obtiene el objeto que controla los
                         # ejes. Una vez teniendo el objeto de los ejes es
                         # posible usar la función invert_xaxis().
```



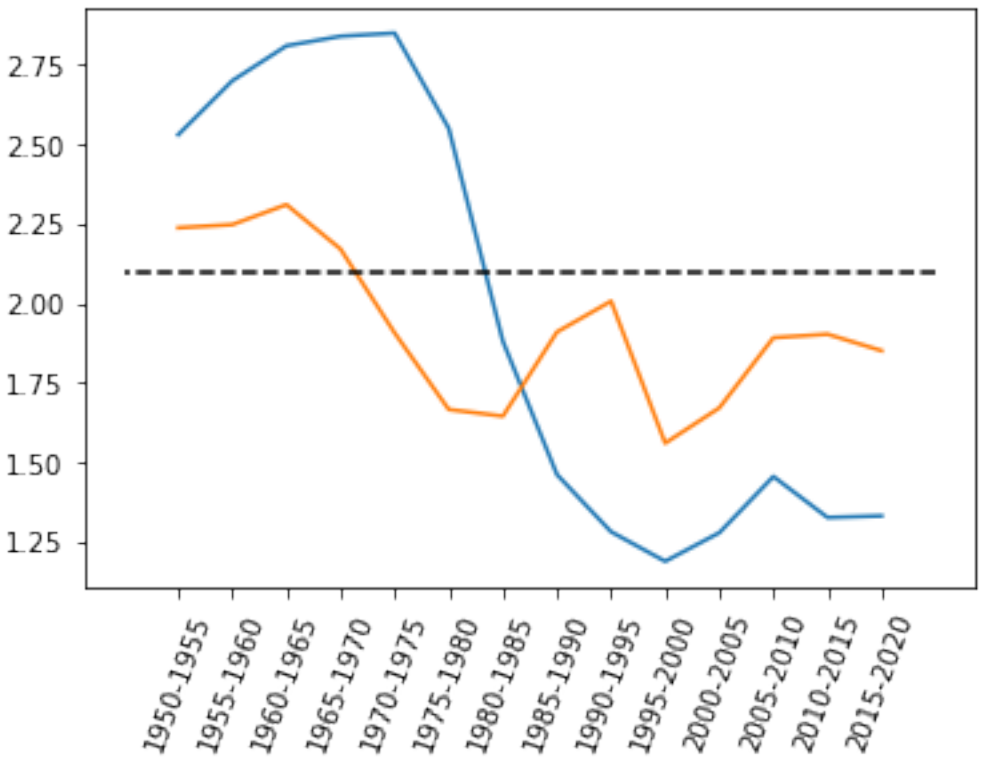
Ahora la información es un poco más clara y observamos que el comportamiento del FR desde los años 50's hasta nuestra epoca.

¿Qué puede decir de estos comporatmientos?

1.5.3 Resaltar el valor de reemplazo.

Graficamos una línea recta en el valor 2.1, que es considerado el valor de reemplazo.

[27]:
```
plt.xticks(rotation=70)
plt.plot(spa['Year(s)'], spa['Value'])
plt.plot(swe['Year(s)'], swe['Value'])
plt.plot([-1,14],[2.1,2.1], 'k--')      # Línea horizontal en 2.1
plt.gca().invert_xaxis()
```

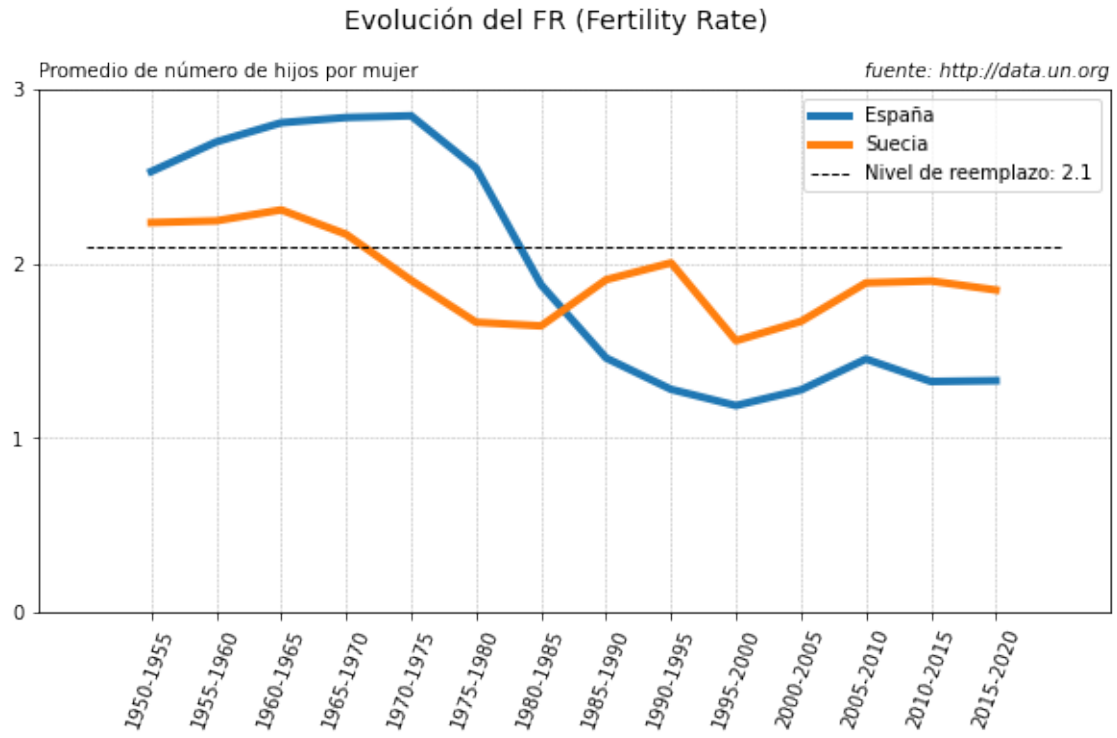


1.5.4 Decoración de la gráfica.

Decoramos la gráfica para tener una primera versión de esta visualización.

[28]:
```
fig = plt.figure(figsize=(10,5)) # Definimos el tamaño de la figura
plt.xticks(rotation=70)
plt.plot(spa['Year(s)'], spa['Value'], lw=4.0, label='España')
plt.plot(swe['Year(s)'], swe['Value'], lw=4.0, label='Suecia')
plt.plot([-1,14],[2.1,2.1], 'k--', lw=1.0, label='Nivel de reemplazo: 2.1')
plt.gca().invert_xaxis()
plt.ylim(0,3)              # Límites en el eje y
plt.yticks([0,1,2,3])      # Marcas en el eje y
plt.grid(ls='--', lw=0.5) # Rejilla

# Información adicional y títulos
plt.title('Promedio de número de hijos por mujer', loc='left', fontsize=10)
plt.title('fuente: http://data.un.org', loc='right', fontstyle='italic',␣
 ↪fontsize=10)
plt.suptitle('Evolución del FR (Fertility Rate)',y = 1.0, fontsize=14)
plt.legend()
```

[28]:
```
<matplotlib.legend.Legend at 0x7f06aa960e20>
```



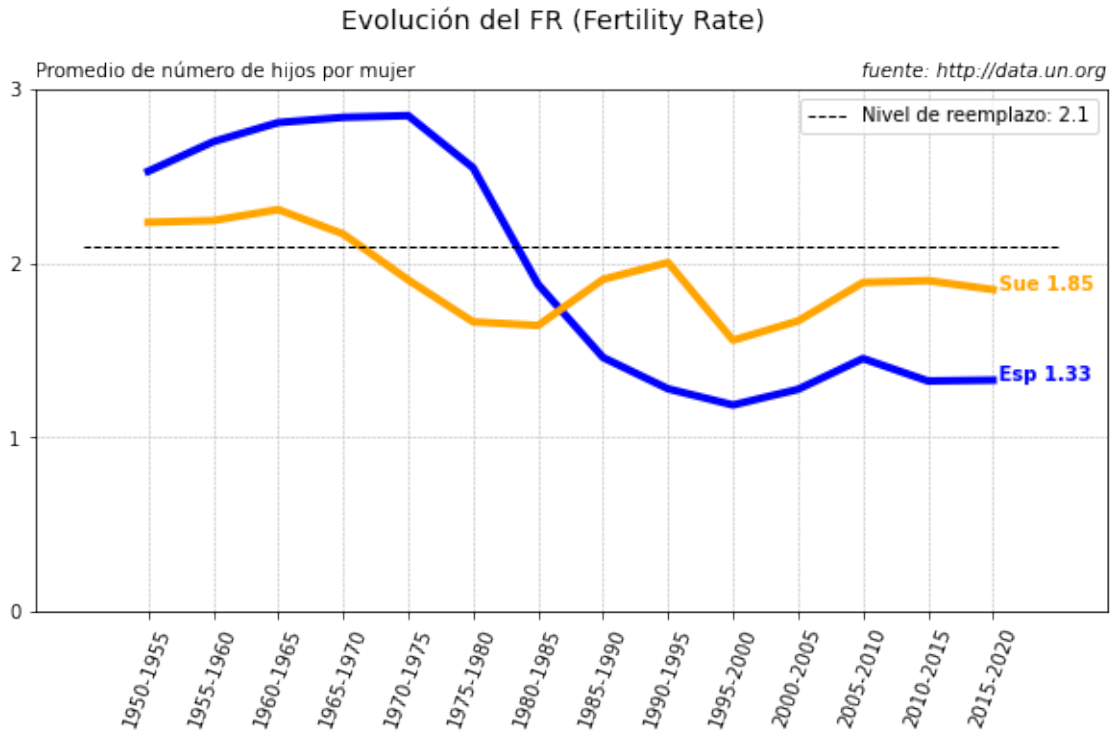
1.5.5 Etiquetado de curvas.

Agregar un texto en el extremo derecho de cada curva para identificarla. El objetivo es eliminar las leyendas de las curvas, pues más adelante tendremos muchas graficas que serán difíciles de distinguir.

[29]:
```
fig = plt.figure(figsize=(10,5))
plt.xticks(rotation=70)

# Definimos colores para las curvas y quitamos las leyendas
spa_color = 'blue'
swe_color = 'orange'
plt.plot(spa['Year(s)'], spa['Value'], lw=4.0, color=spa_color)
plt.plot(swe['Year(s)'], swe['Value'], lw=4.0, color=swe_color)

plt.plot([-1,14],[2.1,2.1], 'k--', lw=1.0, label='Nivel de reemplazo: 2.1')
plt.gca().invert_xaxis()
plt.ylim(0,3)
plt.yticks([0,1,2,3])
plt.grid(ls='--', lw=0.5)

# Información adicional y títulos
plt.title('Promedio de número de hijos por mujer', loc='left', fontsize=10)
```

```
plt.title('fuente: http://data.un.org', loc='right', fontstyle='italic',␣
 ↪fontsize=10)
plt.suptitle('Evolución del FR (Fertility Rate)',y = 1.0, fontsize=14)
plt.legend()

# Obtenemos el valor del FR y lo agregamos al final de cada curva
spa_val = spa['Value'].iloc[0]
swe_val = swe['Value'].iloc[0]
plt.text(x = 0, y = spa_val, s = ' Esp {:1.2f}'.format(spa_val), color =␣
 ↪spa_color, weight = 'bold')
plt.text(x = 0, y = swe_val, s = ' Sue {:1.2f}'.format(swe_val), color =␣
 ↪swe_color, weight = 'bold')
```

[29]: Text(0, 1.85, ' Sue 1.85')



Esta última gráfica muestra como el número promedio de hijos por mujer se incrementó levemente de 1950 a 1965 para Suecia, para después reducirse por debajo del nivel de reemplazo (NR). Desde entonces ha oscilado un par de veces y parece que a partir de 2005 comienza a estabilizarse en un valor por debajo de 2.1. Algo similar sucede con España, donde el incremento fue un poco más notorio de 1950 hasta 1975, para después bajar y estabilizarse en la década pasada. ¿Qué hechos historicos se podrían correlacionar con estos datos?

Lo que se desea mostrar (**la historia que se quiere contar**) es que la razón de nacimientos en todos los países se ha estabilizado, lo cual nos puede llevar a que la población mundial realmente

se estabilice en aproximadamente 90 mil millones de seres humanos.

1.6 Agregar otros países al análisis.

Hacer el mismo análisis para México y Yemen.

[30]:
```
# Extraer los datos de México y Yemen
mex = paises.get_group('Mexico')
yem = paises.get_group('Yemen')
```

1.6.1 Valor máximo del FR.

Obtener el valor máximo de FR de ambos países para usarlo en los límites del eje y.

[31]:
```
# Importamos la biblioteca ceil para redondear el valor máximo
from math import ceil

y_maximo = max(mex['Value'].max(), yem['Value'].max()) # El máximo entre dos␣
 ↪países.
yticks = [i for i in range(0,ceil(y_maximo)+1)] # Lista de ticks para el eje y.
print(y_maximo, yticks)
```

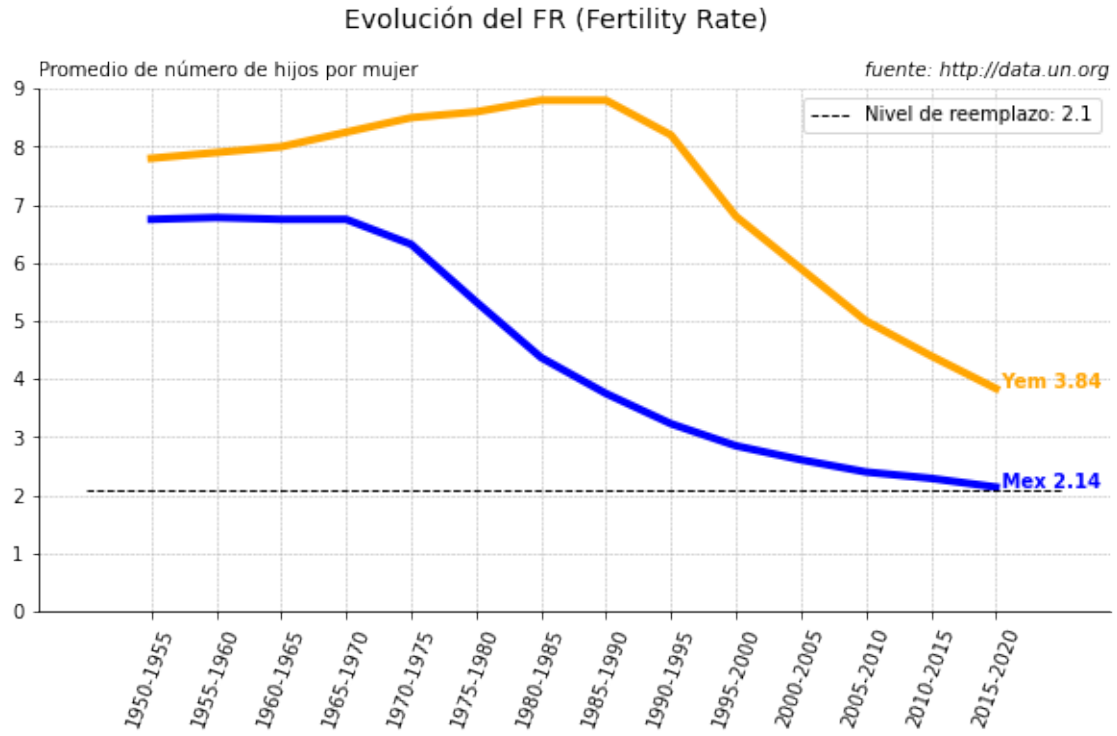
```
8.8 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

1.6.2 Graficación de la nueva información.

Graficar usando el mismo script que se usó en el punto 4.

[32]:
```
fig = plt.figure(figsize=(10,5))
plt.xticks(rotation=70)

# Usamos ahora la información de México y Yemen
mex_color = 'blue'
yem_color = 'orange'
plt.plot(mex['Year(s)'], mex['Value'], lw=4.0, c=mex_color)
plt.plot(yem['Year(s)'], yem['Value'], lw=4.0, c=yem_color)
plt.plot([-1,14],[2.1,2.1], 'k--', lw=1.0, label='Nivel de reemplazo: 2.1')
plt.gca().invert_xaxis()
plt.ylim(0, y_maximo)    # Usamos el FR máximo
plt.yticks(yticks)       # Usamos los yticks calculados antes
plt.grid(ls='--', lw=0.5)

# Información adicional y títulos
plt.title('Promedio de número de hijos por mujer', loc='left', fontsize=10)
plt.title('fuente: http://data.un.org', loc='right', fontstyle='italic',␣
 ↪fontsize=10)
plt.suptitle('Evolución del FR (Fertility Rate)',y = 1.0, fontsize=14)
plt.legend()
```

```
# Agregamos el texto en cada curva
mex_val = mex['Value'].iloc[0]
yem_val = yem['Value'].iloc[0]
plt.text(x = 0, y = mex_val, s = ' Mex {:1.2f}'.format(mex_val), c = mex_color,␣
 ↪weight = 'bold')
plt.text(x = 0, y = yem_val, s = ' Yem {:1.2f}'.format(yem_val), c = yem_color,␣
 ↪weight = 'bold')

# Quitamos la línea derecha y la de arriba del marco de los ejes
ejes = fig.axes
ejes[0].spines['right'].set_visible(False)
ejes[0].spines['top'].set_visible(False)
```



Observamos en esta gráfica que el FR en México y Yemen casi siempre han sido mayores al nivel de reemplazo.

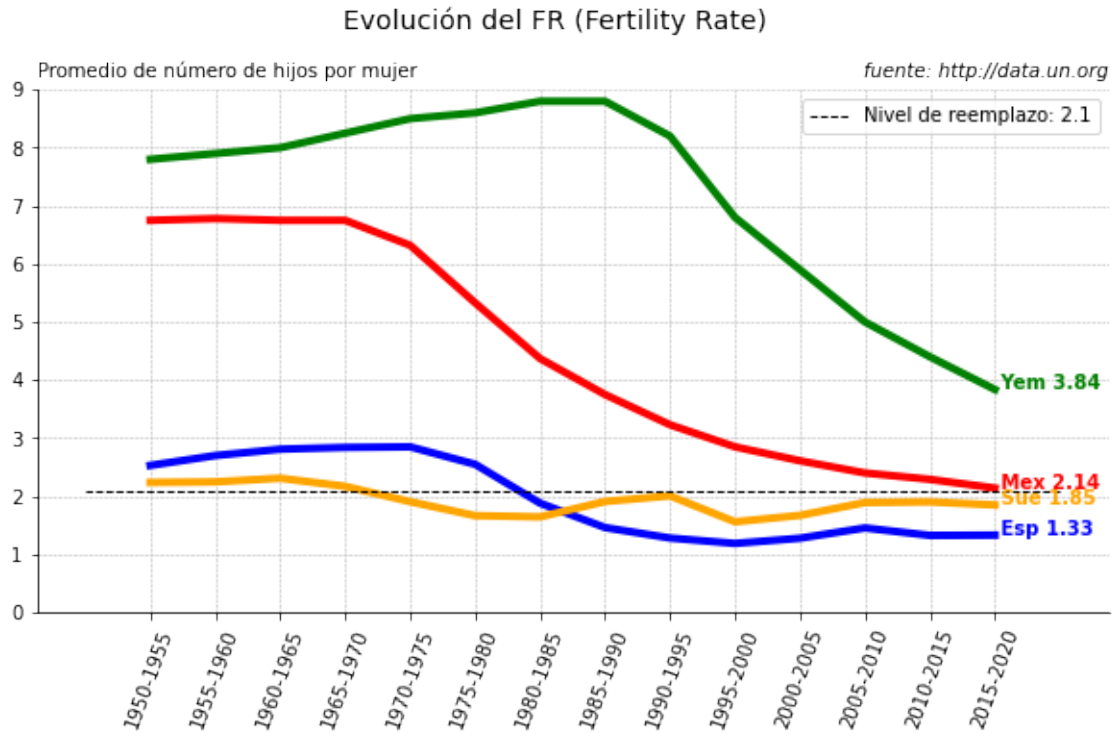
¿Qué más se puede decir de estas curvas con respecto de las de España y Suecia?

1.6.3 Visualización de la información de los cuatro países.

Graficamos los cuatro países juntos para comparar la información.

[33]:
```
fig = plt.figure(figsize=(10,5))
plt.xticks(rotation=70)
```

```
# Definimos todos los colores y quitamos las leyendas
spa_color = 'blue'
swe_color = 'orange'
mex_color = 'red'
yem_color = 'green'
plt.plot(spa['Year(s)'], spa['Value'], lw=4.0, c=spa_color)
plt.plot(swe['Year(s)'], swe['Value'], lw=4.0, c=swe_color)
plt.plot(mex['Year(s)'], mex['Value'], lw=4.0, c=mex_color)
plt.plot(yem['Year(s)'], yem['Value'], lw=4.0, c=yem_color)

plt.plot([-1,14],[2.1,2.1], 'k--', lw=1.0, label='Nivel de reemplazo: 2.1')
plt.gca().invert_xaxis()
plt.ylim(0,y_maximo)      # Límites en el eje y (usamos y_maximo)
plt.yticks(yticks)        # Marcas en el eje y
plt.grid(ls='--', lw=0.5) # Rejilla

# Información adicional y títulos
plt.title('Promedio de número de hijos por mujer', loc='left', fontsize=10)
plt.title('fuente: http://data.un.org', loc='right', fontstyle='italic',␣
 ↪fontsize=10)
plt.suptitle('Evolución del FR (Fertility Rate)',y = 1.0, fontsize=14)
plt.legend()

# Agregamos el texto en cada curva
spa_val = spa['Value'].iloc[0]
swe_val = swe['Value'].iloc[0]
mex_val = mex['Value'].iloc[0]
yem_val = yem['Value'].iloc[0]
plt.text(x = 0, y = spa_val, s = ' Esp {:1.2f}'.format(spa_val), c = spa_color,␣
 ↪weight = 'bold')
plt.text(x = 0, y = swe_val, s = ' Sue {:1.2f}'.format(swe_val), c = swe_color,␣
 ↪weight = 'bold')
plt.text(x = 0, y = mex_val, s = ' Mex {:1.2f}'.format(mex_val), c = mex_color,␣
 ↪weight = 'bold')
plt.text(x = 0, y = yem_val, s = ' Yem {:1.2f}'.format(yem_val), c = yem_color,␣
 ↪weight = 'bold')

# Quitamos la línea derecha y la de arriba del marco de los ejes
ejes = fig.axes
ejes[0].spines['right'].set_visible(False)
ejes[0].spines['top'].set_visible(False)
```



¿Cómo se comparan estas gráficas? ¿Qué se puede pensar de los países que tienen un FR que está por arriba del factor NR? ¿Cómo podrían relacionarse los hechos históricos y geográficos de esos países con esta información? ¿Qué más se observa en estas gráficas?

1.6.4 Visualización de todos los países.

Realizar las gráficas para todos los países de la base de datos.

[34]:
```
# La lista de países se puede obtener usando la función: paises.groups.keys().
# Imprimimos la lista de todos los países:

paises.groups.keys()
```

[34]:
```
dict_keys(['Afghanistan', 'Africa', 'Albania', 'Algeria', 'American Samoa',
'Andorra', 'Angola', 'Anguilla', 'Antigua and Barbuda', 'Argentina', 'Armenia',
'Aruba', 'Asia', 'Australia', 'Australia/New Zealand', 'Austria', 'Azerbaijan',
'Bahamas', 'Bahrain', 'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize',
'Benin', 'Bermuda', 'Bhutan', 'Bolivia (Plurinational State of)', 'Bonaire, Sint
Eustatius and Saba', 'Bosnia and Herzegovina', 'Botswana', 'Brazil', 'British
Virgin Islands', 'Brunei Darussalam', 'Bulgaria', 'Burkina Faso', 'Burundi',
'Cabo Verde', 'Cambodia', 'Cameroon', 'Canada', 'Caribbean', 'Cayman Islands',
'Central African Republic', 'Central America', 'Central Asia', 'Central and
Southern Asia', 'Chad', 'Channel Islands', 'Chile', 'China', 'China, Hong Kong
SAR', 'China, Macao SAR', 'Colombia', 'Comoros', 'Congo', 'Cook Islands', 'Costa
```

Rica', 'Croatia', 'Cuba', 'Curaçao', 'Cyprus', 'Czechia', "Côte d'Ivoire", "Dem. People's Republic of Korea", 'Democratic Republic of the Congo', 'Denmark', 'Djibouti', 'Dominica', 'Dominican Republic', 'Eastern Africa', 'Eastern Asia', 'Eastern Europe', 'Eastern and South-Eastern Asia', 'Ecuador', 'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea', 'Estonia', 'Eswatini', 'Ethiopia', 'Europe', 'Europe and Northern America', 'Falkland Islands (Malvinas)', 'Faroe Islands', 'Fiji', 'Finland', 'France', 'French Guiana', 'French Polynesia', 'Gabon', 'Gambia', 'Geographic regions', 'Georgia', 'Germany', 'Ghana', 'Gibraltar', 'Greece', 'Greenland', 'Grenada', 'Guadeloupe', 'Guam', 'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti', 'High-income countries', 'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India', 'Indonesia', 'Iran (Islamic Republic of)', 'Iraq', 'Ireland', 'Isle of Man', 'Israel', 'Italy', 'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati', 'Kuwait', 'Kyrgyzstan', 'Land-locked Developing Countries (LLDC)', "Lao People's Democratic Republic", 'Latin America and the Caribbean', 'Latvia', 'Least developed countries', 'Lebanon', 'Lesotho', 'Less developed regions', 'Less developed regions, excluding China', 'Less developed regions, excluding least developed countries', 'Liberia', 'Libya', 'Liechtenstein', 'Lithuania', 'Low-income countries', 'Lower-middle-income countries', 'Luxembourg', 'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali', 'Malta', 'Marshall Islands', 'Martinique', 'Mauritania', 'Mauritius', 'Mayotte', 'Melanesia', 'Mexico', 'Micronesia', 'Micronesia (Fed. States of)', 'Middle Africa', 'Middle-income countries', 'Monaco', 'Mongolia', 'Montenegro', 'Montserrat', 'More developed regions', 'Morocco', 'Mozambique', 'Myanmar', 'Namibia', 'Nauru', 'Nepal', 'Netherlands', 'New Caledonia', 'New Zealand', 'Nicaragua', 'Niger', 'Nigeria', 'Niue', 'No income group available', 'North Macedonia', 'Northern Africa', 'Northern Africa and Western Asia', 'Northern America', 'Northern Europe', 'Northern Mariana Islands', 'Norway', 'Oceania', 'Oceania (excluding Australia and New Zealand)', 'Oman', 'Other non-specified areas', 'Pakistan', 'Palau', 'Panama', 'Papua New Guinea', 'Paraguay', 'Peru', 'Philippines', 'Poland', 'Polynesia', 'Portugal', 'Puerto Rico', 'Qatar', 'Republic of Korea', 'Republic of Moldova', 'Romania', 'Russian Federation', 'Rwanda', 'Réunion', 'Saint Helena', 'Saint Kitts and Nevis', 'Saint Lucia', 'Saint Pierre and Miquelon', 'Saint Vincent and the Grenadines', 'Saint-Barthélemy', 'Saint-Martin (French part)', 'Samoa', 'San Marino', 'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia', 'Seychelles', 'Sierra Leone', 'Singapore', 'Sint Maarten (Dutch part)', 'Slovakia', 'Slovenia', 'Small Island Developing States (SIDS)', 'Solomon Islands', 'Somalia', 'South Africa', 'South America', 'South Sudan', 'South-Eastern Asia', 'Southern Africa', 'Southern Asia', 'Southern Europe', 'Spain', 'Sri Lanka', 'State of Palestine', 'Sub-Saharan Africa', 'Sudan', 'Suriname', 'Sustainable Development Goal (SDG) regions', 'Sweden', 'Switzerland', 'Syrian Arab Republic', 'Tajikistan', 'Thailand', 'Timor-Leste', 'Togo', 'Tokelau', 'Tonga', 'Trinidad and Tobago', 'Tunisia', 'Turkey', 'Turkmenistan', 'Turks and Caicos Islands', 'Tuvalu', 'UN development groups', 'Uganda', 'Ukraine', 'United Arab Emirates', 'United Kingdom', 'United Republic of Tanzania', 'United States Virgin Islands', 'United States of America', 'Upper-middle-income countries', 'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela

```
(Bolivarian Republic of)', 'Viet Nam', 'Wallis and Futuna Islands', 'Western
Africa', 'Western Asia', 'Western Europe', 'Western Sahara', 'World', 'World
Bank income groups', 'Yemen', 'Zambia', 'Zimbabwe'])
```

Como se puede observar, la información es amplia. Si observa con cuidado verá que también hay información definida por regiones geográficas y por otros criterios.

Para ver el total podemos hacer lo siguiente:

[35]:
```
pd.DataFrame(paises).index # Primero transformamos el grupo en DataFrame y luego
                           # obtenemos todos sus índices (renglones)
```

[35]:
```
RangeIndex(start=0, stop=286, step=1)
```

Podríamos entonces graficar **286** curvas. ¿Será esta una buena estrategia?

¡Eso haremos!

Para lograr un buen resultado, primero vamos a crear algunas funciones.

Incialización del canvas Primero vamos a definir una función para inicializar el entorno de la gráfica (el canvas).

[36]:
```
def inicializaGrafica(y_maximo, yticks):
    """
    Inicializa algunos parámetros de la figura.

    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)
    plt.xlim(-2,14,1)
    plt.gca().invert_xaxis()
    plt.ylim(0,y_maximo)
    plt.yticks(yticks)      # Marcas en el eje y
    plt.grid(ls='--', lw=0.5) # Rejilla

    # Información adicional y títulos
    plt.title('Promedio de número de hijos por mujer', loc='left', fontsize=10)
    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 = fig.axes
    ejes[0].spines['right'].set_visible(False)
    ejes[0].spines['top'].set_visible(False)
    ejes[0].spines['left'].set_visible(False)
    ejes[0].spines['bottom'].set_visible(False)

    # Modificamos algunos parámetros de los ticks en el eje y
    ejes[0].tick_params(axis='y', width=1, length=25)
```

Función para graficar la info de cualquier país. Ahora definimos una función para graficar la información de cada país.

[37]:
```
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)  # Se obtiene el DataFrame del país
        plt.plot(pais['Year(s)'], pais['Value'], **parametros)

    # Al final de todas las gráficas ponemos la del nivel de reemplazo
    # para que apareza sobre todas ellas y se note mejor (también es posible␣
 ↪usar zorder)
    plt.plot([-1,14],[2.1,2.1], 'k--', lw=1.0, label='Nivel de reemplazo: 2.1')
    plt.legend()
```

Resultado preliminar Calculamos el máximo del valor del FR de todos los países y lo usamos para generar los `yticks`

[38]:
```
# Se obtiene el máximo a partir de la información original
y_maximo = FR['Value'].max()

# Se definen los yticks usando el y_maximo
yticks = [i for i in range(0,ceil(y_maximo)+1)]
print('FR máximo: ', y_maximo)
print('Marcas para el eje y: ', yticks)
```

```
FR máximo:  8.8
Marcas para el eje y:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

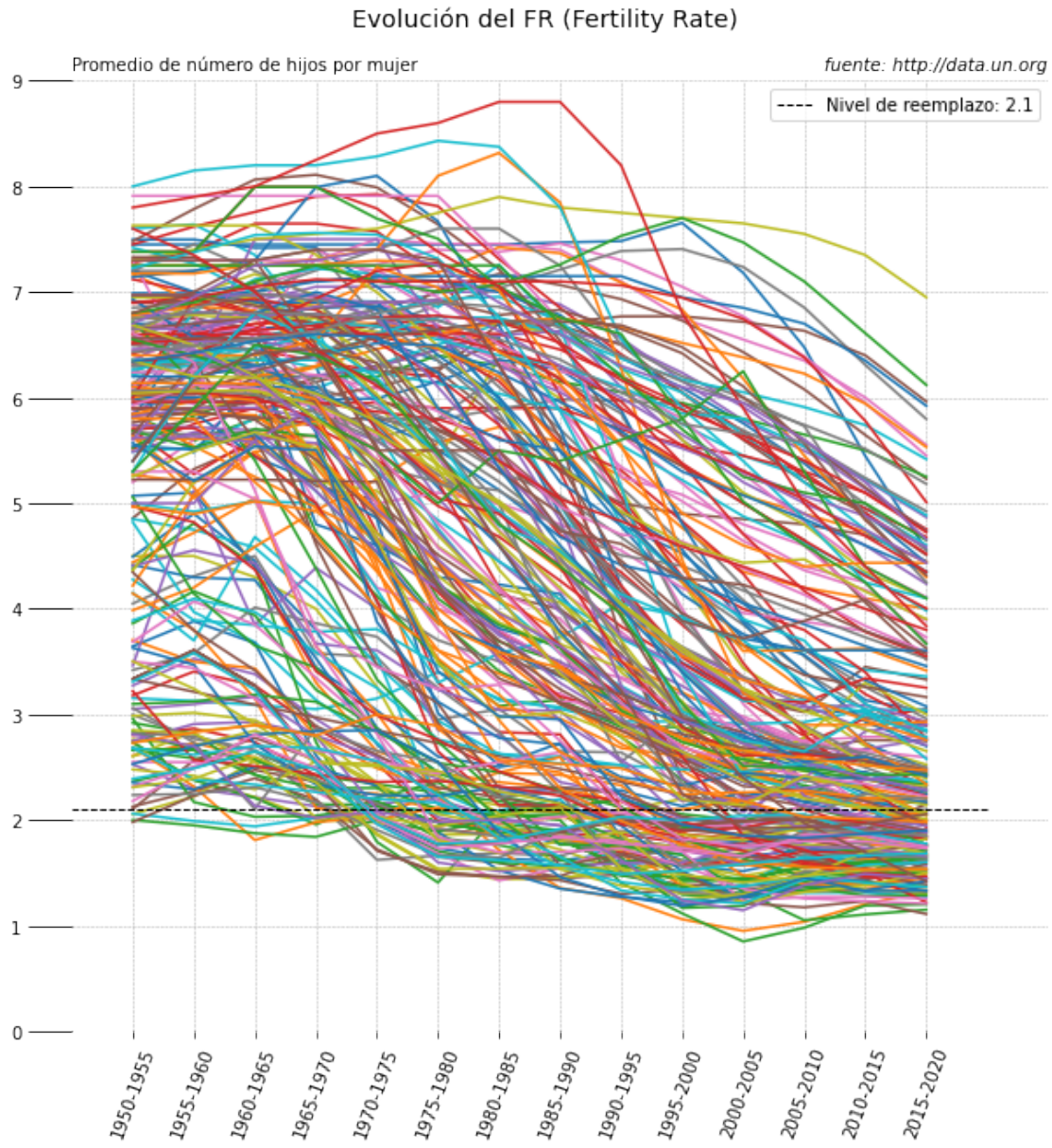
¿Cómo le haríamos para conocer el país o países que tienen el valor máximo de FR?

[39]:
```
FR[FR['Value'] == 8.8]
```

[39]:

	Country or Area	Year(s)	Variant	Value
4010	Yemen	1985-1990	Medium	8.8
4011	Yemen	1980-1985	Medium	8.8

[40]:
```
# Graficamos todos los países.
inicializaGrafica(y_maximo, yticks)
graficaFR(paises)
```



¡Esta gráfica es bastante interesante y colorida! Sin embargo poco útil.

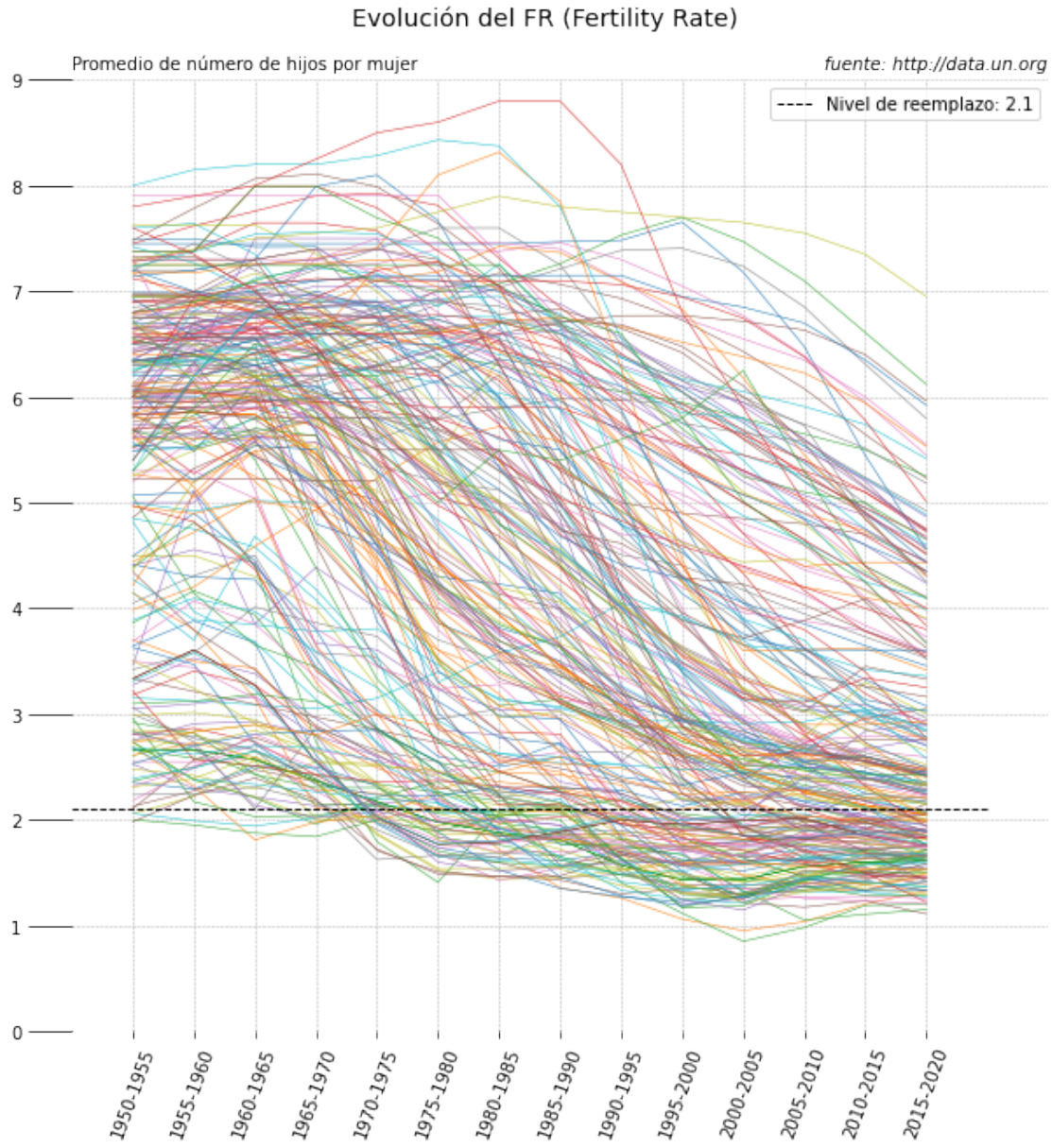
Aunque es posible identificar, si nos fijamos bien, grupos de países que inician con un FR entre 5 y 7, que después bajan conforme pasan las décadas.

Y otro grupo de países entre el 2 y 3 que se matienen al rededor del nivel de reemplazo (2.1).

Como vimos en la celda anterior, Yemen es el país que tiene el máximo FR, durante los años de 1985 a 1990.

Primera mejora. Haremos la misma gráfica, pero con las líneas más delgadas para intentar distinguir algo más.

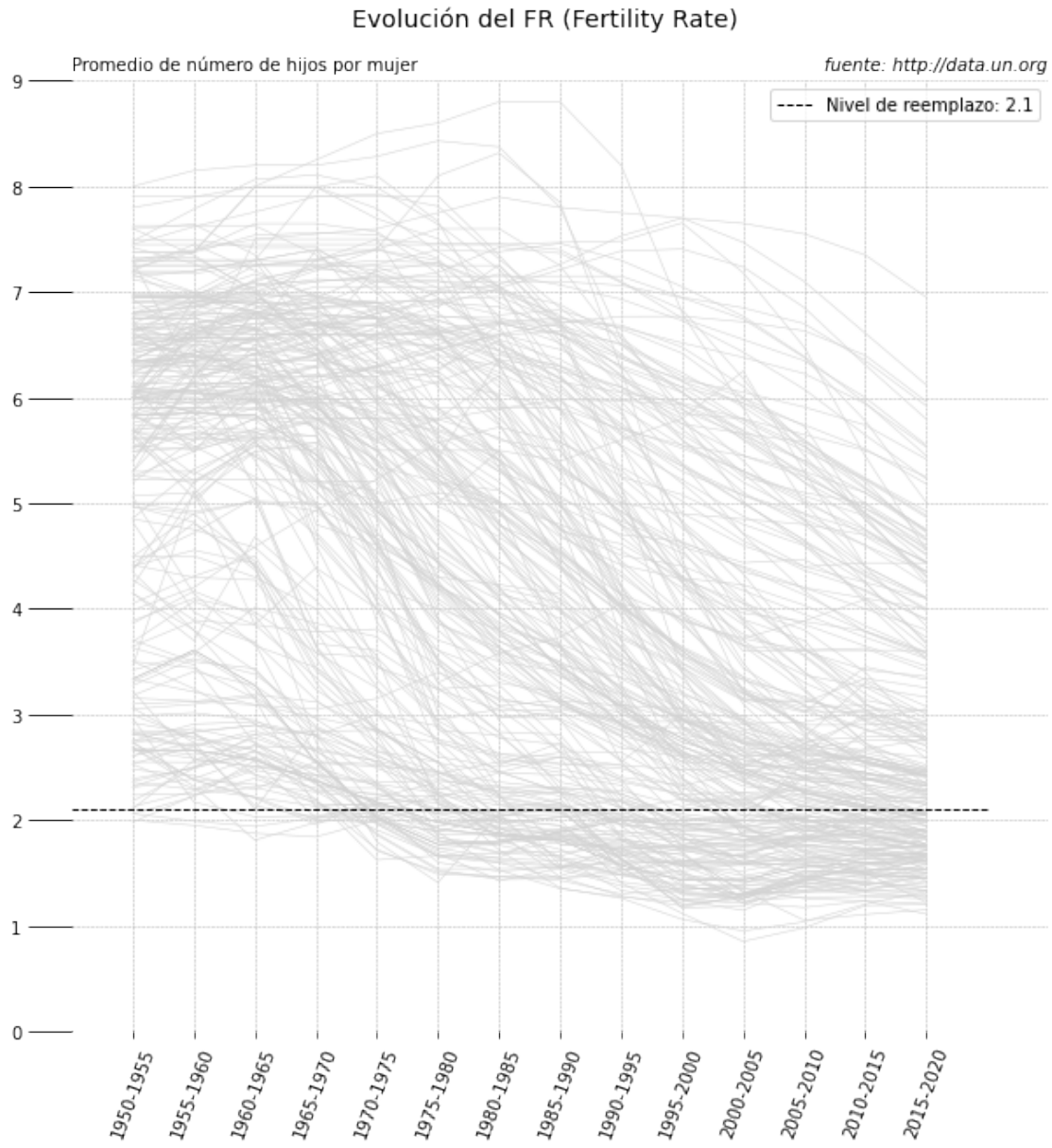
[41]:
```
inicializaGrafica(y_maximo, yticks)
graficaFR(paises, {'lw':0.5})
```



Esta gráfica "desenreda" un poco la información, pero sigue siendo poco útil.

Segunda mejora. Usamos un color muy tenue y el mimos para graficar todos los países.

[42]:
```
inicializaGrafica(y_maximo, yticks)
graficaFR(paises, {'lw':0.5, 'c':'lightgrey'})
```



Esta gráfica es un canvas con toda la información necesaria para analizar cada país por separado. Veamos cómo.

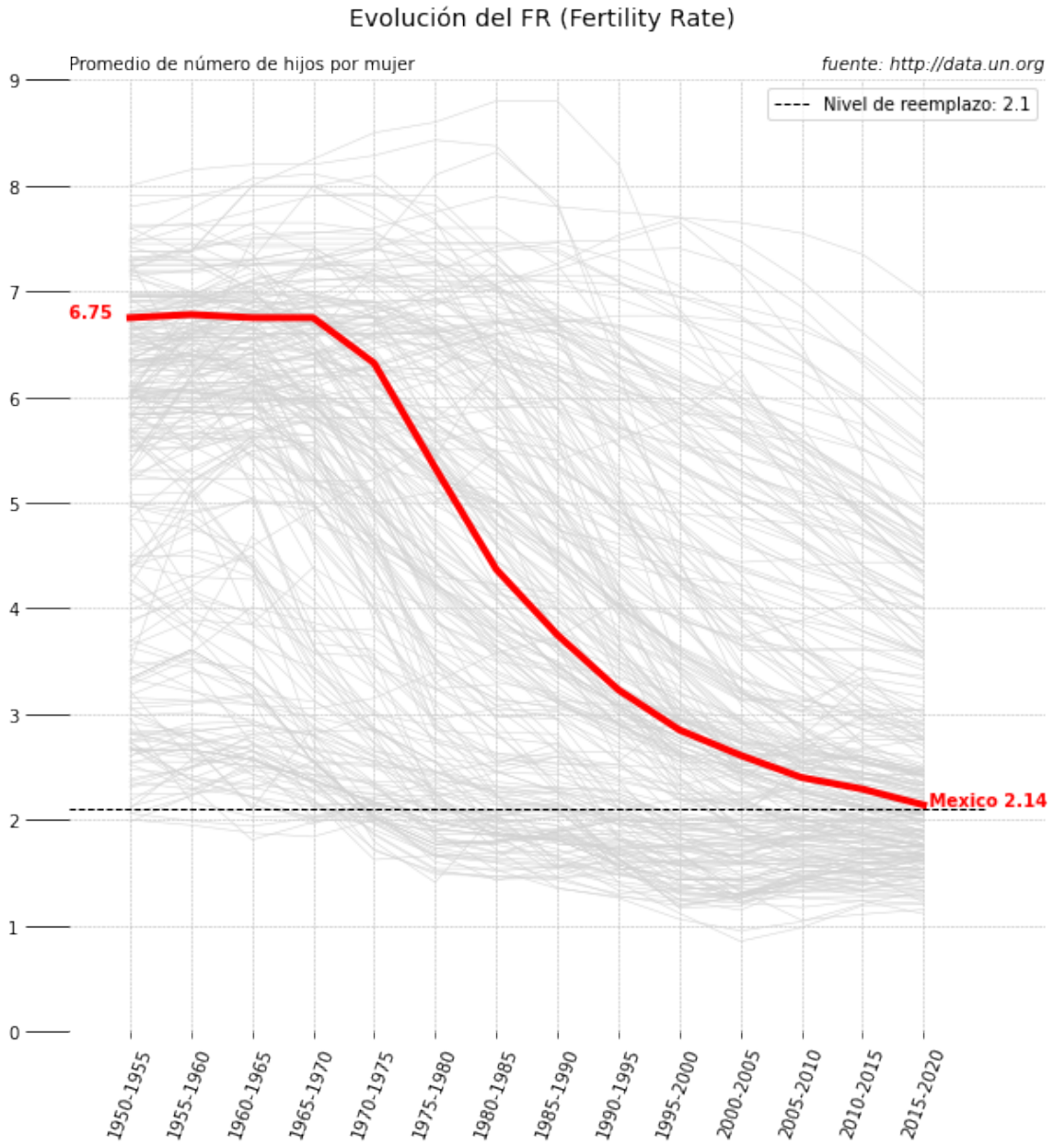
1.7 Resultado final

- Analizaremos los países de Suecia, España, México y Yemen, dentro de las gráficas de todos los países.
- Igualmente se van a agregar algunos otros que sean de interés.

Definimos una función para graficar un solo país con parámetros que permitan realzar la curva

[43]:
```
def graficaFR_Pais(paises, p, parametros={}):
    """
    Realiza la gráfica de un solo país.

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

    parametros : dict
        Parámetros para generar la gráfica.
    """
    pais = paises.get_group(p)

    # Graficamos el país con los parámetros requeridos
    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 FR
    pais_val = pais['Value'].iloc[0]
    plt.text(x = 0, y = pais_val,
             s = ' {} {:1.2f}'.format(p, pais_val),
             c = parametros['c'], weight = 'bold')

    # Ponemos el valor inicial de FR al principio de la curva.
    pais_val = pais['Value'].iloc[-1]
    plt.text(x = 14, y = pais_val,
             s = '{:1.2f} '.format(pais_val),
             c = parametros['c'], weight = 'bold')
```

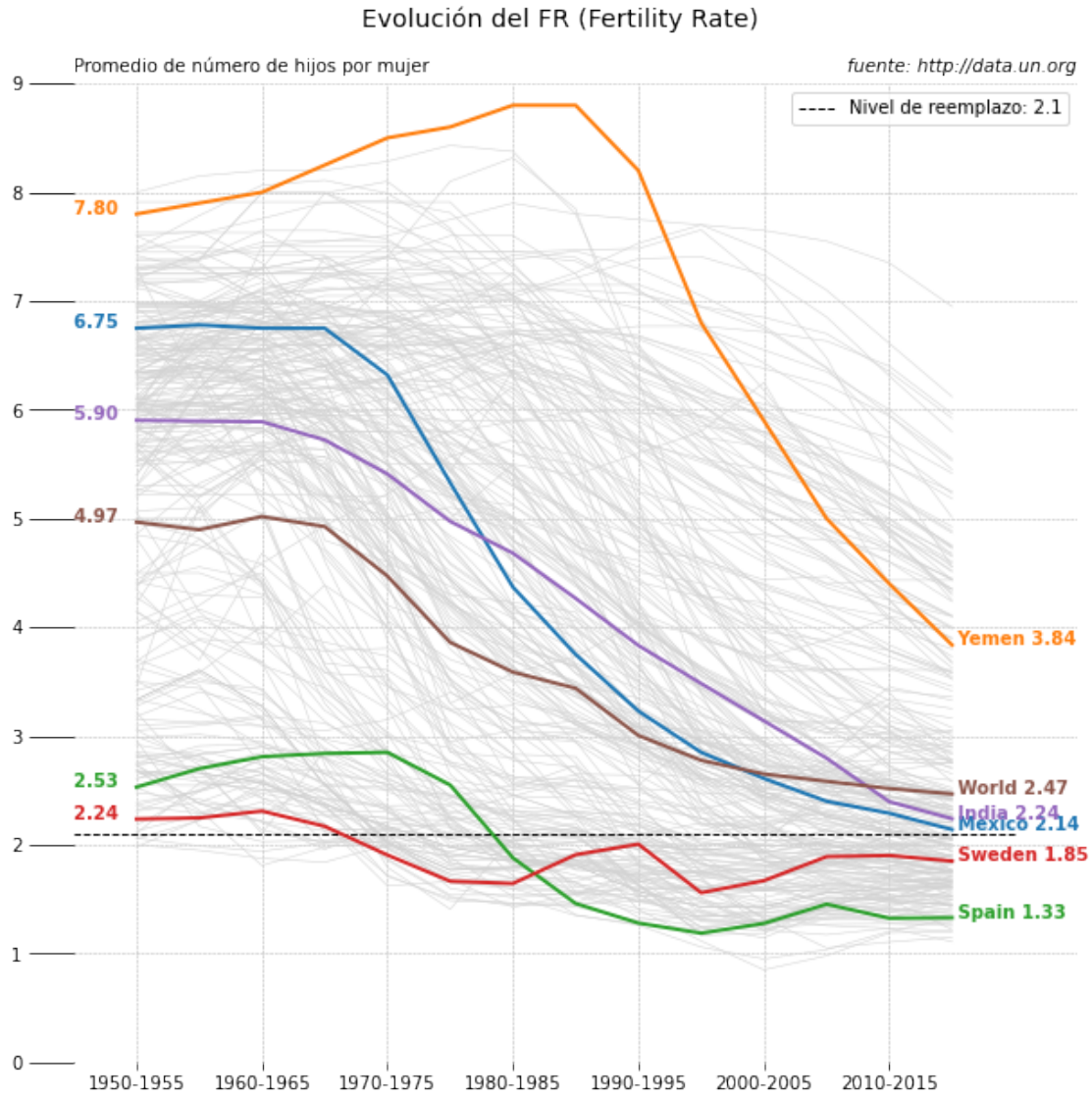
[44]:
```
# Hacemos la gráfica base
inicializaGrafica(y_maximo, yticks)
graficaFR(paises, {'lw':0.5, 'c':'lightgrey'})

# Graficamos para México con parámetros de realce (ancho 4 y color rojo)
par_mex = {'lw':4.0, 'c':'red'}
graficaFR_Pais(paises, 'Mexico', par_mex)
```



Esta gráfica nos da mucho mayor información. Además de tener todo el contexto de los demás países, podemos observar como ha cambiado el número de hijos promedio por mujer en nuestro país, desde los años 50 que tenía un valor de 6.75 y en los años 70s comenzó su declive. Fue en el año de 1974 cuando se instaló la CONAPO (Consejo Nacional de Población) y uno de sus primeros lemas fue *"La familia pequeña vive mejor"* (aún recuerdo esos promocionales de la TV). Es muy probable que este hecho haya impactado en esa disminución de los hijos por familia durante las siguientes décadas. Actualmente el valor es de 2.14, apenas un poco arriba del NR y la tendencia es a la baja. En mi experiencia, al hablar con jóvenes en edad reproductiva, parece que para un porcentaje alto de ellos (no tengo los datos exactos), ya no es un objetivo de vida tener hijos, por lo que la espectativa es que el nivel de hijos pase por abajo del NR en nuestro país en los próximos

años.

[45]:
```
# Hacemos la gráfica base
inicializaGrafica(y_maximo, yticks)
graficaFR(paises, {'lw':0.5, 'c':'lightgrey'})

# Finalmente hacemos la gráfica para varios países para hacer la comparación
# entre ellos.
# NOTA: La base de datos también trae la información del número de hijos
# promedio de todo el mundo ('World') el cual graficamos para comparar.
colores = ['C0', 'C1', 'C2', 'C3', 'C4', 'C5']

paises_l = ['Mexico', 'Yemen', 'Spain', 'Sweden', 'India', 'World']
for p, c in zip(paises_l, colores):
    par = {'lw':2.0, 'c':c}
    graficaFR_Pais(paises, p, par)

# Decidimos dibujar menos xticks para mayor claridad.
plt.xticks([13, 11, 9, 7, 5, 3, 1], rotation=0)

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



¿Puede Usted contar la historia de esta última gráfica?

Responder las preguntas: - ¿Los países ricos están recuperando su FR? - ¿Países en desarrollo como China y Brasil están estabilizando sus poblaciones? - ¿Cómo ha sido la evolución de la población en México? - ¿y en otros países?

[ ]: