**Starter Kernel: Unemployment in EU Countries**

Content

* [Introduction](file:///E:\ONLINE%20WORKING\Data_Visualization_Task\starter-kernel-unemployment-in-eu-countries.html#1)
* [Data Preparation](file:///E:\ONLINE%20WORKING\Data_Visualization_Task\starter-kernel-unemployment-in-eu-countries.html#2)
* [Extract country, age, unit, sex, s\_adj data](file:///E:\ONLINE%20WORKING\Data_Visualization_Task\starter-kernel-unemployment-in-eu-countries.html#3)
* [Filter columns for analysis](file:///E:\ONLINE%20WORKING\Data_Visualization_Task\starter-kernel-unemployment-in-eu-countries.html#4)
* [Data Visualization](file:///E:\ONLINE%20WORKING\Data_Visualization_Task\starter-kernel-unemployment-in-eu-countries.html#5)

**Introduction**

We analyze here the unemployment in EU in recent years. The full data (~1983 - 2020) is grouped on country, sex, age interval (<25, 25-74, total).

The dataset used is migrated from [EU Open Data Portal](https://data.europa.eu/euodp/en/data/dataset).

I also use a dataset with ISO codes for countries.

In this Kernel, we show how we can extract information from this compact dataset and mix it with countries data.



**Data preparation**

In [1]:

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **os**

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

**import** **datetime** **as** **dt**

**import** **folium**

**from** **folium.plugins** **import** HeatMap, HeatMapWithTime

%**matplotlib** inline

**import** **warnings**

warnings.filterwarnings("ignore")

The unemployment data is in TSV (tab separated format). We use read\_csv with \t separator for reading the data.

In [2]:

data\_df = pd.read\_csv(os.path.join("/kaggle", "input", "unemployment-in-european-union", "une\_rt\_m.tsv"), sep='**\t**')

The countries codes data is provided in \*.csv format.

In [3]:

country\_codes\_df = pd.read\_csv(os.path.join("/kaggle", "input", "iso-country-codes-global", "wikipedia-iso-country-codes.csv"))

Let's first glimpse the data.

In [4]:

data\_df.shape

Out[4]:

(2131, 452)

In [5]:

data\_df.head()

Out[5]:

|  | **s\_adj,age,unit,sex,geo\time** | **2020M07** | **2020M06** | **2020M05** | **2020M04** | **2020M03** | **2020M02** | **2020M01** | **2019M12** | **2019M11** | **...** | **1983M10** | **1983M09** | **1983M08** | **1983M07** | **1983M06** | **1983M05** | **1983M04** | **1983M03** | **1983M02** | **1983M01** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NSA,TOTAL,PC\_ACT,F,AT | 6.0 | 6.2 | 5.2 | 4.9 | 4.7 | 4.1 | 4.4 | 4.0 | 3.9 | ... | : | : | : | : | : | : | : | : | : | : |
| **1** | NSA,TOTAL,PC\_ACT,F,BE | 5.4 | 5.0 | 5.0 | 5.2 | 5.0 | 4.9 | 4.8 | 4.6 | 4.5 | ... | : | : | : | : | : | : | : | : | : | : |
| **2** | NSA,TOTAL,PC\_ACT,F,BG | 3.9 | 3.9 | 4.2 | 4.7 | 4.1 | 4.2 | 4.3 | 3.9 | 3.8 | ... | : | : | : | : | : | : | : | : | : | : |
| **3** | NSA,TOTAL,PC\_ACT,F,CH | : | 4.4 | 4.3 | 4.4 | 4.6 | 4.6 | 4.5 | 4.1 | 4.0 | ... | : | : | : | : | : | : | : | : | : | : |
| **4** | NSA,TOTAL,PC\_ACT,F,CY | 7.2 | 6.6 | 6.4 | 6.8 | 7.1 | 7.5 | 7.9 | 7.5 | 7.7 | ... | : | : | : | : | : | : | : | : | : | : |

5 rows × 452 columns

We look as well to the data columns.

In [6]:

data\_df.columns

Out[6]:

Index(['s\_adj,age,unit,sex,geo\time', '2020M07 ', '2020M06 ', '2020M05 ',

'2020M04 ', '2020M03 ', '2020M02 ', '2020M01 ', '2019M12 ', '2019M11 ',

...

'1983M10 ', '1983M09 ', '1983M08 ', '1983M07 ', '1983M06 ', '1983M05 ',

'1983M04 ', '1983M03 ', '1983M02 ', '1983M01 '],

dtype='object', length=452)

Typically time series data in EU Open Data Portal are stored with separate columns for each time value. We will have to pivot these columns during our data processing.

Let's also look to the country codes.

In [7]:

country\_codes\_df.head()

Out[7]:

|  | **English short name lower case** | **Alpha-2 code** | **Alpha-3 code** | **Numeric code** | **ISO 3166-2** |
| --- | --- | --- | --- | --- | --- |
| **0** | Zimbabwe | ZW | ZWE | 716 | ISO 3166-2:ZW |
| **1** | Zambia | ZM | ZMB | 894 | ISO 3166-2:ZM |
| **2** | Yemen | YE | YEM | 887 | ISO 3166-2:YE |
| **3** | Western Sahara | EH | ESH | 732 | ISO 3166-2:EH |
| **4** | Wallis and Futuna | WF | WLF | 876 | ISO 3166-2:WF |

Let's replace the column names.

In [8]:

country\_codes\_df.columns = ['country', 'C2', 'C3', 'numeric', 'iso']

In [9]:

country\_codes\_df.head()

Out[9]:

|  | **country** | **C2** | **C3** | **numeric** | **iso** |
| --- | --- | --- | --- | --- | --- |
| **0** | Zimbabwe | ZW | ZWE | 716 | ISO 3166-2:ZW |
| **1** | Zambia | ZM | ZMB | 894 | ISO 3166-2:ZM |
| **2** | Yemen | YE | YEM | 887 | ISO 3166-2:YE |
| **3** | Western Sahara | EH | ESH | 732 | ISO 3166-2:EH |
| **4** | Wallis and Futuna | WF | WLF | 876 | ISO 3166-2:WF |

**Extract country, age, unit, sex, s\_adj data**

We separate from the first column the various data attributes (country, age, unit, sex, s\_adj) by spliting the complex data in individual values and we store them on separate columns.

In [10]:

data\_df['C2'] = data\_df['s\_adj,age,unit,sex,geo**\\**time'].apply(**lambda** x: x.split(",")[-1])

data\_df['age'] = data\_df['s\_adj,age,unit,sex,geo**\\**time'].apply(**lambda** x: x.split(",")[1])

data\_df['unit'] = data\_df['s\_adj,age,unit,sex,geo**\\**time'].apply(**lambda** x: x.split(",")[2])

data\_df['sex'] = data\_df['s\_adj,age,unit,sex,geo**\\**time'].apply(**lambda** x: x.split(",")[3])

data\_df['s\_adj'] = data\_df['s\_adj,age,unit,sex,geo**\\**time'].apply(**lambda** x: x.split(",")[0])

Let's check again the transformed data.

In [11]:

data\_df.head()

Out[11]:

|  | **s\_adj,age,unit,sex,geo\time** | **2020M07** | **2020M06** | **2020M05** | **2020M04** | **2020M03** | **2020M02** | **2020M01** | **2019M12** | **2019M11** | **...** | **1983M05** | **1983M04** | **1983M03** | **1983M02** | **1983M01** | **C2** | **age** | **unit** | **sex** | **s\_adj** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NSA,TOTAL,PC\_ACT,F,AT | 6.0 | 6.2 | 5.2 | 4.9 | 4.7 | 4.1 | 4.4 | 4.0 | 3.9 | ... | : | : | : | : | : | AT | TOTAL | PC\_ACT | F | NSA |
| **1** | NSA,TOTAL,PC\_ACT,F,BE | 5.4 | 5.0 | 5.0 | 5.2 | 5.0 | 4.9 | 4.8 | 4.6 | 4.5 | ... | : | : | : | : | : | BE | TOTAL | PC\_ACT | F | NSA |
| **2** | NSA,TOTAL,PC\_ACT,F,BG | 3.9 | 3.9 | 4.2 | 4.7 | 4.1 | 4.2 | 4.3 | 3.9 | 3.8 | ... | : | : | : | : | : | BG | TOTAL | PC\_ACT | F | NSA |
| **3** | NSA,TOTAL,PC\_ACT,F,CH | : | 4.4 | 4.3 | 4.4 | 4.6 | 4.6 | 4.5 | 4.1 | 4.0 | ... | : | : | : | : | : | CH | TOTAL | PC\_ACT | F | NSA |
| **4** | NSA,TOTAL,PC\_ACT,F,CY | 7.2 | 6.6 | 6.4 | 6.8 | 7.1 | 7.5 | 7.9 | 7.5 | 7.7 | ... | : | : | : | : | : | CY | TOTAL | PC\_ACT | F | NSA |

5 rows × 457 columns

In [12]:

print(f"countries:**\n{**list(data\_df.C2.unique())**}**")

countries:

['AT', 'BE', 'BG', 'CH', 'CY', 'CZ', 'DE', 'DK', 'EA', 'EA18', 'EA19', 'EE', 'EL', 'ES', 'EU25', 'EU27\_2007', 'EU27\_2020', 'EU28', 'FI', 'FR', 'HR', 'HU', 'IE', 'IS', 'IT', 'JP', 'LT', 'LU', 'LV', 'MT', 'NL', 'NO', 'PL', 'PT', 'RO', 'SE', 'SI', 'SK', 'UK', 'US', 'TR']

In [13]:

print(f"sex:**\n{**list(data\_df.sex.unique())**}**")

sex:

['F', 'M', 'T']

In [14]:

print(f"age intervals:**\n{**list(data\_df.age.unique())**}**")

age intervals:

['TOTAL', 'Y25-74', 'Y\_LT25']

In [15]:

print(f"unit:**\n{**list(data\_df.unit.unique())**}**")

unit:

['PC\_ACT', 'THS\_PER']

In [16]:

print(f"s\_adj:**\n{**list(data\_df.s\_adj.unique())**}**")

s\_adj:

['NSA', 'SA', 'TC']

**Filter columns for analysis**

We select only a part of columns for further analysis.

More specificaly, we are selecting country code (2 letters), age, unit, sex, s\_adj and months from Jan 2015 to June 2020.

In [17]:

selected\_cols = ['C2','age','unit','sex', 's\_adj',

'2020M07 ', '2020M06 ', '2020M05 ', '2020M04 ','2020M03 ','2020M02 ','2020M01 ',

'2019M12 ','2019M11 ','2019M10 ','2019M09 ','2019M08 ','2019M07 ',

'2019M06 ','2019M05 ','2019M04 ','2019M03 ','2019M02 ','2019M01 ',

'2018M12 ','2018M11 ','2018M10 ','2018M09 ','2018M08 ','2018M07 ',

'2018M06 ','2018M05 ','2018M04 ','2018M03 ','2018M02 ','2018M01 ',

'2017M12 ','2017M11 ','2017M10 ','2017M09 ','2017M08 ','2017M07 ',

'2017M06 ','2017M05 ','2017M04 ','2017M03 ','2017M02 ','2017M01 ',

'2016M12 ','2016M11 ','2016M10 ','2016M09 ','2016M08 ','2016M07 ',

'2016M06 ','2016M05 ','2016M04 ','2016M03 ','2016M02 ','2016M01 ',

'2015M12 ','2015M11 ','2015M10 ','2015M09 ','2015M08 ','2015M07 ',

'2015M06 ','2015M05 ','2015M04 ','2015M03 ','2015M02 ','2015M01 ']

In [18]:

data\_sel\_df = data\_df[selected\_cols]

Let's also merge with the country codes.

In [19]:

data\_sel\_df = data\_sel\_df.merge(country\_codes\_df, on="C2")

In [20]:

data\_sel\_df.head()

Out[20]:

|  | **C2** | **age** | **unit** | **sex** | **s\_adj** | **2020M07** | **2020M06** | **2020M05** | **2020M04** | **2020M03** | **...** | **2015M06** | **2015M05** | **2015M04** | **2015M03** | **2015M02** | **2015M01** | **country** | **C3** | **numeric** | **iso** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | AT | TOTAL | PC\_ACT | F | NSA | 6.0 | 6.2 | 5.2 | 4.9 | 4.7 | ... | 5.2 | 5.6 | 5.4 | 5.2 | 5.6 | 5.2 | Austria | AUT | 40 | ISO 3166-2:AT |
| **1** | AT | TOTAL | PC\_ACT | M | NSA | 5.5 | 6.2 | 6.1 | 5.2 | 5.0 | ... | 6.0 | 6.1 | 6.6 | 6.6 | 6.2 | 6.0 | Austria | AUT | 40 | ISO 3166-2:AT |
| **2** | AT | TOTAL | PC\_ACT | T | NSA | 5.7 | 6.2 | 5.7 | 5.1 | 4.9 | ... | 5.6 | 5.8 | 6.0 | 5.9 | 5.9 | 5.6 | Austria | AUT | 40 | ISO 3166-2:AT |
| **3** | AT | TOTAL | THS\_PER | F | NSA | 129 | 132 | 108 | 100 | 97 | ... | 106 | 114 | 109 | 105 | 113 | 106 | Austria | AUT | 40 | ISO 3166-2:AT |
| **4** | AT | TOTAL | THS\_PER | M | NSA | 134 | 149 | 144 | 121 | 118 | ... | 141 | 140 | 152 | 151 | 141 | 136 | Austria | AUT | 40 | ISO 3166-2:AT |

5 rows × 76 columns

Let's look to the data we selected.

In [21]:

print(f"selected data shape: **{**data\_sel\_df.shape**}**")

selected data shape: (1645, 76)

**Pivot time series data using melt**

We are using melt to pivot time series columns. Now for each value of time series we introduce additional rows.

In [22]:

data\_tr\_df = data\_sel\_df.melt(id\_vars=["country", "age", "unit", "sex", "s\_adj", "C2", "C3", "numeric", "iso"],

var\_name="Date",

value\_name="Value")

Let's check the resulted data structure.

In [23]:

data\_tr\_df.head()

Out[23]:

|  | **country** | **age** | **unit** | **sex** | **s\_adj** | **C2** | **C3** | **numeric** | **iso** | **Date** | **Value** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Austria | TOTAL | PC\_ACT | F | NSA | AT | AUT | 40 | ISO 3166-2:AT | 2020M07 | 6.0 |
| **1** | Austria | TOTAL | PC\_ACT | M | NSA | AT | AUT | 40 | ISO 3166-2:AT | 2020M07 | 5.5 |
| **2** | Austria | TOTAL | PC\_ACT | T | NSA | AT | AUT | 40 | ISO 3166-2:AT | 2020M07 | 5.7 |
| **3** | Austria | TOTAL | THS\_PER | F | NSA | AT | AUT | 40 | ISO 3166-2:AT | 2020M07 | 129 |
| **4** | Austria | TOTAL | THS\_PER | M | NSA | AT | AUT | 40 | ISO 3166-2:AT | 2020M07 | 134 |

In [24]:

print(f"new data shape: **{**data\_tr\_df.shape**}**")

new data shape: (110215, 11)

**Clean the time series data**

We do some cleaning on the time series data. We simply filter non-digit data and trim spaces.

In [25]:

**import** **re**

data\_tr\_df['Value'] = data\_tr\_df['Value'].apply(**lambda** x: re.sub(r"[a-zA-Z: ]", "", x))

data\_tr\_df['Value'] = data\_tr\_df['Value'].apply(**lambda** x: x.replace(" ",""))

data\_tr\_df = data\_tr\_df.loc[~(data\_tr\_df.Value=="")]

data\_tr\_df['Value'] = data\_tr\_df['Value'].apply(**lambda** x: float(x))

Let's check the results.

In [26]:

print(f"distinct values: **{**len(list(data\_tr\_df['Value'].unique()))**}**")

print(f"samples values: **{**data\_tr\_df['Value'].unique()**}**")

distinct values: 3948

samples values: [6.000e+00 5.500e+00 5.700e+00 ... 3.346e+03 6.243e+03 3.259e+03]

**Data visualization**

We look to total values for age interval 25-74.

In [27]:

total\_y25\_74\_df = data\_tr\_df.loc[(data\_tr\_df.age=='Y25-74')&(data\_tr\_df.unit=='PC\_ACT')&(data\_tr\_df.sex=='T')&(data\_tr\_df.s\_adj=='TC')]

In [28]:

**def** plot\_time\_variation(df, y='Value', size=1, is\_log=**False**, title=""):

f, ax = plt.subplots(1,1, figsize=(4\*size,3\*size))

countries = list(df.country.unique())

**for** country **in** countries:

df\_ = df[(df['country']==country)]

g = sns.lineplot(x="Date", y=y, data=df\_, label=country)

ax.text(max(df\_['Date']), (df\_.loc[df\_['Date']==max(df\_['Date']), y]), str(country))

plt.xticks(rotation=90)

plt.title(f'Total unemployment, **{**title**}**, grouped by country')

ax.text(max(df\_['Date']), (df\_.loc[df\_['Date']==max(df\_['Date']), y]), str(country))

plt.legend(loc="upper left", bbox\_to\_anchor=(1,1))

**if**(is\_log):

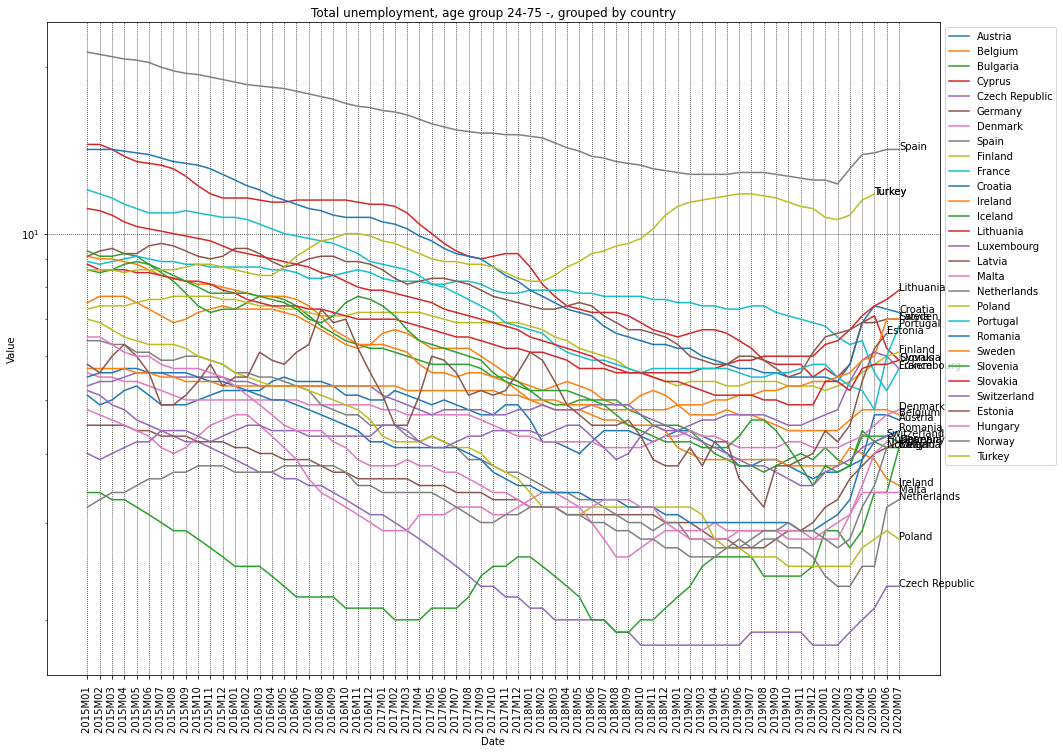
ax.set(yscale="log")

ax.grid(color='black', linestyle='dotted', linewidth=0.75)

plt.show()

In [29]:

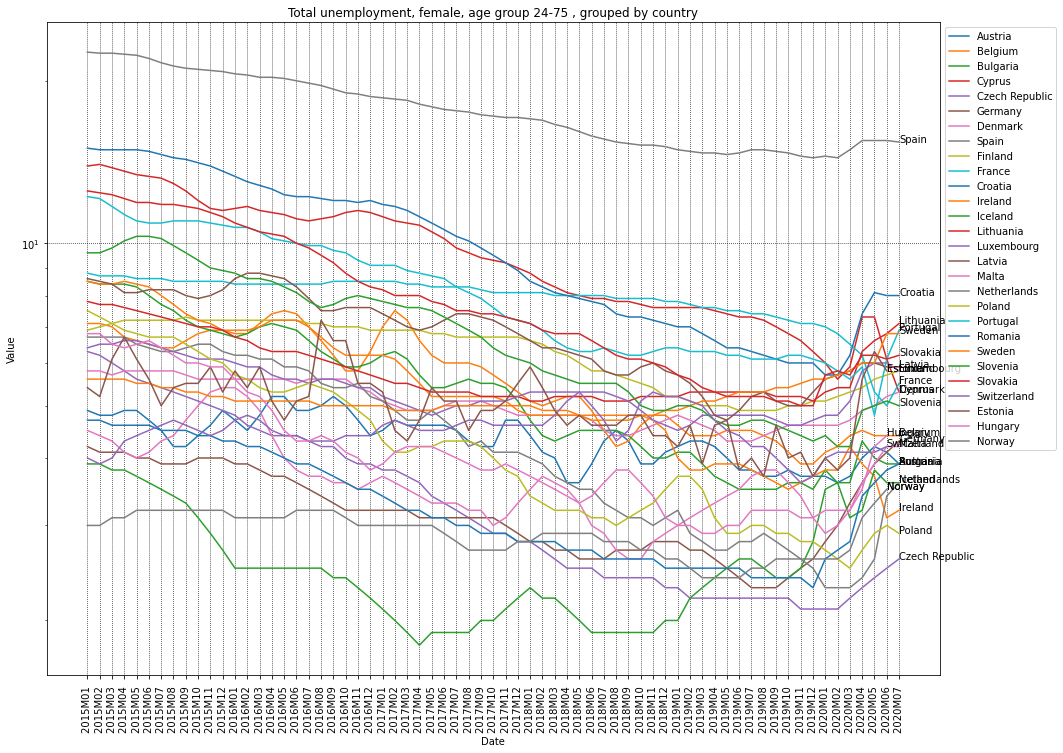
plot\_time\_variation(total\_y25\_74\_df, size=4, is\_log=**True**, title = "age group 24-75 -")



In [30]:

total\_F\_y25\_74\_df = data\_tr\_df.loc[(data\_tr\_df.age=='Y25-74')&(data\_tr\_df.unit=='PC\_ACT')&(data\_tr\_df.sex=='F')&(data\_tr\_df.s\_adj=='TC')]

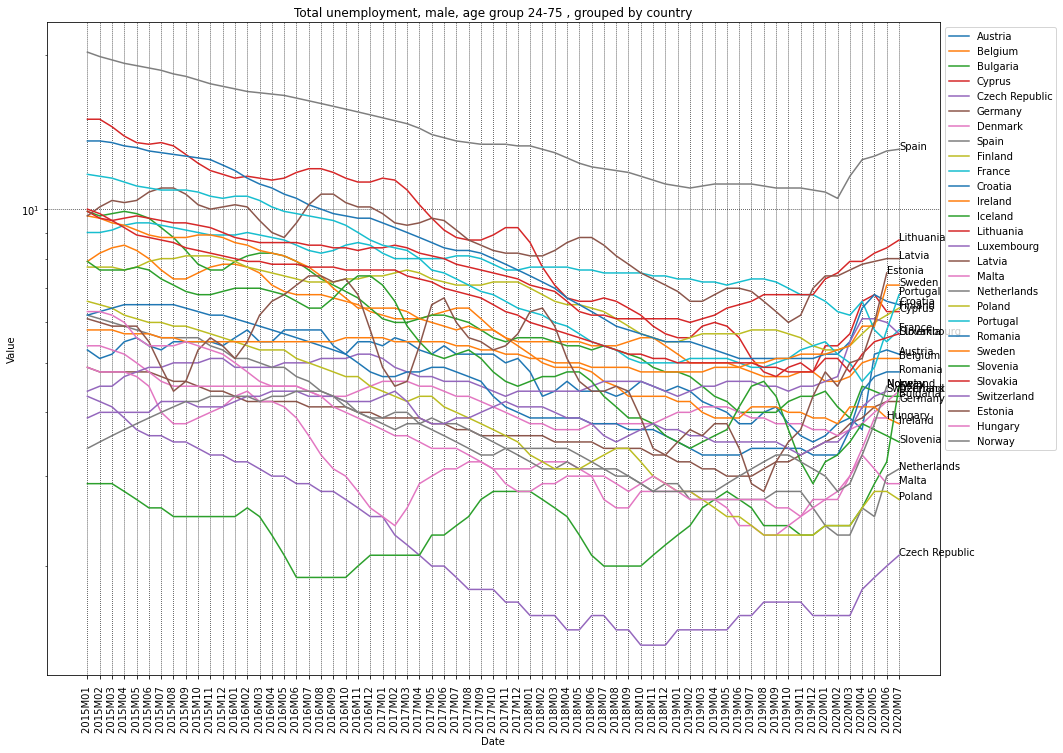
plot\_time\_variation(total\_F\_y25\_74\_df, size=4, is\_log=**True**, title = "female, age group 24-75 ")



In [31]:

total\_M\_y25\_74\_df = data\_tr\_df.loc[(data\_tr\_df.age=='Y25-74')&(data\_tr\_df.unit=='PC\_ACT')&(data\_tr\_df.sex=='M')&(data\_tr\_df.s\_adj=='TC')]

plot\_time\_variation(total\_M\_y25\_74\_df, size=4, is\_log=**True**, title = "male, age group 24-75 ")



In [32]:

total\_M\_y25\_df = data\_tr\_df.loc[(data\_tr\_df.age=='Y\_LT25')&(data\_tr\_df.unit=='PC\_ACT')&(data\_tr\_df.sex=='M')&(data\_tr\_df.s\_adj=='TC')]

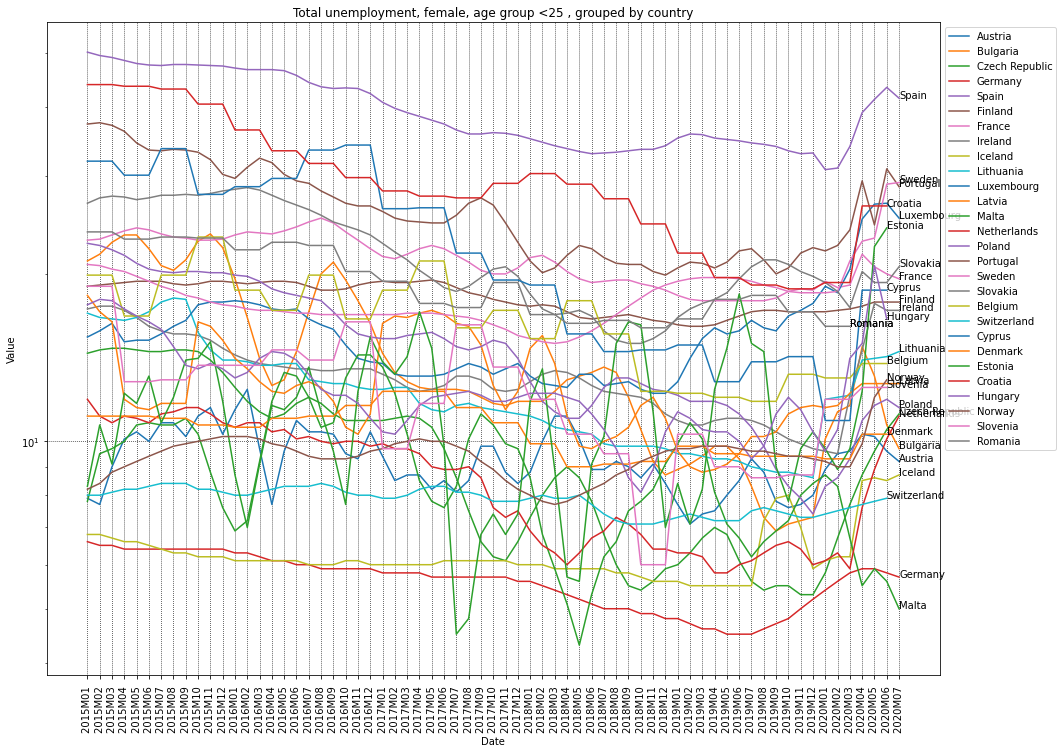
plot\_time\_variation(total\_M\_y25\_df, size=4, is\_log=**True**, title = "male, age group <25 ")



In [33]:

total\_F\_y25\_df = data\_tr\_df.loc[(data\_tr\_df.age=='Y\_LT25')&(data\_tr\_df.unit=='PC\_ACT')&(data\_tr\_df.sex=='F')&(data\_tr\_df.s\_adj=='TC')]

plot\_time\_variation(total\_F\_y25\_df, size=4, is\_log=**True**, title = "female, age group <25 ")



**Compare age group, sex / country**

In [34]:

**def** plot\_time\_variation\_age\_sex(data\_tr\_df, y='Value', country="Netherlands"):

c\_df = data\_tr\_df.loc[(data\_tr\_df.country==country)&(data\_tr\_df.unit=='PC\_ACT')&(data\_tr\_df.s\_adj=='TC')]

f, ax = plt.subplots(1,1, figsize=(16,12))

sns.lineplot(x="Date", y=y, data=c\_df.loc[(c\_df.age=='Y\_LT25')&(c\_df.sex=='F')], label="Female, <25y")

sns.lineplot(x="Date", y=y, data=c\_df.loc[(c\_df.age=='Y\_LT25')&(c\_df.sex=='M')], label="Male, <25y")

sns.lineplot(x="Date", y=y, data=c\_df.loc[(c\_df.age=='Y25-74')&(c\_df.sex=='F')], label="Female, 25-74y")

sns.lineplot(x="Date", y=y, data=c\_df.loc[(c\_df.age=='Y25-74')&(c\_df.sex=='M')], label="Male, <25-74y")

plt.xticks(rotation=90)

plt.title(f'Total unemployment in **{**country**}**, grouped by age & sex')

plt.legend(loc="upper left", bbox\_to\_anchor=(1,1))

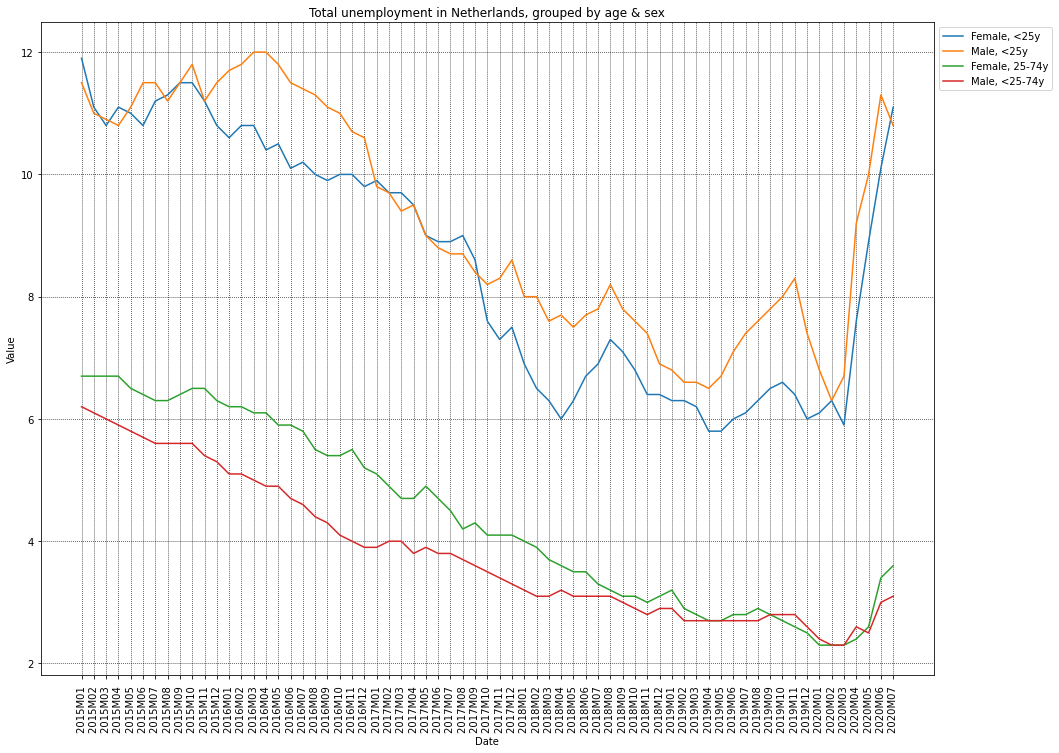
ax.grid(color='black', linestyle='dotted', linewidth=0.75)

plt.show()

Let's look to some specific countries evolution.

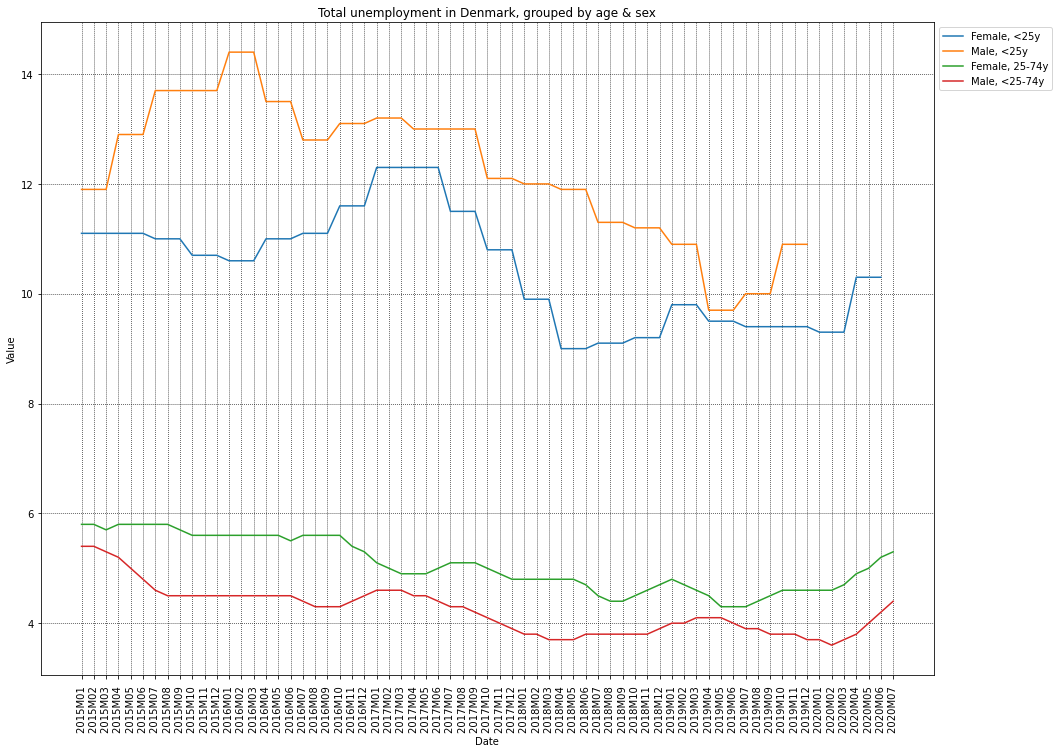
In [35]:

plot\_time\_variation\_age\_sex(data\_tr\_df,country="Netherlands")



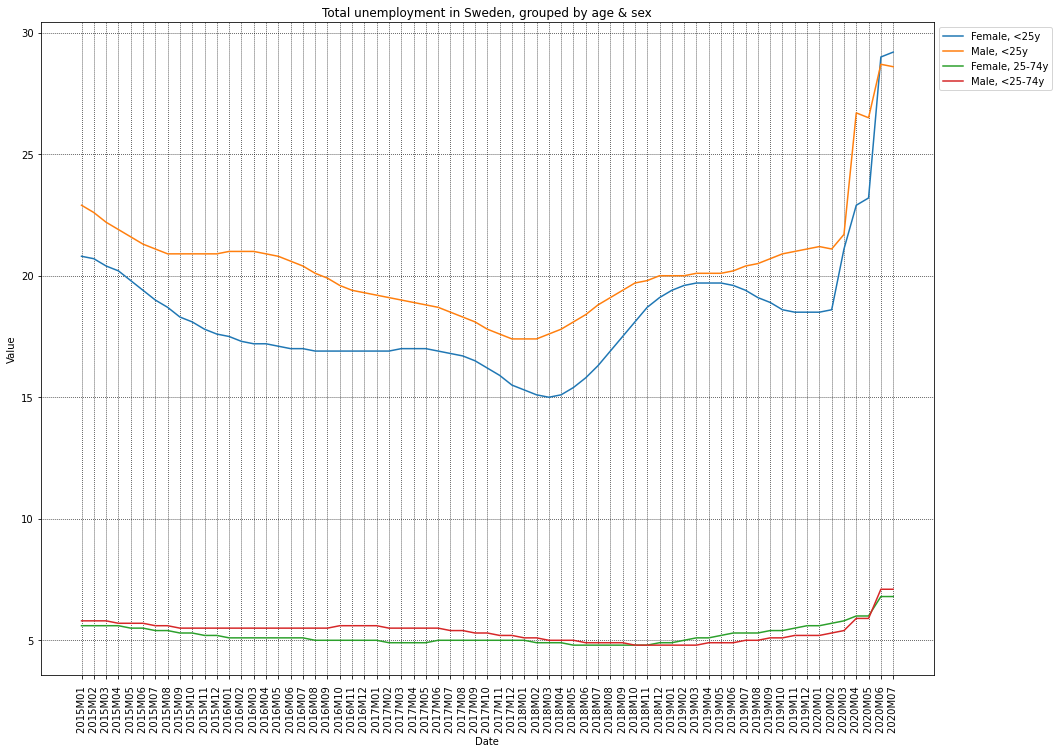
In [36]:

plot\_time\_variation\_age\_sex(data\_tr\_df,country="Denmark")



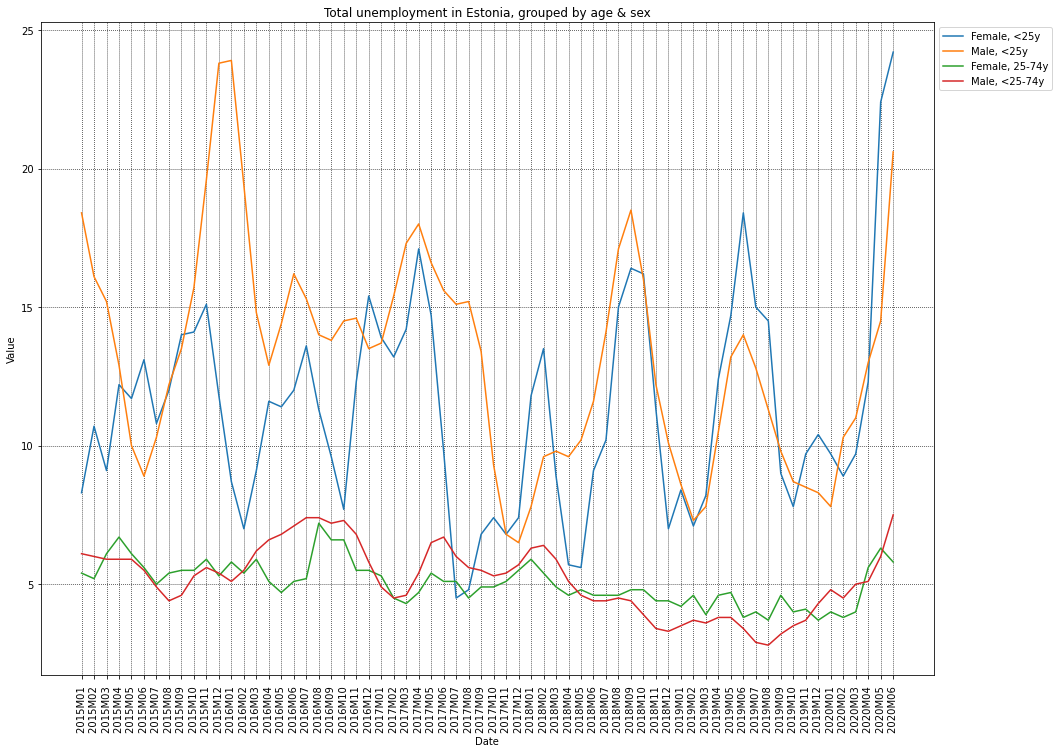
In [37]:

plot\_time\_variation\_age\_sex(data\_tr\_df,country="Sweden")



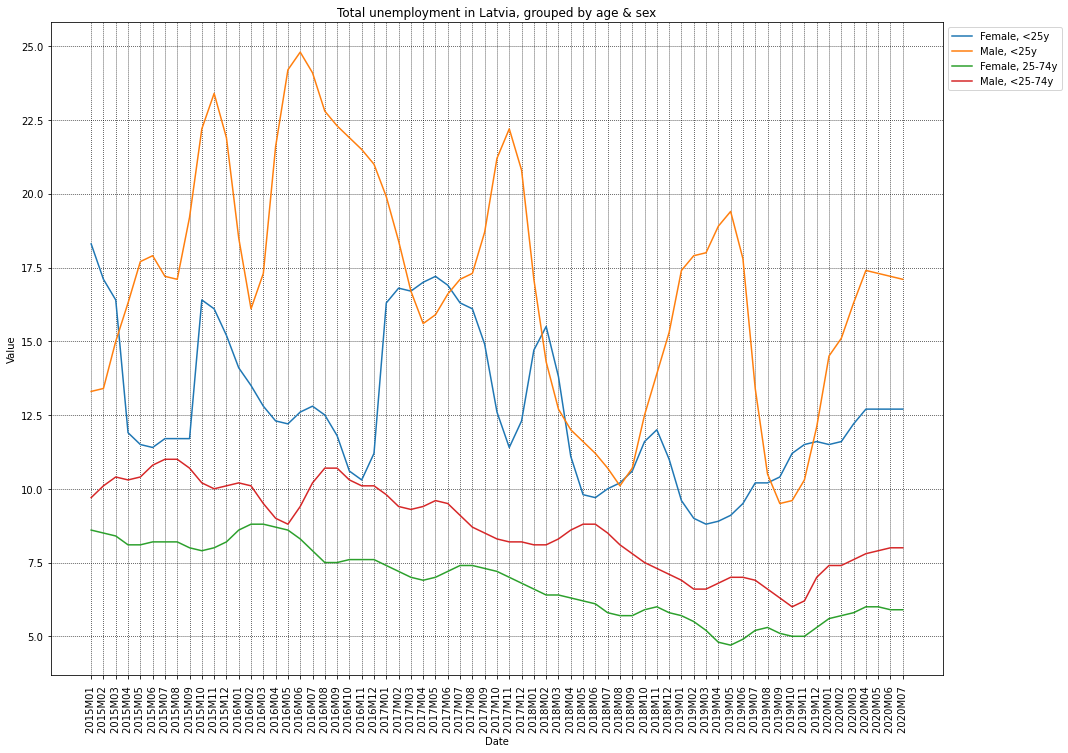
In [38]:

plot\_time\_variation\_age\_sex(data\_tr\_df,country="Estonia")



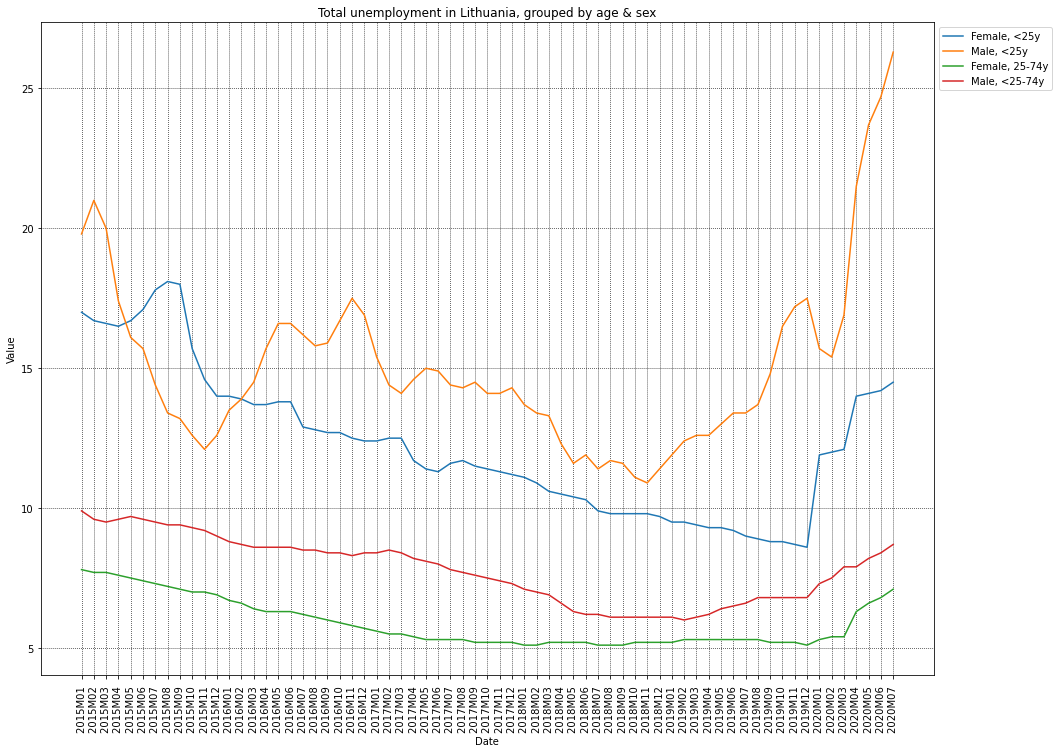
In [39]:

plot\_time\_variation\_age\_sex(data\_tr\_df,country="Latvia")



In [40]:

plot\_time\_variation\_age\_sex(data\_tr\_df,country="Lithuania")



In [41]:

plot\_time\_variation\_age\_sex(data\_tr\_df,country="Romania")



**Animated maps**

We introduce as well animated maps for the unemployments in European countries.

In [42]:

**import** **plotly.express** **as** **px**

**def** plot\_animated\_map(dd\_df, title):

hover\_text = []

**for** index, row **in** dd\_df.iterrows():

hover\_text.append((f"country: **{**row['country']**}**<br>unemployment: **{**row['Value']**}**%<br>country code: **{**row['iso']**}**"))

dd\_df['hover\_text'] = hover\_text

fig = px.choropleth(dd\_df,

locations="C3",

hover\_name='hover\_text',

color="Value",

animation\_frame="Date",

projection="natural earth",

color\_continuous\_scale=px.colors.sequential.Plasma,

width=600, height=600)

fig.update\_geos(

showcoastlines=**True**, coastlinecolor="DarkBlue",

showland=**True**, landcolor="LightGrey",

showocean=**True**, oceancolor="LightBlue",

showlakes=**True**, lakecolor="Blue",

showrivers=**True**, rivercolor="Blue",

showcountries=**True**, countrycolor="DarkBlue"

)

fig.update\_layout(title = title, geo\_scope="europe")

fig.show()

In [43]:

c\_df = data\_tr\_df.loc[(data\_tr\_df.unit=='PC\_ACT')&(data\_tr\_df.s\_adj=='TC')]

dd\_df=c\_df.loc[(c\_df.age=='Y\_LT25')&(c\_df.sex=='F')]

dd\_df = dd\_df.sort\_values(by='Date')

title = 'Percent of unemployed per country<br>Female, under 25 - (hover for details)'

plot\_animated\_map(dd\_df, title)