B2_FlatGlassIndustry_RetrospectiveMEFA

August 23, 2022

This notebook conducts a retrospective analysis of the material and energy flows generated by the European, Belgian and French flat glass industry, with a focus on the architectural flat glass industry. It is part of the doctoral dissertation entitled *Glazing Beyond Energy Efficiency*, and refers to its **Chapter 2, "The Growing Environmental Impact of Architectural Glass."** As such, it should be read in concert with that chapter, which presents the conceptual and methodological framework (Section 2.1) and discusses the results (Sections 2.2 and 2.3).

This notebook uses raw data collected in archives about the flat glass manufacturing process, its environmental impact and the volumes produced. These data are available in the folder "B1_RawData".

This notebook is organised in 8 parts: - the first part recovers data from the Excel files. - the second and third analyse the evolution of flat glass production in Europe, France and Belgium. - the fourth analyses the evolution of the material, energy and CO2 intensity of the flat glass manufacturing process. - the fifth and sixth analyse the evolution of the use of raw materials and energy and the CO2 emissions in absolute values in Europe, Belgium and France. - the seventh part studies the relative and absolute decoupling on a base 100 analysis.

Each of these parts details the method and the calculation, generates graphs and tables. At the end of each part a "Summary" section presents a short overview of the results with the most important graphs, which are published in the thesis.

To cite: Souviron, Jean. 2022. "Glazing Beyond Energy Efficiency: An Environmental Analysis of the Socio-Technical Trajectory of Architectural Glass." PhD diss., Université Libre de Bruxelles

1 Setup

[1]:
```
import os
import re

import pathlib
from pathlib import Path

import sqlite3

import pandas as pd
import numpy as np

from scipy import stats
```

```python
import matplotlib as mpl
import matplotlib.pyplot as plt

import seaborn as sns

from IPython.display import display
```

[2]:
```python
# Helper (support)
from support import input_helper
```

[3]:
```python
# Define size of figure:
mpl.rcParams['figure.figsize'] = (16, 10)
```

[4]:
```python
# Directory with datasets:
ROOT_DIR = Path('./B1_RawData').absolute()

# Define path to save figures:
path_img = os.path.abspath(os.path.join('outputs', 'IMG'))
if not os.path.exists(path_img):
    os.makedirs(path_img)
print(f'Images will be saved in {path_img}')
```

```
Images will be saved in C:\Users\souvi\Documents\These\90_PresentationsAndWritti
ng\90_Manuscript\5_Appendices\Appendix_B\outputs\IMG
```

[5]:
```python
# Define seaborn main parameters:
sns.set_style("ticks")
sns.color_palette("colorblind")
sns.set_context("paper", font_scale=1.5,
                rc={"axes.titlesize": 15, "lines.linewidth": 1.2,
                    "legend.fontsize": 10, "legend.title_fontsize": 10})
```

[6]:
```python
pd.set_option('precision', 2)
```

[7]:
```python
# A custom formatter function which divide by 1000 an axis:
import matplotlib.ticker as tkr

def yfmt1000(x, pos):
    s = f'{x/1000:,.0f}'
    return s

# W/ classes for tick-locating and -formatting:

# A formatter function:
y_1000 = tkr.FuncFormatter(yfmt1000)

# To divide the y-axis by 1000 when needed:
```

```
# ax.yaxis.set_major_formatter(y_1000)
```

[8]:
```
# A function used to define the thickness of x and y axis:
def style_ax(ax):
    for axis in ['top', 'bottom', 'left', 'right']:
        ax.spines[axis].set_linewidth(0.5)
        ax.tick_params(width=0.5)
        ax.set_xlabel(None)
    return ax
```

[9]:
```
# A keyword to export figures, or not:
export = False
```

2 Load Datasets

[10]:
```
# Belgian dataset:
be_data = input_helper.get_data('BE_RawData_VPython.xlsx', directory=ROOT_DIR)

# French dataset:
fr_data = input_helper.get_data('FR_RawData_VPython.xlsx', directory=ROOT_DIR)

# European dataset:
eu_data = input_helper.get_data('EU_RawData_VPython.xlsx', directory=ROOT_DIR)
```

[11]:
```
# Sheets contained by datasets:
print("BE_data, sheet names = \n {}\n".format(be_data.sheet_names))
print("FR_data, sheet names = \n {}\n".format(fr_data.sheet_names))
print("EU_data, sheet names = \n {}\n".format(eu_data.sheet_names))
```

```
BE_data, sheet names =
 ['References', 'prod', 'import', 'export', 'Energy_Intensity', 'Population']

FR_data, sheet names =
 ['References', 'prod', 'import', 'export', 'Population',
'MatEnergy_WindowGlass', 'MatEnergy_PlateGlass', 'RawMat_Intensity',
'Energy_Intensity', 'emissions']

EU_data, sheet names =
 ['References', 'prod', 'import', 'export', 'emissions', 'Energy_Intensity']
```

3 Flat Glass Production in Europe

3.1 Creating a Dataframe for Flat Glass Flows

[12]:
```
# Create dataframe for flat glass production:
df_eu_prod = eu_data.parse('prod').set_index(['Nbr of countries', 'year'])
```

[13]:
```
# Add import and export flows:
df_eu_imp = eu_data.parse('import').set_index(['Nbr of countries', 'year'])
df_eu_exp = eu_data.parse('export').set_index(['Nbr of countries', 'year'])
```

[14]:
```
# Create a single dataframe for flat glass flows in Europe:
df_eu = pd.concat([df_eu_prod['flat glass, kt'],
                   df_eu_imp['flat glass, kt'],
                   df_eu_exp['flat glass, kt']],
                  axis=1, keys=[('Production [kt]'),
                                ('Import [kt]'),
                                ('Export [kt]')],
                  names=['Flow']).stack(dropna=False)

df_eu = df_eu.unstack()
```

3.1.1 First index where we have data

[15]:
```
df_first_valid_index = df_eu.apply(lambda col: col.first_valid_index()).T
df_first_valid_index.columns = df_eu.index.names
df_first_valid_index
```

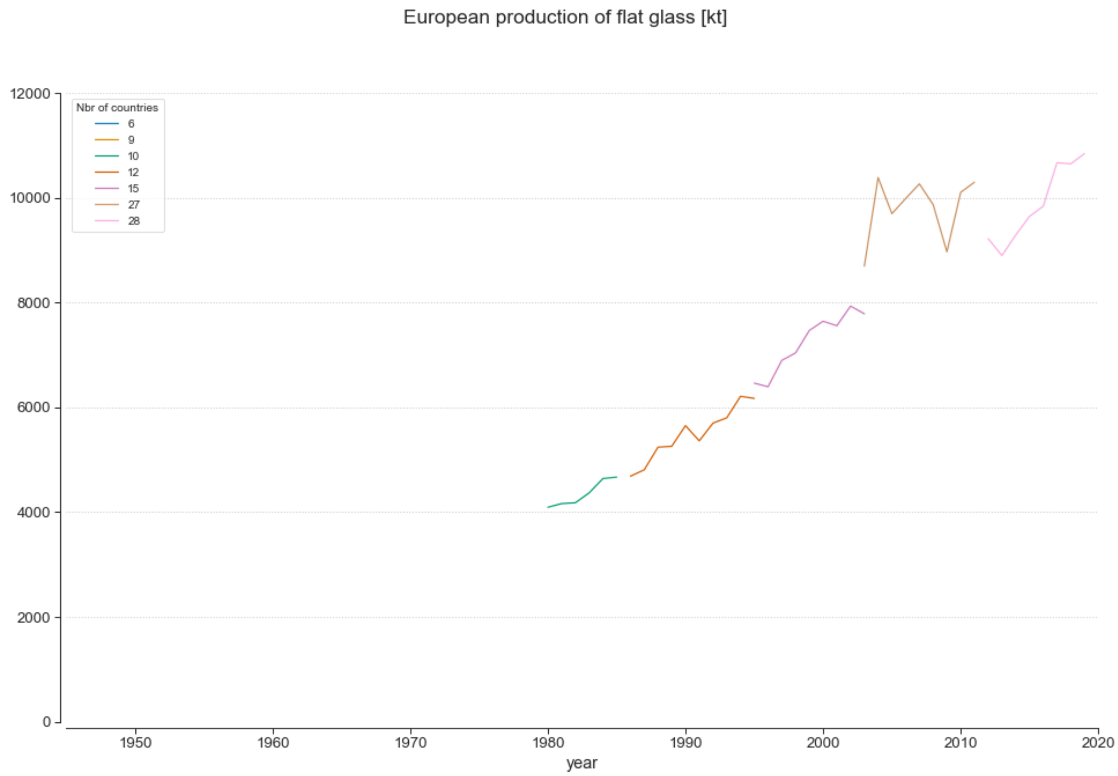
[15]:

	Nbr of countries	year
Flow		
Production [kt]	10	1980
Import [kt]	27	2003
Export [kt]	27	2003

3.2 Flat Glass Production

[16]:
```
# Plot the European production of flat glass:
fig, ax = plt.subplots()

sns.lineplot(data=df_eu.reset_index(),
             x='year', y='Production [kt]',
             hue='Nbr of countries',
             palette='colorblind',
             ax=ax)

ax.yaxis.label.set_visible(False)
ax.set_xlim(1945, 2020)
ax.set_ylim(0, 12000)

ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
```

```
fig.suptitle('European production of flat glass [kt]')
sns.despine(offset=5)
plt.show()
```

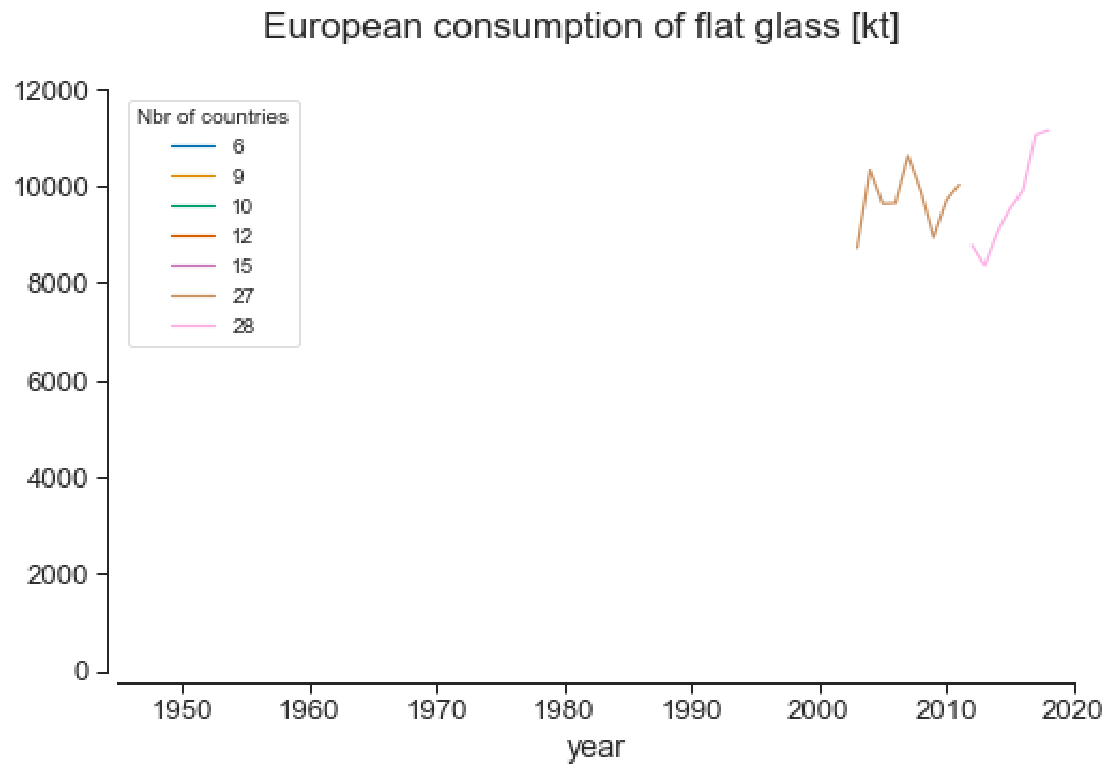


3.3 Flat Glass Consumption

[17]:
```
# Estimate the consumption of flat glass in EU:
df_eu['Consumption [kt]'] = (df_eu['Production [kt]']
                             + df_eu['Import [kt]']
                             - df_eu['Export [kt]'])
```

[18]:
```
# Plot the European consumption of flat glass:
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_eu.reset_index(),
             x='year', y='Consumption [kt]',
             hue='Nbr of countries',
             palette='colorblind',
             ax=ax)

ax.yaxis.label.set_visible(False)
```

```
ax.set_xlim(1945, 2020)
ax.set_ylim(0, 12000)

fig.suptitle('European consumption of flat glass [kt]')
sns.despine(offset=5)
plt.show()
```



3.4 Architectural Flat Glass Production

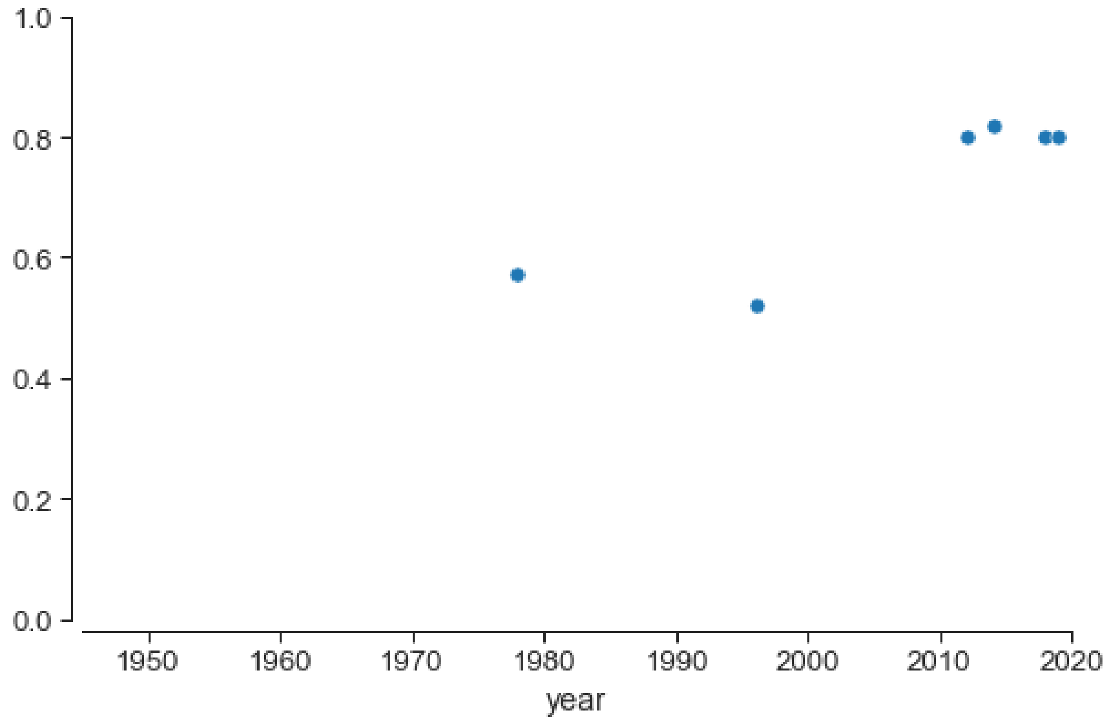
[19]:
```
# Share of architectural glass in the European production of flat glass:
df_eu_bldgshare = df_eu_prod[[
    'bldg glass/flat glass, %']].dropna(inplace=False)
```

[20]:
```
print(f'There {df_eu_bldgshare.shape[0]} datapoints')
```

There 6 datapoints

[21]:
```
# Plot share of architectural glass/total flat glass:
fig, ax = plt.subplots(figsize=(8, 5))

sns.scatterplot(data=df_eu_bldgshare.reset_index(),
                x='year', y='bldg glass/flat glass, %',
                markers="o", s=50, ax=ax)
```

```
ax.yaxis.label.set_visible(False)
ax.set_xlim(1945, 2020)
ax.set_ylim(0, 1)

sns.despine(offset=5)
```

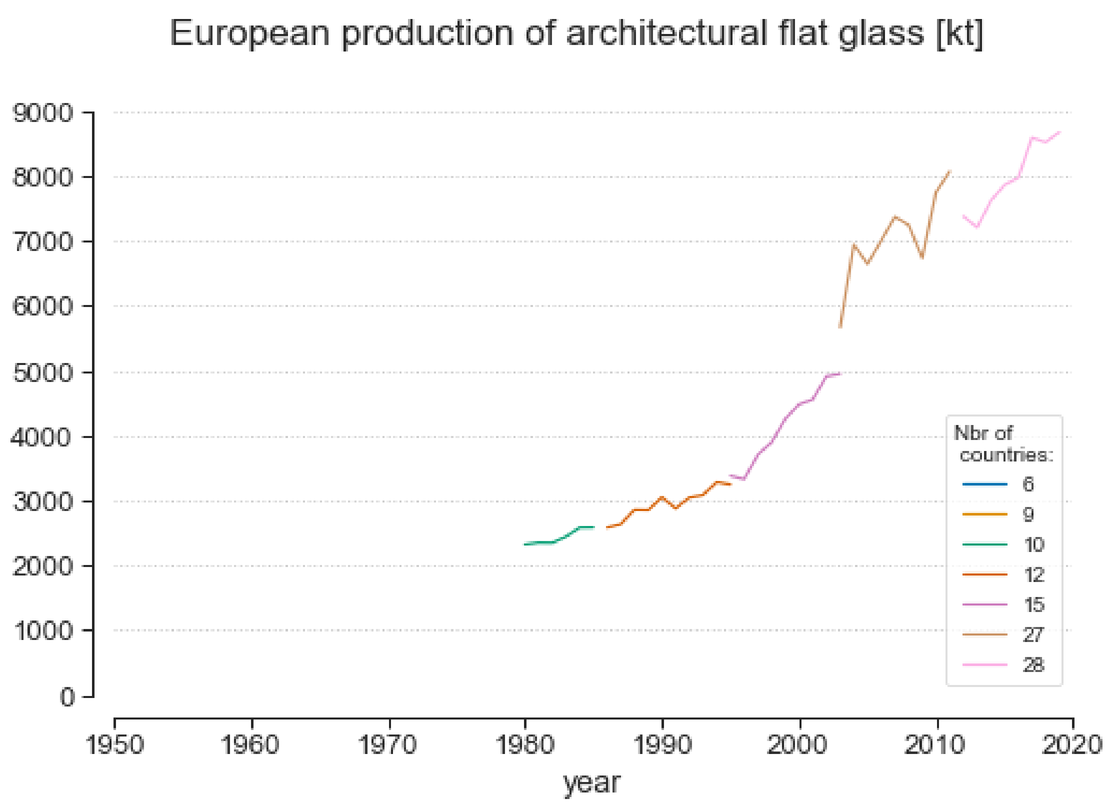


[22]:
```
# Linear interpolation according to available data:
df_eu['bldg glass/flat glass, %'] = (df_eu_prod['bldg glass/flat glass, %']
                                      .interpolate(method="linear",
                                                   limit_area='inside'))
```

[23]:
```
# Calculation of the European production of architectural flat glass:
df_eu['Architectural glass production [kt]'] = (df_eu['Production [kt]']
                                                 * df_eu['bldg glass/flat glass,␣
 ↪%'])
```

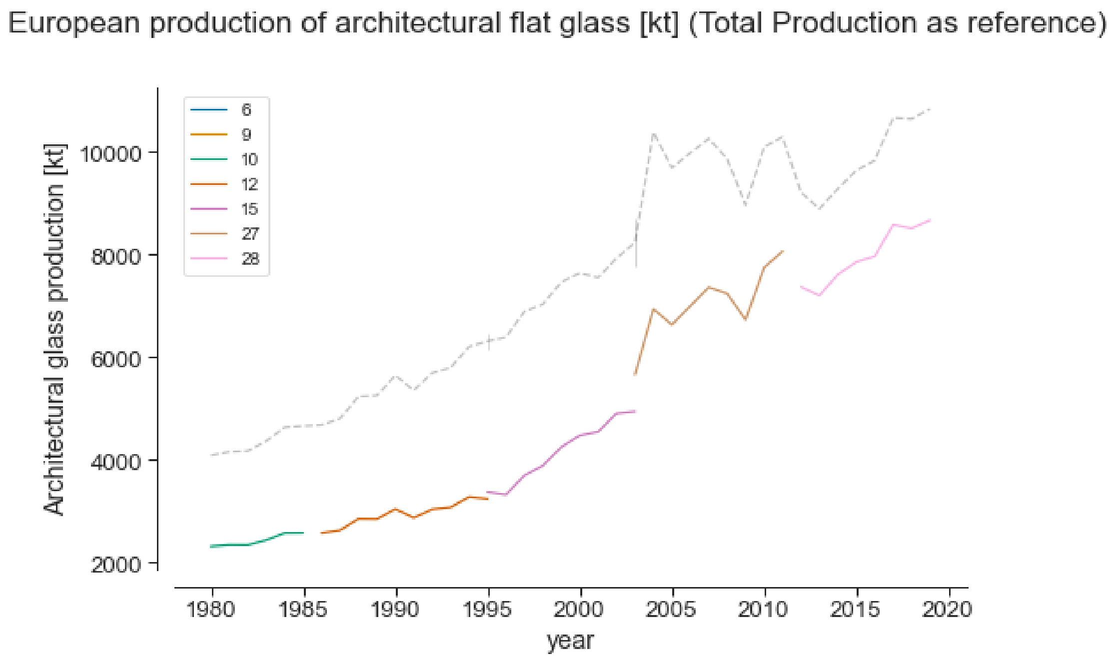
[24]:
```
# Plot the European production of architectural flat glass:
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_eu.reset_index(),
             x='year', y='Architectural glass production [kt]',
             hue='Nbr of countries',
```

```
                palette='colorblind',
                ax=ax)

# Add legend:
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, labels, loc='lower right', ncol=1,
          title='Nbr of \n countries:', bbox_to_anchor=(1, 0))

ax.set_xlim(1950, 2020)
ax.set_ylim(0, 9000)
ax.yaxis.label.set_visible(False)
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

fig.suptitle('European production of architectural flat glass [kt]', y=1)
sns.despine(offset=10)
plt.show()
```



[25]:
```
# Plot the European production of architectural flat glass:
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_eu.reset_index(),
```

```
              x='year', y='Architectural glass production [kt]',
              hue='Nbr of countries',
              palette='colorblind',
              ax=ax)

sns.lineplot(data=df_eu.reset_index(),
             x='year', y='Production [kt]',
             color='black', linestyle='--', alpha=0.25,
             ax=ax)

fig.suptitle(
    'European production of architectural flat glass [kt] (Total Production as␣
 ↪reference)', y=1)
sns.despine(offset=10)
plt.show()
```



3.5 Summary

[26]:
```
# Assess the annual growth:
df_eu['Growth, flat glass [%]'] = df_eu['Production [kt]'].pct_change()
df_eu['Growth, arch glass [%]'] = (df_eu['Architectural glass production [kt]']
                                     .pct_change())
```

[27]:
```
n_countries = df_eu.index.get_level_values(0).unique()
```

```
for i, start_n_countries in enumerate(n_countries[:-1]):
    if start_n_countries < 10:
        continue
    end_n_countries = n_countries[i+1]
    start_year = df_eu.loc[start_n_countries].index.max()
    end_year = df_eu.loc[end_n_countries].index.max()
    a = df_eu.loc[start_n_countries:end_n_countries,
                  'Growth, flat glass [%]'].mean()*100
    b = df_eu.loc[start_n_countries:end_n_countries,
                  'Growth, arch glass [%]'].mean()*100

    print(
        f'Years {start_year}-{end_year} ({start_n_countries} to␣
 ↪{end_n_countries} countries)')
    print(f'Flat glass production growth in EU: {a:.2f}%\n',
          f'Architectural glass production growth in EU: {b:.2f}%')
    print('')
```

```
Years 1985-1995 (10 to 12 countries)
Flat glass production growth in EU: 2.84%
 Architectural glass production growth in EU: 2.33%

Years 1995-2003 (12 to 15 countries)
Flat glass production growth in EU: 2.80%
 Architectural glass production growth in EU: 3.57%

Years 2003-2011 (15 to 27 countries)
Flat glass production growth in EU: 3.10%
 Architectural glass production growth in EU: 5.43%

Years 2011-2019 (27 to 28 countries)
Flat glass production growth in EU: 2.25%
 Architectural glass production growth in EU: 3.65%
```

[28]:
```
n_countries
```

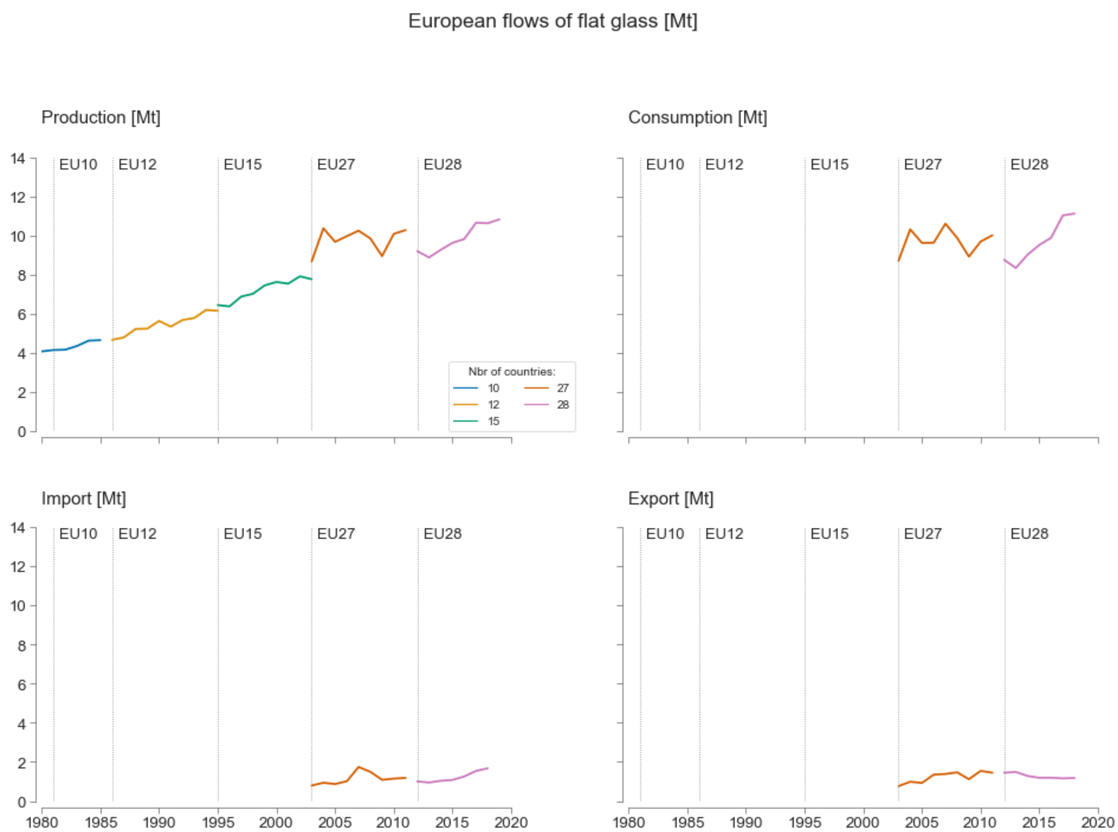
[28]:
```
Int64Index([6, 9, 10, 12, 15, 27, 28], dtype='int64', name='Nbr of countries')
```

[29]:
```
# Key dates for the changes in the MFA scope, i.e. EU enlargement:
EU_KDATES = {'EU10': 1981, 'EU12': 1986, 'EU15': 1995,
             'EU27': 2003, 'EU28': 2012}
```

[30]:
```
# Plot a synthesis of the results concerning European flat glass flows:
data1 = ['Production', 'Consumption']
data2 = ['Import', 'Export']

fig, axes = plt.subplots(nrows=2, ncols=2,
```

```python
                         sharex=True, sharey=True,
                         figsize=(16, 10))

# Plot production and consumption:
for col, data in enumerate(data1):
    ax = axes[0][col]
    sns.lineplot(data=df_eu.loc[10:28].reset_index(),
                 x='year', y=f"{data} [kt]",
                 hue='Nbr of countries',
                 palette='colorblind', linewidth=1.75,
                 ax=ax)

    # Plot lines highlighting changes in the scope, i.e., EU enlargement:
    for y, x in EU_KDATES.items():
        ax.axvline(x=x, c='grey', linestyle=':', linewidth=0.75)
        ax.text(x+0.5, 14000, f"{y}", fontsize=13,
                verticalalignment='top')

    style_ax(ax)

    # Divide by 1000 the y-axis, results in Mt:
    ax.yaxis.set_major_formatter(y_1000)

    ax.set_title(f"{data} [Mt]", pad=30, loc='left')
    #ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
    ax.get_legend().remove()
    ax.yaxis.label.set_visible(False)

# Plot import and export below:
for col, data in enumerate(data2):
    ax = axes[1][col]
    sns.lineplot(data=df_eu.loc[10:28].reset_index(),
                 x='year', y=f"{data} [kt]",
                 hue='Nbr of countries',
                 palette='colorblind', linewidth=1.75,
                 ax=ax)

    # Plot lines highlighting changes in the scope, i.e., EU enlargement:
    for y, x in EU_KDATES.items():
        ax.axvline(x=x, c='grey', linestyle=':', linewidth=0.75)
        ax.text(x+0.5, 14000, f"{y}", fontsize=13,
                verticalalignment='top')

    style_ax(ax)

    # Divide by 1000 the y-axis, results in Mt:
    ax.yaxis.set_major_formatter(y_1000)
```

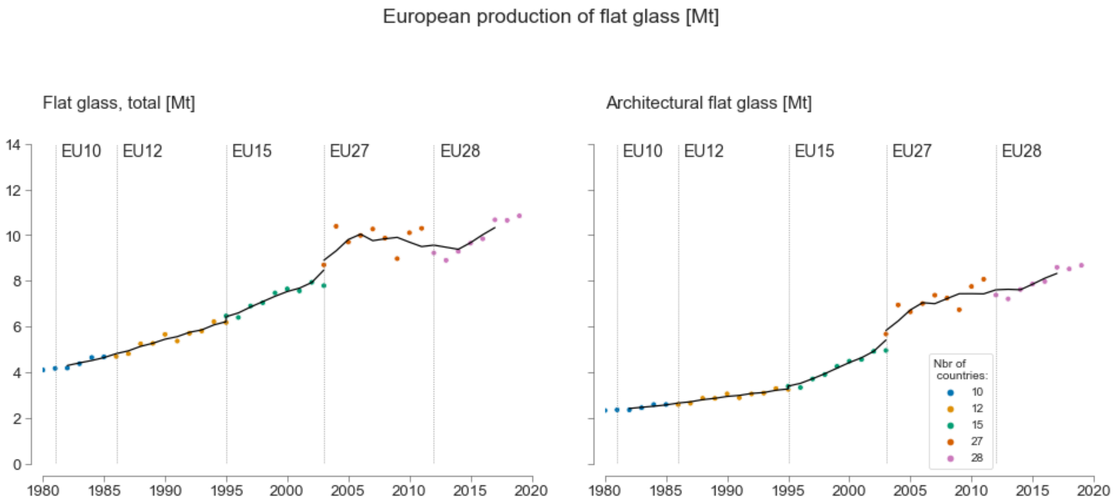
```python
    ax.set_title(f"{data} [Mt]", pad=20, loc='left')
    #ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
    ax.get_legend().remove()
    ax.yaxis.label.set_visible(False)


# Add legend:
handles, labels = ax.get_legend_handles_labels()
fig.legend(handles, labels, loc='center', ncol=2,
           title='Nbr of countries:',
           bbox_to_anchor=(0.47, 0.6))  # (0.83, 0.98)

ax.set_xlim(1980, 2020)
ax.set_ylim(0, 14000)

fig.suptitle("European flows of flat glass [Mt]", y=1.05)
fig.subplots_adjust(wspace=0.25, hspace=0.35)

sns.despine(offset=5)
plt.show()

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'AppendixB_EU_FlatGlassFlows.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'AppendixB_EU_FlatGlassFlows.pdf'),
                bbox_inches='tight')
```



[31]:
```python
# Number of countries for trend curves:
EU_COUNTRIES = [(10, 15), (27, 28)]
```

[32]:
```python
# Plot the European production of flat glass (total and only arch glass):
data = ['Production', 'Architectural glass production']

# Plot two figures:
fig, axes = plt.subplots(nrows=1, ncols=2,
                         sharex=True, sharey=True,
                         figsize=(16, 5))

for col, data in enumerate(data):
    for i, j in EU_COUNTRIES:
        ax = axes[col]
        # Plot the trend curve for flat glass production:
        ax.plot(df_eu.interpolate().rolling(5, center=True).mean().loc[i:j]
                .reset_index(level='Nbr of countries')
                .sort_values(by=['year'])
                [f"{data} [kt]"],
                c='black', linestyle='-')
```

```python
    # Plot data gathered from literature review:
    sns.scatterplot(data=df_eu.loc[10:28].reset_index(),
                    x='year', y=f"{data} [kt]",
                    hue='Nbr of countries',
                    palette='colorblind',
                    marker='.', s=100,
                    ax=ax)
    ax.get_legend().remove()

    # Divide by 1000 the y-axis, results in Mt:
    ax.yaxis.set_major_formatter(y_1000)

    if col == 0:
        ax.set_title('Flat glass, total [Mt]', pad=30, loc='left')
        ax.yaxis.label.set_visible(False)
    else:
        ax.set_title('Architectural flat glass [Mt]', pad=30, loc='left')

    # Plot lines highlighting changes in the scope, i.e., EU enlargement:
    for y, x in EU_KDATES.items():
        ax.axvline(x=x, c='grey', linestyle=':', linewidth=0.75)
        ax.text(x+0.5, 14000, f"{y}", fontsize=14,
                verticalalignment='top')

    style_ax(ax)
    ax.set_xlim(1980, 2020)
    ax.set_ylim(0, 14000)

# Add legend:
handles, labels = ax.get_legend_handles_labels()
fig.legend(handles, labels, ncol=1, loc='lower right',
           title='Nbr of \n countries:',
           bbox_to_anchor=(0.83, 0.1))

plt.suptitle('European production of flat glass [Mt]', weight='light', y=1.2)

fig.subplots_adjust(wspace=0.15)

sns.despine(offset=10)
plt.show()

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig2_EU_FlatGlassProd.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig2_EU_FlatGlassProd.pdf'),
                bbox_inches='tight')
```



4 Flat Glass Production in France and Belgium

4.1 Creating a Dataframe for Flat Glass Flows

[33]:
```
# Create a DataFrame from production datasets:
df_fr_prod = fr_data.parse('prod').set_index('year')
df_be_prod = be_data.parse('prod').set_index('year')
```

[34]:
```
# Create DataFrames from import and export datasets:
df_fr_imp = fr_data.parse('import').set_index('year')
df_fr_exp = fr_data.parse('export').set_index('year')

df_be_imp = be_data.parse('import').set_index('year')
df_be_exp = be_data.parse('export').set_index('year')
```

[35]:
```
# Create a unique DataFrame for FR and BE flat glass flows:
df_be_fr = pd.concat([df_fr_prod['flat glass, kt'],
                      df_be_prod['flat glass, kt'],
                      df_fr_imp['flat glass, kt'],
                      df_be_imp['flat glass, kt'],
                      df_fr_exp['flat glass, kt'],
                      df_be_exp['flat glass, kt']],
                     axis=1, keys=[('Production [kt]', 'France'),
                                   ('Production [kt]', 'Belgium'),
                                   ('Import [kt]', 'France'),
                                   ('Import [kt]', 'Belgium'),
                                   ('Export [kt]', 'France'),
                                   ('Export [kt]', 'Belgium')],
                     names=['Info', 'Country']).stack()
```

```
df_be_fr = df_be_fr.unstack()
```

[36]:
```
df_first_valid_index = df_be_fr.apply(lambda col: col.first_valid_index()).T
df_first_valid_index.columns = df_be_fr.index.names
df_first_valid_index.unstack().style.set_caption('First year with data')
```

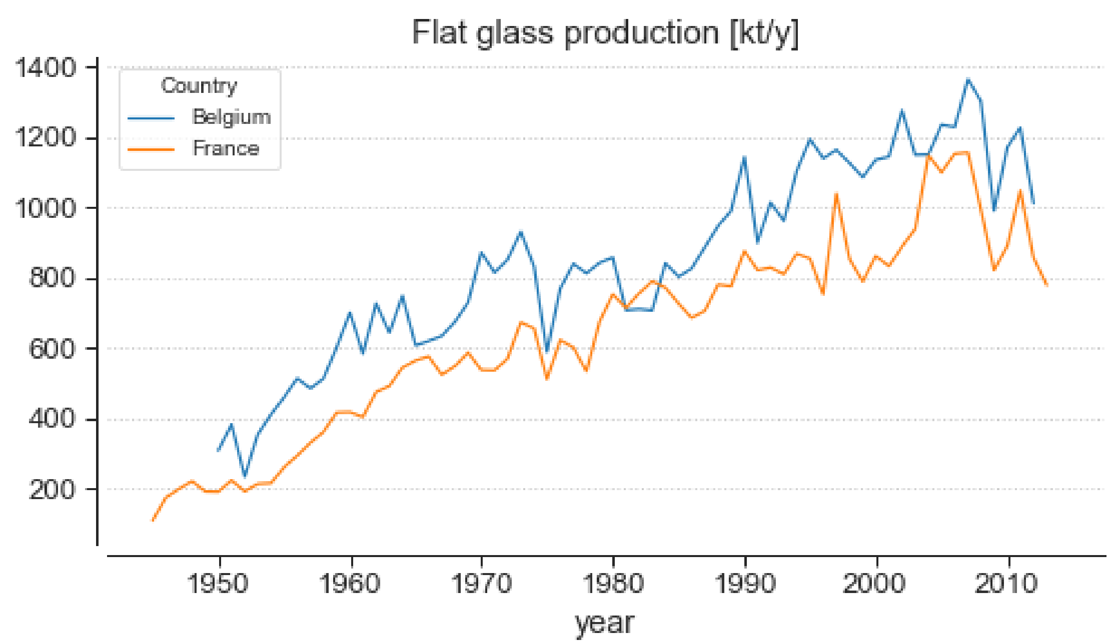
[36]: <pandas.io.formats.style.Styler at 0x1c38a24d310>

4.2 Production of Flat Glass

[37]:
```
countries = ['Belgium', 'France']
```

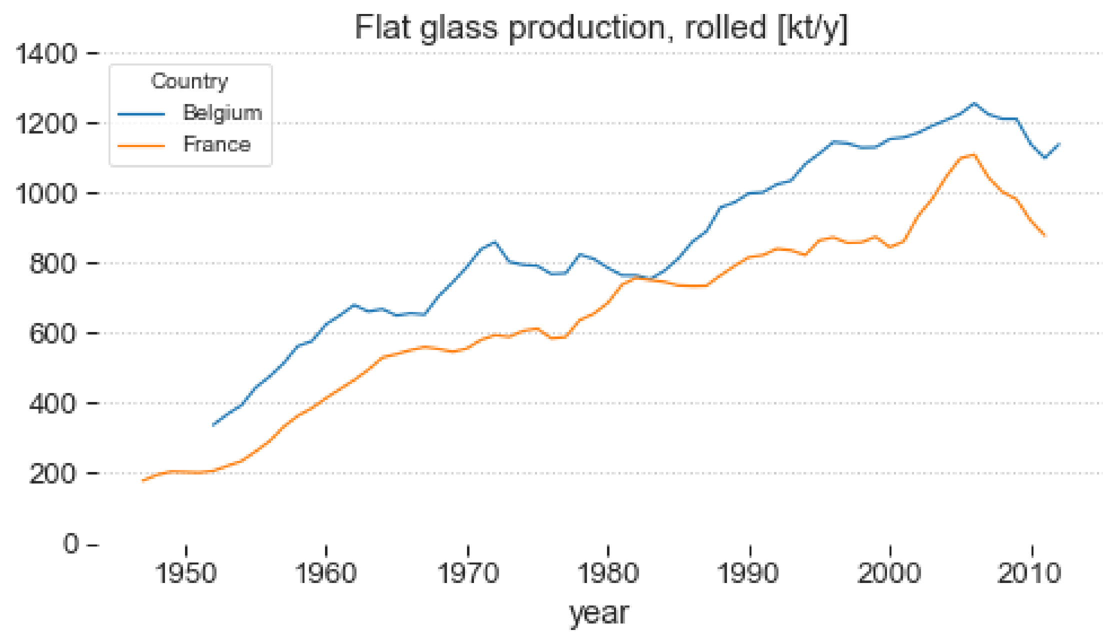
[38]:
```
# Plot the BE and FR production of flat glass:
fig, ax = plt.subplots(figsize=(8, 4))

df_be_fr['Production [kt]'].plot(ax=ax)
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

sns.despine(offset=5)
ax.set_title('Flat glass production [kt/y]')
```

[38]: Text(0.5, 1.0, 'Flat glass production [kt/y]')



[39]:
```
fig, ax = plt.subplots(figsize=(8, 4))
```

```python
df_be_fr['Production [kt]'].interpolate(
    method="linear", limit_area='inside').rolling(
    5, center=True).mean().plot(ax=ax)

sns.despine(bottom=True, left=True)
ax.set_ylim(0, 1400)
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
ax.set_title('Flat glass production, rolled [kt/y]')

plt.show()
```

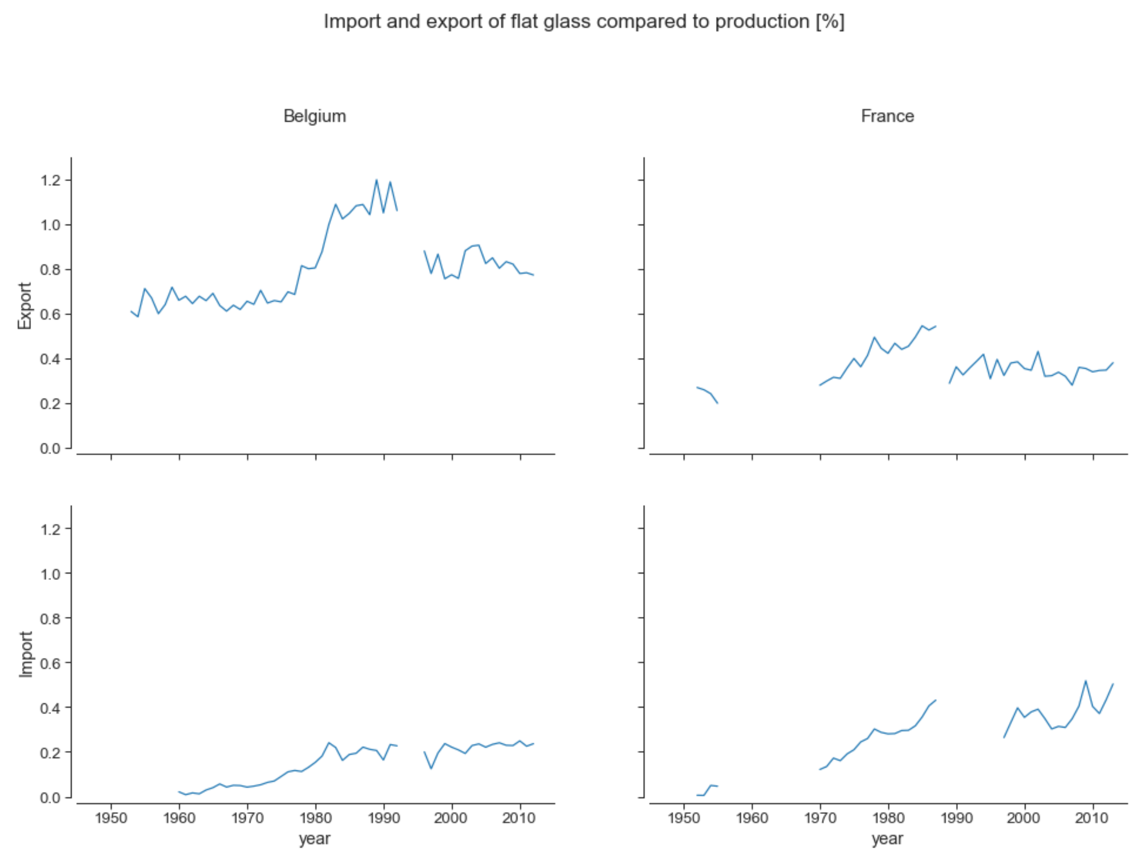


4.3 Consumption of Flat Glass

[40]:
```python
# Caculate import and export ratios relative to production:
df_be_fr = df_be_fr.stack()

df_be_fr['Import Ratio'] = df_be_fr['Import [kt]'] / \
    df_be_fr['Production [kt]']
df_be_fr['Export Ratio'] = df_be_fr['Export [kt]'] / \
    df_be_fr['Production [kt]']

df_be_fr = df_be_fr.unstack()
```

[41]:
```python
ratios = ['Export', 'Import']
```

[42]:
```
# Plot import and export ratios for FR and BE:
nrows = len(ratios)
ncols = len(countries)

fig, axes = plt.subplots(nrows=nrows, ncols=ncols,
                         sharex=True, sharey=True)

for row, ratio in enumerate(ratios):
    for col, country in enumerate(countries):
        ax = axes[row][col]
        (df_be_fr[(f"{ratio} Ratio", country)].plot(ax=ax))

        if row == 0:
            ax.set_title(country, y=1.1)
        if col == 0:
            ax.set_ylabel(ratio)


ax.set_ylim(0, 1.3)
ax.set_xlim(1945, 2015)

fig.suptitle("Import and export of flat glass compared to production [%]",
             y=1.05)
sns.despine(offset=5)
plt.show()
```



[43]:
```
# Interpolation for missing data:
cols = [x for x in df_be_fr.columns if x[0]
        in ['Import Ratio', 'Export Ratio']]

df_be_fr[cols] = (df_be_fr[cols].interpolate(method='linear'))
```

[44]:
```
df_be_fr.loc[1950:1960, cols]
```

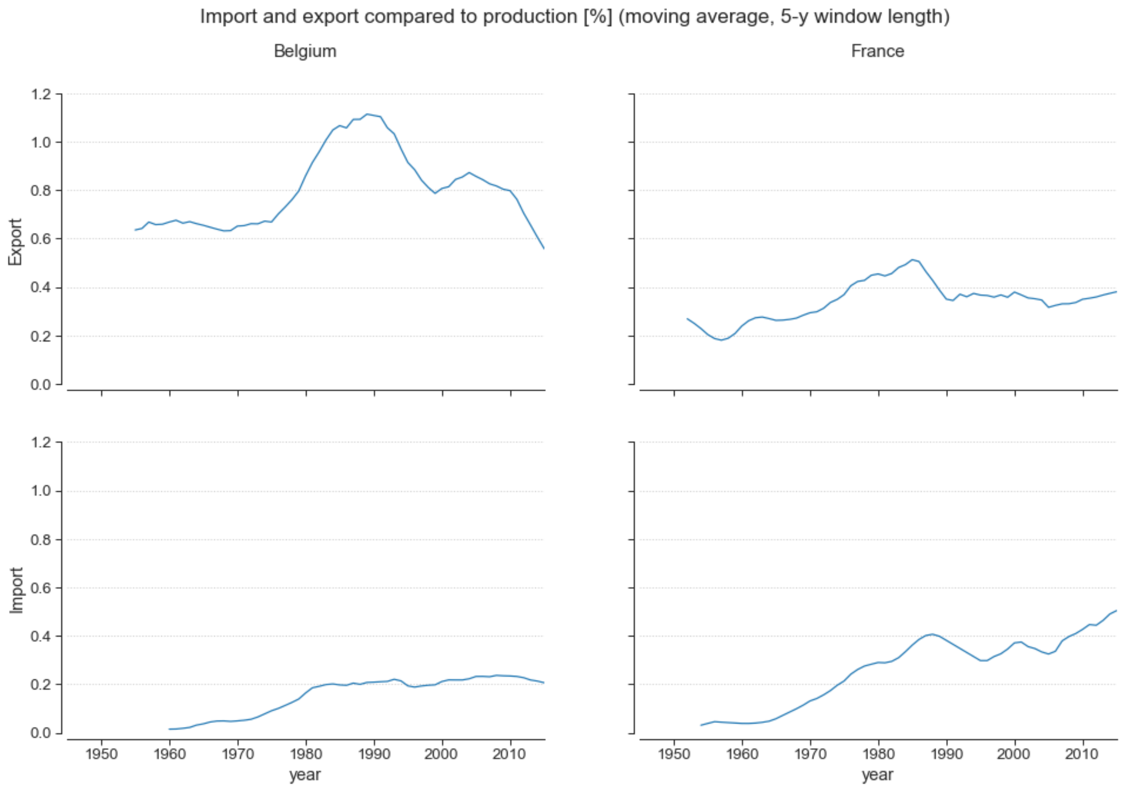
[44]:

Info	Import Ratio		Export Ratio	
Country	Belgium	France	Belgium	France
year				
1950	NaN	NaN	NaN	0.29
1951	NaN	NaN	NaN	0.28
1952	NaN	5.99e-03	NaN	0.27
1953	NaN	5.33e-03	0.61	0.26
1954	NaN	4.99e-02	0.59	0.24
1955	NaN	4.64e-02	0.71	0.20
1956	NaN	4.47e-02	0.67	0.17
1957	NaN	4.30e-02	0.60	0.15
1958	9.23e-03	4.14e-02	0.64	0.18

```
1959        1.53e-02  3.97e-02          0.72    0.21
1960        2.14e-02  3.80e-02          0.66    0.24
```

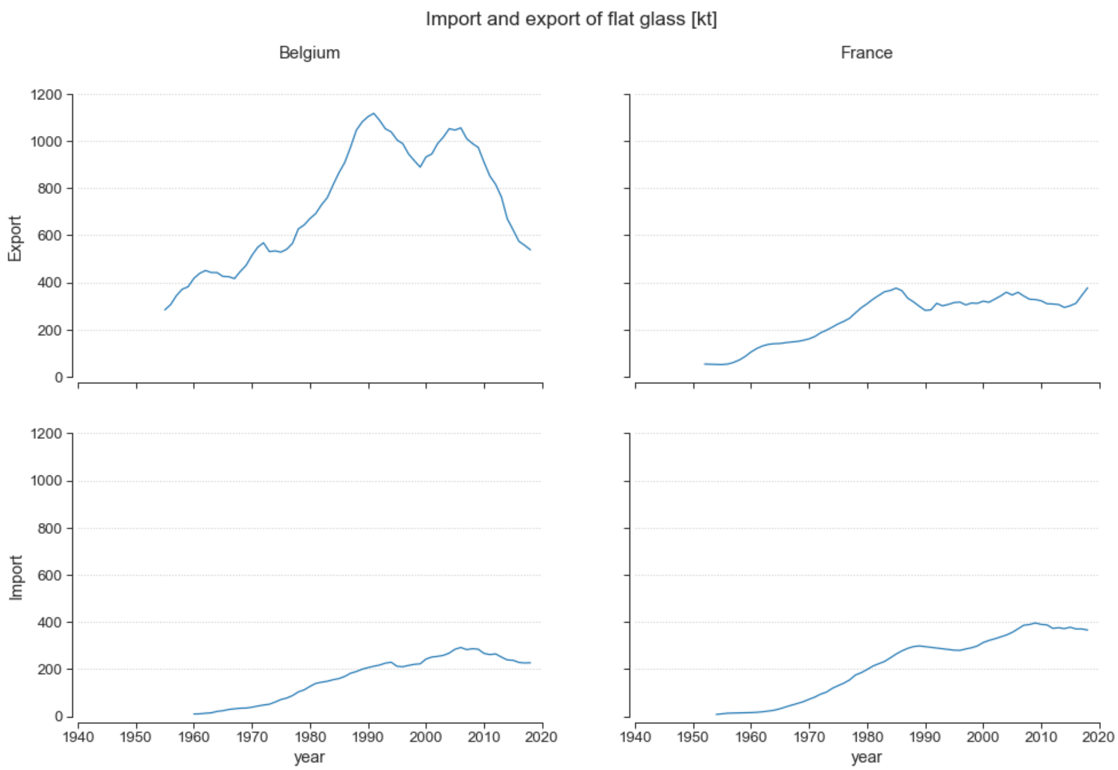
[45]:

```
# Plot import and export ratios, w/ a moving average (5-y. window length):
nrows = len(ratios)
ncols = len(countries)

fig, axes = plt.subplots(nrows=nrows, ncols=ncols,
                         sharex=True, sharey=True)

for row, ratio in enumerate(ratios):
    for col, country in enumerate(countries):
        ax = axes[row][col]
        (df_be_fr[(f"{ratio} Ratio", country)]
         .rolling(5, center=True).mean()
         .plot(ax=ax))

        ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
        if row == 0:
            ax.set_title(country, y=1.1)
        if col == 0:
            ax.set_ylabel(ratio)

ax.set_xlim(1945, 2015)
ax.set_ylim(0, 1.2)

fig.suptitle(
    "Import and export compared to production [%] (moving average, 5-y window␣
 ↪length)")
sns.despine(offset=5)
plt.show()
```




```
[46]: # Plot total import and export flows, moving average w/ a 5-y. window length:
fig, axes = plt.subplots(nrows=nrows, ncols=ncols,
                         sharex=True, sharey=True,
                         figsize=(16, 10))

for row, ratio in enumerate(ratios):
    for col, country in enumerate(countries):
        ax = axes[row][col]
        (df_be_fr[(f"{ratio} [kt]", country)]
         .interpolate(method='linear')
         .rolling(5, center=True).mean()
         .plot(ax=ax))

        ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
        if row == 0:
            ax.set_title(country, y=1.1)
        if col == 0:
            ax.set_ylabel(ratio)

ax.set_xlim(1940, 2020)
ax.set_ylim(0, 1200)
```

```
fig.suptitle("Import and export of flat glass [kt]")
sns.despine(offset=5)
plt.show()
```



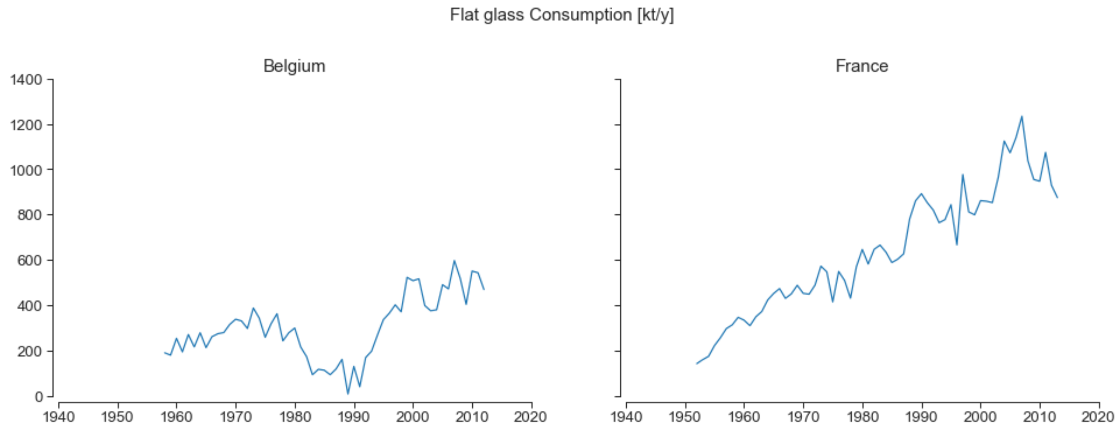
[47]:

```
# Calulate the total consumption of flat glass in BE and FR:
df_be_fr = df_be_fr.stack()

df_be_fr['Consumption [kt]'] = df_be_fr['Production [kt]'] * \
    (1 + df_be_fr['Import Ratio'] - df_be_fr['Export Ratio'])

df_be_fr = df_be_fr.unstack()
```

[48]:

```
# Plot total consumption of flat glass:
fig, axes = plt.subplots(nrows=1, ncols=2,
                         figsize=(16, 5),
                         sharex=True, sharey=True)

for i, (ax, country) in enumerate(zip(axes.flatten(), countries)):

    ax.plot(df_be_fr.index,
            df_be_fr['Consumption [kt]'][country])
    ax.set_title(country)
```

```python
ax.set_xlim(1940, 2020)
ax.set_ylim(0, 1400)

fig.suptitle('Flat glass Consumption [kt/y]', fontsize=15, y=1.05)
sns.despine(offset=5)
plt.show()
```



[49]:
```python
# Create a DataFrame for the FR and BE demography:
df_fr_pop = fr_data.parse('Population').set_index('year')
df_be_pop = be_data.parse('Population').set_index('year')
```

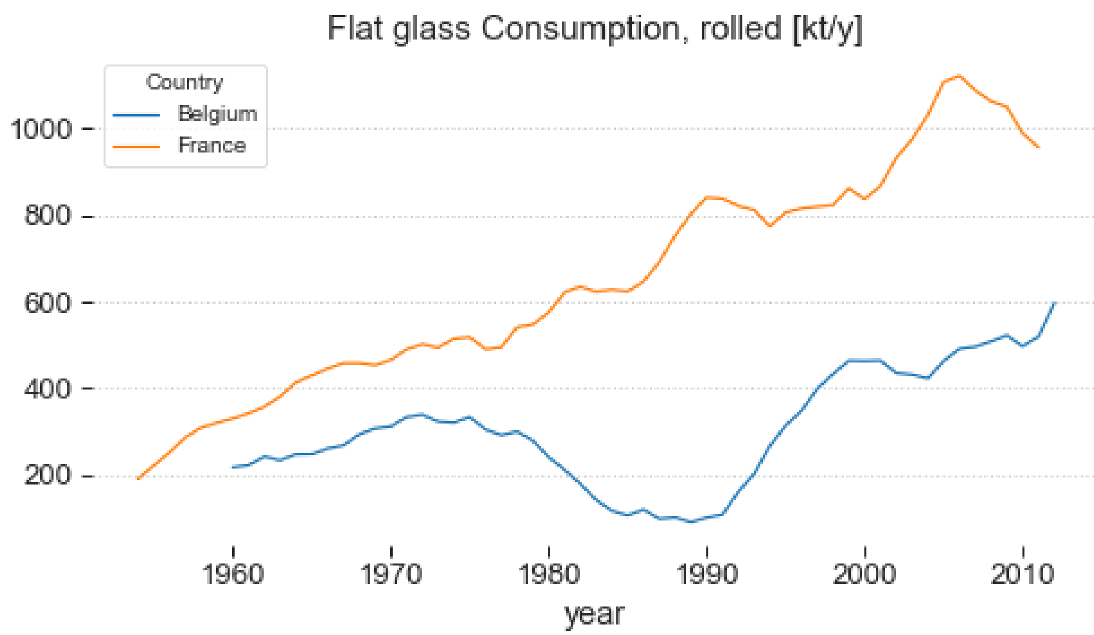
[50]:
```python
# Create a single DataFrame for French and Belgian population since 1945:
df_be_fr_pop = pd.concat([df_fr_pop['Population, x1000'],
                          df_be_pop['Population, x1000']],
                         axis=1, keys=[('Population [x1000]', 'France'),
                                       ('Population [x1000]', 'Belgium')],
                         names=['Info', 'Country']).stack()

df_be_fr_pop = df_be_fr_pop.unstack()
```

[51]:
```python
# Linear interpolation for a few missing data (BE, from 1947 to 1959):
df_be_fr_pop[('Population [x1000]', 'Belgium')] = (
    df_be_fr_pop[('Population [x1000]', 'Belgium')
                 ].interpolate(method='linear')
)
```

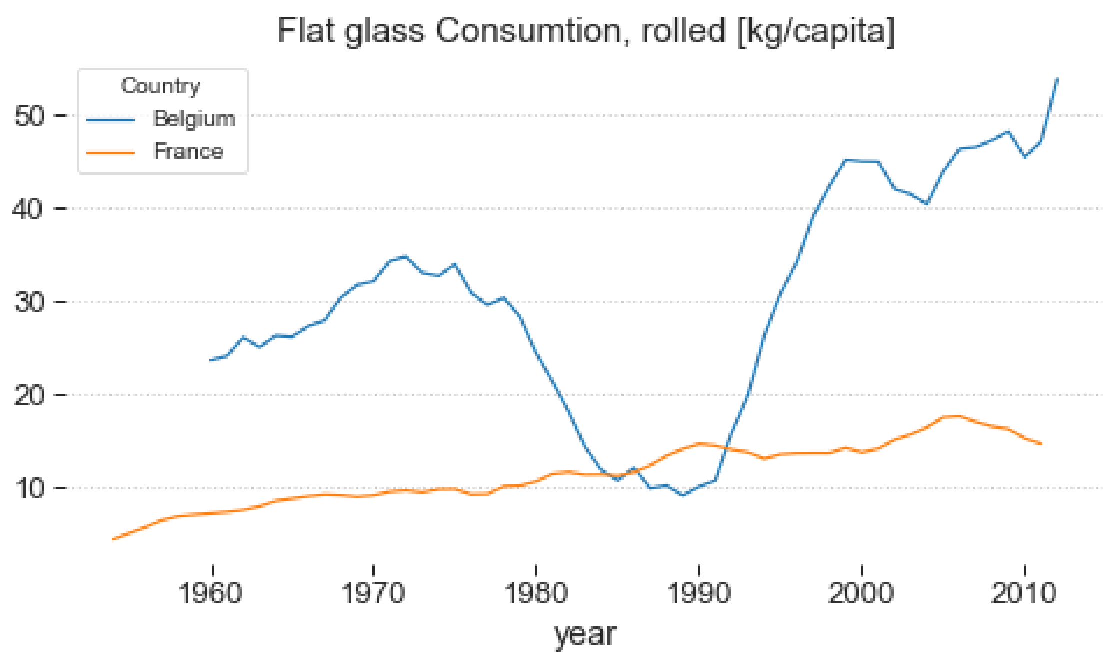
[52]:
```python
# Estimate the consumption per capita:
for country in countries:
    df_be_fr[('Consumption [kg/cap]', country)] = (
        df_be_fr[('Consumption [kt]', country)]
        / df_be_fr_pop[('Population [x1000]', country)] * 1000
```

```
    )
```

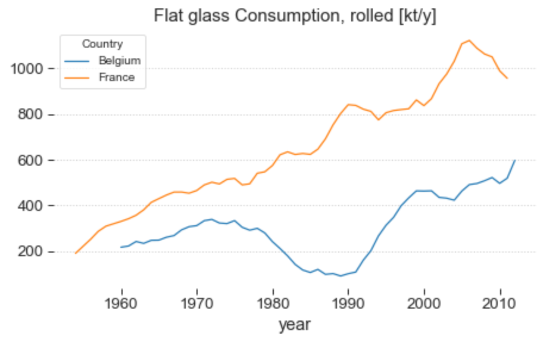
[53]:
```
fig, ax = plt.subplots(figsize=(8, 4))

df_be_fr['Consumption [kt]'].interpolate(
    method="linear", limit_area='inside').rolling(5, center=True).mean().
 ↪plot(ax=ax)
sns.despine(bottom=True, left=True)
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
ax.set_title('Flat glass Consumption, rolled [kt/y]')

plt.show()
```

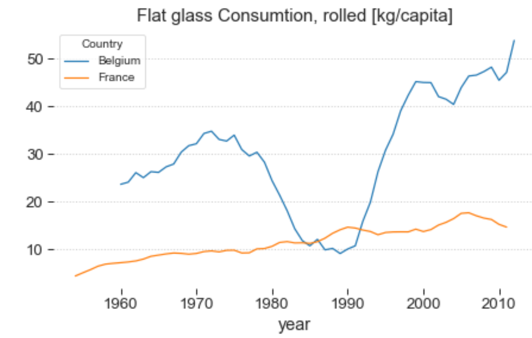


[54]:
```
fig, ax = plt.subplots(figsize=(8, 4))

df_be_fr['Consumption [kg/cap]'].interpolate(
    method="linear", limit_area='inside').rolling(5, center=True).mean().
 ↪plot(ax=ax)
sns.despine(bottom=True, left=True)
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
ax.set_title('Flat glass Consumtion, rolled [kg/capita]')

plt.show()
```



[55]:
```
fig, (ax0, ax1) = plt.subplots(
    figsize=(16, 4), ncols=2, sharex=True, sharey=False)

(df_be_fr['Consumption [kt]']
 .interpolate(method="linear", limit_area='inside')
 .rolling(5, center=True).mean()
 .plot(ax=ax0)
 )
sns.despine(bottom=True, left=True)
ax0.grid(which='major', axis='y', linestyle=':', linewidth=1)
ax0.set_title('Flat glass Consumption, rolled [kt/y]')

(df_be_fr['Consumption [kg/cap]']
 .interpolate(method="linear", limit_area='inside')
 .rolling(5, center=True).mean()
 .plot(ax=ax1)
 )
sns.despine(bottom=True, left=True)
ax1.grid(which='major', axis='y', linestyle=':', linewidth=1)
ax1.set_title('Flat glass Consumtion, rolled [kg/capita]')


plt.show()
```





4.4 Production of Architectural Flat Glass

[56]:
```
# Share of building glass relative to the total flat glass production in FR:
df_share_fr = df_fr_prod[['bldg glass/flat glass, %']].dropna()

print(df_share_fr.isnull().value_counts())
df_share_fr
```

```
bldg glass/flat glass, %
False                       13
dtype: int64
```

[56]:

year	bldg glass/flat glass, %
1945	0.90
1952	0.80
1962	0.68
1968	0.50
1980	0.50
1994	0.50
2002	0.52
2004	0.49
2009	0.52
2010	0.50
2011	0.43
2012	0.51
2013	0.54

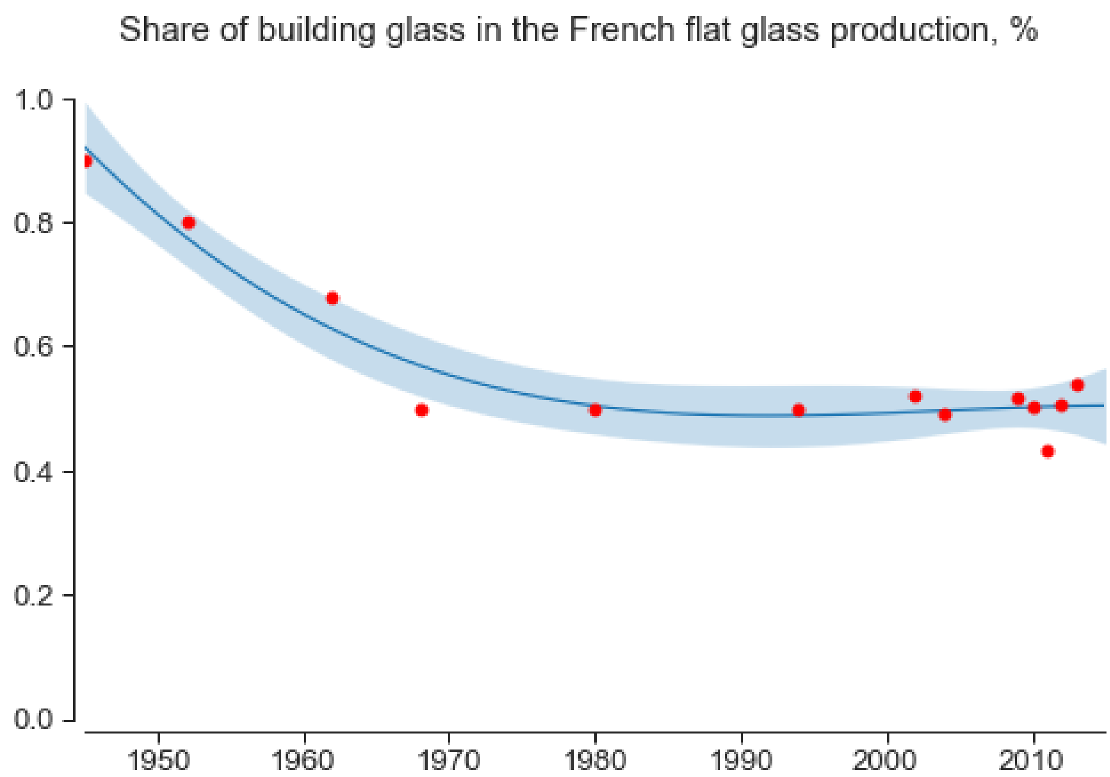
[57]:
```
# Calculate the main parameters of a regression curve, second order
# to interpolate "arch glass/total flat glass" ratio, with uncertainties:
df_share_fr['Trend'] = df_fr_prod[['bldg glass/flat glass, %']].copy()

df_share_fr = df_share_fr.reset_index()

X_FR = df_share_fr['year'].values
```

```python
Y_FR = df_share_fr['Trend'].values

df_share_fr = df_share_fr.set_index('year')

# Order of the regression:
N = 3

# Polynomial coefficients and covariance matrix:
coeffs, cov = np.polyfit(X_FR, Y_FR, N, cov=True)

results_fr = {}
results_fr['Polynomial'] = coeffs.tolist()
results_fr['Covariance'] = cov.tolist()

# Interpolation:
T_FR = np.linspace(1945, 2020, 76)

# Matrix with rows 1, t, t**2...:
TT = np.vstack([T_FR**(N-i) for i in range(1+N)]).T

# Matrix multiplication for the polynomial values:
Z_FR = np.dot(TT, coeffs)

# Standard deviations (sqrt of diagonal):
SIG_FR = np.sqrt(
    np.diag(
        np.dot(TT, np.dot(cov, TT.T))
    ))

# r-squared
P_FR = np.poly1d(coeffs)

# fit values, and mean
YHAT_FR = P_FR(X_FR)                         # vector y = p(z) for z in x
YBAR_FR = np.sum(Y_FR)/len(Y_FR)             # mean of y data
SSRES_FR = np.sum((YHAT_FR-YBAR_FR)**2)      # residual sum of squares
SSTOT_FR = np.sum((Y_FR-YBAR_FR)**2)     # total sum squares

results_fr['Coefficient of determination'] = SSRES_FR / SSTOT_FR

results_fr
```

[57]:
```
{'Polynomial': [-2.378960893414362e-06,
  0.014301359629274039,
  -28.656983590865888,
  19140.70797173813],
 'Covariance': [[2.8029866613415203e-12,
```

```
   -1.6645777335936083e-08,
   3.2948302953039774e-05,
   -0.021737417444942602],
  [-1.6645777335958778e-08,
   9.88533429177357e-05,
   -0.19567014611392025,
   129.09330281953902],
  [3.294830295312971e-05,
   -0.1956701461141876,
   387.3129582861321,
   -255532.078955934],
  [-0.021737417445031697,
   129.09330281989222,
   -255532.0789562839,
   168590499.05965033]],
 'Coefficient of determination': 0.9311453347637837}
```

[58]:
```
# Scatterplot of the ratio and plot of a regression curve, 2nd order:
fig, ax = plt.subplots(figsize=(8, 5))

ax.fill_between(T_FR, Z_FR+1.96*SIG_FR, Z_FR-1.96*SIG_FR, alpha=0.25)
ax.plot(T_FR, Z_FR, '-')
ax.plot(X_FR, Y_FR, 'ro')

ax.set_ylim(0, 1)
ax.set_xlim(1945, 2015)

fig.suptitle('Share of building glass in the French flat glass production, %',
             fontsize=15)
sns.despine(offset=5)
plt.show()
```



[59]:
```
# Share of building glass relative to the total flat glass production in BE:
df_share_be = df_be_prod[['bldg glass/flat glass, %']].dropna()

print(df_share_be.isnull().sum())
df_share_be
```

```
bldg glass/flat glass, %    0
dtype: int64
```

[59]:

year	bldg glass/flat glass, %
1945	0.90
1950	0.90
1960	0.88
1965	0.85
1970	0.82
1975	0.76
1980	0.66
1990	0.63
1992	0.62
2002	0.70
2003	0.72

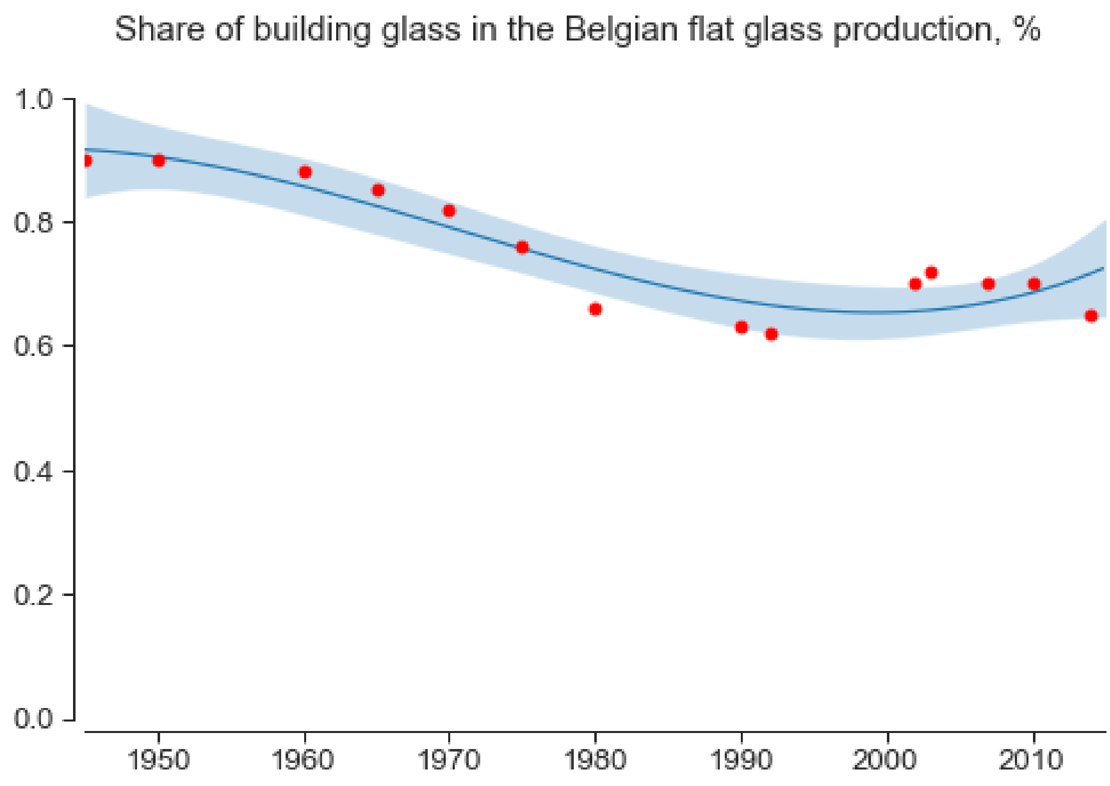
```
2007                      0.70
2010                      0.70
2014                      0.65
```

[60]:

```python
# Calculate the main parameters of a regression curve, second order
# to interpolate "arch glass/total flat glass" ratio, with uncertainties:
df_share_be['Trend'] = df_be_prod[['bldg glass/flat glass, %']].copy()
df_share_be = df_share_be.reset_index()

X_BE = df_share_be['year'].values
Y_BE = df_share_be['Trend'].values

df_share_be = df_share_be.set_index('year')

# Order of the regression:
N = 3

# Polynomial coefficients and covariance matrix:
coeffs, cov = np.polyfit(X_BE, Y_BE, N, cov=True)

results_be = {}
results_be['Polynomial'] = coeffs.tolist()
results_be['Covariance'] = cov.tolist()

# Interpolation:
T_BE = np.linspace(1945, 2020, 76)

# Matrix with rows 1, t, t**2...:
TT = np.vstack([T_BE**(N-i) for i in range(1+N)]).T

# Matrix multiplication for the polynomial values:
Z_BE = np.dot(TT, coeffs)

# Standard deviations (sqrt of diagonal):
SIG_BE = np.sqrt(
    np.diag(
        np.dot(TT, np.dot(cov, TT.T))
    ))

# r-squared
P_BE = np.poly1d(coeffs)

# fit values, and mean
YHAT_BE = P_BE(X_BE)                      # vector y = p(z) for z in x
YBAR_BE = np.sum(Y_BE)/len(Y_BE)          # mean of y data
SSRES_BE = np.sum((YHAT_BE-YBAR_BE)**2)   # residual sum of squares
SSTOT_BE = np.sum((Y_BE-YBAR_BE)**2)    # total sum squares
```

```
results_be['Coefficient of determination'] = SSRES_BE / SSTOT_BE

# results_be
```

[61]:
```
# Scatterplot of the ratio and plot of a regression curve, 2nd order:
fig, ax = plt.subplots(figsize=(8, 5))

ax.fill_between(T_BE, Z_BE+1.96*SIG_BE, Z_BE-1.96*SIG_BE, alpha=0.25)
ax.plot(T_BE, Z_BE, '-')
ax.plot(X_BE, Y_BE, 'ro')

ax.set_ylim(0, 1)
ax.set_xlim(1945, 2015)

fig.suptitle('Share of building glass in the Belgian flat glass production, %',
             fontsize=15)
sns.despine(offset=5)
plt.show()
```



[62]:
```
# Create a unique DataFrame for FR and BE flat glass flows:
df_bldg_share = pd.concat([df_fr_prod['bldg glass/flat glass, %'],
                           df_be_prod['bldg glass/flat glass, %']],
                          axis=1, keys=[('bldg glass/flat glass, %', 'France'),
                                        ('bldg glass/flat glass, %',␣
 ↪'Belgium')],
                          names=['Info', 'Country']).stack(dropna=False)

df_bldg_share = df_bldg_share.unstack()
```

[63]:
```
# Interpolate values according to
# nonlinear regression previously estimated
df_bldg_share[('Mean ratio arch glass/flat glass, %', 'France')
             ] = Z_FR.tolist()
df_bldg_share[('Mean ratio arch glass/flat glass, %',
              'Belgium')] = Z_BE.tolist()
```

[64]:
```
# Integrate standard deviation in df_fr_prod
df_bldg_share[('std for ratio arch glass/flat glass', 'France')
             ] = SIG_FR.tolist()
df_bldg_share[('std for ratio arch glass/flat glass',
              'Belgium')] = SIG_BE.tolist()

# df_bldg_share
```

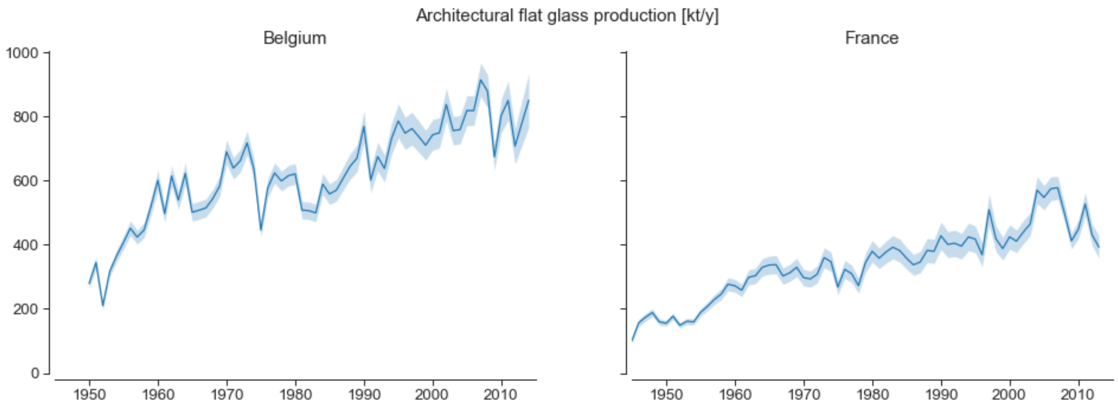
[65]:
```
# Estimate architectural flat glass production in FR and BE:
for country in countries:
    df_be_fr[('Architectural glass production [kt]', country)] = (
        df_be_fr[('Production [kt]', country)]
        * df_bldg_share[('Mean ratio arch glass/flat glass, %', country)]
    )

for country in countries:
    df_be_fr[('std for arch glass production', country)] = (
        df_be_fr[('Production [kt]', country)]
        * df_bldg_share[('std for ratio arch glass/flat glass', country)]
    )
```

[66]:
```
# Calculate data for uncertainties according to the standard deviaton, 95%:
x = np.array(df_be_fr.index)

yminFR = np.array(df_be_fr[('Architectural glass production [kt]', 'France')]
                  .interpolate(method="linear", limit_area='inside')
                  + 1.96
                  * df_be_fr[('std for arch glass production', 'France')]
                  .interpolate(method="linear", limit_area='inside'))
```

```python
ymaxFR = np.array(df_be_fr[('Architectural glass production [kt]', 'France')]
                  .interpolate(method="linear", limit_area='inside')
                  - 1.96
                  * df_be_fr[('std for arch glass production', 'France')]
                  .interpolate(method="linear", limit_area='inside'))

yminBE = np.array(df_be_fr[('Architectural glass production [kt]', 'Belgium')]
                  .interpolate(method="linear", limit_area='inside')
                  + 1.96
                  * df_be_fr[('std for arch glass production', 'Belgium')]
                  .interpolate(method="linear", limit_area='inside'))

ymaxBE = np.array(df_be_fr[('Architectural glass production [kt]', 'Belgium')]
                  .interpolate(method="linear", limit_area='inside')
                  - 1.96
                  * df_be_fr[('std for arch glass production', 'Belgium')]
                  .interpolate(method="linear", limit_area='inside'))
```

[67]:

```python
# Plot architectural flat glass production with uncertainties (95%):
fig, axes = plt.subplots(nrows=1, ncols=2,
                         figsize=(16, 5),
                         sharex=True, sharey=True)

for i, (ax, country) in enumerate(zip(axes.flatten(), countries)):

    ax.plot(df_be_fr.index,
            df_be_fr['Architectural glass production [kt]'][country]
            .interpolate(method="linear", limit_area='inside'))

    if country == 'France':
        ax.fill_between(x, yminFR, ymaxFR, alpha=0.25)
    elif country == 'Belgium':
        ax.fill_between(x, yminBE, ymaxBE, alpha=0.25)

    ax.set_title(country)
    ax.set_xlim(1945, 2015)
    ax.set_ylim(ymin=0)

fig.suptitle('Architectural flat glass production [kt/y]', fontsize=15)
sns.despine(offset=5)
plt.show()
```

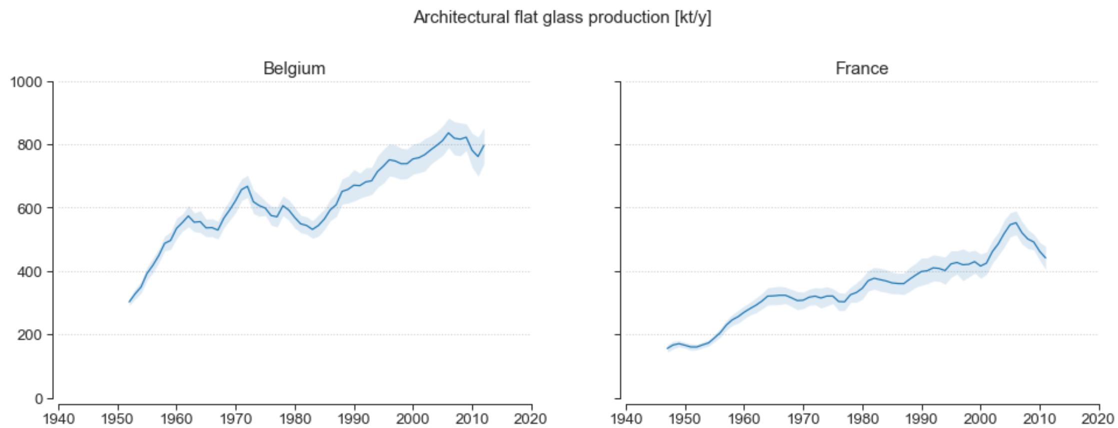


[68]:
```python
# Plot architectural flat glass prod, with a moving average, no uncertainty:
fig, axes = plt.subplots(nrows=1, ncols=2,
                         figsize=(16, 5),
                         sharex=True, sharey=True)

for i, (ax, country) in enumerate(zip(axes.flatten(), countries)):
    ax.plot(df_be_fr.index,
            df_be_fr['Architectural glass production [kt]'][country]
            .interpolate(method="linear", limit_area='inside')
            .rolling(5, center=True).mean())

    ax.set_title(country)
    ax.set_xlim(1945, 2020)
    ax.set_ylim(0, 1000)

fig.suptitle('Architectural flat glass production, rolled, [kt]', fontsize=15)
sns.despine(offset=5)
plt.show()
```

[69]:
```
# Calculate data for uncertainties according to the standard deviaton, 95%,
# and undertake a moving average with a 5-y. window length:
ymin_fr = np.array((df_be_fr[('Architectural glass production [kt]', 'France')]
                   .interpolate(method="linear", limit_area='inside')
                   .rolling(5, center=True).mean())
                   + 1.96*df_be_fr[('std for arch glass production', 'France')]
                   .interpolate(method="linear", limit_area='inside')
                   )
ymax_fr = np.array((df_be_fr[('Architectural glass production [kt]', 'France')]
                   .interpolate(method="linear", limit_area='inside')
                   .rolling(5, center=True).mean())
                   - 1.96*df_be_fr[('std for arch glass production', 'France')]
                   .interpolate(method="linear", limit_area='inside')
                   )

ymin_be = np.array((df_be_fr[('Architectural glass production [kt]', 'Belgium')]
                   .interpolate(method="linear", limit_area='inside')
                   .rolling(5, center=True).mean())
                   + 1.96 *
                   df_be_fr[('std for arch glass production', 'Belgium')]
                   .interpolate(method="linear", limit_area='inside')
                   )
ymax_be = np.array((df_be_fr[('Architectural glass production [kt]', 'Belgium')]
                    .interpolate(method="linear", limit_area='inside')
                    .rolling(5, center=True).mean())
                   - 1.96 *
                   df_be_fr[('std for arch glass production', 'Belgium')]
                   .interpolate(method="linear", limit_area='inside')
                   )
```

[70]:
```
# Plot architectural flat glass prod, with a moving average and uncertainties:
fig, axes = plt.subplots(nrows=1, ncols=2,
                         figsize=(16, 5),
                         sharex=True, sharey=True)

for i, (ax, country) in enumerate(zip(axes.flatten(), countries)):
    ax.plot(df_be_fr.index,
            (df_be_fr[('Architectural glass production [kt]', country)]
             .interpolate(method="linear", limit_area='inside')
             .rolling(5, center=True).mean()))

    if country == 'France':
        ax.fill_between(x, ymin_fr, ymax_fr, alpha=0.15)
    elif country == 'Belgium':
        ax.fill_between(x, ymin_be, ymax_be, alpha=0.15)

    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
```

```
    ax.set_title(country)
    ax.set_xlim(1940, 2020)
    ax.set_ylim(0, 1000)

fig.suptitle('Architectural flat glass production [kt/y]',
             fontsize=15, y=1.05)
sns.despine(offset=5)
plt.show()
```



4.5 Consumption of Architectural Flat Glass

[71]:
```
# Calculate Belgian and French consumption of architectural flat glass:
df_be_fr = df_be_fr.stack()

df_be_fr['Architectural glass consumption [kt]'] = (
    df_be_fr['Architectural glass production [kt]']
    + df_be_fr['Import Ratio'] *
    df_be_fr['Architectural glass production [kt]']
    - df_be_fr['Export Ratio'] *
    df_be_fr['Architectural glass production [kt]']
)

df_be_fr = df_be_fr.unstack()
```

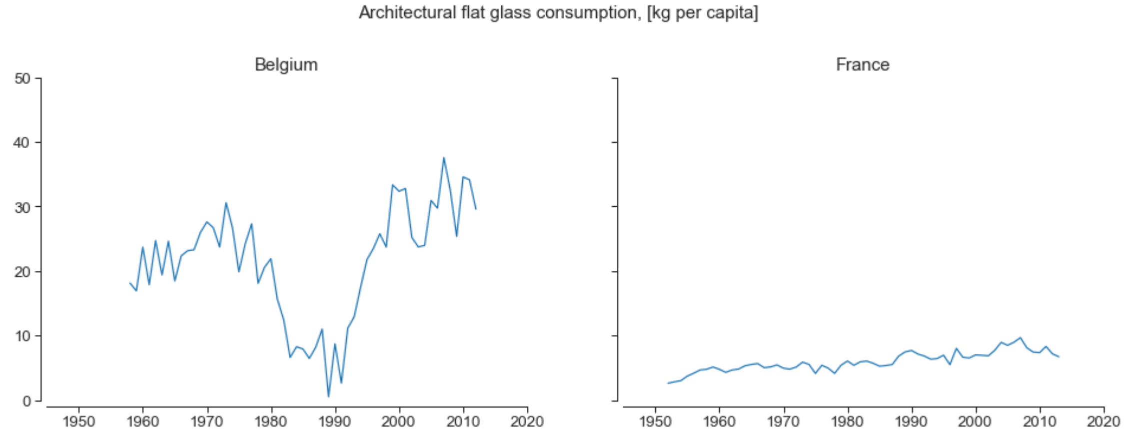
[72]:
```
# Interpolate and calculate a mean average for architectural glass consumption
# and plot:
fig, axes = plt.subplots(nrows=1, ncols=2,
                         figsize=(16, 5),
                         sharex=True, sharey=True)

for i, (ax, country) in enumerate(zip(axes.flatten(), countries)):
```

```
    ax.plot(df_be_fr.index,
            df_be_fr['Architectural glass consumption [kt]'][country]
            .interpolate(method="linear", limit_area='inside')
            .rolling(5, center=True).mean())

    ax.set_title(country)
    ax.set_xlim(1945, 2020)
    ax.set_ylim(0, 700)

fig.suptitle('Architectural flat glass consumption, rolled, [kt]',
             fontsize=15, y=1.05)
sns.despine(offset=5)
plt.show()
```

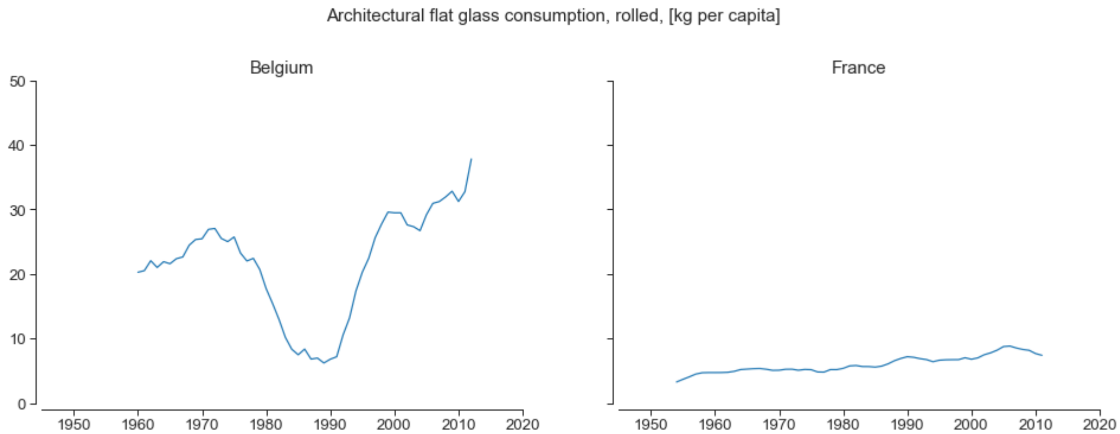
[73]:
```
# Estimate the consumption per capita:
for country in countries:
    df_be_fr[('Architectural glass consumption [kg/cap]', country)] = (
        df_be_fr[('Architectural glass consumption [kt]', country)]
        / df_be_fr_pop[('Population [x1000]', country)] * 1000
    )
```

[74]:
```
# Plot architectural glass consumption per capita:
fig, axes = plt.subplots(nrows=1, ncols=2,
                         figsize=(16, 5),
                         sharex=True, sharey=True)

for i, (ax, country) in enumerate(zip(axes.flatten(), countries)):
    ax.plot(df_be_fr.index,
            df_be_fr['Architectural glass consumption [kg/cap]'][country])

    ax.set_title(country)
    ax.set_xlim(1945, 2020)
```

```python
    ax.set_ylim(0, 50)

fig.suptitle('Architectural flat glass consumption, [kg per capita]',
             fontsize=15, y=1.05)
sns.despine(offset=5)
plt.show()
```



[75]:
```python
# Interpolation and moving average for architectural glass consuption / cap
# and plot:
fig, axes = plt.subplots(nrows=1, ncols=2,
                         figsize=(16, 5),
                         sharex=True, sharey=True)

for i, (ax, country) in enumerate(zip(axes.flatten(), countries)):
    ax.plot(df_be_fr.index,
            df_be_fr['Architectural glass consumption [kg/cap]'][country]
            .interpolate(method="linear", limit_area='inside')
            .rolling(5, center=True).mean())

    ax.set_title(country)
    ax.set_xlim(1945, 2020)
    ax.set_ylim(0, 50)

fig.suptitle('Architectural flat glass consumption, rolled, [kg per capita]',
             fontsize=15, y=1.05)
sns.despine(offset=5)
plt.show()
```



4.6 Summary

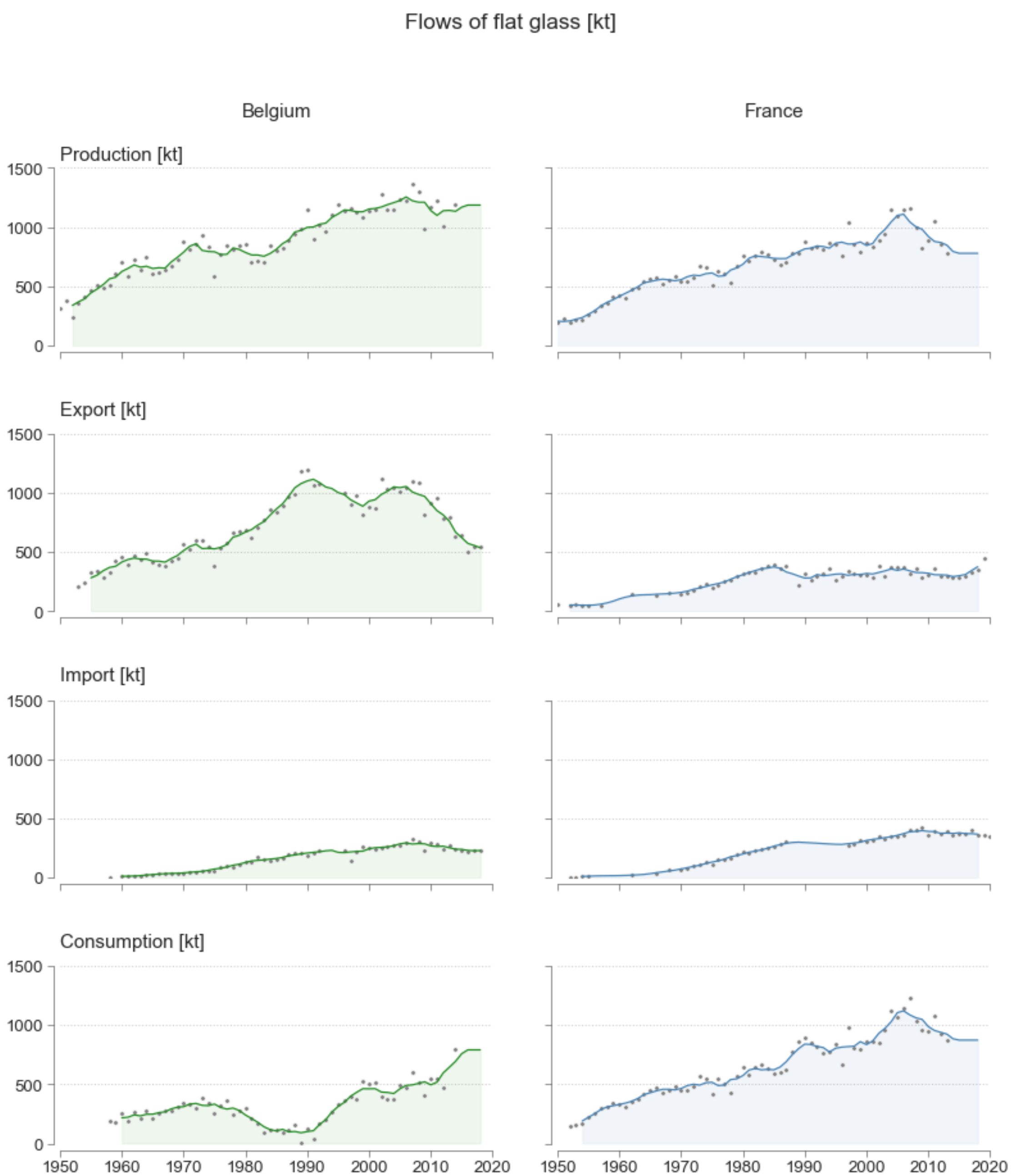
[76]:
```
df_be_fr['Consumption [kt]'].describe()
```

[76]:

Country	Belgium	France
count	56.00	62.00
mean	307.75	640.08
std	154.00	273.17
min	7.09	141.09
25%	195.46	434.50
50%	287.25	614.18
75%	389.49	856.54
max	791.58	1233.49

[77]:
```
# Plot total import and export flows, moving average w/ a 5-y. window length:
toplot = ['Production', 'Export', 'Import', 'Consumption']
nrows = len(toplot)
ncols = len(countries)

fig, axes = plt.subplots(nrows=nrows, ncols=ncols,
                         sharex=True, sharey=True,
                         figsize=(13, 14))

for row, data in enumerate(toplot):
    for col, country in enumerate(countries):
        ax = axes[row][col]
        # Define the color and title for the subplots:
        if country == 'Belgium':
            color = 'forestgreen'
            ax.set_title(f"{data} [kt]", loc='left', pad=15)
        else:
            color = 'steelblue'
```

```
        (df_be_fr[(f"{data} [kt]", country)]
         .interpolate(method='linear')
         .rolling(5, center=True).mean()
         .plot(ax=ax, c=color)
         )

        x = df_be_fr.index
        y = (df_be_fr[(f"{data} [kt]", country)]
             .interpolate(method='linear')
             .rolling(5, center=True).mean())
        ax.fill_between(x, y, color=color, alpha=0.075)

        sns.scatterplot(data=df_be_fr[f"{data} [kt]"][country],
                        marker='.', s=50, color='grey',
                        ax=ax)

        ax.get_legend().remove()

        ax.minorticks_off()
        ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
        style_ax(ax)
        ax.set_ylabel(None)

        if row == 0:
            ax.set_title(country, y=1.25)

ax.set_xlim(1950, 2020)
ax.set_ylim(0, 1500)
plt.xticks(np.arange(1950, 2021, 10))

fig.subplots_adjust(wspace=0.15, hspace=0.5)

fig.suptitle("Flows of flat glass [kt]", y=1)
sns.despine(offset=5)

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'AppendixC_BEFR_FlatGlassFlows.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'AppendixC_BEFR_FlatGlassFlows.pdf'),
                bbox_inches='tight')
```




```
[78]: # Calculate annual growth:
      for country in countries:
          df_be_fr[('Growth, flat glass [%]', country)] = (
              df_be_fr[('Production [kt]', country)].pct_change()
          )

          df_be_fr[('Growth, arch glass [%]', country)] = (
              df_be_fr[('Architectural glass production [kt]', country)].pct_change()
```

```
    )

    df_be_fr[('Growth, arch glass consumption [%]', country)] = (
        df_be_fr[('Architectural glass consumption [kt]', country)].pct_change()
    )
```

[79]:

```
# Growth of the production in FR from 1950 to 1965:
A = df_be_fr[('Growth, flat glass [%]', 'France')].loc[1951:1966].mean()*100
B = df_be_fr[('Growth, arch glass [%]', 'France')].loc[1951:1966].mean()*100
print(f'Flat glass production growth in FR, 1950-1965:\n {A:.2f}%\n',
      f'Architectural glass production growth in FR, 1950-1965:\n {B:.2f}%')

# Growth of the production in FR from 1965 to 2000:
C = df_be_fr[('Growth, flat glass [%]', 'France')].loc[1965:2001].mean()*100
D = df_be_fr[('Growth, arch glass [%]', 'France')].loc[1965:2001].mean()*100
print(f'Flat glass production growth in FR, 1965-2000:\n {C:.2f}%\n ',
      f'Architectural glass production growth in FR, 1965-2000:\n {D:.2f}%')

# Growth of the consumption in FR from 1955 to 2000:
E = (df_be_fr[('Growth, arch glass consumption [%]', 'France')]
     .loc[1955:2001].mean()*100)
print(f'Architectural glass consumption growth in FR, 1955-2000:\n {E:.2f}%')
```

```
Flat glass production growth in FR, 1950-1965:
 7.53%
 Architectural glass production growth in FR, 1950-1965:
 5.36%
Flat glass production growth in FR, 1965-2000:
 1.80%
  Architectural glass production growth in FR, 1965-2000:
 1.24%
Architectural glass consumption growth in FR, 1955-2000:
 3.44%
```

[80]:

```
# Growth of the production in BE from 1985 to 2005:
F = df_be_fr[('Growth, flat glass [%]', 'Belgium')].loc[1985:2006].mean()*100
G = df_be_fr[('Growth, arch glass [%]', 'Belgium')].loc[1985:2006].mean()*100
print(f'Flat glass production growth in BE, 1985-2005:\n {F:.2f}%\n ',
      f'Architectural glass production growth in BE, 1985-2005:\n {G:.2f}')
```

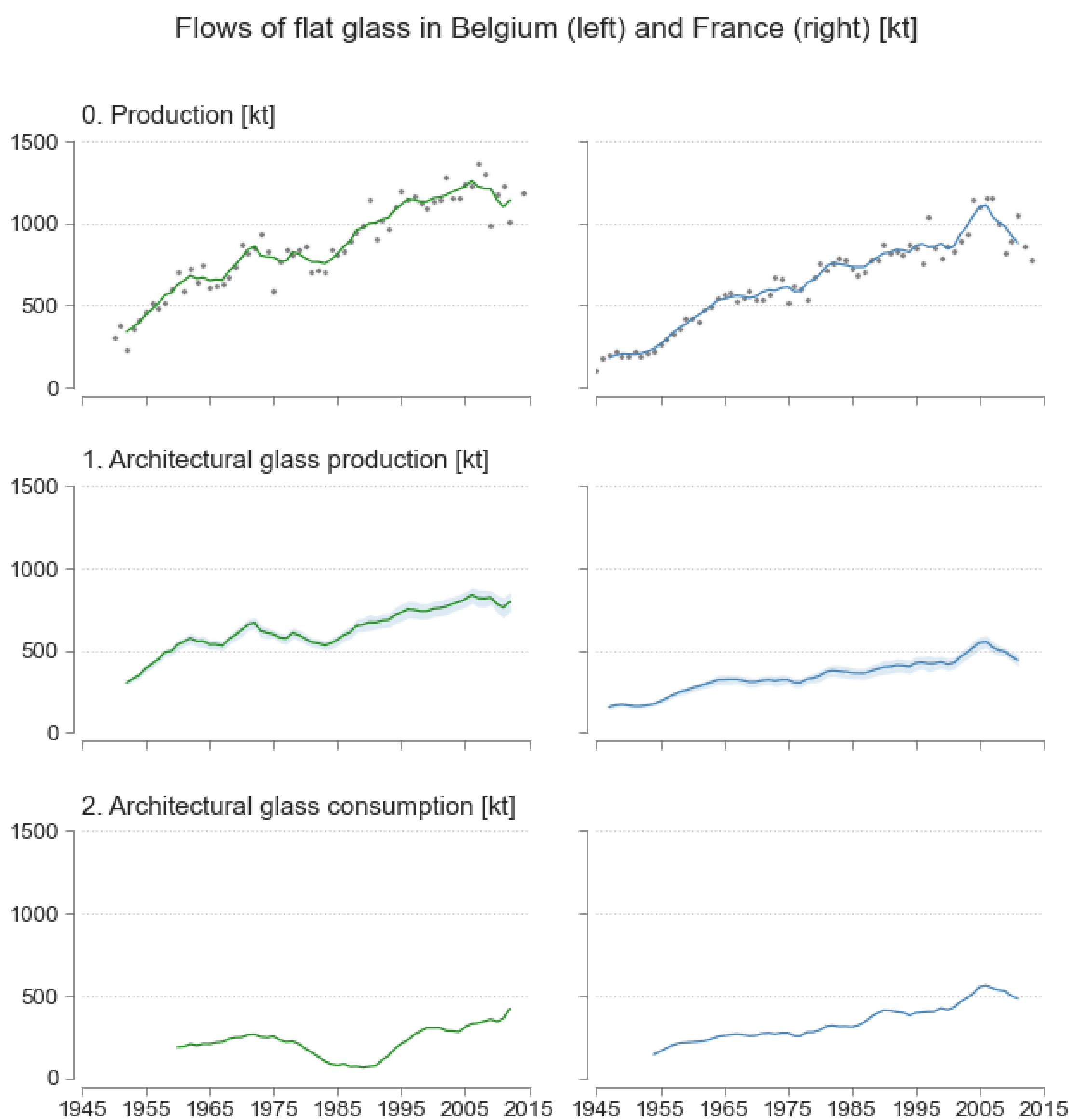
```
Flat glass production growth in BE, 1985-2005:
 2.11%
  Architectural glass production growth in BE, 1985-2005:
 1.87
```

[81]:

```
# Plot graphs to summarize main results:
toplot = ['Production',
          'Architectural glass production',
```

```
          'Architectural glass consumption']

nrows = len(toplot)
ncols = len(countries)

fig, axes = plt.subplots(nrows=nrows, ncols=ncols,
                         sharex=True, sharey=True,
                         figsize=(10, 10))

for row, plot in enumerate(toplot):
    for col, country in enumerate(countries):
        ax = axes[row][col]
        # Define the color and title for the subplots:
        if country == 'Belgium':
            color = 'forestgreen'
            ax.set_title(f"{row}. {plot} [kt]", pad=10, loc='left')
            if plot == 'Architectural glass production':
                ax.fill_between(x, ymin_be, ymax_be, alpha=0.15)
        else:
            color = 'steelblue'
            if plot == 'Architectural glass production':
                ax.fill_between(x, ymin_fr, ymax_fr, alpha=0.15)

        # Plot the 5-year rolled data:
        (df_be_fr[(f"{plot} [kt]", country)]
         .interpolate(method="linear", limit_area='inside')
         .rolling(5, center=True).mean()
         .plot(ax=ax, c=color)
         )
        ax.minorticks_off()
        ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
        style_ax(ax)

        # Plot data gathered from literature review:
        if row == 0:
            sns.scatterplot(data=df_be_fr['Production [kt]'][country],
                            marker='.', s=50, color='grey',
                            ax=ax)
            ax.set_ylabel(None)
            ax.get_legend().remove()

ax.set_ylim(0, 1500)
ax.set_xlim(1945, 2015)

plt.xticks(np.arange(1945, 2016, 10))

fig.suptitle("Flows of flat glass in Belgium (left) and France (right) [kt]")
```

```python
fig.subplots_adjust(wspace=0.15, hspace=0.4)

sns.despine(offset=5)
plt.show()

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig3_BE-FR_FlatGlassFlows.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig3_BE-FR_FlatGlassFlows.pdf'),
                bbox_inches='tight')
```



5 Material, Energy and Carbon Intensity in the Flat Glass Industry

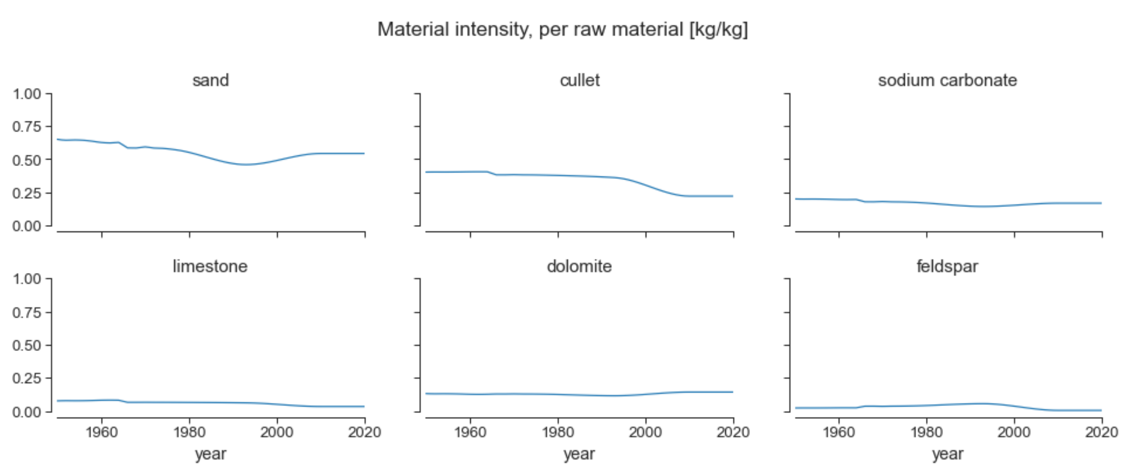
5.1 Material Intensity since 1945, France

[82]:
```
# Creating a DataFrame for material intensity:
df_rawmat = fr_data.parse('RawMat_Intensity').set_index('year')
```

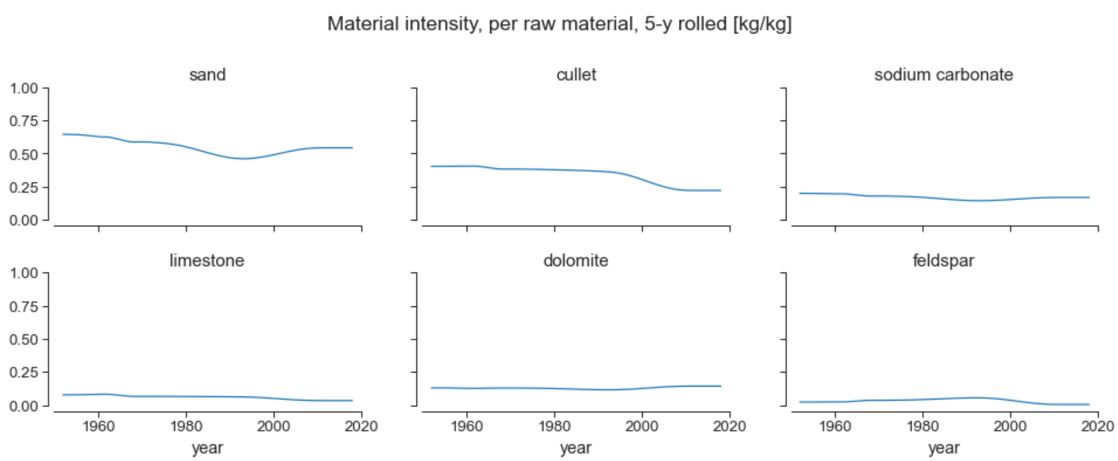
[83]:
```
df_rawmat['cullet, kg/kg'] = (df_rawmat['internal cullet, kg/kg']
                              + df_rawmat['external cullet, kg/kg'])
```

[84]:
```
materials = ['sand', 'cullet', 'sodium carbonate',
             'limestone', 'dolomite', 'feldspar']
```

[85]:
```
fig, axes = plt.subplots(nrows=2, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 5))

for i, (ax, mat) in enumerate(zip(axes.flatten(), materials)):
    ax.set_title(mat)
    (df_rawmat[f"{mat}, kg/kg"]
     .interpolate(method='pchip', limit_direction='forward')
     .plot(ax=ax)
     )

ax.set_xlim(1950, 2020)
ax.set_ylim(0, 1)

fig.suptitle("Material intensity, per raw material [kg/kg]", y=1.05)
fig.subplots_adjust(hspace=0.4)

sns.despine(offset=5)
plt.show()
```

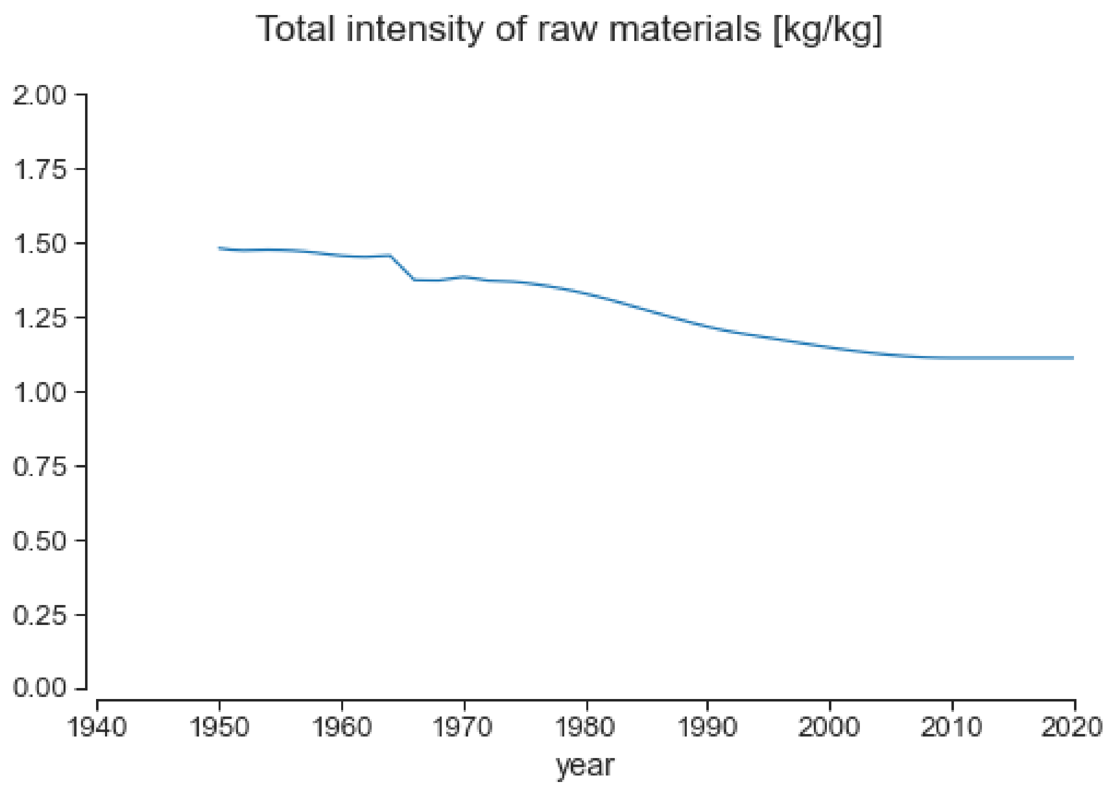


[86]:
```python
fig, axes = plt.subplots(nrows=2, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 5))

for i, (ax, mat) in enumerate(zip(axes.flatten(), materials)):
    ax.set_title(mat)
    (df_rawmat[f"{mat}, kg/kg"]
     .interpolate(method='pchip', limit_direction='forward')
     .rolling(5, center=True).mean()
     .plot(ax=ax))

ax.set_xlim(1950, 2020)
ax.set_ylim(0, 1)

fig.suptitle("Material intensity, per raw material, 5-y rolled [kg/kg]",
             y=1.05)
fig.subplots_adjust(hspace=0.4)

sns.despine(offset=5)
plt.show()
```



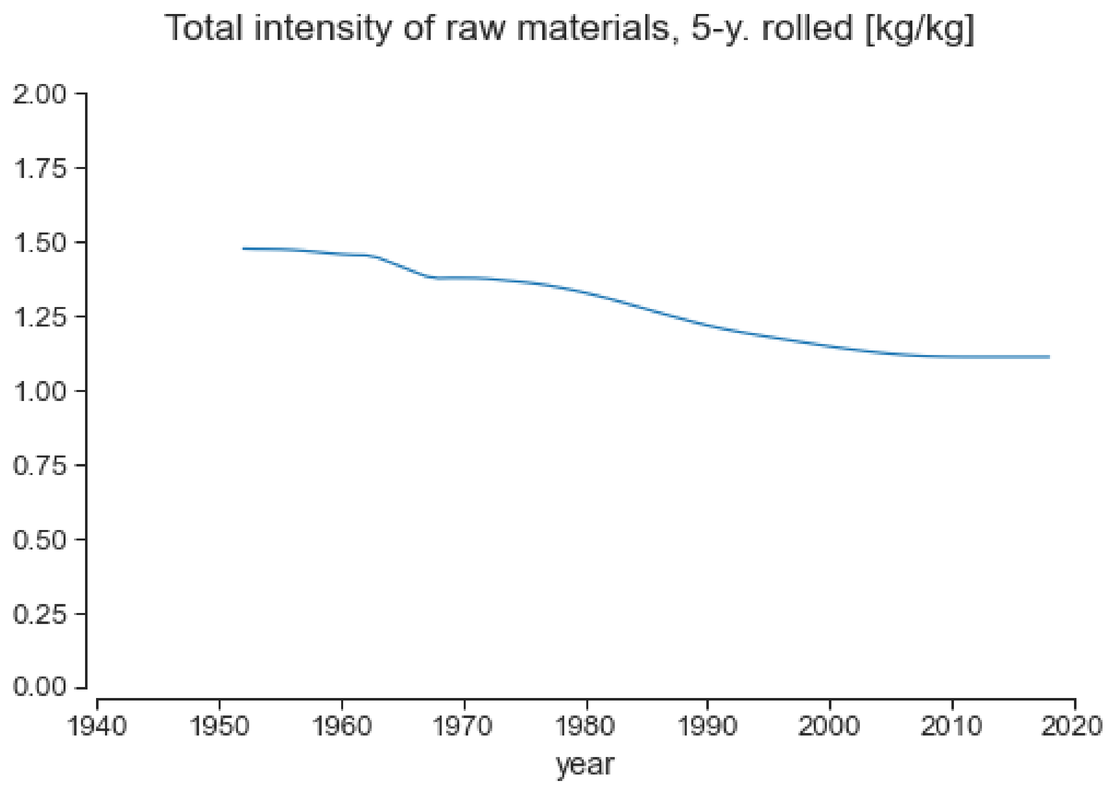
[87]:
```python
# Estimate the material intensity (sum of the intenisities per material):
df_rawmat['Total raw materials, kg/kg'] = 0

for mat in materials:
    df_rawmat['Total raw materials, kg/kg'] = (
        df_rawmat[f"{mat}, kg/kg"]
        + df_rawmat['Total raw materials, kg/kg']
    )
```

[88]:
```python
# Plot the material intensity:
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_rawmat,
             x=df_rawmat.index,
             y=(df_rawmat['Total raw materials, kg/kg']
                .interpolate(method='pchip', limit_direction='forward')),
             ax=ax)

ax.yaxis.label.set_visible(False)
ax.set_xlim(1940, 2020)
ax.set_ylim(0, 2)

fig.suptitle("Total intensity of raw materials [kg/kg]")

sns.despine(offset=5)
plt.show()
```



[89]:
```python
# Plot a 5-y rolled material intensity:
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_rawmat,
```

```
             x=df_rawmat.index,
             y=(df_rawmat['Total raw materials, kg/kg']
                .interpolate(method='pchip', limit_direction='forward')
                 .rolling(5, center=True).mean()),
             ax=ax)

ax.yaxis.label.set_visible(False)
ax.set_xlim(1940, 2020)
ax.set_ylim(0, 2)

fig.suptitle("Total intensity of raw materials, 5-y. rolled [kg/kg]")

sns.despine(offset=5)
plt.show()
```



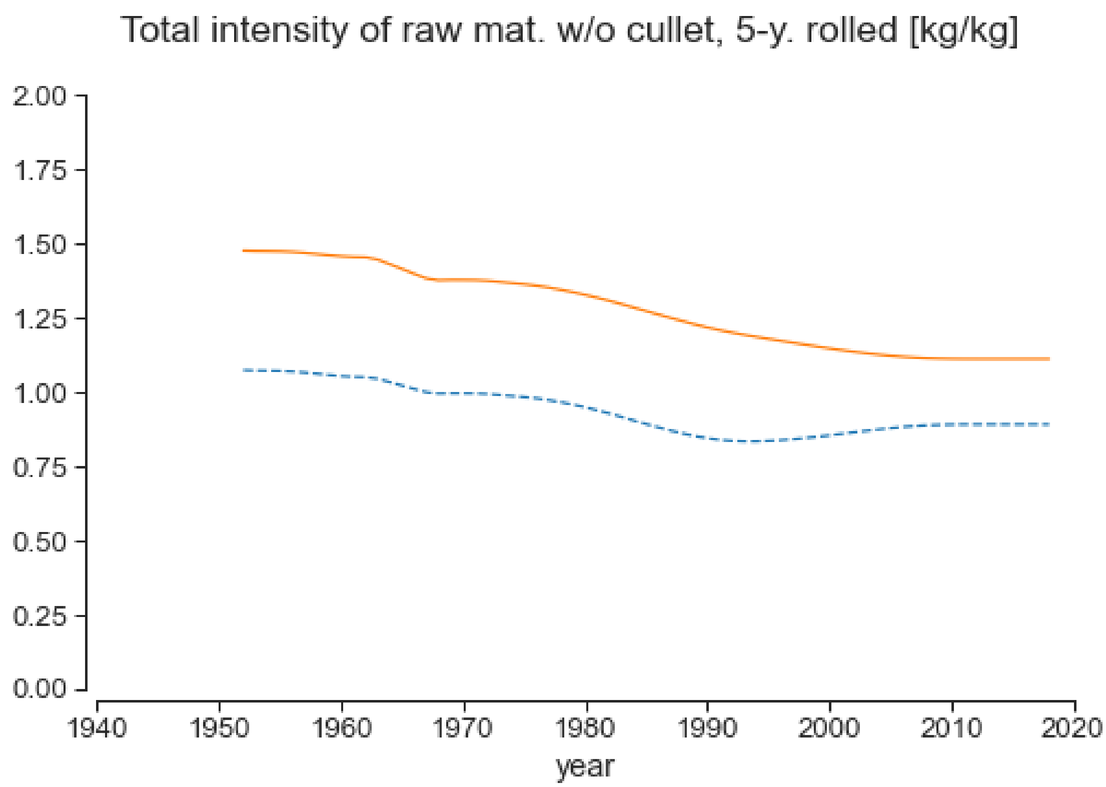
[90]:
```
matwocullet = ['sand', 'sodium carbonate', 'limestone',
               'dolomite', 'feldspar']
```

[91]:
```
# Estimate the material intensity without cullet:
df_rawmat['Total raw mat w/o cullet, kg/kg'] = 0

for mat in matwocullet:
```

```
    df_rawmat['Total raw mat w/o cullet, kg/kg'] = (
        df_rawmat[f"{mat}, kg/kg"]
        + df_rawmat['Total raw mat w/o cullet, kg/kg']
    )
```

[92]:
```
# Alternatively:
df_rawmat['Total raw mat w/o cullet, kg/kg'] = df_rawmat[[
    f"{mat}, kg/kg" for mat in matwocullet]].dropna(how='any', axis=0).
 ↪sum(axis=1)
```

[93]:
```
# Plot a 5-y rolled material intensity:
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_rawmat,
             x=df_rawmat.index,
             y=(df_rawmat['Total raw mat w/o cullet, kg/kg']
                .interpolate(method='pchip', limit_direction='forward')
                 .rolling(5, center=True).mean()),
             ax=ax, linestyle='--')

sns.lineplot(data=df_rawmat,
             x=df_rawmat.index,
             y=(df_rawmat['Total raw materials, kg/kg']
                .interpolate(method='pchip', limit_direction='forward')
                 .rolling(5, center=True).mean()),
             ax=ax)

ax.yaxis.label.set_visible(False)
ax.set_xlim(1940, 2020)
ax.set_ylim(0, 2)

fig.suptitle("Total intensity of raw mat. w/o cullet, 5-y. rolled [kg/kg]")

sns.despine(offset=5)
plt.show()
```



5.2 Energy Intensity since 1945, France and Belgium

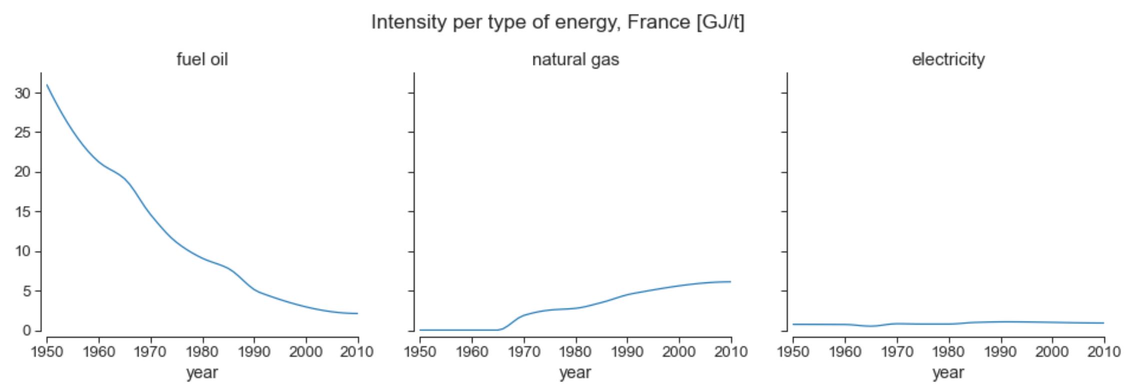
[94]:
```
# Create a DataFrame for the energy intensity in France:
df_fr_energy = fr_data.parse('Energy_Intensity').set_index('year')

df_be_energy = be_data.parse('Energy_Intensity').set_index('year')
```

[95]:
```
# Create a unique DataFrame for FR and BE flat glass flows:
df_be_fr_energy = pd.concat(
    [df_fr_energy['fuel oil, GJ/t'],
     df_be_energy['fuel oil, GJ/t'],
     df_fr_energy['natural gas, GJ/t'],
     df_be_energy['natural gas, GJ/t'],
     df_fr_energy['electricity, GJ/t'],
     df_be_energy['electricity, GJ/t']],
    axis=1, keys=[('fuel oil, GJ/t', 'France'),
                  ('fuel oil, GJ/t', 'Belgium'),
                  ('natural gas, GJ/t', 'France'),
                  ('natural gas, GJ/t', 'Belgium'),
                  ('electricity, GJ/t', 'France'),
                  ('electricity, GJ/t', 'Belgium')],
    names=['Info', 'Country']
```

```
).stack(dropna=False)

df_be_fr_energy = df_be_fr_energy.unstack()
```

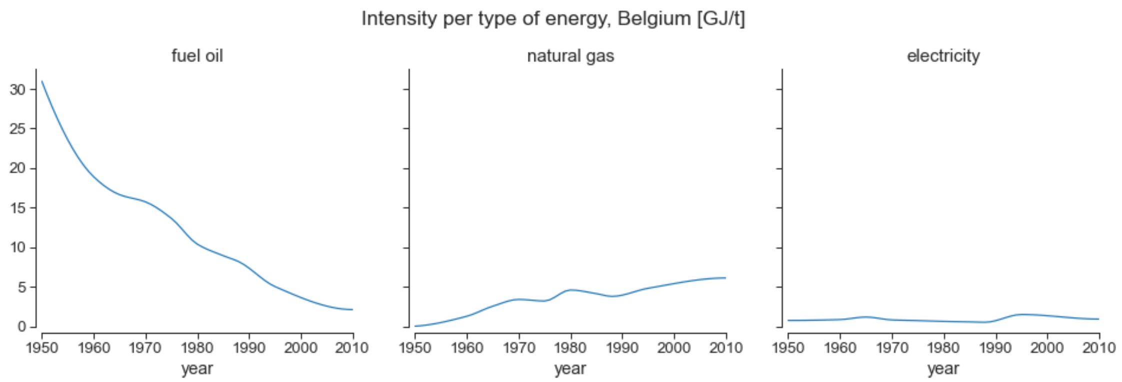
[96]:
```
energies = ['fuel oil', 'natural gas', 'electricity']
```

[97]:
```
# Plot the intensity per type of energy, France:
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 4))

for i, (ax, energy) in enumerate(zip(axes.flatten(), energies)):
    ax.set_title(energy)
    (df_be_fr_energy[(f"{energy}, GJ/t", "France")]  # Interpolate:
     .interpolate(method='pchip', limit_direction='forward')
     .plot(ax=ax))

ax.set_xlim(1950, 2010)
ax.set_ylim(ymin=0)

fig.suptitle("Intensity per type of energy, France [GJ/t]", y=1.05)

sns.despine(offset=5)
plt.show()
```



[98]:
```
# Plot the intensity per type of energy, Belgium:
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 4))

for i, (ax, energy) in enumerate(zip(axes.flatten(), energies)):
    ax.set_title(energy)
    (df_be_fr_energy[(f"{energy}, GJ/t", "Belgium")]  # Interpolate:
```

```
    .interpolate(method='pchip', limit_direction='forward')
    .plot(ax=ax))

ax.set_xlim(1950, 2010)
ax.set_ylim(ymin=0)

fig.suptitle("Intensity per type of energy, Belgium [GJ/t]", y=1.05)

sns.despine(offset=5)
plt.show()
```

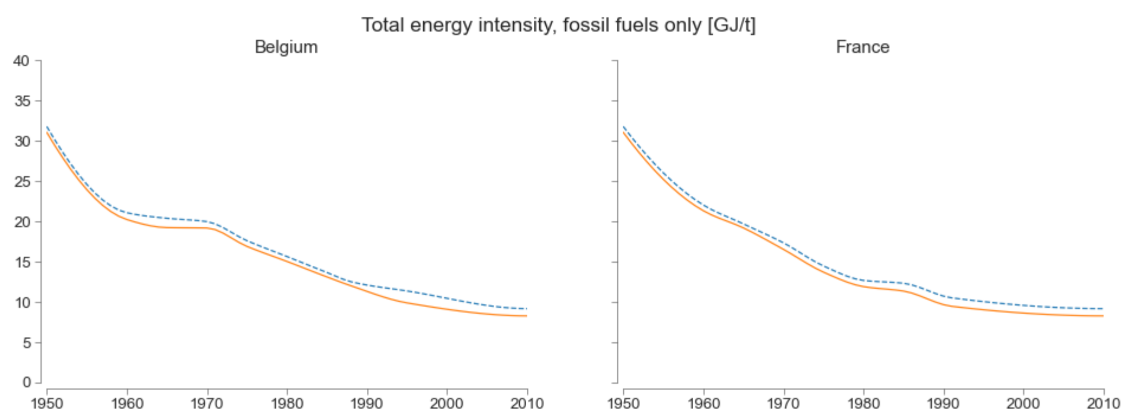


[99]:
```
# Estimate the total energy intensity
# (sum of the intensities per type of energy):
for country in countries:
    df_be_fr_energy[('Total energy, GJ/t', country)] = 0
    for energy in energies:
        df_be_fr_energy[('Total energy, GJ/t', country)] = (
            df_be_fr_energy[('Total energy, GJ/t', country)]
            + df_be_fr_energy[(f"{energy}, GJ/t", country)]
        )
```

[100]:
```
fossilfuels = ['fuel oil', 'natural gas']
```

[101]:
```
# Estimate the total energy intensity
# (without electricity)
for country in countries:
    df_be_fr_energy[('Total energy w/o elec., GJ/t', country)] = 0
    for energy in fossilfuels:
        df_be_fr_energy[('Total energy w/o elec., GJ/t', country)] = (
            df_be_fr_energy[(
                'Total energy w/o elec., GJ/t', country)]
            + df_be_fr_energy[(f"{energy}, GJ/t", country)]
        )
```

[102]:
```
# Plot the total energy intensity:
fig, axes = plt.subplots(nrows=1, ncols=2,
                         sharex=True, sharey=True,
                         figsize=(16, 5))

for i, (ax, country) in enumerate(zip(axes.flatten(), countries)):
    sns.lineplot(data=df_be_fr_energy,
                 x=df_be_fr_energy.index,
                 y=(df_be_fr_energy[('Total energy, GJ/t', country)]
                    .interpolate(method='pchip', limit_direction='forward')),
                 ax=ax, linestyle='--')

    sns.lineplot(data=df_be_fr_energy,
                 x=df_be_fr_energy.index,
                 y=(df_be_fr_energy[('Total energy w/o elec., GJ/t',
                                     country)]
                    .interpolate(method='pchip', limit_direction='forward')),
                 ax=ax)

    ax.set_xlim(1950, 2010)
    ax.set_ylim(0, 40)
    ax.set_title(country)
    ax.yaxis.label.set_visible(False)
    style_ax(ax)


fig.suptitle("Total energy intensity, fossil fuels only [GJ/t]")

sns.despine(offset=5)
plt.show()
```



[103]:

```python
# Estimate energy intensity according to ratios:
for energy in energies:
    for country in countries:
        df_be_fr_energy[(f"{energy}, %", country)] = (
            (df_be_fr_energy[(f"{energy}, GJ/t", country)]
             .interpolate(method='pchip', limit_direction='forward'))
            / (df_be_fr_energy[('Total energy, GJ/t', country)]
               .interpolate(method='pchip', limit_direction='forward'))
        )
```

[104]:

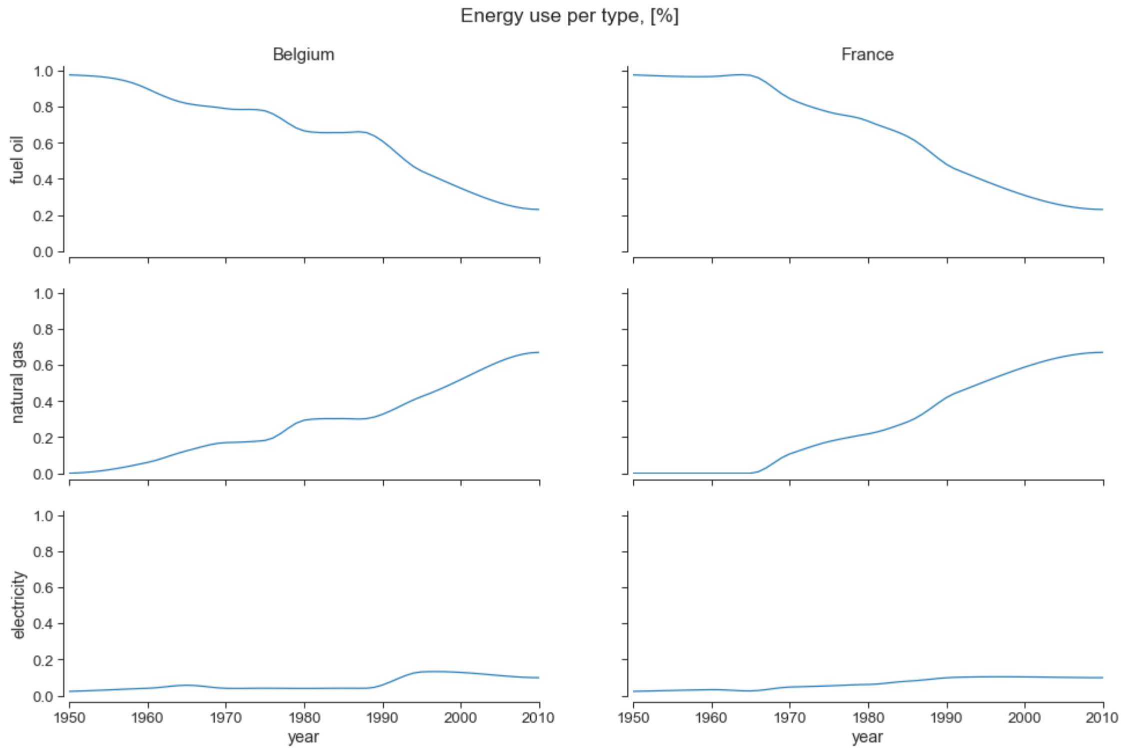
```python
df_be_fr_energy
```

[104]:

```
Info    electricity, GJ/t        fuel oil, GJ/t        natural gas, GJ/t  \
Country           Belgium France         Belgium France           Belgium
year
1945                  NaN    NaN             NaN    NaN               NaN
1946                  NaN    NaN             NaN    NaN               NaN
1947                  NaN    NaN             NaN    NaN               NaN
1948                  NaN    NaN             NaN    NaN               NaN
1949                  NaN    NaN             NaN    NaN               NaN
…                    …     …              …     …                 …
2016                  NaN    NaN             NaN    NaN               NaN
2017                  NaN    NaN             NaN    NaN               NaN
2018                  NaN    NaN             NaN    NaN               NaN
2019                  0.9    0.9             2.1    2.1               6.1
2020                  NaN    NaN             NaN    NaN               NaN

Info           Total energy, GJ/t        Total energy w/o elec., GJ/t         \
Country France            Belgium France                      Belgium France
year
1945       NaN                NaN    NaN                          NaN    NaN
1946       NaN                NaN    NaN                          NaN    NaN
1947       NaN                NaN    NaN                          NaN    NaN
1948       NaN                NaN    NaN                          NaN    NaN
1949       NaN                NaN    NaN                          NaN    NaN
…         …                 …     …                            …     …
2016       NaN                NaN    NaN                          NaN    NaN
2017       NaN                NaN    NaN                          NaN    NaN
2018       NaN                NaN    NaN                          NaN    NaN
2019       6.1                9.1    9.1                          8.2    8.2
2020       NaN                NaN    NaN                          NaN    NaN

Info    fuel oil, %         natural gas, %        electricity, %
Country     Belgium France         Belgium France        Belgium France
year
1945            NaN    NaN             NaN    NaN            NaN    NaN
1946            NaN    NaN             NaN    NaN            NaN    NaN
```

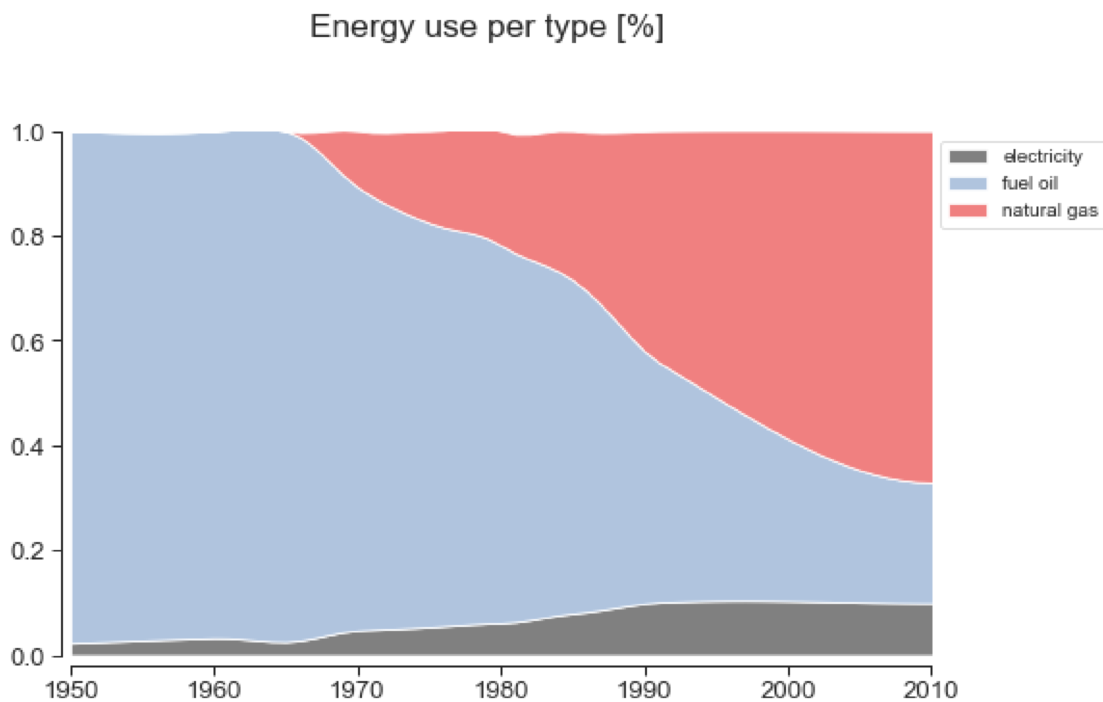
```
1947            NaN    NaN            NaN    NaN            NaN    NaN
1948            NaN    NaN            NaN    NaN            NaN    NaN
1949            NaN    NaN            NaN    NaN            NaN    NaN
…             …    …          …    …          …    …
2016           0.23   0.23           0.67   0.67            0.1    0.1
2017           0.23   0.23           0.67   0.67            0.1    0.1
2018           0.23   0.23           0.67   0.67            0.1    0.1
2019           0.23   0.23           0.67   0.67            0.1    0.1
2020           0.23   0.23           0.67   0.67            0.1    0.1

[76 rows x 16 columns]
```

[105]:
```
# Plot energy intensity per type of energy and in percent:
fig, axes = plt.subplots(nrows=3, ncols=2,
                         sharex=True, sharey=True,
                         figsize=(16, 10))

for row, energy in enumerate(energies):
    for col, country in enumerate(countries):
        ax = axes[row][col]
        df_be_fr_energy[(f"{energy}, %", country)].plot(ax=ax)

        if row == 0:
            ax.set_title(country)
        if col == 0:
            ax.set_ylabel(energy)

ax.set_xlim(1950, 2010)
ax.set_ylim(ymin=0)

fig.suptitle("Energy use per type, [%]", y=0.95)

sns.despine(offset=5)
plt.show()
```



[106]:

```
# Plot energy intensity per type of energy and in percent:
fig, ax = plt.subplots(figsize=(8, 5))

y = np.array([df_be_fr_energy[('electricity, %', 'France')].dropna(),
              df_be_fr_energy[('fuel oil, %', 'France')].dropna(),
              df_be_fr_energy[('natural gas, %', 'France')].dropna()])

x = range(1950, 2021)

plt.stackplot(x, y,
              labels=['electricity', 'fuel oil', 'natural gas'],
              colors=["grey", "lightsteelblue", "lightcoral"])
plt.legend(loc='upper left')

ax.set_xlim(1950, 2010)
ax.set_ylim(0, 1)
plt.legend(bbox_to_anchor=(1, 1), loc='upper left')

fig.suptitle("Energy use per type [%]", y=1.05)

sns.despine(offset=5)
plt.show()
```



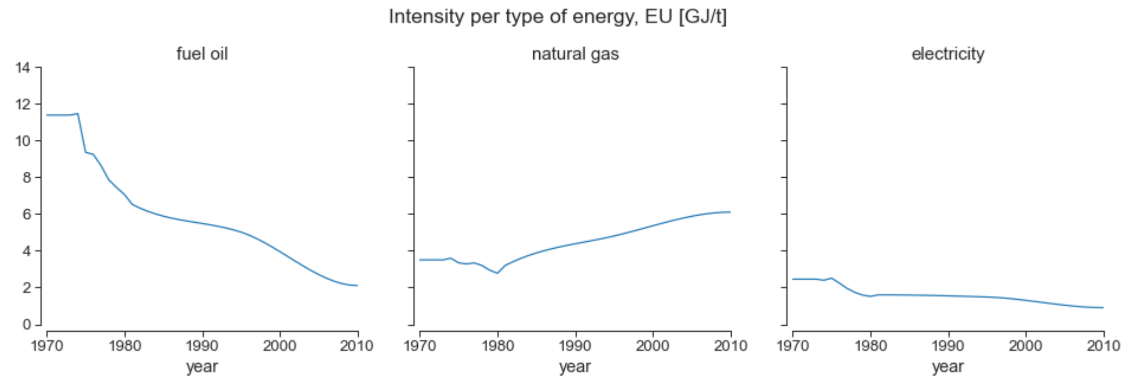
5.3 Energy Intensity in the EU since 1970

[107]:
```
# Create a DataFrame for the energy intensity in the EU:
df_eu_energy = eu_data.parse('Energy_Intensity').set_index('year')
```

[108]:
```
energies = ['fuel oil', 'natural gas', 'electricity']
```

[109]:
```
# Plot the intensity per type of energy:
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 4))

for i, (ax, energy) in enumerate(zip(axes.flatten(), energies)):
    ax.set_title(energy)
    (df_eu_energy[f"{energy}, GJ/t"]  # Interpolate:
     .interpolate(method='pchip', limit_direction='forward')
     .plot(ax=ax))

ax.set_xlim(1970, 2010)
ax.set_ylim(0, 14)

fig.suptitle("Intensity per type of energy, EU [GJ/t]", y=1.05)

sns.despine(offset=5)
```

```
plt.show()
```



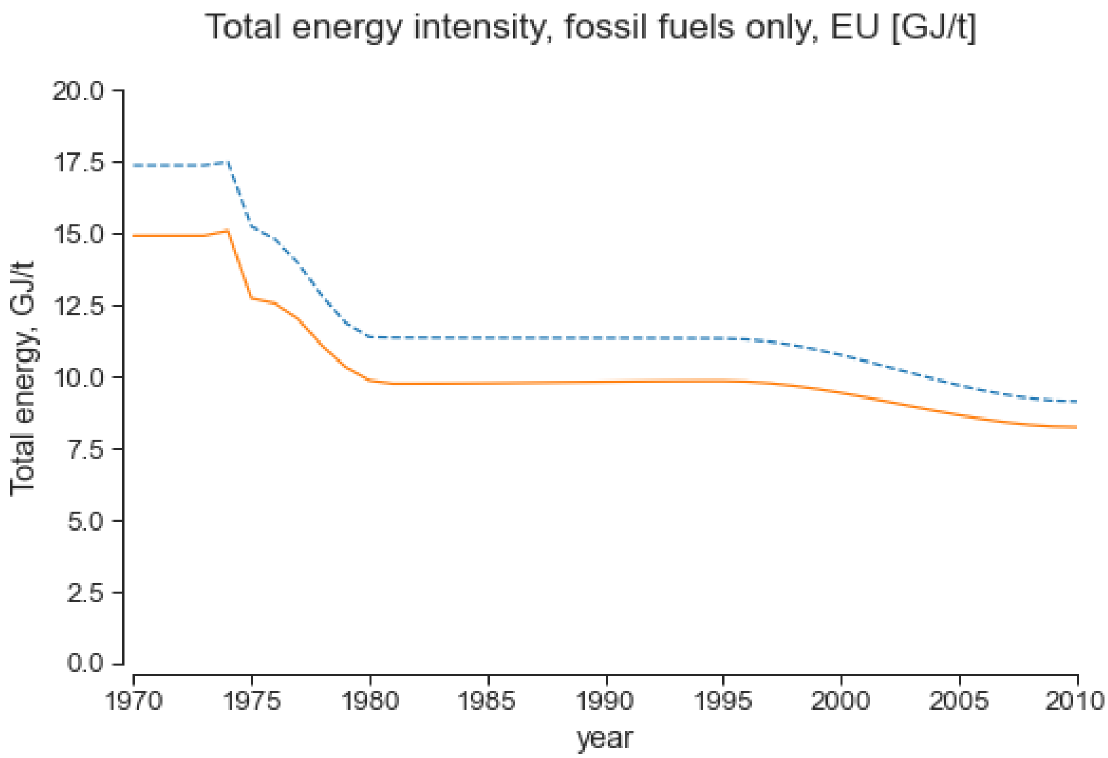
[110]:
```
# Estimate the total energy intensity
# (sum of the intensities per type of energy)
# In Europe:
df_eu_energy['Total energy, GJ/t'] = 0

for energy in energies:
    df_eu_energy['Total energy, GJ/t'] = (
        df_eu_energy['Total energy, GJ/t']
        + df_eu_energy[f"{energy}, GJ/t"]
    )
```

[111]:
```
fossilfuels = ['fuel oil', 'natural gas']
```

[112]:
```
# Estimate the total energy intensity
# (without electricity)
df_eu_energy['Total energy w/o elec., GJ/t'] = 0

for energy in fossilfuels:
    df_eu_energy['Total energy w/o elec., GJ/t'] = (
        df_eu_energy['Total energy w/o elec., GJ/t']
        + df_eu_energy[f"{energy}, GJ/t"]
    )
```

[113]:
```
# Plot the total energy intensity:
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_eu_energy,
             x=df_eu_energy.index,
             y=(df_eu_energy['Total energy, GJ/t']
                .interpolate(method='pchip', limit_direction='forward')),
             ax=ax, linestyle='--')
```

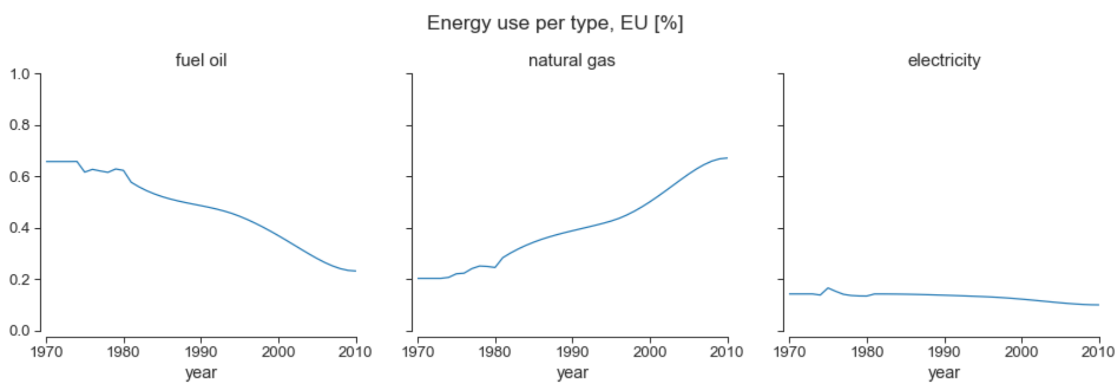
```python
sns.lineplot(data=df_eu_energy,
             x=df_eu_energy.index,
             y=(df_eu_energy['Total energy w/o elec., GJ/t']
                .interpolate(method='pchip', limit_direction='forward')),
             ax=ax)

ax.set_xlim(1970, 2010)
ax.set_ylim(0, 20)

fig.suptitle("Total energy intensity, fossil fuels only, EU [GJ/t]")

sns.despine(offset=5)  # , left=True, bottom=True)
plt.show()
```



[114]:
```python
# Estimate energy intensity according to ratios:
for energy in energies:
    df_eu_energy[f"{energy}, %"] = (
        (df_eu_energy[(f"{energy}, GJ/t")]
         .interpolate(method='pchip', limit_direction='forward'))
        / (df_eu_energy['Total energy, GJ/t']
           .interpolate(method='pchip', limit_direction='forward'))
```

```
    )
```

[115]:
```
# Plot energy intensity per type of energy and in percent:
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 4))

for i, (ax, energy) in enumerate(zip(axes.flatten(), energies)):
    ax.set_title(energy)
    df_eu_energy[f"{energy}, %"].plot(ax=ax)

ax.set_xlim(1970, 2010)
ax.set_ylim(0, 1)

fig.suptitle("Energy use per type, EU [%]", y=1.05)

sns.despine(offset=5)
plt.show()
```



5.4 Carbon Intensity of the Flat Glass Manufacturing Process in France

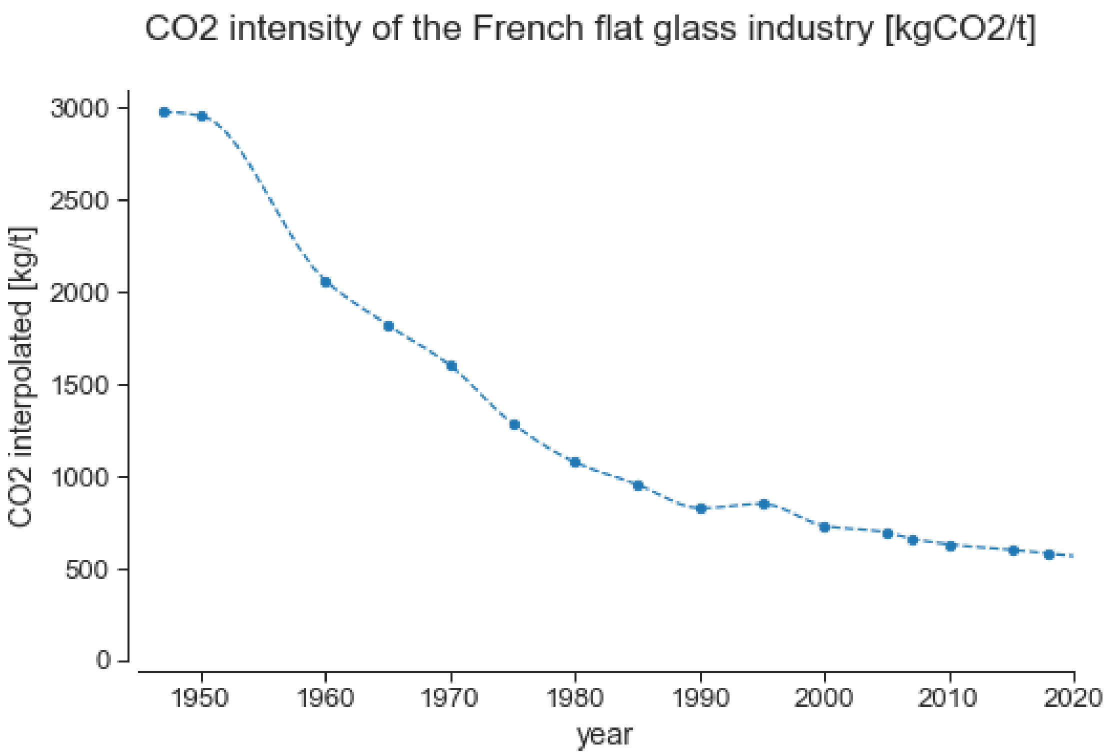
[116]:
```
# Create a DataFrame only with the CO2 emission data:
fr_emissions = fr_data.parse('emissions').set_index('year')

fr_co2_intensity = fr_emissions[['CO2 glass ind, kg/t']].copy()
fr_co2_intensity.rename(columns={'CO2 glass ind, kg/t': 'CO2 [kg/t]'},
                        inplace=True)

fr_co2_intensity.dropna()
```

[116]:
```
      CO2 [kg/t]
year
1947      2970.0
```

```
1950      2950.0
1960      2055.0
1965      1815.0
1970      1595.0
1975      1280.0
1980      1070.0
1985       950.0
1990       820.0
1995       845.0
2000       725.0
2005       690.0
2007       655.0
2010       625.0
2015       595.0
2018       575.0
```

[117]:
```
# Interpolate CO2 intensity accoring to a pchip method:
fr_co2_intensity['CO2 interpolated [kg/t]'] = (
    fr_co2_intensity['CO2 [kg/t]'].interpolate('pchip')
)
```

[118]:
```
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=fr_co2_intensity['CO2 interpolated [kg/t]'])

ax.lines[0].set_linestyle("--")

ax.scatter(x=fr_co2_intensity.index, y=fr_co2_intensity['CO2 [kg/t]'],
           s=20)

ax.set_xlim(1945, 2020)
ax.set_ylim(ymin=0)
fig.suptitle("CO2 intensity of the French flat glass industry [kgCO2/t]")
sns.despine(offset=5)
plt.show()
```



5.5 Carbon Intensity of the Flat Glass Manufacturing Process in the EU

[119]:
```python
# Create a DataFrame only with the CO2 emission data:
df_eu_emissions = eu_data.parse('emissions').set_index('year')

df_eu_co2 = df_eu_emissions[['CO2 glass ind, kg/t']].copy()
df_eu_co2.rename(columns={'CO2 glass ind, kg/t': 'CO2 [kg/t]'},
                 inplace=True)

df_eu_co2.dropna()
```

[119]:

year	CO2 [kg/t]
1970	1605.74
1975	1364.88
1980	1204.38
1985	1083.90
1990	1043.78
1991	1012.47
1992	986.35
1993	960.22
1994	929.03

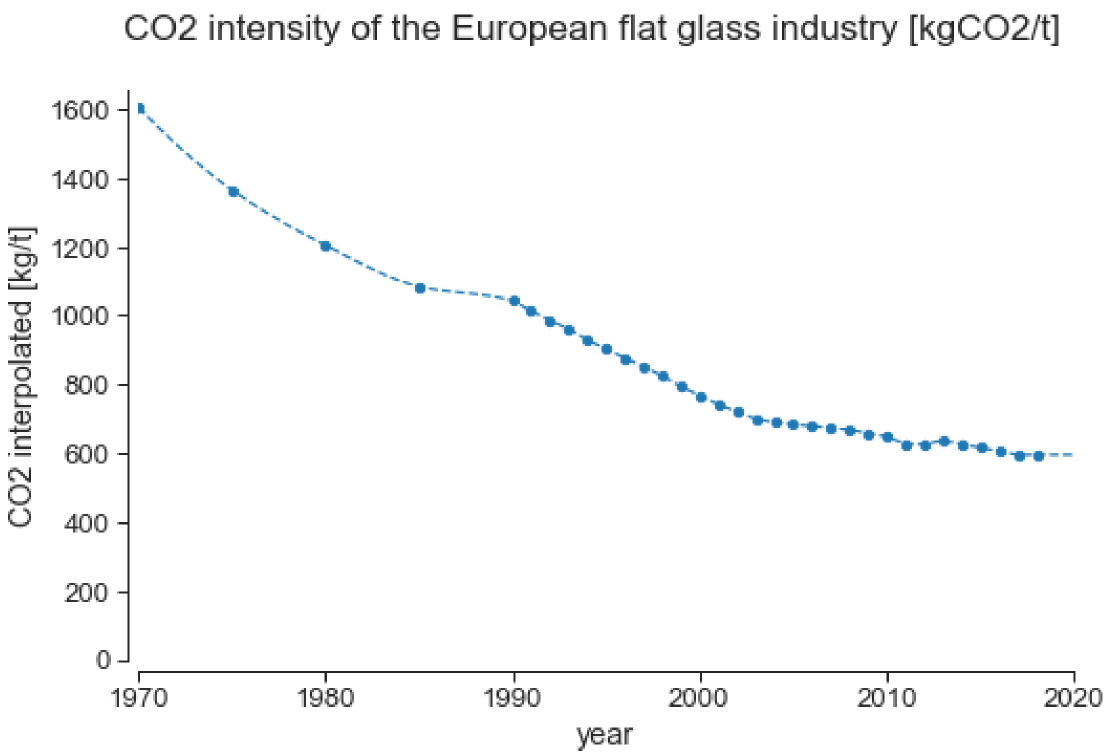
```
1995      902.79
1996      876.78
1997      850.65
1998      824.53
1999      793.22
2000      767.21
2001      741.08
2002      720.14
2003      699.31
2004      688.94
2005      683.65
2006      678.47
2007      673.18
2008      668.00
2009      657.53
2010      647.06
2011      626.22
2012      626.22
2013      636.69
2014      626.22
2015      615.86
2016      605.39
2017      594.92
2018      594.92
```

[120]:

```
# Interpolate CO2 intensity accoring to a pchip method:
df_eu_co2['CO2 interpolated [kg/t]'] = (
    df_eu_co2['CO2 [kg/t]'].interpolate('pchip')
)
```

[121]:

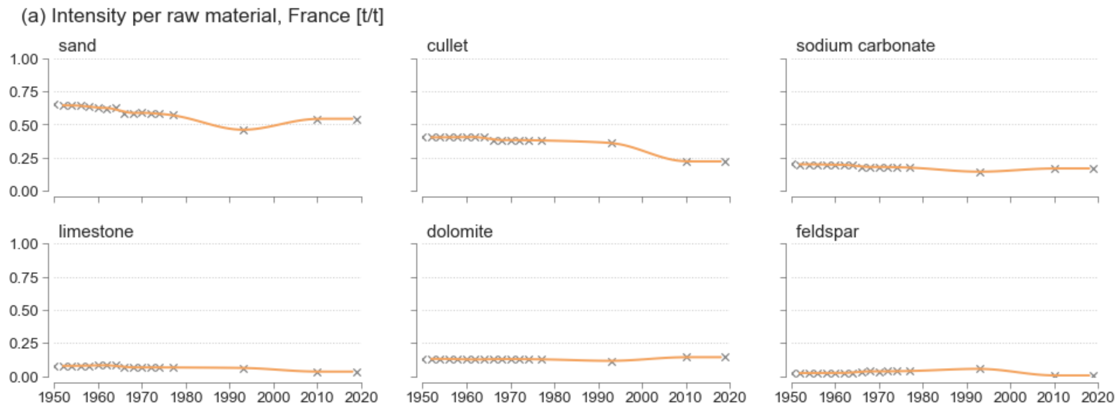
```
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_eu_co2['CO2 interpolated [kg/t]'])

ax.lines[0].set_linestyle("--")

ax.scatter(x=df_eu_co2.index, y=df_eu_co2['CO2 [kg/t]'],
           s=20)

ax.set_xlim(1970, 2020)
ax.set_ylim(ymin=0)
fig.suptitle("CO2 intensity of the European flat glass industry [kgCO2/t]")
sns.despine(offset=5)
plt.show()
```



5.6 Summary

[122]:
```python
# Plot the material instensity per raw material:
fig, axes = plt.subplots(nrows=2, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 5))

for i, (ax, mat) in enumerate(zip(axes.flatten(), materials)):
    (df_rawmat[f"{mat}, kg/kg"]
     .interpolate(method='pchip', limit_direction='forward')
     .rolling(5, center=True).mean()
     .plot(ax=ax, c='sandybrown', linewidth=2))

    ax.scatter(x=df_rawmat.index, y=df_rawmat[f"{mat}, kg/kg"],
               s=40, marker='x', c='grey')
    plt.xticks(np.arange(1950, 2021, 10), rotation=45)

    ax.set_title(f" {mat}", loc='left')
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
    style_ax(ax)

ax.set_xlim(1950, 2020)
```

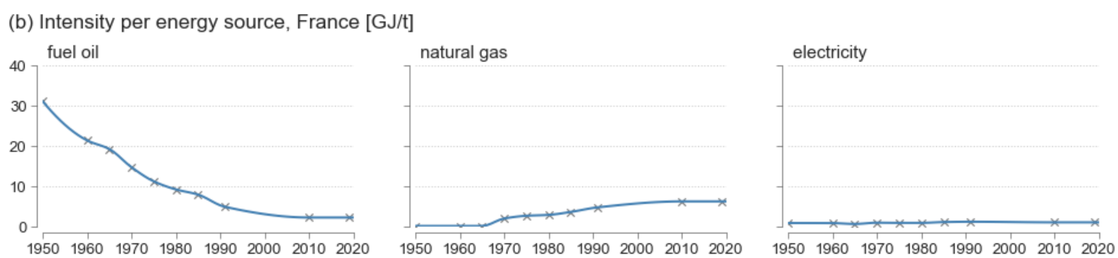
```
ax.set_ylim(0, 1)

ax.minorticks_off()
plt.xticks(np.arange(1950, 2021, 10))

fig.suptitle(
    "(a) Intensity per raw material, France [t/t]", x=0.1, y=1, ha='left')

fig.subplots_adjust(wspace=0.2, hspace=0.4)

sns.despine(offset=5)

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig4_FR_MatIntensity.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig4_FR_MatIntensity.pdf'),
                bbox_inches='tight')

plt.show()
```



[123]:

```
# Plot the energy intensity per source of energy, France:
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 2.5)
                         )

for i, (ax, energy) in enumerate(zip(axes.flatten(), energies)):
    (df_be_fr_energy[(f"{energy}, GJ/t", "France")]
     .interpolate(method='pchip', limit_direction='forward')
     .plot(ax=ax, color='steelblue', linewidth=2))
```

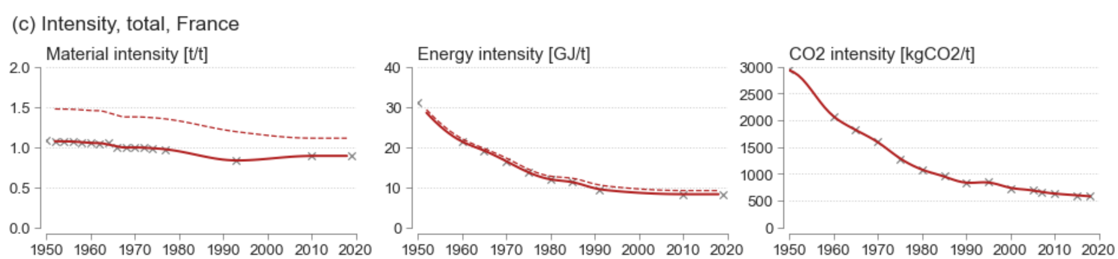
```
    ax.scatter(x=df_be_fr_energy.index,
               y=df_be_fr_energy[(f"{energy}, GJ/t", "France")],
               s=40, marker='x', c='grey')

    ax.set_title(f" {energy}", loc='left')
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
    style_ax(ax)

ax.set_xlim(1950, 2020)
ax.set_ylim(0, 40)

fig.suptitle("(b) Intensity per energy source, France [GJ/t]",
             x=0.1, y=1.12, ha='left')
fig.subplots_adjust(wspace=0.2)

ax.minorticks_off()
plt.xticks(np.arange(1950, 2021, 10))

sns.despine(offset=5)

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig4_FR_EnergyIntensity.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig4_FR_EnergyIntensity.pdf'),
                bbox_inches='tight')

plt.show()
```



[124]:
```
# Plot the intensity for energy, material and CO2 emissions:
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True,
                         figsize=(16, 2.5))

# Material intensity
ax = axes[0]
```

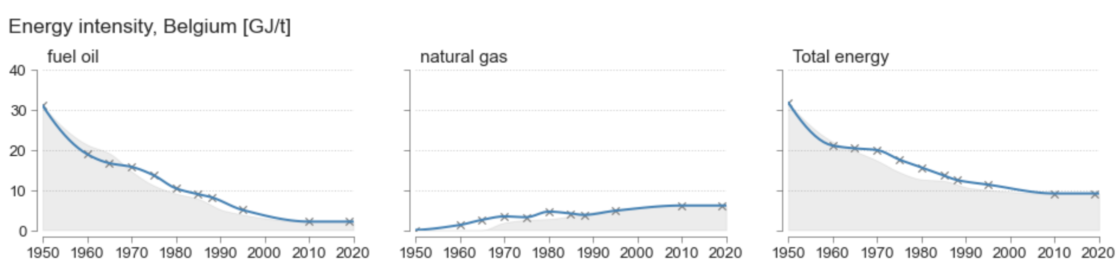
```
(df_rawmat['Total raw materials, kg/kg']
 .interpolate(method='pchip', limit_direction='forward')
 .rolling(5, center=True).mean()
 .plot(ax=ax, c='firebrick', linestyle='--', linewidth=1.25)
 )

(df_rawmat['Total raw mat w/o cullet, kg/kg']
 .interpolate(method='pchip', limit_direction='forward')
 .rolling(5, center=True).mean()
 .plot(ax=ax, c='firebrick', linestyle='-', linewidth=2)
 )

ax.scatter(x=df_rawmat.index,
           y=df_rawmat['Total raw mat w/o cullet, kg/kg'],
           s=40, marker='x', c='grey')

ax.set_title("Material intensity [t/t]", loc='left')
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
style_ax(ax)
ax.set_ylim(0, 2)
ax.set_xlim(1950, 2020)

# Energy intensity
ax = axes[1]
(df_be_fr_energy[('Total energy, GJ/t', 'France')]
 .interpolate(method='pchip', limit_direction='forward')
 .rolling(5, center=True).mean()
 .plot(ax=ax, c='firebrick', linestyle='--', linewidth=1.25))

(df_be_fr_energy[('Total energy w/o elec., GJ/t', 'France')]
 .interpolate(method='pchip', limit_direction='forward')
 .rolling(5, center=True).mean()
 .plot(ax=ax, c='firebrick', linewidth=2))

ax.scatter(x=df_be_fr_energy.index,
           y=df_be_fr_energy[(
               'Total energy w/o elec., GJ/t', 'France')],
           s=40, marker='x', c='grey')

ax.set_title("Energy intensity [GJ/t]", loc='left')
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
style_ax(ax)
ax.set_ylim(0, 40)

# Carbon dioxide intensity
ax = axes[2]
(fr_co2_intensity['CO2 interpolated [kg/t]'].rolling(5, center=True).mean()
```

```
 .plot(ax=ax, c='firebrick', linewidth=2))

ax.scatter(x=fr_co2_intensity.index, y=fr_co2_intensity['CO2 [kg/t]'],
           s=40, marker='x', c='grey')

ax.set_title("CO2 intensity [kgCO2/t]", loc='left')
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
style_ax(ax)
ax.set_ylim(0, 3000)

ax.minorticks_off()
plt.xticks(np.arange(1950, 2021, 10))
plt.yticks(np.arange(0, 3001, 500))

fig.suptitle("(c) Intensity, total, France", x=0.1, y=1.12, ha='left')
fig.subplots_adjust(wspace=0.20)

sns.despine(offset=5)

if export:
    # Save image:
    plt.savefig(os.path.join(path_img, 'Fig4_FR_TotalIntensity.png'),
                dpi=600, bbox_inches='tight')
    plt.savefig(os.path.join(path_img, 'Fig4_FR_TotalIntensity.pdf'),
                bbox_inches='tight')

plt.show()
```



[125]:
```
toplot = ['fuel oil', 'natural gas', 'Total energy']

# Plot the energy intensity for Belgium:
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 2.5)
                         )
```

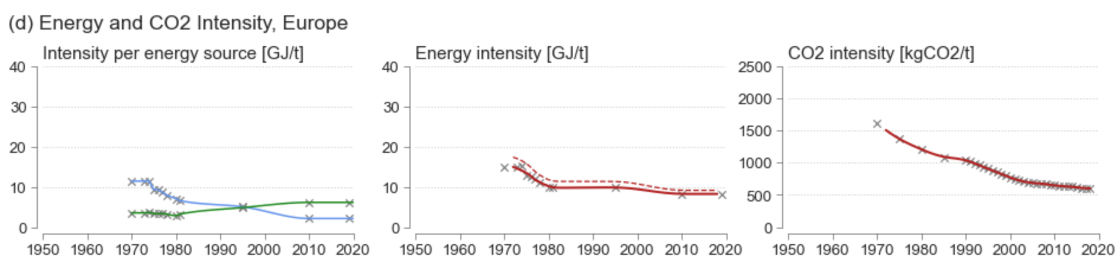
```
for i, (ax, plot) in enumerate(zip(axes.flatten(), toplot)):
    (df_be_fr_energy[(f"{plot}, GJ/t", "Belgium")]
     .interpolate(method='pchip', limit_direction='forward')
     .plot(ax=ax, color='steelblue', linewidth=2))

    ax.scatter(x=df_be_fr_energy.index,
               y=df_be_fr_energy[(f"{plot}, GJ/t", "Belgium")],
               s=40, marker='x', c='grey')

    x = (df_be_fr_energy.reset_index()['year'])
    y = (df_be_fr_energy[(f"{plot}, GJ/t", "France")]
         .interpolate(method='pchip', limit_direction='forward'))
    ax.fill_between(x, y, color="grey", alpha=0.15)

    ax.set_title(f" {plot}", loc='left')
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
    style_ax(ax)

ax.set_xlim(1950, 2020)
ax.set_ylim(0, 40)

fig.suptitle("Energy intensity, Belgium [GJ/t]",
             x=0.1, y=1.12, ha='left')
fig.subplots_adjust(wspace=0.2)

ax.minorticks_off()
plt.xticks(np.arange(1950, 2021, 10))

sns.despine(offset=5)

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig5_BE_EnergyIntensity.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig5_BE_EnergyIntensity.pdf'),
                bbox_inches='tight')

plt.show()
```



[126]:

```
# Plot the intensity for energy, material and CO2 emissions:
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True,
                         figsize=(16, 2.5))

# Intensity, per energy source
ax = axes[0]
(df_eu_energy["fuel oil, GJ/t"]
 .interpolate(method='pchip', limit_direction='forward')
 .plot(ax=ax, color='cornflowerblue', linestyle='-', linewidth=1.5))

ax.scatter(x=df_eu_energy.index,
           y=df_eu_energy["fuel oil, GJ/t"],
           s=40, marker='x', c='grey')

(df_eu_energy["natural gas, GJ/t"]
 .interpolate(method='pchip', limit_direction='forward')
 .plot(ax=ax, color='forestgreen', linewidth=1.5, linestyle='-'))

ax.scatter(x=df_eu_energy.index,
           y=df_eu_energy["natural gas, GJ/t"],
           s=40, marker='x', c='grey')

ax.set_title("Intensity per energy source [GJ/t]", loc='left')
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
style_ax(ax)
ax.set_ylim(0, 40)
ax.set_xlim(1950, 2020)

# Energy intensity, total
ax = axes[1]
(df_eu_energy['Total energy, GJ/t']
 .interpolate(method='pchip', limit_direction='forward')
 .rolling(5, center=True).mean()
 .plot(ax=ax, c='firebrick', linestyle='--', linewidth=1.25))

(df_eu_energy['Total energy w/o elec., GJ/t']
 .interpolate(method='pchip', limit_direction='forward')
 .rolling(5, center=True).mean()
 .plot(ax=ax, c='firebrick', linewidth=2))

ax.scatter(x=df_eu_energy.index,
           y=df_eu_energy['Total energy w/o elec., GJ/t'],
           s=40, marker='x', c='grey')
```

```python
ax.set_title("Energy intensity [GJ/t]", loc='left')
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
style_ax(ax)
ax.set_ylim(0, 40)

# Carbon dioxide intensity
ax = axes[2]
(df_eu_co2['CO2 interpolated [kg/t]'].rolling(5, center=True).mean()
 .plot(ax=ax, c='firebrick', linewidth=2))

ax.scatter(x=df_eu_co2.index, y=df_eu_co2['CO2 [kg/t]'],
           s=40, marker='x', c='grey')

ax.set_title("CO2 intensity [kgCO2/t]", loc='left')
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
style_ax(ax)
ax.set_ylim(0, 2500)

ax.minorticks_off()
plt.xticks(np.arange(1950, 2021, 10))

fig.suptitle("(d) Energy and CO2 Intensity, Europe", x=0.1, y=1.12, ha='left')
fig.subplots_adjust(wspace=0.20)

sns.despine(offset=5)

if export:
    # Save image:
    plt.savefig(os.path.join(path_img, 'Fig6_EU_Intensity.png'),
                dpi=600, bbox_inches='tight')
    plt.savefig(os.path.join(path_img, 'Fig6_EU_Intensity.pdf'),
                bbox_inches='tight')

plt.show()
```

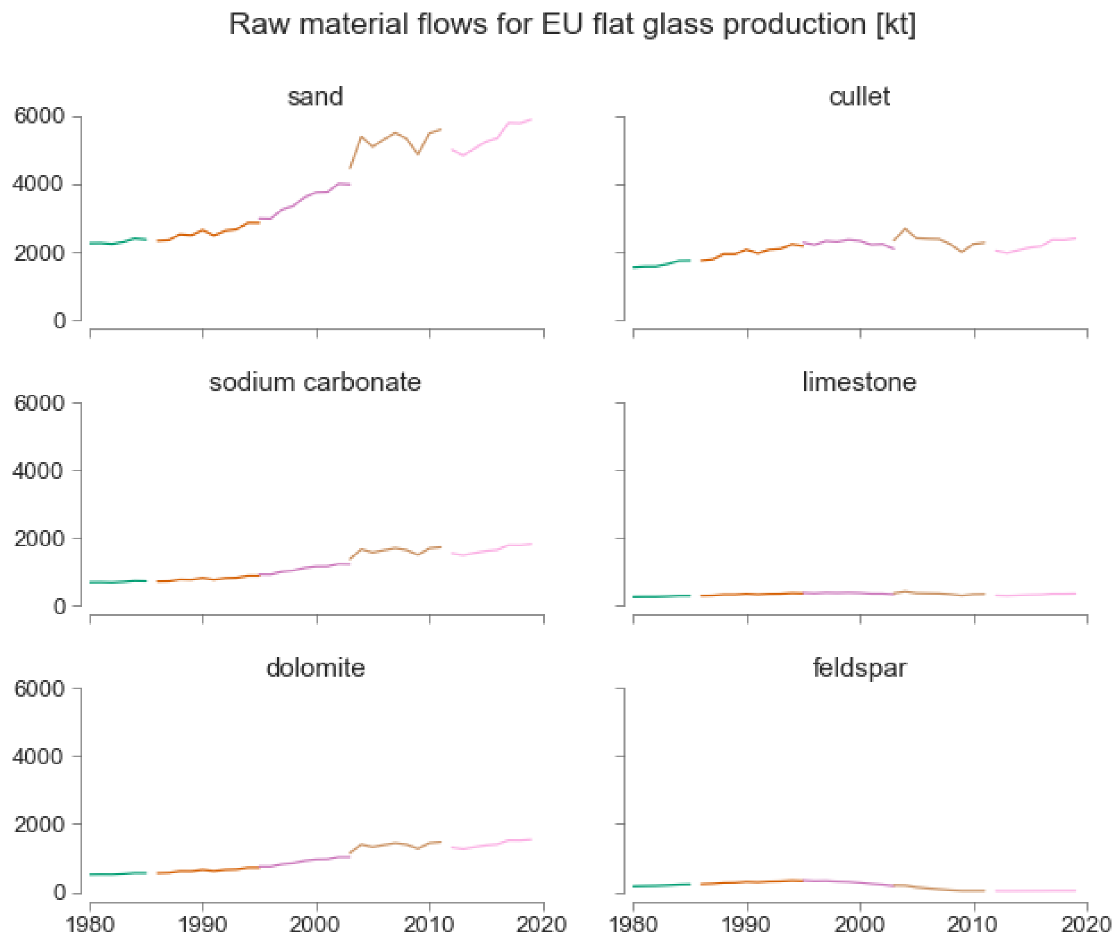


6 Raw Material and Energy Use and CO2 Emissions in the European Flat Glass Industry

6.1 Use of Raw Materials in the Production of Flat Glass

[127]:
```
# Absolute use of raw materials for fat glass production:
for mat in materials:
    df_eu[f"{mat}, flat glass [kt]"] = (
        df_eu['Production [kt]']
        * (df_rawmat[f"{mat}, kg/kg"]
           .interpolate(method='pchip', limit_direction='forward'))
    )
```

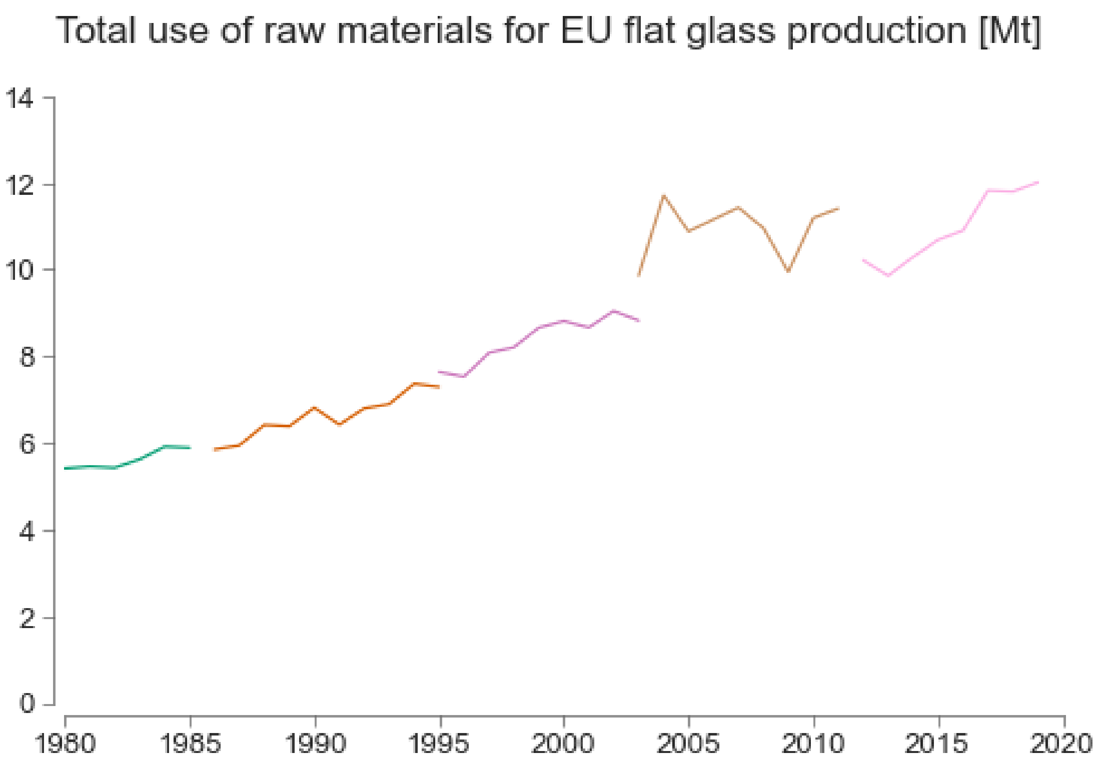
[128]:
```
# Plot raw material flows for EU flat glass production
fig, axes = plt.subplots(nrows=3, ncols=2,
                         sharex=True, sharey=True,
                         figsize=(10, 8))

for i, (ax, material) in enumerate(zip(axes.flatten(), materials)):
    sns.lineplot(data=df_eu.reset_index(),
                 x='year', y=f"{material}, flat glass [kt]",
                 hue='Nbr of countries',
                 palette='colorblind',
                 ax=ax)
    ax.set_title(material)
    ax.set_ylabel(None)
    style_ax(ax)
    ax.get_legend().remove()

ax.set_xlim(1980, 2020)
ax.set_ylim(0, 6000)

fig.suptitle("Raw material flows for EU flat glass production [kt]")
fig.subplots_adjust(hspace=0.4)
sns.despine(offset=5)
plt.show()
```




```
[129]: # Estimate total use of raw materials (sum per material):
df_eu['Total raw materials, flat glass [Mt]'] = 0

for material in materials:
    df_eu['Total raw materials, flat glass [Mt]'] = (
        df_eu[f"{material}, flat glass [kt]"]/1000
        + df_eu['Total raw materials, flat glass [Mt]']
    )
```

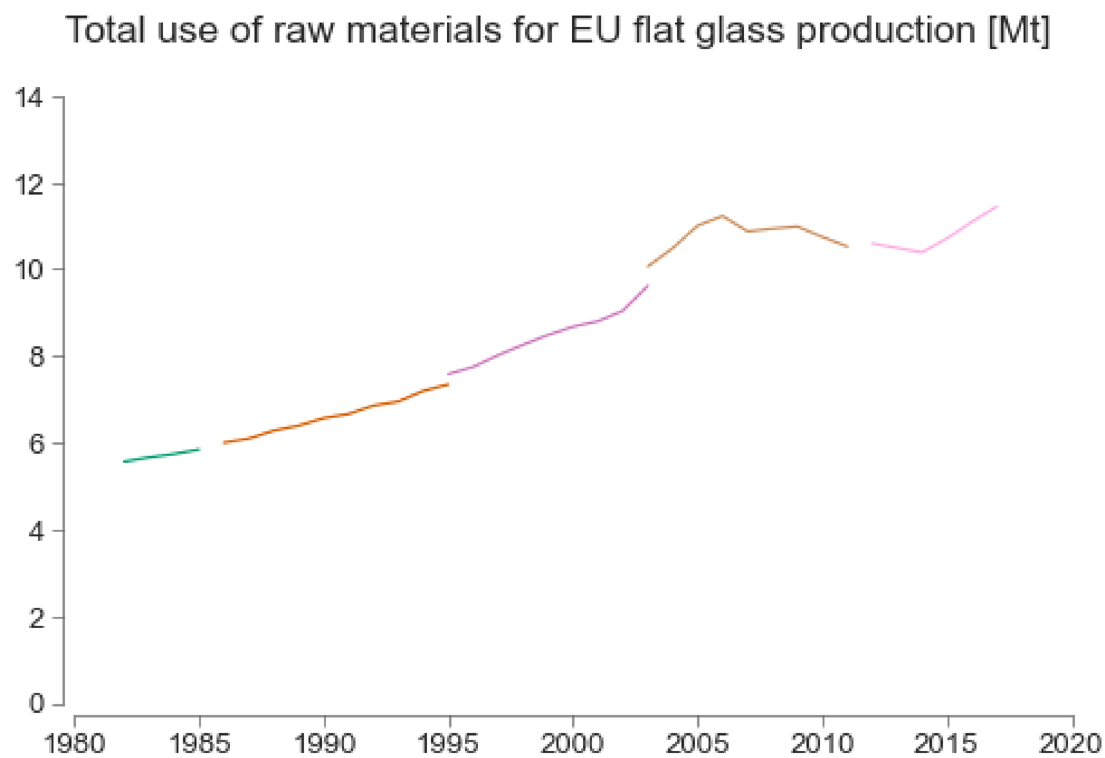
```
[130]: # Plot total material flows for EU flat glass production
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_eu.reset_index(),
             x='year', y='Total raw materials, flat glass [Mt]',
             hue='Nbr of countries',
             palette='colorblind',
             ax=ax)
```

```
ax.set_xlim(1980, 2020)
ax.set_ylabel(None)
style_ax(ax)
ax.set_ylim(0, 14)
ax.get_legend().remove()

fig.suptitle("Total use of raw materials for EU flat glass production [Mt]")
fig.subplots_adjust(hspace=0.4)
sns.despine(offset=5)
plt.show()
```



[131]:
```
# Plot total material flows for EU flat glass production
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_eu.rolling(5, center=True).mean().reset_index(),
             x='year', y='Total raw materials, flat glass [Mt]',
             hue='Nbr of countries',
             palette='colorblind',
             ax=ax)

ax.set_xlim(1980, 2020)
```

```python
ax.set_ylim(0, 14)
ax.get_legend().remove()
ax.set_ylabel(None)
style_ax(ax)

fig.suptitle("Total use of raw materials for EU flat glass production [Mt]")
fig.subplots_adjust(hspace=0.4)
sns.despine(offset=5)
plt.show()
```

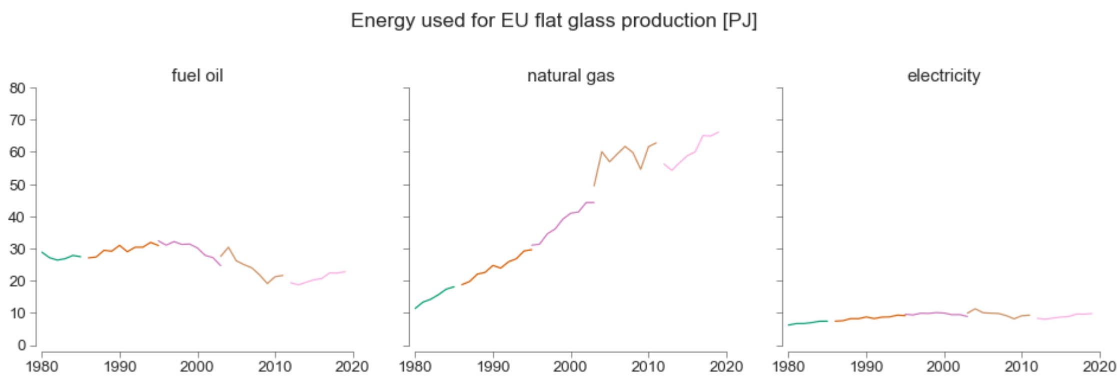


6.2 Energy Use in the Production of Flat Glass

[132]:
```python
# Absolute use of energy for flat glass production:
for energy in energies:
    df_eu[f"{energy}, flat glass [PJ]"] = (
        df_eu['Production [kt]']
        * (df_eu_energy[f"{energy}, GJ/t"]
           .interpolate(method='pchip', limit_direction='forward'))/1000
    )
```

[133]:
```python
# Plot energy used for EU flat glass production
fig, axes = plt.subplots(nrows=1, ncols=3,
```

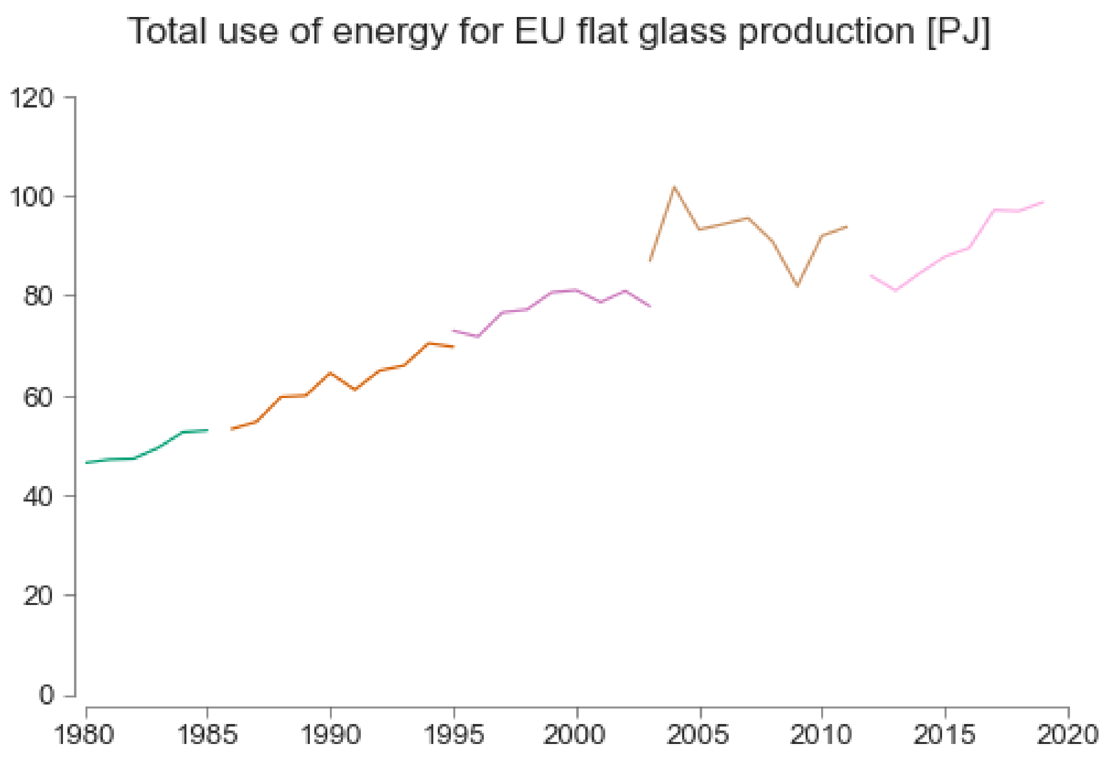
```
                            sharex=True, sharey=True,
                            figsize=(16, 4))

for i, (ax, energy) in enumerate(zip(axes.flatten(), energies)):
    sns.lineplot(data=df_eu.reset_index(),
                 x='year', y=f"{energy}, flat glass [PJ]",
                 hue='Nbr of countries',
                 palette='colorblind',
                 ax=ax)

    ax.set_title(energy)
    ax.get_legend().remove()
    ax.set_ylabel(None)
    style_ax(ax)

ax.set_xlim(1980, 2020)
ax.set_ylim(0, 80)

fig.suptitle("Energy used for EU flat glass production [PJ]", y=1.1)
fig.subplots_adjust(hspace=0.4)
sns.despine(offset=5)
plt.show()
```



[134]:
```
df_eu['Total energy, flat glass [PJ]'] = 0

for energy in energies:
    df_eu['Total energy, flat glass [PJ]'] = (
        df_eu[f"{energy}, flat glass [PJ]"]
        + df_eu['Total energy, flat glass [PJ]']
    )
```

[135]:
```
# Plot total energy use for EU flat glass production
fig, ax = plt.subplots(figsize=(8, 5))
```

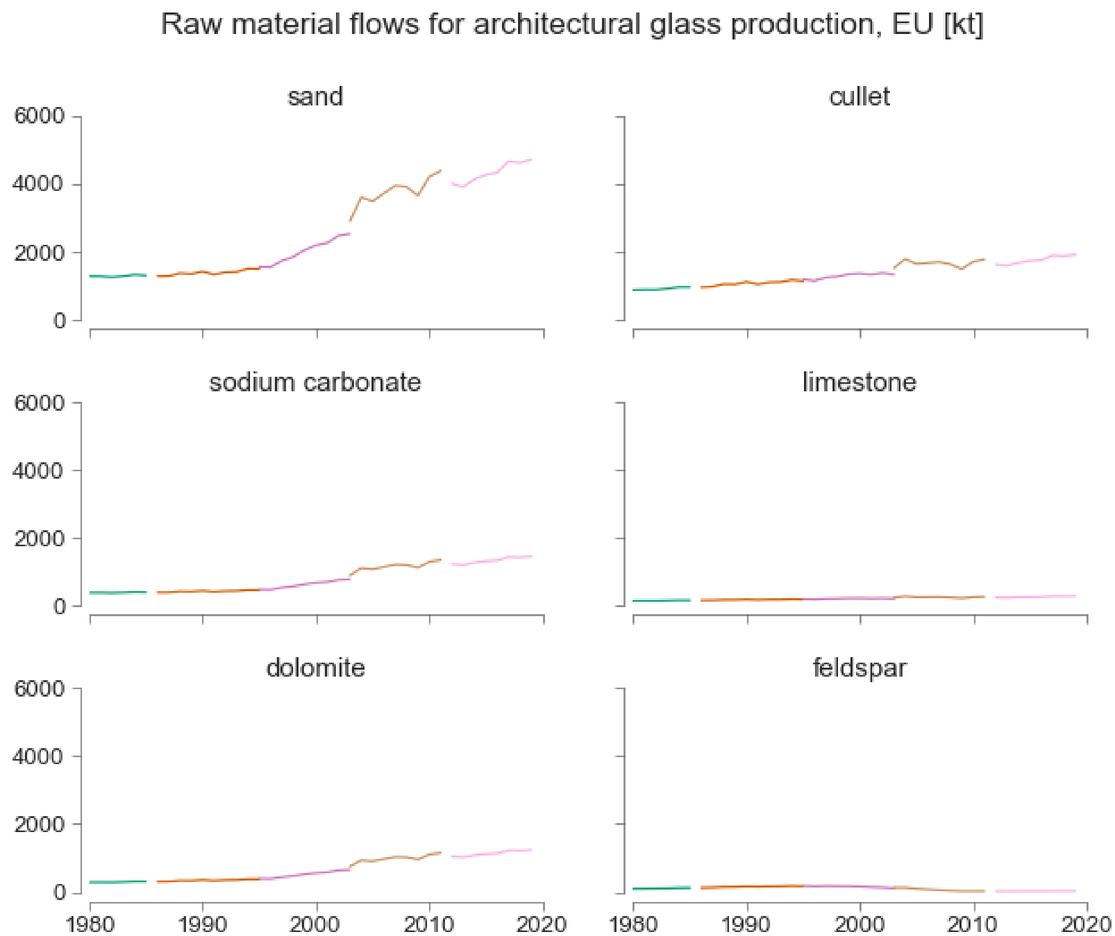
```
sns.lineplot(data=df_eu.reset_index(),
             x='year', y='Total energy, flat glass [PJ]',
             hue='Nbr of countries',
             palette='colorblind',
             ax=ax)

ax.set_xlim(1980, 2020)
ax.set_ylim(0, 120)
ax.get_legend().remove()
ax.set_ylabel(None)
style_ax(ax)

fig.suptitle("Total use of energy for EU flat glass production [PJ]")
fig.subplots_adjust(hspace=0.4)
sns.despine(offset=5)
plt.show()
```



6.3 Use of Raw Materials in the Production of Architectural Flat Glass

[136]:
```
# Absolute use of raw materials for fat glass production:
for mat in materials:
    df_eu[f"{mat}, arch glass [kt]"] = (
        df_eu['Architectural glass production [kt]']
        * (df_rawmat[f"{mat}, kg/kg"]
           .interpolate(method='pchip', limit_direction='forward'))
    )
```

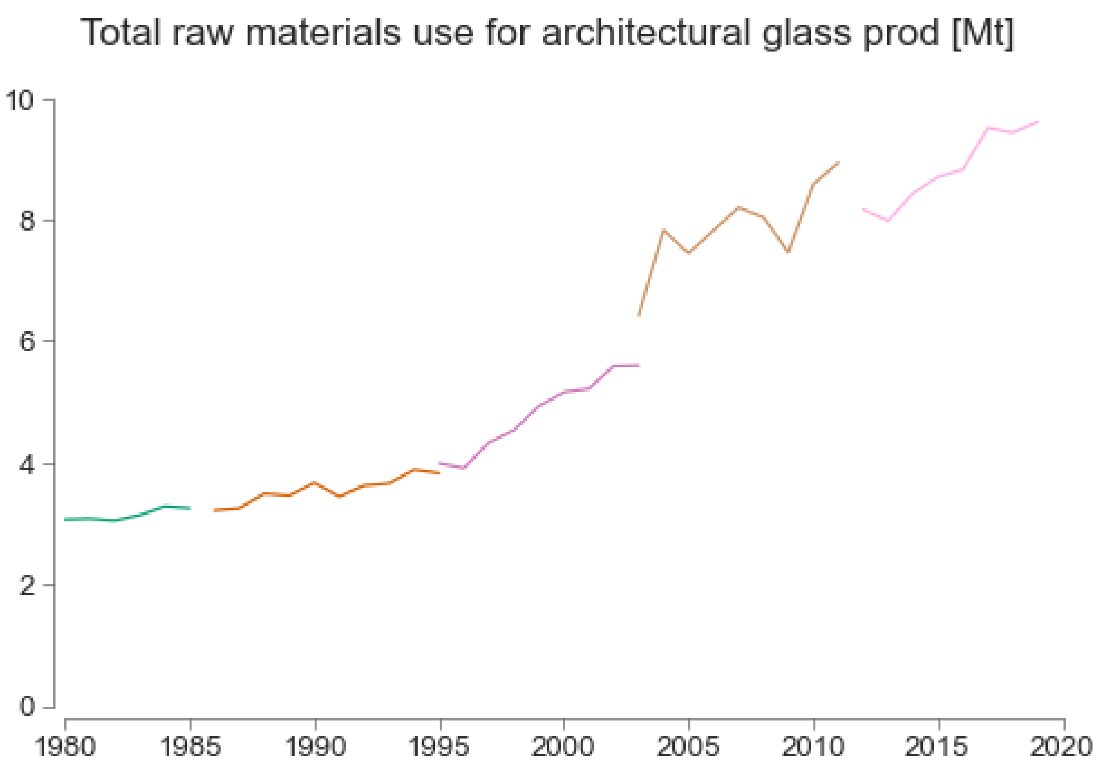
[137]:
```
# Plot raw material flows for EU flat glass production
fig, axes = plt.subplots(nrows=3, ncols=2,
                         sharex=True, sharey=True,
                         figsize=(10, 8))

for i, (ax, material) in enumerate(zip(axes.flatten(), materials)):
    sns.lineplot(data=df_eu.reset_index(),
                 x='year', y=f"{material}, arch glass [kt]",
                 hue='Nbr of countries',
                 palette='colorblind',
                 ax=ax)

    ax.set_title(material)
    ax.set_xlim(1980, 2020)
    ax.set_ylim(0, 6000)
    ax.get_legend().remove()
    ax.set_ylabel(None)
    style_ax(ax)

fig.suptitle("Raw material flows for architectural glass production, EU [kt]")
fig.subplots_adjust(hspace=0.4)
sns.despine(offset=5)
plt.show()
```



[138]:
```
df_eu['Total raw materials, arch glass [Mt]'] = 0

for material in materials:
    df_eu['Total raw materials, arch glass [Mt]'] = (
        df_eu[f"{material}, arch glass [kt]"]/1000
        + df_eu['Total raw materials, arch glass [Mt]']
    )
```

[139]:
```
# Plot total material flows for EU architectural glass production
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_eu.reset_index(),
             x='year', y='Total raw materials, arch glass [Mt]',
             hue='Nbr of countries',
             palette='colorblind',
             ax=ax)
```

```
ax.set_xlim(1980, 2020)
ax.set_ylim(0, 10)
ax.get_legend().remove()
ax.set_ylabel(None)
style_ax(ax)

fig.suptitle("Total raw materials use for architectural glass prod [Mt]")
fig.subplots_adjust(hspace=0.4)
sns.despine(offset=5)
plt.show()
```

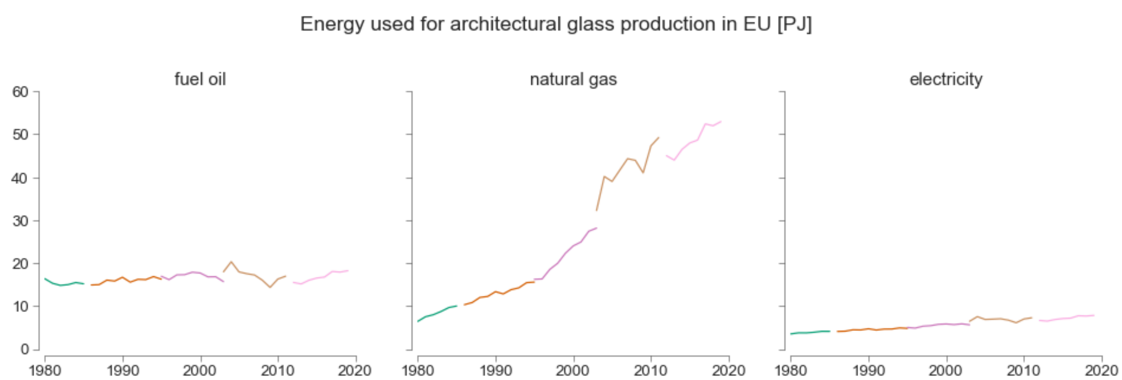


6.4 Energy Use in the Production of Architectural Flat Glass

[140]:
```
# Absolute use of energy for flat glass production:
for energy in energies:
    df_eu[f"{energy}, arch glass [PJ]"] = (
        df_eu['Architectural glass production [kt]']
        * (df_eu_energy[f"{energy}, GJ/t"]
           .interpolate(method='pchip', limit_direction='forward'))/1000
    )
```

[141]:

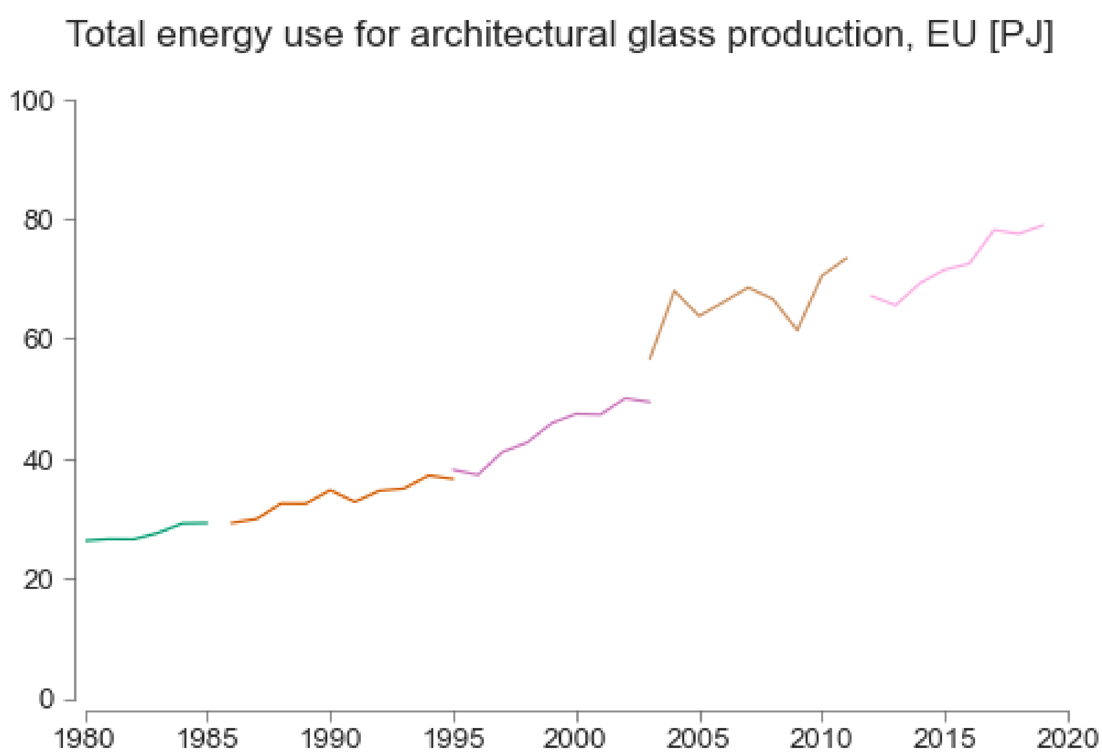
```
# Plot energy used for EU flat glass production
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 4))

for i, (ax, energy) in enumerate(zip(axes.flatten(), energies)):
    sns.lineplot(data=df_eu.reset_index(),
                 x='year', y=f"{energy}, arch glass [PJ]",
                 hue='Nbr of countries',
                 palette='colorblind',
                 ax=ax)

    ax.set_title(energy)
    ax.set_xlim(1980, 2020)
    ax.set_ylim(0, 60)
    ax.get_legend().remove()
    ax.set_ylabel(None)
    style_ax(ax)

fig.suptitle("Energy used for architectural glass production in EU [PJ]",
             y=1.1)
fig.subplots_adjust(hspace=0.4)
sns.despine(offset=5)
plt.show()
```



[142]:

```
df_eu['Total energy, arch glass [PJ]'] = 0

for energy in energies:
    df_eu['Total energy, arch glass [PJ]'] = (
        df_eu[f"{energy}, arch glass [PJ]"]
        + df_eu['Total energy, arch glass [PJ]']
    )
```

[143]:
```python
# Plot total energy use for EU flat glass production
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=df_eu.reset_index(),
             x='year', y='Total energy, arch glass [PJ]',
             hue='Nbr of countries',
             palette='colorblind',
             ax=ax)

ax.set_xlim(1980, 2020)
ax.set_ylim(0, 100)
ax.get_legend().remove()
ax.set_ylabel(None)
style_ax(ax)

fig.suptitle("Total energy use for architectural glass production, EU [PJ]")
fig.subplots_adjust(hspace=0.4)
sns.despine(offset=5)
plt.show()
```



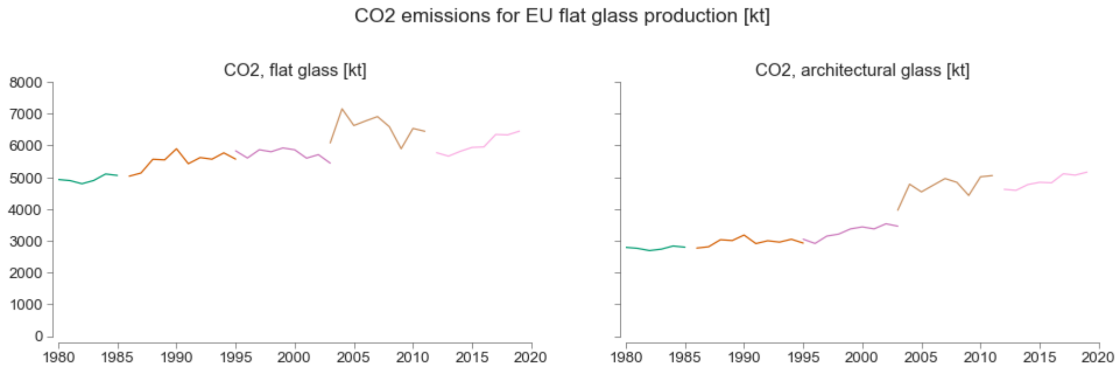
6.5 Absolute CO2 Emissions, Europe

[144]:
```
# Estimate the CO2 emission related to flat glass production:
df_eu['CO2, flat glass [kt]'] = (
    df_eu['Production [kt]']
    * df_eu_co2['CO2 interpolated [kg/t]'] / 1000
)
```

[145]:
```
# Estimate the CO2 emission related to architectural flat glass production:
df_eu['CO2, architectural glass [kt]'] = (
    df_eu['Architectural glass production [kt]']
    * df_eu_co2['CO2 interpolated [kg/t]'] / 1000
)
```

[146]:
```
CO2_G = ['CO2, flat glass [kt]', 'CO2, architectural glass [kt]']
```

[147]:
```
fig, axes = plt.subplots(nrows=1, ncols=2,
                         sharex=True, sharey=True,
                         figsize=(16, 4))

for i, (ax, CO2) in enumerate(zip(axes.flatten(), CO2_G)):
    sns.lineplot(data=df_eu.reset_index(),
                 x='year', y=f"{CO2}",
                 hue='Nbr of countries',
                 palette='colorblind',
                 ax=ax)
    ax.set_title(CO2)
    ax.set_xlim(1980, 2020)
    ax.set_ylim(0, 8000)
    ax.get_legend().remove()
    ax.set_ylabel(None)
    style_ax(ax)

fig.suptitle("CO2 emissions for EU flat glass production [kt]", y=1.1)
fig.subplots_adjust(hspace=0.4)
sns.despine(offset=5)
plt.show()
```

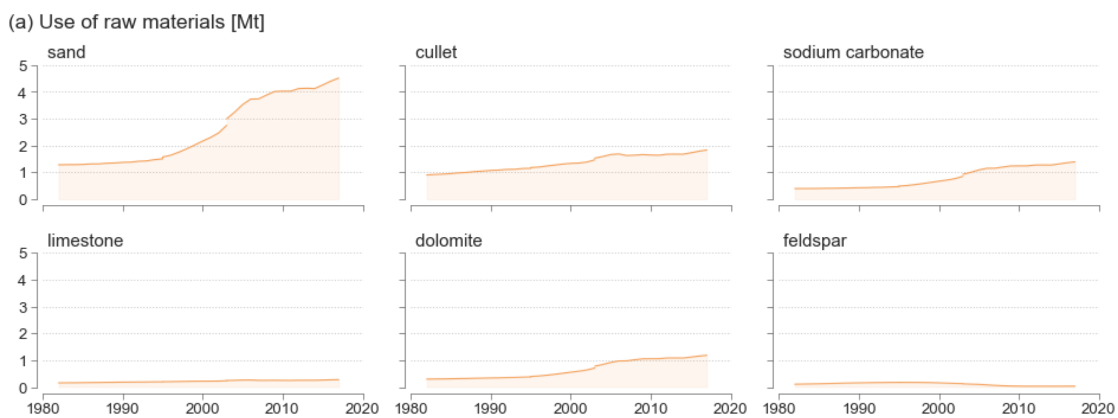


6.6 Summary

[148]:
```
# Number of countries for trend curves:
EU_COUNTRIES = [(10, 15), (27, 28)]
```

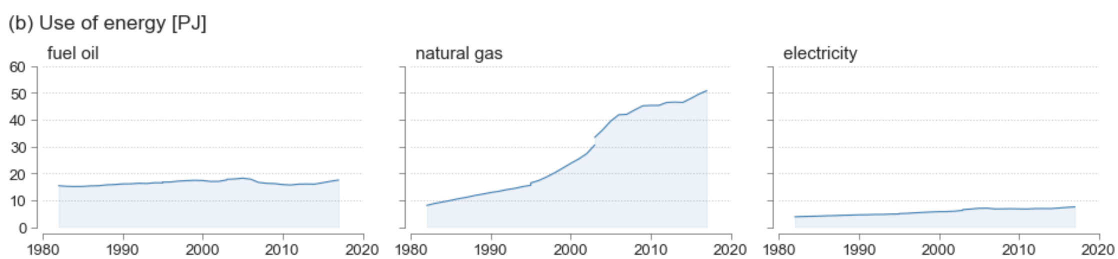
[149]:
```
# Plot the absolute use of raw materials in EU for architectural glass prod:
fig, axes = plt.subplots(nrows=2, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 5))

for i, (ax, mat) in enumerate(zip(axes.flatten(), materials)):
    for a, b in EU_COUNTRIES:
        ax.plot(df_eu.interpolate().rolling(5, center=True).mean().loc[a:b]
                .reset_index(level='Nbr of countries')
                .sort_values(by=['year'])
                [f"{mat}, arch glass [kt]"],
                linewidth=1.1, c='sandybrown', linestyle='-')

    x = (df_eu.reset_index()['year'])
    y = (df_eu[f"{mat}, arch glass [kt]"].interpolate()
         .rolling(5, center=True).mean())

    ax.fill_between(x, y, color="sandybrown", alpha=0.1)
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

    ax.set_title(f" {mat}", loc='left')
    style_ax(ax)

ax.set_ylim(0, 5000)
ax.set_xlim(1980, 2020)
plt.yticks(np.arange(0, 5001, 1000))

# Divide by 1000 the y-axis, results in Mt
```

```
ax.yaxis.set_major_formatter(y_1000)

fig.suptitle("(a) Use of raw materials [Mt]", x=0.1, y=1, ha='left')
fig.subplots_adjust(wspace=0.15, hspace=0.4)

sns.despine(offset=5)

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig7_EU_Use of raw materials.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig7_EU_Use of raw materials.pdf'),
                bbox_inches='tight')

plt.show()
```



[150]:
```
# Plot the absolute use of energy in the EU for architectural glass prod:
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(16, 2.5))

for i, (ax, energy) in enumerate(zip(axes.flatten(), energies)):
    for a, b in EU_COUNTRIES:
        ax.plot(df_eu.interpolate().rolling(5, center=True).mean().loc[a:b]
                .reset_index(level='Nbr of countries')
                .sort_values(by=['year'])
                [f"{energy}, arch glass [PJ]"],
                linewidth=1.1, c='steelblue', linestyle='-')

    x = (df_eu.reset_index()['year'])
    y = (df_eu[f"{energy}, arch glass [PJ]"].interpolate()
          .rolling(5, center=True).mean())
```

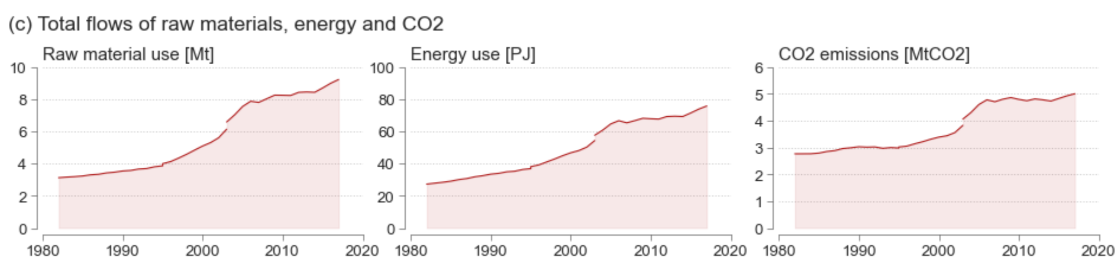
```
    ax.fill_between(x, y, color="steelblue", alpha=0.10)
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

    ax.set_title(f" {energy}", loc='left')
    style_ax(ax)

ax.set_xlim(1980, 2020)
ax.set_ylim(0, 60)
plt.yticks(np.arange(0, 61, 10))

fig.suptitle("(b) Use of energy [PJ]", x=0.1, y=1.12, ha='left')
fig.subplots_adjust(wspace=0.15)

sns.despine(offset=5)

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig7_EU_Use of energy.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig7_EU_Use of energy.pdf'),
                bbox_inches='tight')

plt.show()
```



[151]:
```
# Plot the total energy and material use, and total CO2 emissions:
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True,
                         figsize=(16, 2.5))

# Total use of raw materials:
ax = axes[0]
for a, b in EU_COUNTRIES:
    ax.plot(df_eu.interpolate().rolling(5, center=True).mean().loc[a:b]
            .reset_index(level='Nbr of countries')
            .sort_values(by=['year'])
            ['Total raw materials, arch glass [Mt]'],
```

```
            linewidth=1.1, c='firebrick', linestyle='-')

x = (df_eu.reset_index()['year'])
y = (df_eu['Total raw materials, arch glass [Mt]'].interpolate()
     .rolling(5, center=True).mean())
ax.fill_between(x, y, color="firebrick", alpha=0.1)

ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

ax.set_title("Raw material use [Mt]", loc='left')
style_ax(ax)
ax.set_ylim(0, 10)
ax.set_xlim(1980, 2020)

# Total use of energy:
ax = axes[1]
for a, b in EU_COUNTRIES:
    ax.plot(df_eu.interpolate().rolling(5, center=True).mean().loc[a:b]
            .reset_index(level='Nbr of countries')
            .sort_values(by=['year'])
            ['Total energy, arch glass [PJ]'],
            linewidth=1.1, c='firebrick', linestyle='-')

x = (df_eu.reset_index()['year'])
y = (df_eu['Total energy, arch glass [PJ]'].interpolate()
     .rolling(5, center=True).mean())
ax.fill_between(x, y, color="firebrick", alpha=0.1)

ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

ax.set_title("Energy use [PJ]", loc='left')
style_ax(ax)
ax.set_ylim(0, 100)

# Total emissions of CO2:
ax = axes[2]
for a, b in EU_COUNTRIES:
    ax.plot(df_eu.interpolate().rolling(5, center=True).mean().loc[a:b]
            .reset_index(level='Nbr of countries')
            .sort_values(by=['year'])
            ['CO2, architectural glass [kt]'],
            linewidth=1.1, c='firebrick', linestyle='-')

x = (df_eu.reset_index()['year'])
y = (df_eu['CO2, architectural glass [kt]'].interpolate()
     .rolling(5, center=True).mean())
ax.fill_between(x, y, color="firebrick", alpha=0.1)
```

```python
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

# Divide by 1000 the y-axis, results in Mt
ax.yaxis.set_major_formatter(y_1000)

ax.set_title("CO2 emissions [MtCO2]", loc='left')
style_ax(ax)
ax.set_ylim(0, 6000)
plt.yticks(np.arange(0, 6001, 1000))

fig.suptitle("(c) Total flows of raw materials, energy and CO2",
             x=0.1, y=1.12, ha='left')
fig.subplots_adjust(wspace=0.15)

sns.despine(offset=5)

if export:
    # Save image:
    plt.savefig(os.path.join(path_img, 'Fig7_EU_Total flows.png'),
                dpi=600, bbox_inches='tight')
    plt.savefig(os.path.join(path_img, 'Fig7_EU_Total flows.pdf'),
                bbox_inches='tight')

plt.show()
```



7 Raw Material and Energy Use and CO2 Emissions in the Belgian and French Flat Glass Industries

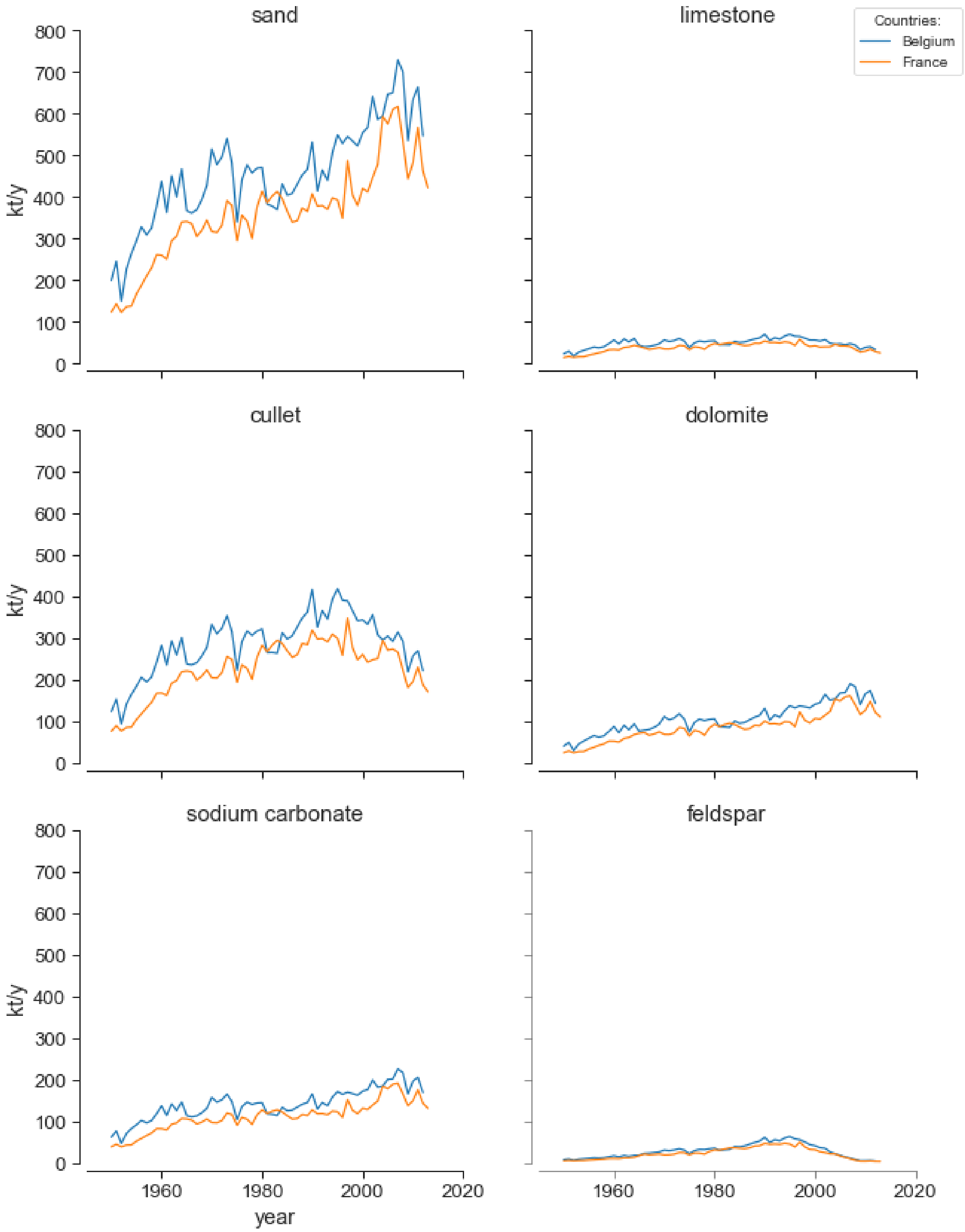
7.1 Use of Raw Materials in the Production of Flat Glass

[152]:
```python
# Absolute use of raw materials for fat glass production:
for mat in materials:
    for country in countries:
        df_be_fr[(f"{mat}, flat glass [kt]", country)] = (
            df_be_fr[('Production [kt]', country)]
```

```
            * (df_rawmat[f"{mat}, kg/kg"]
               .interpolate(method='pchip', limit_direction='forward'))
        )
```

[153]:

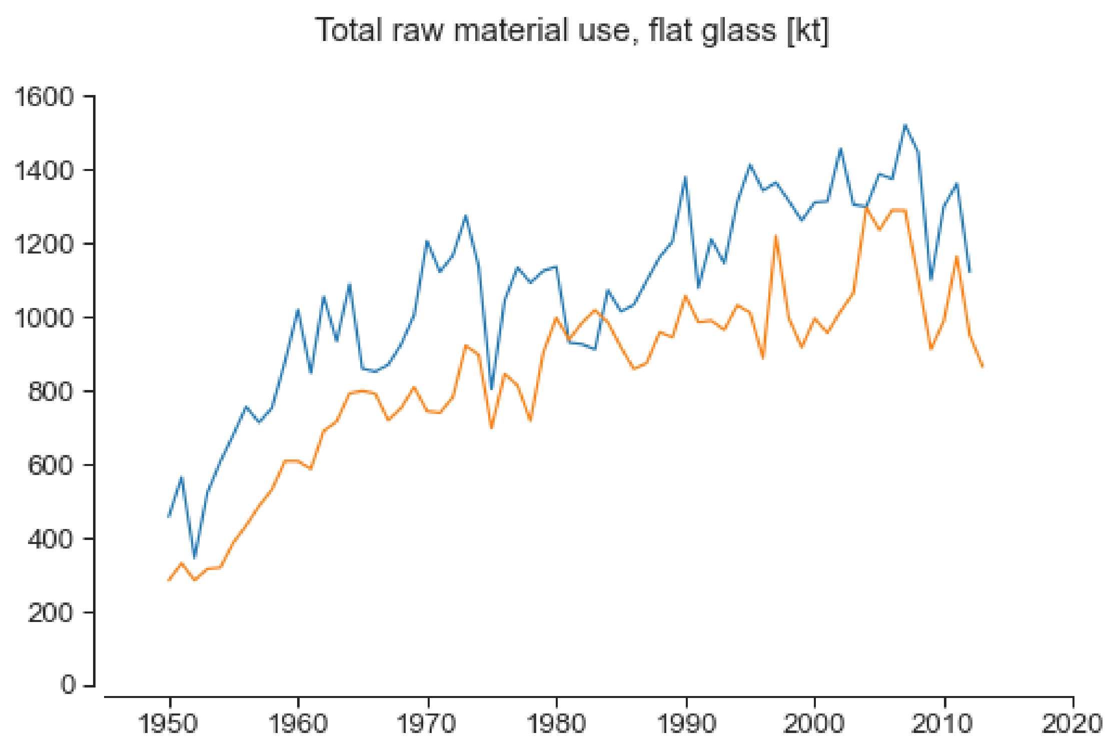
```
nrows = int(len(materials)/2)

fig, axes = plt.subplots(nrows=nrows, ncols=2,
                         sharex=True, sharey=True,
                         figsize=(10, 14))
row = 0
col = 0

for material in materials:
    ax = axes[row][col]
    ax.set_title(material)
    if col == 0:
        ax.set_ylabel("kt/y")
    if row == 2:
        row = 0
        col = 1
    else:
        row += 1
    df_be_fr[f"{material}, flat glass [kt]"].plot(ax=ax)
    ax.get_legend().remove()

ax.set_ylim(0, 800)
ax.set_xlim(1945, 2020)

# Add legend:
handles, labels = ax.get_legend_handles_labels()
fig.legend(handles, labels, ncol=1,
           title='Countries:', bbox_to_anchor=(0.95, 0.9))

style_ax(ax)

sns.despine(offset=5)
plt.show()
```



[154]:
```
for country in countries:
    df_be_fr[('Total raw material use, flat glass [kt]', country)] = 0

for material in materials:
```

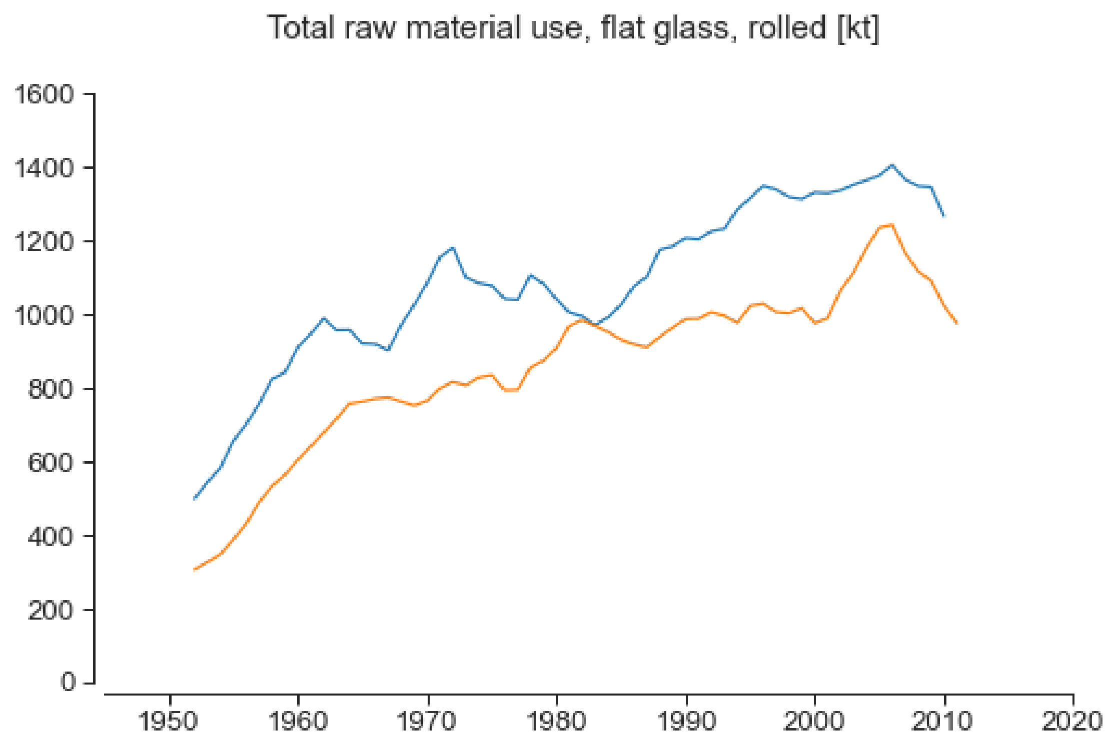
```
    df_be_fr['Total raw material use, flat glass [kt]'] = (
        (df_be_fr[f"{material}, flat glass [kt]"])
        + df_be_fr['Total raw material use, flat glass [kt]']
    )
```

[155]:
```
fig, ax = plt.subplots(figsize=(8, 5))

ax.plot(df_be_fr.index,
        df_be_fr['Total raw material use, flat glass [kt]'])

ax.set_xlim(1945, 2020)
ax.set_ylim(0, 1600)

fig.suptitle('Total raw material use, flat glass [kt]', fontsize=15)
sns.despine(offset=5)
plt.show()
```



[156]:
```
fig, ax = plt.subplots(figsize=(8, 5))


ax.plot(df_be_fr.index,
        df_be_fr['Total raw material use, flat glass [kt]']
        .rolling(5, center=True).mean())
```

```
ax.set_xlim(1945, 2020)
ax.set_ylim(0, 1600)

fig.suptitle('Total raw material use, flat glass, rolled [kt]', fontsize=15)
sns.despine(offset=5)
plt.show()
```



7.2 Energy Use in the Production of Flat Glass

[157]:
```
# Absolute use of energy for flat glass production:
for energy in energies:
    for country in countries:
        df_be_fr[(f"{energy}, flat glass [TJ]", country)] = (
            df_be_fr[('Production [kt]', country)]
            * (df_be_fr_energy[(f"{energy}, GJ/t", country)]
              .interpolate(method='pchip', limit_direction='forward'))
        )
```

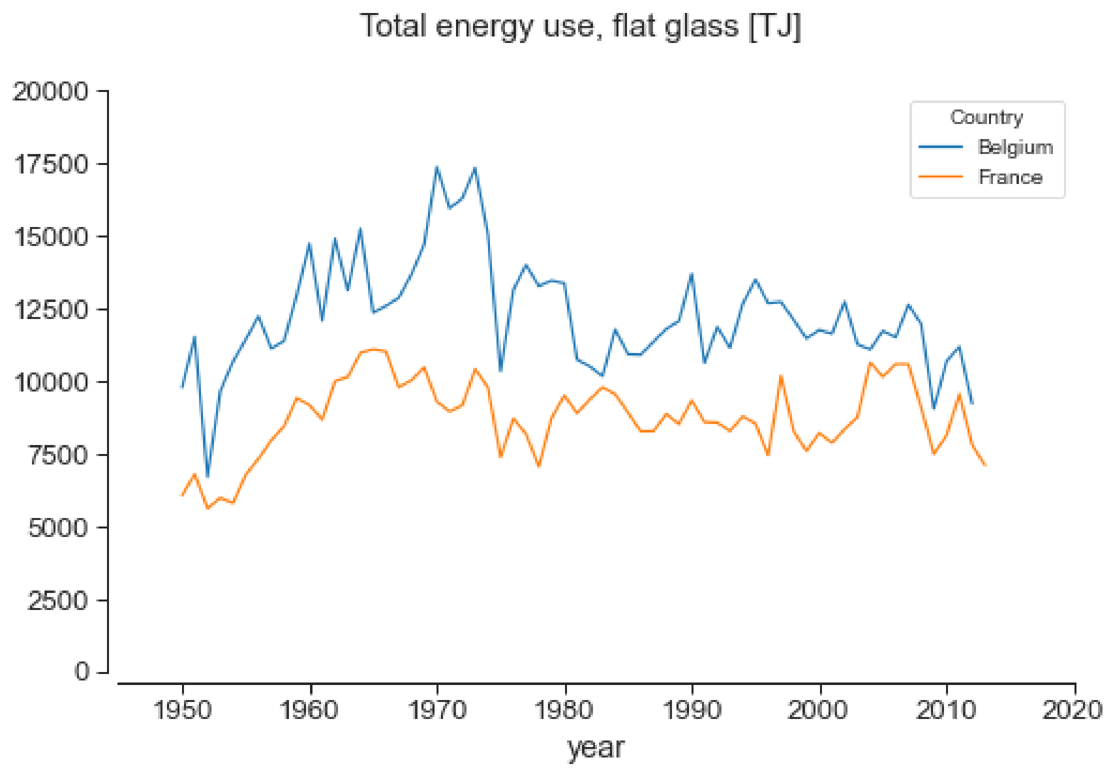
[158]:
```
nrows = len(energies)

fig, axes = plt.subplots(nrows=nrows, ncols=1,
                         sharex=True, sharey=True,
```

```
                         figsize=(8, 10))

for row, energy in enumerate(energies):
    ax = axes[row]
    df_be_fr[f"{energy}, flat glass [TJ]"].plot(ax=ax)
    ax.set_ylabel(energy)

    style_ax(ax)
    ax.get_legend().remove()

# Add legend:
handles, labels = ax.get_legend_handles_labels()
fig.legend(handles, labels, ncol=1,
           title='Countries:', bbox_to_anchor=(0.95, 0.9))

ax.set_xlim(1945, 2020)
ax.set_ylim(0, 15000)

sns.despine(offset=5)
plt.show()
```

[159]:
```
for country in countries:
    df_be_fr[('Total energy use, flat glass [TJ]', country)] = 0

for energy in energies:
    df_be_fr['Total energy use, flat glass [TJ]'] = (
        df_be_fr[f"{energy}, flat glass [TJ]"]
        + df_be_fr['Total energy use, flat glass [TJ]']
    )
```

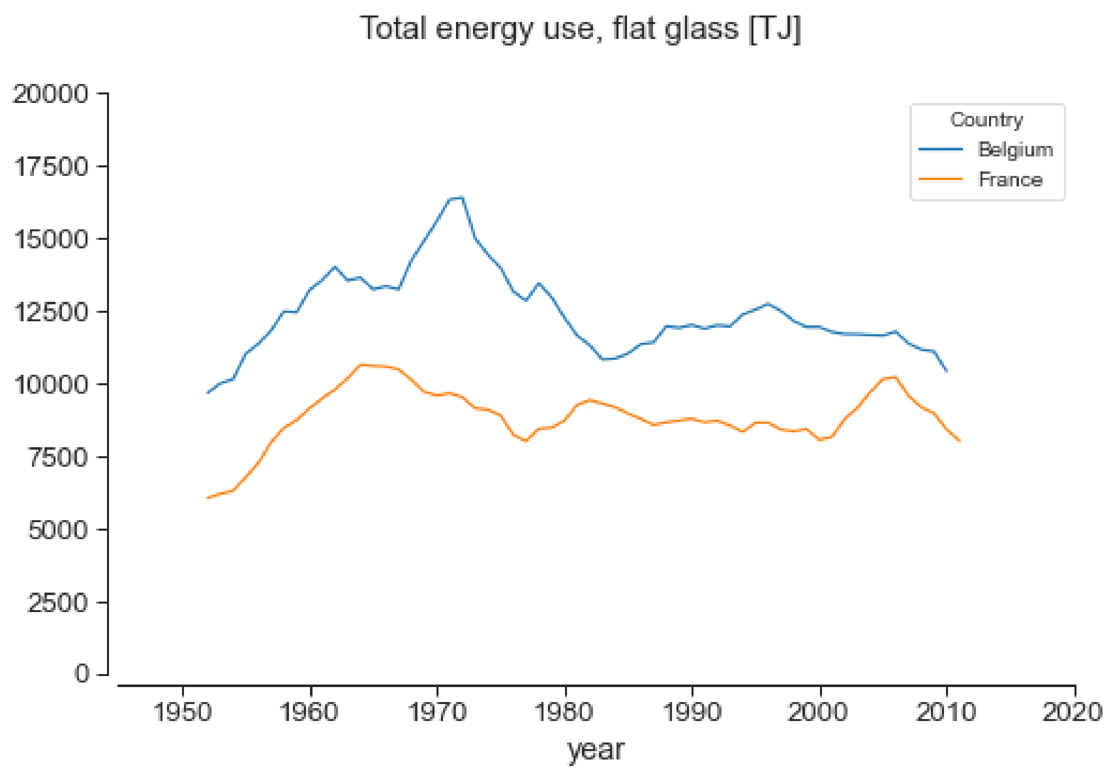
[160]:

```
fig, ax = plt.subplots(figsize=(8, 5))

(df_be_fr['Total energy use, flat glass [TJ]']).plot(ax=ax)
ax.set_xlim(1945, 2020)
ax.set_ylim(0, 20000)

fig.suptitle('Total energy use, flat glass [TJ]', fontsize=15)
sns.despine(offset=5)
plt.show()
```



[161]:

```
fig, ax = plt.subplots(figsize=(8, 5))

(df_be_fr['Total energy use, flat glass [TJ]']
 .rolling(5, center=True).mean()).plot(ax=ax)
ax.set_xlim(1945, 2020)
ax.set_ylim(0, 20000)

fig.suptitle('Total energy use, flat glass [TJ]', fontsize=15)
sns.despine(offset=5)
plt.show()
```

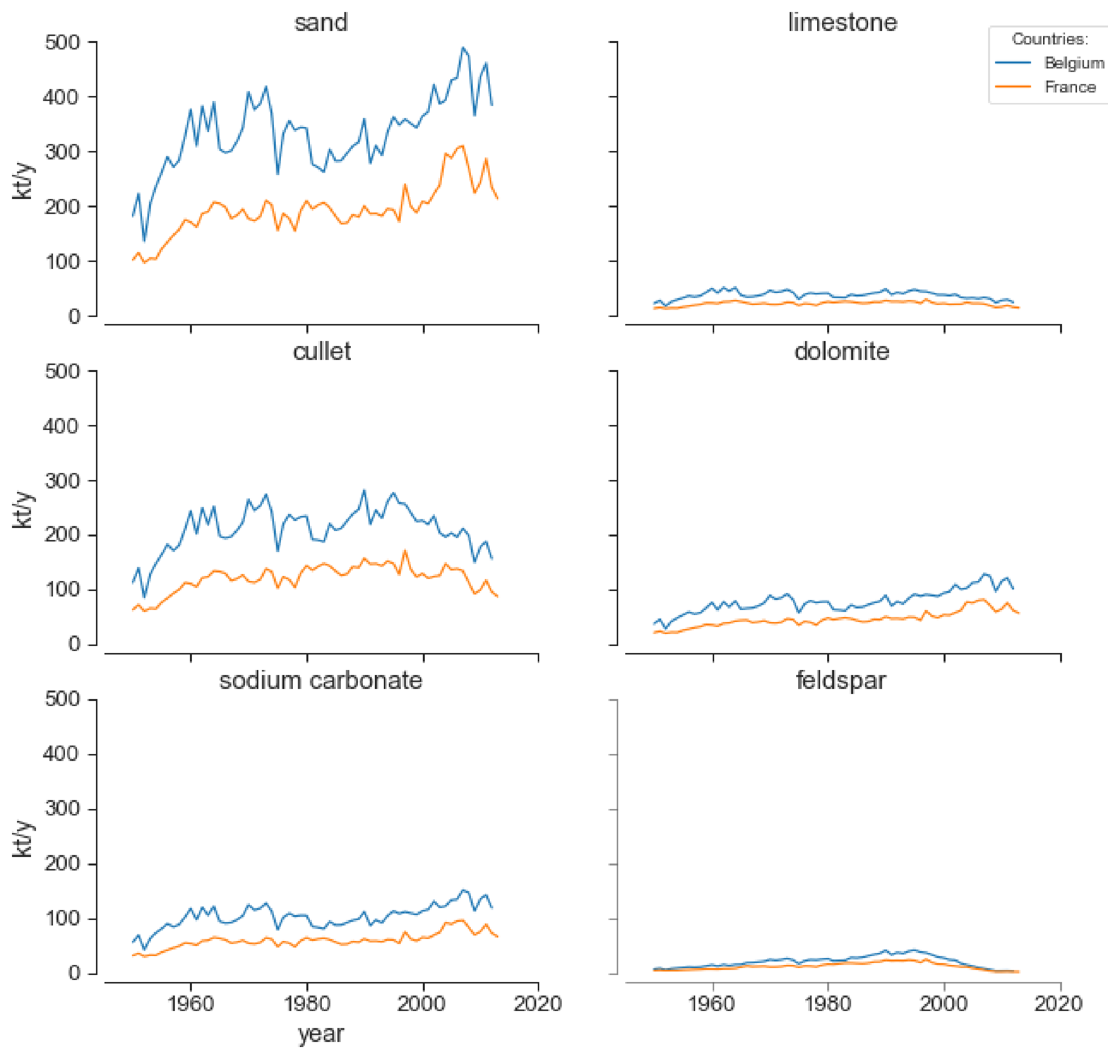


7.3 Use of Raw Materials in the Production of Architectural Glass

[162]:
```
# Absolute use of raw materials for architectural flat glass production:
for mat in materials:
    for country in countries:
        df_be_fr[(f"{mat}, architectural glass [kt]", country)] = (
            df_be_fr[('Architectural glass production [kt]', country)]
            * (df_rawmat[f"{mat}, kg/kg"]
               .interpolate(method='pchip', limit_direction='forward'))
        )
```

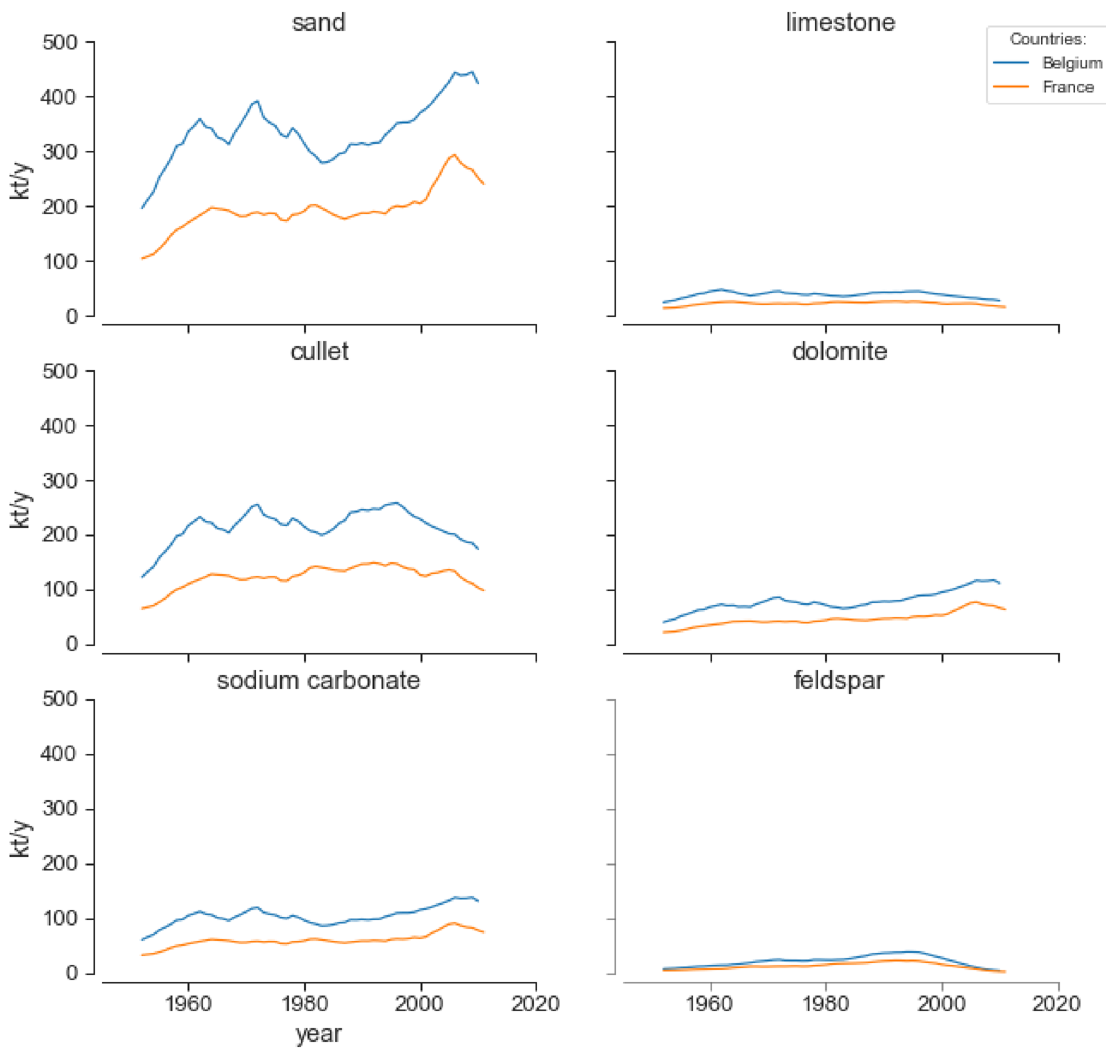
[163]:
```
nrows = int(len(materials)/2)

fig, axes = plt.subplots(nrows=nrows, ncols=2,
                         sharex=True, sharey=True,
                         figsize=(10, 10))
row = 0
col = 0

for material in materials:
    ax = axes[row][col]
    ax.set_title(material)
```

```
    if col == 0:
        ax.set_ylabel("kt/y")
    if row == 2:
        row = 0
        col = 1
    else:
        row += 1
    df_be_fr[f"{material}, architectural glass [kt]"].plot(ax=ax)
    ax.get_legend().remove()

ax.set_ylim(0, 500)
ax.set_xlim(1945, 2020)

# Add legend:
handles, labels = ax.get_legend_handles_labels()
fig.legend(handles, labels, ncol=1,
           title='Countries:', bbox_to_anchor=(0.95, 0.9))

style_ax(ax)

sns.despine(offset=5)
plt.show()
```



[164]:
```
nrows = int(len(materials)/2)

fig, axes = plt.subplots(nrows=nrows, ncols=2,
                         sharex=True, sharey=True,
                         figsize=(10, 10))
row = 0
col = 0

for material in materials:
    ax = axes[row][col]
    ax.set_title(material)
    if col == 0:
        ax.set_ylabel("kt/y")
    if row == 2:
```

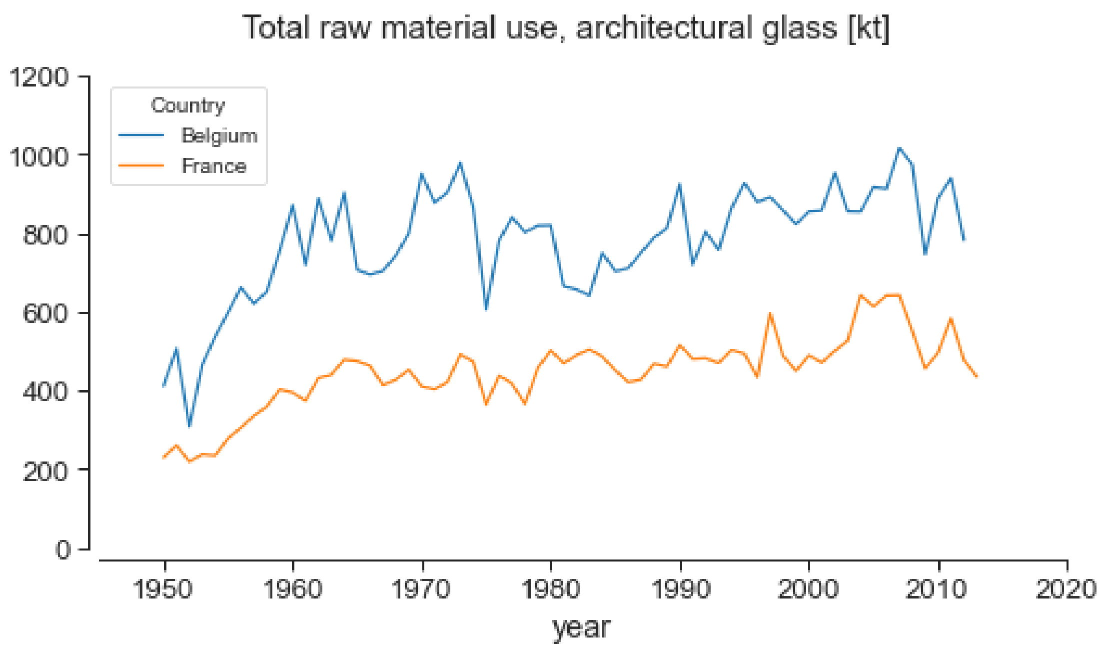
```
        row = 0
        col = 1
    else:
        row += 1
    (df_be_fr[f"{material}, architectural glass [kt]"]
     .rolling(5, center=True).mean().plot(ax=ax))
    ax.get_legend().remove()

ax.set_ylim(0, 500)
ax.set_xlim(1945, 2020)

# Add legend:
handles, labels = ax.get_legend_handles_labels()
fig.legend(handles, labels, ncol=1,
           title='Countries:', bbox_to_anchor=(0.95, 0.9))

style_ax(ax)

sns.despine(offset=5)
plt.show()
```



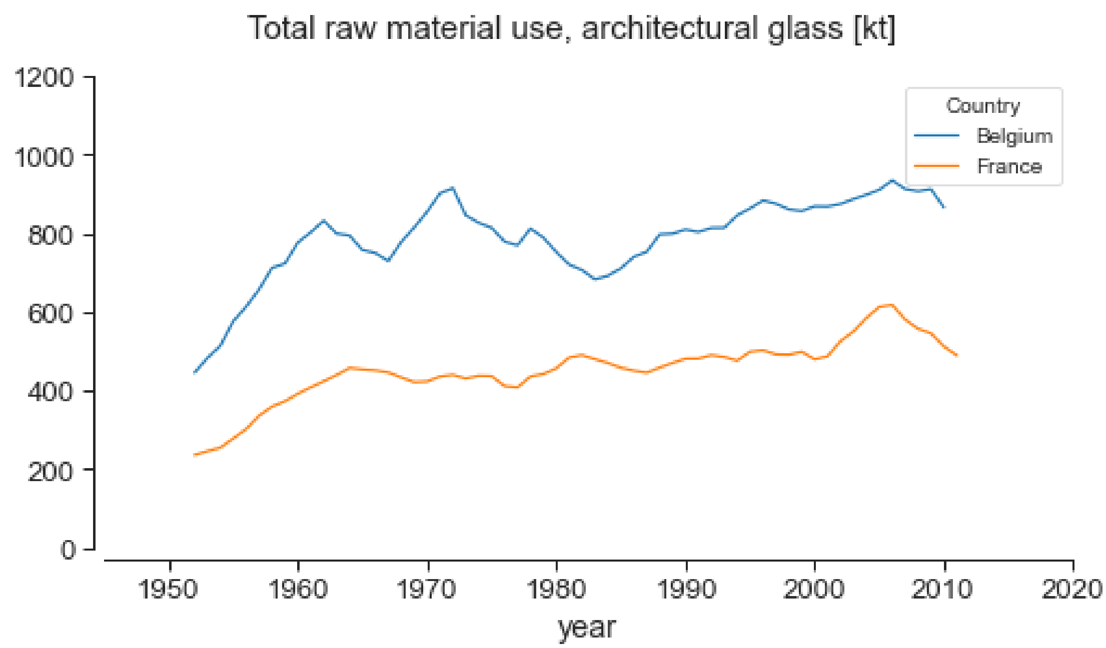
[165]:
```
for country in countries:
    df_be_fr[('Total raw material use, architectural glass [kt]', country)] = 0

for material in materials:
    df_be_fr['Total raw material use, architectural glass [kt]'] = (
        df_be_fr[f"{material}, architectural glass [kt]"]
        + df_be_fr['Total raw material use, architectural glass [kt]']
    )
```

[166]:
```
fig, ax = plt.subplots(figsize=(8, 4))

(df_be_fr['Total raw material use, architectural glass [kt]']).plot(ax=ax)

ax.set_xlim(1945, 2020)
```

```
ax.set_ylim(0, 1200)

fig.suptitle('Total raw material use, architectural glass [kt]', fontsize=15)
sns.despine(offset=5)
plt.show()
```



[167]:
```
fig, ax = plt.subplots(figsize=(8, 4))

(df_be_fr['Total raw material use, architectural glass [kt]']
 .rolling(5, center=True).mean().plot(ax=ax))

ax.set_xlim(1945, 2020)
ax.set_ylim(0, 1200)

fig.suptitle('Total raw material use, architectural glass [kt]', fontsize=15)
sns.despine(offset=5)
plt.show()
```



[168]:
```python
(df_be_fr['Total raw material use, architectural glass [kt]']['France']
 .rolling(5, center=True).mean())[2006]
```

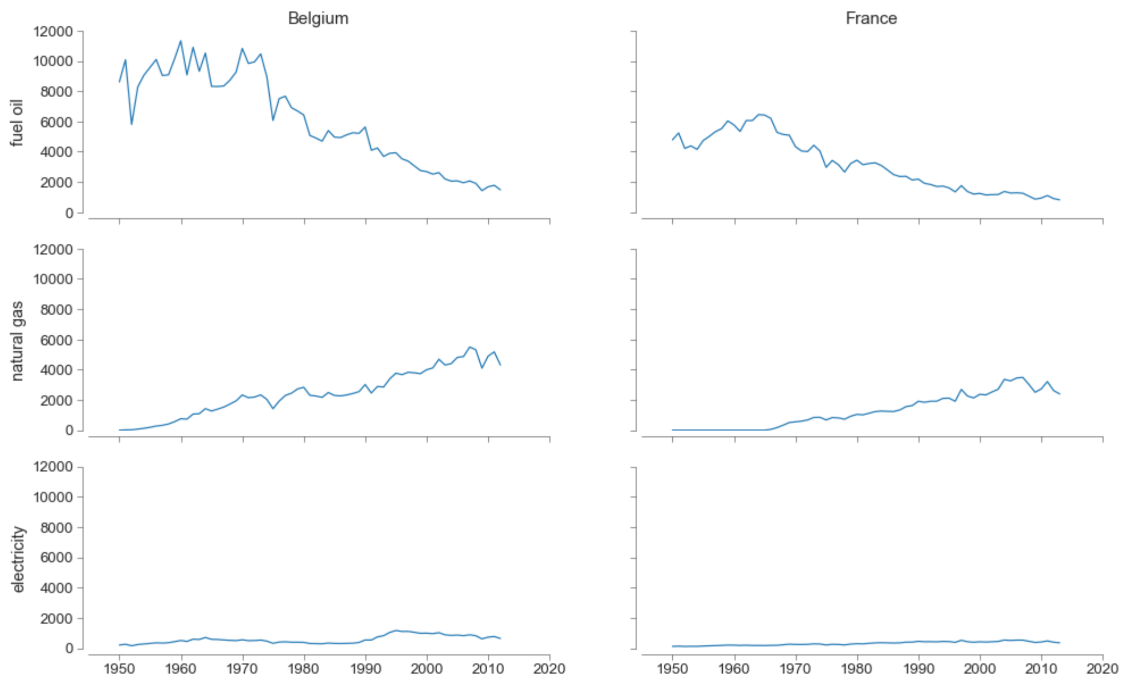
[168]: 617.6551457030012

7.4 Energy Use in the Production of Architectural Glass

[169]:
```python
# Absolute use of energy for flat glass production:
for energy in energies:
    for country in countries:
        df_be_fr[(f"{energy}, architectural glass [TJ]", country)] = (
            df_be_fr[('Architectural glass production [kt]', country)]
            * (df_be_fr_energy[(f"{energy}, GJ/t", country)]
               .interpolate(method='pchip', limit_direction='forward'))
        )
```

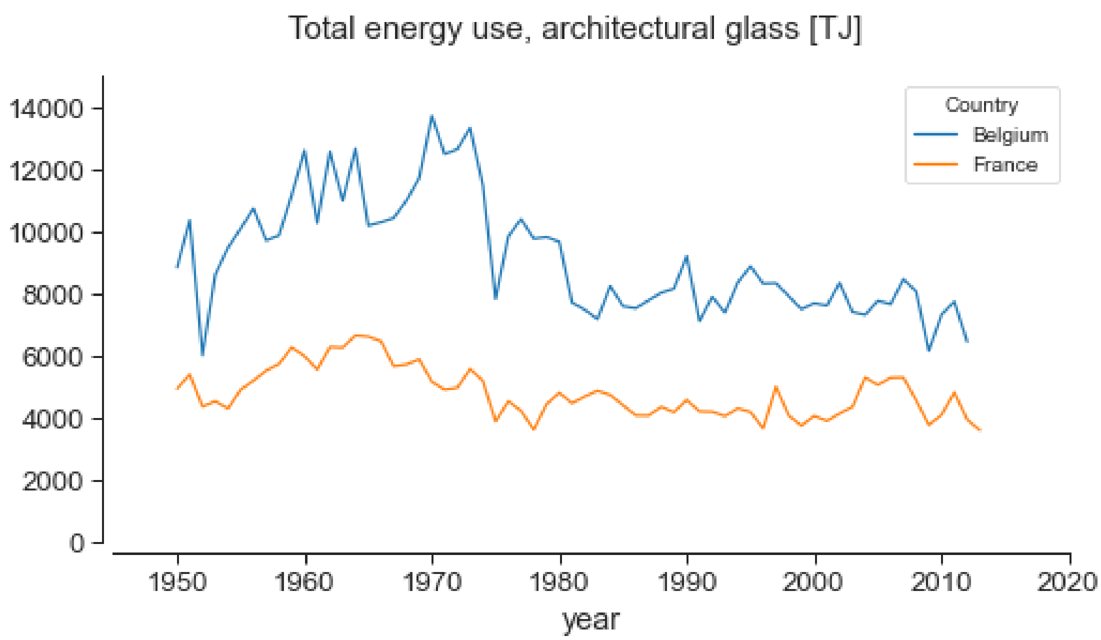
[170]:
```python
nrows = len(energies)
ncols = len(countries)

fig, axes = plt.subplots(nrows=nrows, ncols=ncols,
                         sharex=True, sharey=True,
                         figsize=(16, 10))

for row, energy in enumerate(energies):
    for col, country in enumerate(countries):
        ax = axes[row][col]
```

```
        df_be_fr[(f"{energy}, architectural glass [TJ]", country)].plot(ax=ax)

        if row == 0:
            ax.set_title(country)
        if col == 0:
            ax.set_ylabel(energy)

        style_ax(ax)

ax.set_xlim(1945, 2020)
ax.set_ylim(0, 12000)

sns.despine(offset=5)
plt.show()
```

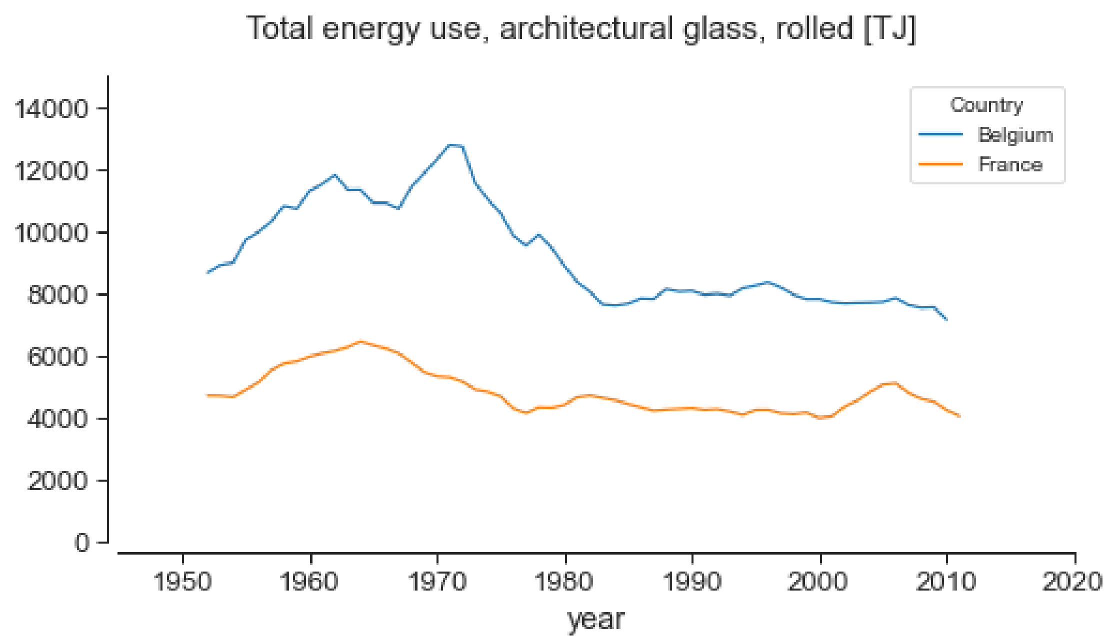


[171]:
```
for country in countries:
    df_be_fr[('Total energy use, architectural glass [TJ]', country)] = 0

for energy in energies:
    df_be_fr['Total energy use, architectural glass [TJ]'] = (
        df_be_fr[f"{energy}, architectural glass [TJ]"]
        + df_be_fr['Total energy use, architectural glass [TJ]']
    )
```

[172]:
```
fig, ax = plt.subplots(figsize=(8, 4))

(df_be_fr['Total energy use, architectural glass [TJ]']).plot(ax=ax)

ax.set_xlim(1945, 2020)
ax.set_ylim(0, 15000)

fig.suptitle('Total energy use, architectural glass [TJ]', fontsize=15)
sns.despine(offset=5)
plt.show()
```



[173]:
```
fig, ax = plt.subplots(figsize=(8, 4))

(df_be_fr['Total energy use, architectural glass [TJ]']
 .rolling(5, center=True).mean().plot(ax=ax))

ax.set_xlim(1945, 2020)
ax.set_ylim(0, 15000)

fig.suptitle('Total energy use, architectural glass, rolled [TJ]',
             fontsize=15)
sns.despine(offset=5)
plt.show()
```



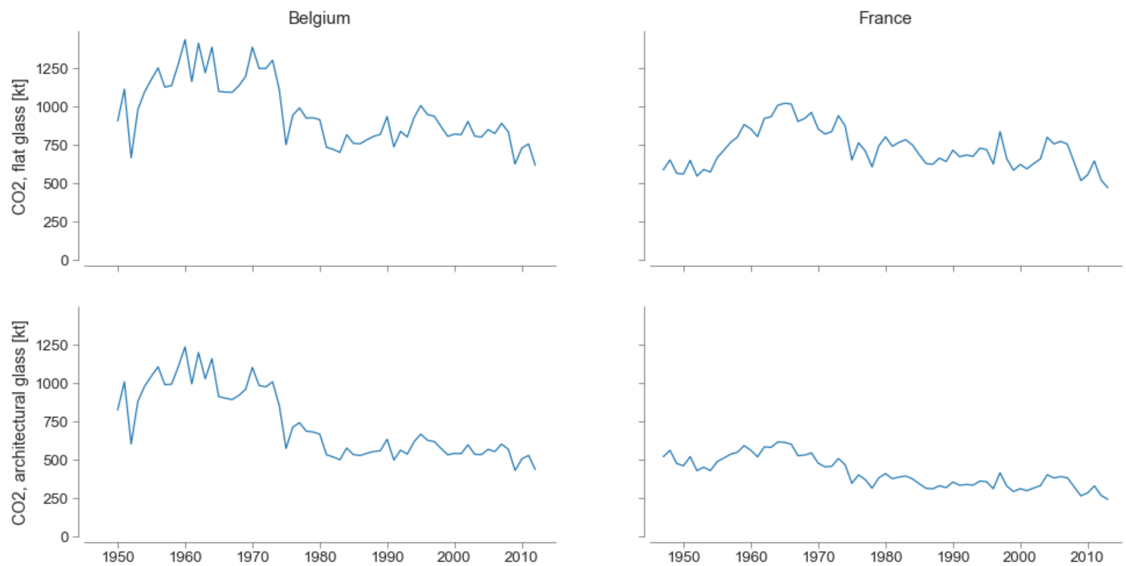
7.5 Absolute CO2 Emissions in Belgium and France since 1945

[174]:
```
# Estimate the CO2 emission related to flat glass production:
for country in countries:
    df_be_fr[('CO2, flat glass [kt]', country)] = (
        df_be_fr[('Production [kt]', country)]
        * fr_co2_intensity['CO2 interpolated [kg/t]'] / 1000
    )
```

[175]:
```
# Estimate the CO2 emission related to architectural flat glass production:
for country in countries:
    df_be_fr[('CO2, architectural glass [kt]', country)] = (
        df_be_fr[('Architectural glass production [kt]', country)]
        * fr_co2_intensity['CO2 interpolated [kg/t]'] / 1000
    )
```

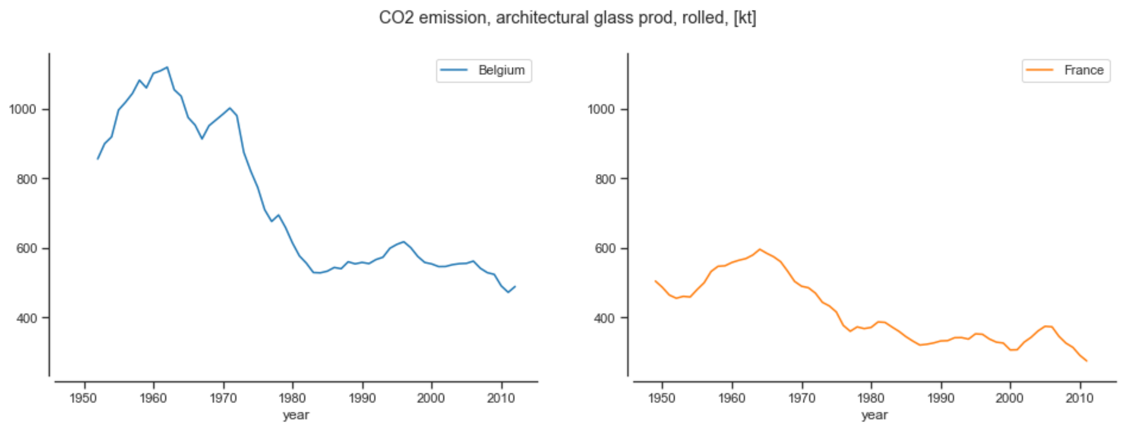
[176]:
```
nrows = len(CO2_G)
ncols = len(countries)

fig, axes = plt.subplots(nrows=nrows, ncols=ncols,
                         sharex=True, sharey=True,
                         figsize=(16, 8))

for row, CO2 in enumerate(CO2_G):
    for col, country in enumerate(countries):
        ax = axes[row][col]
```

```python
        df_be_fr[(f"{CO2}", country)].plot(ax=ax)
        if row == 0:
            ax.set_title(country)
        if col == 0:
            ax.set_ylabel(CO2)

        style_ax(ax)

ax.set_xlim(1945, 2015)
ax.set_ylim(ymin=0)

sns.despine(offset=5)
plt.show()
```



[177]:
```python
# 5-y. rolling of the CO2 emission related to architectural glass prod
# and plot:
with sns.plotting_context("notebook"):
    # Rolling flat glass production according to a 5-year mean
    (df_be_fr['CO2, architectural glass [kt]']
     .interpolate(method="linear", limit_area='inside')
     .rolling(5, center=True)
     .mean()).plot(subplots=True,
                   grid=False,
                   layout=(1, 2),
                   figsize=(16, 5),
                   sharex=True, sharey=True,
                   title=(
                       "CO2 emission, architectural glass prod, rolled, [kt]")
```

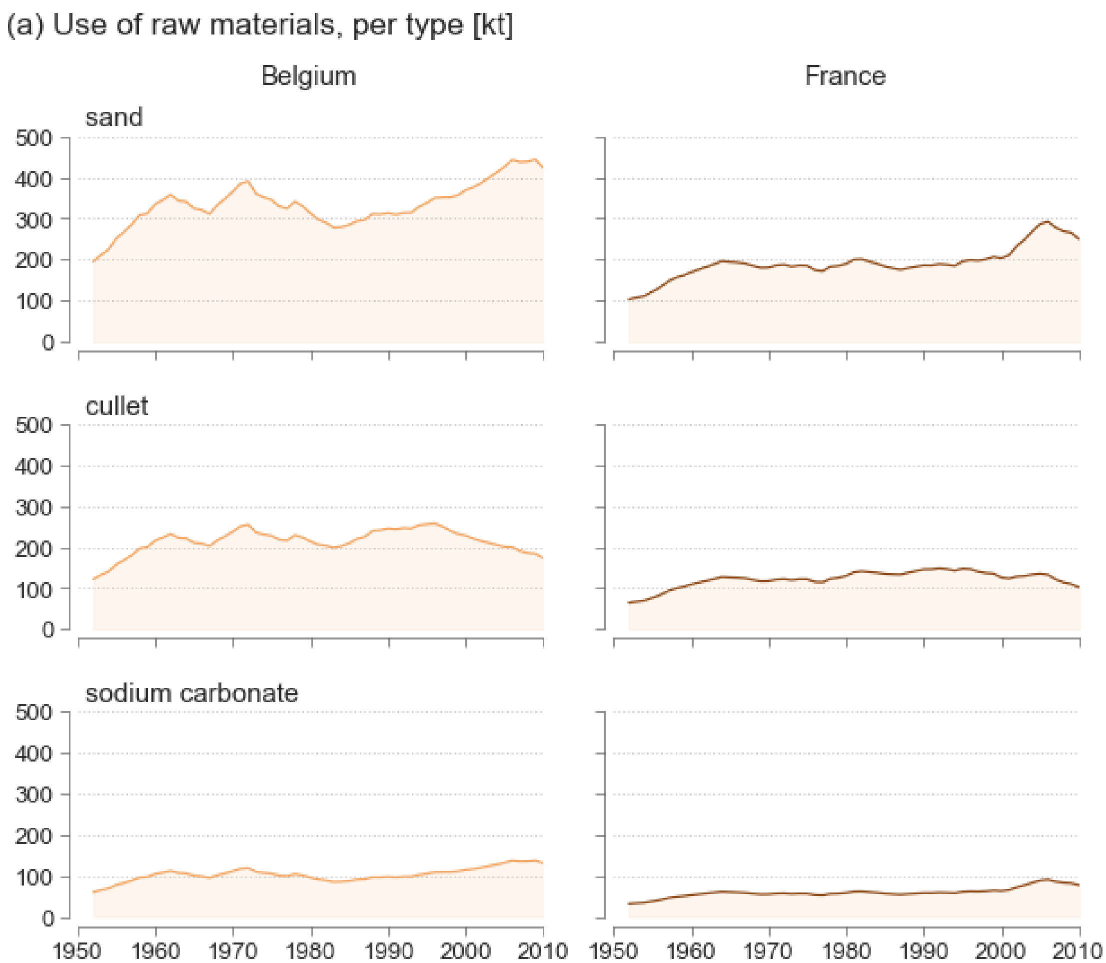
```
                   )

    ax.set_xlim(1945, 2015)
    ax.set_ylim(ymin=0)
    sns.despine(offset=5)
    plt.show()
```



7.6 Summary

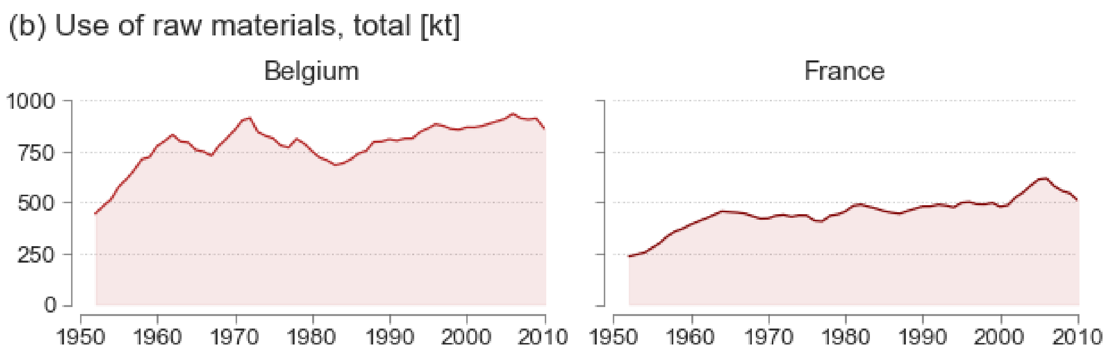
[178]:
```
# Raw materials, per type:
mat_reduced = ['sand', 'cullet', 'sodium carbonate']

nrows = len(mat_reduced)
ncols = len(countries)

fig, axes = plt.subplots(nrows=nrows, ncols=ncols,
                         sharex=True, sharey=True,
                         figsize=(10, 8))

for row, material in enumerate(mat_reduced):
    for col, country in enumerate(countries):
        ax = axes[row][col]
        if country == 'Belgium':
            color = 'sandybrown'
            ax.set_title(f" {material}", loc='left')
        else:
            color = 'saddlebrown'

        (df_be_fr[(f"{material}, architectural glass [kt]", country)]
         .interpolate(method="linear", limit_area='inside')
         .rolling(5, center=True).mean()
         .plot(ax=ax, c=color))
```

```
        x = df_be_fr.index
        y = (df_be_fr[(f"{material}, architectural glass [kt]", country)]
             .interpolate(method="linear", limit_area='inside')
             .rolling(5, center=True).mean())
        ax.fill_between(x, y, color="sandybrown", alpha=0.1)

        ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
        style_ax(ax)
        if row == 0:
            ax.set_title(country, y=1.2)

ax.set_xlim(1950, 2010)
ax.set_ylim(0, 500)
ax.set_xticks(np.arange(1950, 2011, 10))
ax.set_yticks(np.arange(0, 501, 100))

fig.suptitle("(a) Use of raw materials, per type [kt]",
             x=0.07, y=1, ha='left')
fig.subplots_adjust(wspace=0.15, hspace=0.4)

sns.despine(offset=5)
plt.show()

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'AppendixD_BEFR_UseOfRawMaterials.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'AppendixD_BEFR_UseOfRawMaterials.pdf'),
                bbox_inches='tight')
```



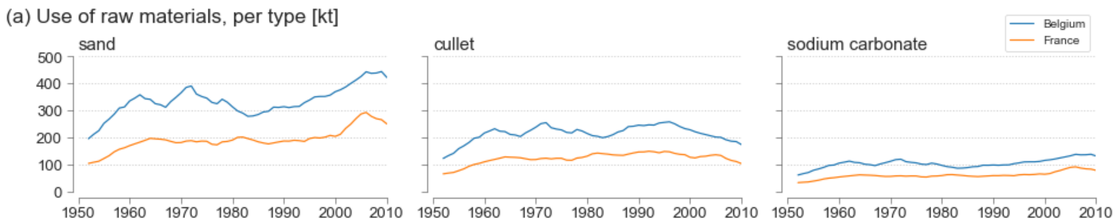
[179]:
```python
# Raw materials, total:
ncols = len(countries)

fig, axes = plt.subplots(nrows=1, ncols=ncols,
                         sharex=True, sharey=True,
                         figsize=(10, 2.1))

for col, country in enumerate(countries):
    ax = axes[col]
    if country == 'Belgium':
        color = 'firebrick'
    else:
        color = 'maroon'

    (df_be_fr[("Total raw material use, architectural glass [kt]", country)]
     .interpolate(method="linear", limit_area='inside')
     .rolling(5, center=True).mean()
```

```
            .plot(ax=ax, c=color))

        x = df_be_fr.index
        y = (df_be_fr[("Total raw material use, architectural glass [kt]", country)]
             .interpolate(method="linear", limit_area='inside')
             .rolling(5, center=True).mean())
        ax.fill_between(x, y, color="firebrick", alpha=0.1)

        ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
        style_ax(ax)
        ax.set_title(country, y=1.05)

ax.set_xlim(1950, 2010)
plt.xticks(np.arange(1950, 2011, 10))

ax.set_ylim(0, 1000)

fig.suptitle("(b) Use of raw materials, total [kt]",
             x=0.07, y=1.2, ha='left')
fig.subplots_adjust(wspace=0.15, hspace=0.4)

sns.despine(offset=5)
plt.show()

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'AppendixD_BEFR_TotalRawMaterials.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'AppendixD_BEFR_TotalRawMaterials.pdf'),
                bbox_inches='tight')
```



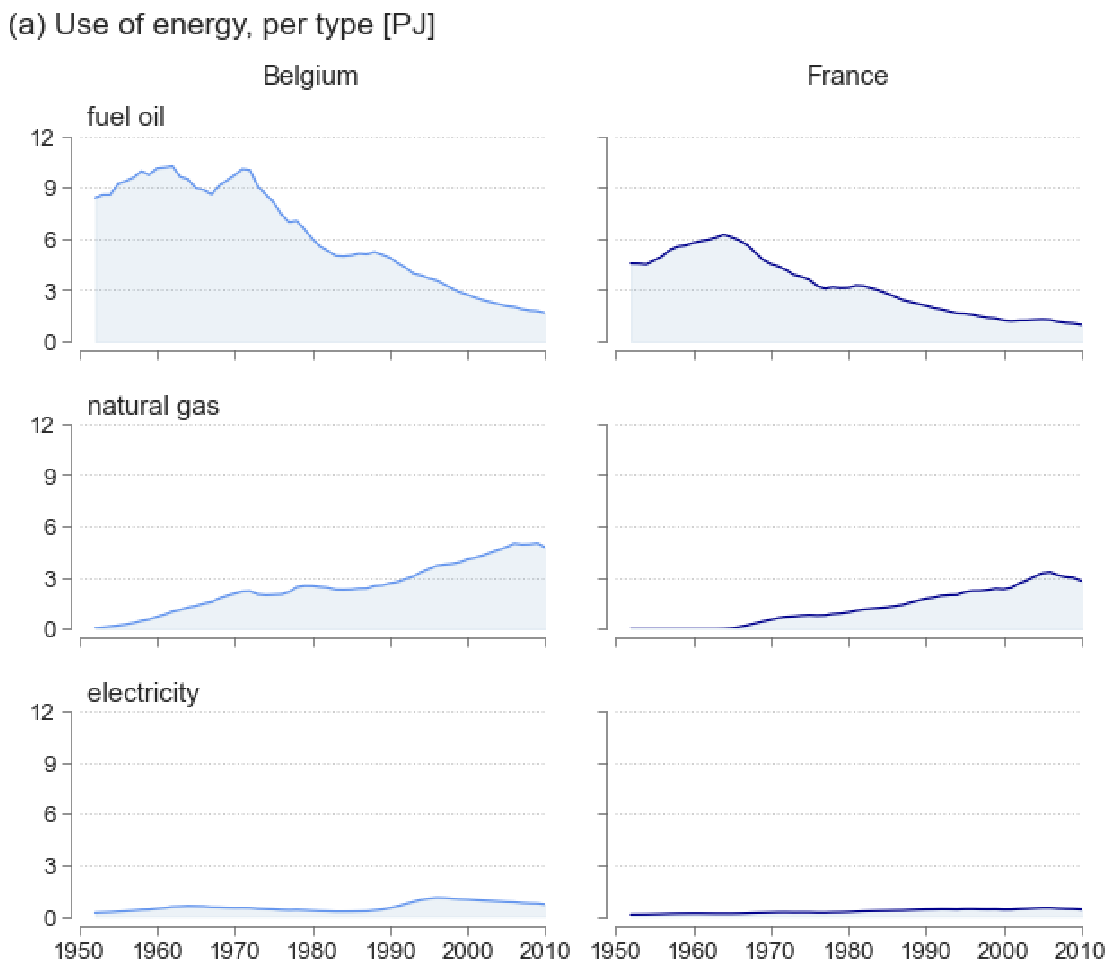
[180]:
```
toplot = ['sand', 'cullet', 'sodium carbonate']

fig, axes = plt.subplots(nrows=1, ncols=3,
```

```
                         sharex=True, sharey=True,
                         figsize=(15.5, 2.1))
col = 0

for plot in toplot:
    ax = axes[col]
    ax.set_title(plot, loc='left')
    col += 1
    (df_be_fr[f"{plot}, architectural glass [kt]"]
     .interpolate(method="linear", limit_area='inside')
     .rolling(5, center=True).mean().plot(ax=ax))
    ax.get_legend().remove()
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

    style_ax(ax)

ax.set_xlim(1950, 2010)
plt.xticks(np.arange(1950, 2011, 10))

ax.set_ylim(0, 500)
plt.yticks(np.arange(0, 501, 100))

# Add legend:
handles, labels = ax.get_legend_handles_labels()
fig.legend(handles, labels, ncol=1, bbox_to_anchor=(0.9, 1.15))

fig.suptitle("(a) Use of raw materials, per type [kt]",
             x=0.07, y=1.15, ha='left')
fig.subplots_adjust(wspace=0.15, hspace=0.4)

sns.despine(offset=5)
plt.show()

if export:
    # Save image:
    fig.savefig(os.path.join(
        path_img, 'AppendixD_BEFR_RawMaterials_3Graphs.png'),
        dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(
        path_img, 'AppendixD_BEFR_RawMaterials_3Graphs.pdf'),
        bbox_inches='tight')
```



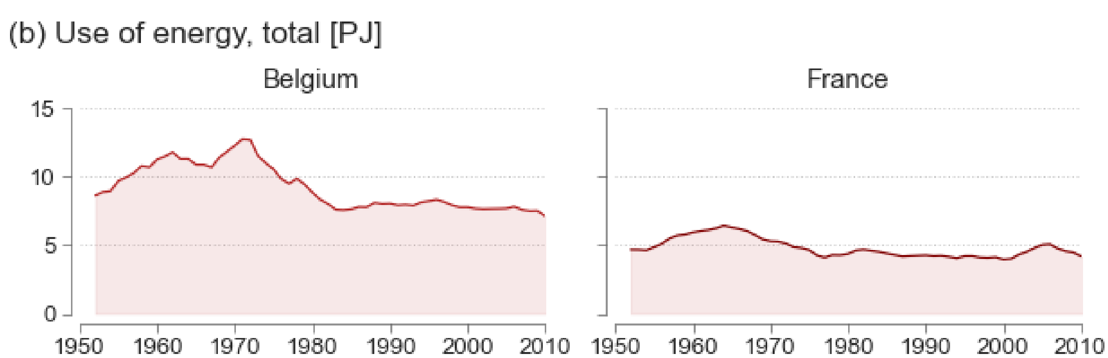
[181]:
```python
# Energy use, per type:
nrows = len(energies)
ncols = len(countries)

fig, axes = plt.subplots(nrows=nrows, ncols=ncols,
                         sharex=True, sharey=True,
                         figsize=(10, 8))

for row, energy in enumerate(energies):
    for col, country in enumerate(countries):
        ax = axes[row][col]
        if country == 'Belgium':
            color = 'cornflowerblue'
            ax.set_title(f" {energy}", loc='left')
        else:
            color = 'darkblue'

        (df_be_fr[(f"{energy}, architectural glass [TJ]", country)]
         .interpolate(method="linear", limit_area='inside')
         .rolling(5, center=True).mean()
         .plot(ax=ax, c=color)
         )

        x = df_be_fr.index
        y = (df_be_fr[(f"{energy}, architectural glass [TJ]", country)]
             .interpolate(method="linear", limit_area='inside')
             .rolling(5, center=True).mean())
        ax.fill_between(x, y, color="steelblue", alpha=0.1)

        # Divide by 1000 the y-axis, results in PJ
        ax.yaxis.set_major_formatter(y_1000)

        ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
        style_ax(ax)
        if row == 0:
            ax.set_title(country, y=1.2)
```

```
ax.set_xlim(1950, 2010)
ax.set_ylim(0, 12000)
ax.minorticks_off()

plt.xticks(np.arange(1950, 2011, 10))
plt.yticks(np.arange(0, 12001, 3000))

fig.suptitle("(a) Use of energy, per type [PJ]",
             x=0.07, y=1, ha='left')
fig.subplots_adjust(wspace=0.15, hspace=0.4)

sns.despine(offset=5)
plt.show()

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'AppendixE_BEFR_UseOfEnergy.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'AppendixE_BEFR_UseOfEnergy.pdf'),
                bbox_inches='tight')
```



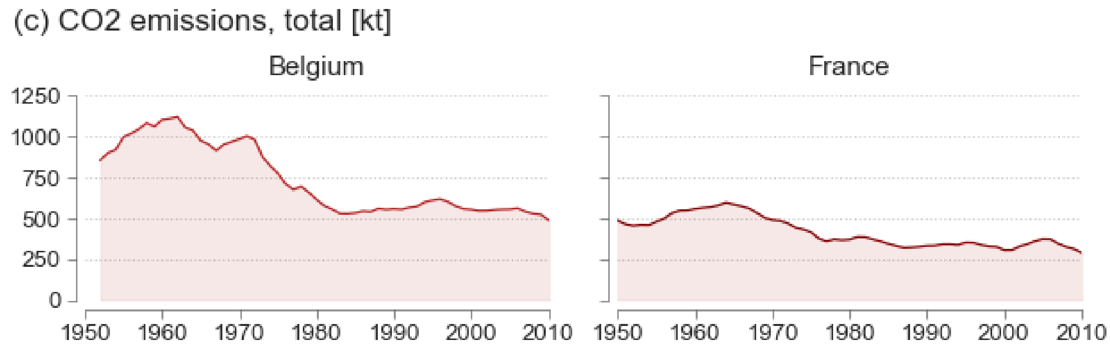
[182]:
```
# Energy use, total:
ncols = len(countries)

fig, axes = plt.subplots(nrows=1, ncols=ncols,
                         sharex=True, sharey=True,
                         figsize=(10, 2.1))

for col, country in enumerate(countries):
    ax = axes[col]
    if country == 'Belgium':
        color = 'firebrick'
    else:
        color = 'maroon'

    (df_be_fr[('Total energy use, architectural glass [TJ]', country)]
     .interpolate(method="linear", limit_area='inside')
     .rolling(5, center=True).mean()
```

```python
        .plot(ax=ax, c=color))

    x = df_be_fr.index
    y = (df_be_fr[('Total energy use, architectural glass [TJ]', country)]
        .interpolate(method="linear", limit_area='inside')
        .rolling(5, center=True).mean())
    ax.fill_between(x, y, color="firebrick", alpha=0.1)

    # Divide by 1000 the y-axis, results in PJ
    ax.yaxis.set_major_formatter(y_1000)

    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
    style_ax(ax)
    ax.set_title(country, y=1.05)

ax.set_xlim(1950, 2010)
ax.set_ylim(0, 15000)

plt.xticks(np.arange(1950, 2011, 10))

fig.suptitle("(b) Use of energy, total [PJ]",
             x=0.07, y=1.2, ha='left')
fig.subplots_adjust(wspace=0.15, hspace=0.4)

sns.despine(offset=5)
plt.show()

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'AppendixE_BEFR_TotalEnergy.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'AppendixE_BEFR_TotalEnergy.pdf'),
                bbox_inches='tight')
```



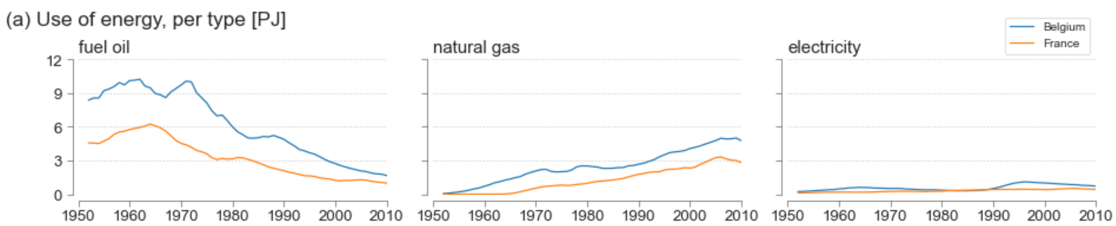
[183]:
```python
# CO2 emissions, total:
ncols = len(countries)

fig, axes = plt.subplots(nrows=1, ncols=ncols,
                         sharex=True, sharey=True,
                         figsize=(10, 2.1))

for col, country in enumerate(countries):
    ax = axes[col]
    ax = axes[col]
    if country == 'Belgium':
        color = 'firebrick'
    else:
        color = 'maroon'

    (df_be_fr[('CO2, architectural glass [kt]', country)]
     .interpolate(method="linear", limit_area='inside')
     .rolling(5, center=True).mean()
     .plot(ax=ax, c=color))

    x = df_be_fr.index
    y = (df_be_fr[('CO2, architectural glass [kt]', country)]
         .interpolate(method="linear", limit_area='inside')
         .rolling(5, center=True).mean())
    ax.fill_between(x, y, color="firebrick", alpha=0.1)

    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)
    style_ax(ax)
    ax.set_title(country, y=1.05)

ax.set_xlim(1950, 2010)
ax.set_ylim(0, 1250)

plt.xticks(np.arange(1950, 2011, 10))
plt.yticks(np.arange(0, 1251, 250))

fig.suptitle("(c) CO2 emissions, total [kt]",
             x=0.07, y=1.2, ha='left')
fig.subplots_adjust(wspace=0.15, hspace=0.4)

sns.despine(offset=5)
plt.show()

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'AppendixE_BEFR_TotalCO2.png'),
                dpi=600, bbox_inches='tight')
```

```
    fig.savefig(os.path.join(path_img, 'AppendixE_BEFR_TotalCO2.pdf'),
                bbox_inches='tight')
```



[184]:
```
fig, axes = plt.subplots(nrows=1, ncols=3,
                         sharex=True, sharey=True,
                         figsize=(15.5, 2.1))
col = 0

for energy in energies:
    ax = axes[col]
    ax.set_title(energy, loc='left')
    col += 1
    (df_be_fr[f"{energy}, architectural glass [TJ]"]
     .interpolate(method="linear", limit_area='inside')
     .rolling(5, center=True).mean().plot(ax=ax))
    ax.get_legend().remove()
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

    # Divide by 1000 the y-axis, results in PJ
    ax.yaxis.set_major_formatter(y_1000)
    ax.minorticks_off()
    style_ax(ax)

ax.set_xlim(1950, 2010)
plt.xticks(np.arange(1950, 2011, 10))

ax.set_ylim(0, 12000)
plt.yticks(np.arange(0, 12001, 3000))

# Add legend:
handles, labels = ax.get_legend_handles_labels()
fig.legend(handles, labels, ncol=1, bbox_to_anchor=(0.9, 1.15))

fig.suptitle("(a) Use of energy, per type [PJ]",
```

```
            x=0.07, y=1.15, ha='left')
fig.subplots_adjust(wspace=0.15, hspace=0.4)

sns.despine(offset=5)
plt.show()

if export:
    # Save image:
    fig.savefig(os.path.join(
        path_img, 'AppendixE_BEFR_UseOfEnergy_3Graphs.png'),
        dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(
        path_img, 'AppendixE_BEFR_UseOfEnergy_3Graphs.pdf'),
        bbox_inches='tight')
```



[185]:
```
(df_be_fr[('CO2, architectural glass [kt]', 'France')]
 .interpolate(method="linear", limit_area='inside')
 .rolling(5, center=True).mean())[2010]
```

[185]: 289.40848192628147

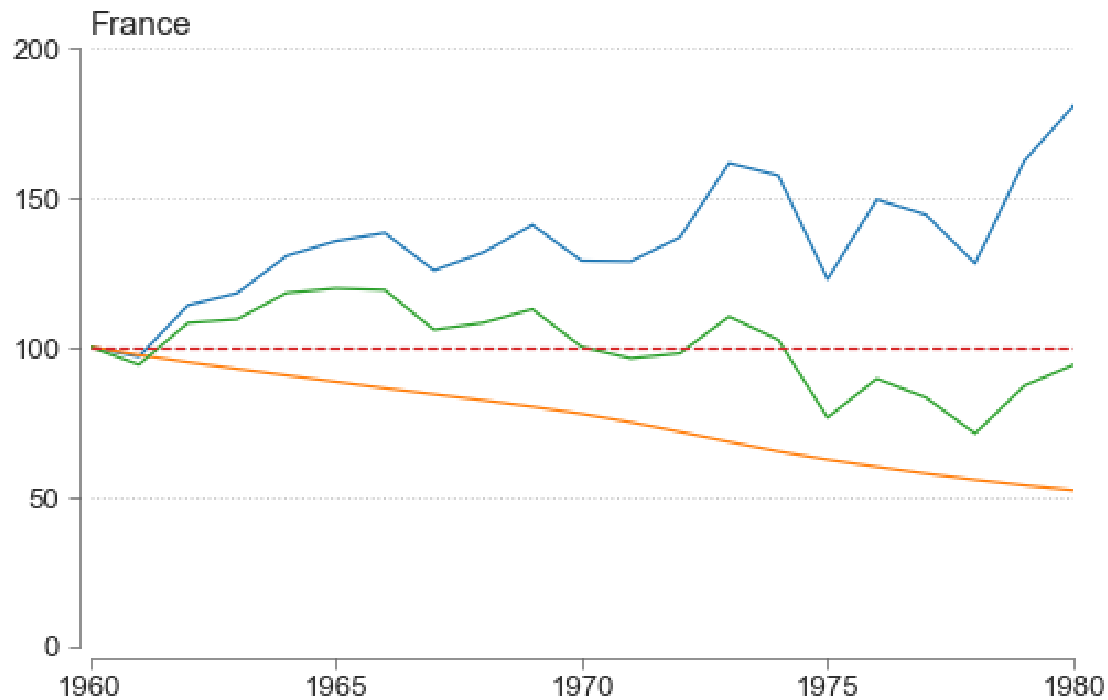
8 Relative v. Absolute Decoupling: A Base 100 Analysis

8.1 Flat Glass Production, France and Belgium, 1960-1980

[186]:
```
D1 = 1960
D2 = 1980
```

[187]:
```
# Base 100 = 1960
FGProd_diff = (100 * df_be_fr['Production [kt]']
               / df_be_fr['Production [kt]'].loc[D1])

co2intensity_diff = 100 * fr_co2_intensity / fr_co2_intensity.loc[D1]

co2abs_diff = (100 * df_be_fr['CO2, flat glass [kt]']
               / df_be_fr['CO2, flat glass [kt]'].loc[D1])
```

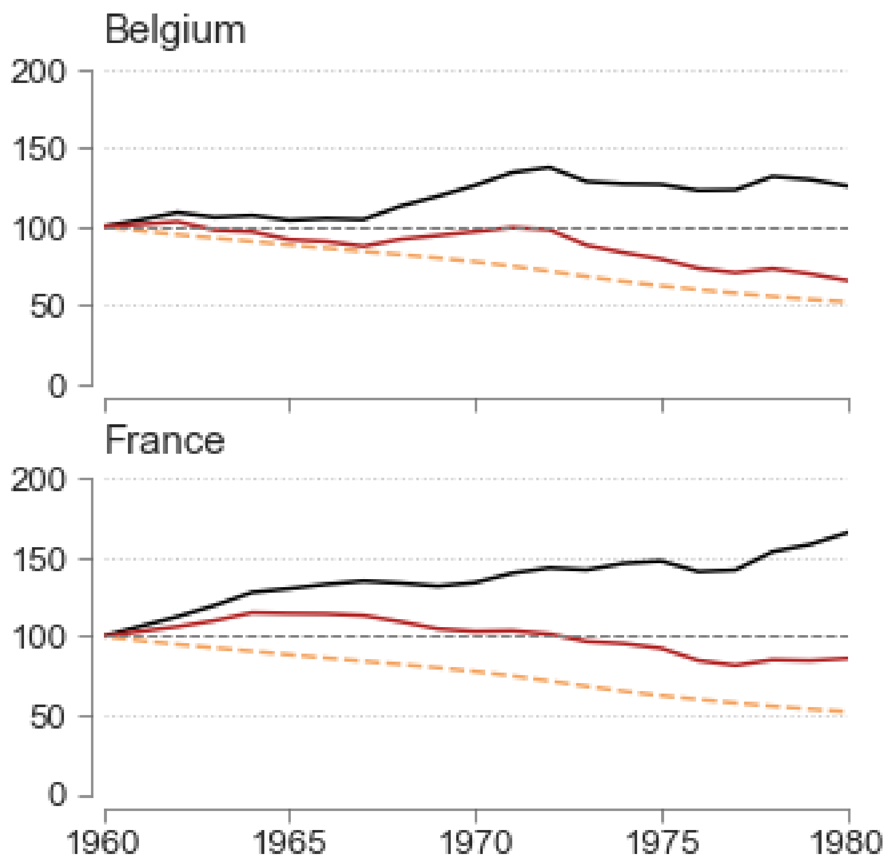
[188]:
```python
# Plot production, carbon intensity and CO2 absolute emissions for France:
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=FGProd_diff,
             x=FGProd_diff.index,
             y='France',
             ax=ax)

ax.plot(co2intensity_diff.index,
        co2intensity_diff['CO2 interpolated [kg/t]'])

ax.plot(co2abs_diff.index,
        co2abs_diff['France'])

ax.plot([D1, D2], [100, 100], '--')
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

ax.set_ylim(0, 200)
plt.yticks(np.arange(0, 201, 50))
ax.set_xlim(D1, D2)
plt.xticks(np.arange(D1, D2+1, 5))

ax.set_title("France", loc='left')
ax.set_ylabel(None)
style_ax(ax)

sns.despine(offset=5)
```



[189]:
```
# Base 100 = 1960, with a 5-year moving average
fg_prod_diffroll = (100 * df_be_fr['Production [kt]']
                     .interpolate(method="linear", limit_area='inside')
                     .rolling(5, center=True).mean()
                    / df_be_fr['Production [kt]']
                     .interpolate(method="linear", limit_area='inside')
                     .rolling(5, center=True).mean().loc[D1])

co2abs_diffroll = (100 * (df_be_fr['CO2, flat glass [kt]']
                          .interpolate(method="linear", limit_area='inside')
                          .rolling(5, center=True).mean())
                   / (df_be_fr['CO2, flat glass [kt]']
                      .interpolate(method="linear", limit_area='inside')
                      .rolling(5, center=True).mean().loc[D1]))
```

[190]:
```
# Plot production, carbon intensity, CO2 absolute emissions for France:
fig, axes = plt.subplots(nrows=2, ncols=1,
                         sharex=True, sharey=True,
                         figsize=(5, 5))



for col, country in enumerate(countries):
    ax = axes[col]
```

```python
    sns.lineplot(data=fg_prod_diffroll,
                 x=fg_prod_diffroll.index,
                 y=f'{country}',
                 ax=ax, color='black', linewidth=1.5)

    ax.plot(co2intensity_diff.index,
            co2intensity_diff['CO2 interpolated [kg/t]'],
            color='sandybrown', linestyle='--', linewidth=1.5)

    ax.plot(co2abs_diffroll.index,
            co2abs_diffroll[f'{country}'],
            color='firebrick', linewidth=1.5)

    ax.set_xlim(D1, D2)
    plt.xticks(np.arange(D1, D2+1, 5))
    ax.set_ylabel(None)

    ax.plot([D1, D2], [100, 100], '--', c='grey', linewidth=1)
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

    ax.set_title(f"{country}", loc='left', pad=10)
    style_ax(ax)

ax.set_ylim(0, 200)
plt.yticks(np.arange(0, 201, 50))

fig.subplots_adjust(hspace=0.3)

sns.despine(offset=5)

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig8_FlatGlass_Base100_60-80.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig8_FlatGlass_Base100_60-80.pdf'),
                bbox_inches='tight')
```



[191]:
```
for country in countries:
    print(f"Flat Glass Prod in {country}:\n",
          '%.2f' % fg_prod_diffroll[f'{country}'].loc[D2])
    print(f"CO2 absolute emissions in {country}:\n",
          '%.2f' % co2abs_diffroll[f'{country}'].loc[D2])
    print("CO2 intensity:\n",
          '%.2f' % co2intensity_diff['CO2 interpolated [kg/t]'].loc[D2])
```

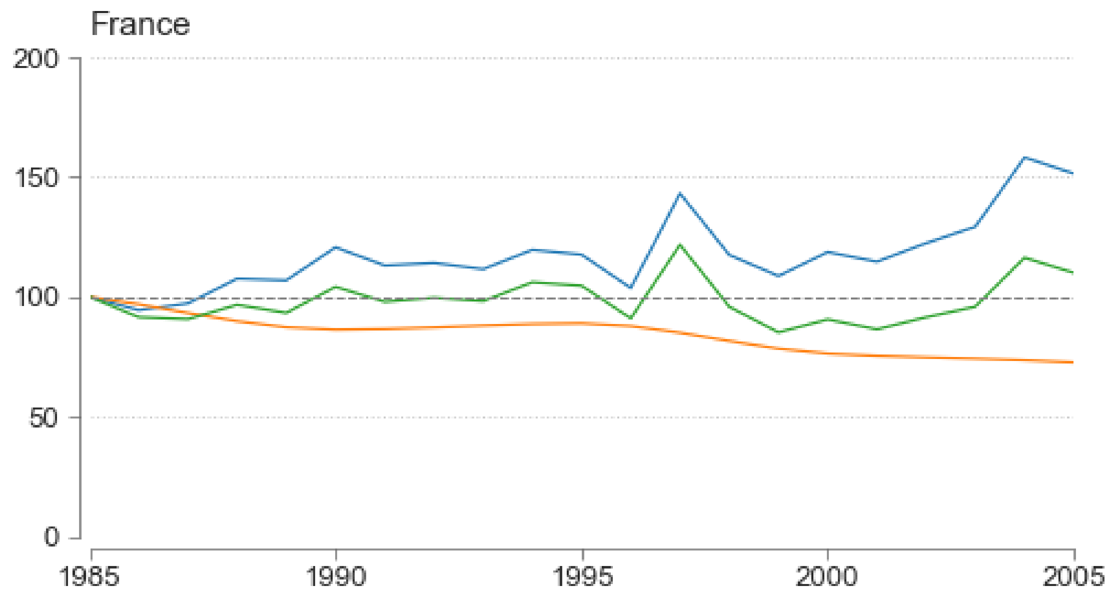
```
Flat Glass Prod in Belgium:
 125.74
CO2 absolute emissions in Belgium:
 65.73
CO2 intensity:
 52.07
Flat Glass Prod in France:
 165.57
CO2 absolute emissions in France:
 85.88
CO2 intensity:
 52.07
```

8.2 Flat Glass Production, France and Belgium, 1985-2005

[192]:
```
D3 = 1985
D4 = 2005
```

[193]:
```
# Base 100 = 1985
FGProd_diff = (100 * df_be_fr['Production [kt]']
               / df_be_fr['Production [kt]'].loc[D3])

co2intensity_diff = 100 * fr_co2_intensity / fr_co2_intensity.loc[D3]

co2abs_diff = (100 * df_be_fr['CO2, flat glass [kt]']
               / df_be_fr['CO2, flat glass [kt]'].loc[D3])
```

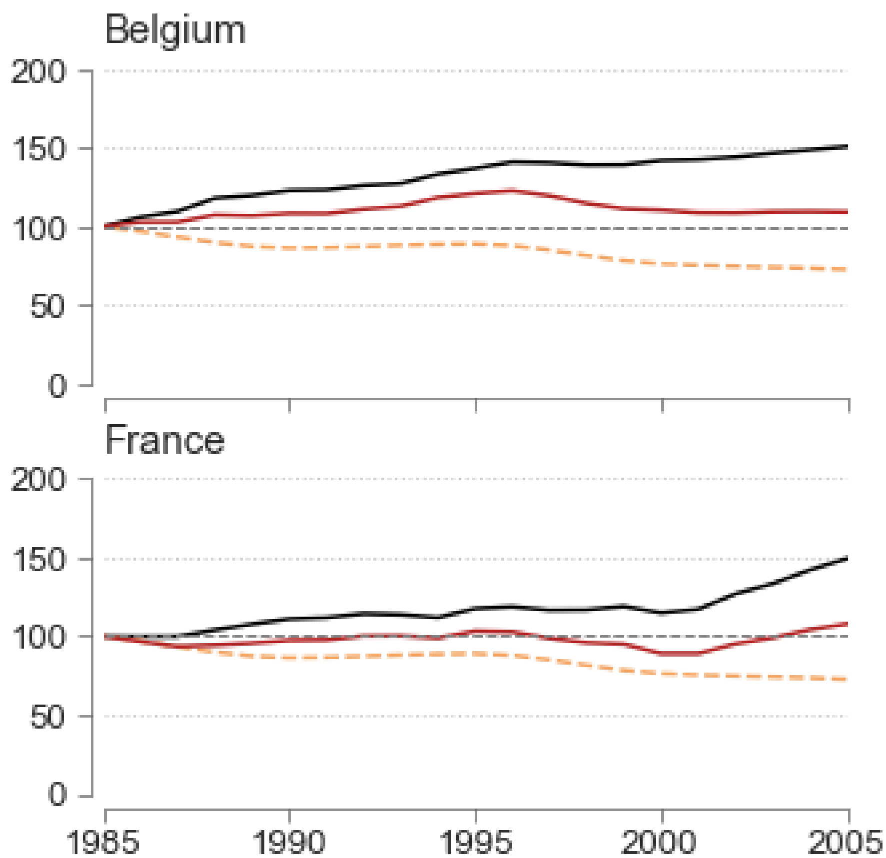
[194]:
```
# Plot production, carbon intensity and CO2 absolute emissions for France:
fig, ax = plt.subplots(figsize=(8, 4))

sns.lineplot(data=FGProd_diff,
             x=FGProd_diff.index,
             y='France',
             ax=ax)

ax.plot(co2intensity_diff.index,
        co2intensity_diff['CO2 interpolated [kg/t]'])
ax.plot(co2abs_diff.index,
        co2abs_diff['France'])

ax.plot([D3, D4], [100, 100], '--', c='grey', linewidth=1)
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

ax.set_ylim(0, 200)
plt.yticks(np.arange(0, 201, 50))
ax.set_xlim(D3, D4)
plt.xticks(np.arange(D3, D4+1, 5))

ax.set_title("France", loc='left', pad=10)
ax.set_ylabel(None)
style_ax(ax)

sns.despine(offset=5)
```




```
[195]: # Base 100 = 1985, with a 5-year rolling
fg_prod_diffroll = (100 * df_be_fr['Production [kt]']
                    .rolling(5, center=True).mean()
                    / df_be_fr['Production [kt]']
                    .rolling(5, center=True).mean().loc[D3])

co2abs_diffroll = (100 * (df_be_fr['CO2, flat glass [kt]']
                          .interpolate(method="linear", limit_area='inside')
                          .rolling(5, center=True).mean())
                   / (df_be_fr['CO2, flat glass [kt]']
                       .interpolate(method="linear", limit_area='inside')
                       .rolling(5, center=True).mean().loc[D3]))
```

```
[196]: # Plot production, carbon intensity, CO2 absolute emissions for France:
fig, axes = plt.subplots(nrows=2, ncols=1,
                         sharex=True, sharey=True,
                         figsize=(5, 5))

for col, country in enumerate(countries):
    ax = axes[col]

    sns.lineplot(data=fg_prod_diffroll,
                 x=fg_prod_diffroll.index,
                 y=f'{country}',
                 ax=ax, color='black', linewidth=1.5)

    ax.plot(co2intensity_diff.index,
```

```
            co2intensity_diff['CO2 interpolated [kg/t]'],
            color='sandybrown', linestyle='--', linewidth=1.5)

    ax.plot(co2abs_diffroll.index,
            co2abs_diffroll[f'{country}'],
            color='firebrick', linewidth=1.5)

    ax.set_xlim(D3, D4)
    plt.xticks(np.arange(D3, D4+1, 5))
    ax.set_ylabel(None)

    ax.plot([D3, D4], [100, 100], '--', c='grey', linewidth=1)
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

    ax.set_title(f"{country}", loc='left', pad=10)
    style_ax(ax)

ax.set_ylim(0, 200)
plt.yticks(np.arange(0, 201, 50))

fig.subplots_adjust(hspace=0.3)

sns.despine(offset=5)

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig8_FlatGlass_Base100_85-05.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig8_FlatGlass_Base100_85-05.pdf'),
                bbox_inches='tight')
```



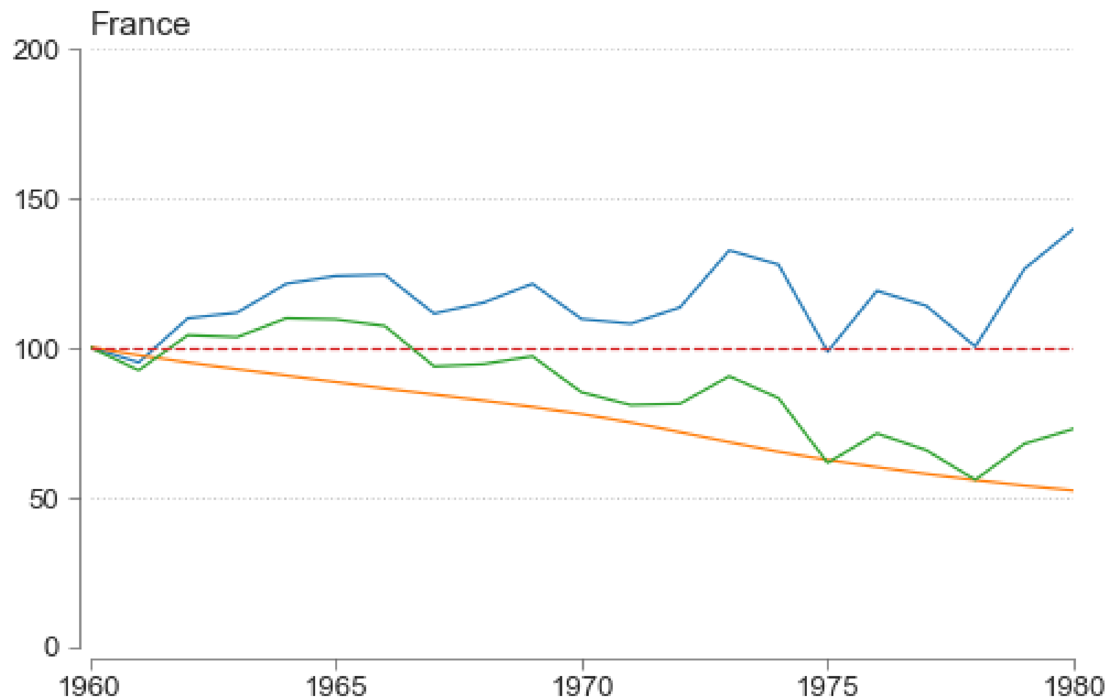
[197]:
```
for country in countries:
    print(f"Flat Glass Prod in {country}:\n",
          '%.2f' % fg_prod_diffroll[f'{country}'].loc[D4])
    print(f"CO2 absolute emissions in {country}:\n",
          '%.2f' % co2abs_diffroll[f'{country}'].loc[D4])
    print("CO2 intensity:\n",
          '%.2f' % co2intensity_diff['CO2 interpolated [kg/t]'].loc[D4])
```

```
Flat Glass Prod in Belgium:
 150.80
CO2 absolute emissions in Belgium:
 109.27
CO2 intensity:
 72.63
Flat Glass Prod in France:
 149.35
CO2 absolute emissions in France:
 107.78
CO2 intensity:
 72.63
```

8.3 Architectural Glass Production, France and Belgium, 1960-1980

[198]:
```
# Base 100 = 1960
FGProd_diff = (100 * df_be_fr['Architectural glass production [kt]']
               / df_be_fr['Architectural glass production [kt]'].loc[D1])

co2intensity_diff = 100 * fr_co2_intensity / fr_co2_intensity.loc[D1]

co2abs_diff = (100 * df_be_fr['CO2, architectural glass [kt]']
               / df_be_fr['CO2, architectural glass [kt]'].loc[D1])
```

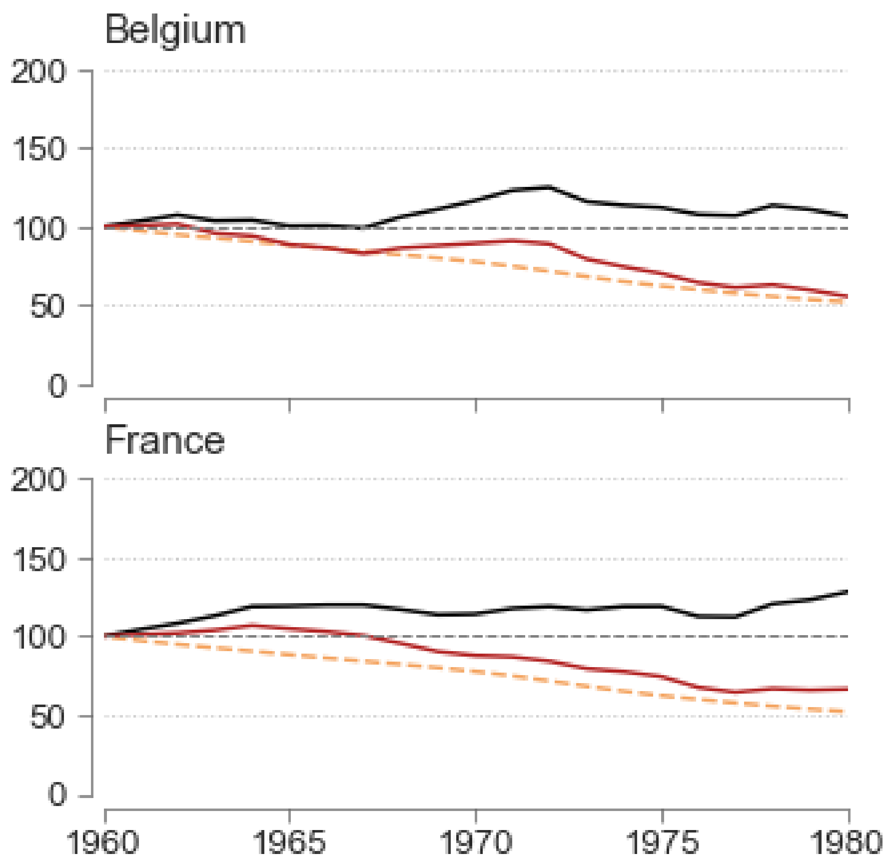
[199]:
```
# Plot production, carbon intensity and CO2 absolute emissions for France:
fig, ax = plt.subplots(figsize=(8, 5))

sns.lineplot(data=FGProd_diff,
             x=FGProd_diff.index,
             y='France',
             ax=ax)

ax.plot(co2intensity_diff.index,
        co2intensity_diff['CO2 interpolated [kg/t]'])

ax.plot(co2abs_diff.index,
        co2abs_diff['France'])

ax.plot([D1, D2], [100, 100], '--')
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

ax.set_ylim(0, 200)
plt.yticks(np.arange(0, 201, 50))
ax.set_xlim(D1, D2)
plt.xticks(np.arange(D1, D2+1, 5))

ax.set_title("France", loc='left')
ax.set_ylabel(None)
style_ax(ax)

sns.despine(offset=5)
```



[200]:
```
# Base 100 = 1960, with a 5-year moving average
fg_prod_diffroll = (100 * df_be_fr['Architectural glass production [kt]']
                    .interpolate(method="linear", limit_area='inside')
                    .rolling(5, center=True).mean()
                    / df_be_fr['Architectural glass production [kt]']
                    .interpolate(method="linear", limit_area='inside')
                    .rolling(5, center=True).mean().loc[D1])

co2abs_diffroll = (100 * (df_be_fr['CO2, architectural glass [kt]']
                          .interpolate(method="linear", limit_area='inside')
                          .rolling(5, center=True).mean())
                   / (df_be_fr['CO2, architectural glass [kt]']
                      .interpolate(method="linear", limit_area='inside')
                      .rolling(5, center=True).mean().loc[D1]))
```

[201]:
```
# Plot production, carbon intensity, CO2 absolute emissions for France:
fig, axes = plt.subplots(nrows=2, ncols=1,
                         sharex=True, sharey=True,
                         figsize=(5, 5))

for col, country in enumerate(countries):
    ax = axes[col]

    sns.lineplot(data=fg_prod_diffroll,
```

```
                 x=fg_prod_diffroll.index,
                 y=f'{country}',
                 ax=ax, color='black', linewidth=1.5)

    ax.plot(co2intensity_diff.index,
            co2intensity_diff['CO2 interpolated [kg/t]'],
            color='sandybrown', linestyle='--', linewidth=1.5)

    ax.plot(co2abs_diffroll.index,
            co2abs_diffroll[f'{country}'],
            color='firebrick', linewidth=1.5)

    ax.set_xlim(D1, D2)
    plt.xticks(np.arange(D1, D2+1, 5))
    ax.set_ylabel(None)

    ax.plot([D1, D2], [100, 100], '--', c='grey', linewidth=1)
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

    ax.set_title(f"{country}", loc='left', pad=10)
    style_ax(ax)

ax.set_ylim(0, 200)
plt.yticks(np.arange(0, 201, 50))

fig.subplots_adjust(hspace=0.3)

sns.despine(offset=5)

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig9_ArchGlass_Base100_60-80.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig9_ArchGlass_Base100_60-80.pdf'),
                bbox_inches='tight')
```



[202]:
```
for country in countries:
    print(f"Architectural Glass Prod in {country}:\n",
          '%.2f' % fg_prod_diffroll[f'{country}'].loc[D2])
    print(f"CO2 absolute emissions in {country}:\n",
          '%.2f' % co2abs_diffroll[f'{country}'].loc[D2])
    print("CO2 intensity:\n",
          '%.2f' % co2intensity_diff['CO2 interpolated [kg/t]'].loc[D2])
```

```
Architectural Glass Prod in Belgium:
 106.42
CO2 absolute emissions in Belgium:
 55.63
CO2 intensity:
 52.07
Architectural Glass Prod in France:
 128.11
CO2 absolute emissions in France:
 66.39
CO2 intensity:
 52.07
```

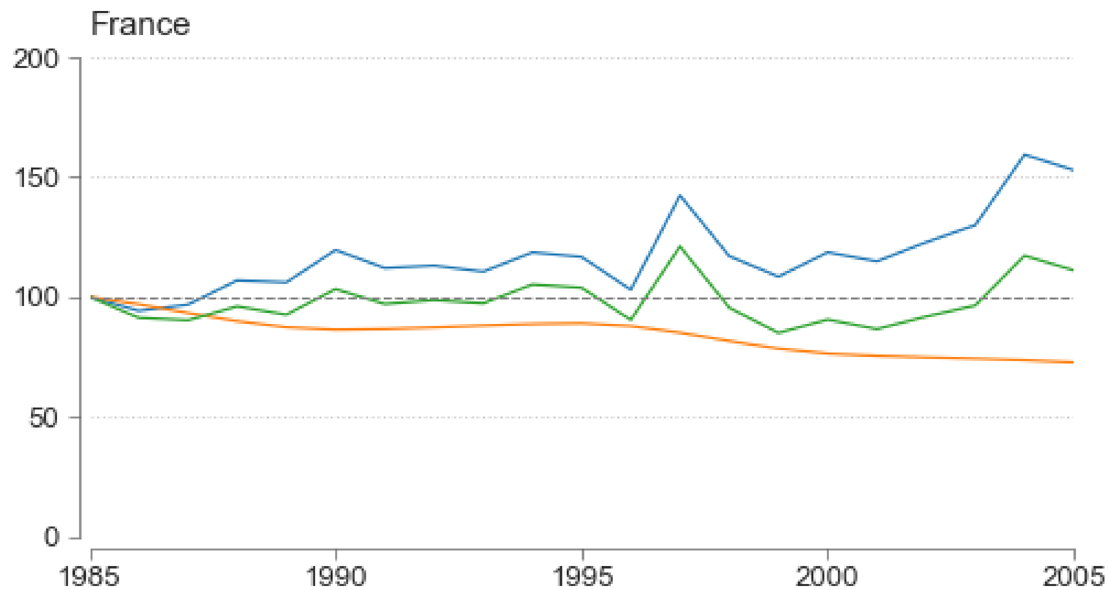
8.4 Architectural Glass Production, France and Belgium, 1985-2005

[203]:

```
# Base 100 = 1985
FGProd_diff = (100 * df_be_fr['Architectural glass production [kt]']
               / df_be_fr['Architectural glass production [kt]'].loc[D3])

co2intensity_diff = 100 * fr_co2_intensity / fr_co2_intensity.loc[D3]

co2abs_diff = (100 * df_be_fr['CO2, architectural glass [kt]']
               / df_be_fr['CO2, architectural glass [kt]'].loc[D3])
```

[204]:

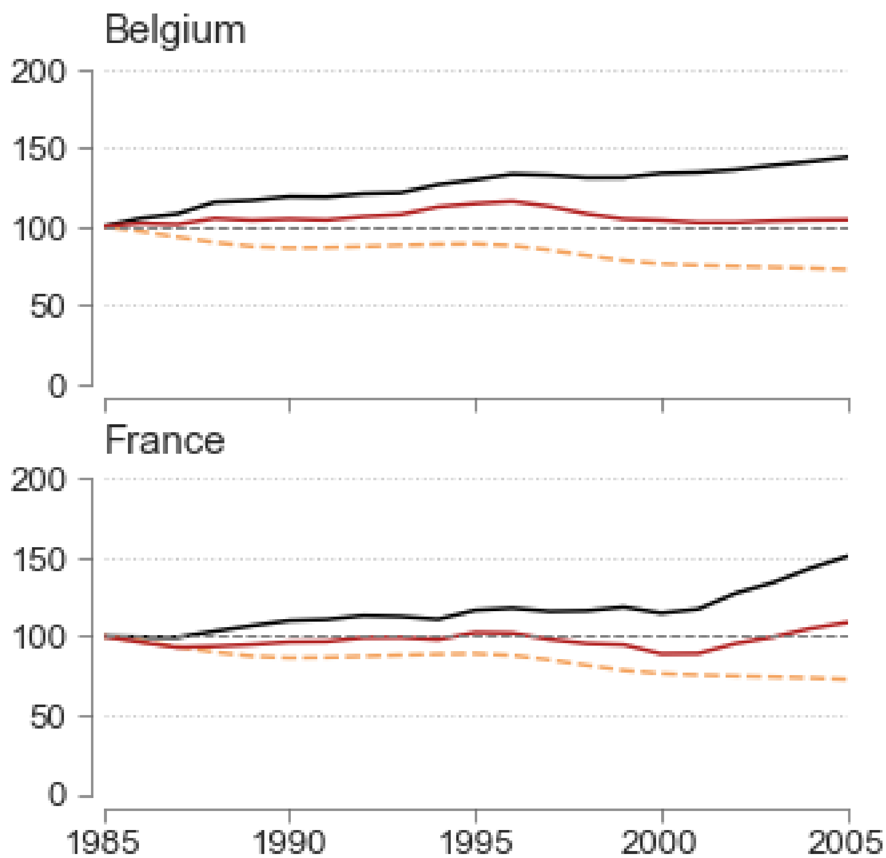
```
# Plot production, carbon intensity and CO2 absolute emissions for France:
fig, ax = plt.subplots(figsize=(8, 4))

sns.lineplot(data=FGProd_diff,
             x=FGProd_diff.index,
             y='France',
             ax=ax)

ax.plot(co2intensity_diff.index,
        co2intensity_diff['CO2 interpolated [kg/t]'])
ax.plot(co2abs_diff.index,
        co2abs_diff['France'])

ax.plot([D3, D4], [100, 100], '--', c='grey', linewidth=1)
ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

ax.set_ylim(0, 200)
plt.yticks(np.arange(0, 201, 50))
ax.set_xlim(D3, D4)
plt.xticks(np.arange(D3, D4+1, 5))

ax.set_title("France", loc='left', pad=10)
ax.set_ylabel(None)
style_ax(ax)

sns.despine(offset=5)
```



[205]:
```
# Base 100 = 1985, with a 5-year rolling
fg_prod_diffroll = (100 * df_be_fr['Architectural glass production [kt]']
                    .rolling(5, center=True).mean()
                    / df_be_fr['Architectural glass production [kt]']
                    .rolling(5, center=True).mean().loc[D3])

co2abs_diffroll = (100 * (df_be_fr['CO2, architectural glass [kt]']
                          .interpolate(method="linear", limit_area='inside')
                          .rolling(5, center=True).mean())
                   / (df_be_fr['CO2, architectural glass [kt]']
                      .interpolate(method="linear", limit_area='inside')
                      .rolling(5, center=True).mean().loc[D3]))
```

[206]:
```
# Plot production, carbon intensity, CO2 absolute emissions for France:
fig, axes = plt.subplots(nrows=2, ncols=1,
                         sharex=True, sharey=True,
                         figsize=(5, 5))

for col, country in enumerate(countries):
    ax = axes[col]

    sns.lineplot(data=fg_prod_diffroll,
                 x=fg_prod_diffroll.index,
                 y=f'{country}',
                 ax=ax, color='black', linewidth=1.5)

    ax.plot(co2intensity_diff.index,
```

```
            co2intensity_diff['CO2 interpolated [kg/t]'],
            color='sandybrown', linestyle='--', linewidth=1.5)

    ax.plot(co2abs_diffroll.index,
            co2abs_diffroll[f'{country}'],
            color='firebrick', linewidth=1.5)

    ax.set_xlim(D3, D4)
    plt.xticks(np.arange(D3, D4+1, 5))
    ax.set_ylabel(None)

    ax.plot([D3, D4], [100, 100], '--', c='grey', linewidth=1)
    ax.grid(which='major', axis='y', linestyle=':', linewidth=1)

    ax.set_title(f"{country}", loc='left', pad=10)
    style_ax(ax)

ax.set_ylim(0, 200)
plt.yticks(np.arange(0, 201, 50))

fig.subplots_adjust(hspace=0.3)

sns.despine(offset=5)

if export:
    # Save image:
    fig.savefig(os.path.join(path_img, 'Fig9_ArchGlass_Base100_85-05.png'),
                dpi=600, bbox_inches='tight')
    fig.savefig(os.path.join(path_img, 'Fig9_ArchGlass_Base100_85-05.pdf'),
                bbox_inches='tight')
```



[207]:
```
for country in countries:
    print(f"Architectural Glass Prod in {country}:\n",
          '%.2f' % fg_prod_diffroll[f'{country}'].loc[D4])
    print(f"CO2 absolute emissions in {country}:\n",
          '%.2f' % co2abs_diffroll[f'{country}'].loc[D4])
    print("CO2 intensity:\n",
          '%.2f' % co2intensity_diff['CO2 interpolated [kg/t]'].loc[D4])
```

```
Architectural Glass Prod in Belgium:
 143.94
CO2 absolute emissions in Belgium:
 104.24
CO2 intensity:
 72.63
Architectural Glass Prod in France:
 150.66
CO2 absolute emissions in France:
 108.70
CO2 intensity:
 72.63
```