sng-team-assignment

April 3, 2024

```
[ ]: #Our team decided to use as a bonus other variables, for example we took (as␣
      ↪target) Real Personal Income (and as xvars) IP: Financial Products and␣
      ↪Nonindustrial Supplies, IP: Consumer Goods
```

```
[ ]: # Installing packages
     import pandas as pd
     from numpy.linalg import solve
     import numpy as np
```

```
[ ]: # Setting directory to the csv file
     from google.colab import drive
     drive.mount('/content/drive')

     SNG_Team = '/content/drive/My Drive/comput/current.csv'

     # Loading the dataframe
     df = pd.read_csv(SNG_Team)
     df_cleaned = df.drop(index=0)
     df_cleaned.reset_index(drop=True, inplace=True)
     df_cleaned['sasdate'] = pd.to_datetime(df_cleaned['sasdate'], format='%m/%d/%Y')
     df_cleaned
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

```
[ ]:        sasdate        RPI  W875RX1  DPCERA3M086SBEA     CMRMTSPLx  \
     0   1959-01-01   2583.560   2426.0           15.188  2.766768e+05
     1   1959-02-01   2593.596   2434.8           15.346  2.787140e+05
     2   1959-03-01   2610.396   2452.7           15.491  2.777753e+05
     3   1959-04-01   2627.446   2470.0           15.435  2.833627e+05
     4   1959-05-01   2642.720   2486.4           15.622  2.853072e+05
     ..         …          …        …                …             …
     776 2023-09-01  19111.748  15741.9          116.594  1.507530e+06
     777 2023-10-01  19145.402  15784.6          116.663  1.505477e+06
     778 2023-11-01  19213.108  15859.9          117.127  1.514733e+06
     779 2023-12-01  19251.946  15899.0          117.773  1.530296e+06
     780 2024-01-01  19377.558  15948.8          117.639           NaN
```

```
         RETAILx    INDPRO   IPFPNSS   IPFINAL   IPCONGD  …  \
0     18235.77392   21.9665   23.3891   22.2688   31.7011  …
1     18369.56308   22.3966   23.7048   22.4617   31.9337  …
2     18523.05762   22.7193   23.8483   22.5719   31.9337  …
3     18534.46600   23.2032   24.1927   22.9026   32.4374  …
4     18679.66354   23.5528   24.3936   23.1231   32.5925  …
..            …         …         …        …         …   …
776  705304.00000  103.2096  101.0935  101.3665  102.1034  …
777  703528.00000  102.3722  100.5292  100.5527  101.1664  …
778  703336.00000  102.6710  100.9362  101.2159  101.8557  …
779  706180.00000  102.6715  100.8332  101.2843  101.9884  …
780  700291.00000  102.5739  100.9984  101.7258  102.6235  …

     DNDGRG3M086SBEA  DSERRG3M086SBEA  CES0600000008  CES2000000008  \
0             18.294           10.152           2.13           2.45
1             18.302           10.167           2.14           2.46
2             18.289           10.185           2.15           2.45
3             18.300           10.221           2.16           2.47
4             18.280           10.238           2.17           2.48
..                 …                …              …              …
776          120.395          123.976          29.90          34.55
777          120.040          124.228          29.97          34.67
778          119.325          124.551          30.26          34.96
779          119.193          124.917          30.45          35.01
780          118.745          125.662          30.56          35.21

     CES3000000008  UMCSENTx  DTCOLNVHFNM   DTCTHFNM     INVEST  VIXCLSx
0             2.04       NaN      6476.00   12298.00    84.2043      NaN
1             2.05       NaN      6476.00   12298.00    83.5280      NaN
2             2.07       NaN      6508.00   12349.00    81.6405      NaN
3             2.08       NaN      6620.00   12484.00    81.8099      NaN
4             2.08      95.3      6753.00   12646.00    80.7315      NaN
..               …         …            …          …          …        …
776          26.62      67.9    508808.61  913938.95  5074.6108  15.0424
777          26.65      63.8    513229.64  918210.64  5015.5456  19.0462
778          26.89      61.3    517434.30  922552.40  4999.7208  13.8563
779          27.14      69.7    522366.13  928336.14  5077.4222  12.6960
780          27.22       NaN          NaN        NaN  5105.3504  13.3453

[781 rows x 128 columns]
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785a9b6ecb20>
```

```
from matplotlib import pyplot as plt
_df_32['RPI'].plot(kind='hist', bins=20, title='RPI')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
```

```
_df_33['W875RX1'].plot(kind='hist', bins=20, title='W875RX1')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_34['DPCERA3M086SBEA'].plot(kind='hist', bins=20, title='DPCERA3M086SBEA')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_35['CMRMTSPLx'].plot(kind='hist', bins=20, title='CMRMTSPLx')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785a8dd41c30>
```

```
from matplotlib import pyplot as plt
_df_36.plot(kind='scatter', x='RPI', y='W875RX1', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_37.plot(kind='scatter', x='W875RX1', y='DPCERA3M086SBEA', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_38.plot(kind='scatter', x='DPCERA3M086SBEA', y='CMRMTSPLx', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_39.plot(kind='scatter', x='CMRMTSPLx', y='RETAILx', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785a9b5b3e50>
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['sasdate']
  ys = series['RPI']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_40.sort_values('sasdate', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('sasdate')
_ = plt.ylabel('RPI')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['sasdate']
  ys = series['W875RX1']
  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_41.sort_values('sasdate', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('sasdate')
_ = plt.ylabel('W875RX1')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['sasdate']
  ys = series['DPCERA3M086SBEA']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_42.sort_values('sasdate', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('sasdate')
_ = plt.ylabel('DPCERA3M086SBEA')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['sasdate']
  ys = series['CMRMTSPLx']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_43.sort_values('sasdate', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('sasdate')
_ = plt.ylabel('CMRMTSPLx')
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785a9b5b0fa0>
```

```
from matplotlib import pyplot as plt
_df_44['RPI'].plot(kind='line', figsize=(8, 4), title='RPI')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_45['W875RX1'].plot(kind='line', figsize=(8, 4), title='W875RX1')
```

```
plt.gca().spines[['top', 'right']].set_visible(False)

from matplotlib import pyplot as plt
_df_46['DPCERA3M086SBEA'].plot(kind='line', figsize=(8, 4),␣
 ↪title='DPCERA3M086SBEA')
plt.gca().spines[['top', 'right']].set_visible(False)

from matplotlib import pyplot as plt
_df_47['CMRMTSPLx'].plot(kind='line', figsize=(8, 4), title='CMRMTSPLx')
plt.gca().spines[['top', 'right']].set_visible(False)
```

[ ]:

```
# Extract transformation codes
transformation_codes = df.iloc[0, 1:].to_frame().reset_index()
transformation_codes.columns = ['Series', 'Transformation_Code']
```

[ ]:

```
import numpy as np

# Function to apply transformations based on the transformation code
def apply_transformation(series, code):
    if code == 1:
        # No transformation
        return series
    elif code == 2:
        # First difference
        return series.diff()
    elif code == 3:
        # Second difference
        return series.diff().diff()
    elif code == 4:
        # Log
        return np.log(series)
    elif code == 5:
        # First difference of log
        return np.log(series).diff()
    elif code == 6:
        # Second difference of log
        return np.log(series).diff().diff()
    elif code == 7:
        # Delta (x_t/x_{t-1} - 1)
        return series.pct_change()
    else:
        raise ValueError("Invalid transformation code")

# Applying the transformations to each column in df_cleaned based on␣
 ↪transformation_codes
for series_name, code in transformation_codes.values:
    df_cleaned[series_name] = apply_transformation(df_cleaned[series_name].
 ↪astype(float), float(code))
```

```
df_cleaned = df_cleaned[2:]
df_cleaned.reset_index(drop=True, inplace=True)
df_cleaned.head()
```

```
[ ]:      sasdate       RPI   W875RX1  DPCERA3M086SBEA  CMRMTSPLx   RETAILx  \
     0 1959-03-01  0.006457  0.007325         0.009404  -0.003374  0.008321
     1 1959-04-01  0.006510  0.007029        -0.003622   0.019915  0.000616
     2 1959-05-01  0.005796  0.006618         0.012043   0.006839  0.007803
     3 1959-06-01  0.003068  0.003012         0.003642  -0.000097  0.009064
     4 1959-07-01 -0.000580 -0.000762        -0.003386   0.012155 -0.000330

          INDPRO   IPFPNSS   IPFINAL   IPCONGD  …  DNDGRG3M086SBEA  \
     0  0.014306  0.006035  0.004894  0.000000  …        -0.001148
     1  0.021075  0.014338  0.014545  0.015650  …         0.001312
     2  0.014955  0.008270  0.009582  0.004770  …        -0.001695
     3  0.001141  0.007034  0.007128 -0.004767  …         0.003334
     4 -0.024240  0.001168  0.008249  0.013054  …        -0.001204

        DSERRG3M086SBEA  CES0600000008  CES2000000008  CES3000000008  UMCSENTx  \
     0         0.000292      -0.000022      -0.008147       0.004819       NaN
     1         0.001760      -0.000022       0.012203      -0.004890       NaN
     2        -0.001867      -0.000021      -0.004090      -0.004819       NaN
     3         0.001946      -0.004619       0.003992       0.004796       NaN
     4        -0.000013       0.000000      -0.004040      -0.004796       NaN

        DTCOLNVHFNM  DTCTHFNM    INVEST  VIXCLSx
     0     0.004929  0.004138 -0.014792      NaN
     1     0.012134  0.006734  0.024929      NaN
     2     0.002828  0.002020 -0.015342      NaN
     3     0.009726  0.009007 -0.012252      NaN
     4    -0.004631 -0.001000  0.029341      NaN

     [5 rows x 128 columns]
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785a8ddd8d30>
```

```
from matplotlib import pyplot as plt
_df_16['RPI'].plot(kind='hist', bins=20, title='RPI')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_17['W875RX1'].plot(kind='hist', bins=20, title='W875RX1')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_18['DPCERA3M086SBEA'].plot(kind='hist', bins=20, title='DPCERA3M086SBEA')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_19['CMRMTSPLx'].plot(kind='hist', bins=20, title='CMRMTSPLx')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785a9e0c6b30>
```

```
from matplotlib import pyplot as plt
_df_20.plot(kind='scatter', x='RPI', y='W875RX1', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_21.plot(kind='scatter', x='W875RX1', y='DPCERA3M086SBEA', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_22.plot(kind='scatter', x='DPCERA3M086SBEA', y='CMRMTSPLx', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_23.plot(kind='scatter', x='CMRMTSPLx', y='RETAILx', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785a8de093f0>
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['sasdate']
  ys = series['RPI']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_24.sort_values('sasdate', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('sasdate')
_ = plt.ylabel('RPI')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['sasdate']
  ys = series['W875RX1']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_25.sort_values('sasdate', ascending=True)
_plot_series(df_sorted, '')
```

```
sns.despine(fig=fig, ax=ax)
plt.xlabel('sasdate')
_ = plt.ylabel('W875RX1')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['sasdate']
  ys = series['DPCERA3M086SBEA']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_26.sort_values('sasdate', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('sasdate')
_ = plt.ylabel('DPCERA3M086SBEA')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['sasdate']
  ys = series['CMRMTSPLx']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_27.sort_values('sasdate', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('sasdate')
_ = plt.ylabel('CMRMTSPLx')
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785a8de09390>
```

```
from matplotlib import pyplot as plt
_df_28['RPI'].plot(kind='line', figsize=(8, 4), title='RPI')
plt.gca().spines[['top', 'right']].set_visible(False)
```

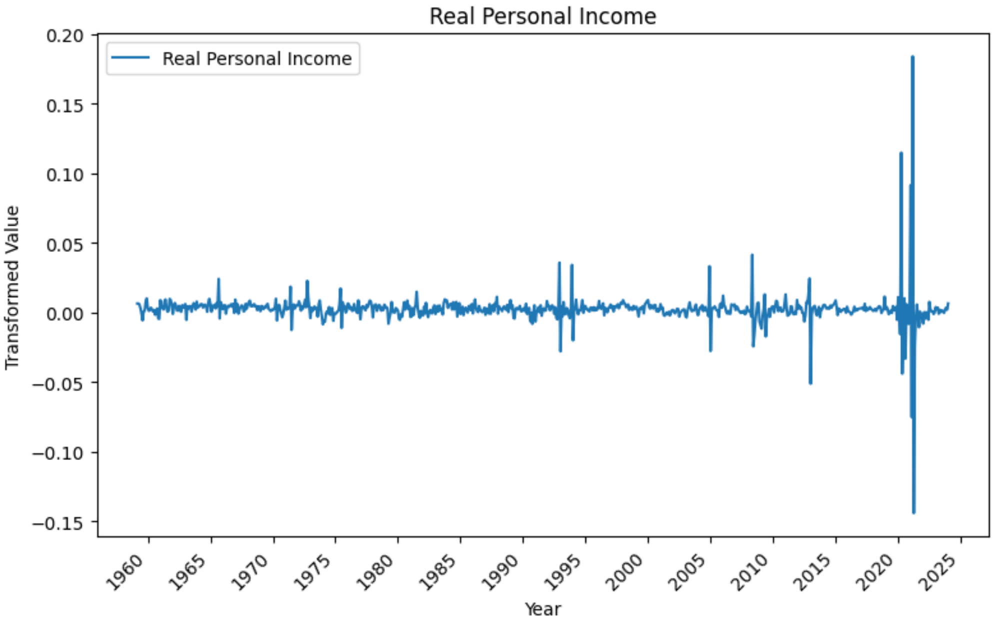
```
from matplotlib import pyplot as plt
_df_29['W875RX1'].plot(kind='line', figsize=(8, 4), title='W875RX1')
plt.gca().spines[['top', 'right']].set_visible(False)
```

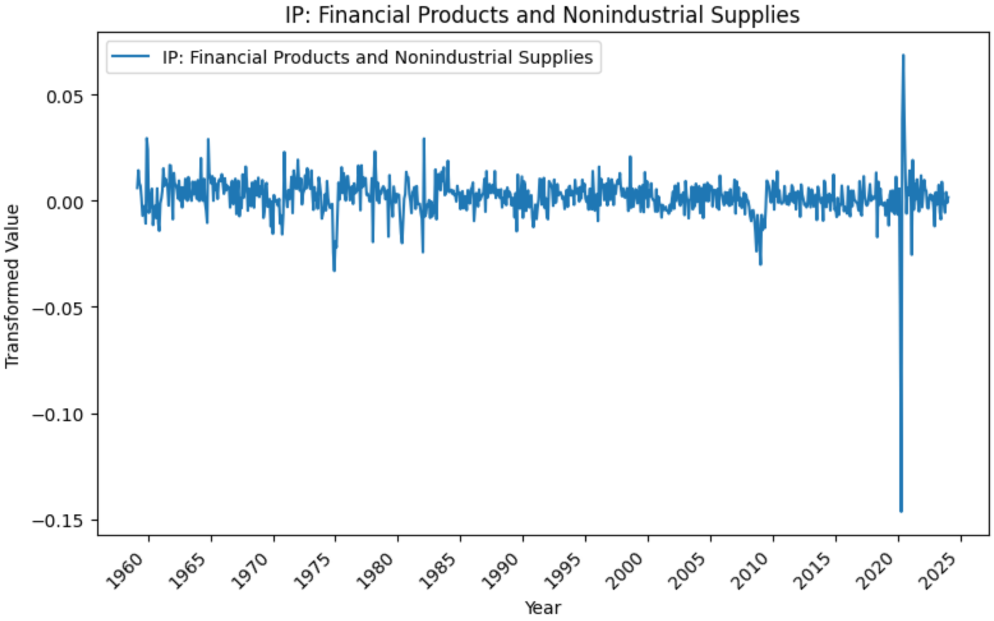
```
from matplotlib import pyplot as plt
_df_30['DPCERA3M086SBEA'].plot(kind='line', figsize=(8, 4),␣
 ↪title='DPCERA3M086SBEA')
plt.gca().spines[['top', 'right']].set_visible(False)
```

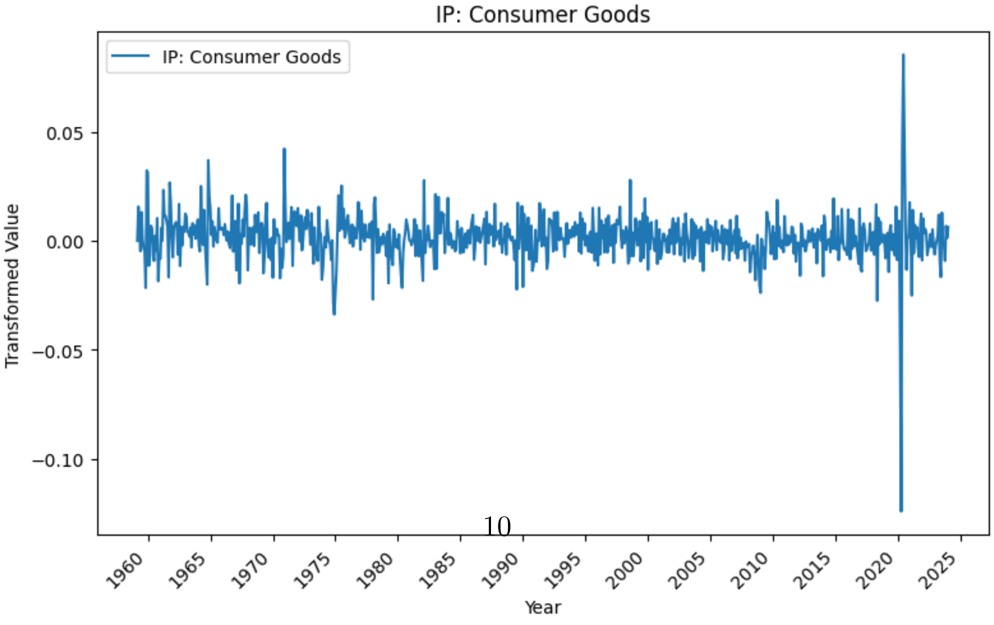
```
from matplotlib import pyplot as plt
_df_31['CMRMTSPLx'].plot(kind='line', figsize=(8, 4), title='CMRMTSPLx')
plt.gca().spines[['top', 'right']].set_visible(False)
```

[ ]:

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

series_to_plot = ['RPI', 'IPFPNSS', 'IPCONGD']
series_names = ['Real Personal Income',
                'IP: Financial Products and Nonindustrial Supplies',
                'IP: Consumer Goods']



# Create a figure and a grid of subplots
fig, axs = plt.subplots(len(series_to_plot), 1, figsize=(8, 15))

# Iterate over the selected series and plot each one
for ax, series_name, plot_title in zip(axs, series_to_plot, series_names):
    if series_name in df_cleaned.columns:
        dates = pd.to_datetime(df_cleaned['sasdate'], format='%m/%d/%Y')
        ax.plot(dates, df_cleaned[series_name], label=plot_title)
        ax.xaxis.set_major_locator(mdates.YearLocator(base=5))
        ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        ax.set_title(plot_title)
        ax.set_xlabel('Year')
        ax.set_ylabel('Transformed Value')
        ax.legend(loc='upper left')
        plt.setp(ax.xaxis.get_majorticklabels(), rotation=45, ha='right')
    else:
        ax.set_visible(False) # Hide plots for which the data is not available

plt.tight_layout()
plt.show()
```


Real Personal Income


IP: Financial Products and Nonindustrial Supplies


IP: Consumer Goods

```
[ ]: Yraw = df_cleaned['RPI']
     Xraw = df_cleaned[['IPFPNSS', 'IPCONGD']]

     num_lags = 4 ## this is p
     num_leads = 1 ## this is h
     X = pd.DataFrame()
     ## Add the lagged values of Y
     col = 'RPI'
     for lag in range(0,num_lags+1):
             # Shift each column in the DataFrame and name it with a lag suffix
             X[f'{col}_lag{lag}'] = Yraw.shift(lag)

     for col in Xraw.columns:
         for lag in range(0,num_lags+1):
             # Shift each column in the DataFrame and name it with a lag suffix
             X[f'{col}_lag{lag}'] = Xraw[col].shift(lag)
     ## Add a column on ones (for the intercept)
     X.insert(0, 'Ones', np.ones(len(X)))


     ## X is now a DataFrame
     X.head()
```

```
[ ]:    Ones  RPI_lag0  RPI_lag1  RPI_lag2  RPI_lag3  RPI_lag4  IPFPNSS_lag0  \
     0   1.0  0.006457       NaN       NaN       NaN       NaN      0.006035
     1   1.0  0.006510  0.006457       NaN       NaN       NaN      0.014338
     2   1.0  0.005796  0.006510  0.006457       NaN       NaN      0.008270
     3   1.0  0.003068  0.005796  0.006510  0.006457       NaN      0.007034
     4   1.0 -0.000580  0.003068  0.005796  0.006510  0.006457      0.001168

        IPFPNSS_lag1  IPFPNSS_lag2  IPFPNSS_lag3  IPFPNSS_lag4  IPCONGD_lag0  \
     0           NaN           NaN           NaN           NaN      0.000000
     1      0.006035           NaN           NaN           NaN      0.015650
     2      0.014338      0.006035           NaN           NaN      0.004770
     3      0.008270      0.014338      0.006035           NaN     -0.004767
     4      0.007034      0.008270      0.014338      0.006035      0.013054

        IPCONGD_lag1  IPCONGD_lag2  IPCONGD_lag3  IPCONGD_lag4
     0           NaN           NaN           NaN           NaN
     1      0.000000           NaN           NaN           NaN
     2      0.015650       0.00000           NaN           NaN
     3      0.004770       0.01565       0.00000           NaN
     4     -0.004767       0.00477       0.01565           0.0
```

[ ]:

```
y = Yraw.shift(-num_leads)
y
```

[ ]:

```
0      0.006510
1      0.005796
2      0.003068
3     -0.000580
4     -0.005653
         …
774    0.001759
775    0.003530
776    0.002019
777    0.006503
778         NaN
Name: RPI, Length: 779, dtype: float64
```

[ ]:

```
## Save last row of X (converted to numpy)
X_T = X.iloc[-1:].values
## Subset getting only rows of X and y from p+1 to h-1
## and convert to numpy array
y = y.iloc[num_lags:-num_leads].values
X = X.iloc[num_lags:-num_leads].values



X_T
```

[ ]:

```
array([[ 1.00000000e+00,  6.50344580e-03,  2.01939211e-03,
         3.53017207e-03,  1.75935785e-03, -3.00450660e-04,
         1.63700864e-03, -1.02096759e-03,  4.04040151e-03,
        -5.59759862e-03, -1.84411458e-03,  6.20787002e-03,
         1.30197557e-03,  6.79041985e-03, -9.21933939e-03,
        -1.94613030e-03]])
```

[ ]:

```
from numpy.linalg import solve
# Solving for the OLS estimator beta: (X'X)^{-1} X'Y
beta_ols = solve(X.T @ X, X.T @ y)

## Produce the One step ahead forecast
## % change month-to-month INDPRO
forecast = X_T@beta_ols*100
forecast
```

[ ]:

```
array([0.15461092])
```

[ ]:

```
def calculate_forecast(df_cleaned,
                       p = 4,
                       H = [1,4,8],
```

```python
                       end_date = '12/1/2021',
                       target = 'RPI',
                       xvars = ['IPFPNSS', 'IPCONGD']):
    ## Subset df_cleaned to use only data up to end_date
    rt_df = df_cleaned[df_cleaned['sasdate'] <= pd.Timestamp(end_date)]
    ## Get the actual values of the target at different steps ahead
    Y_actual = []
    for h in H:
        os = pd.Timestamp(end_date) + pd.DateOffset(months=h)
        Y_actual.append(df_cleaned[df_cleaned['sasdate'] == os][target]*100)
        ## Now Y contains the true values at T+H (multiplying * 100)

    Yraw = rt_df[target]
    Xraw = rt_df[xvars]

    X = pd.DataFrame()
    ## Add the lagged values of Y
    for lag in range(0,p):
        # Shift each column in the DataFrame and name it with a lag suffix
        X[f'{target}_lag{lag}'] = Yraw.shift(lag)

    for col in Xraw.columns:
        for lag in range(0,p):
            X[f'{col}_lag{lag}'] = Xraw[col].shift(lag)

    ## Add a column on ones (for the intercept)
    X.insert(0, 'Ones', np.ones(len(X)))

    ## Save the last row of X (converted to a `numpy` array)
    X_T = X.iloc[-1:].values

    ## While the X will be the same, Y needs to be leaded differently
    Yhat = []
    for h in H:
        y_h = Yraw.shift(-h)
        ## Subset getting only rows of X and y from p+1 to h-1
        y = y_h.iloc[p:-h].values
        X_ = X.iloc[p:-h].values
        # Solving for the OLS estimator beta: (X'X)^{-1} X'Y
        beta_ols = solve(X_.T @ X_, X_.T @ y)
        ## Produce the One step ahead forecast
        ## % change month-to-month RPI
        Yhat.append(X_T@beta_ols*100)

    ## Now calculate the forecasting error and return

    return np.array(Y_actual) - np.array(Yhat)
```

[ ]:

```
t0 = pd.Timestamp('12/1/2021')
e = []
T = []
for j in range(0, 10):
    t0 = t0 + pd.DateOffset(months=1)
    print(f'Using data up to {t0}')
    ehat = calculate_forecast(df_cleaned, p = 4, H = [1,4,8], end_date = t0)
    e.append(ehat.flatten())
    T.append(t0)

## Create a pandas DataFrame from the list
edf = pd.DataFrame(e)
## Calculate the RMSFE, that is, the square root of the MSFE
np.sqrt(edf.apply(np.square).mean())
```

```
Using data up to 2022-01-01 00:00:00
Using data up to 2022-02-01 00:00:00
Using data up to 2022-03-01 00:00:00
Using data up to 2022-04-01 00:00:00
Using data up to 2022-05-01 00:00:00
Using data up to 2022-06-01 00:00:00
Using data up to 2022-07-01 00:00:00
Using data up to 2022-08-01 00:00:00
Using data up to 2022-09-01 00:00:00
Using data up to 2022-10-01 00:00:00
```

[ ]:

```
0    0.708419
1    0.439215
2    0.426919
dtype: float64
```