1 Description:

This project work mainly focuses on analysing the Consumer Price Index data of whole India from January 2013 to March 2023. In the following the work, the aim is to achieve a good insight about the data and to build a model to forecast the Index number in upcoming future. The following dataset contains various columns such regarding the changing CPI values of all over the year. Those columns are Cereals and products, Meat and Fish, Egg, Milk and products and many more. Also the following data contains Consumer Price Index (CPI) values for Rural, Urban and Rural and Urban combined sector.

The dataset is collected from here

2 Problem Formulation:

1. Collect the data and pre-process the data before analysis.
2. Dividing the data into different sectors in terms of Rural, Urban, and Rural and Urban Combined.
3. Developing an insight about the Consumer Price Index Data over different sectors.
4. Compare different products under a single sector in terms of their Consumer Price Index Value
5. Finally, predict the CPI numbers by building the best time series model.

`Software Used: Python and R-Studio`

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

sns.set_theme(style="darkgrid",palette='deep', font='sans-serif')
import warnings
warnings.filterwarnings(action= 'ignore')
```

[2]:
```python
df = pd.read_csv('/content/drive/MyDrive/EDA_DA/
↪All_India_Index_july2019_20Aug2020_dec20_0.csv',dtype={'Year':str})
```

[3]:
```python
# first 2 rows of the data
df.head(2)
```

[3]:
```
  Sector  Year    Month  Cereals and products  Meat and fish    Egg  \
0  Rural  2013  January                 107.5          106.3  108.1
1  Urban  2013  January                 110.5          109.1  113.0

   Milk and products  Oils and fats  Fruits  Vegetables  ...  Housing  \
0              104.9          106.1   103.9       101.9  ...      NaN
1              103.6          103.4   102.3       102.9  ...    100.3

   Fuel and light  Household goods and services  Health  \
0           105.5                         104.8   104.0
1           105.4                         104.8   104.1

   Transport and communication  Recreation and amusement  Education  \
0                        103.3                     103.4      103.8
1                        103.2                     102.9      103.5

   Personal care and effects  Miscellaneous  General index
```

```
0                       104.7           104.0           105.1
1                       104.3           103.7           104.0

[2 rows x 30 columns]
```

[4]:
```
# describtive statistics
df.describe().T
```

[4]:
```
                                     count        mean        std    min  \
Cereals and products                 363.0  136.070523  14.328877  107.5
Meat and fish                        360.0  155.327222  32.819364  106.3
Egg                                  363.0  140.285124  22.347112  102.7
Milk and products                    363.0  139.686226  17.812480  103.6
Oils and fats                        363.0  131.473278  29.759782  101.1
Fruits                               363.0  140.286777  16.959676  102.3
Vegetables                           363.0  155.646556  28.054053  101.4
Pulses and products                  363.0  140.880716  23.747592  103.5
Sugar and Confectionery              363.0  110.737466   8.970903   85.3
Spices                               363.0  143.365289  25.558353  101.8
Non-alcoholic beverages              363.0  133.459229  18.982685  104.8
Prepared meals, snacks, sweets etc.  360.0  148.338611  22.473193  106.7
Food and beverages                   363.0  141.905510  19.279498  105.5
Pan, tobacco and intoxicants         360.0  154.618056  29.240688  105.1
Clothing                             360.0  141.934722  20.652368  105.9
Footwear                             360.0  135.271667  19.096613  105.0
Clothing and footwear                360.0  140.949167  20.388539  105.8
Fuel and light                       363.0  135.752617  21.633542  105.4
Household goods and services         360.0  136.052778  18.836591  104.8
Health                               363.0  137.750138  23.124888  104.0
Transport and communication          360.0  126.543056  18.685162  103.2
Recreation and amusement             360.0  133.270833  19.706305  102.9
Education                            360.0  140.532778  20.862561  103.5
Personal care and effects            360.0  132.495000  22.360294  102.1
Miscellaneous                        360.0  133.551944  20.283842  103.7
General index                        360.0  138.914444  19.907439  104.0

                                         25%     50%      75%    max
Cereals and products                 124.050  135.60  146.000  174.7
Meat and fish                        129.800  143.90  190.150  223.4
Egg                                  122.000  134.90  157.150  197.0
Milk and products                    128.250  140.80  153.750  177.9
Oils and fats                        110.400  119.90  138.200  209.9
Fruits                               130.250  140.20  151.800  179.5
Vegetables                           134.800  153.20  171.200  245.3
Pulses and products                  119.850  138.30  164.100  191.6
Sugar and Confectionery              103.400  113.20  118.150  123.9
Spices                               127.100  139.50  160.600  212.1
Non-alcoholic beverages              119.600  128.80  142.150  177.6
Prepared meals, snacks, sweets etc.  131.175  148.95  162.975  196.6
Food and beverages                   128.700  138.40  156.950  183.3
Pan, tobacco and intoxicants         130.325  154.65  184.450  202.7
Clothing                             125.400  142.40  153.475  190.0
Footwear                             120.575  132.70  146.100  187.0
Clothing and footwear                124.850  140.80  152.300  189.6
```

```
Fuel and light                      116.300  131.20  148.150  183.2
Household goods and services        120.850  134.80  150.100  178.6
Health                              118.350  133.10  156.250  186.6
Transport and communication         111.675  119.40  139.800  169.0
Recreation and amusement            117.125  129.55  148.800  172.8
Education                           123.475  139.30  159.250  178.5
Personal care and effects           112.375  127.05  154.975  181.5
Miscellaneous                       116.450  129.00  149.475  177.9
General index                       123.150  136.45  155.250  178.0
```

[5]:
```
df.columns
```

[5]:
```
Index(['Sector', 'Year', 'Month', 'Cereals and products', 'Meat and fish',
       'Egg', 'Milk and products', 'Oils and fats', 'Fruits', 'Vegetables',
       'Pulses and products', 'Sugar and Confectionery', 'Spices',
       'Non-alcoholic beverages', 'Prepared meals, snacks, sweets etc.',
       'Food and beverages', 'Pan, tobacco and intoxicants', 'Clothing',
       'Footwear', 'Clothing and footwear', 'Housing', 'Fuel and light',
       'Household goods and services', 'Health', 'Transport and communication',
       'Recreation and amusement', 'Education', 'Personal care and effects',
       'Miscellaneous', 'General index'],
      dtype='object')
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 30 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   Sector                               366 non-null    object
 1   Year                                 366 non-null    object
 2   Month                                366 non-null    object
 3   Cereals and products                 363 non-null    float64
 4   Meat and fish                        360 non-null    float64
 5   Egg                                  363 non-null    float64
 6   Milk and products                    363 non-null    float64
 7   Oils and fats                        363 non-null    float64
 8   Fruits                               363 non-null    float64
 9   Vegetables                           363 non-null    float64
 10  Pulses and products                  363 non-null    float64
 11  Sugar and Confectionery              363 non-null    float64
 12  Spices                               363 non-null    float64
 13  Non-alcoholic beverages              363 non-null    float64
 14  Prepared meals, snacks, sweets etc.  360 non-null    float64
 15  Food and beverages                   363 non-null    float64
 16  Pan, tobacco and intoxicants         360 non-null    float64
 17  Clothing                             360 non-null    float64
 18  Footwear                             360 non-null    float64
 19  Clothing and footwear                360 non-null    float64
 20  Housing                              244 non-null    object
 21  Fuel and light                       363 non-null    float64
 22  Household goods and services         360 non-null    float64
 23  Health                               363 non-null    float64
 24  Transport and communication          360 non-null    float64
```

```
 25  Recreation and amusement             360 non-null    float64
 26  Education                            360 non-null    float64
 27  Personal care and effects            360 non-null    float64
 28  Miscellaneous                        360 non-null    float64
 29  General index                        360 non-null    float64
dtypes: float64(26), object(4)
memory usage: 85.9+ KB
```

Dividing the dataset in terms of sectors.

[7]:
```
df_r = df[df.Sector == 'Rural']
df_u = df[df.Sector == 'Urban']
df_ru = df[df.Sector == 'Rural+Urban']
```

[8]:
```
print('The shape of rural sector dataframe is {} and urban sector dataframe is {} and␣
↪rural + urban dataframe is {} '.format(df_r.shape, df_u.shape, df_ru.shape))
```

```
The shape of rural sector dataframe is (122, 30) and urban sector dataframe is
(122, 30) and rural + urban dataframe is (122, 30)
```

3 Rural

[9]:
```
df_r.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 122 entries, 0 to 363
Data columns (total 30 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   Sector                               122 non-null    object
 1   Year                                 122 non-null    object
 2   Month                                122 non-null    object
 3   Cereals and products                 121 non-null    float64
 4   Meat and fish                        120 non-null    float64
 5   Egg                                  121 non-null    float64
 6   Milk and products                    121 non-null    float64
 7   Oils and fats                        121 non-null    float64
 8   Fruits                               121 non-null    float64
 9   Vegetables                           121 non-null    float64
 10  Pulses and products                  121 non-null    float64
 11  Sugar and Confectionery              121 non-null    float64
 12  Spices                               121 non-null    float64
 13  Non-alcoholic beverages              121 non-null    float64
 14  Prepared meals, snacks, sweets etc.  120 non-null    float64
 15  Food and beverages                   121 non-null    float64
 16  Pan, tobacco and intoxicants         120 non-null    float64
 17  Clothing                             120 non-null    float64
 18  Footwear                             120 non-null    float64
 19  Clothing and footwear                120 non-null    float64
 20  Housing                              2 non-null      object
 21  Fuel and light                       121 non-null    float64
 22  Household goods and services         120 non-null    float64
 23  Health                               121 non-null    float64
 24  Transport and communication          120 non-null    float64
 25  Recreation and amusement             120 non-null    float64
```

```
 26  Education                             120 non-null    float64
 27  Personal care and effects             120 non-null    float64
 28  Miscellaneous                         120 non-null    float64
 29  General index                         120 non-null    float64
dtypes: float64(26), object(4)
memory usage: 29.5+ KB
```

Formatting the Date column into a timestamp format

```
[10]: df_r['Date'] = pd.to_datetime(df_r['Year'] + ' ' + df_r['Month'])
      # shift column 'Name' to first position
      first_column = df_r.pop('Date')

      # insert column using insert(position,column_name,
      # first_column) function
      df_r.insert(0, 'Date', first_column)
```

```
[11]: df_r.drop(['Year','Month'],axis = 1, inplace = True)
      df_r.reset_index(drop=True,inplace=True)
```

Removing the non significant columns

```
[12]: df_r.drop(['Housing'],axis = 1, inplace = True)
```

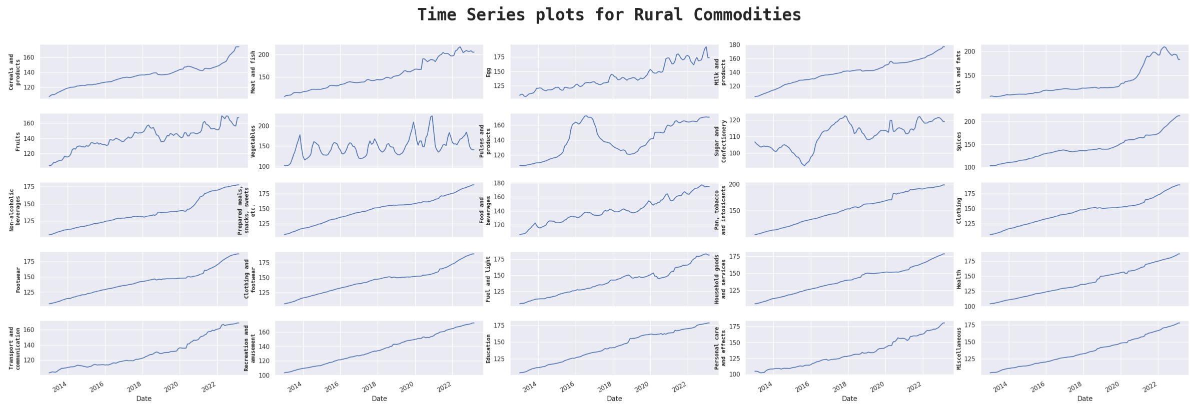
Replacing the null values with forward fill method

```
[13]: df_r = df_r.fillna(method='ffill')
```

4 Visualisation of Rural Data

```
[14]: import textwrap
```

```
[15]: # Plot the responses for different events and regions
      cols = df_r.columns[2:27]
      length = len(cols)
      wrapped_labels = [textwrap.fill(label, 15) for label in cols]
      # Creating subplot axes
      fig, axes = plt.subplots(5,5,figsize=(30,12),sharex = True)
      fig.tight_layout(h_pad = 2)
      i = 0
      for name, ax in zip(cols, axes.flatten()):
        # Adjust the width parameter as needed
        sns.lineplot(y=name, x= "Date", data=df_r, ax=ax)
        ax.set_ylabel(wrapped_labels[i],fontproperties={'family':'monospace', 'size':
       ↪10,'weight':'demibold'})
        i += 1

      plt.suptitle('Time Series plots for Rural Commodities',size = 30,␣
       ↪fontproperties={'family':'monospace', 'weight':'bold'})
      fig.subplots_adjust(top=0.9)

      plt.gcf().autofmt_xdate()
      plt.show()
```



5 Urban

[16]:
```
df_u.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 122 entries, 1 to 364
Data columns (total 30 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Sector                            122 non-null    object
 1   Year                              122 non-null    object
 2   Month                             122 non-null    object
 3   Cereals and products              121 non-null    float64
 4   Meat and fish                     120 non-null    float64
 5   Egg                               121 non-null    float64
 6   Milk and products                 121 non-null    float64
 7   Oils and fats                     121 non-null    float64
 8   Fruits                            121 non-null    float64
 9   Vegetables                        121 non-null    float64
 10  Pulses and products               121 non-null    float64
 11  Sugar and Confectionery           121 non-null    float64
 12  Spices                            121 non-null    float64
 13  Non-alcoholic beverages           121 non-null    float64
 14  Prepared meals, snacks, sweets etc.  120 non-null    float64
 15  Food and beverages                121 non-null    float64
 16  Pan, tobacco and intoxicants      120 non-null    float64
 17  Clothing                          120 non-null    float64
 18  Footwear                          120 non-null    float64
 19  Clothing and footwear             120 non-null    float64
 20  Housing                           121 non-null    object
 21  Fuel and light                    121 non-null    float64
 22  Household goods and services     120 non-null    float64
 23  Health                            121 non-null    float64
 24  Transport and communication       120 non-null    float64
 25  Recreation and amusement          120 non-null    float64
 26  Education                         120 non-null    float64
 27  Personal care and effects         120 non-null    float64
 28  Miscellaneous                     120 non-null    float64
 29  General index                     120 non-null    float64
```

```
dtypes: float64(26), object(4)
memory usage: 29.5+ KB
```

Formatting the Date column to timestamp format

[17]:
```
df_u['Date'] = pd.to_datetime(df_u['Year'] + ' ' + df_u['Month'])
# shift column 'Name' to first position
first_column = df_u.pop('Date')

# insert column using insert(position,column_name,
# first_column) function
df_u.insert(0, 'Date', first_column)
```

[18]:
```
df_u.drop(['Year','Month'],axis = 1, inplace = True)
df_u.reset_index(drop=True,inplace=True)
```

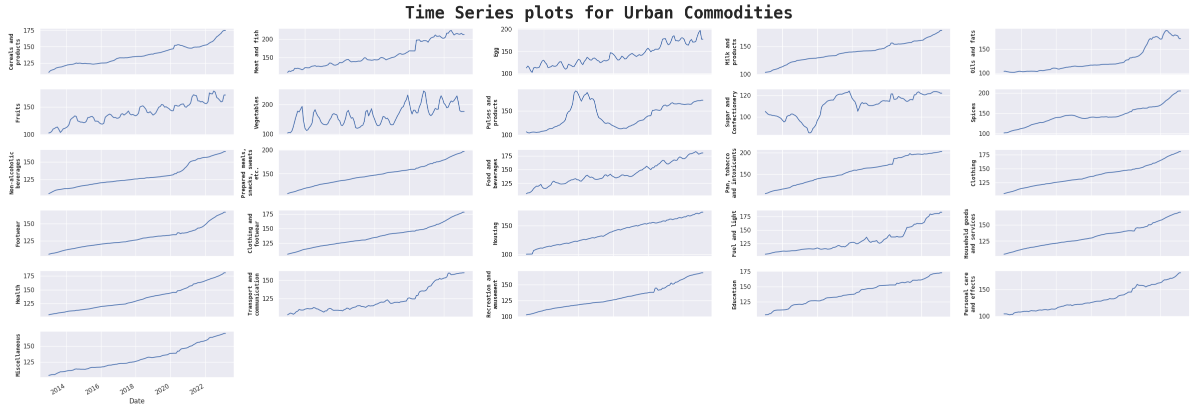
[19]:
```
df_u['Housing'] = df_u.Housing.astype(float)
```

Removing Null Values

[20]:
```
df_u = df_u.fillna(method='ffill')
```

6 Visualisation of Urban Data

[21]:
```
# Plot the responses for different events and regions
cols = df_u.columns[2:28]
length = len(cols)
wrapped_labels = [textwrap.fill(label, 15) for label in cols]
# Creating subplot axes
fig, axes = plt.subplots(6,5,figsize=(30,12),sharex = True)
i = 0
for name, ax in zip(cols, axes.flatten()):
  # Adjust the width parameter as needed
  sns.lineplot(y=name, x= "Date", data=df_u, ax=ax)
  ax.set_ylabel(wrapped_labels[i],fontproperties={'family':'monospace', 'size':
↪10,'weight':'demibold'})
  i += 1

plt.suptitle('Time Series plots for Urban Commodities',size = 30,␣
↪fontproperties={'family':'monospace', 'weight':'bold'})
fig.subplots_adjust(top=0.9)

for i in range(26, len(axes.flatten())):
  fig.delaxes(axes.flatten()[i])
fig.tight_layout(h_pad = 2)

plt.gcf().autofmt_xdate()
plt.show()
```



7 Rural and Urban (Combined)

[22]:
```
df_ru.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 122 entries, 2 to 365
Data columns (total 30 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Sector                           122 non-null    object
 1   Year                             122 non-null    object
 2   Month                            122 non-null    object
 3   Cereals and products             121 non-null    float64
 4   Meat and fish                    120 non-null    float64
 5   Egg                              121 non-null    float64
 6   Milk and products                121 non-null    float64
 7   Oils and fats                    121 non-null    float64
 8   Fruits                           121 non-null    float64
 9   Vegetables                       121 non-null    float64
 10  Pulses and products              121 non-null    float64
 11  Sugar and Confectionery          121 non-null    float64
 12  Spices                           121 non-null    float64
 13  Non-alcoholic beverages          121 non-null    float64
 14  Prepared meals, snacks, sweets etc.  120 non-null    float64
 15  Food and beverages               121 non-null    float64
 16  Pan, tobacco and intoxicants     120 non-null    float64
 17  Clothing                         120 non-null    float64
 18  Footwear                         120 non-null    float64
 19  Clothing and footwear            120 non-null    float64
 20  Housing                          121 non-null    object
 21  Fuel and light                   121 non-null    float64
 22  Household goods and services     120 non-null    float64
 23  Health                           121 non-null    float64
 24  Transport and communication      120 non-null    float64
 25  Recreation and amusement         120 non-null    float64
 26  Education                        120 non-null    float64
 27  Personal care and effects        120 non-null    float64
 28  Miscellaneous                    120 non-null    float64
 29  General index                    120 non-null    float64
```

```
dtypes: float64(26), object(4)
memory usage: 29.5+ KB
```

Data Cleaning:

Formatting the misspelled words from the dataset.

[23]:
```
df_ru[df_ru['Month']=='Marcrh']
```

[23]:
```
         Sector  Year   Month  Cereals and products  Meat and fish    Egg  \
44  Rural+Urban  2014  Marcrh                 120.7          119.3  121.0

    Milk and products  Oils and fats  Fruits  Vegetables  ...  Housing  \
44              116.1          106.9   118.7       116.3  ...    113.2

    Fuel and light  Household goods and services  Health  \
44           112.5                         113.2   111.2

    Transport and communication  Recreation and amusement  Education  \
44                        111.4                     110.6      112.0

    Personal care and effects  Miscellaneous  General index
44                      109.0          111.3          114.2

[1 rows x 30 columns]
```

[24]:
```
df_ru[df_ru['Month']=='Marcrh'].index
```

[24]:
```
Int64Index([44], dtype='int64')
```

[25]:
```
df_ru.at[44,'Month']='March'
```

[26]:
```
df_ru[df_ru['Month']=='Marcrh']
```

[26]:
```
Empty DataFrame
Columns: [Sector, Year, Month, Cereals and products, Meat and fish, Egg, Milk
and products, Oils and fats, Fruits, Vegetables, Pulses and products, Sugar and
Confectionery, Spices, Non-alcoholic beverages, Prepared meals, snacks, sweets
etc., Food and beverages, Pan, tobacco and intoxicants, Clothing, Footwear,
Clothing and footwear, Housing, Fuel and light, Household goods and services,
Health, Transport and communication, Recreation and amusement, Education,
Personal care and effects, Miscellaneous, General index]
Index: []

[0 rows x 30 columns]
```

Formatting the Date column to timestamp format

[27]:
```
df_ru['Date'] = pd.to_datetime(df_ru['Year'] + ' ' + df_ru['Month'])
# shift column 'Name' to first position
first_column = df_ru.pop('Date')

# insert column using insert(position,column_name,
# first_column) function
df_ru.insert(0, 'Date', first_column)
```

[28]:
```
df_ru.drop(['Year','Month'],axis = 1, inplace = True)
df_ru.reset_index(drop=True,inplace=True)
```

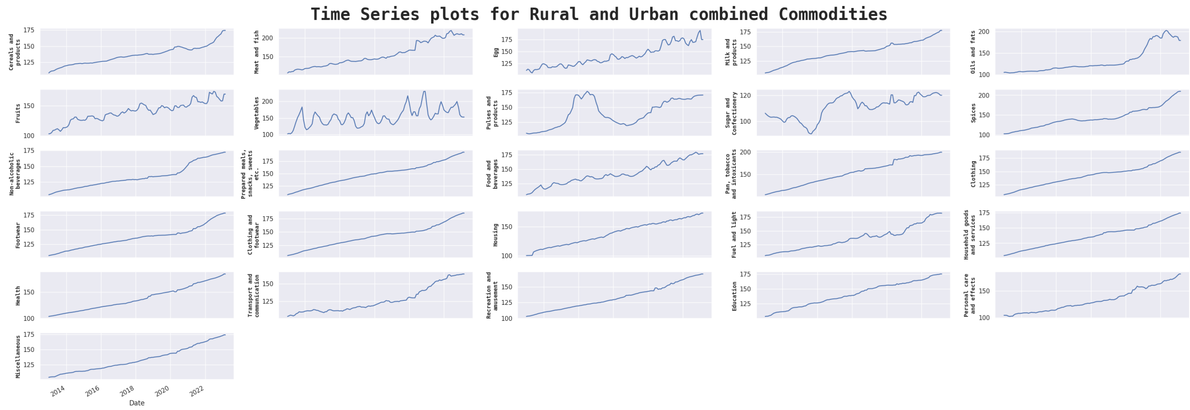
[29]:
```
df_ru['Housing'] = df_ru.Housing.astype(float)
```

Replacing Null Values with forward fill method

[30]:
```
df_ru = df_ru.fillna(method='ffill')
```

8 Visualisation of Rural and Urban Data Combined

[31]:
```
# Plot the responses for different events and regions
cols = df_ru.columns[2:28]
length = len(cols)
wrapped_labels = [textwrap.fill(label, 15) for label in cols]
# Creating subplot axes
fig, axes = plt.subplots(6,5,figsize=(30,12),sharex = True)
i = 0
for name, ax in zip(cols, axes.flatten()):
    # Adjust the width parameter as needed
  sns.lineplot(y=name, x= "Date", data=df_ru, ax=ax, errorbar=('se',2))
  ax.set_ylabel(wrapped_labels[i],fontproperties={'family':'monospace', 'size':
↪10,'weight':'demibold'})
  i += 1

plt.suptitle('Time Series plots for Rural and Urban combined Commodities',size = 30,␣
↪fontproperties={'family':'monospace', 'weight':'bold'})
fig.subplots_adjust(top=0.9)
for i in range(26, len(axes.flatten())):
  fig.delaxes(axes.flatten()[i])
fig.tight_layout(h_pad = 2)

plt.gcf().autofmt_xdate()
plt.show()
```

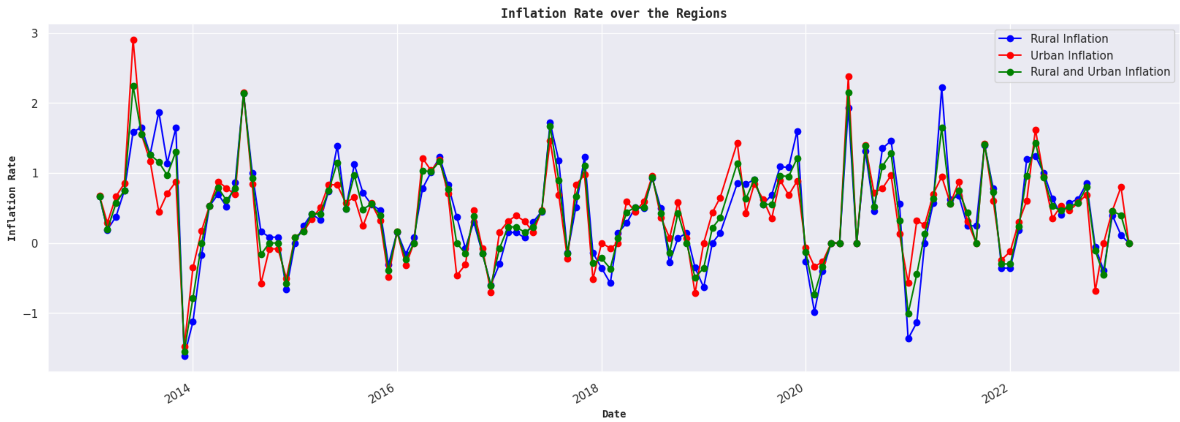


9 Inflation calculation over months

[32]:
```
infl_r = (df_r['General index'].diff()/df_r['General index'].shift())*100
infl_u = (df_u['General index'].diff()/df_u['General index'].shift())*100
infl_ru = (df_ru['General index'].diff()/df_ru['General index'].shift())*100
```

10 Visualisation of Inflation for different sectors

[33]:
```
fig, ax = plt.subplots(figsize = (20,7))

ax.plot_date(df_r.Date, infl_r, color="blue", label="Rural Inflation", linestyle="-")
ax.plot_date(df_u.Date, infl_u, color="red", label="Urban Inflation", linestyle="-")
ax.plot_date(df_ru.Date,infl_ru , color="green", label="Rural and Urban Inflation",␣
↪linestyle="-")

ax.set_xlabel('Date',fontproperties={'family':'monospace', 'size':10,'weight':
↪'demibold'})
ax.set_ylabel('Inflation Rate',fontproperties={'family':'monospace', 'size':
↪10,'weight':'demibold'})
ax.set_title('Inflation Rate over the Regions',size = 30, fontproperties={'family':
↪'monospace', 'weight':'bold'})
ax.legend()

plt.gcf().autofmt_xdate()
plt.show()
```



11 Observation:

1. In all the sectors there is a positive growth in index numbers over the period of time.
2. For Rural sector, the *Housing* column was a redundant column.
3. Maximum fluctuations can be seen in *Vegetables, Egg, Fruits* columns for all the sectors.
4. Remaining many columns exhibit very low fluctuations over the years.
5. From the *Inflation* plots of the sectors, it can be claimed that the inflations are a stationary process.

[34]:
```
data = pd.DataFrame()
data['Date'] = df_r.Date
data['infl_r'] = infl_r
data['infl_u'] = infl_u
```

```
data['infl_ru'] = infl_ru
data['ind_r'] = df_r['General index']
data['ind_u'] = df_u['General index']
data['ind_ru'] = df_ru['General index']
```

Saving the data into a dataframe

[35]:
```
data.to_csv('inflation_data.csv',index=False)
```

`Further analysis and model fitting on the general CPI numbers were done in R-Studio`

[ ]:
```
```

1 Analysis and Forecasting of the Consumer Price Index Values

`The Analsis part of this work was done in R-Studio`

```
version[['version.string']]

## [1] "R version 4.2.1 (2022-06-23 ucrt)"
```

1.1 Loading Required Prerequisites

```
library(tidyverse)
library(tidyquant)
library(gridExtra)
library(tibbletime)
library(forecast)
library(itsmr)
library(here)
library(fpp2)
library(tseries)
library(ggplot2)
library(ggthemes)
library(patchwork)

#set theme
theme_set(theme_gdocs())
theme_set(theme_stata())
```

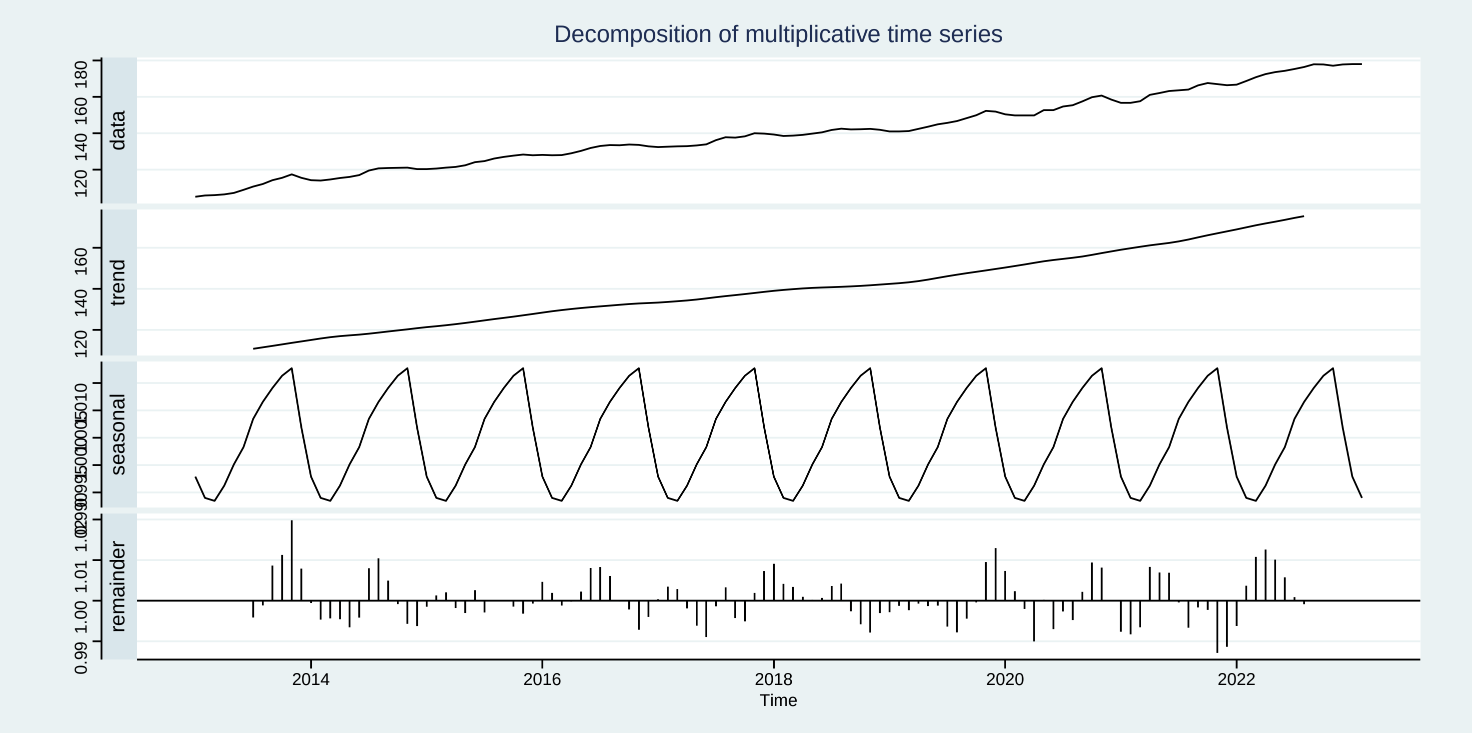
```
df = read.csv('inflation_data.csv')
head(df)

##         Date    infl_r    infl_u   infl_ru ind_r ind_u ind_ru
## 1 2013-01-01        NA        NA        NA 105.1 104.0  104.6
## 2 2013-02-01 0.6660324 0.6730769 0.6692161 105.8 104.7  105.3
## 3 2013-03-01 0.1890359 0.2865330 0.1899335 106.0 105.0  105.5
## 4 2013-04-01 0.3773585 0.6666667 0.5687204 106.4 105.7  106.1
## 5 2013-05-01 0.7518797 0.8514664 0.7540057 107.2 106.6  106.9
## 6 2013-06-01 1.5858209 2.9080675 2.2450889 108.9 109.7  109.3
```

1.2 Rural Consumer Price Index

1.2.1 Decomposition:

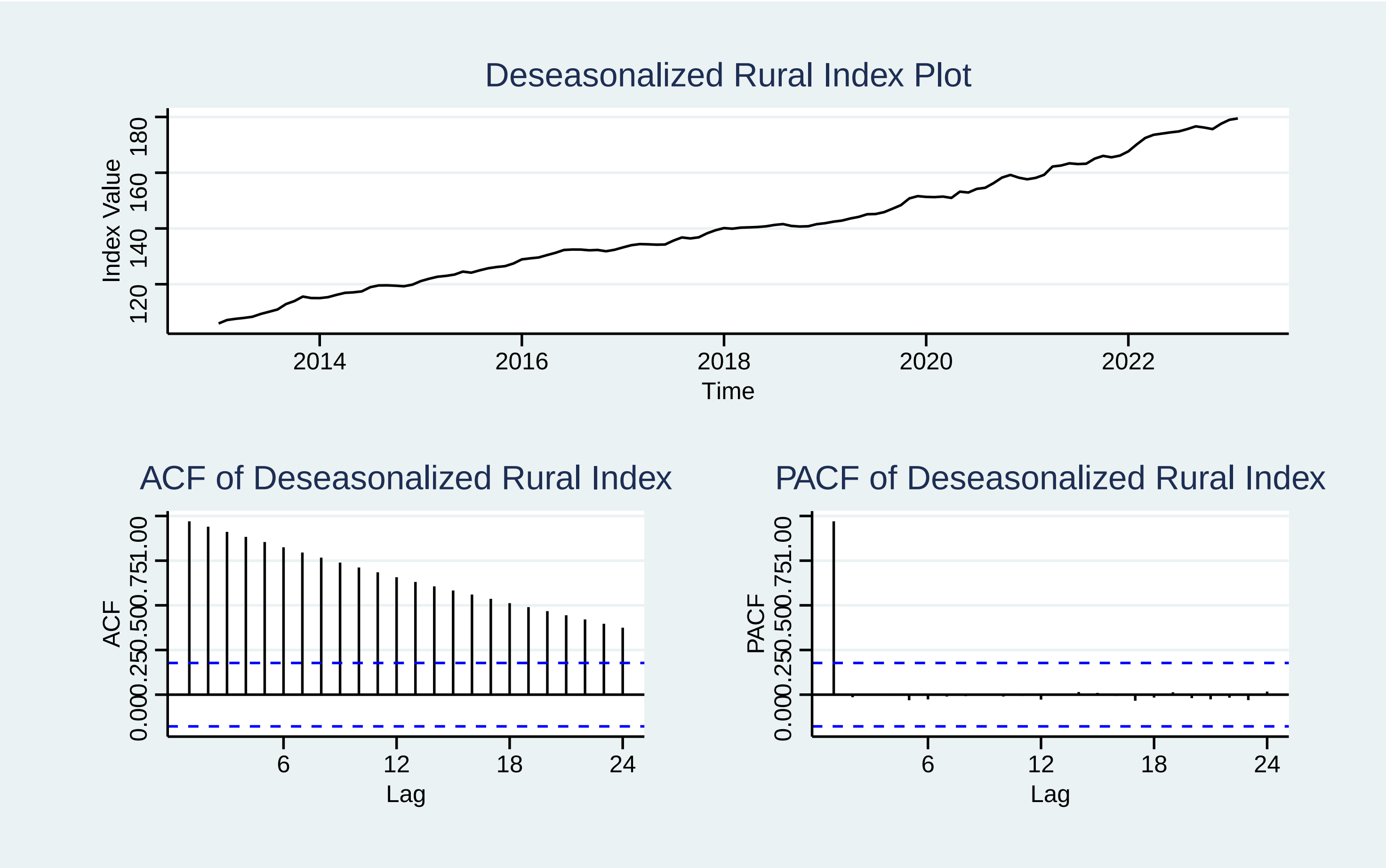
Decomposing the Rural Consumer Price Index (CPI) into different time series components:

```
rural <- ts(df$ind_r, start=c(2013,1),frequency = 12)
forecast::autoplot(decompose(rural,type="multiplicative"))
```



From the above graph, it is clear that the rural CPI has an influence of trend and seasonal components. Those factors are need to be removed in order to fit a model over the data.

```
rur_decomp <- stl(rural, s.window = 12) # estimate the seasonal component
rur_adjseason <- rural - seasonal(rur_decomp) #deseason the data
plot1 <- autoplot(rur_adjseason,ylab='Index Value',main='Deseasonalized Rural Index Plot') #plot
plot2 <- ggAcf(rur_adjseason) + ggtitle('ACF of Deseasonalized Rural Index')
plot3 <- ggPacf(rur_adjseason) + ggtitle('PACF of Deseasonalized Rural Index')
plot1 /(plot2 | plot3)
```



1.2.2 Model Building:

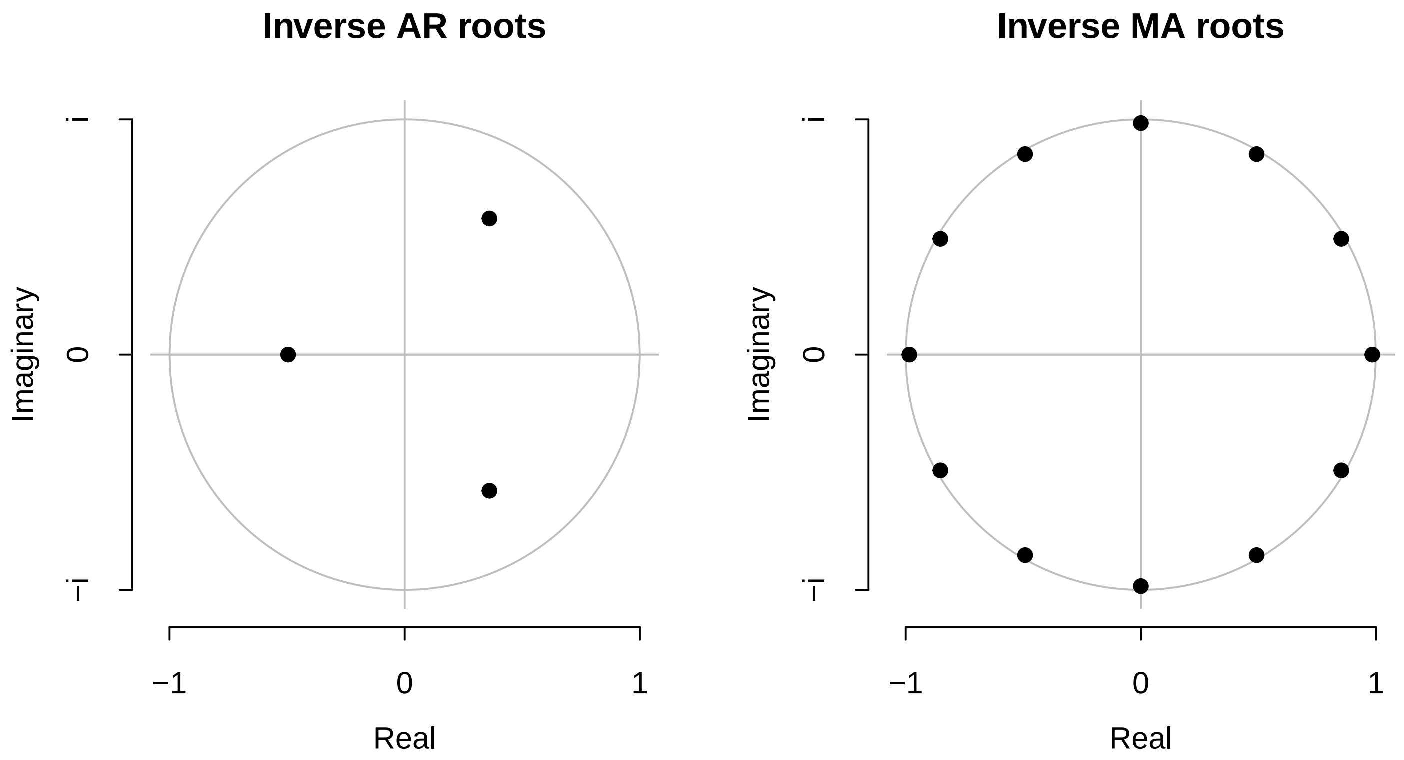
Fitting the best fitted SARIMA model over the Rural Index Number

```r
rur_sarima <- auto.arima(rural, stepwise = F, approximation = F, seasonal = T, allowdrift = F,
                         parallel = F, trace = F)
summary(rur_sarima)

## Series: rural
## ARIMA(3,1,0)(0,1,1)[12]
##
## Coefficients:
##          ar1      ar2      ar3     sma1
##       0.2251  -0.1075  -0.2303  -0.8275
## s.e.  0.0931   0.0954   0.0947   0.1539
##
## sigma^2 = 0.653:  log likelihood = -136.21
## AIC=282.41   AICc=282.99   BIC=295.87
##
## Training set error measures:
##                       ME      RMSE     MAE         MPE     MAPE       MASE
## Training set 0.002089793 0.7496647 0.55366 -0.01108663 0.381082 0.07761259
##                     ACF1
## Training set 0.001313648
```

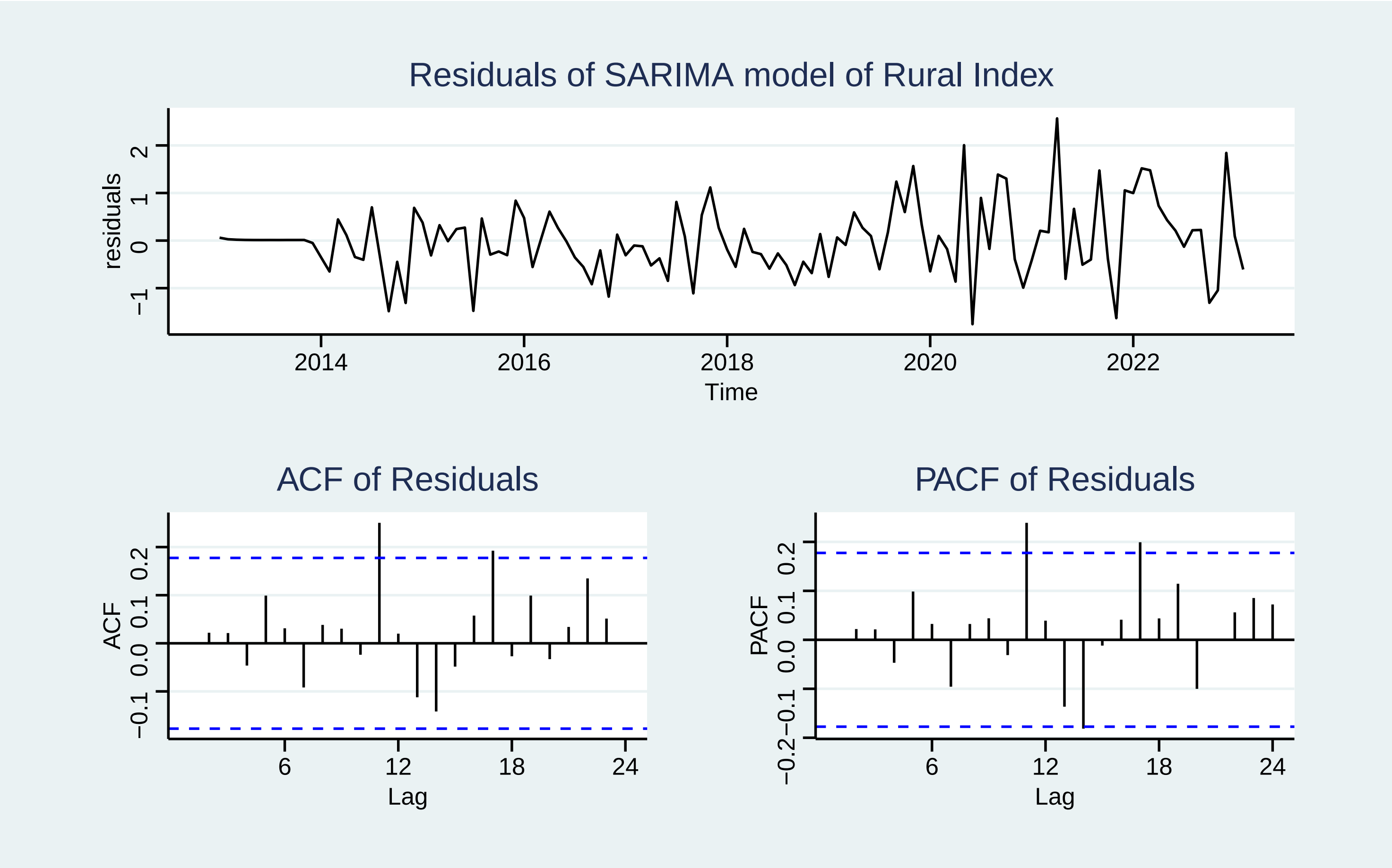
∴ The best fitted model is:

$$(1 - 0.2251B + 0.1075B^2 + 0.2303B^3)(1 - B)(1 - B^{12})X_t = (1 - 0.8275B^{12})Z_t$$

```r
plot(rur_sarima) # inspect the roots
```




```r
plot1 <- autoplot(residuals(rur_sarima),ylab='residuals',main='Residuals of SARIMA model of Rural Index')
plot2 <- ggAcf(residuals(rur_sarima)) +  ggtitle('ACF of Residuals')
plot3 <- ggPacf(residuals(rur_sarima))+  ggtitle('PACF of Residuals')

plot1 / (plot2 | plot3)
```



After successfully fitting the SARIMA model, the periodicity in the residuals have been removed which can be shown in the ACF and PACF plots.

1.2.3 Forecasting:

Performing forecasting from the given data

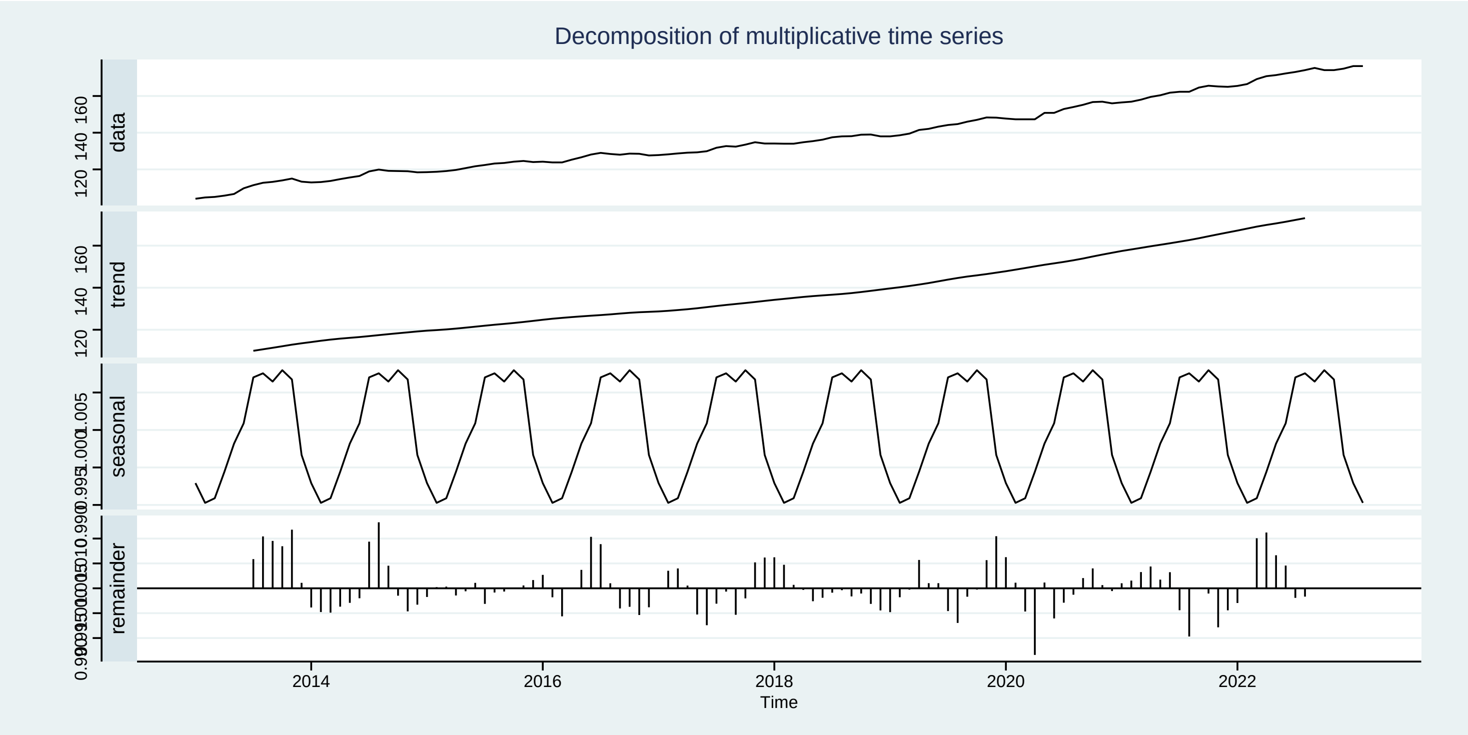
```
train_rur <- window(rural, end=c(2022,8))
test <- window(rural, start=c(2022,9))
sarimab2 <- auto.arima(train_rur,stepwise = F, approximation = F, seasonal = T, allowdrift = F,
                       parallel = F, trace = F)

forecasts <- forecast::forecast(sarimab2, h = 40)
fore1 <-autoplot(forecasts)+autolayer(rural) + ggtitle("Rural CPI Number")
forecast::accuracy(forecasts,test)

##                        ME      RMSE       MAE          MPE      MAPE       MASE
## Training set  0.005771456 0.7251087 0.5135852 -0.008749863 0.3597190 0.07410219
## Test set     -0.351384173 0.9406011 0.7141639 -0.198477039 0.4023338 0.10304251
##                    ACF1 Theil's U
## Training set 0.02709658        NA
## Test set     0.18289626  2.221401
```

1.3 Urban Consumer Price Index

1.3.1 Decomposition:

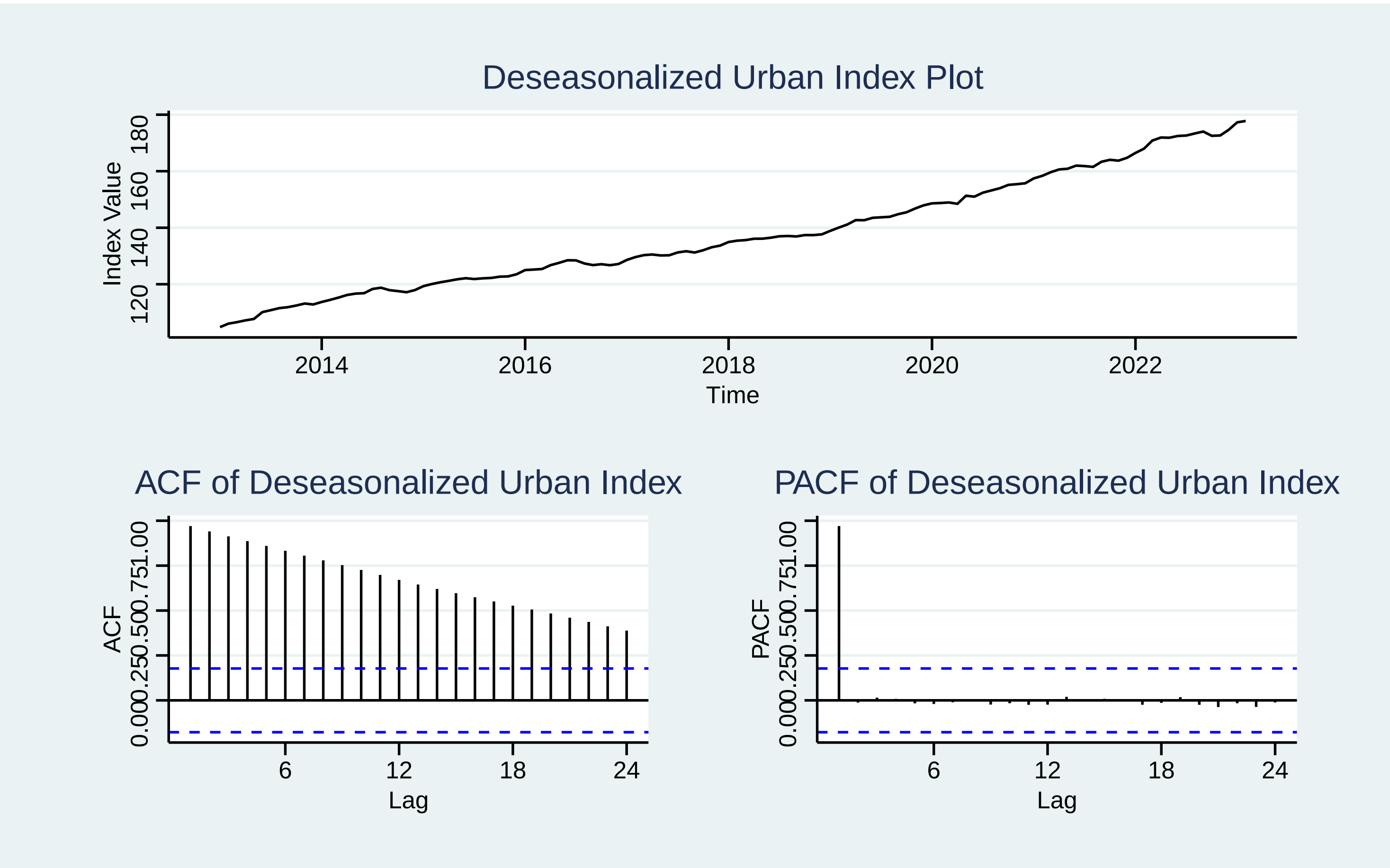
Decomposing the Urban Consumer Price Index (CPI) into different time series components:

```
urban <- ts(df$ind_u, start=c(2013,1),frequency = 12)
forecast::autoplot(decompose(urban,type="multiplicative"))
```



From the above graph, it is clear that the urban CPI has an influence of trend and seasonal components. Those factors are need to be removed in order to fit a model over the data.

```
ur_decomp <- stl(urban, s.window = 12) # estimate the seasonal component
ur_adjseason <- urban - seasonal(rur_decomp) #deseason the data
plot1 <- autoplot(ur_adjseason,ylab=’Index Value’,main=’Deseasonalized Urban Index Plot’) #plot
plot2 <- ggAcf(ur_adjseason) + ggtitle(’ACF of Deseasonalized Urban Index’)
plot3 <- ggPacf(ur_adjseason) + ggtitle(’PACF of Deseasonalized Urban Index’)
plot1 /(plot2 | plot3)
```



1.3.2 Model Building:

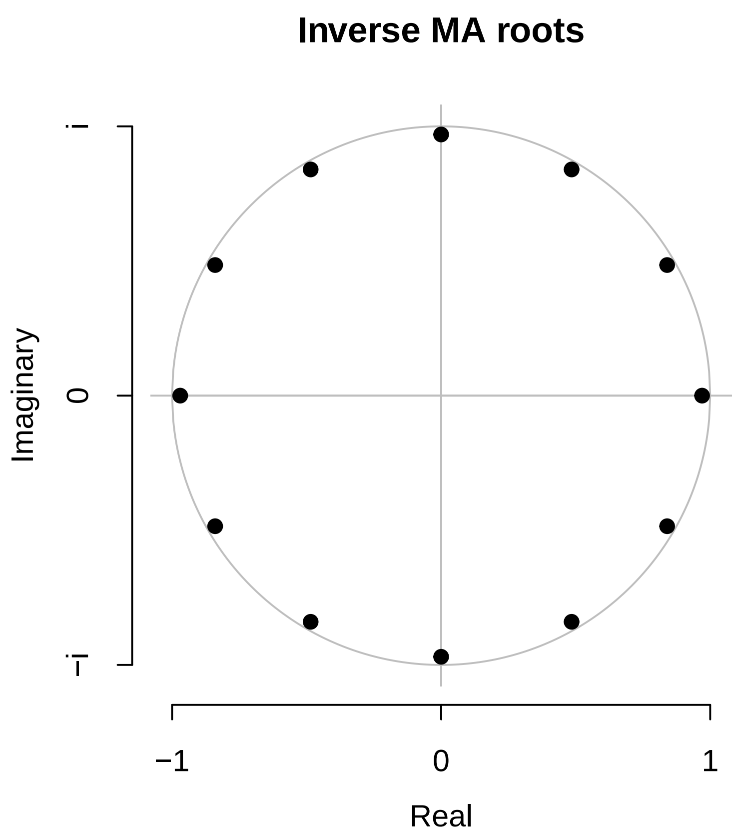
Fitting the best fitted SARIMA model over the Urban Index Number

```r
ur_sarima <- auto.arima(urban, stepwise = F, approximation = F, seasonal = T, allowdrift = F,
                         parallel = F, trace = F)
summary(ur_sarima)

## Series: urban
## ARIMA(0,1,0)(0,1,1)[12]
##
## Coefficients:
##          sma1
##       -0.6922
## s.e.   0.1022
##
## sigma^2 = 0.5895:  log likelihood = -129.15
## AIC=262.31   AICc=262.42   BIC=267.69
##
## Training set error measures:
##                      ME      RMSE       MAE         MPE      MAPE       MASE
## Training set 0.02249555 0.7223709 0.5243375 0.003239058 0.3690756 0.07523757
##                    ACF1
## Training set 0.06816845
```

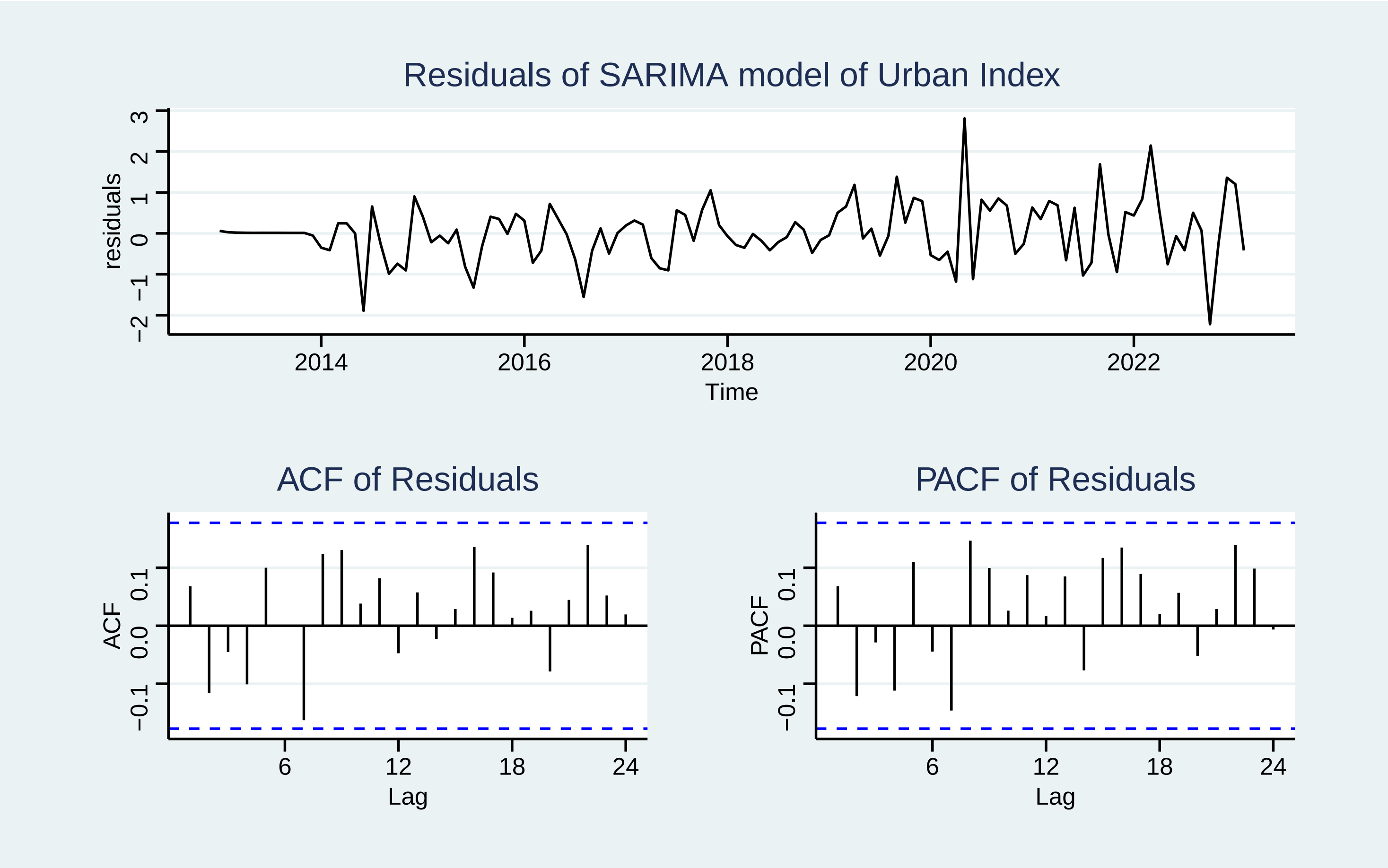
∴ The best fitted model is:

$$(1-B)(1-B^{12})X_t = (1-0.6922B^{12})Z_t$$

```r
plot(ur_sarima) # inspect the roots
```




```r
plot1 <- autoplot(residuals(ur_sarima),ylab='residuals',main='Residuals of SARIMA model of Urban Index')
plot2 <- ggAcf(residuals(ur_sarima))+ ggtitle('ACF of Residuals')
plot3 <- ggPacf(residuals(ur_sarima)) + ggtitle('PACF of Residuals')

plot1 / (plot2 | plot3)
```



After successfully fitting the SARIMA model, the periodicity in the residuals have been removed which can be shown in the ACF and PACF plots.

1.3.3 Forecasting:

Performing forecasting from the given data.

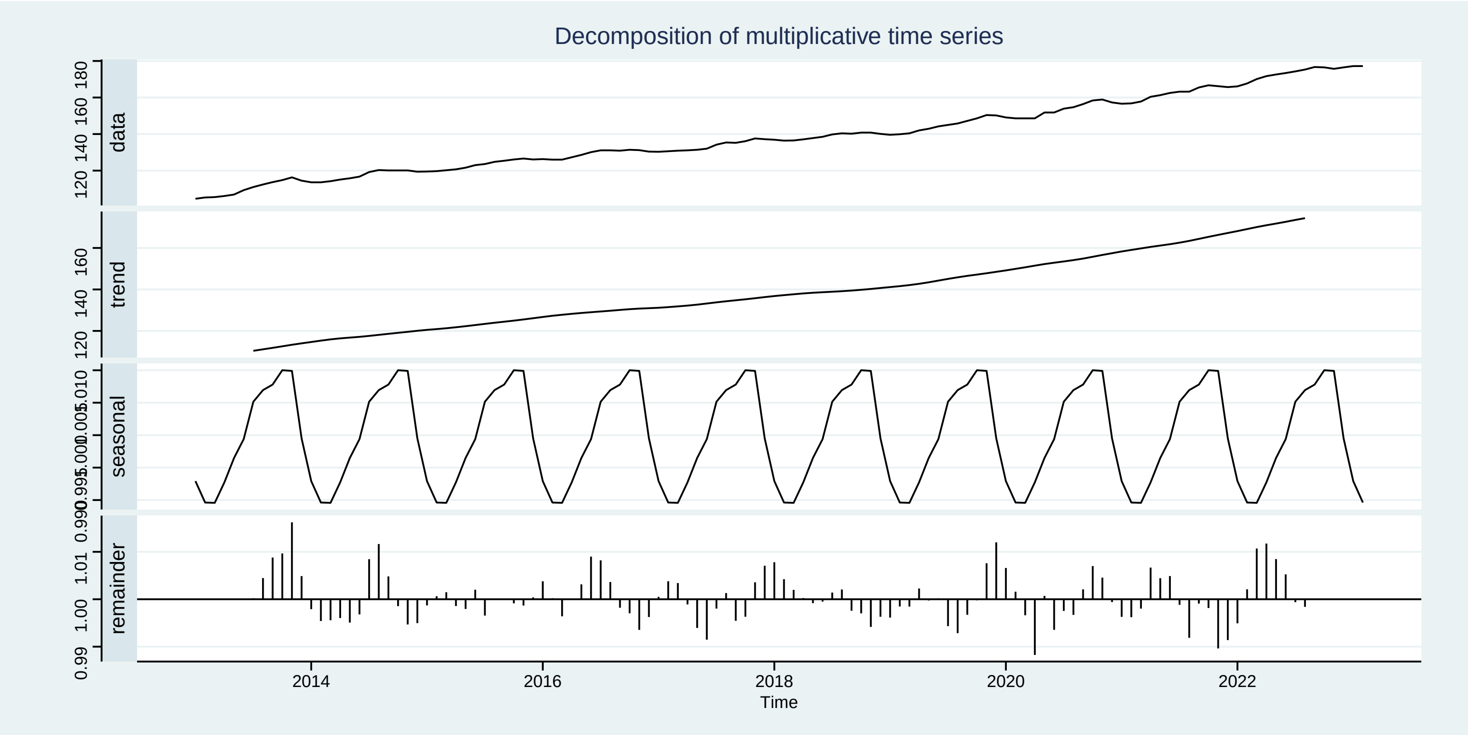
```
train_ur <- window(urban, end=c(2022,8))
test <- window(urban, start=c(2022,9))
sarimab2 <- auto.arima(train_ur,stepwise = F, approximation = F, seasonal = T, allowdrift = F,
                       parallel = F, trace = F)

forecasts <- forecast::forecast(sarimab2, h = 40)
fore2 <-autoplot(forecasts)+autolayer(urban)+ ggtitle("Urban CPI Number")
forecast::accuracy(forecasts,test)

##                       ME      RMSE       MAE          MPE      MAPE       MASE
## Training set  0.02599193 0.6872455 0.5013260  0.004731289 0.3591034 0.07364111
## Test set     -0.86120116 1.3449544 0.9869431 -0.494431286 0.5658951 0.14497470
##                    ACF1 Theil's U
## Training set 0.06000911        NA
## Test set     0.22241215   1.64087
```

1.4 Rural and Urban Consumer Price Index

1.4.1 Decomposition:

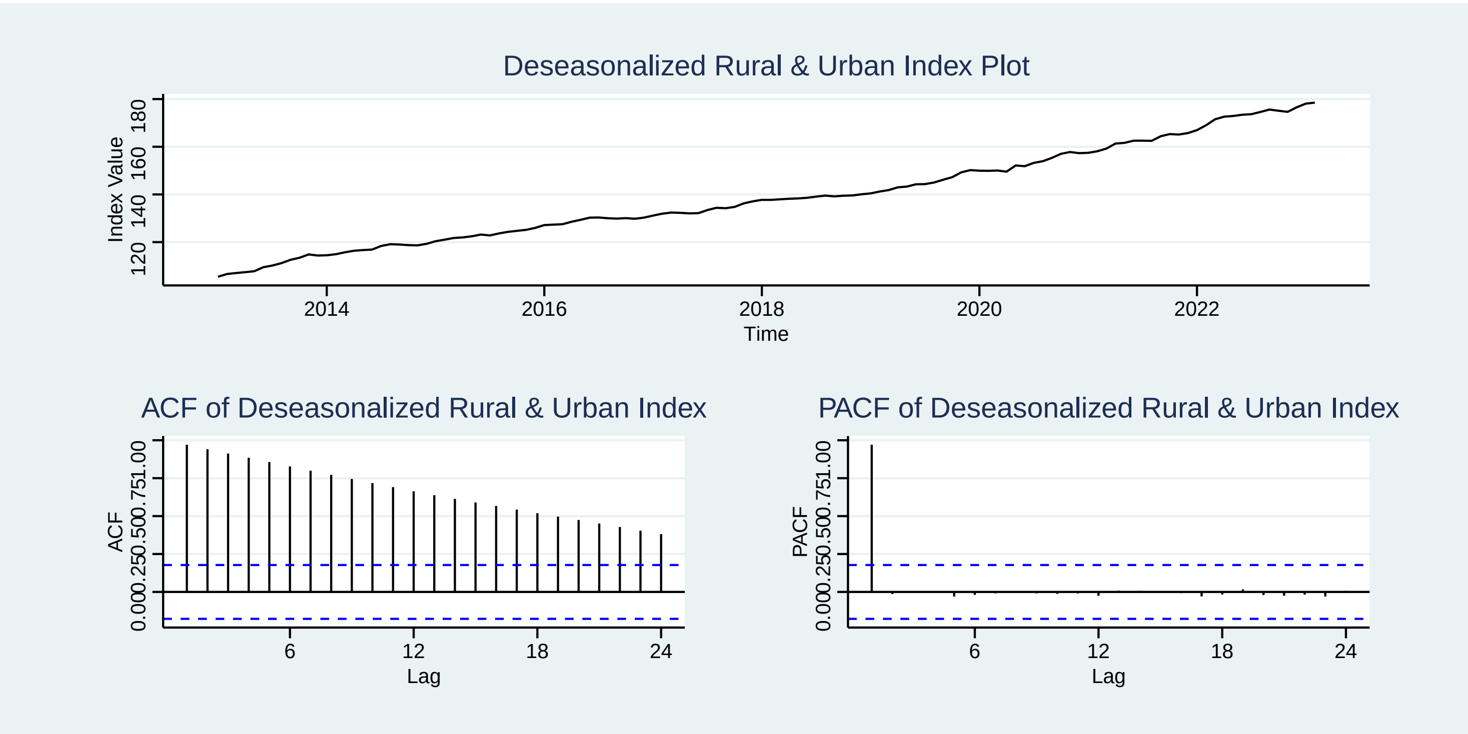
Decomposing the Rural and Urban Combined Consumer Price Index (CPI) into different time series components:

```
ru <- ts(df$ind_ru, start=c(2013,1),frequency = 12)
#options(repr.plot.width=3,repr.plot.height=3)
forecast::autoplot(decompose(ru,type="multiplicative"))
```



From the above graph, it is clear that the rural and urban combined CPI has an influence of trend and seasonal components. Those factors are need to be removed in order to fit a model over the data.

```
ru_decomp <- stl(ru, s.window = 12) # estimate the seasonal component
ru_adjseason <- ru - seasonal(ru_decomp) #deseason the data
plot1 <- autoplot(ru_adjseason,ylab= 'Index Value', main='Deseasonalized Rural & Urban Index Plot') #plot
plot2 <- ggAcf(ru_adjseason) + ggtitle('ACF of Deseasonalized Rural & Urban Index')
plot3 <- ggPacf(ru_adjseason) + ggtitle('PACF of Deseasonalized Rural & Urban Index')
plot1 /(plot2 | plot3)
```



1.4.2 Model Building:

Fitting the best fitted SARIMA model over the Rural and Urban Combined Index Number

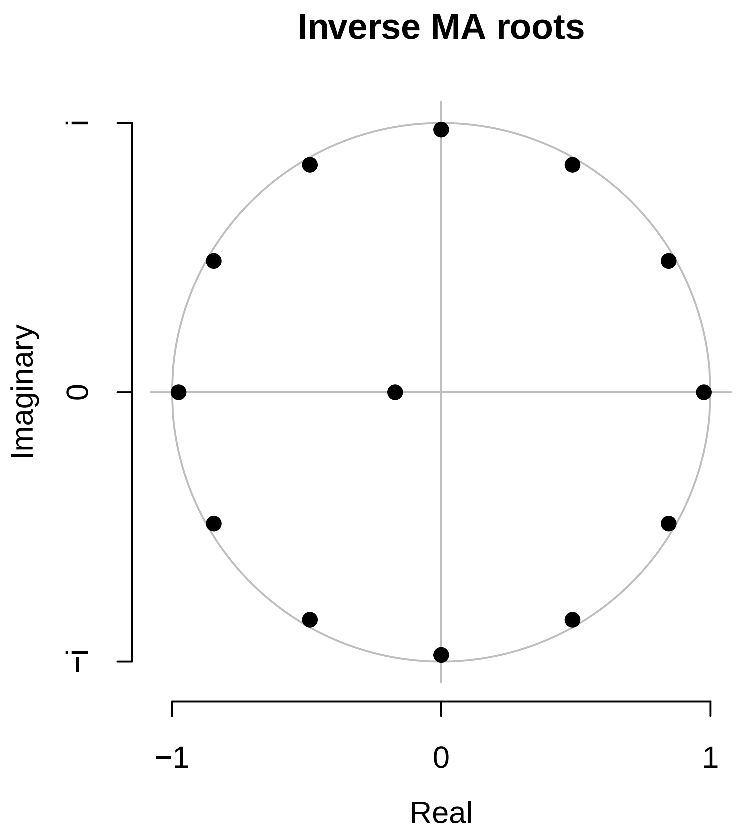
```
ru_sarima <- auto.arima(ru, stepwise = F, approximation = F, seasonal = T, allowdrift = F,
                        parallel = F, trace = F)
summary(ru_sarima)

## Series: ru
```

```
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##          ma1     sma1
##       0.1711  -0.7422
## s.e.  0.0993   0.1113
##
## sigma^2 = 0.5952:  log likelihood = -130.07
## AIC=266.13   AICc=266.36   BIC=274.21
##
## Training set error measures:
##                      ME      RMSE      MAE          MPE      MAPE       MASE
## Training set 0.01165685 0.7225065 0.534809 -0.002287157 0.3715703 0.07573248
##                      ACF1
## Training set -0.009995881
```

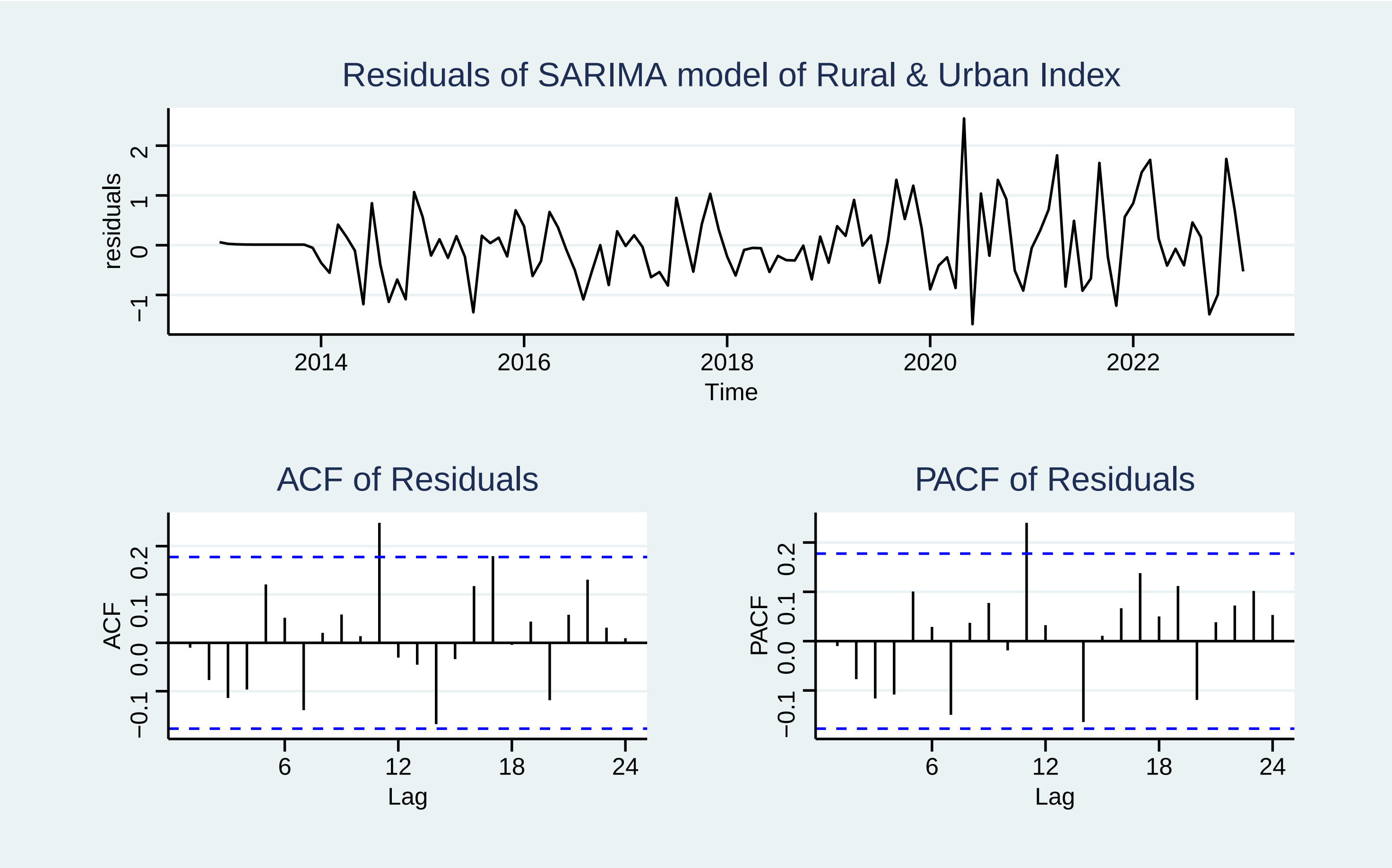
∴ The best fitted model is:

$$(1 - B)(1 - B^{12})X_t = (1 + 0.1711B)(1 - 0.7422B^{12})Z_t$$

```
plot(ru_sarima) # inspect the roots
```




```
plot1 <- autoplot(residuals(ru_sarima),ylab='residuals',main='Residuals of SARIMA model of Rural & Urban Index
plot2 <- ggAcf(residuals(ru_sarima))+ ggtitle('ACF of Residuals')
plot3 <- ggPacf(residuals(ru_sarima))+ ggtitle('PACF of Residuals')

plot1 / (plot2 | plot3)
```



After successfully fitting the SARIMA model, the periodicity in the residuals have been removed which can be shown in the ACF and PACF plots.

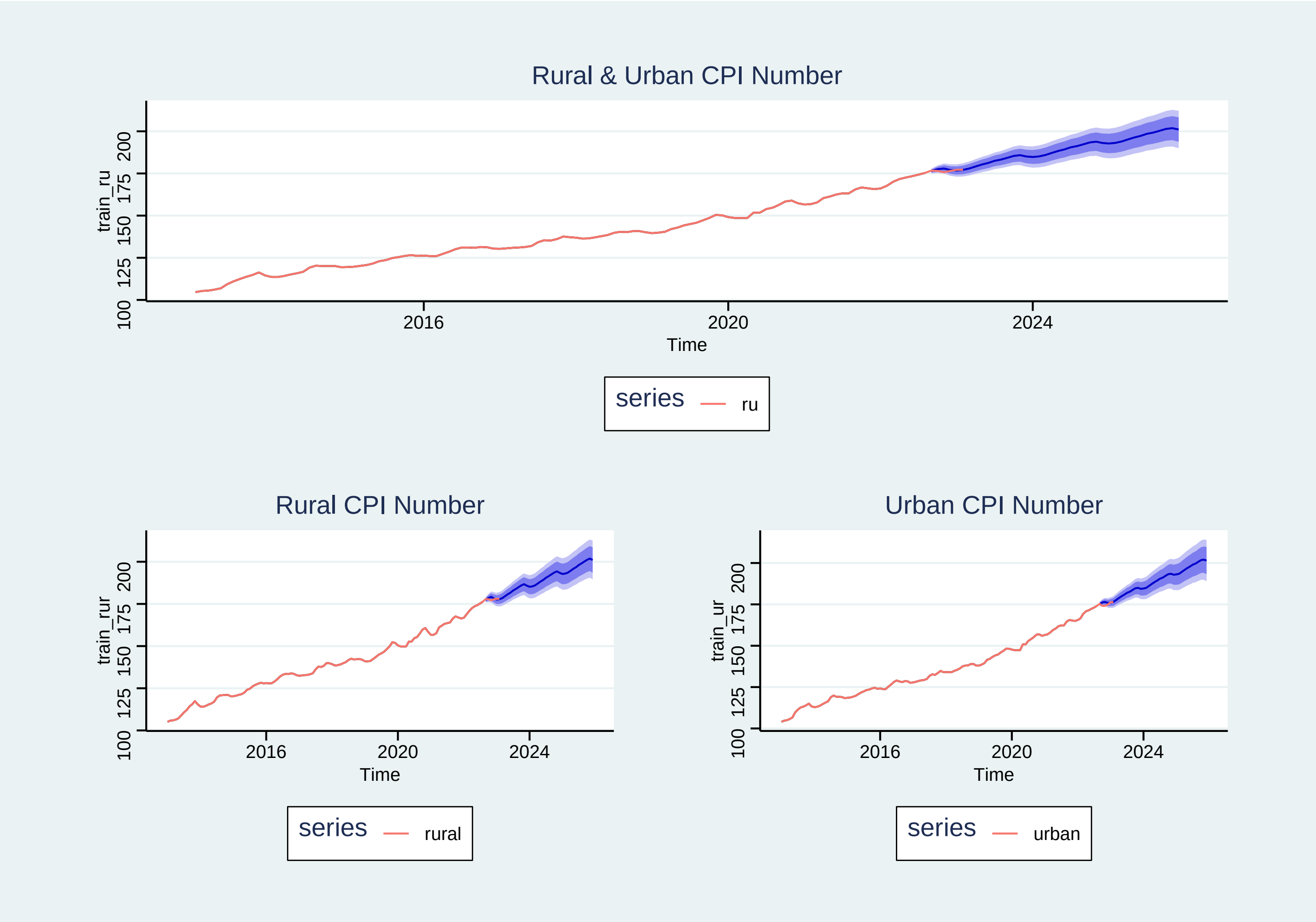
1.4.3 Forecasting:

Performing forecasting from the given data.

```
train_ru <- window(ru, end=c(2022,8))
test <- window(ru, start=c(2022,9))
sarimab2 <- auto.arima(train_ru,stepwise = F, approximation = F, seasonal = T, allowdrift = F,
                       parallel = F, trace = F)

forecasts <- forecast::forecast(sarimab2, h = 40)
fore3 <- autoplot(forecasts)+autolayer(ru)+ ggtitle("Rural & Urban CPI Number")
forecast::accuracy(forecasts,test)

##                      ME      RMSE       MAE          MPE      MAPE       MASE
## Training set  0.0174232 0.6699495 0.4864256 -0.001910502 0.3446993 0.07076272
## Test set     -0.4975685 1.0471078 0.7693449 -0.283095274 0.4366137 0.11192037
##                    ACF1 Theil's U
## Training set -0.0455699        NA
## Test set      0.2173080  1.892605
```

Forecasts for the given Index numbers

```
fore3 / (fore1 | fore2)
```


Rural & Urban CPI Number
train_ru
100
125
150
175
200
2016
2020
2024
Time
series
ru
Rural CPI Number
train_rur
100
125
150
175
200
2016
2020
2024
Time
series
rural
Urban CPI Number
train_ur
100
125
150
175
200
2016
2020
2024
Time
series
urban

The End