ts_xterisation_rca

October 16, 2019

Time Series Characterisation

Table of Contents

DATA SOURCES

This project makes use of is a time series of export data from **The Observatory of Economic Complexity(OEC)** based on the SITC (Standard International Trade Classification)** from 1962 - 2000, with data from **The Center for International Data** from Robert Feenstra. The more recent data 2001 - 2017, is sourced from **UN COMTRADE**.

Full **OEC** trade datasets are also available from a data dump on **SITC Product 4 Digit**

1 Importing Libraries for time series forecasting

[1]:
```
!pip install -r requirements.txt
```

```
Requirement already satisfied: attrs==19.3.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 1)) (19.3.0)
Requirement already satisfied: backcall==0.1.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 2)) (0.1.0)
Requirement already satisfied: bleach==3.1.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 3)) (3.1.0)
Requirement already satisfied: cycler==0.10.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 4)) (0.10.0)
Requirement already satisfied: decorator==4.4.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 5)) (4.4.0)
Requirement already satisfied: defusedxml==0.6.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 6)) (0.6.0)
Requirement already satisfied: entrypoints==0.3 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 7)) (0.3)
Requirement already satisfied: importlib-metadata==0.23 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 8)) (0.23)
Requirement already satisfied: ipykernel==5.1.2 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 9)) (5.1.2)
Requirement already satisfied: ipython==7.8.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 10)) (7.8.0)
Requirement already satisfied: ipython-genutils==0.2.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 11)) (0.2.0)
Requirement already satisfied: ipywidgets==7.5.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 12)) (7.5.1)
Requirement already satisfied: jedi==0.15.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 13)) (0.15.1)
Requirement already satisfied: Jinja2==2.10.3 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 14)) (2.10.3)
Requirement already satisfied: joblib==0.14.0 in
```

```
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 15)) (0.14.0)
Requirement already satisfied: jsonschema==3.1.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 16)) (3.1.1)
Requirement already satisfied: jupyter==1.0.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 17)) (1.0.0)
Requirement already satisfied: jupyter-client==5.3.4 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 18)) (5.3.4)
Requirement already satisfied: jupyter-console==6.0.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 19)) (6.0.0)
Requirement already satisfied: jupyter-core==4.6.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 20)) (4.6.0)
Requirement already satisfied: kiwisolver==1.1.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 21)) (1.1.0)
Requirement already satisfied: MarkupSafe==1.1.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 22)) (1.1.1)
Requirement already satisfied: matplotlib==3.1.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 23)) (3.1.1)
Requirement already satisfied: mistune==0.8.4 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 24)) (0.8.4)
Requirement already satisfied: more-itertools==7.2.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 25)) (7.2.0)
Requirement already satisfied: nbconvert==5.6.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 26)) (5.6.0)
Requirement already satisfied: nbformat==4.4.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 27)) (4.4.0)
Requirement already satisfied: notebook==6.0.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 28)) (6.0.1)
Requirement already satisfied: numpy==1.17.2 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 29)) (1.17.2)
Requirement already satisfied: pandas==0.25.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 30)) (0.25.1)
Requirement already satisfied: pandocfilters==1.4.2 in
```

```
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 31)) (1.4.2)
Requirement already satisfied: parso==0.5.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 32)) (0.5.1)
Requirement already satisfied: patsy==0.5.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 33)) (0.5.1)
Requirement already satisfied: pexpect==4.7.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 34)) (4.7.0)
Requirement already satisfied: pickleshare==0.7.5 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 35)) (0.7.5)
Requirement already satisfied: prometheus-client==0.7.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 36)) (0.7.1)
Requirement already satisfied: prompt-toolkit==2.0.10 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 37)) (2.0.10)
Requirement already satisfied: ptyprocess==0.6.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 38)) (0.6.0)
Requirement already satisfied: Pygments==2.4.2 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 39)) (2.4.2)
Requirement already satisfied: pyparsing==2.4.2 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 40)) (2.4.2)
Requirement already satisfied: pyrsistent==0.15.4 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 41)) (0.15.4)
Requirement already satisfied: python-dateutil==2.8.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 42)) (2.8.0)
Requirement already satisfied: pytz==2019.3 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 43)) (2019.3)
Requirement already satisfied: pyzmq==18.1.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 44)) (18.1.0)
Requirement already satisfied: qtconsole==4.5.5 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 45)) (4.5.5)
Requirement already satisfied: scikit-learn==0.21.3 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 46)) (0.21.3)
Requirement already satisfied: scipy==1.3.1 in
```

```
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 47)) (1.3.1)
Requirement already satisfied: seaborn==0.9.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 48)) (0.9.0)
Requirement already satisfied: Send2Trash==1.5.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 49)) (1.5.0)
Requirement already satisfied: six==1.12.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 50)) (1.12.0)
Requirement already satisfied: sklearn==0.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 51)) (0.0)
Requirement already satisfied: statsmodels==0.10.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 52)) (0.10.1)
Requirement already satisfied: terminado==0.8.2 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 53)) (0.8.2)
Requirement already satisfied: testpath==0.4.2 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 54)) (0.4.2)
Requirement already satisfied: tornado==6.0.3 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 55)) (6.0.3)
Requirement already satisfied: traitlets==4.3.3 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 56)) (4.3.3)
Requirement already satisfied: wcwidth==0.1.7 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 57)) (0.1.7)
Requirement already satisfied: webencodings==0.5.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 58)) (0.5.1)
Requirement already satisfied: widgetsnbextension==3.5.1 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 59)) (3.5.1)
Requirement already satisfied: zipp==0.6.0 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from -r requirements.txt
(line 60)) (0.6.0)
Requirement already satisfied: setuptools>=18.5 in
/home/webber/.Envs/tsa/lib/python3.7/site-packages (from ipython==7.8.0->-r
requirements.txt (line 10)) (41.4.0)
```

[3]:
```
import numpy as np
import pandas as pd
```

```
pd.set_option('display.expand_frame_repr', False)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)

import statsmodels.robust
import statsmodels.tsa.stattools as tsa

from statsmodels import api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose

from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from pylab import rcParams

from math import sqrt

import matplotlib
import matplotlib.pyplot as plt
matplotlib.colors
matplotlib.rcParams['axes.labelsize'] = 14
matplotlib.rcParams['xtick.labelsize'] = 12
matplotlib.rcParams['ytick.labelsize'] = 12
matplotlib.rcParams['text.color'] = 'k'
import seaborn as sns

from random import random

import warnings; import os
warnings.filterwarnings("ignore")
plt.style.available
plt.style.use('bmh')
import os
```

2 Importing data

- Dataset: Trade data:
  - location_id: /- product_id: SITC 4 digit code system
  - year: from 1963 to 2017 - export_value: / - import_value: - / - export_rca:
  - is_new: / - cog: / - distance: / - normalized_distance: / - normalized_cog:
  - normalized_pci: /- export_rpop: / - sitc_eci: / - sitc_coi: / - pci: /- location_code: /- location_name_short_en:
  - sitc_product_code: / - sitc_product_name_short_en: /- Unit: arbitrary

[4]:
```
product = 'engine_parts' # Create a dictionary to store product code and sitc␣
↪code
experiment = 'rca_tests'

def load_data(csv_name,trade_form,target_value, code_system,sitc_code,ts_type):
    '''
    Ingests SITC multi-time series trade data in csv format, filters for one␣
↪product code, removes zero values for
    target value, screens target values and renames target columns & creates␣
↪dataframe for the target columns

    Parameters:
    -----------
    csv:
    trade_form:
    target_value:
    code_system:
    sitc_code:
    ts_type:
    '''
    data = pd.read_csv(csv_name)#,header=None
    # TO DO: Create for loop to read sitc code from dictionary
    dframe = data.loc[data[code_system]==sitc_code]
    dframe.rename(columns={'location_name_short_en': 'exporter','export_rca':␣
↪'rca'}, inplace=True)
    dframe = dframe[[ts_type,trade_form,target_value]]
    dframe = dframe[dframe[trade_form] != 'Undeclared Countries']
    dframe[target_value].replace('',0, inplace=True)
    #dframe = dframe[dframe[target_value] != 0]
    dframe = dframe[dframe[target_value] >= 0.01]
    return dframe
```

[5]:
```
xdf = load_data('sitc4digit_year.
↪csv','exporter','rca','sitc_product_code',7132,'year')
xdf.columns
```

[5]:
```
Index(['year', 'exporter', 'rca'], dtype='object')
```

[6]:
```
xdf.describe()
```

[6]:

	year	rca
count	4282.000000	4282.000000
mean	1989.665343	0.636428
std	17.112305	2.854036
min	1962.000000	0.010011
25%	1974.000000	0.024436
50%	1990.000000	0.070735

```
75%     2006.000000      0.392209
max     2017.000000    114.697680
```

[261]:
```
exporters = xdf['exporter'].unique()
```

[7]:
```
time  = xdf['year'].unique()
```

[18]:
```
print(f'Shape of the dataframe: {xdf.shape} \n Data ranges from {xdf.year.
 ↪min()} to {xdf.year.max()}')
print(f'The rca values range from {xdf.rca.min()} to {xdf.rca.max()}'))
print('Countries not declared are {} out of {}'.format(xdf['exporter'].
 ↪loc[xdf['exporter']=='Undeclared Countries'].count(),xdf.exporter.count()))
print('Zero export values: {}'.format(xdf['rca'][xdf['rca']==0].count()))
print('Null export values: {}'.format(xdf['rca'].isnull().sum()))
```

```
Shape of the dataframe: (4282, 3)
 Data ranges from 1962 to 2017
The rca values range from 0.010010773 to 114.69768
Countries not declared are 0 out of 4282
Zero export values: 0
Null export values: 0
```

3 Data Preprocessing and Visualization

Converting to datetime format:

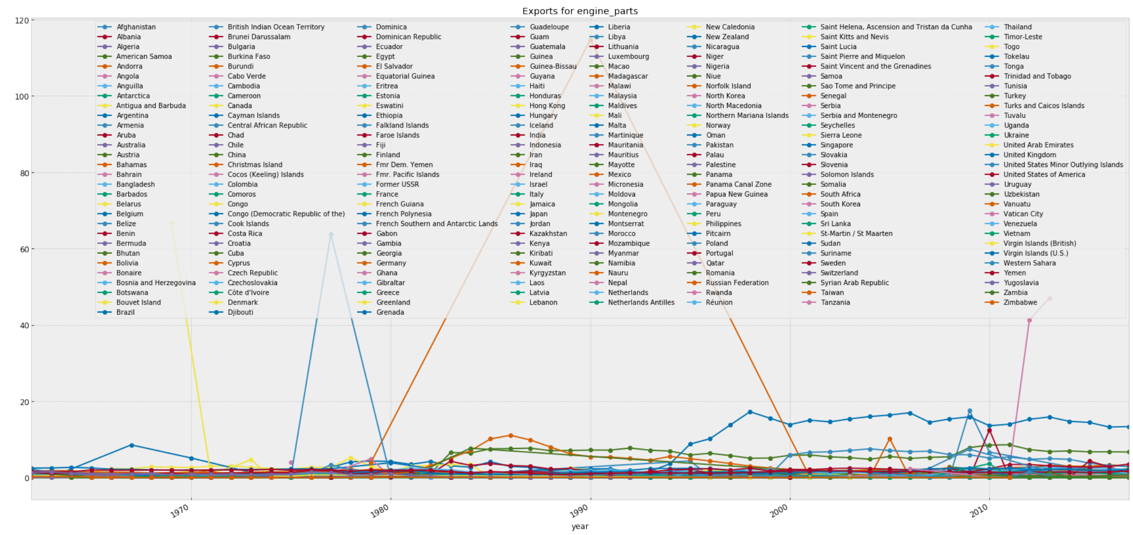
[41]:
```
xdf['year'] = pd.to_datetime(xdf['year'], format='%Y')
xdf.columns
```

[41]:
```
Index(['year', 'exporter', 'rca'], dtype='object')
```

[42]:
```
#xdf = xdf.drop(columns=["rca"])
#xdf.columns
```

[285]:
```
#sns.scatterplot(x='year',y='rca',data=xdf)
#sns.pairplot(xdf)
```

[128]:
```
fig, ax = plt.subplots(figsize=(30,15))
for key, grp in xdf.groupby(['exporter']):
        ax = grp.plot(ax=ax, kind='line', x='year', y='rca', label=key,␣
 ↪marker='o',title='Exports for {}'.format(product))
        ax.legend(ncol=8, loc='best')
        target = 'images\\{}\\{}'.format(product,experiment)
        savefile = os.path.join(os.getcwd(),target,'Exports for {}'.
 ↪format(product))
plt.savefig(savefile)
```



[45]:
```
df_grp = xdf.set_index('year')
df_grp.head(2)
```

[45]:
```
             exporter       rca
year
2003-01-01    Aruba  0.018273
2010-01-01    Aruba  0.092508
```

[294]:
```
df_grp = xdf.groupby([xdf.year.name, xdf.exporter.name]).mean().unstack()
df_grps = df_grp.apply(pd.to_numeric, errors='coerce').fillna(0,␣
 ↪downcast='infer')
df_grps.columns = df_grps.columns.droplevel()
df_grps.head(1)
```

[294]:
```
exporter    Afghanistan  Albania  Algeria  American Samoa  Andorra  Angola
Anguilla  Antarctica  Antigua and Barbuda  Argentina  Armenia  Aruba  Australia
Austria  Bahamas   Bahrain  Bangladesh  Barbados  Belarus   Belgium  Belize
Benin   Bermuda  Bhutan   Bolivia  Bonaire  Bosnia and Herzegovina  Botswana
Bouvet Island  Brazil  British Indian Ocean Territory  Brunei Darussalam
Bulgaria  Burkina Faso   Burundi  Cabo Verde  Cambodia  Cameroon   Canada
Cayman Islands  Central African Republic  Chad     Chile  China  Christmas
Island  Cocos (Keeling) Islands  Colombia  Comoros     Congo  Congo (Democratic
Republic of the)  Cook Islands  Costa Rica  Croatia  Cuba     Cyprus  Czech
Republic  Czechoslovakia  Côte d'Ivoire   Denmark  Djibouti  Dominica  Dominican
Republic  Ecuador     Egypt  El Salvador  Equatorial Guinea  Eritrea  Estonia
Eswatini  Ethiopia  Falkland Islands  Faroe Islands     Fiji   Finland  Fmr
Dem. Yemen  Fmr. Pacific Islands  Former USSR    France  French Guiana  French
Polynesia  French Southern and Antarctic Lands     Gabon    Gambia  Georgia
Germany     Ghana  Gibraltar    Greece  Greenland  Grenada  Guadeloupe  Guam
```

Guatemala Guinea Guinea-Bissau Guyana Haiti Honduras Hong Kong Hungary Iceland India Indonesia Iran Iraq Ireland Israel Italy Jamaica Japan Jordan Kazakhstan Kenya Kiribati Kuwait Kyrgyzstan Laos Latvia Lebanon Liberia Libya Lithuania Luxembourg Macao Madagascar Malawi Malaysia Maldives Mali Malta Martinique Mauritania Mauritius Mayotte Mexico Micronesia Moldova Mongolia Montenegro Montserrat Morocco Mozambique Myanmar Namibia Nauru Nepal Netherlands Netherlands Antilles New Caledonia New Zealand Nicaragua Niger Nigeria Niue Norfolk Island North Korea North Macedonia Northern Mariana Islands Norway Oman Pakistan Palau Palestine Panama Panama Canal Zone Papua New Guinea Paraguay Peru Philippines Pitcairn Poland Portugal Qatar Romania Russian Federation Rwanda Réunion Saint Helena, Ascension and Tristan da Cunha Saint Kitts and Nevis Saint Lucia Saint Pierre and Miquelon Saint Vincent and the Grenadines Samoa Sao Tome and Principe Senegal Serbia Serbia and Montenegro Seychelles Sierra Leone Singapore Slovakia Slovenia Solomon Islands Somalia South Africa South Korea Spain Sri Lanka St-Martin / St Maarten Sudan Suriname Sweden Switzerland Syrian Arab Republic Taiwan Tanzania Thailand Timor-Leste Togo Tokelau Tonga Trinidad and Tobago Tunisia Turkey Turks and Caicos Islands Tuvalu Uganda Ukraine United Arab Emirates United Kingdom United States Minor Outlying Islands United States of America Uruguay Uzbekistan Vanuatu Vatican City Venezuela Vietnam Virgin Islands (British) Virgin Islands (U.S.) Western Sahara Yemen Yugoslavia Zambia Zimbabwe
year
1962-01-01 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.018789
0.580496 0.14806 0.012963 0.0 0.01253 0.0 0.473824 0.0
0.076695 2.348061 0.0 0.017842 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.197875 0.021476
0.263002 0.014674 0.0 0.020073 0.208462 0.230185 0.0
0.033782 0.0 0.017811 0.0 0.0 0.0 0.0
0.0 0.0 0.060467 0.016014 0.0
0.022408 0.0 0.0 0.043419 0.0 1.192845 0.173982
1.848899 0.0 0.0 0.0 0.0 0.023439 0.0
0.0 0.0 0.0 0.0 0.021552 0.0 0.0
0.110766 0.047126 0.290738 0.0 0.127338 0.820721
0.0 0.0 0.0 0.027884 0.062736
0.0 1.674 0.042989 0.0 0.043051 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.028872 0.010109 0.0 0.0 0.148012
0.502856 0.011379 0.14151 0.0 0.0 0.0 0.108328 0.067291 1.321232
0.053677 0.569255 0.068373 0.0 0.014842 0.0 0.0 0.0 0.0
0.0 0.0 0.026381 0.01846 0.048717 0.0 0.0 0.0
0.046311 0.0 0.0 0.0 0.0 0.456691 0.0 0.029758
0.0 0.0 0.012943 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.011091 0.0 0.0 0.0 0.602013 0.491603
0.041534 0.0 0.0 0.0 0.036193 0.059694 0.0
0.0 0.0 0.0 0.0 0.554667 0.107453

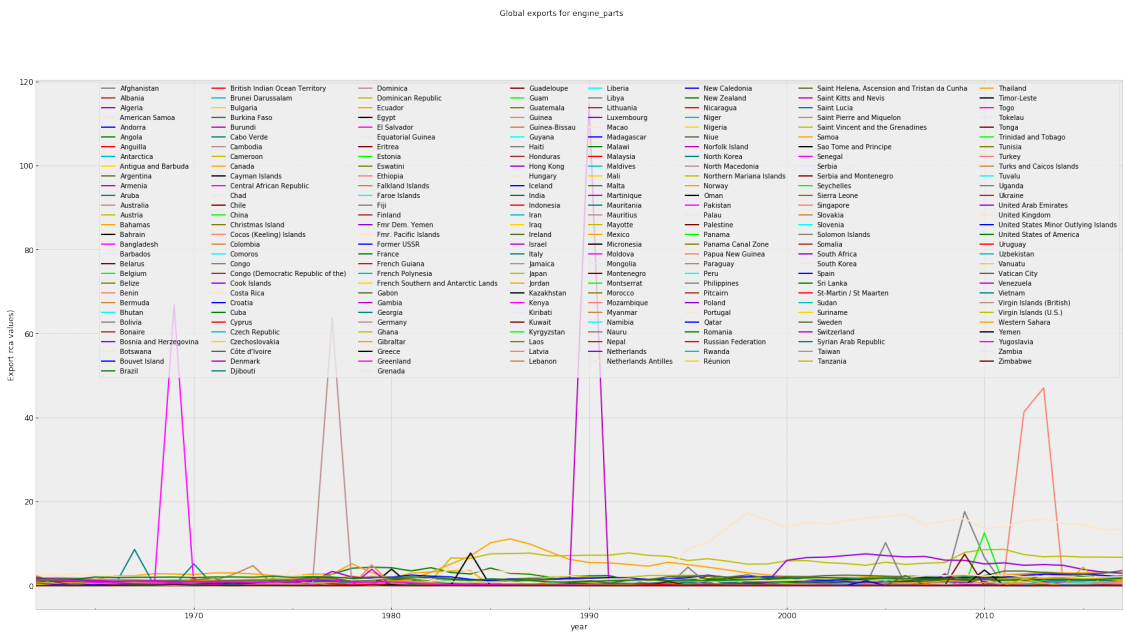
```
0.03923    0.0        0.0  0.120572           0.135594                0.0
0.0   0.0     0.022207       0.0  0.0824  0.072541  0.034016  0.011424
0.0     0.0      0.0                                          0.0
0.0          0.0                   2.183242                                0.0
0.0                    0.0   0.0108     0.0                    0.0          0.0
0.07462   0.265236       0.0       0.0              0.0  1.950224      0.024796
0.157027  0.350309   0.011113                      0.0    0.0  0.016039  1.234533
1.777193                   0.0  0.100061       0.0       0.0          0.0
0.038379      0.0    0.0                  0.0  0.027125     0.0
0.0     0.0     0.0      0.0                    0.0        2.501614
0.0                   1.55642      0.0         0.0  0.280693           0.0
0.0      0.0                          0.0                    0.0             0.0
0.0    1.354641     0.0       0.0
```

Setting index as the datetime column for easier manipulations:

Image storage

[131]:
```
location = os.getcwd()
target = 'images\\{}\\{}'.format(product,experiment)
path = os.path.join(location, target)
```

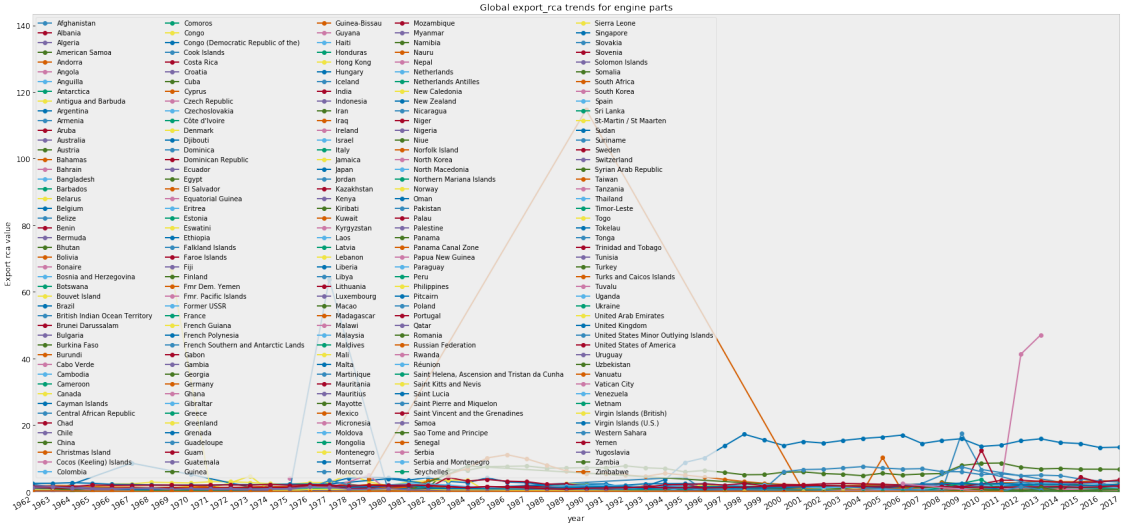
[298]:
```
#df_pivoted = xdf.
 ↪pivot_table(index='year',columns='exporter',values='export_value',aggfunc='sum')
fig1, ax1 = plt.subplots(figsize=(30,15))
#https://matplotlib.org/3.1.0/gallery/color/named_colors.html
colors =␣
 ↪["gray","brown","darkviolet","bisque","b","g","r","c","gold","darkolivegreen","m","teal","r
          "olive","salmon","peru","aqua"]
label='global trends in export rca values for engine parts'
df_grps.plot(ax=ax1,label=label,color=colors)
ax1.set_ylabel('Export rca values)')
fig1.suptitle('Global exports for {}'.format(product))

plt.legend(ncol=8,loc='best')

root = os.path.join(path, 'export rca trends')
plt.savefig(str(root))
#.plot(legend=False)#subplots=True, legend=False)
```



[235]:
```
# Adapted from https://stackoverflow.com/questions/30942755/
 ↪plotting-multiple-time-series-after-a-groupby-in-pandas
# Modified from https://www.programcreek.com/python/example/98021/matplotlib.
 ↪dates.YearLocator
from matplotlib.dates import YearLocator, MonthLocator, DateFormatter

def plot_multiple_time_series(dframe, ts_label, grp_label, values_label,␣
 ↪product,experiment=experiment,figsize=(30,15), title=None):
    '''
    Plots multiple time series on one chart by first grouping time series based␣
 ↪on series names from column in dataframe

    Parameters:
    ----------
    dframe : timeseries Pandas dataframe
    ts_label : string
        The name of the df column that has the datetime timestamp x-axis values.
    grp_label : string
        The column name in dframe for groupby method.
    values_label : string
        The column name in dframe for the y-axis.
    figsize : tuple of two integers
        Figure size of the resulting plot, e.g. (20, 7)
    title : string
        Optional title
    '''
```

```
    xtick_locator = YearLocator()
    xtick_dateformatter = DateFormatter('%Y')
    fig, ax = plt.subplots(figsize=figsize)
    location = os.getcwd()
    target = 'images\\{}\\{}'.format(product,experiment)
    path = os.path.join(location, target)
    for key, grp in dframe.groupby([grp_label]):
        ax = grp.plot(ax=ax, kind='line', x=ts_label, y=values_label,␣
↪label=key, marker='o')
    ax.xaxis.set_major_locator(xtick_locator)
    ax.xaxis.set_major_formatter(xtick_dateformatter)
    ax.autoscale_view()
    ax.legend(ncol=5, loc='upper left')
    #_ = plt.xticks(rotation=0, )
    _ = plt.grid()
    _ = plt.xlabel('year')
    _ = plt.ylabel('Export rca value')
    _ = plt.ylim(0, dframe[values_label].max() * 1.25)
    if title is not None:
        _ = plt.title(title)
    savefile = os.path.join(path,label)
    plt.savefig(str(savefile))
    _ = plt.show()
```

[236]:
```
plot_multiple_time_series(xdf, 'year', 'exporter', 'rca',␣
↪'engine_parts',experiment,title='Global export_rca trends for engine parts')
```



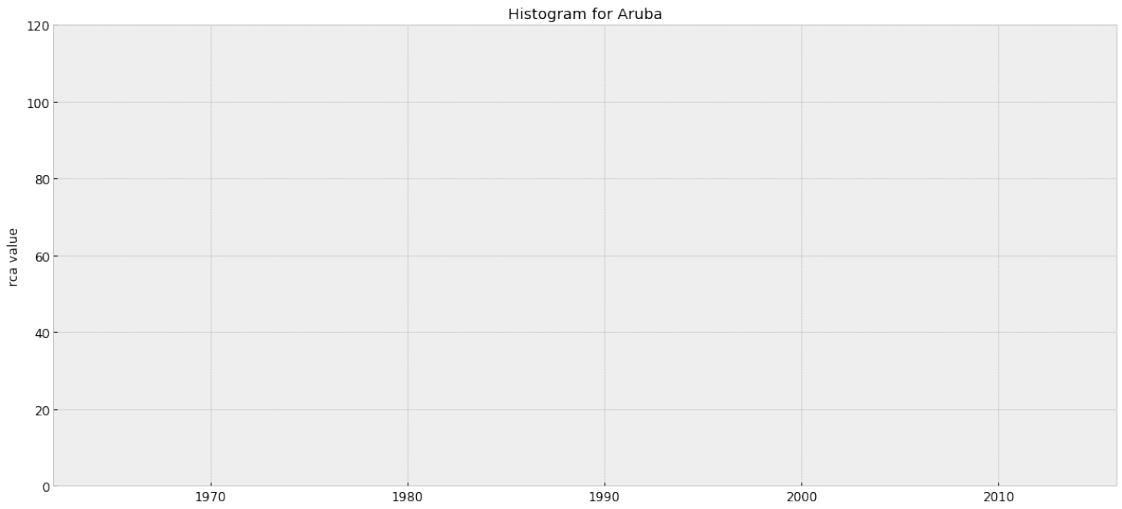
Reviewing plots of the density of observations can provide further insight into the structure of the data: - Is the distribution perfectly Gaussian (normal distribution)? - Skewness: which direction is distribution? - If not Gaussian and skew, transformations will be necessary prior

to modelling

[356]:
```
for exporter in exporters:
    root = save_images(df_grps,'rca_histograms')
    plt.figure(1);
    plt.subplot(111)
    plt.title('Histogram for {}'.format(exporter))
    #sns.distplot(df_grps[exporter].values().index.year)
    plt.hist(df_grps[exporter])
    plt.ylabel('rca value')
    plt.xlim(1962,2016)
    plt.ylim(0,120)

    plt.subplot(112)
    plt.title('Density plot for {}'.format(exporter))
    df_grps[exporter].plot(kind='kde')
    plt.ylabel('Density')


    display(plt.savefig(str(root)))
    plt.show()
```

```
␣
↪---------------------------------------------------------------------------

ValueError                                Traceback (most recent call␣
↪last)

<ipython-input-356-4cd825fc3b19> in <module>
 10     plt.ylim(0,120)
 11
---> 12     plt.subplot(112)
 13     plt.title('Density plot for {}'.format(exporter))
 14     df_grps[exporter].plot(kind='kde')


~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in subplot(*args,␣
↪**kwargs)
1082
1083     fig = gcf()
-> 1084     a = fig.add_subplot(*args, **kwargs)
1085     bbox = a.bbox
1086     byebye = []


~\Anaconda3\lib\site-packages\matplotlib\figure.py in add_subplot(self,␣
↪*args, **kwargs)
```

```
      1365                     self._axstack.remove(ax)
      1366
   -> 1367             a = subplot_class_factory(projection_class)(self, *args,␣
↪**kwargs)
      1368         self._axstack.add(key, a)
      1369         self.sca(a)


       ~\Anaconda3\lib\site-packages\matplotlib\axes\_subplots.py in␣
↪__init__(self, fig, *args, **kwargs)
        58                     raise ValueError(
        59                         ("num must be 1 <= num <= {maxn}, not {num}"
   ---> 60                         ).format(maxn=rows*cols, num=num))
        61                 self._subplotspec = GridSpec(
        62                         rows, cols, figure=self.figure)[int(num) - 1]


       ValueError: num must be 1 <= num <= 1, not 2
```



Box and Whisker Plots:

No high resolution in dataset to check: - what time series have similar median values across different years - whether there is a steady increase in the spread, or middle 50% of the data (boxes) over time - best model if considering seasonality

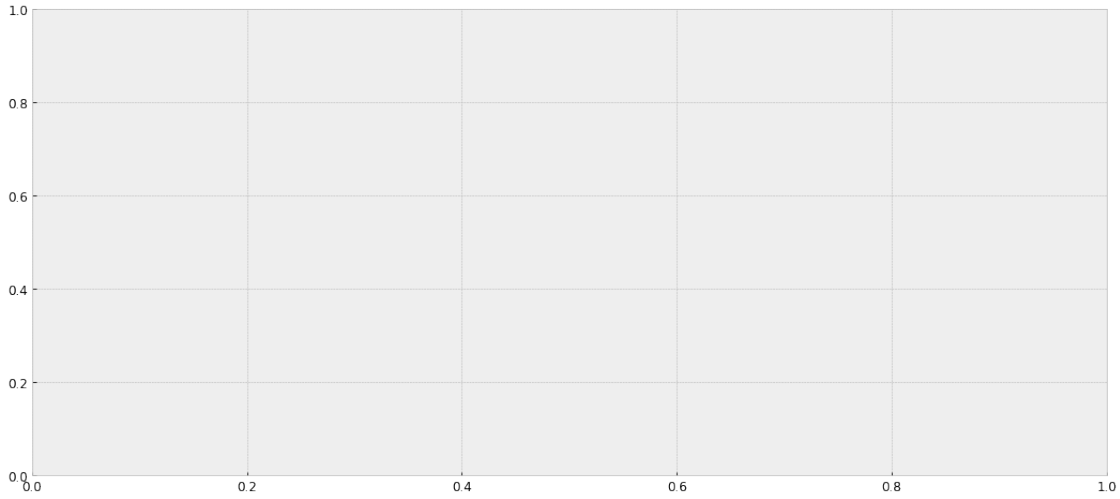
[276]:
```
df_grps.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 56 entries, 1962-01-01 to 2017-01-01
Columns: 235 entries, Afghanistan to Zimbabwe
```

```
dtypes: float64(235)
memory usage: 103.2 KB
```

[366]:
```
#plt.subplots(figsize=(15,6))
sns.distplot(xdf)
#savefile = os.path.join(path,'global trend')
#plt.savefig(savefile)
plt.show()
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-366-673e01afd3b9> in <module>
  1 #plt.subplots(figsize=(15,6))
----> 2 sns.distplot(xdf)
  3 #savefile = os.path.join(path,'global trend')
  4 #plt.savefig(savefile)
  5 plt.show()


~\Anaconda3\lib\site-packages\seaborn\distributions.py in distplot(a,␣
↪bins, hist, kde, rug, fit, hist_kws, kde_kws, rug_kws, fit_kws, color,␣
↪vertical, norm_hist, axlabel, label, ax)
  196             line, = ax.plot(0, a.mean())
  197         else:
--> 198             line, = ax.plot(a.mean(), 0)
  199         color = line.get_color()
  200         line.remove()


~\Anaconda3\lib\site-packages\numpy\core\_methods.py in _mean(a, axis,␣
↪dtype, out, keepdims)
   73             is_float16_result = True
   74
---> 75     ret = umr_sum(arr, axis, dtype, out, keepdims)
   76     if isinstance(ret, mu.ndarray):
   77         ret = um.true_divide(


TypeError: unsupported operand type(s) for +: 'Timestamp' and 'str'
```



[59]:
```
countries = df_grps.columns
```

Target folder to store individual images

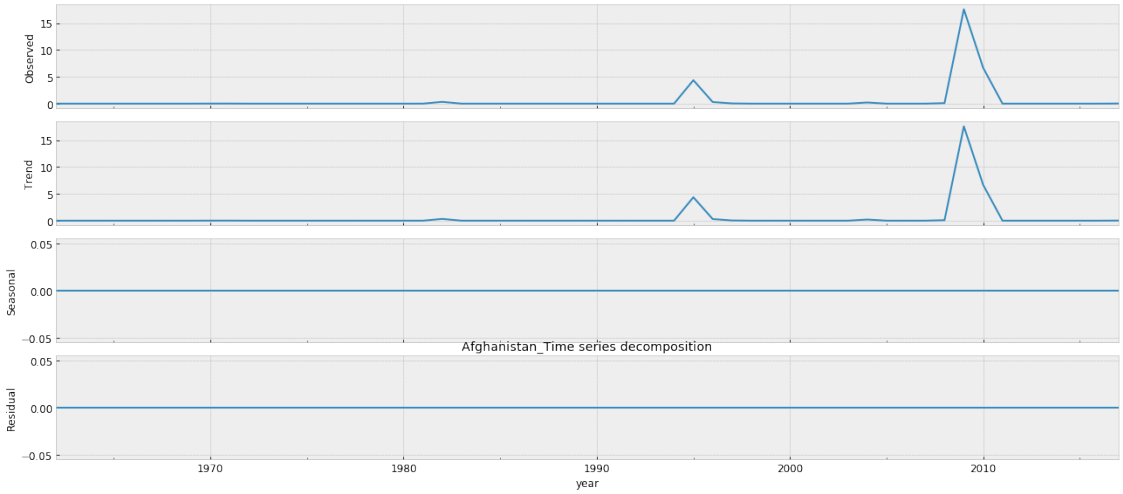
[197]:
```
def save_images(ts_groups,analysis):
    '''
    args: can be from these options␣
↪['decompose',acf_pacf','adf_test','rolling_stats']
    '''
    path = os.getcwd()+'\\images\\{}\\{}\\{}'.
↪format(product,experiment,analysis)
    savefile = os.path.join(path, ts_groups[country].name)
    return savefile
```

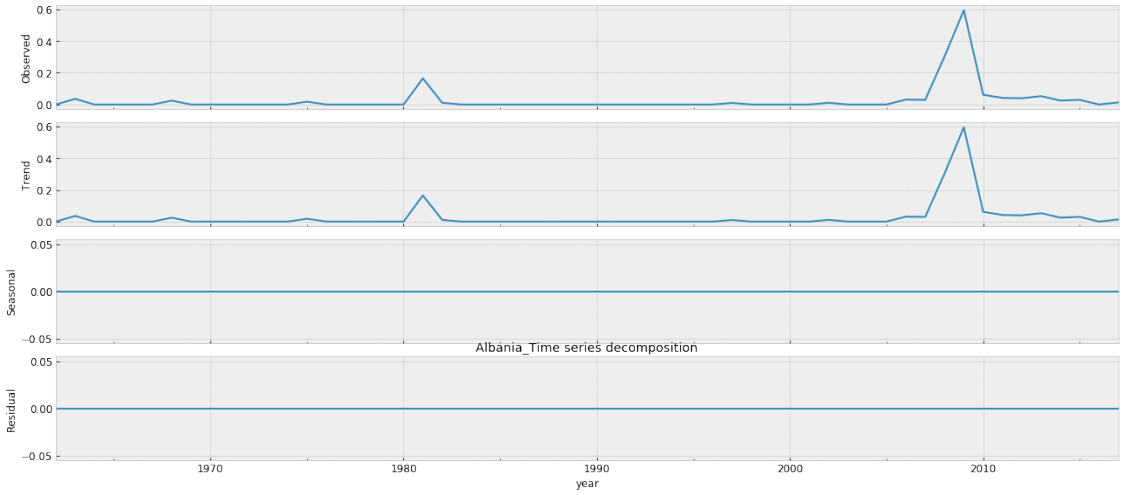
Decomposing using statsmodel: - We can use statsmodels to perform a decomposition of this time series. - The decomposition of time series is a statistical task that deconstructs a time series into several components, each representing one of the underlying categories of patterns. - With statsmodels we will be able to see the trend, seasonal, and residual components of our data.

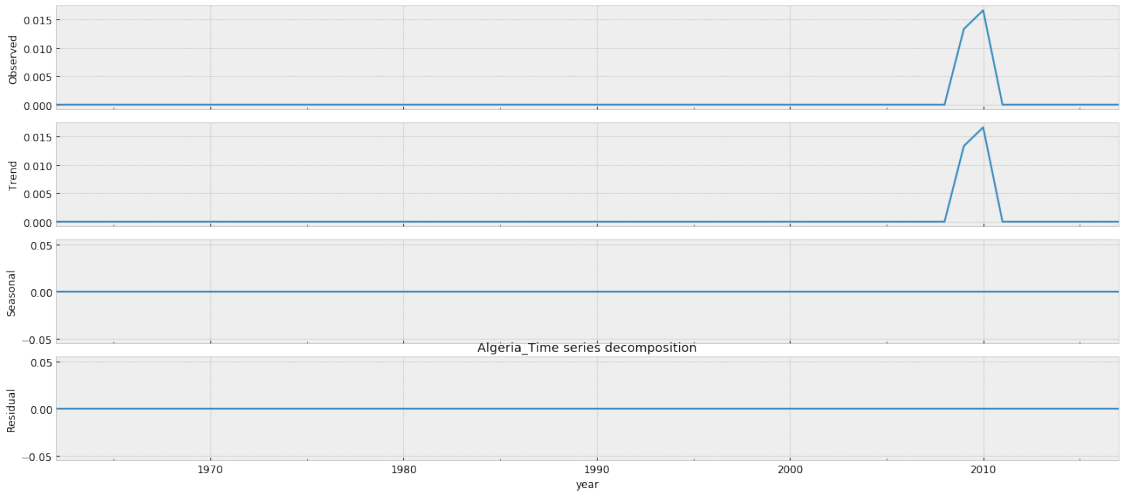
Decomposition

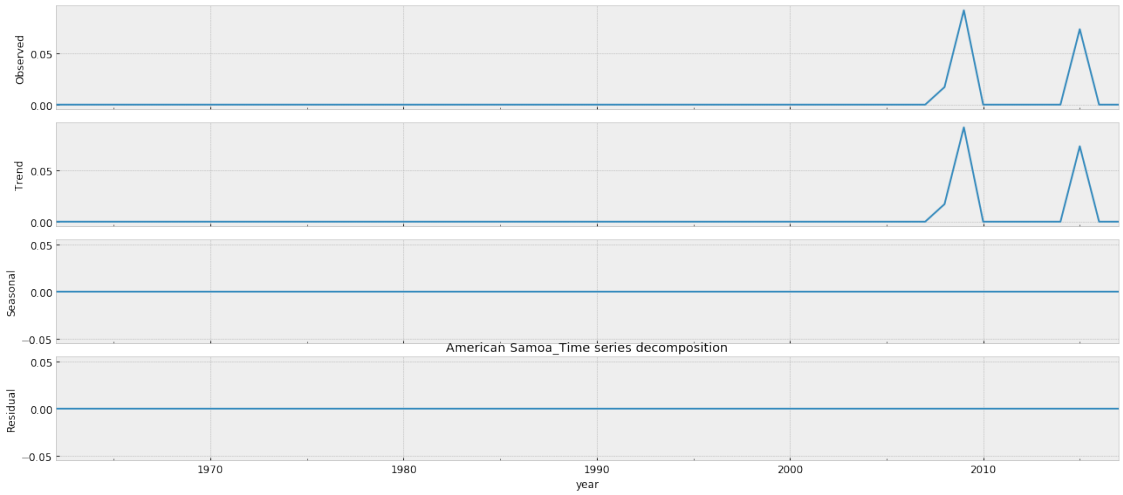
[203]:
```
for country in countries:
    analysis = 'decomposition'
    savefile = save_images(df_grps,analysis)
    decomposition = sm.tsa.seasonal_decompose(df_grps[country],␣
↪model='additive')
    rcParams['figure.figsize'] = 18, 8
    fig = decomposition.plot()

    plt.title('{}_Time series decomposition'.format(df_grps[country].name))
    plt.savefig(str('{}_{}'.format(savefile,analysis)))
```

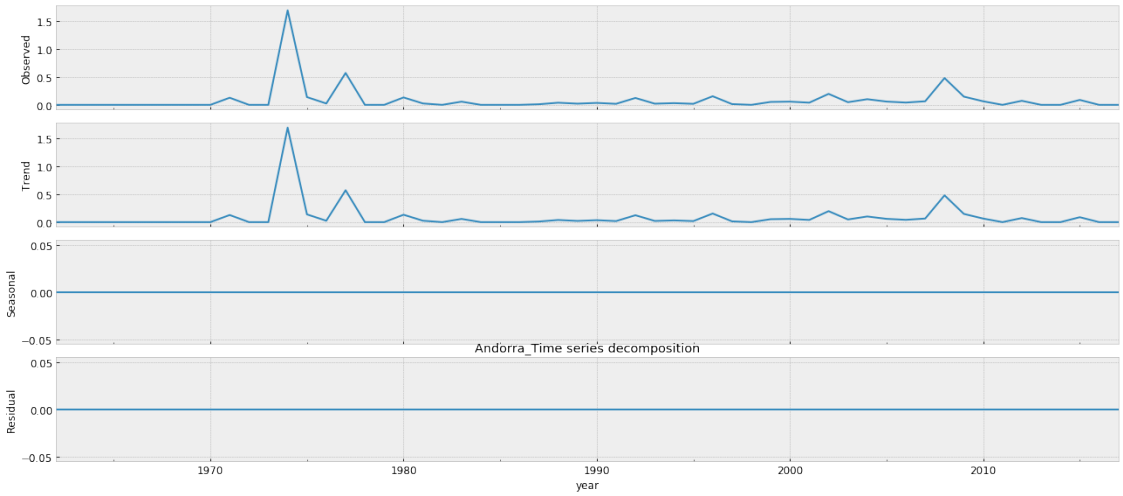
```
plt.show()
```

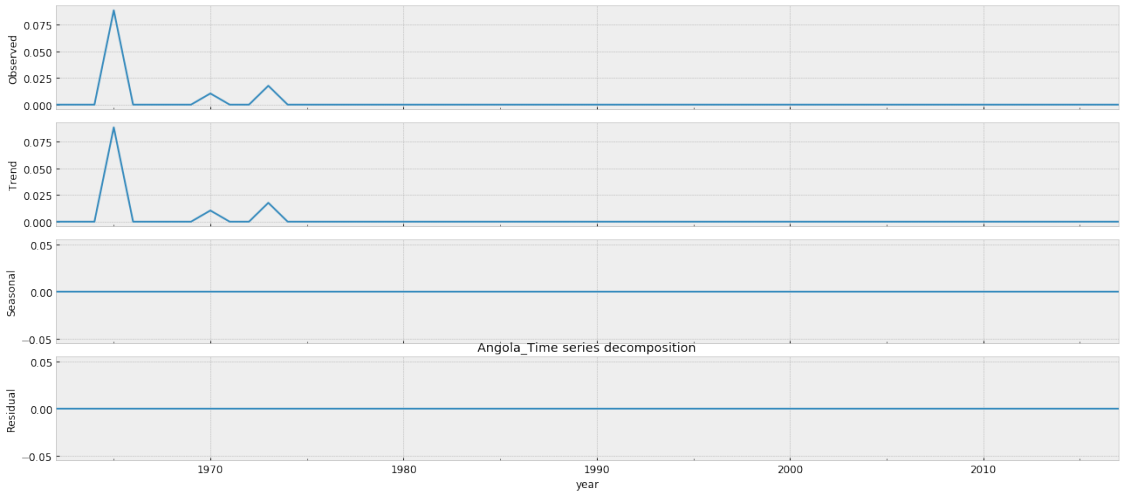

Observed
Trend
Seasonal
Residual
Afghanistan_Time series decomposition
1970
1980
1990
2000
2010
year

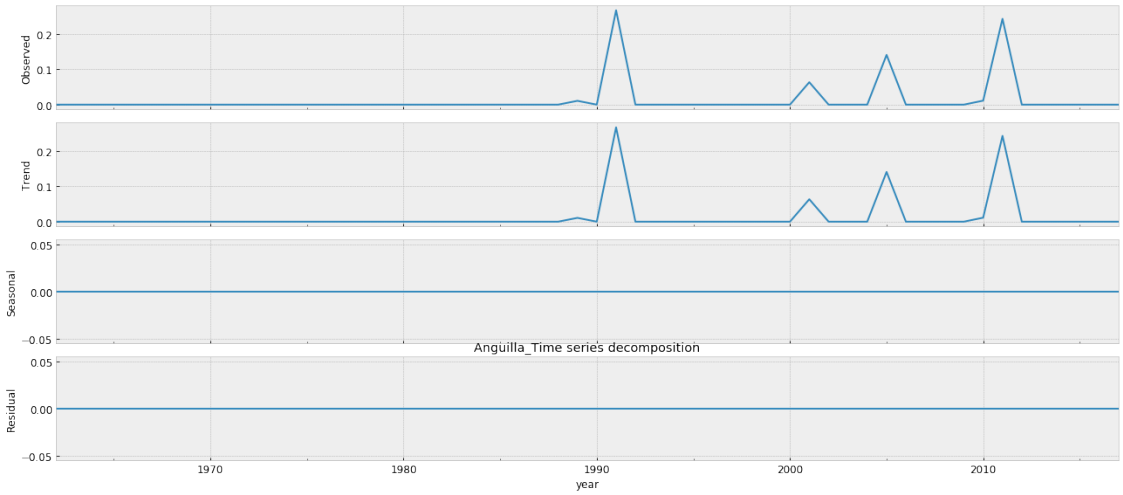

Observed
Trend
Seasonal
Residual
Albania_Time series decomposition
1970
1980
1990
2000
2010
year

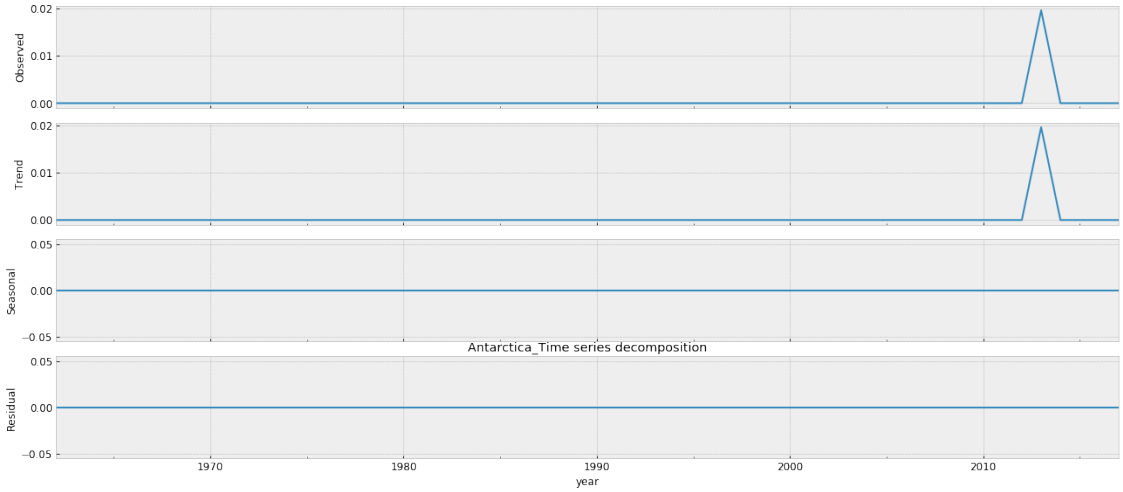

Observed
Trend
Seasonal
Residual
Algeria_Time series decomposition
year

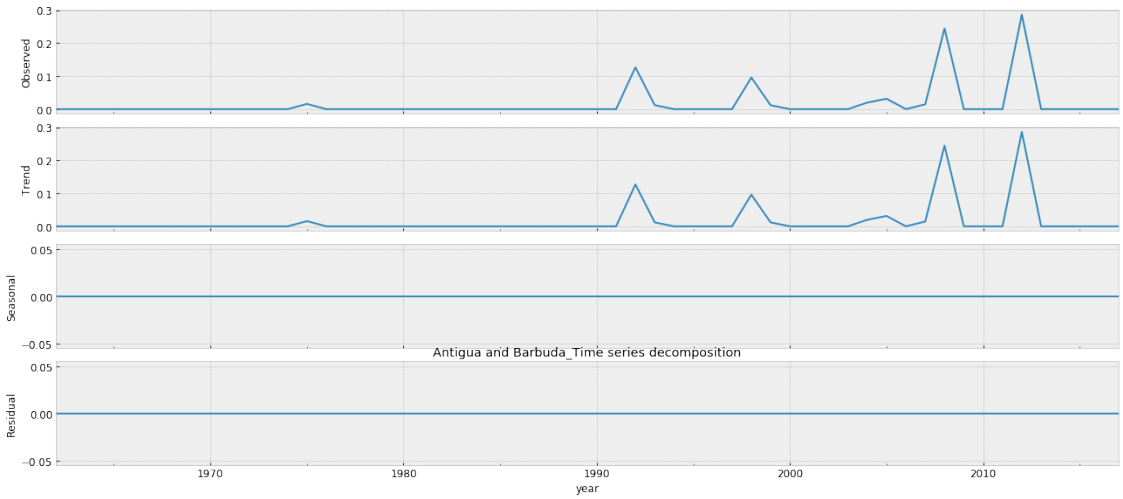

Observed
Trend
Seasonal
Residual
American Samoa_Time series decomposition
year

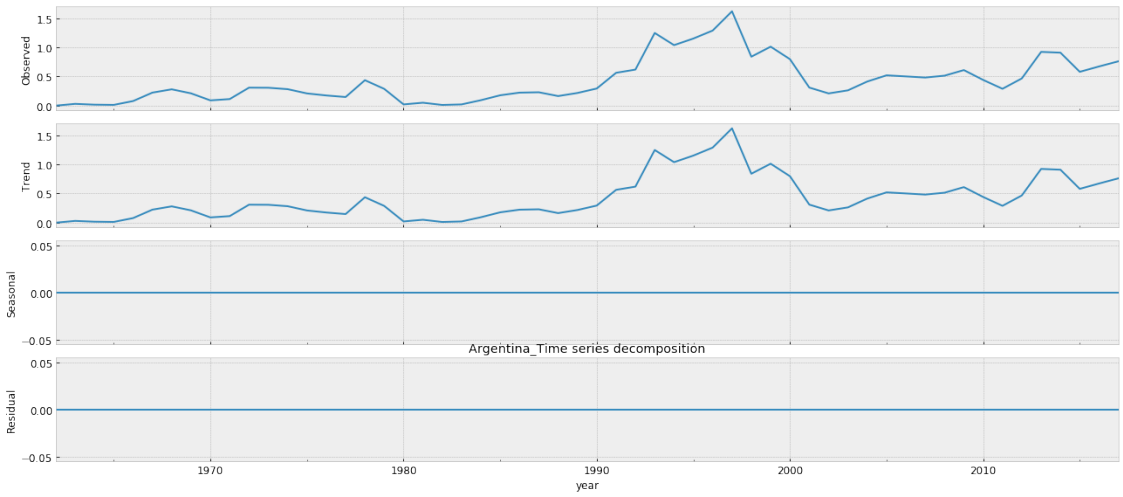

Observed
Trend
Seasonal
Residual
Andorra_Time series decomposition
year

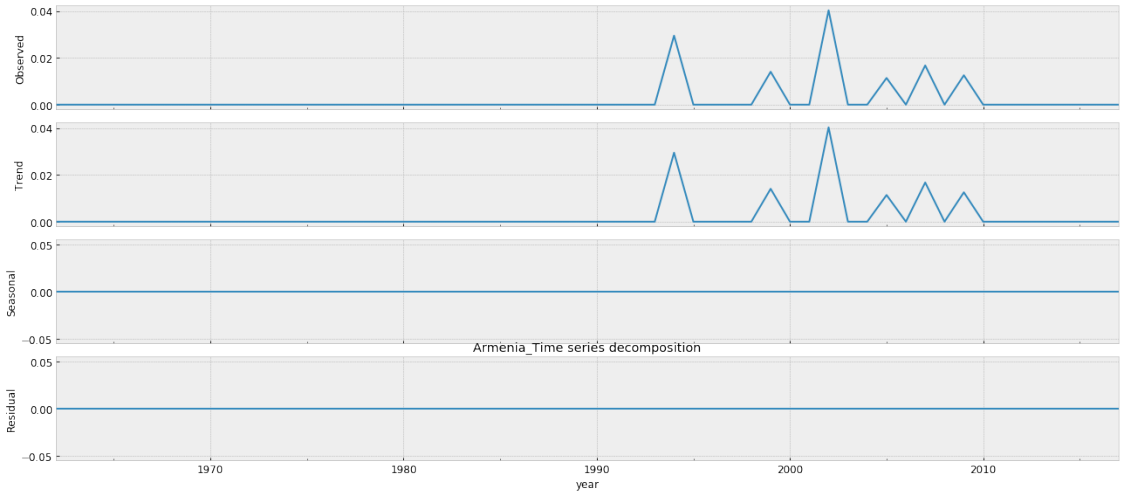

Observed
Trend
Seasonal
Residual
Angola_Time series decomposition
1970
1980
1990
2000
2010
year

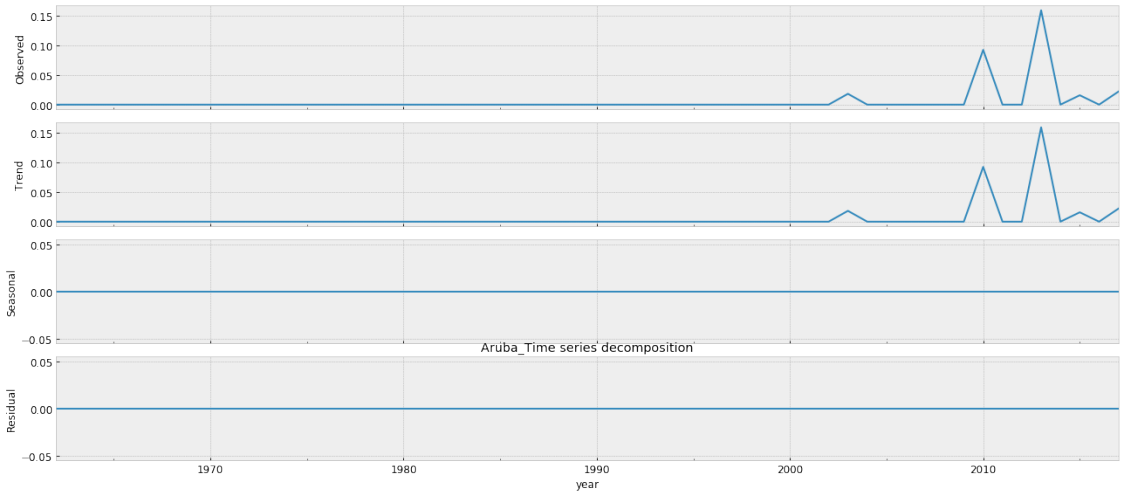

Observed
Trend
Seasonal
Residual
Anguilla_Time series decomposition
1970
1980
1990
2000
2010
year

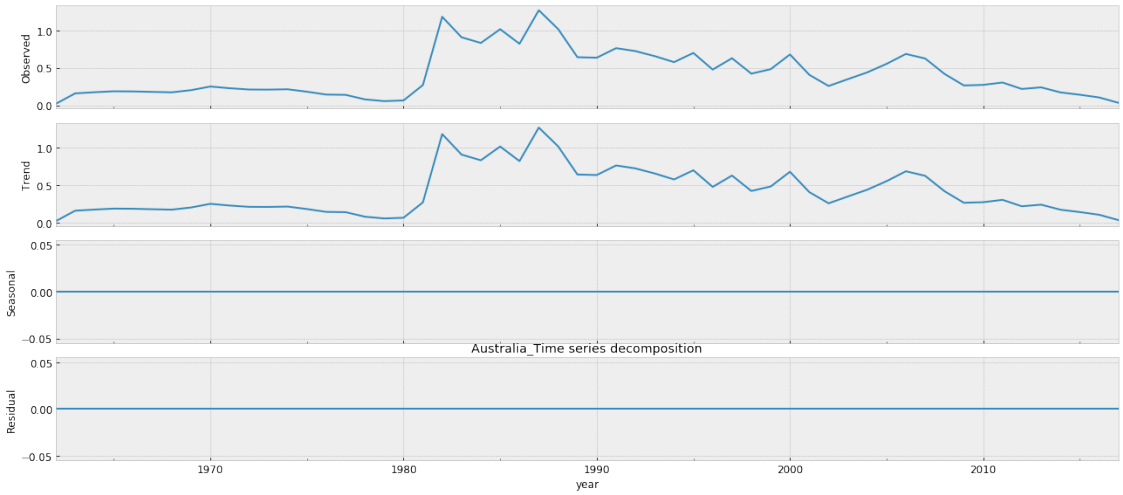

Observed
Trend
Seasonal
Residual
Antarctica_Time series decomposition
year

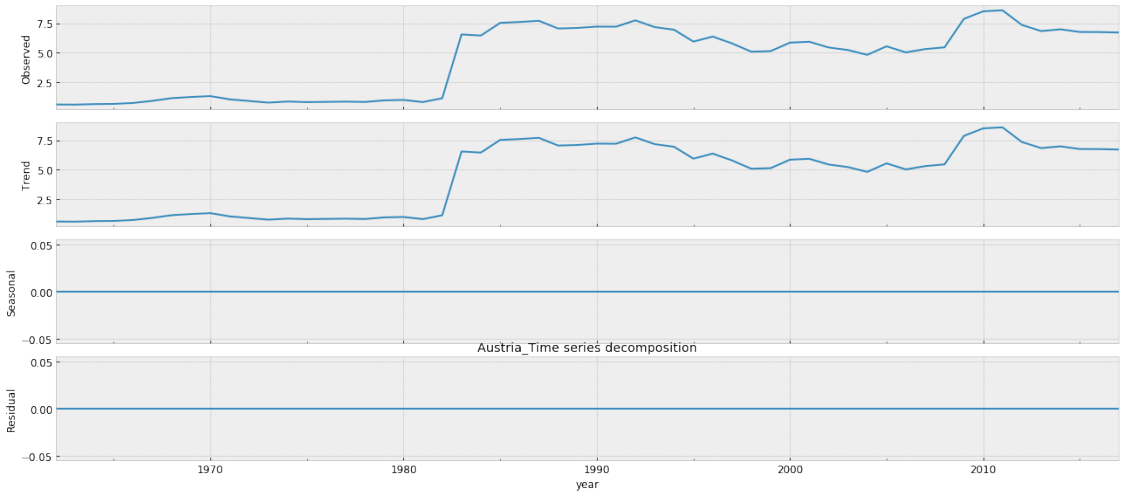

Observed
Trend
Seasonal
Residual
Antigua and Barbuda_Time series decomposition
year

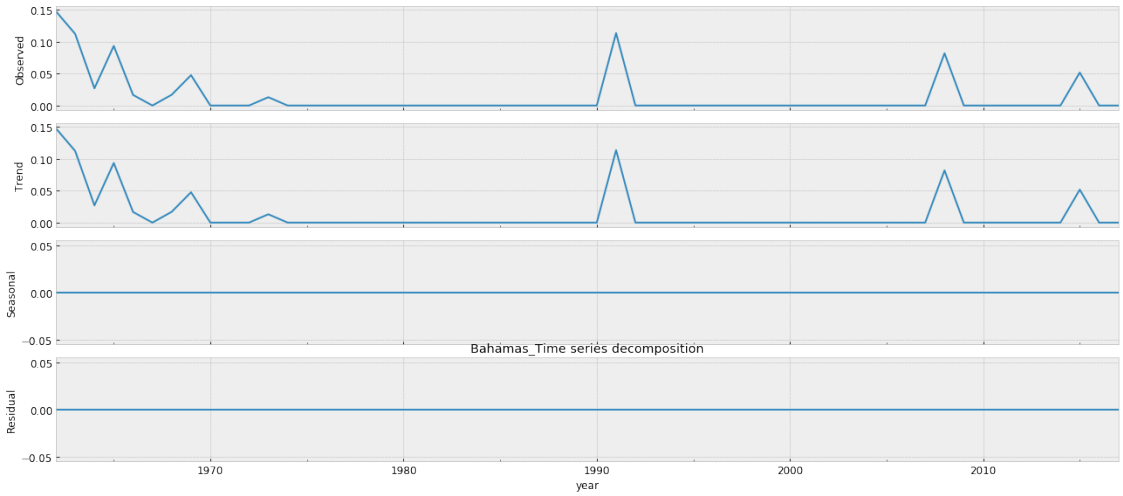

Observed
Trend
Seasonal
Residual
Argentina_Time series decomposition
year

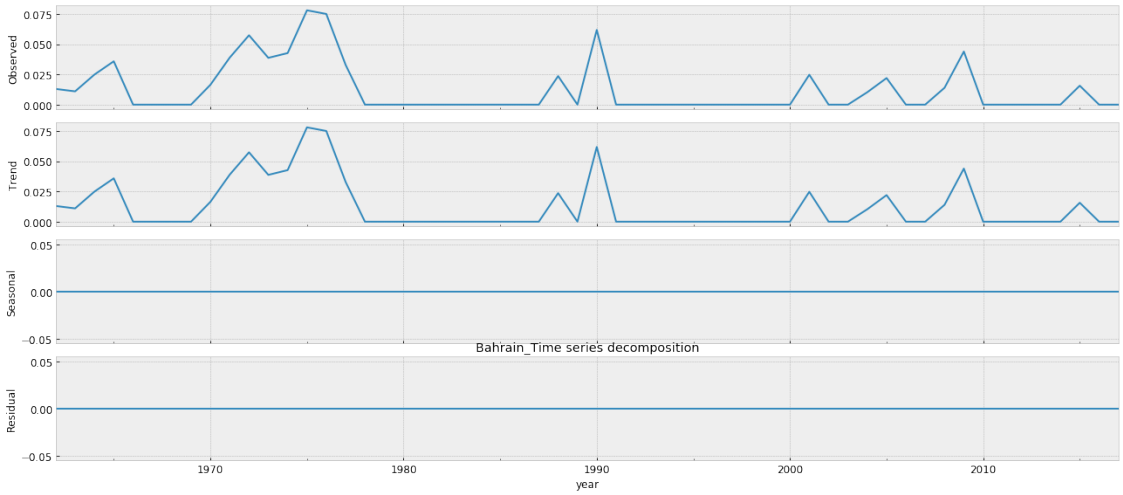

Observed
Trend
Seasonal
Residual
Armenia_Time series decomposition
1970
1980
1990
2000
2010
year

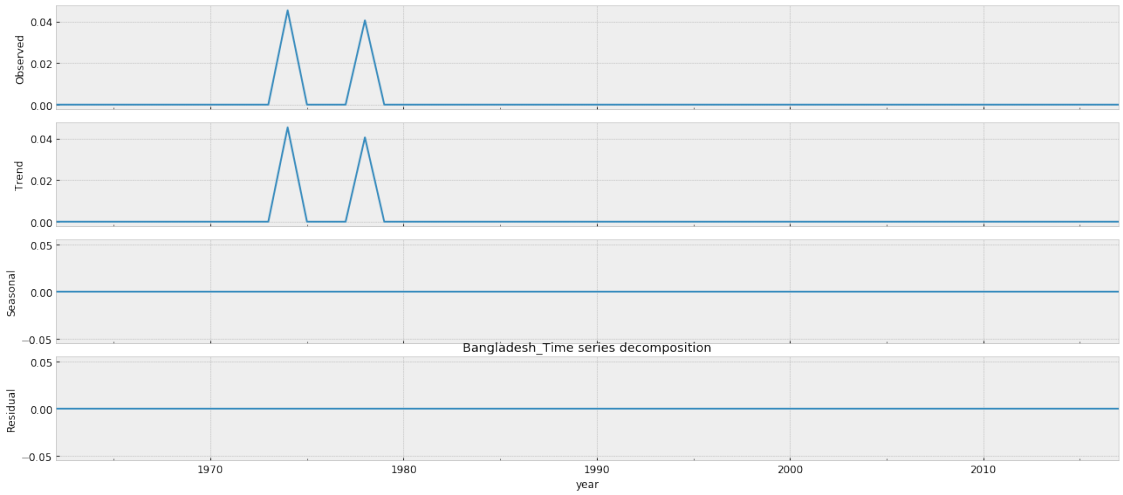

Observed
Trend
Seasonal
Residual
Aruba_Time series decomposition
1970
1980
1990
2000
2010
year

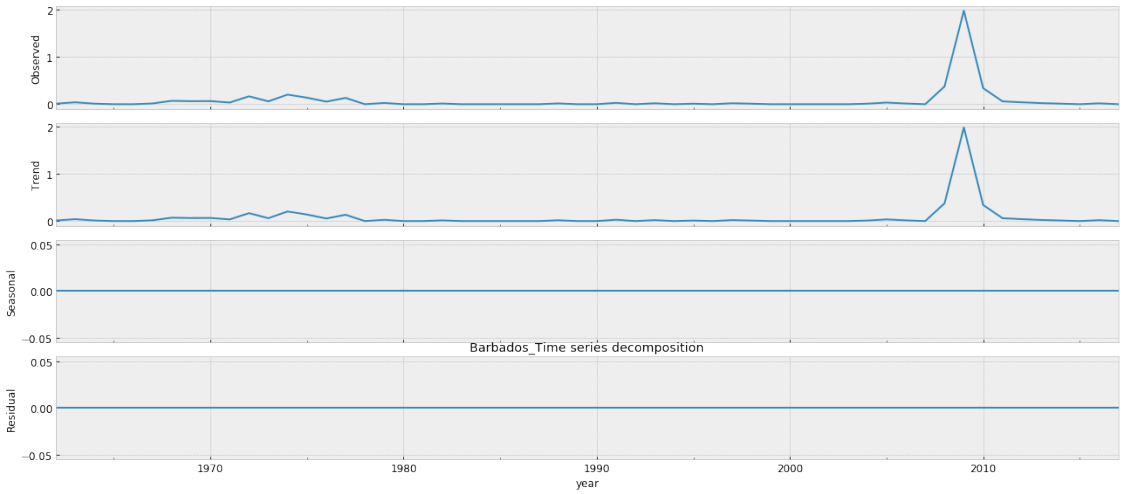

Observed
Trend
Seasonal
Residual
Australia_Time series decomposition
year

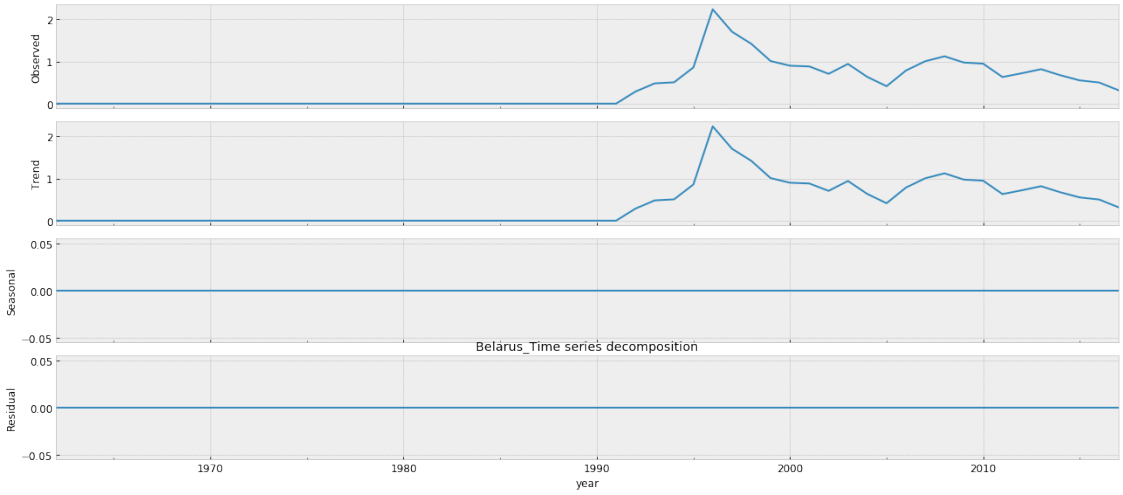

Observed
Trend
Seasonal
Residual
Austria_Time series decomposition
year

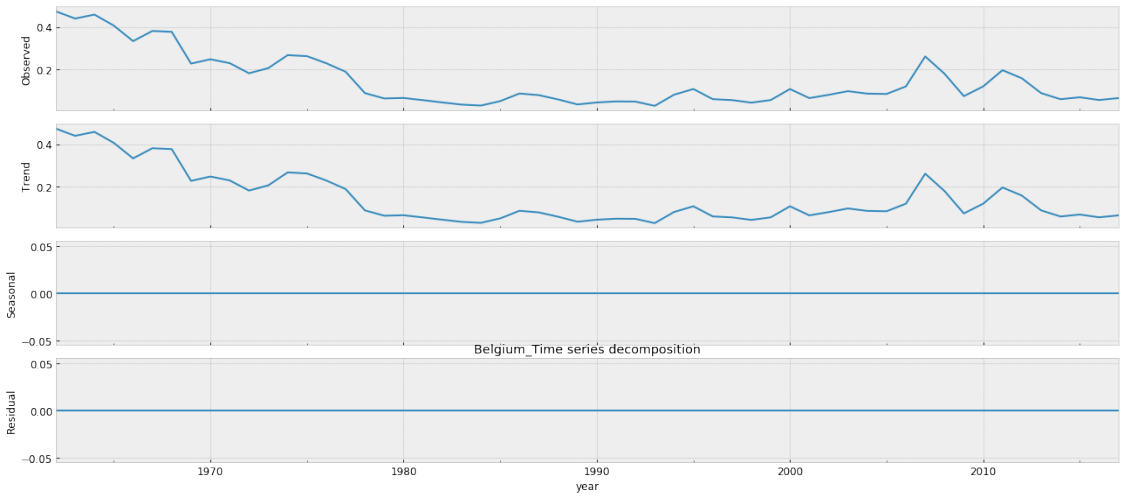

Observed
Trend
Seasonal
Residual
Bahamas_Time series decomposition
year

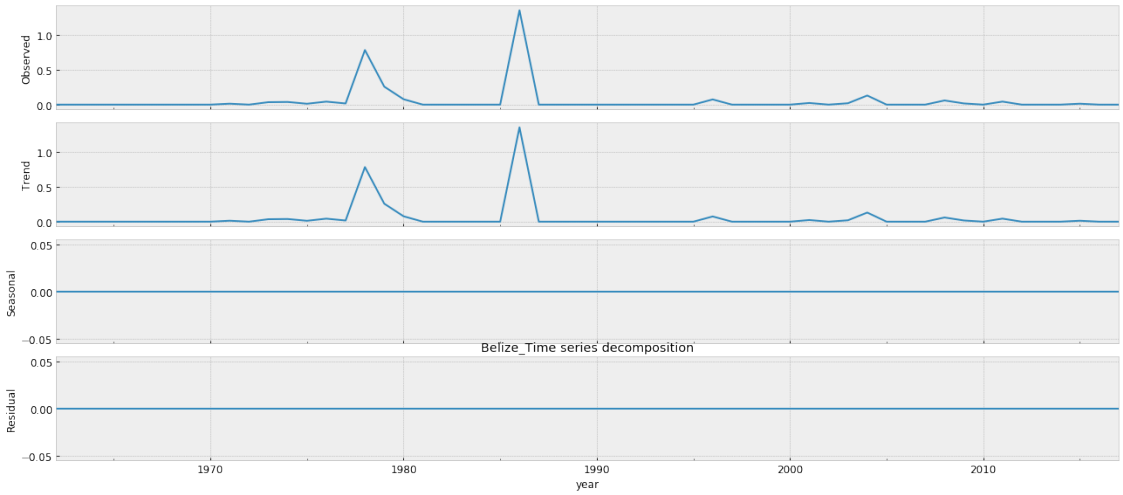

Observed
Trend
Seasonal
Residual
Bahrain_Time series decomposition
1970
1980
1990
2000
2010
year

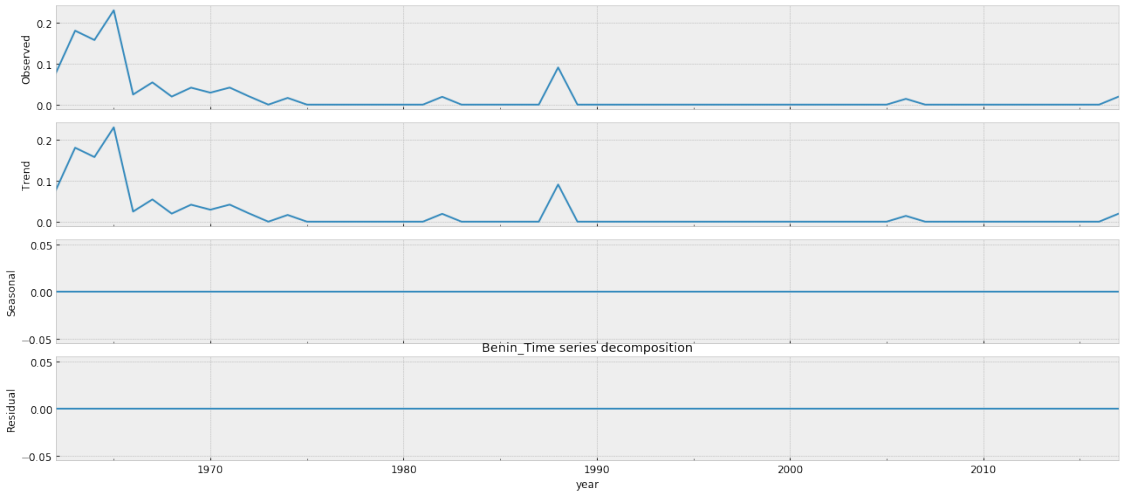

Observed
Trend
Seasonal
Residual
Bangladesh_Time series decomposition
1970
1980
1990
2000
2010
year

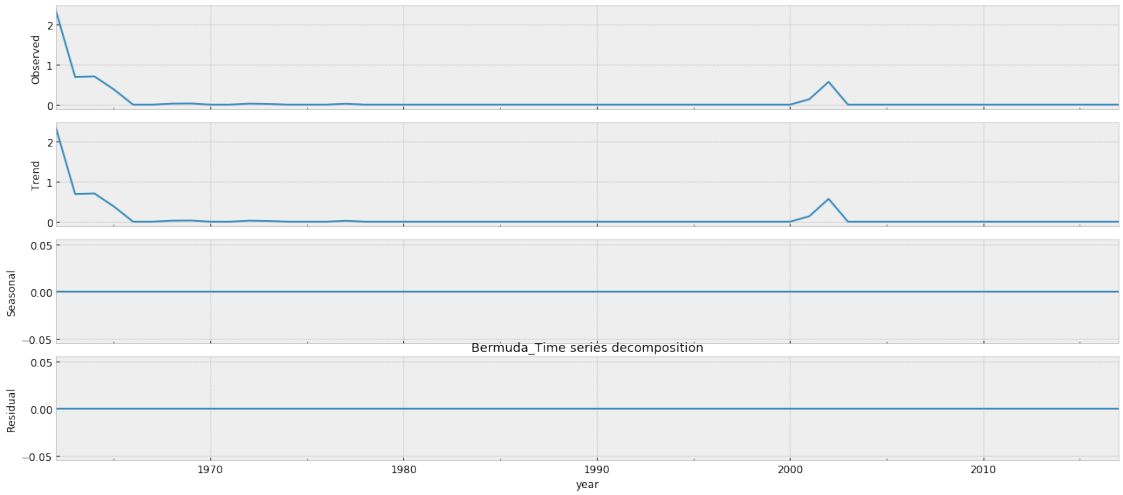

Observed
Trend
Seasonal
Residual
Barbados_Time series decomposition
1970
1980
1990
2000
2010
year

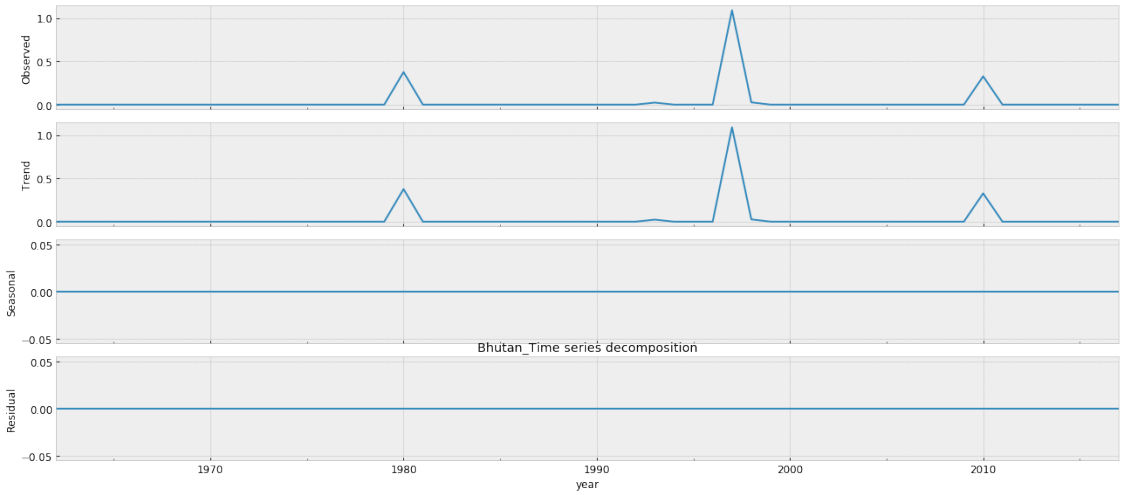

Observed
Trend
Seasonal
Residual
Belarus_Time series decomposition
1970
1980
1990
2000
2010
year

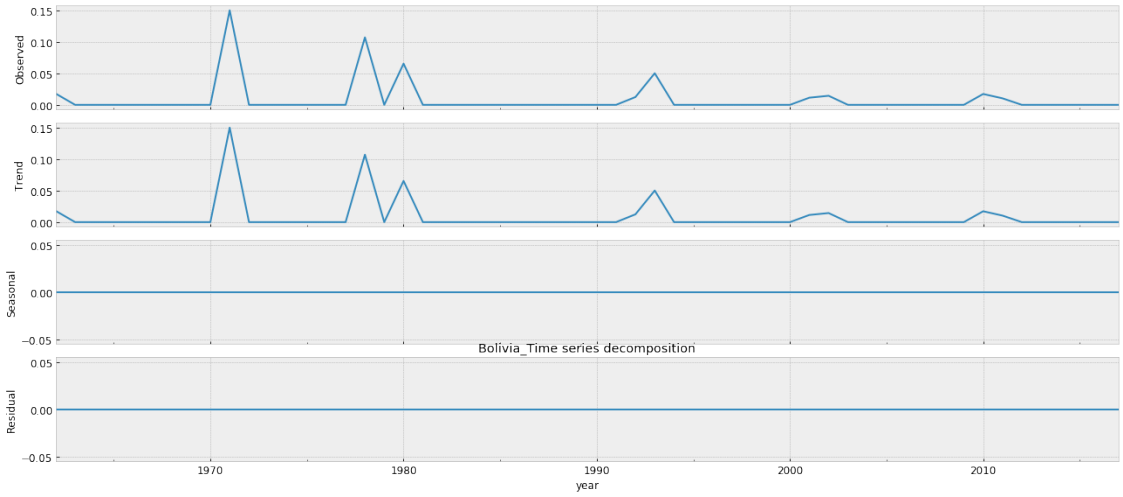

Observed
Trend
Seasonal
Residual
Belgium_Time series decomposition
1970
1980
1990
2000
2010
year

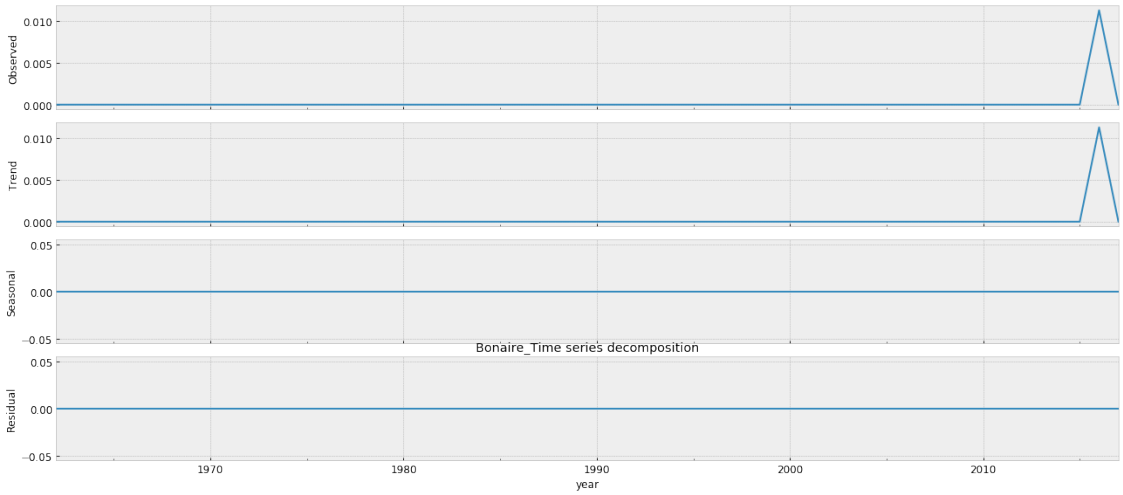

Observed
Trend
Seasonal
Residual
Belize_Time series decomposition
1970
1980
1990
2000
2010
year

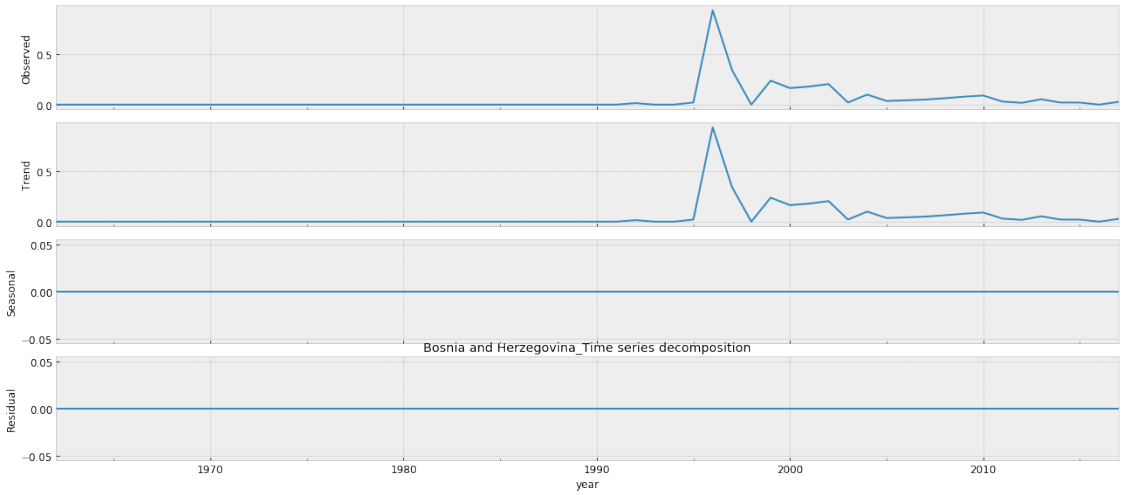

Observed
Trend
Seasonal
Residual
Benin_Time series decomposition
1970
1980
1990
2000
2010
year

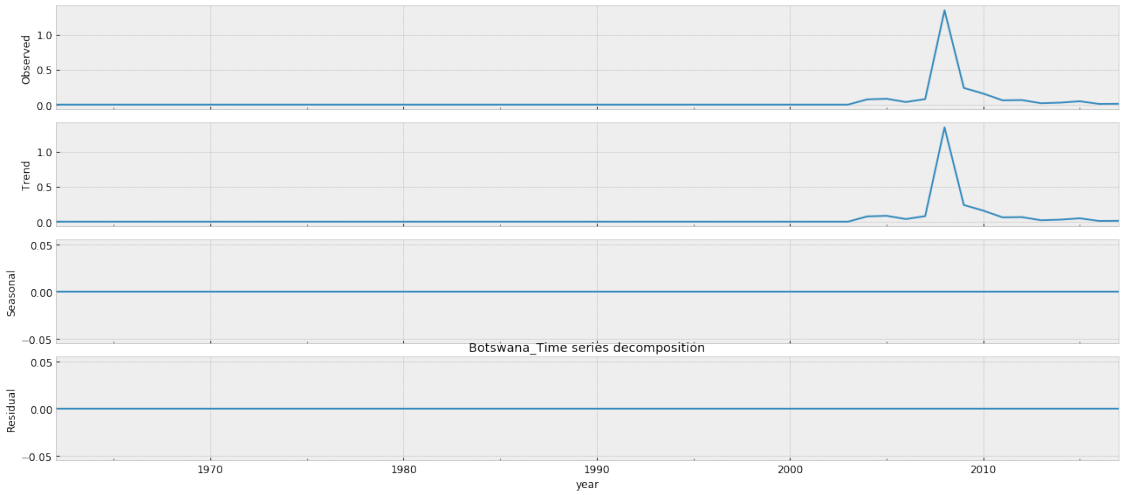

Observed
Trend
Seasonal
Residual
Bermuda_Time series decomposition
1970
1980
1990
2000
2010
year

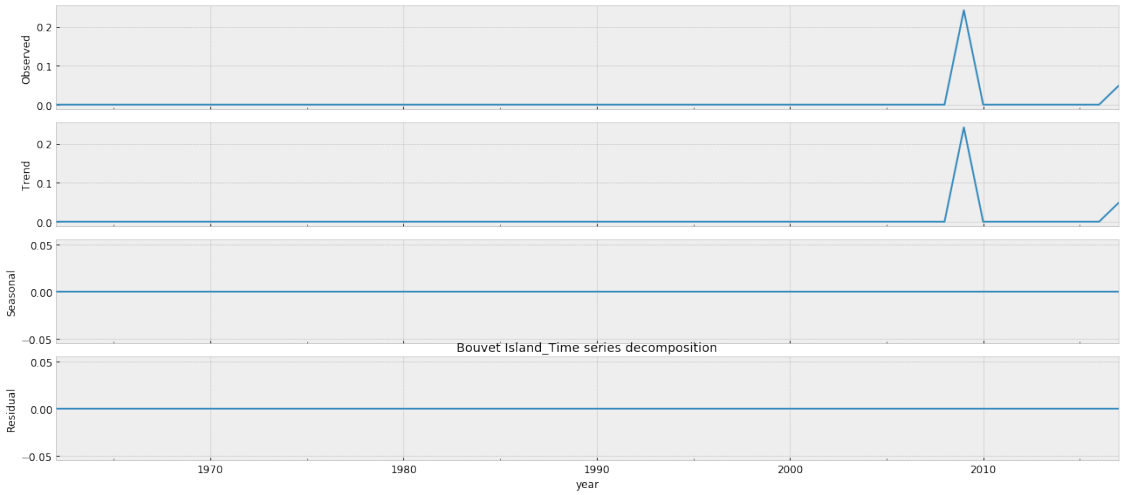

Observed
Trend
Seasonal
Residual
Bhutan_Time series decomposition
1970
1980
1990
2000
2010
year

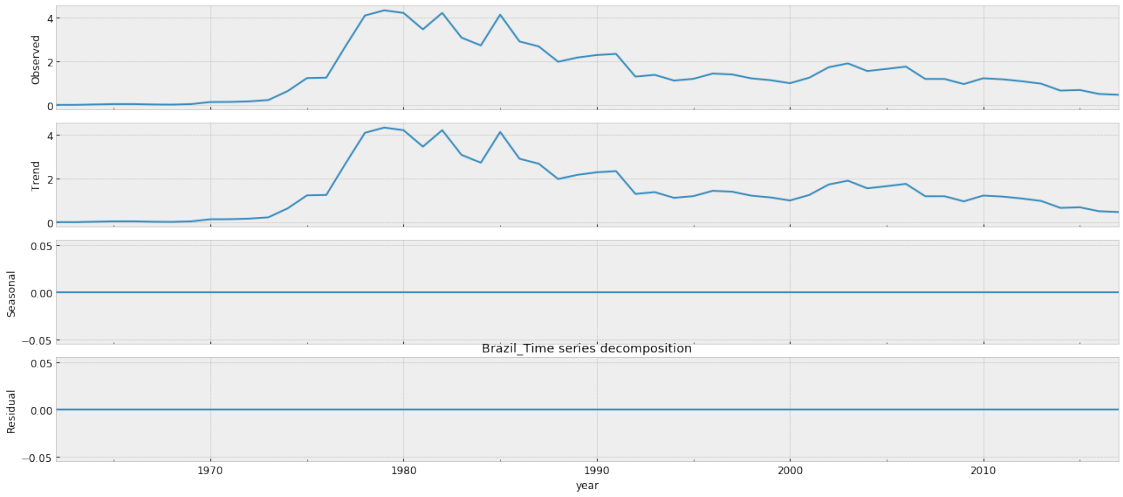

Observed
Trend
Seasonal
Residual
Bolivia_Time series decomposition
1970
1980
1990
2000
2010
year

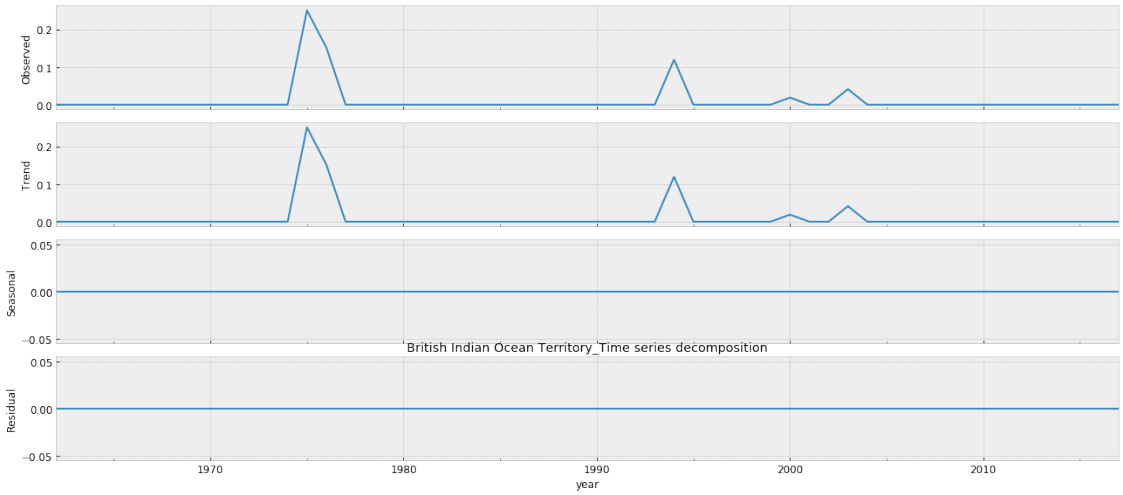

Observed
0.010
0.005
0.000
Trend
0.010
0.005
0.000
Seasonal
0.05
0.00
−0.05
Bonaire_Time series decomposition
Residual
0.05
0.00
−0.05
1970
1980
1990
2000
2010
year

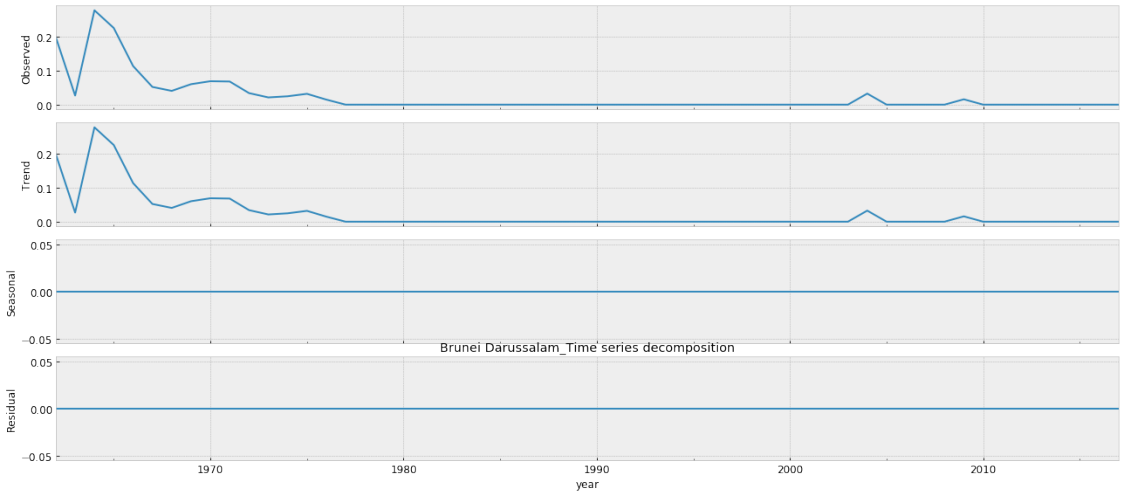

Observed
0.5
0.0
Trend
0.5
0.0
Seasonal
0.05
0.00
−0.05
Bosnia and Herzegovina_Time series decomposition
Residual
0.05
0.00
−0.05
1970
1980
1990
2000
2010
year

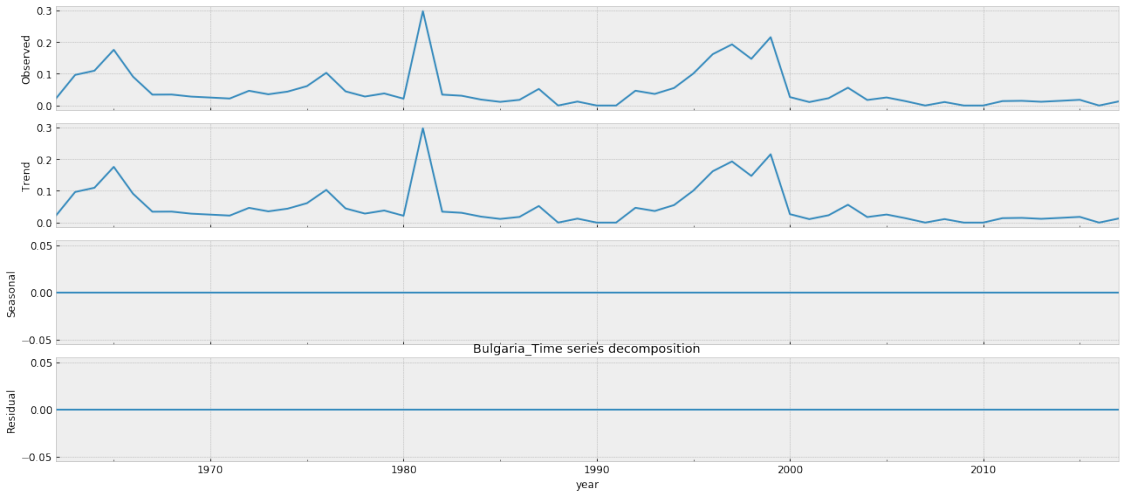

Observed
Trend
Seasonal
Botswana_Time series decomposition
Residual
1970
1980
1990
2000
2010
year

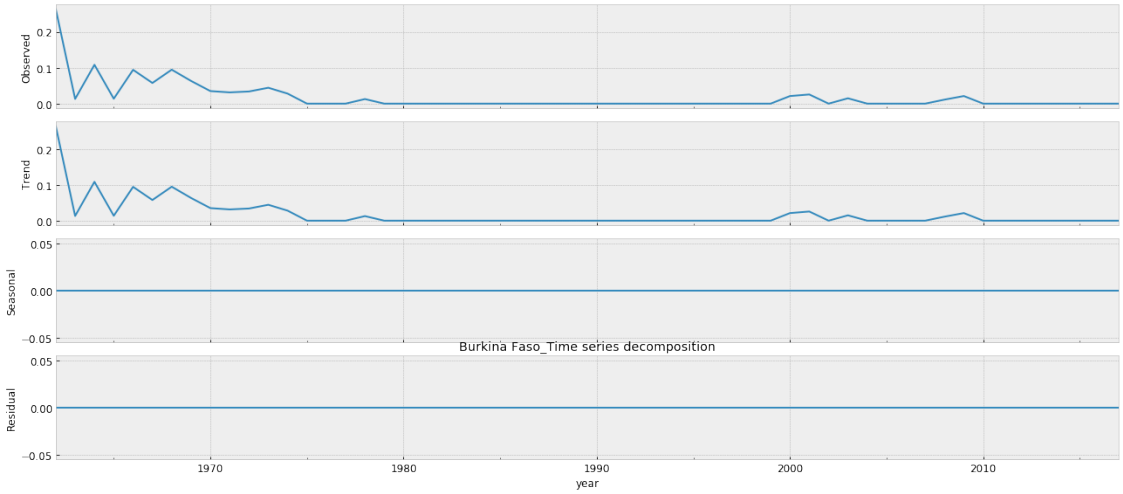

Observed
Trend
Seasonal
Bouvet Island_Time series decomposition
Residual
1970
1980
1990
2000
2010
year

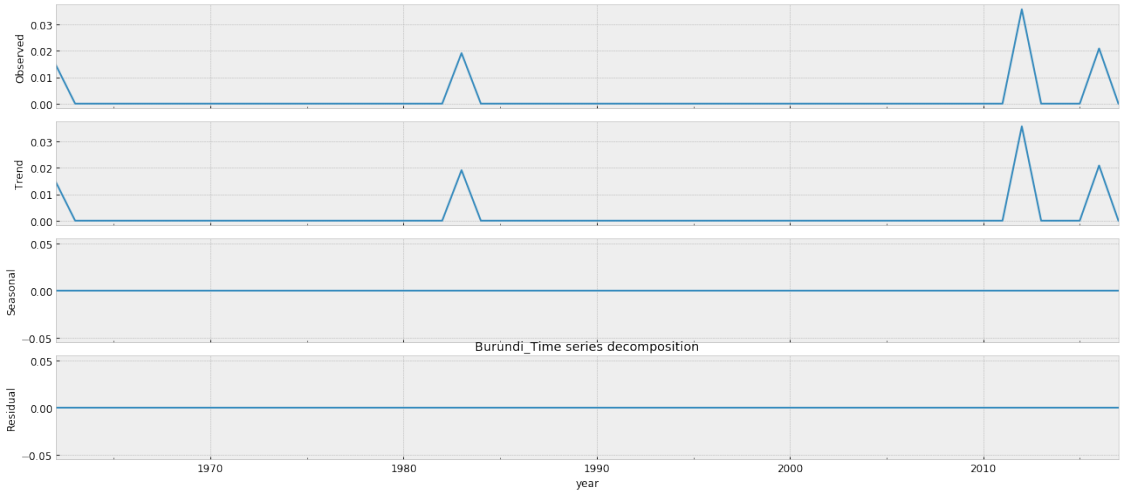

Observed
Trend
Seasonal
Brazil_Time series decomposition
Residual
1970
1980
1990
2000
2010
year

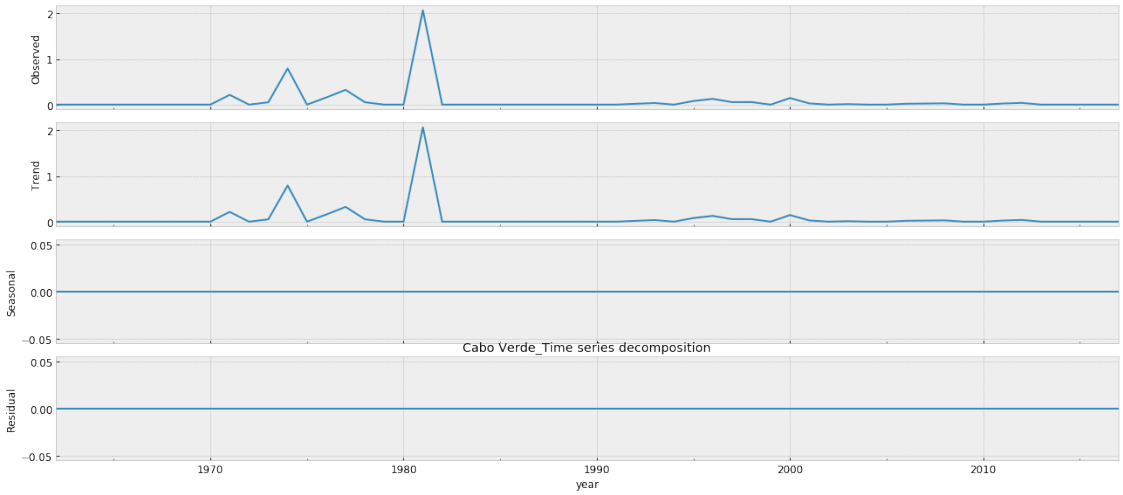

Observed
Trend
Seasonal
Residual
British Indian Ocean Territory_Time series decomposition
1970
1980
1990
2000
2010
year

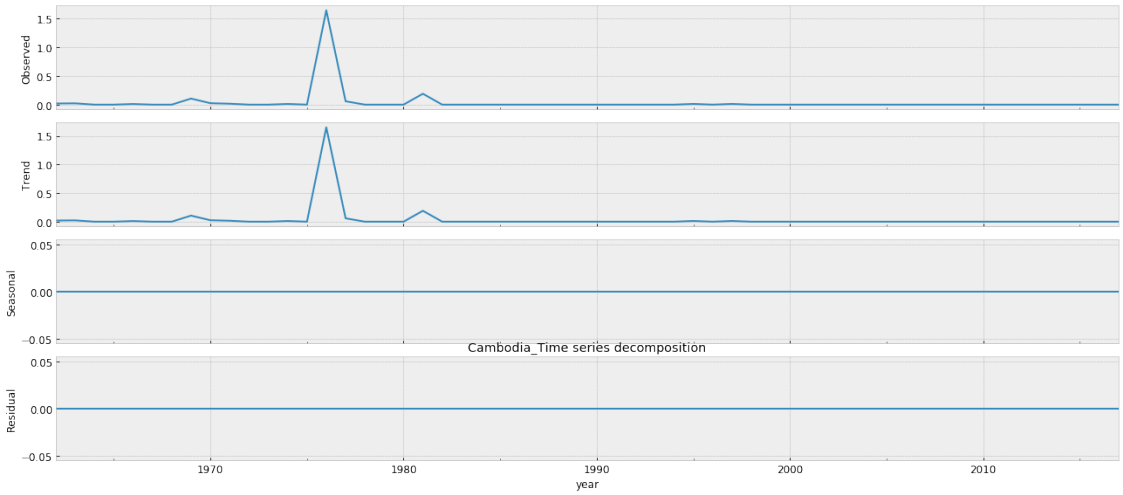

Observed
Trend
Seasonal
Residual
Brunei Darussalam_Time series decomposition
1970
1980
1990
2000
2010
year

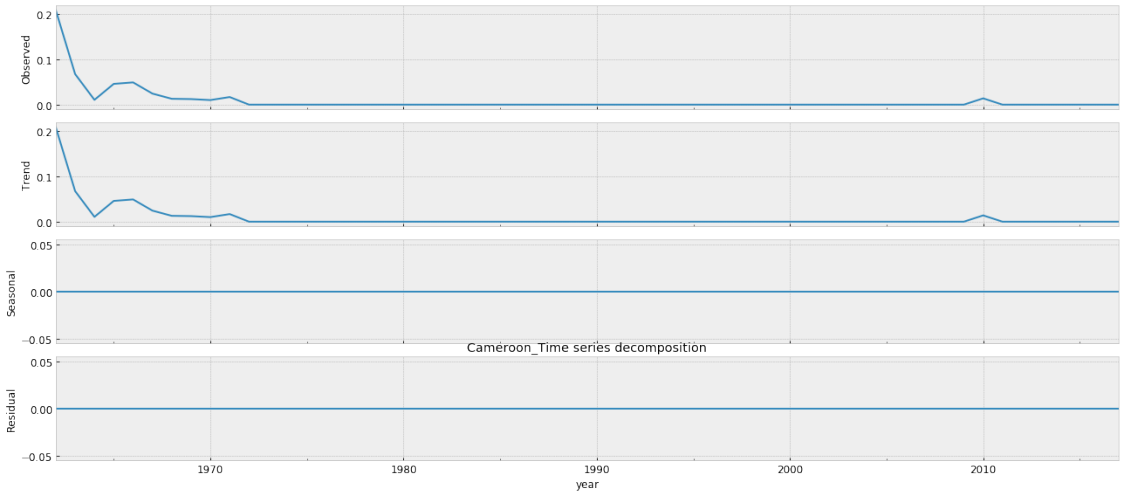

Observed
Trend
Seasonal
Residual
Bulgaria_Time series decomposition
year

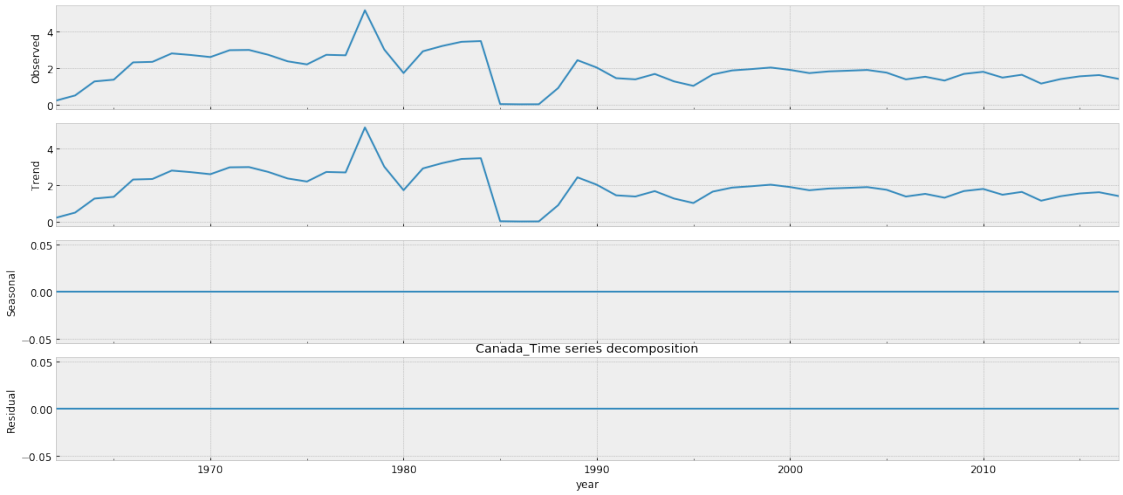

Observed
Trend
Seasonal
Residual
Burkina Faso_Time series decomposition
year

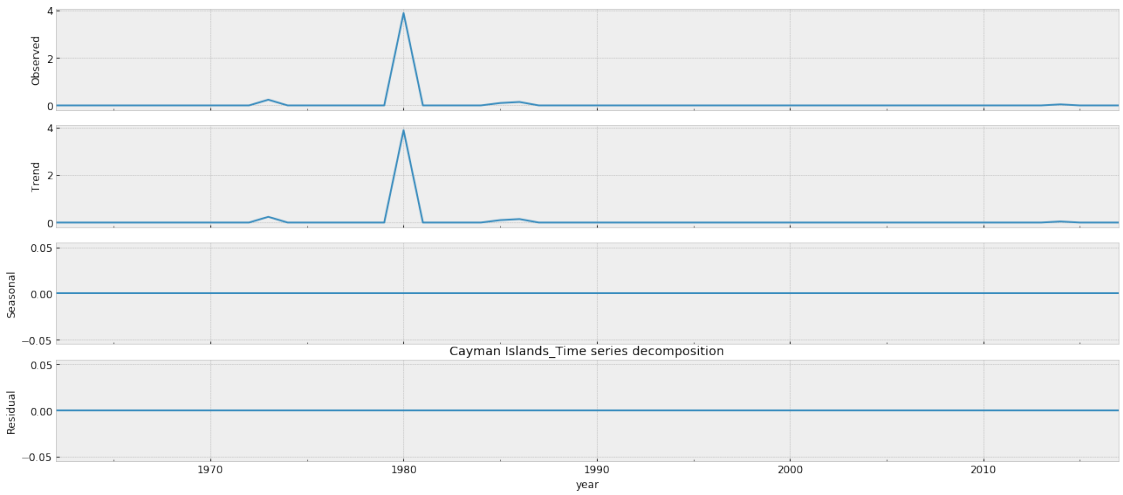

Observed
Trend
Seasonal
Residual
Burundi_Time series decomposition
year

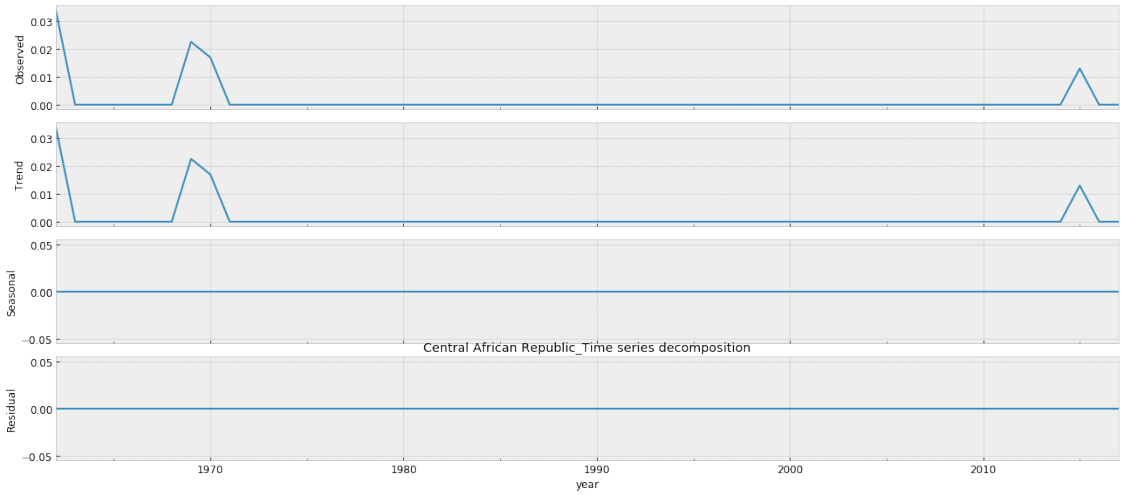

Observed
Trend
Seasonal
Residual
Cabo Verde_Time series decomposition
1970
1980
1990
2000
2010
year

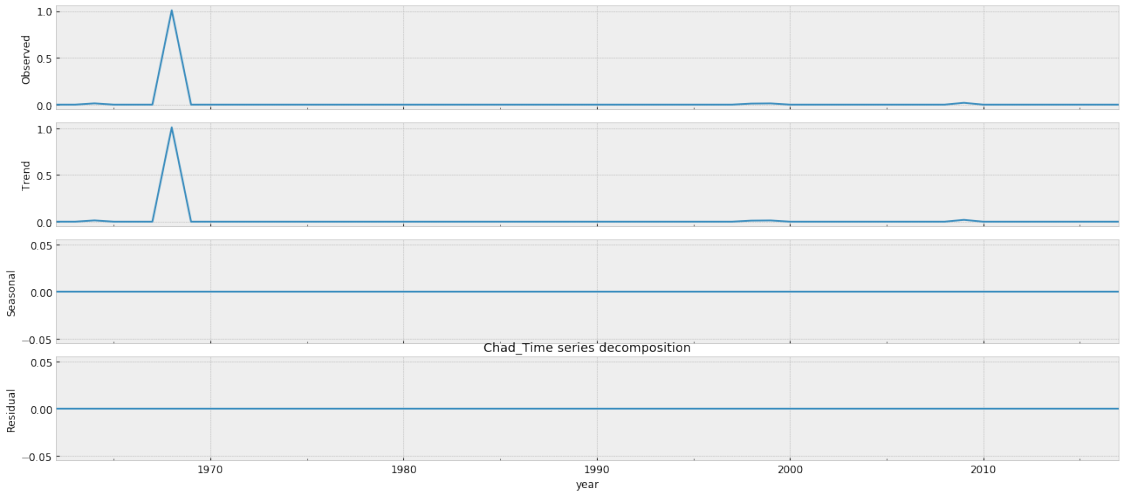

Observed
Trend
Seasonal
Residual
Cambodia_Time series decomposition
1970
1980
1990
2000
2010
year

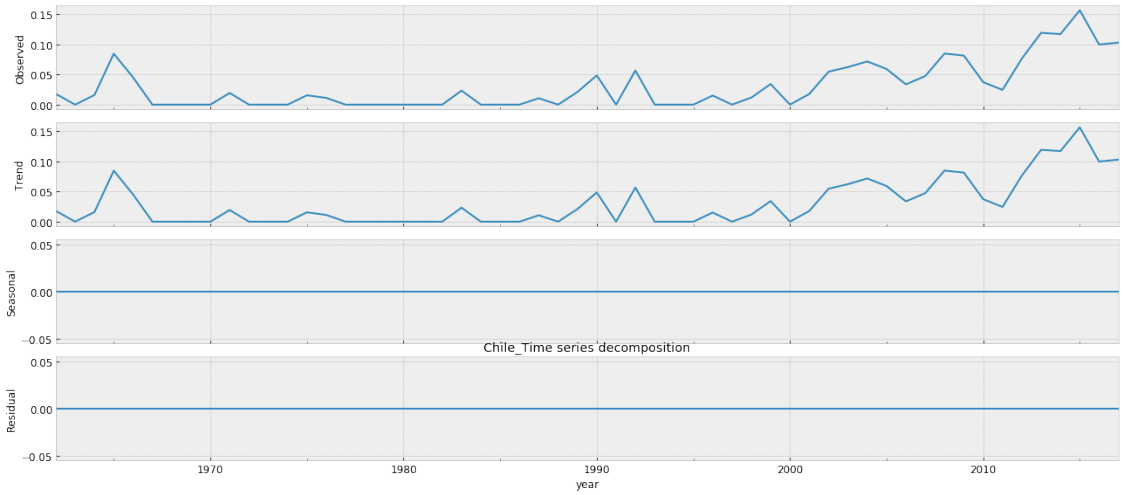

Observed
Trend
Seasonal
Residual
Cameroon_Time series decomposition
year

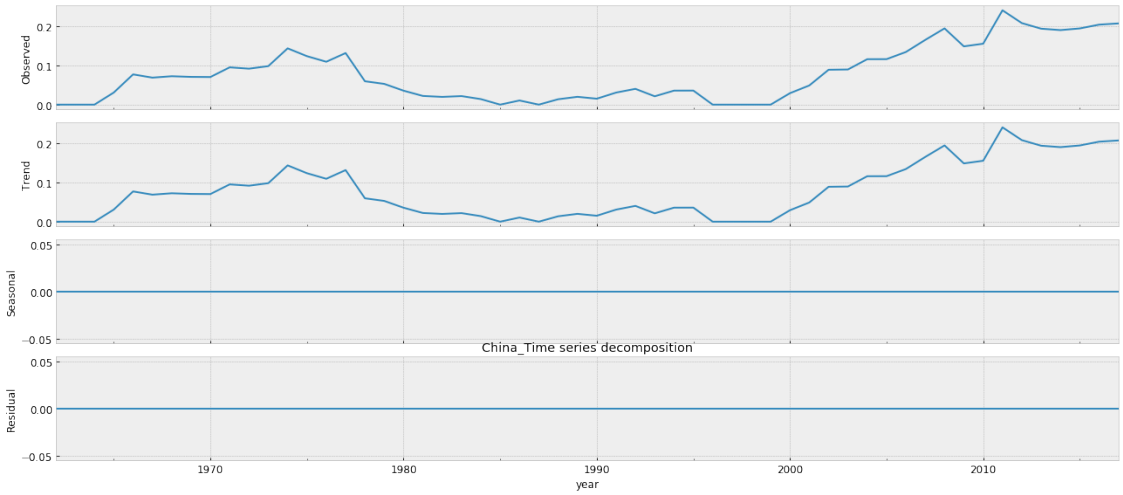

Observed
Trend
Seasonal
Residual
Canada_Time series decomposition
year

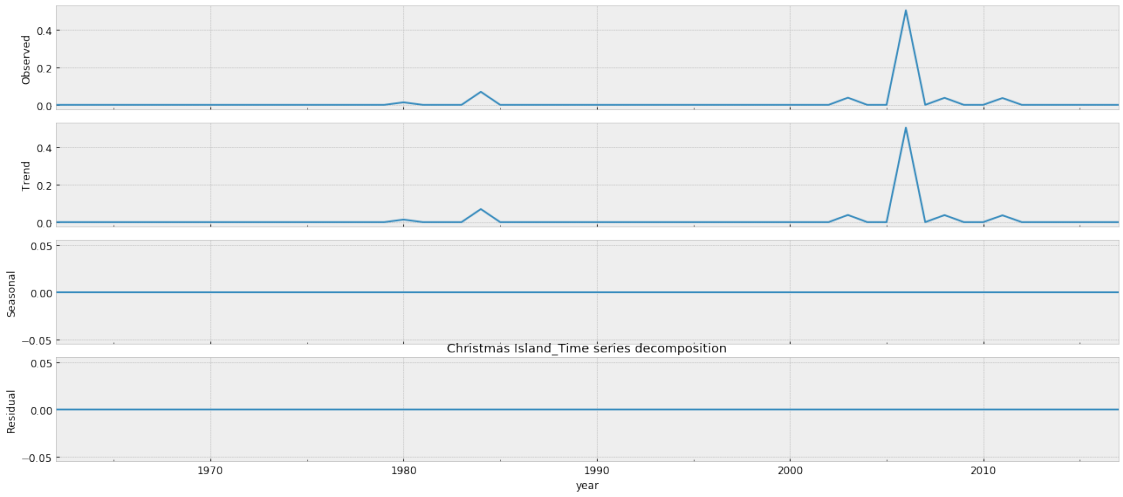

Observed
Trend
Seasonal
Residual
Cayman Islands_Time series decomposition
year

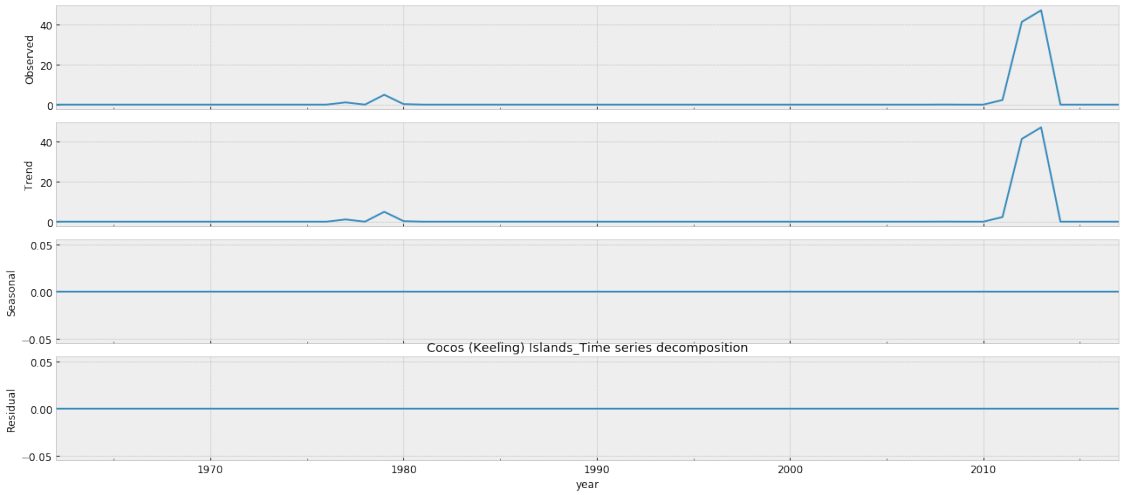

Observed
Trend
Seasonal
Residual
Central African Republic_Time series decomposition
1970
1980
1990
2000
2010
year

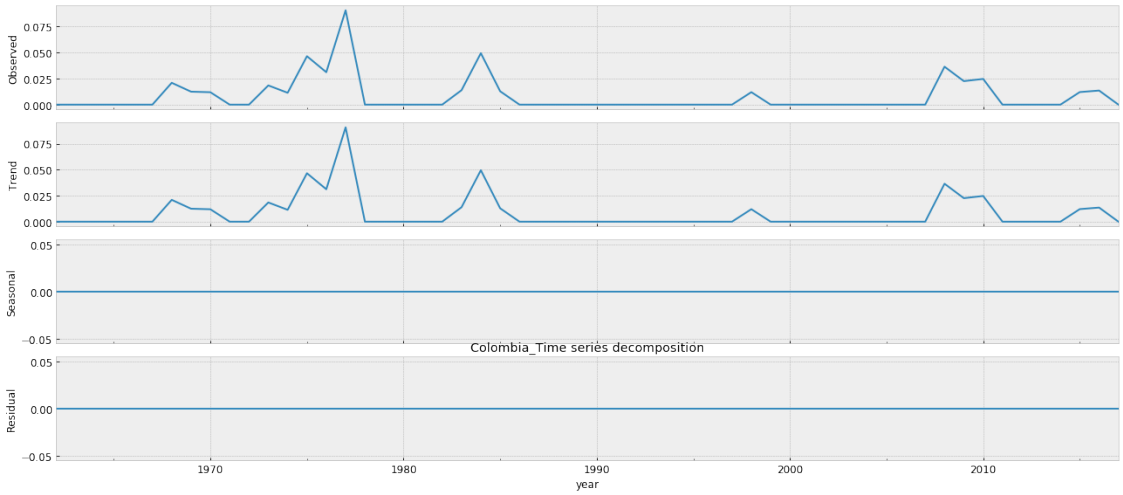

Observed
Trend
Seasonal
Residual
Chad_Time series decomposition
1970
1980
1990
2000
2010
year

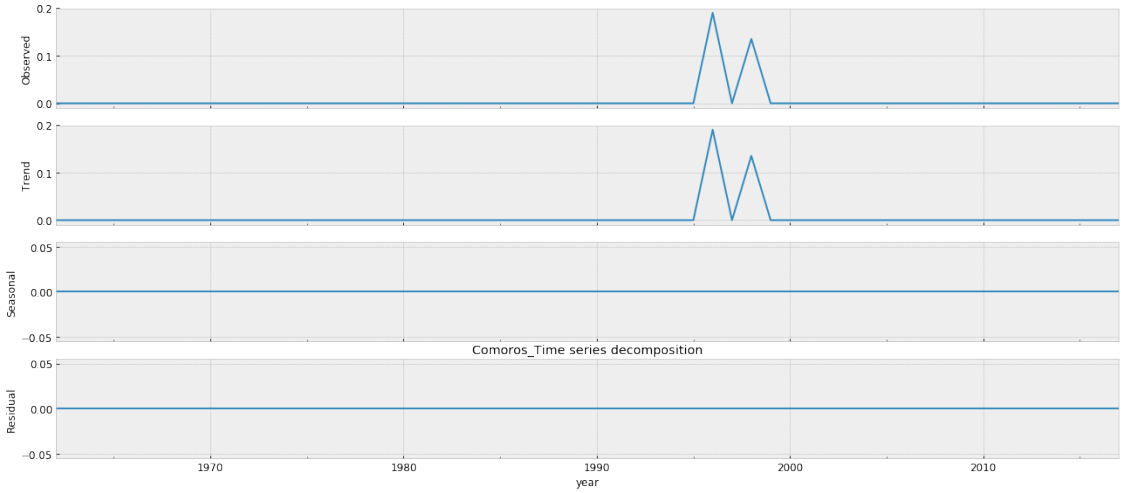

Observed
Trend
Seasonal
Residual
Chile_Time series decomposition
year

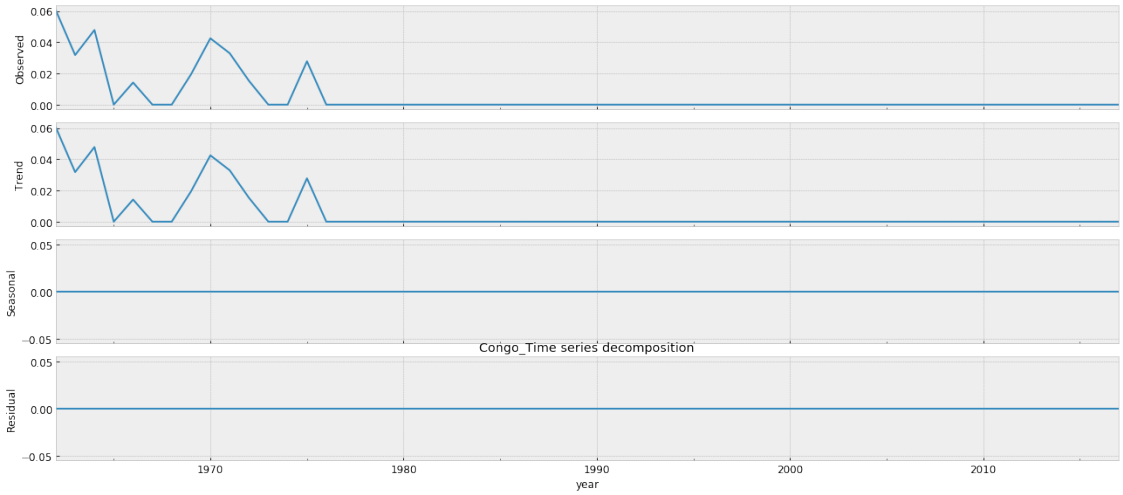

Observed
Trend
Seasonal
Residual
China_Time series decomposition
year

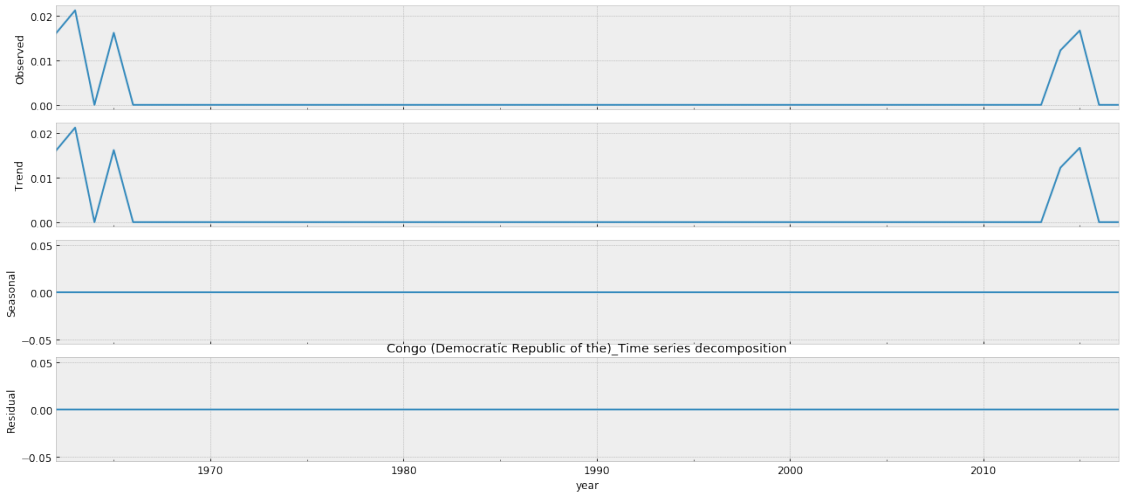

Observed
Trend
Seasonal
Residual
Christmas Island_Time series decomposition
year

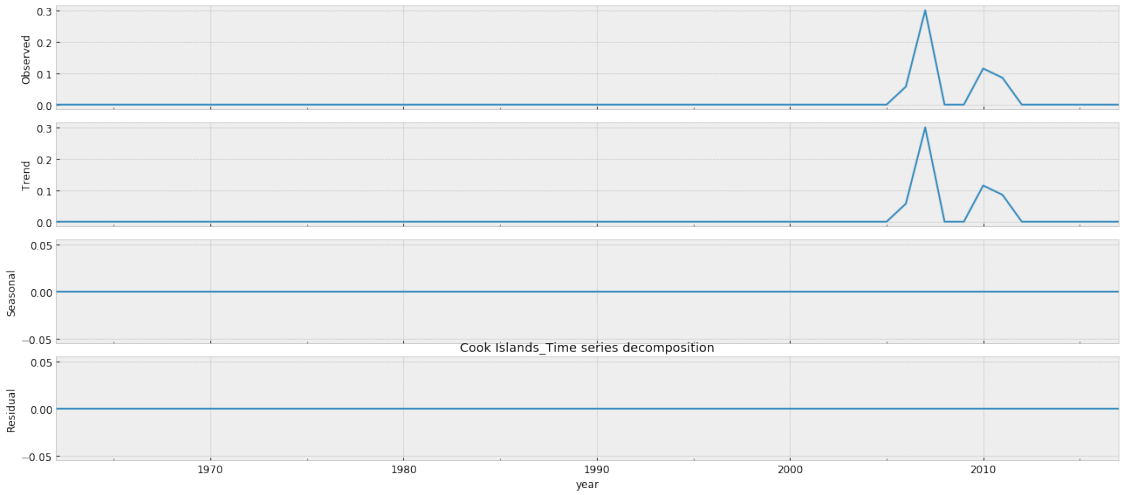

Observed
Trend
Seasonal
Residual
Cocos (Keeling) Islands_Time series decomposition
year

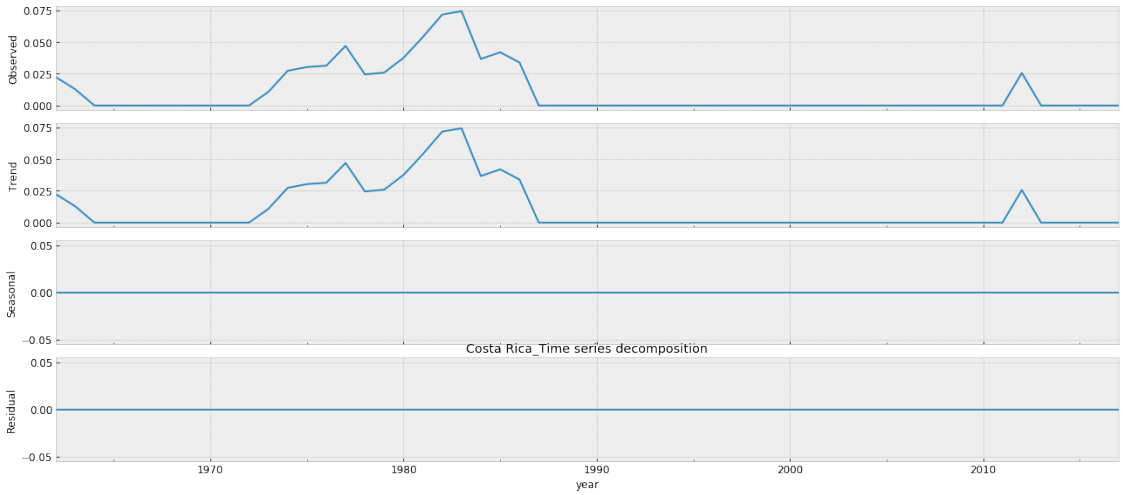

Observed
Trend
Seasonal
Residual
Colombia_Time series decomposition
year

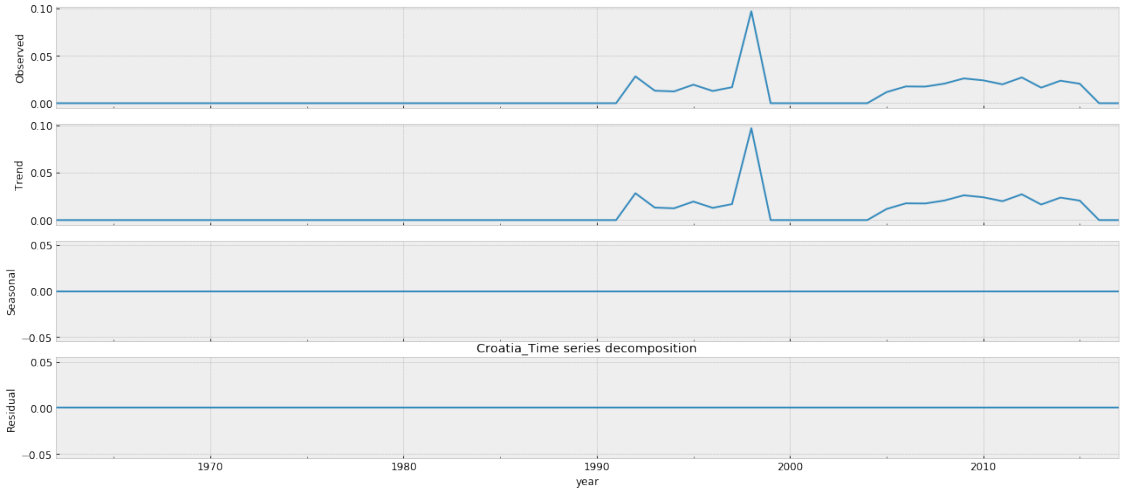

Observed
Trend
Seasonal
Residual
Comoros_Time series decomposition
1970
1980
1990
2000
2010
year

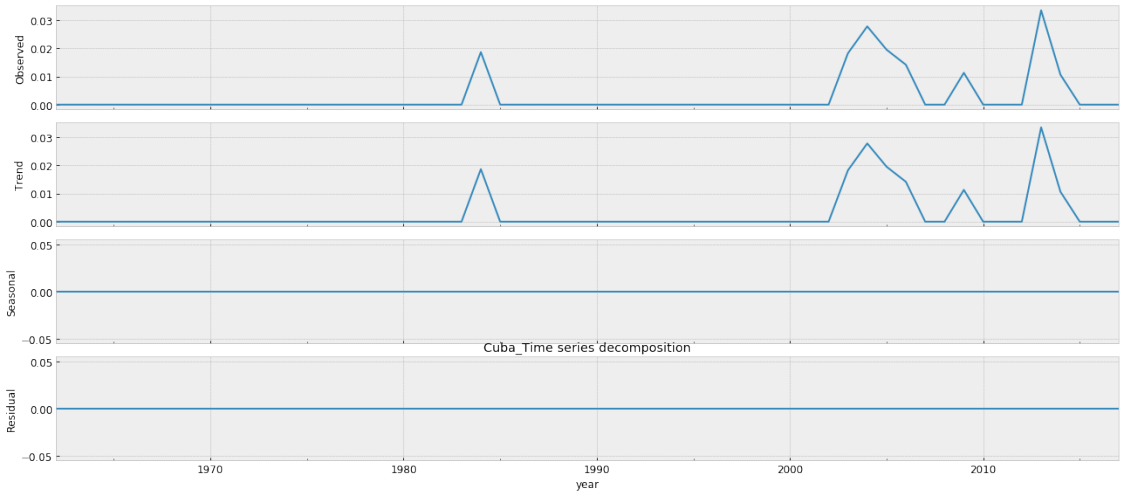

Observed
Trend
Seasonal
Residual
Congo_Time series decomposition
1970
1980
1990
2000
2010
year

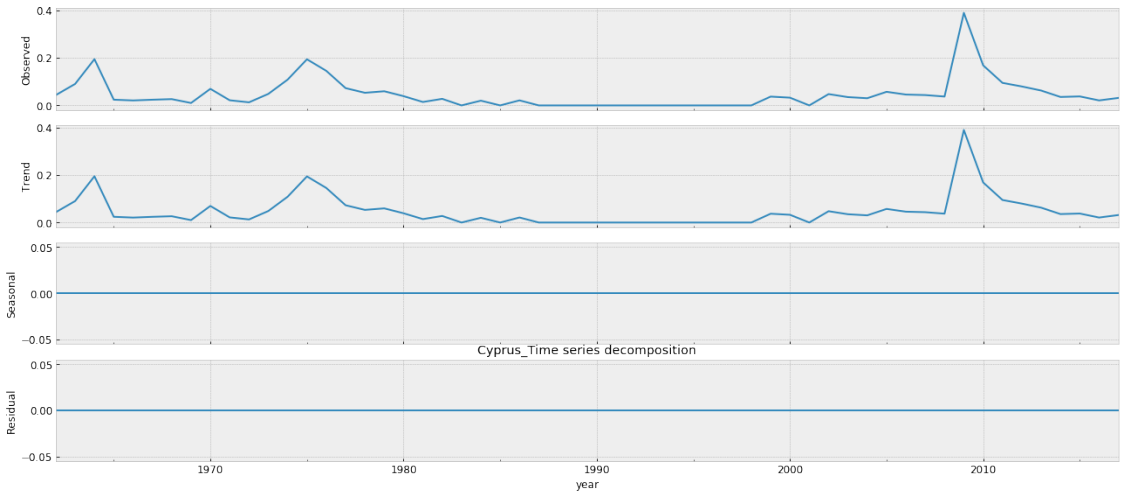

Observed
Trend
Seasonal
Residual
Congo (Democratic Republic of the)_Time series decomposition
1970
1980
1990
2000
2010
year

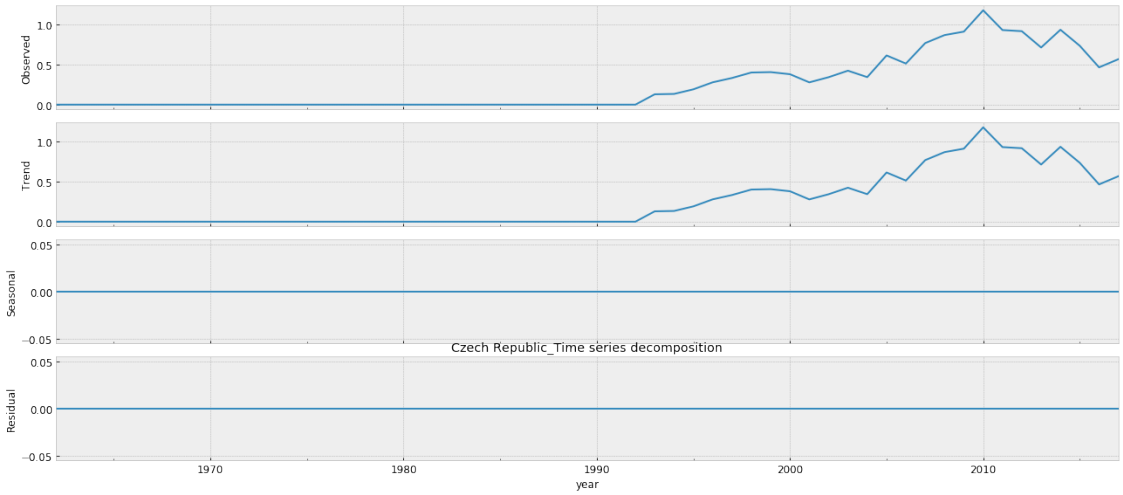

Observed
Trend
Seasonal
Residual
Cook Islands_Time series decomposition
1970
1980
1990
2000
2010
year

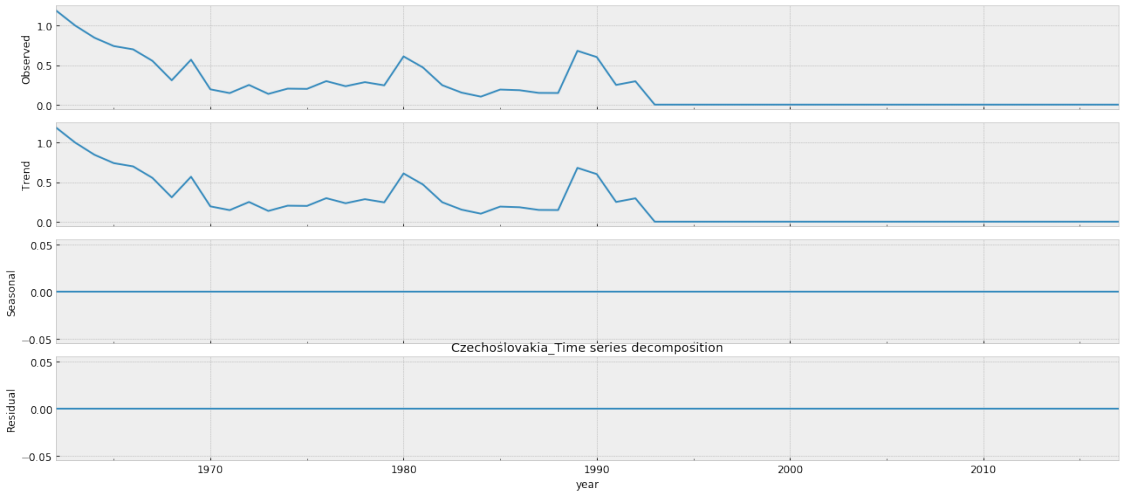

Observed
Trend
Seasonal
Residual
Costa Rica_Time series decomposition
1970
1980
1990
2000
2010
year

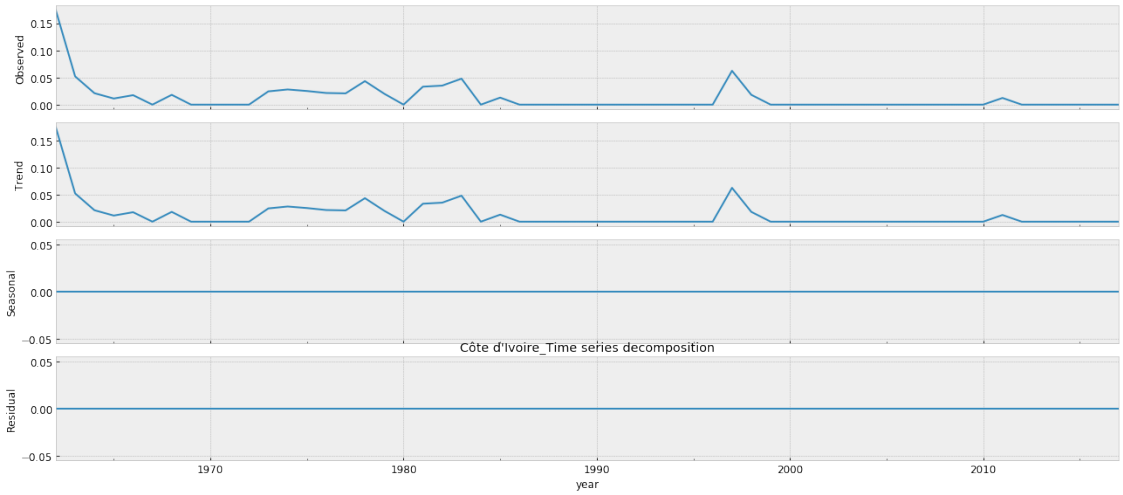

Observed
Trend
Seasonal
Residual
Croatia_Time series decomposition
year

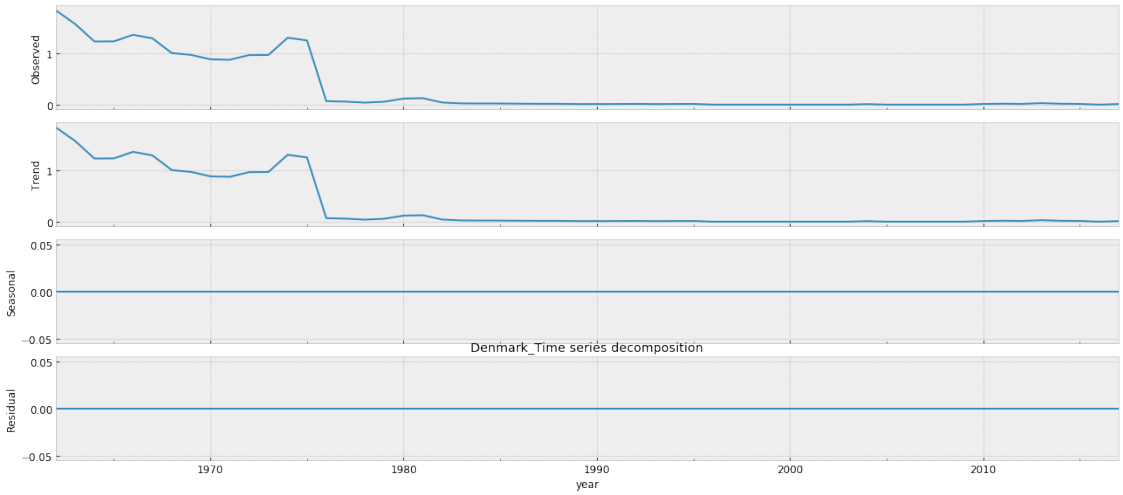

Observed
Trend
Seasonal
Residual
Cuba_Time series decomposition
year

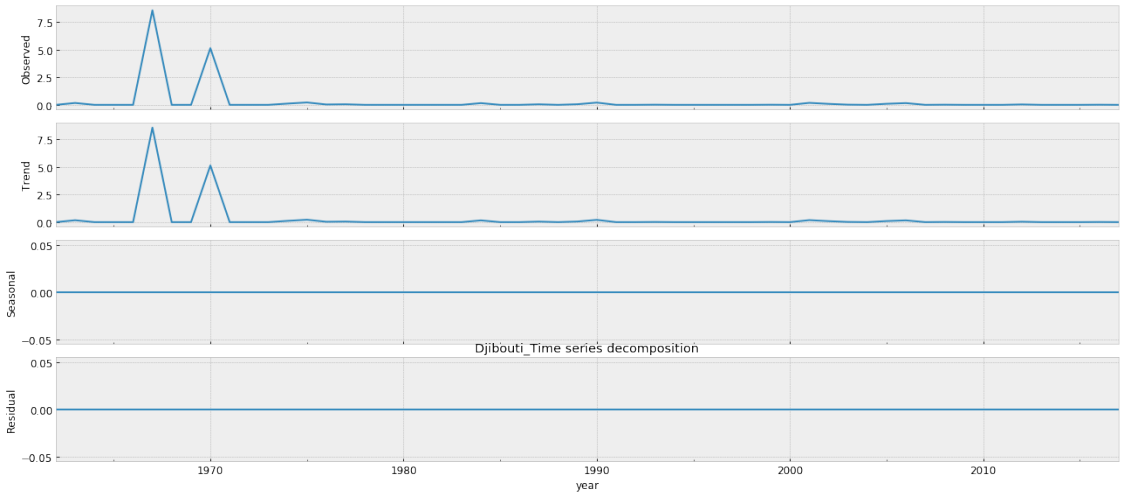

Observed
Trend
Seasonal
Residual
Cyprus_Time series decomposition
year

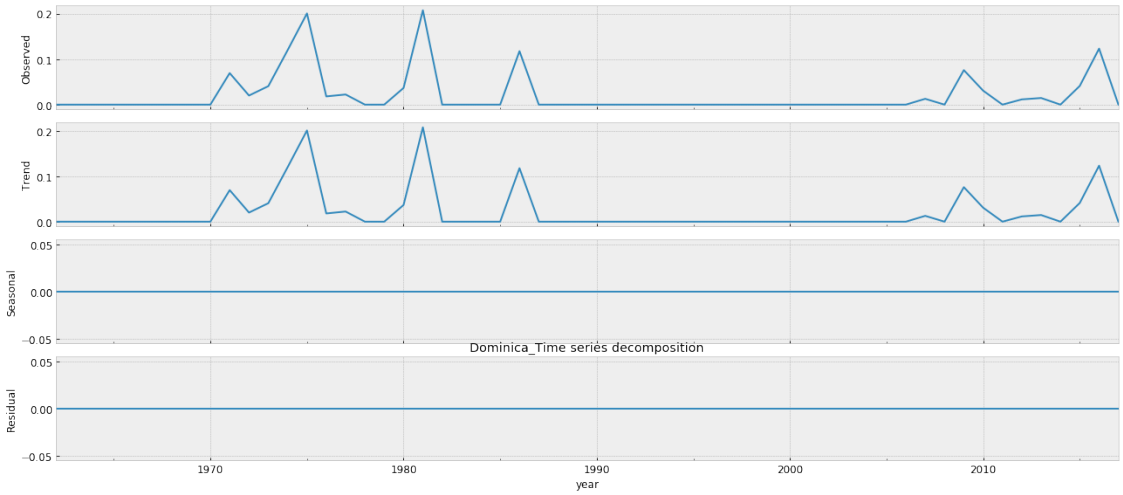

Observed
Trend
Seasonal
Residual
Czech Republic_Time series decomposition
1970
1980
1990
2000
2010
year

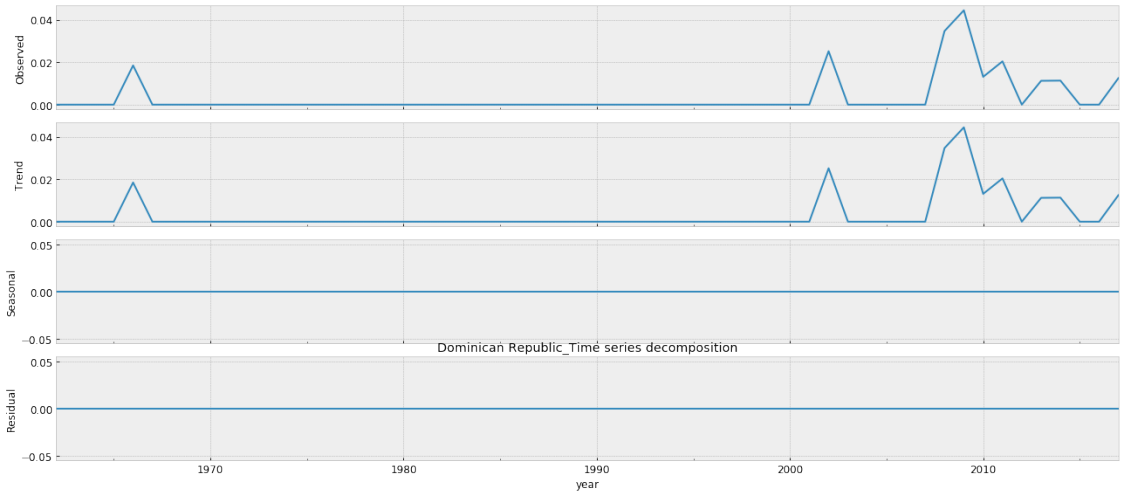

Observed
Trend
Seasonal
Residual
Czechoslovakia_Time series decomposition
1970
1980
1990
2000
2010
year

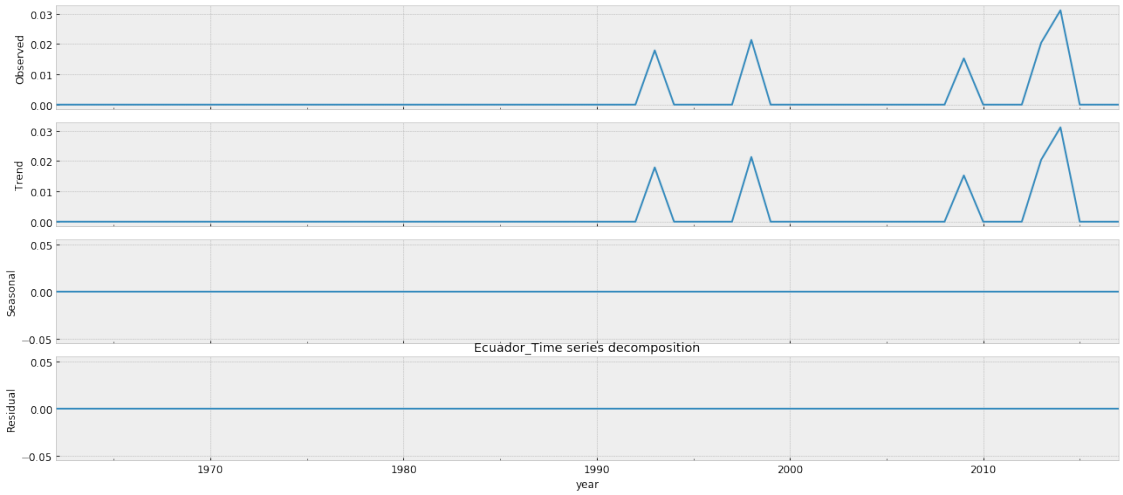

Observed
Trend
Seasonal
Residual
Côte d'Ivoire_Time series decomposition
1970
1980
1990
2000
2010
year

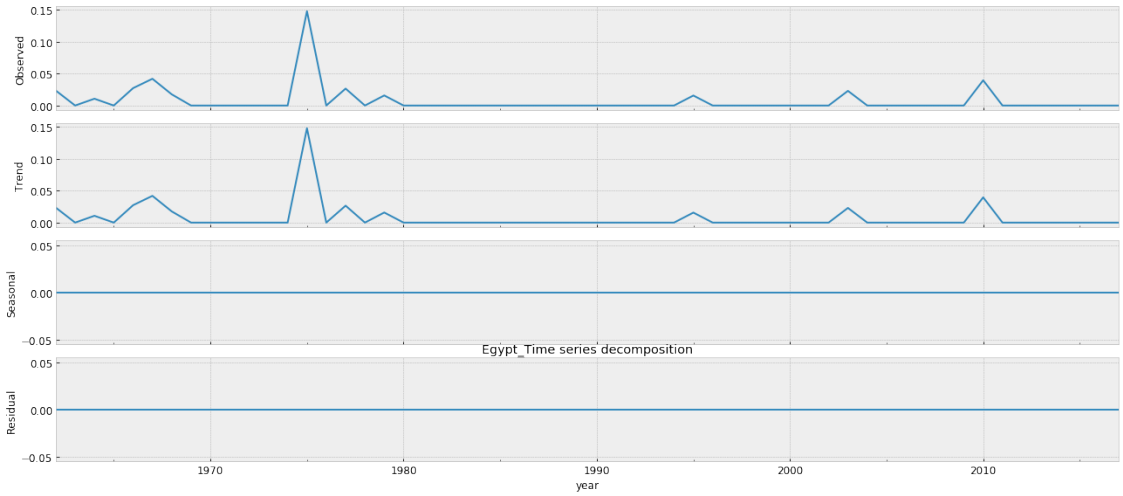

Observed
Trend
Seasonal
Residual
Denmark_Time series decomposition
1970
1980
1990
2000
2010
year

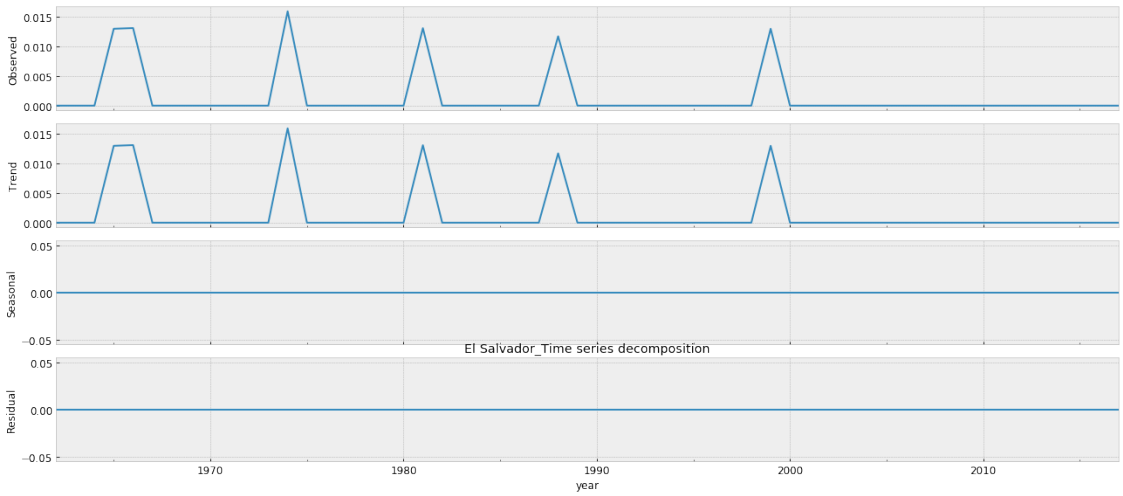

Observed
Trend
Seasonal
Residual
Djibouti_Time series decomposition
1970
1980
1990
2000
2010
year

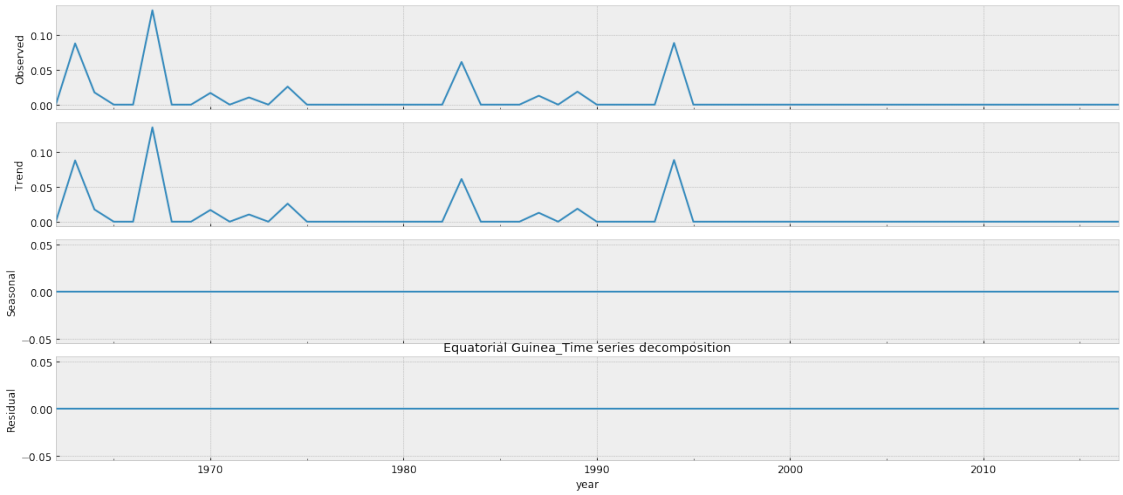

Observed
Trend
Seasonal
Residual
Dominica_Time series decomposition
1970
1980
1990
2000
2010
year

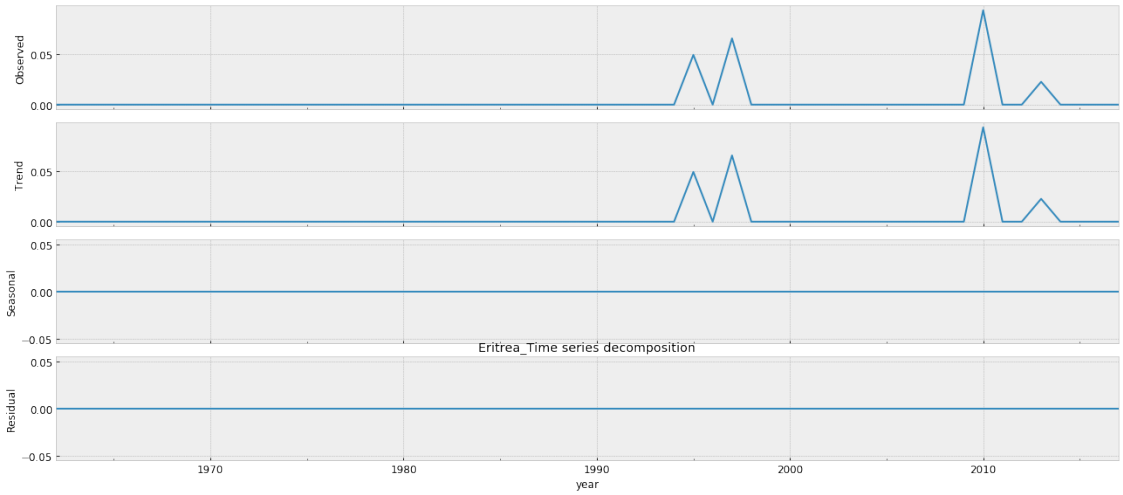

Observed
Trend
Seasonal
Residual
Dominican Republic_Time series decomposition
1970
1980
1990
2000
2010
year

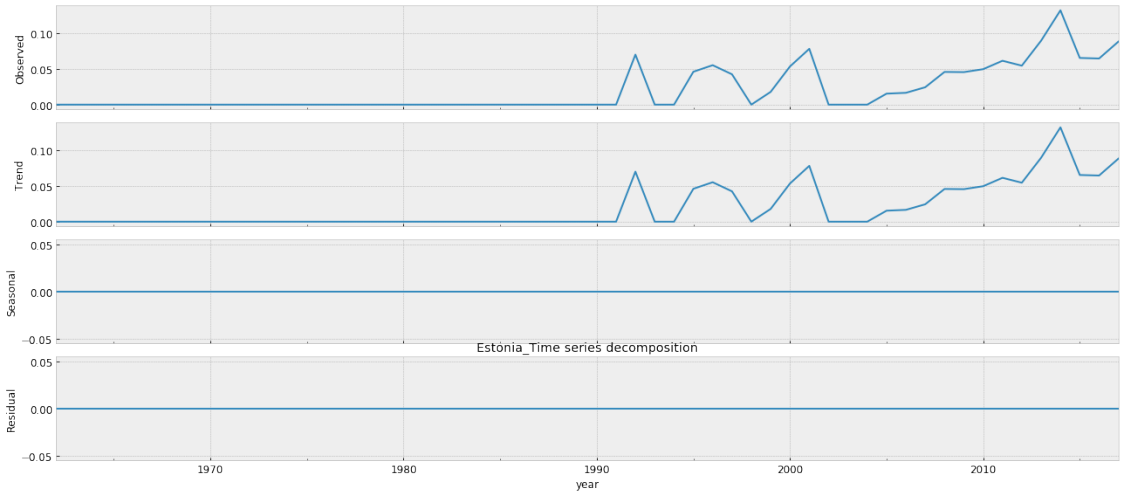

Observed
Trend
Seasonal
Residual
Ecuador_Time series decomposition
1970
1980
1990
2000
2010
year

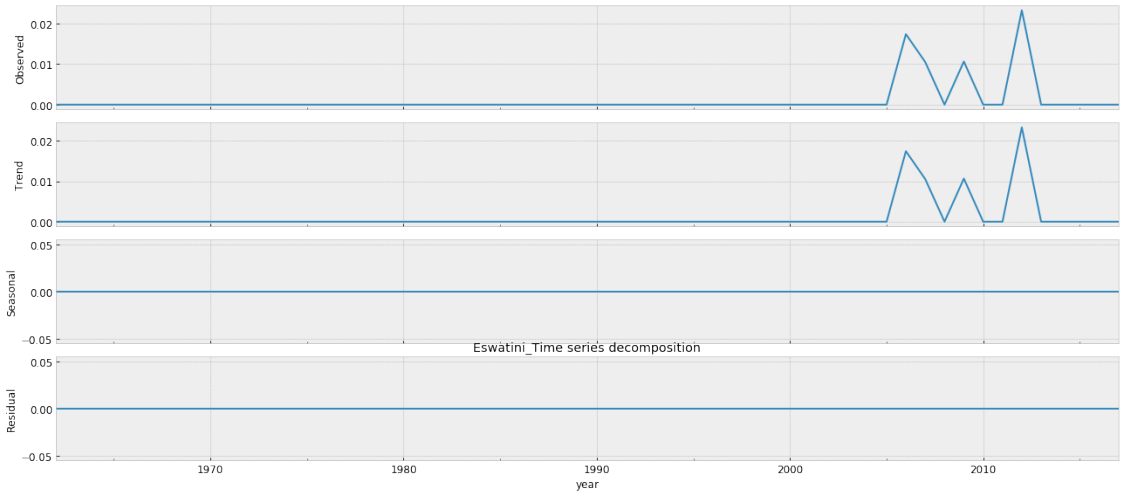

Observed
Trend
Seasonal
Residual
Egypt_Time series decomposition
1970
1980
1990
2000
2010
year

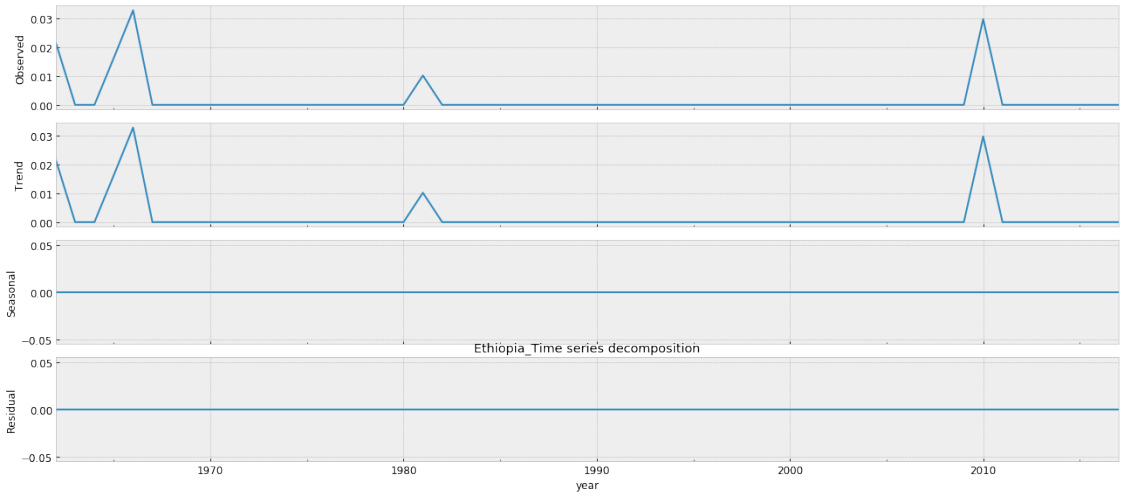

Observed
Trend
Seasonal
Residual
El Salvador_Time series decomposition
1970
1980
1990
2000
2010
year

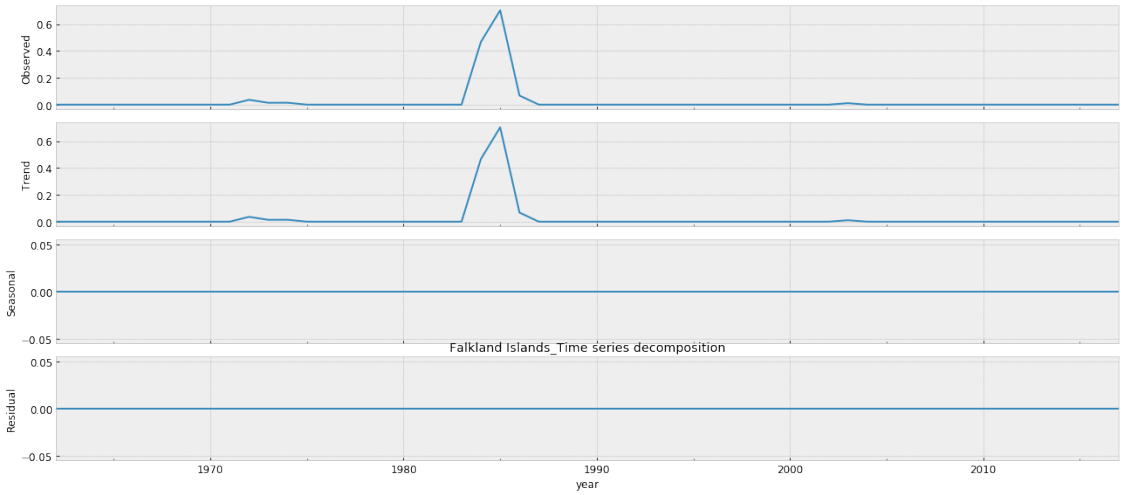

Observed
Trend
Seasonal
Residual
Equatorial Guinea_Time series decomposition
1970
1980
1990
2000
2010
year

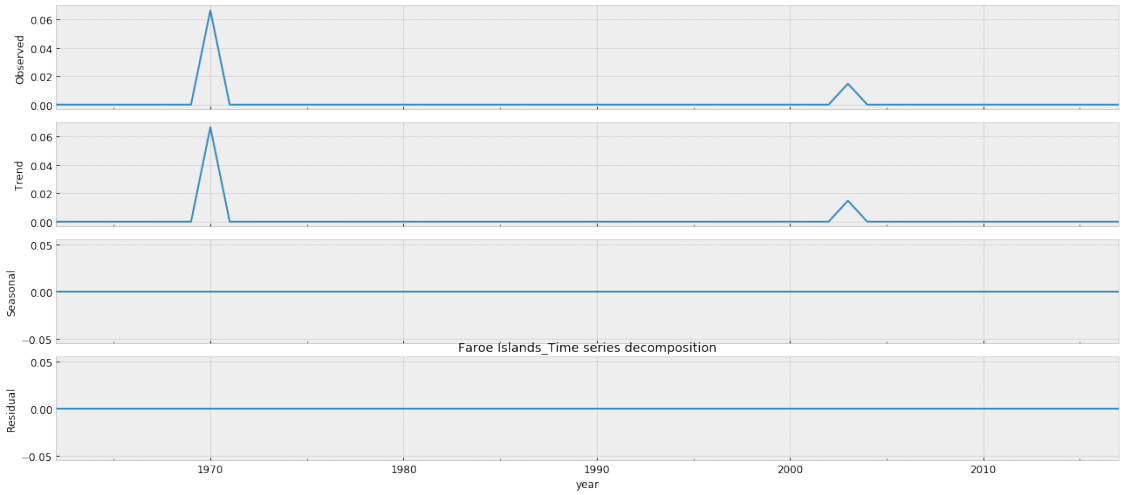

Observed
Trend
Seasonal
Residual
Eritrea_Time series decomposition
1970
1980
1990
2000
2010
year

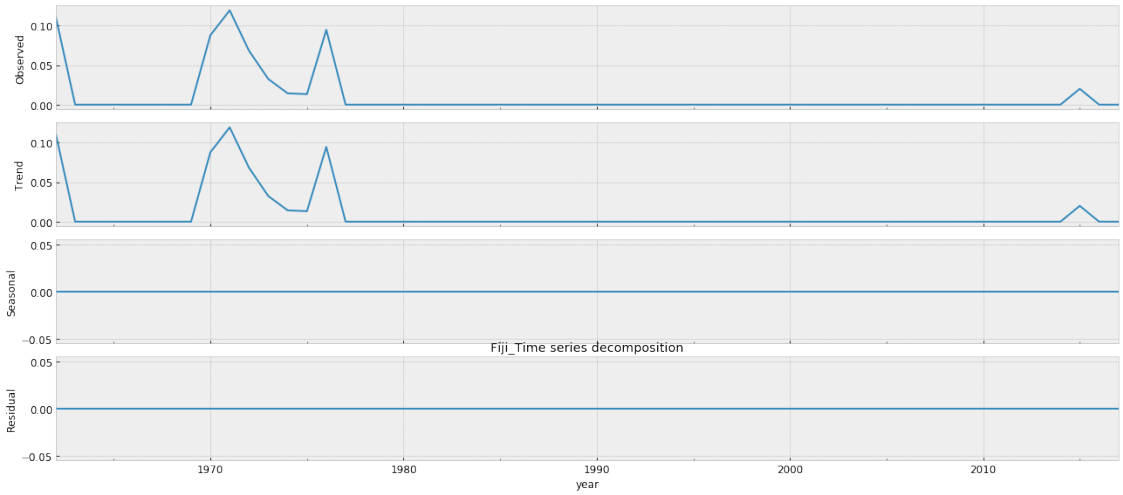

Observed
Trend
Seasonal
Residual
Estonia_Time series decomposition
1970
1980
1990
2000
2010
year

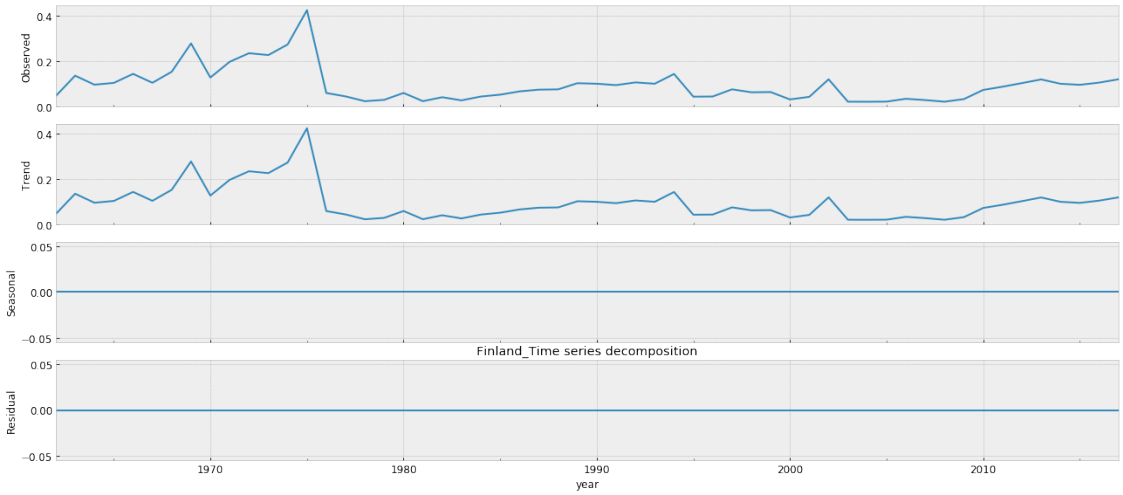

Observed
Trend
Seasonal
Residual
Eswatini_Time series decomposition
1970
1980
1990
2000
2010
year


Observed
Trend
Seasonal
Residual
Ethiopia_Time series decomposition
1970
1980
1990
2000
2010
year


Observed
Trend
Seasonal
Residual
Falkland Islands_Time series decomposition
1970
1980
1990
2000
2010
year

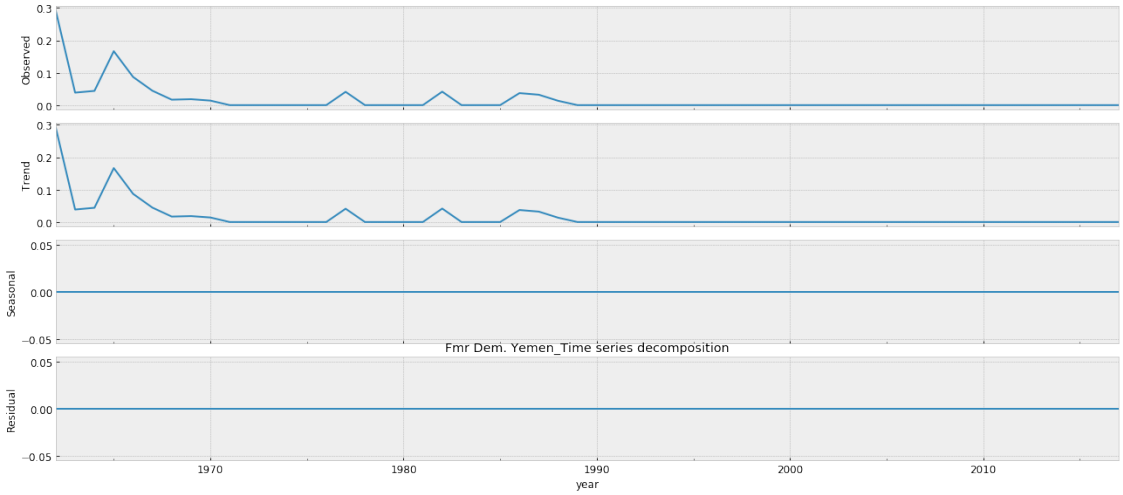

Observed
Trend
Seasonal
Residual
Faroe Islands_Time series decomposition
1970
1980
1990
2000
2010
year






```
    ␣
↪---------------------------------------------------------------------------

      ValueError                                Traceback (most recent call␣
↪last)

      <ipython-input-203-ac560bc7000d> in <module>
        7
        8     plt.title('{}_Time series decomposition'.format(df_grps[country].
↪name))
   ----> 9     plt.savefig(str('{}_{}'.format(savefile,analysis)))
       10     plt.show()
```

```
       ~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args,␣
↪**kwargs)
       687 def savefig(*args, **kwargs):
       688     fig = gcf()
   --> 689     res = fig.savefig(*args, **kwargs)
       690     fig.canvas.draw_idle()   # need this if 'transparent=True' to␣
↪reset colors
       691     return res


       ~\Anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self,␣
↪fname, frameon, transparent, **kwargs)
      2092             self.set_frameon(frameon)
      2093
   -> 2094         self.canvas.print_figure(fname, **kwargs)
      2095
      2096         if frameon:


       ~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in␣
↪print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format,␣
↪bbox_inches, **kwargs)
      2004
      2005         # get canvas object and print method for format
   -> 2006         canvas = self._get_output_canvas(format)
      2007         print_method = getattr(canvas, 'print_%s' % format)
      2008


       ~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in␣
↪_get_output_canvas(self, fmt)
      1946         raise ValueError(
      1947             "Format {!r} is not supported (supported formats: {})"
   -> 1948             .format(fmt, ", ".join(sorted(self.
↪get_supported_filetypes()))))
      1949
      1950     def print_figure(self, filename, dpi=None, facecolor=None,␣
↪edgecolor=None,


       ValueError: Format ' yemen_decomposition' is not supported (supported␣
↪formats: eps, jpeg, jpg, pdf, pgf, png, ps, raw, rgba, svg, svgz, tif, tiff)
```



3.1 Stationarity

- Most of the Time Series models work on the assumption that the TS is stationary.
- Major reason for this is that there are many ways in which a series can be non-stationary, but only one way for stationarity.
- Intuitively, we can say that if a Time Series has a particular behaviour over time, there is a very high probability that it will follow the same behaviour in the future.
- Also, the theories related to stationary series are more mature and easier to implement as compared to non-stationary series.

Notes from Alkaline ML

- tests of stationarity for testing a null hypothesis that an observable univariate time series is stationary around a deterministic trend (i.e. trend-stationary).
- A time series is **stationary when its mean, variance and auto-correlation, etc., are constant over time**.
- Many time-series methods may perform better when a time-series is stationary, since forecasting values becomes a far easier task for a stationary time series (high probability of behaving the same way)
- ARIMAs that include differencing (i.e., d > 0) assume that the data becomes stationary after differencing.
- This is called **difference-stationary**
- Auto-correlation plots are an easy way to determine whether your time series is sufficiently stationary for modeling.
- If the plot does not appear relatively stationary, your model will likely need a differencing term.
- These can be determined by using an Augmented Dickey-Fuller test, or various other statistical testing methods.
- Note that auto_arima will automatically determine the appropriate differencing term for you by default.

In order to quantitatively determine whether we need to difference our data in order to make it

stationary, we can conduct a test of stationarity

We can check stationarity using the following:

1. **ACF and PACF plots**:
   - If the time series is stationary, the ACF/PACF plots will show a **quick drop-off in correlation** after a small amount of lag between points.
2. **Plotting Rolling Statistics**:
   - We can plot the moving average or moving variance and see if it varies with time.
   - **Moving average/variance** is for any instant 't', the average/variance of the last period (e.g year or last 12 months)
3. **Augmented Dickey-Fuller Test [*]:**
   - This is one of the statistical tests for checking stationarity.
   - Here the **null hypothesis is that the TS is non-stationary**.
   - The null hypothesis of the Augmented Dickey-Fuller is that there is a unit root, with the alternative that there is no unit root.
   - If the pvalue is above a critical size, then we cannot reject that there is a unit root.
   - The p-values are obtained through regression surface approximation from MacKinnon 1994, but using the updated 2010 tables
   - If the p-value is close to significant, then the critical values should be used to judge whether to reject the null.
   - The autolag option and maxlag for it are described in Greene
   - The test results comprise of a Test Statistic and some Critical Values for difference confidence levels.
   - If the 'Test Statistic' is less than the 'Critical Value', we can reject the null hypothesis and say that the series is stationary. **Based on papers by**:

.. [*] W. Green. "Econometric Analysis," 5th ed., Pearson, 2003.

.. [*] Hamilton, J.D. "Time Series Analysis". Princeton, 1994.

.. [*] MacKinnon, J.G. 1994. "Approximate asymptotic distribution functions for unit-root and cointegration tests. `Journal of Business and Economic Statistics` 12, 167-76.

.. [*] MacKinnon, J.G. 2010. "Critical Values for Cointegration Tests." Queen's University, Dept of Economics, Working Papers. Available at http://ideas.repec.org/p/qed/wpaper/1227.html

3.1.1 ACF and PACF plots

- Let's review the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots
- If the time series is stationary, the ACF/PACF plots will show a **quick drop-off in correlation** after a small amount of lag between points.
- This data is non-stationary as a high number of previous observations are correlated with future values.
- Confidence intervals are drawn as a cone.
- By default, this is set to a 95% confidence interval, suggesting that correlation values outside of this code are very likely a correlation and not a statistical fluke.
- The partial autocorrelation at lag k is the correlation that results after removing the effect of any correlations due to the terms at shorter lags.

[ ]:
```
for country in countries:
    savefile = save_images(df_grps,'acf_pacf')

    plt.figure(figsize=(15,6))
    plt.subplot(121)
    plot_acf(df_grps[country], ax=pyplot.gca(),title="{} - Autocorrelation".
 ↪format(df_grps[country].name))

    plt.subplot(122);
    plot_pacf(df_grps[country], ax=pyplot.gca(),title="{} - Partial␣
 ↪Autocorrelation".format(df_grps[country].name))

    plt.savefig(str('{}_ACF_PACF'.format(savefile)))
    plt.show()
```

3.2 Interpreting ACF plots

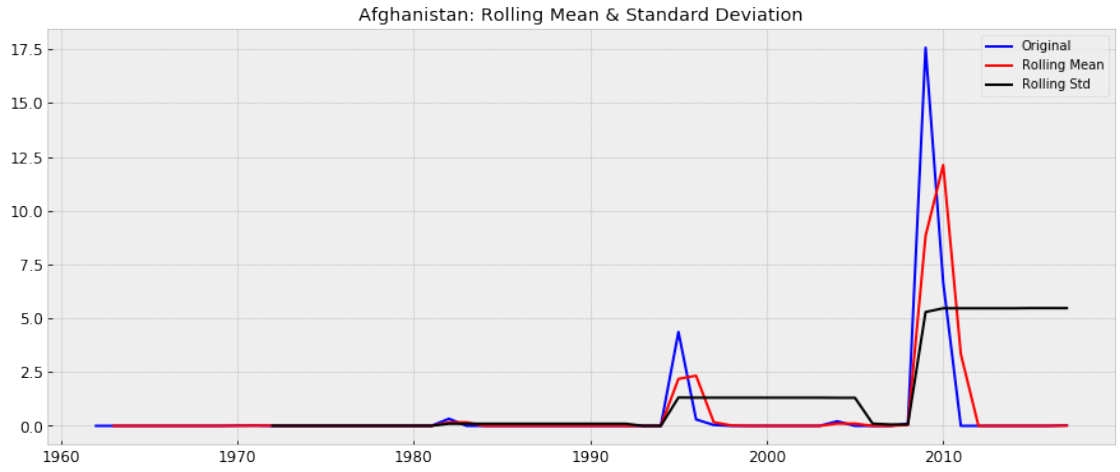
ACF Shape	Indicated Model
Exponential, decaying to zero	Autoregressive model. Use the partial autocorrelation plot to identify the order of the autoregressive model
Alternating positive and negative, decaying to zero Autoregressive model.	Use the partial autocorrelation plot to help identify the order.
One or more spikes, rest are essentially zero	Moving average model, order identified by where plot becomes zero.
Decay, starting after a few lags	Mixed autoregressive and moving average (ARMA) model.
All zero or close to zero	Data are essentially random.
High values at fixed intervals	Include seasonal autoregressive term.
No decay to zero	Series is not stationary

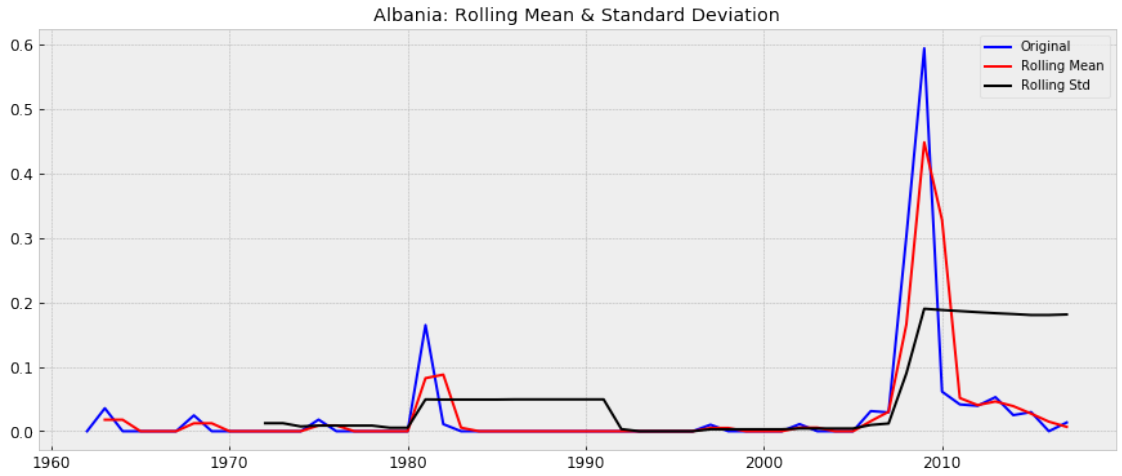
3.2.1 Plotting Rolling Statistics

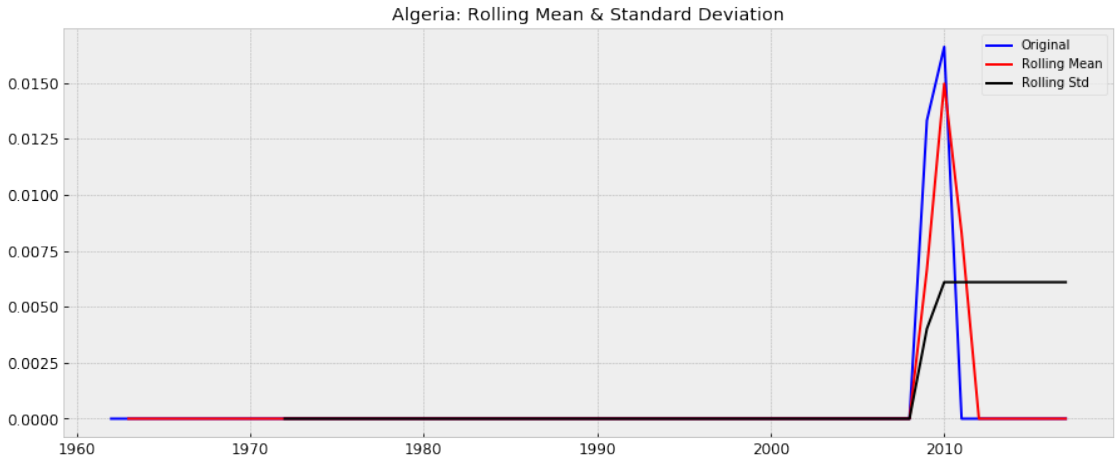
- We observe that the rolling mean and Standard deviation are not constant with respect to time (increasing trend)
- The time series is hence not stationary

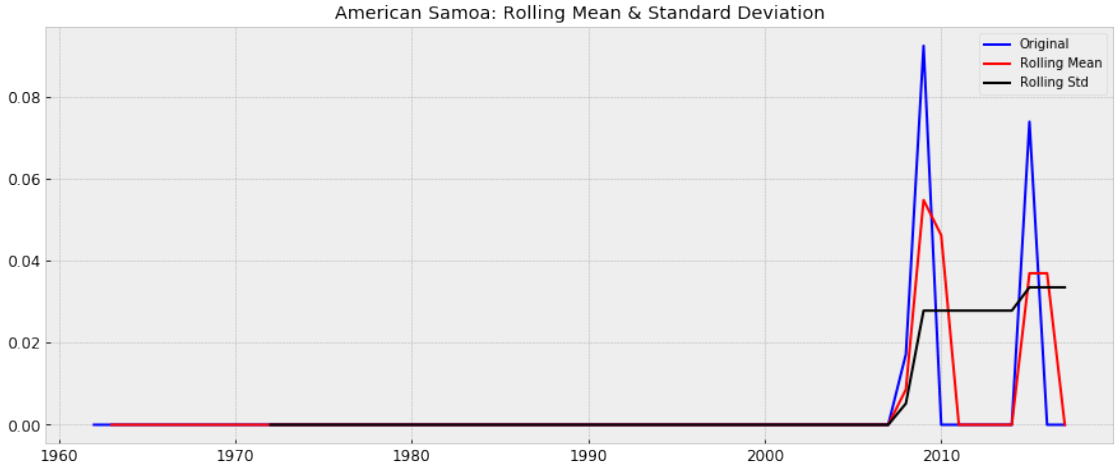
[64]:
```
cycles = int((len(df_grps))/5)

for country in countries:
    savefile = save_images(df_grps,'rolling_stats')

    plt.figure(figsize=(15,6))

    rolmean = df_grps[country].rolling(2).mean()
    rolstd = df_grps[country].rolling(cycles).std()
```

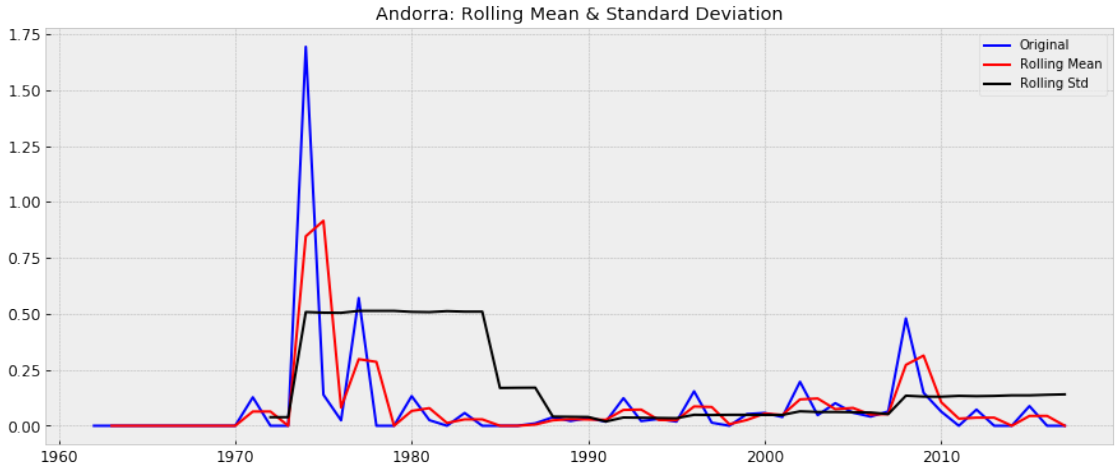
```
    #Plot rolling statistics:
    orig = plt.plot(df_grps[country], color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')

    plt.legend(loc='best')
    plt.title('{}: Rolling Mean & Standard Deviation'.format(df_grps[country].
↪name))
    plt.savefig(str('{}_Rolling_Stats'.format(savefile)))
    plt.show(block=False)
```

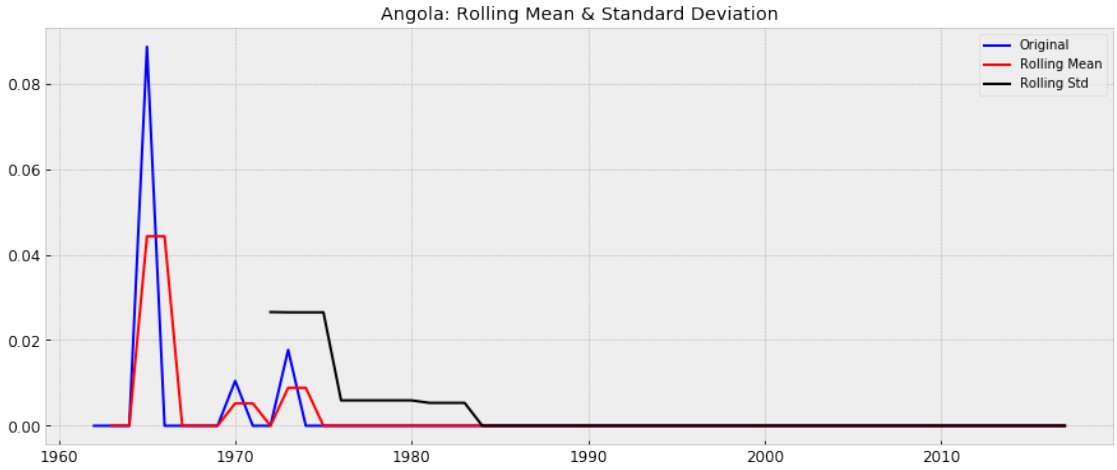


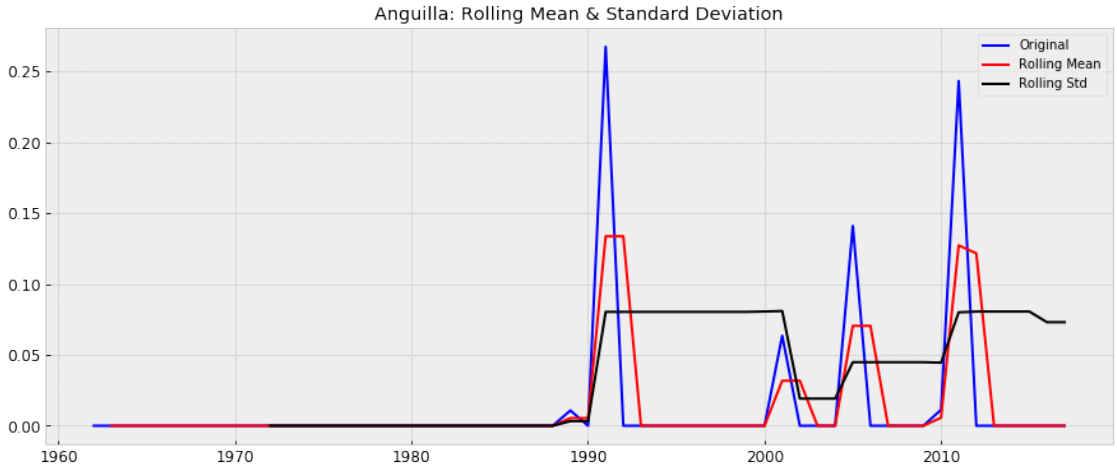


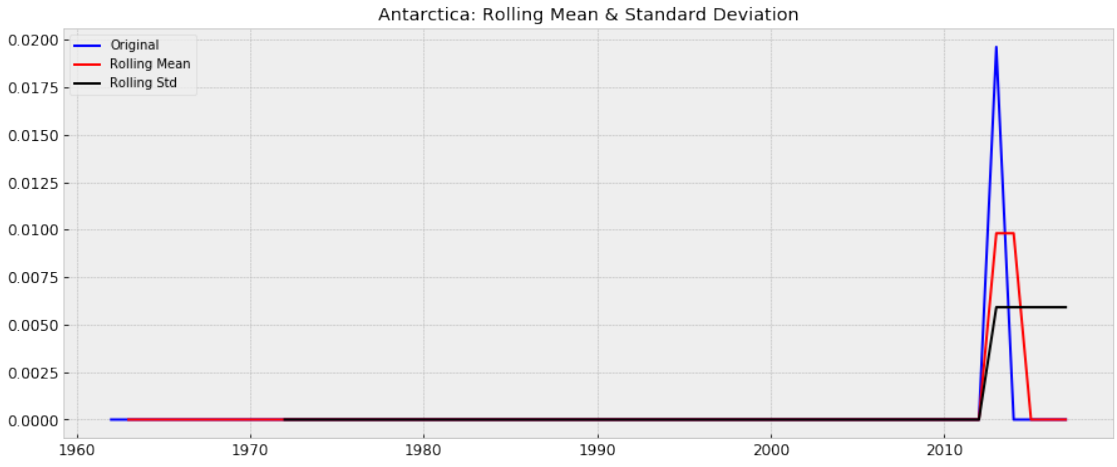

Algeria: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
1960
1970
1980
1990
2000
2010

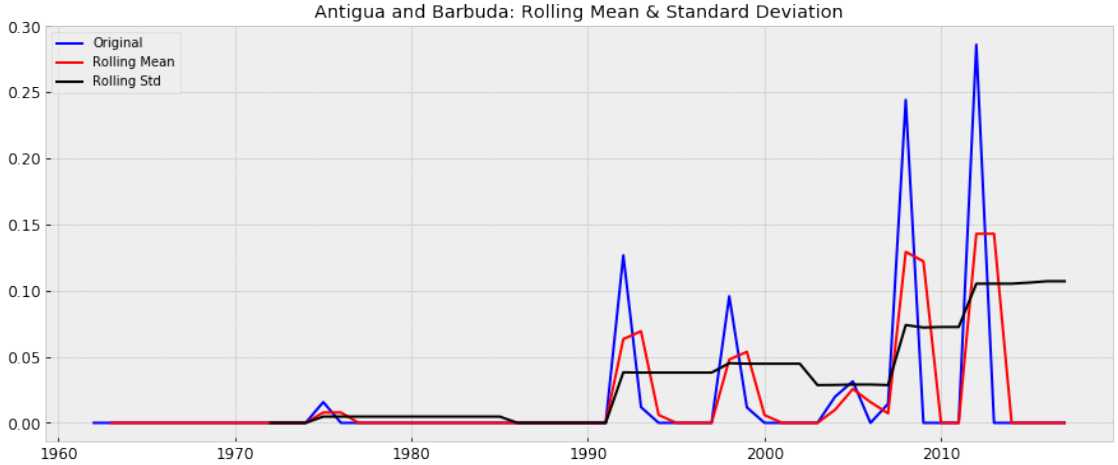

American Samoa: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

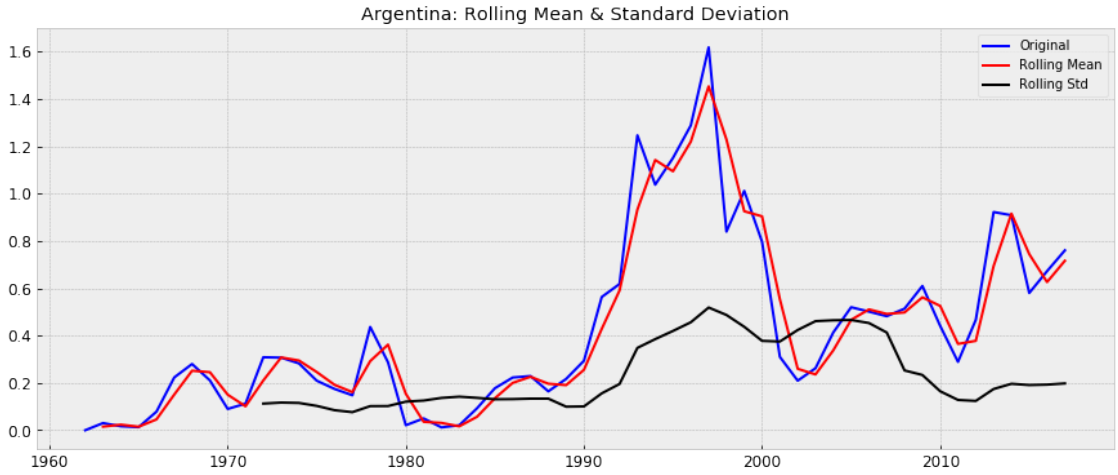

Andorra: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
1960
1970
1980
1990
2000
2010

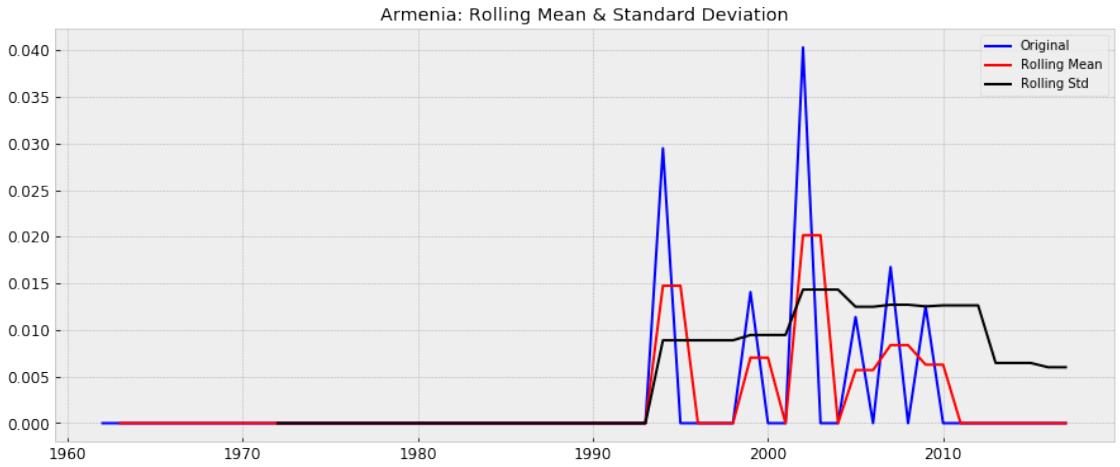

Angola: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

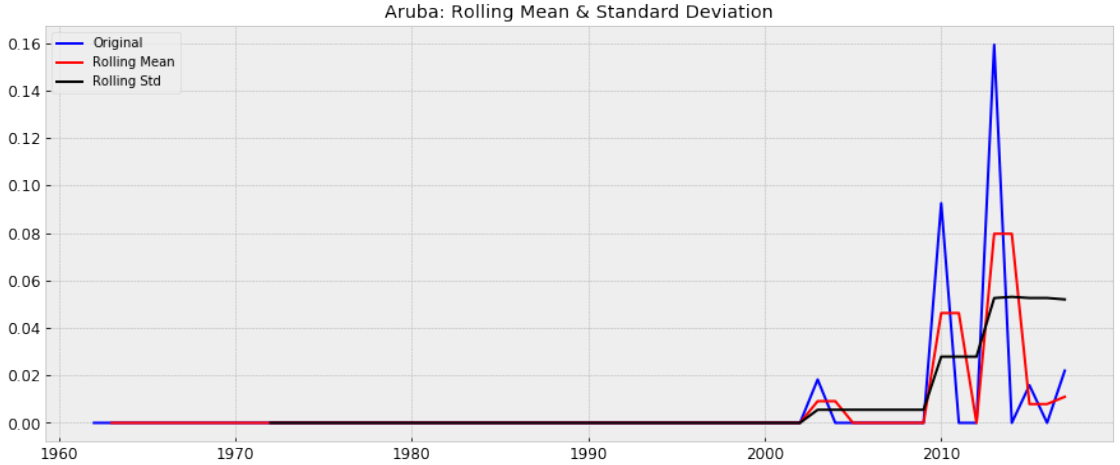

Anguilla: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.25
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010

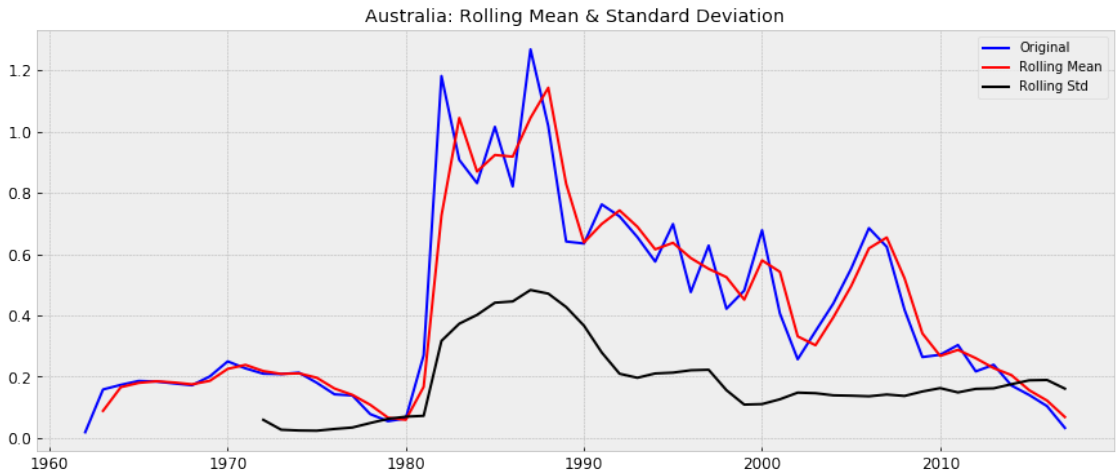

Antarctica: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
1960
1970
1980
1990
2000
2010

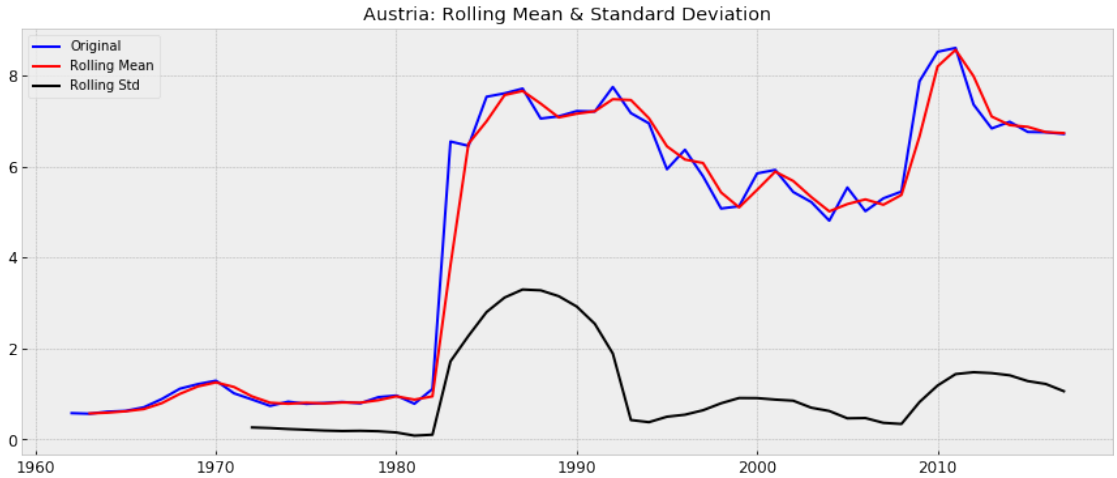

Antigua and Barbuda: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010

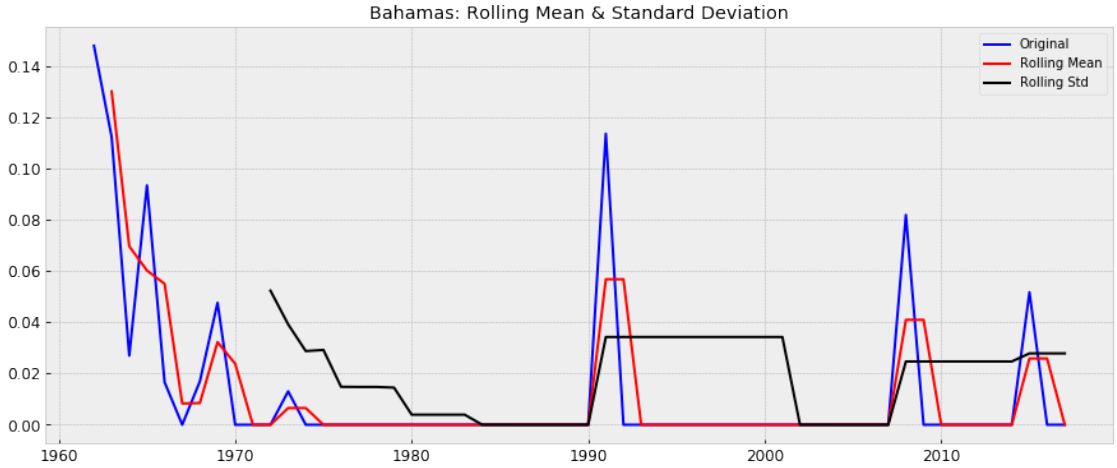

Argentina: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1960
1970
1980
1990
2000
2010

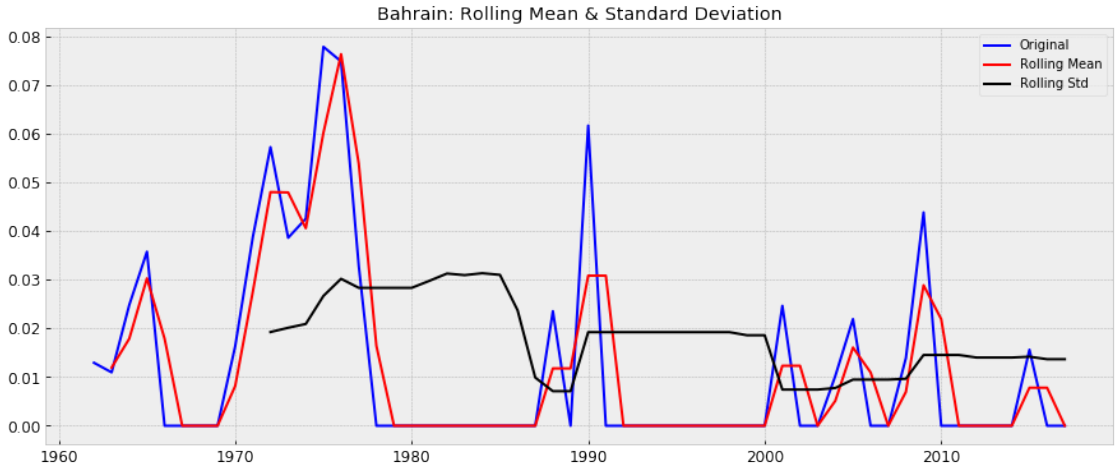

Armenia: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
1960
1970
1980
1990
2000
2010

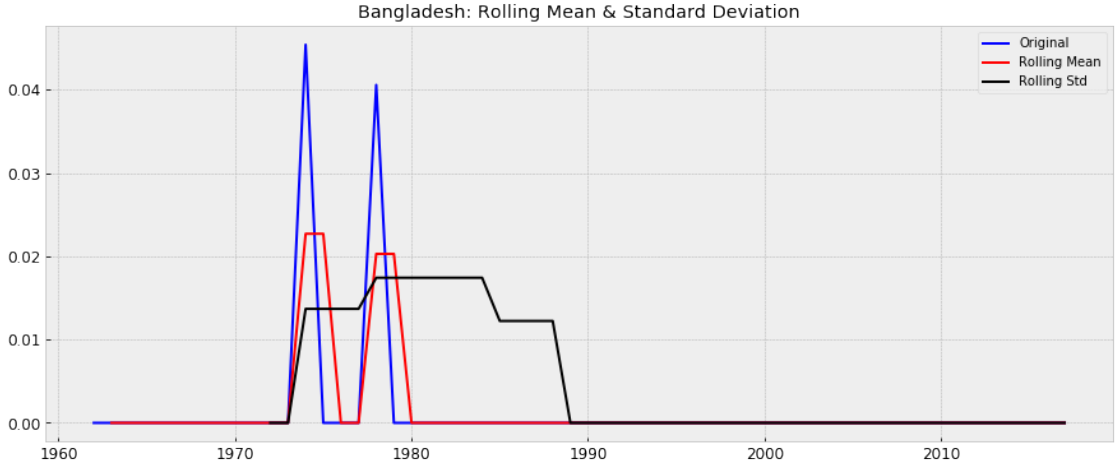

Aruba: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

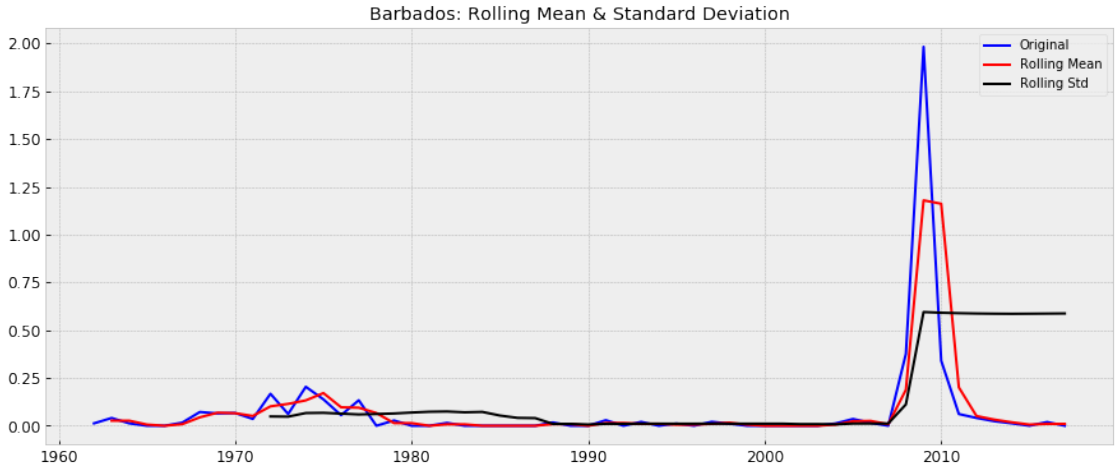

Australia: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1960
1970
1980
1990
2000
2010

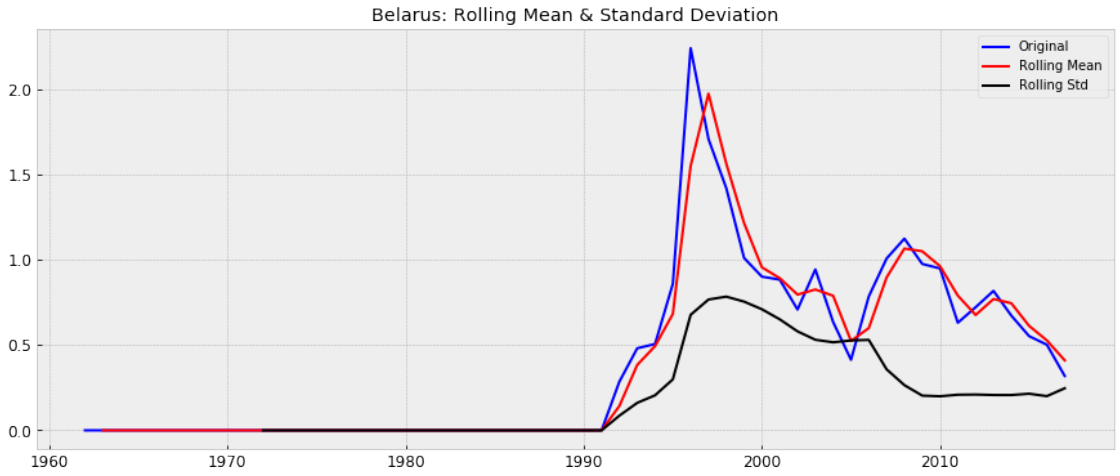

Austria: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0
2
4
6
8
1960
1970
1980
1990
2000
2010

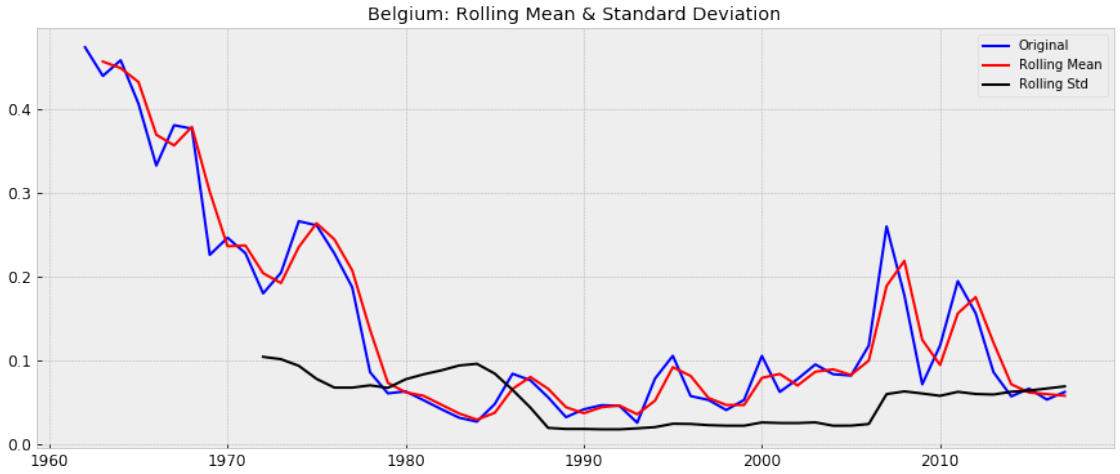

Bahamas: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
1960
1970
1980
1990
2000
2010

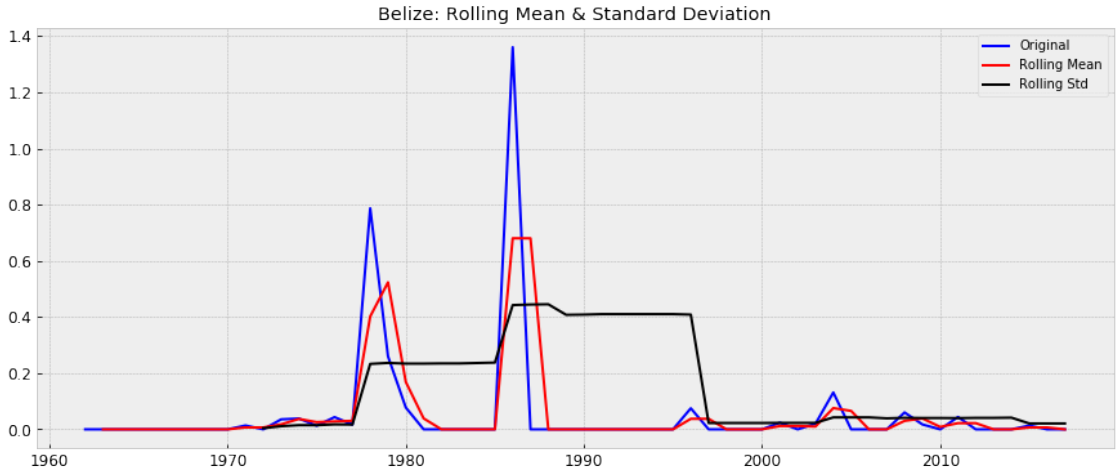

Bahrain: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
1960
1970
1980
1990
2000
2010

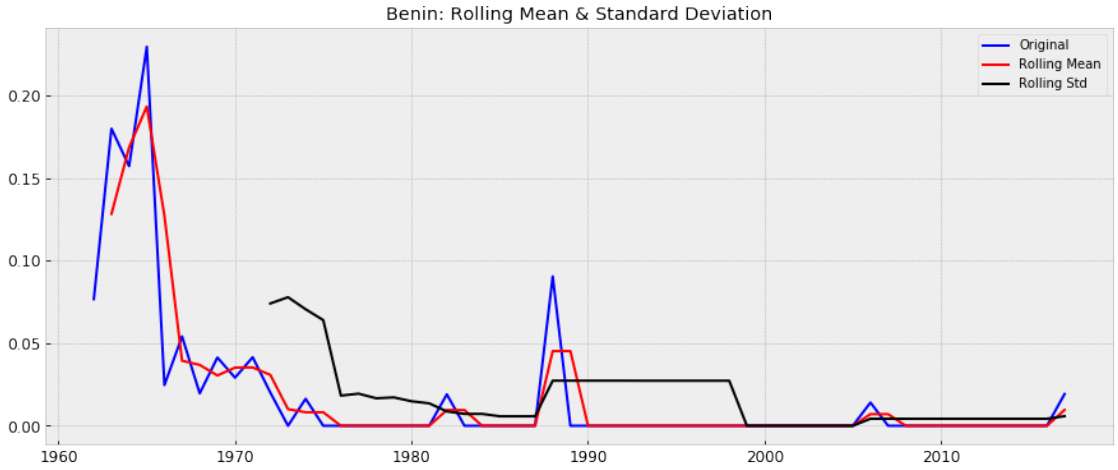

Bangladesh: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.04
0.03
0.02
0.01
0.00
1960
1970
1980
1990
2000
2010

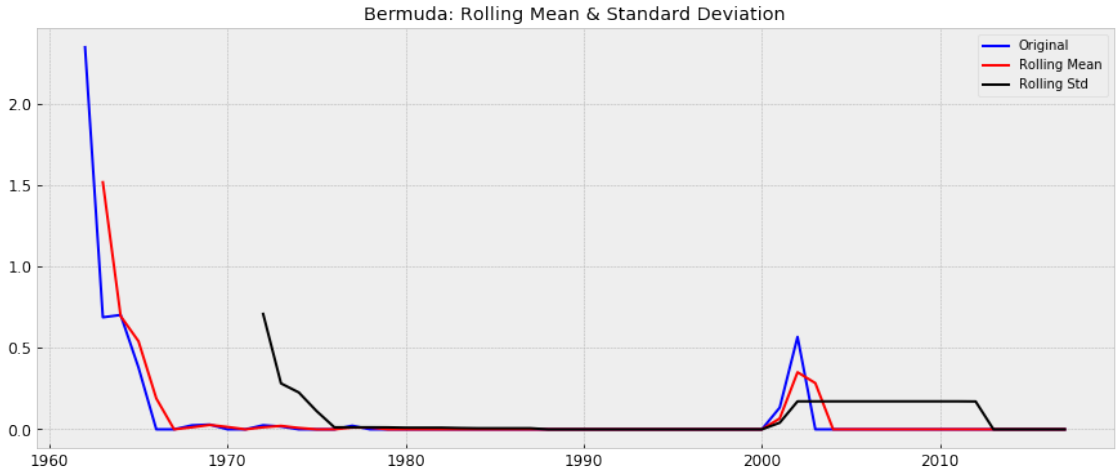

Barbados: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
1960
1970
1980
1990
2000
2010

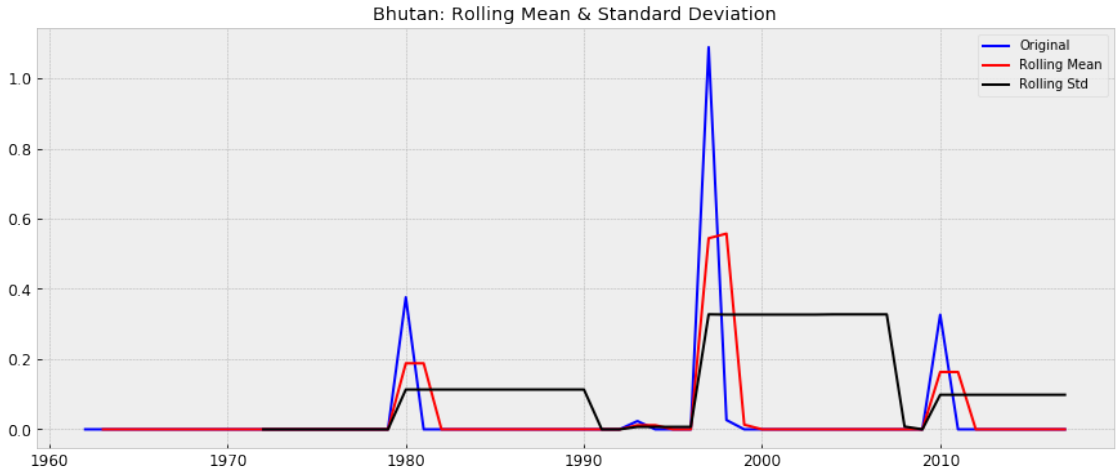

Belarus: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
2.0
1.5
1.0
0.5
0.0
1960
1970
1980
1990
2000
2010

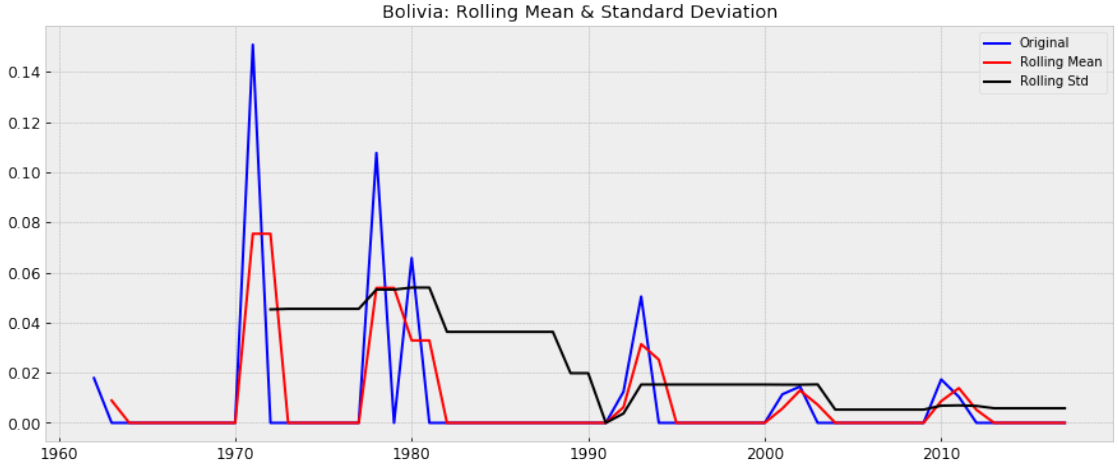

Belgium: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.4
0.3
0.2
0.1
0.0
1960
1970
1980
1990
2000
2010

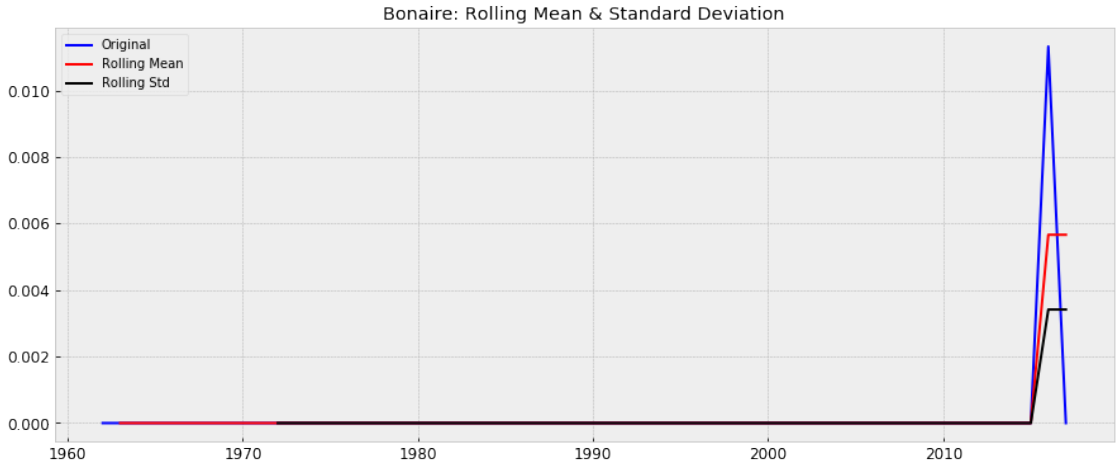

Belize: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1960
1970
1980
1990
2000
2010

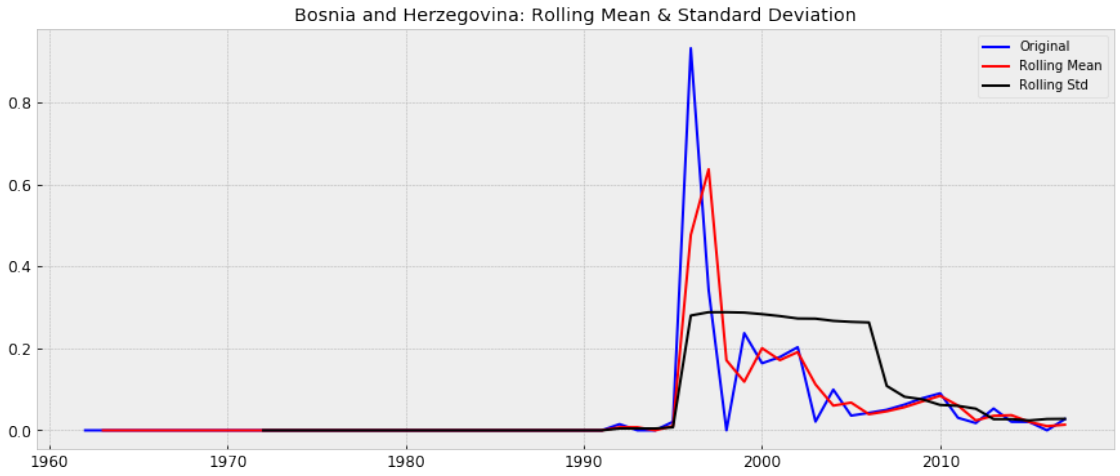

Benin: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010

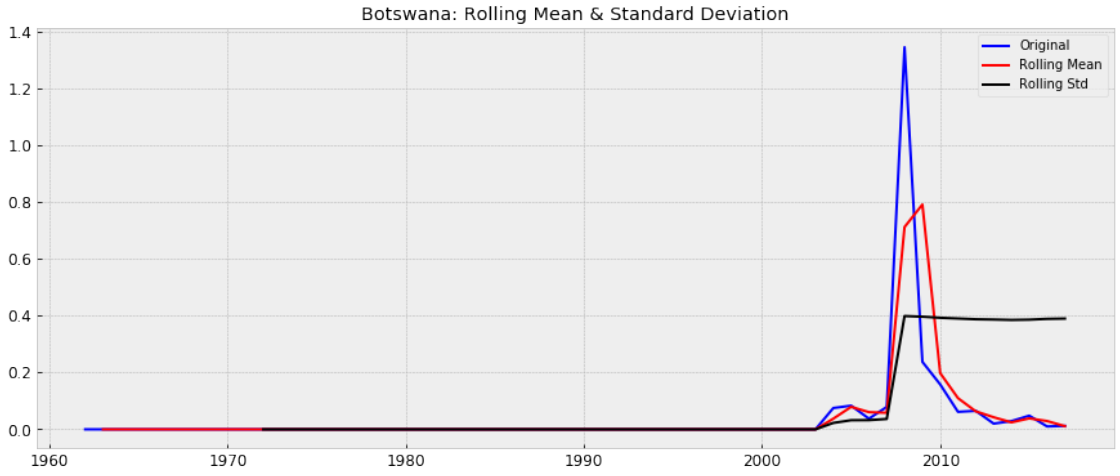

Bermuda: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
2.0
1.5
1.0
0.5
0.0
1960
1970
1980
1990
2000
2010

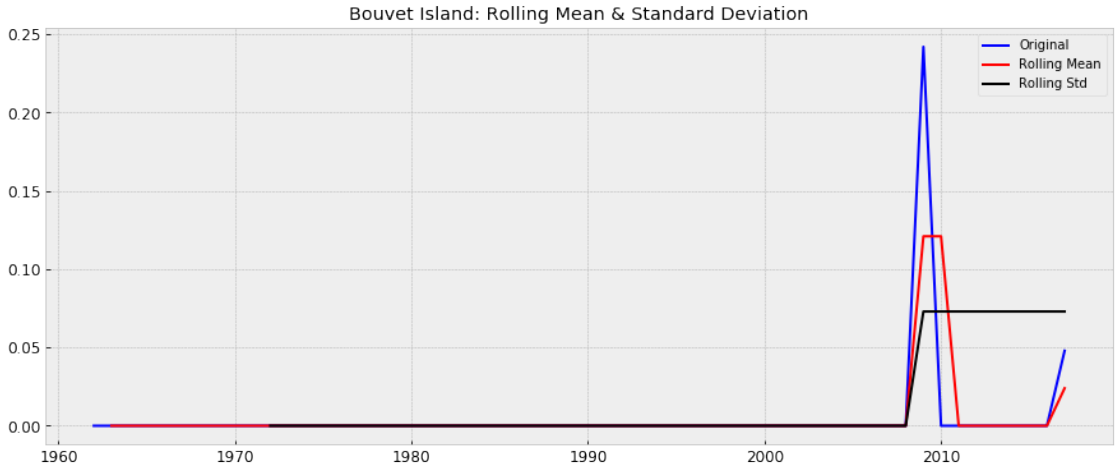

Bhutan: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
1.0
0.8
0.6
0.4
0.2
0.0
1960
1970
1980
1990
2000
2010

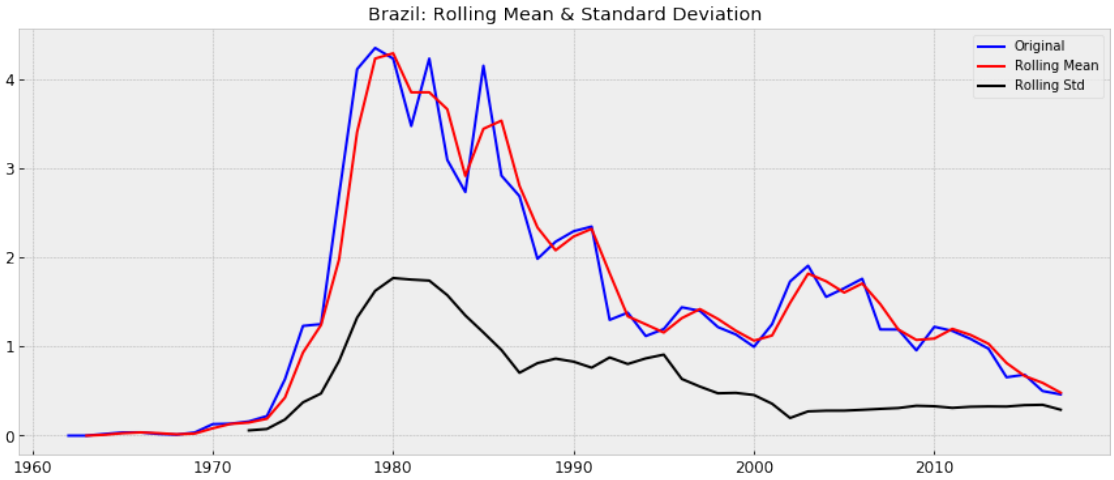

Bolivia: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

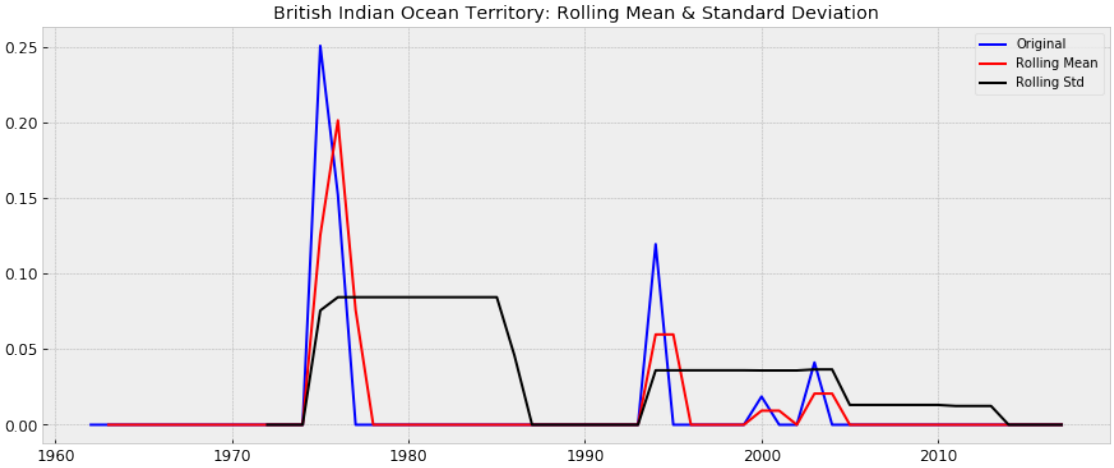

Bonaire: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.010
0.008
0.006
0.004
0.002
0.000
1960
1970
1980
1990
2000
2010

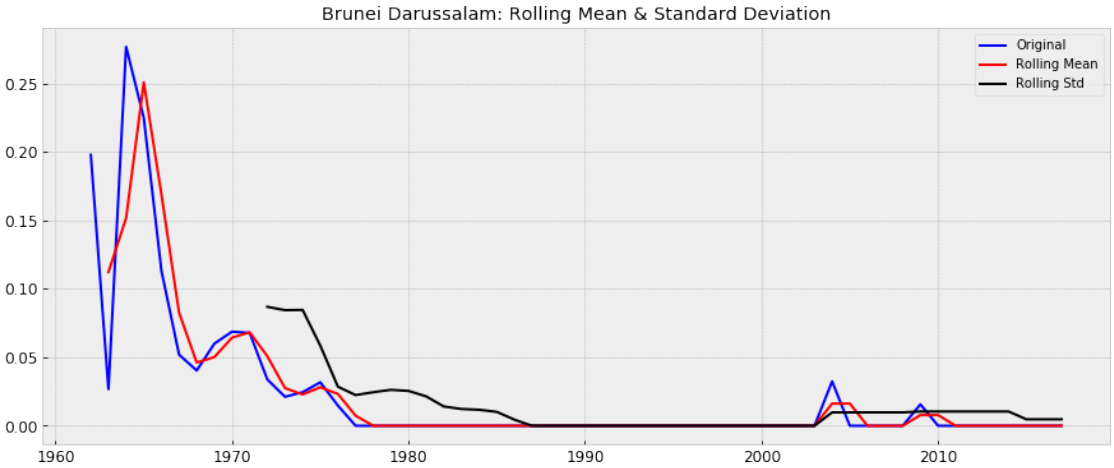

Bosnia and Herzegovina: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.0
0.2
0.4
0.6
0.8
1960
1970
1980
1990
2000
2010

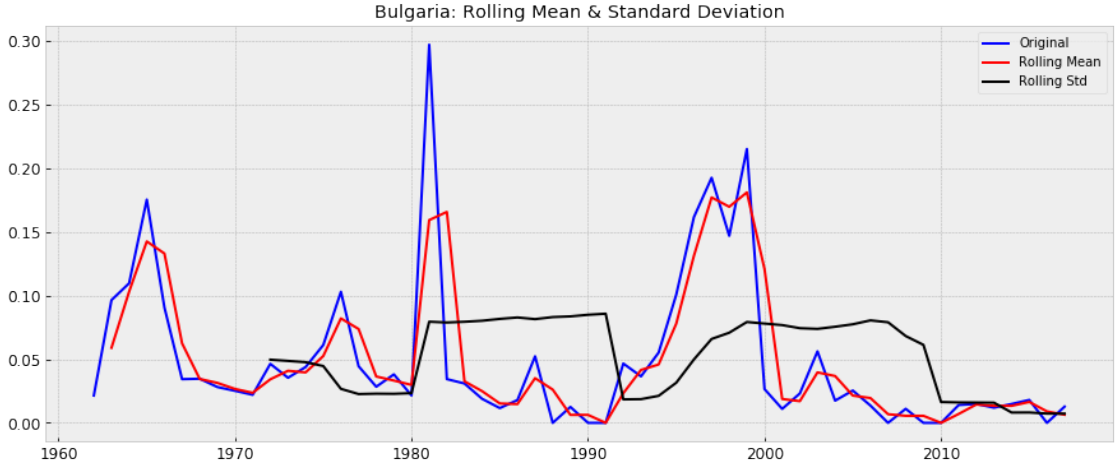

Botswana: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1960
1970
1980
1990
2000
2010

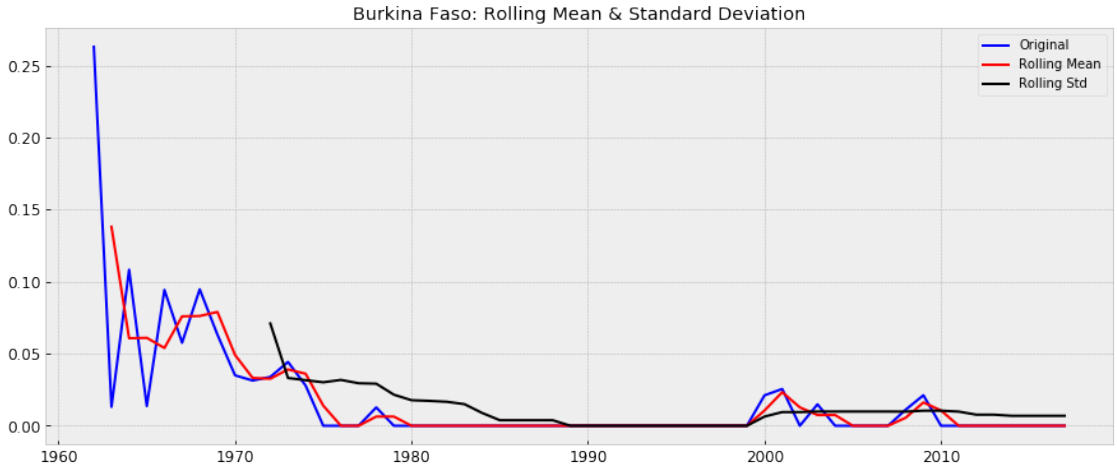

Bouvet Island: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.00
0.05
0.10
0.15
0.20
0.25
1960
1970
1980
1990
2000
2010

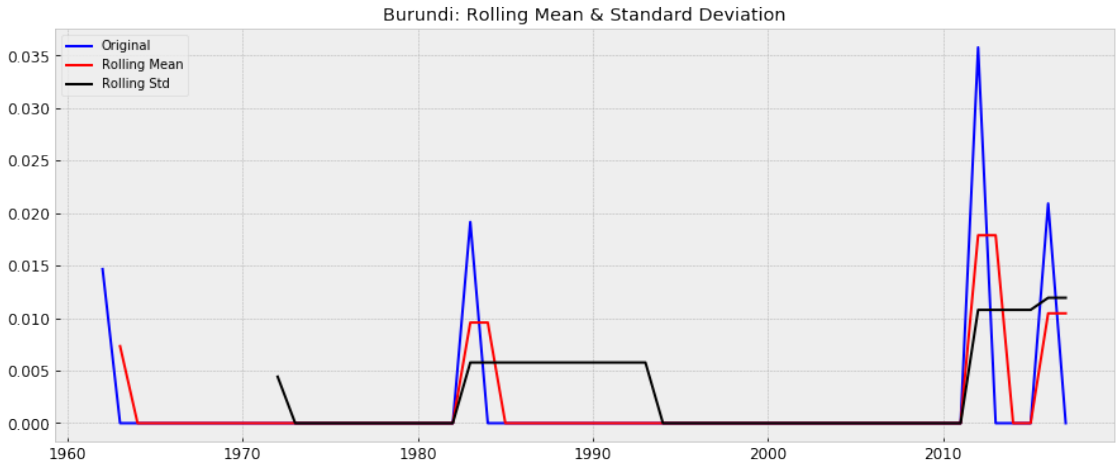

Brazil: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0
1
2
3
4
1960
1970
1980
1990
2000
2010

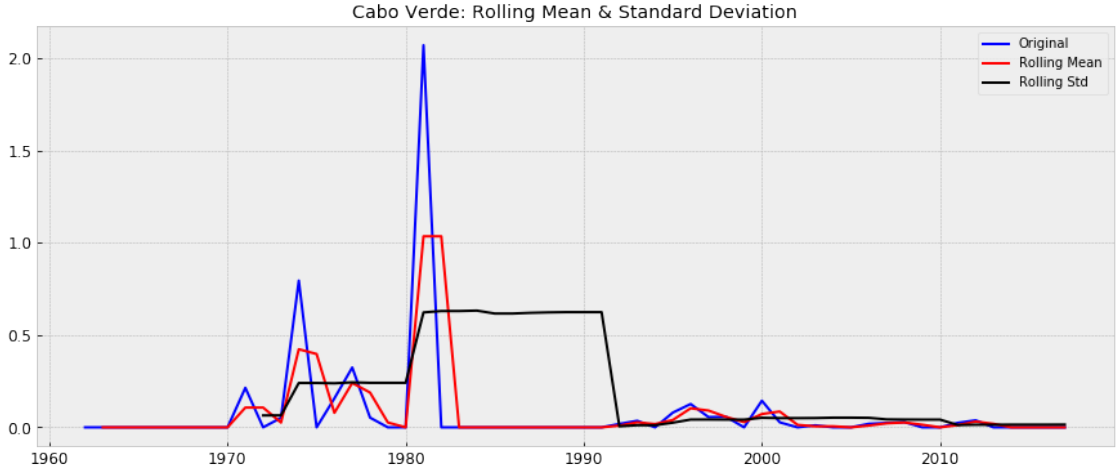

British Indian Ocean Territory: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.00
0.05
0.10
0.15
0.20
0.25
1960
1970
1980
1990
2000
2010

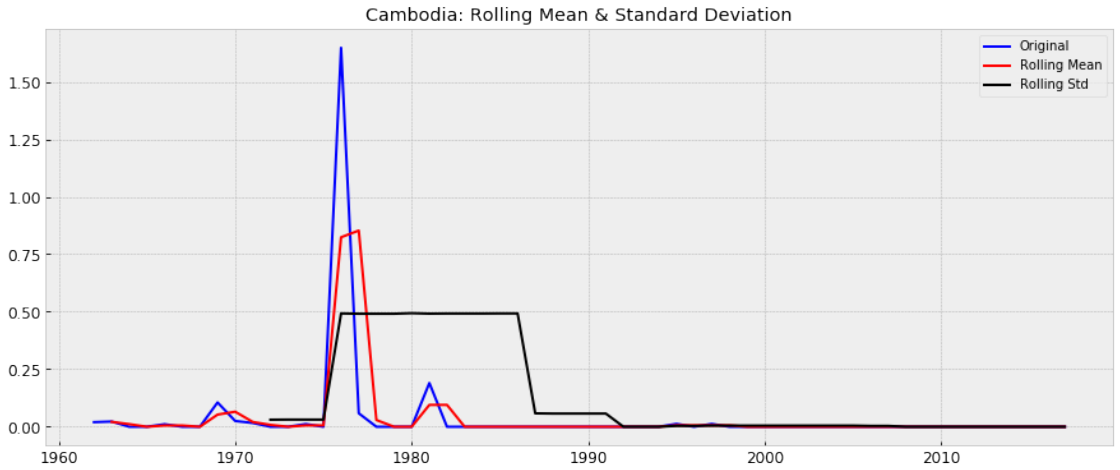

Brunei Darussalam: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.00
0.05
0.10
0.15
0.20
0.25
1960
1970
1980
1990
2000
2010

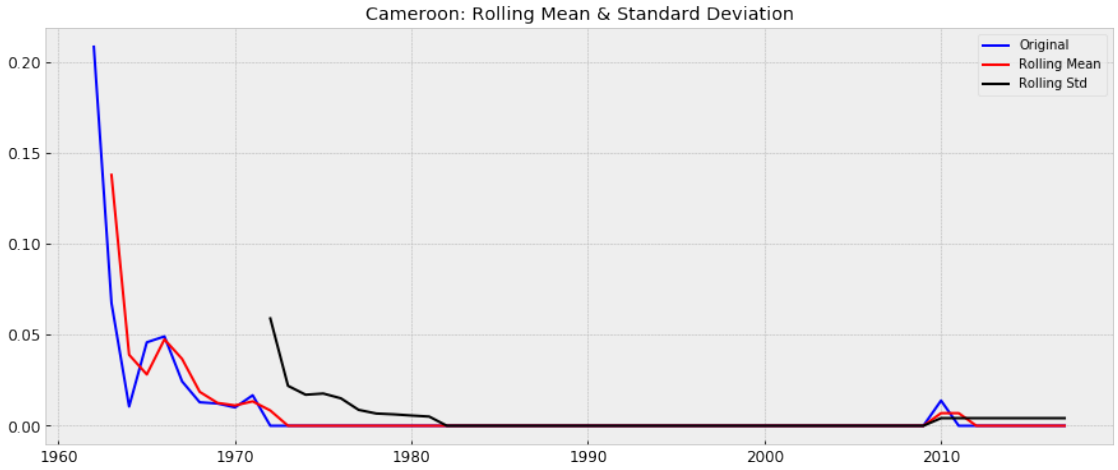

Bulgaria: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010

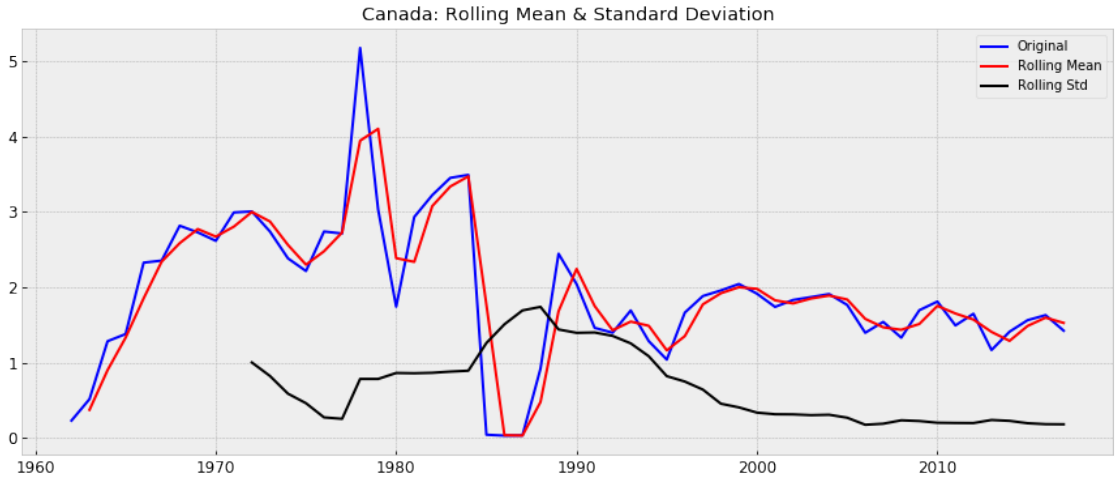

Burkina Faso: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.25
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010

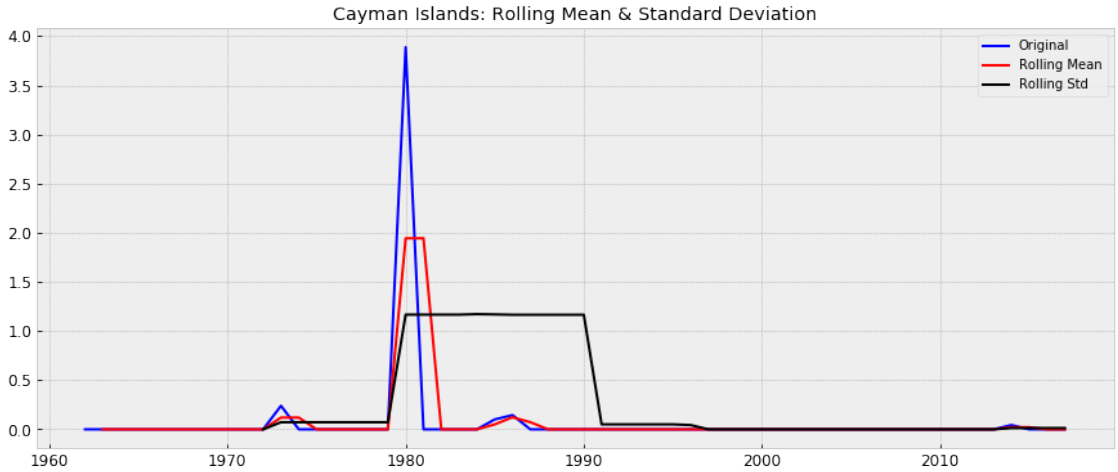

Burundi: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
1960
1970
1980
1990
2000
2010

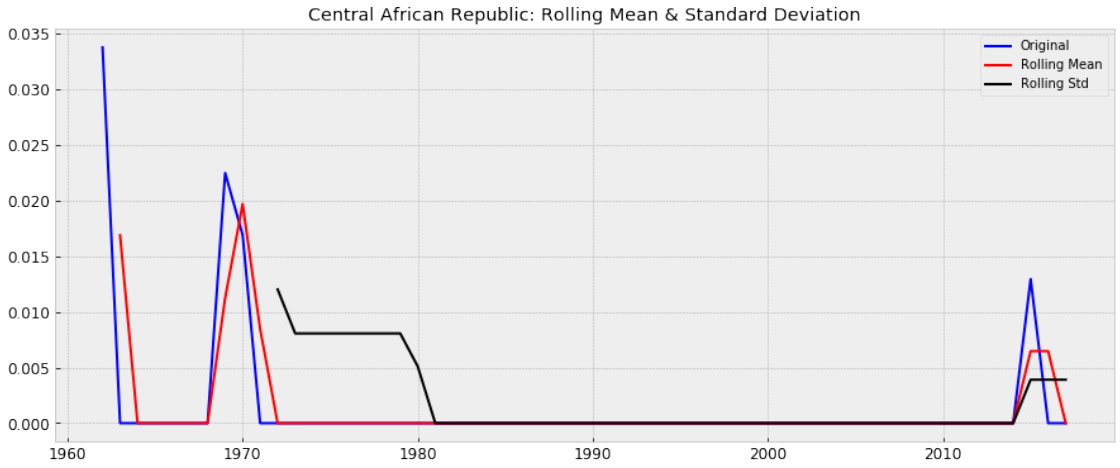

Cabo Verde: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
2.0
1.5
1.0
0.5
0.0
1960
1970
1980
1990
2000
2010

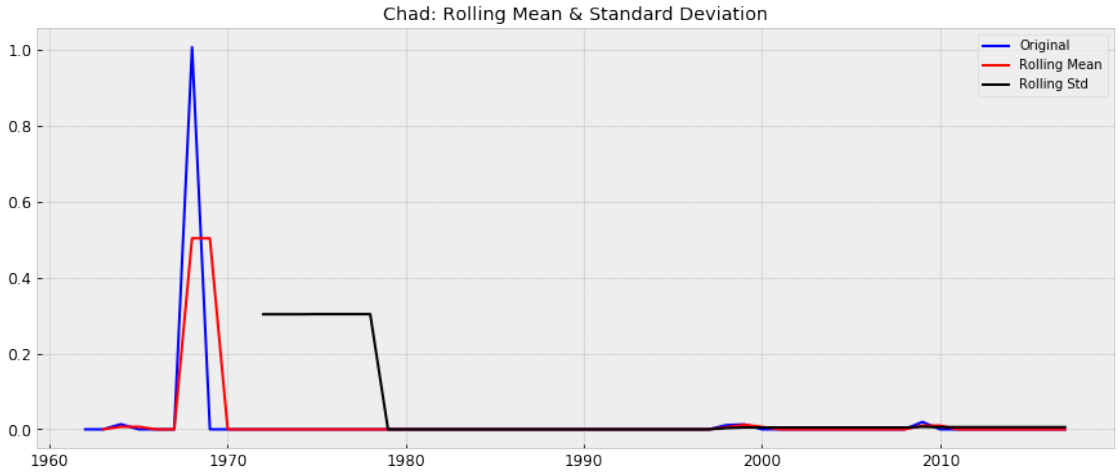

Cambodia: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
1.50
1.25
1.00
0.75
0.50
0.25
0.00
1960
1970
1980
1990
2000
2010

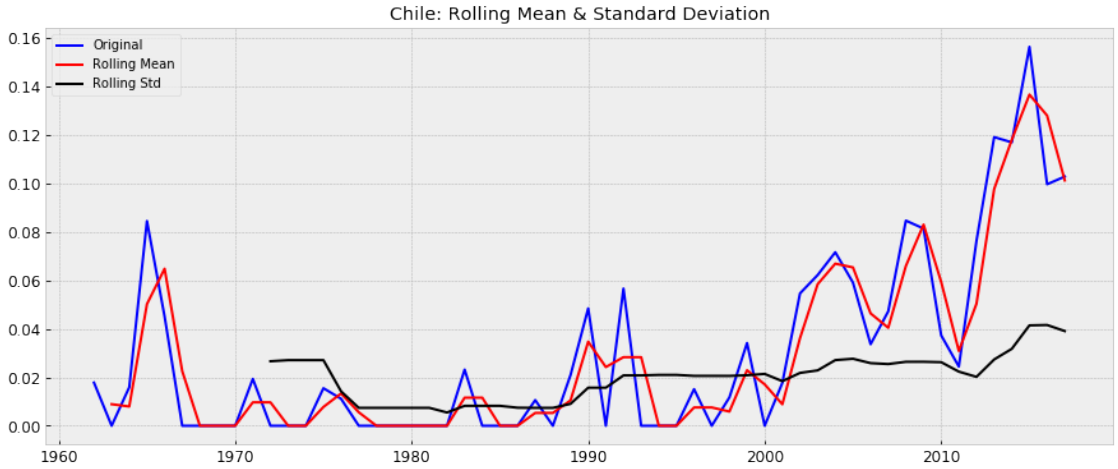

Cameroon: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010

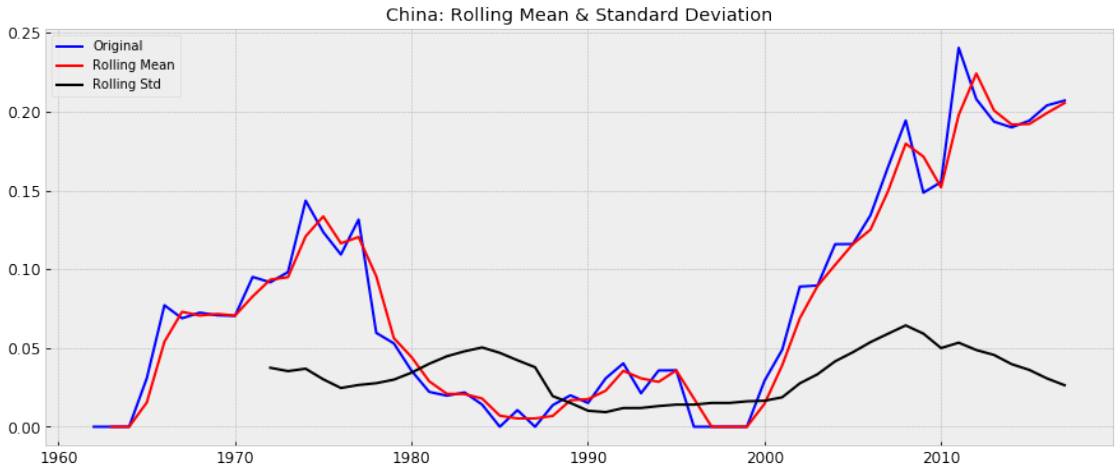

Canada: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
5
4
3
2
1
0
1960
1970
1980
1990
2000
2010

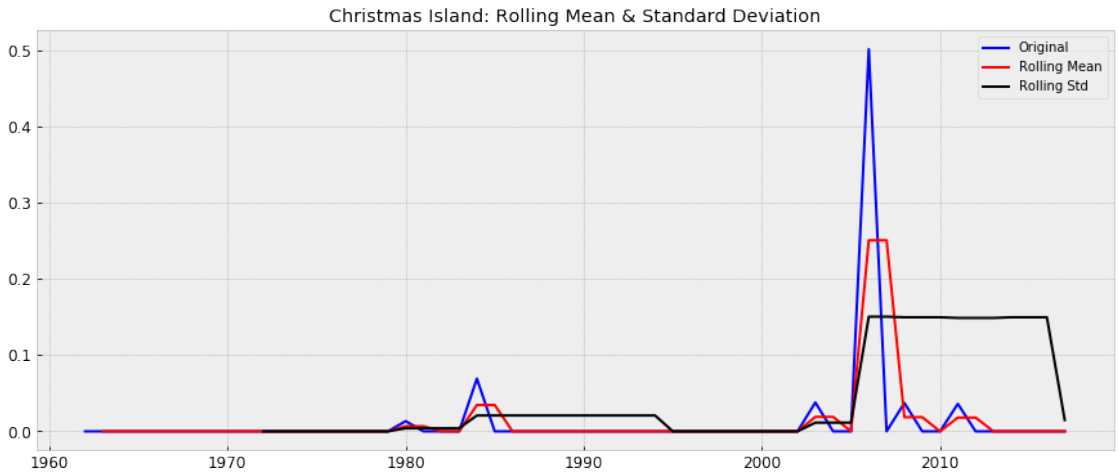

Cayman Islands: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1960
1970
1980
1990
2000
2010

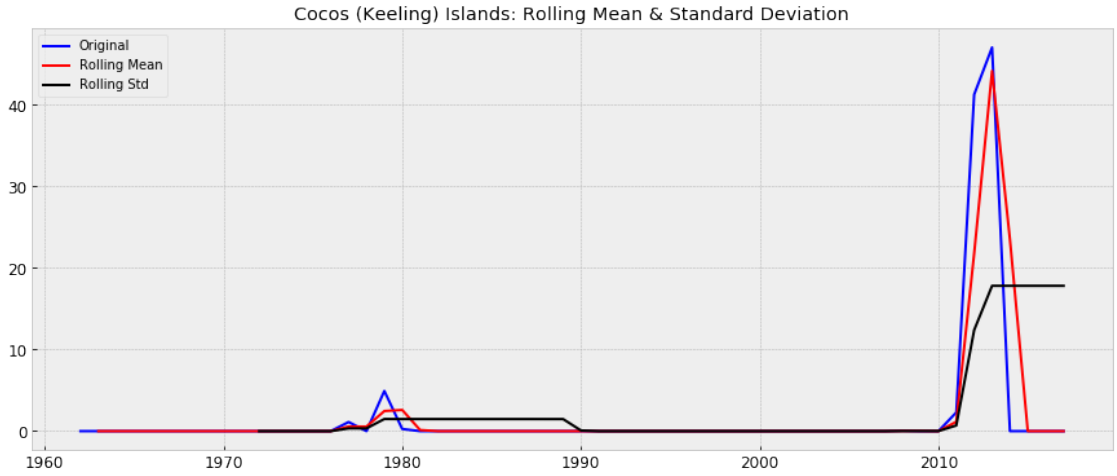

Central African Republic: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
1960
1970
1980
1990
2000
2010

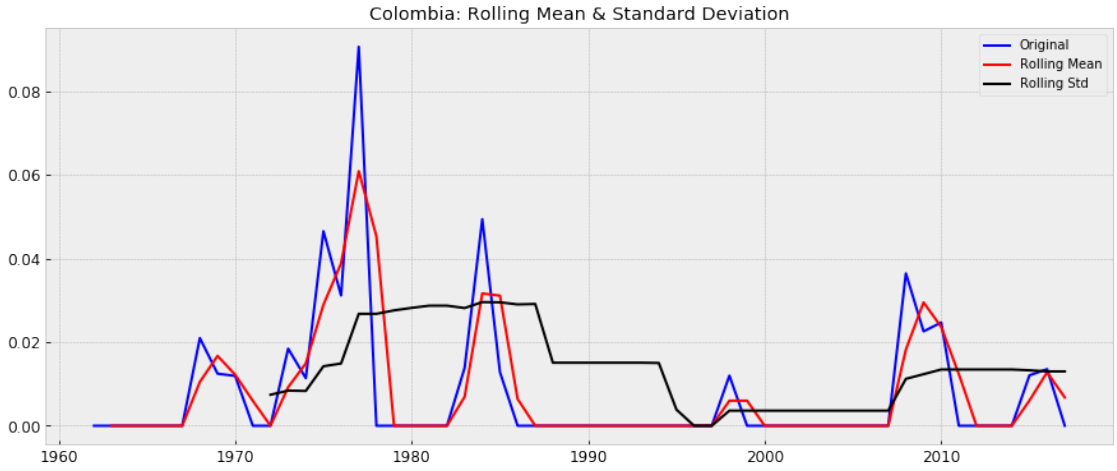

Chad: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
1.0
0.8
0.6
0.4
0.2
0.0
1960
1970
1980
1990
2000
2010

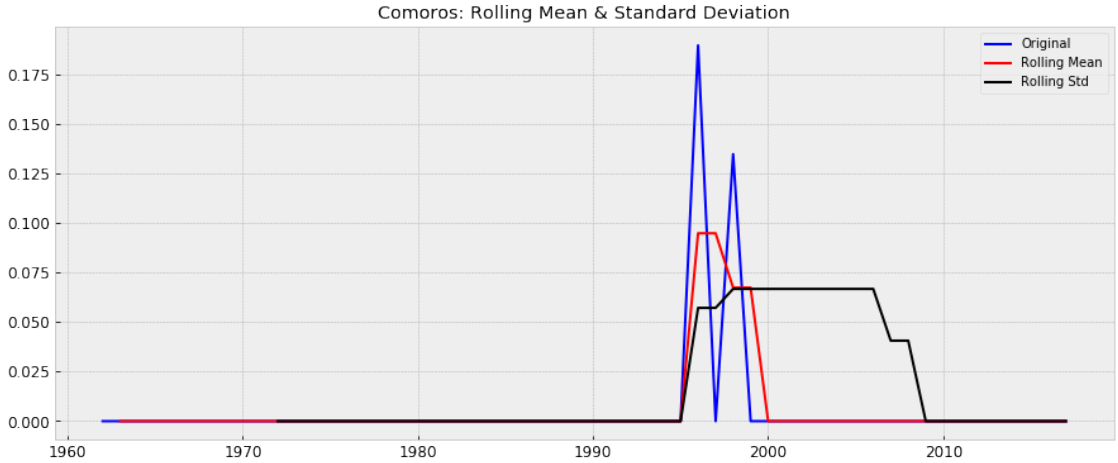

Chile: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

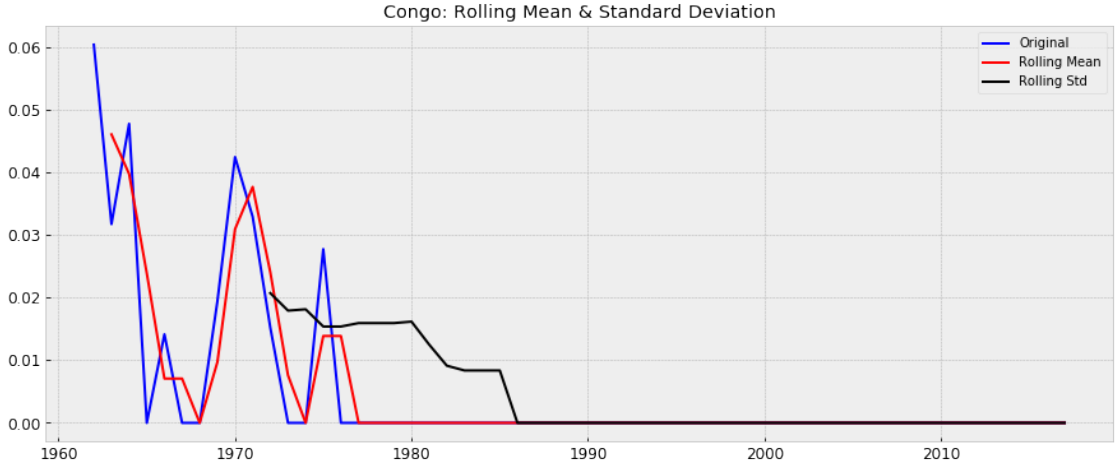

China: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.25
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010

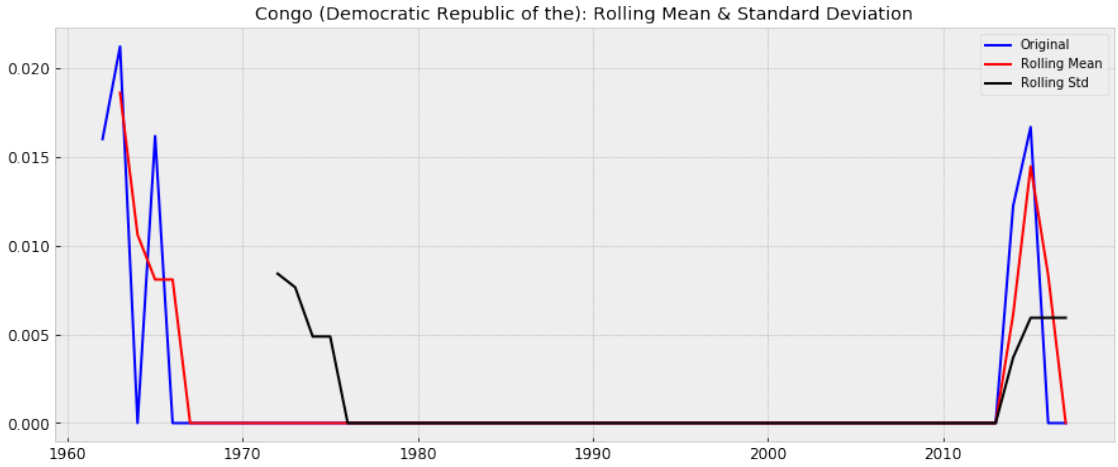

Christmas Island: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.5
0.4
0.3
0.2
0.1
0.0
1960
1970
1980
1990
2000
2010

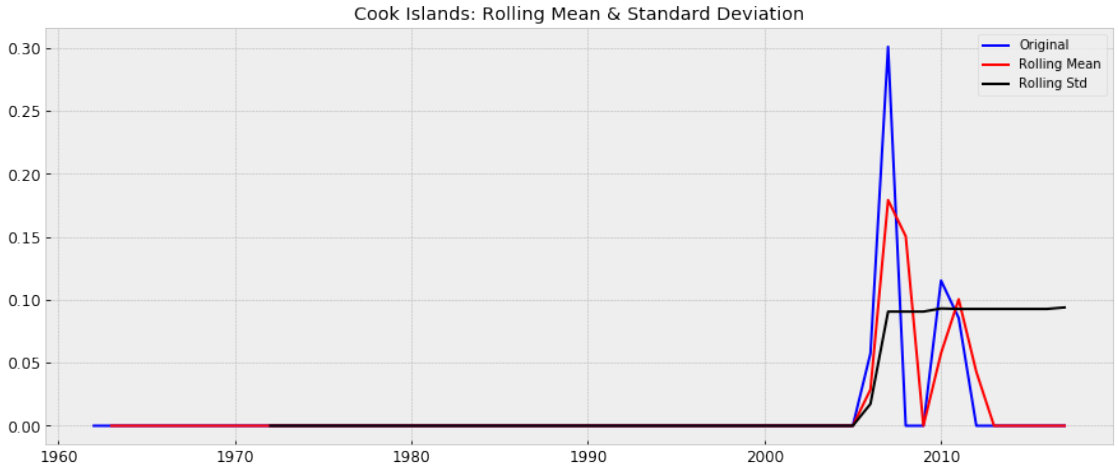

Cocos (Keeling) Islands: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
40
30
20
10
0
1960
1970
1980
1990
2000
2010

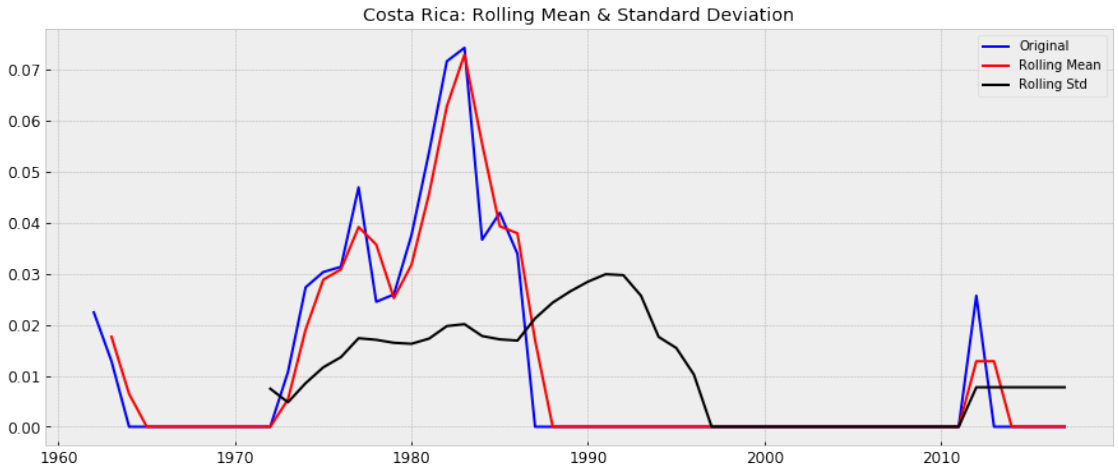

Colombia: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

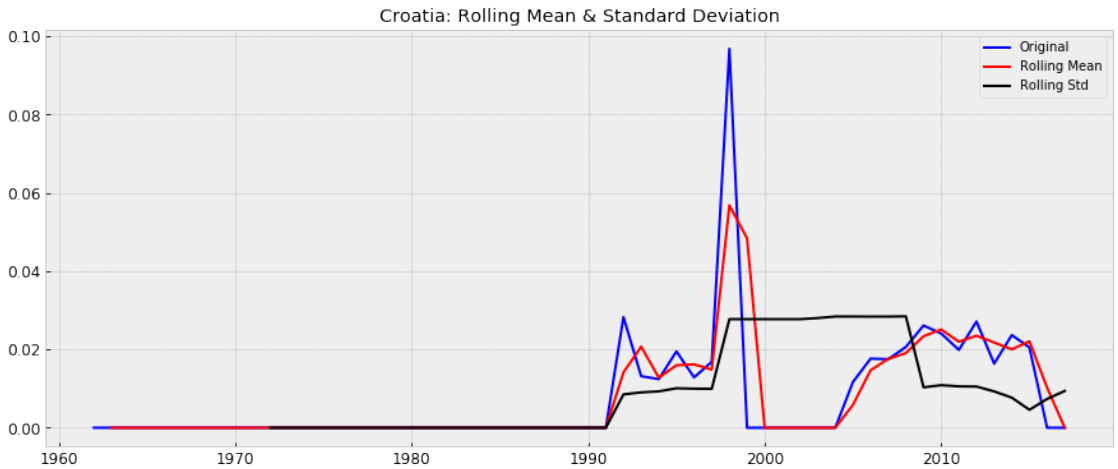

Comoros: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
1960
1970
1980
1990
2000
2010

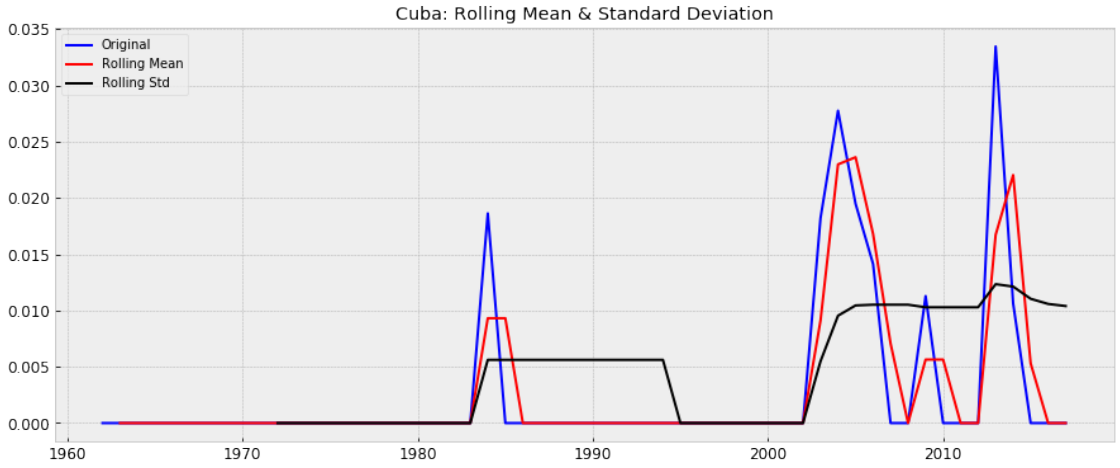

Congo: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.06
0.05
0.04
0.03
0.02
0.01
0.00
1960
1970
1980
1990
2000
2010

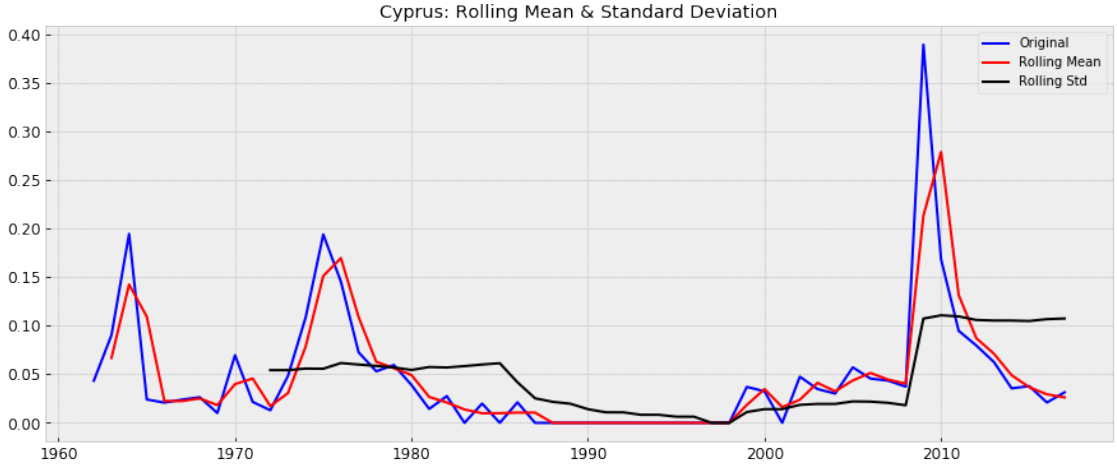

Congo (Democratic Republic of the): Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.020
0.015
0.010
0.005
0.000
1960
1970
1980
1990
2000
2010

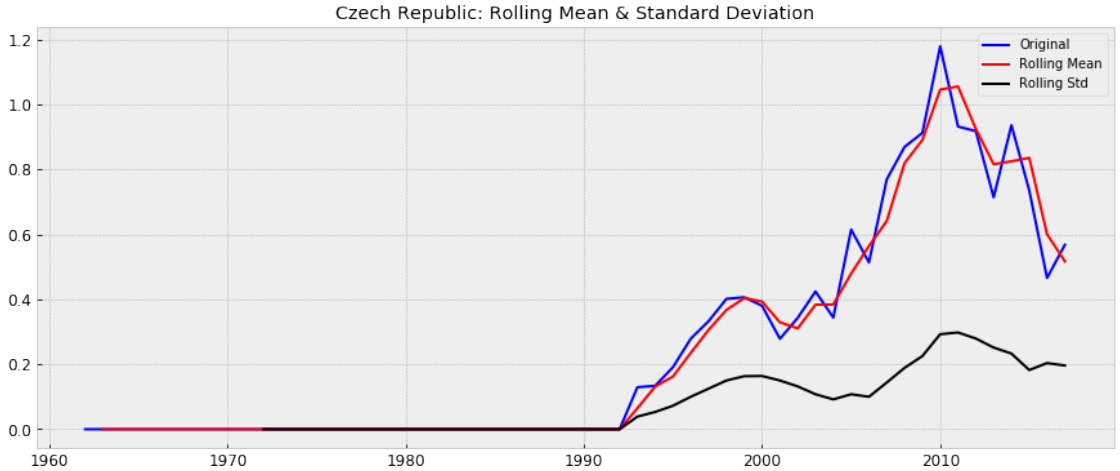

Cook Islands: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010

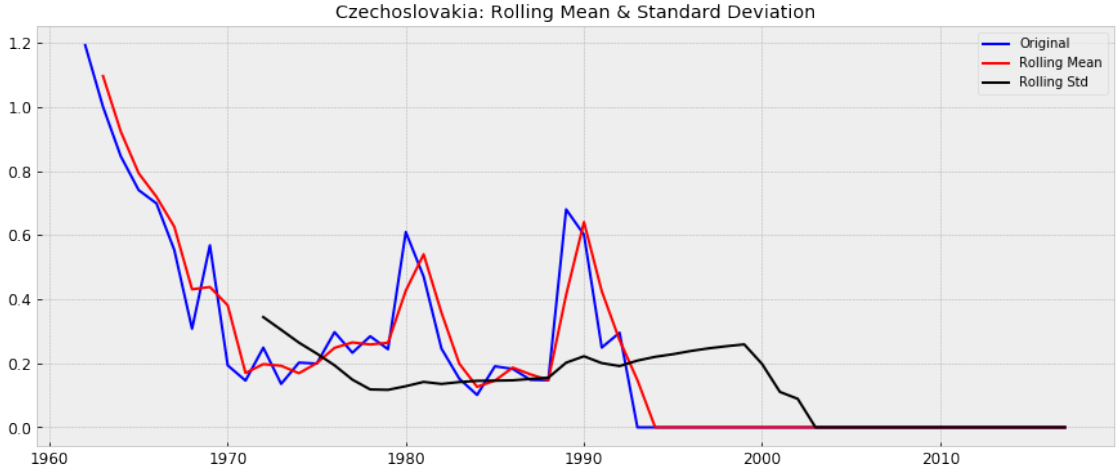

Costa Rica: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
1960
1970
1980
1990
2000
2010

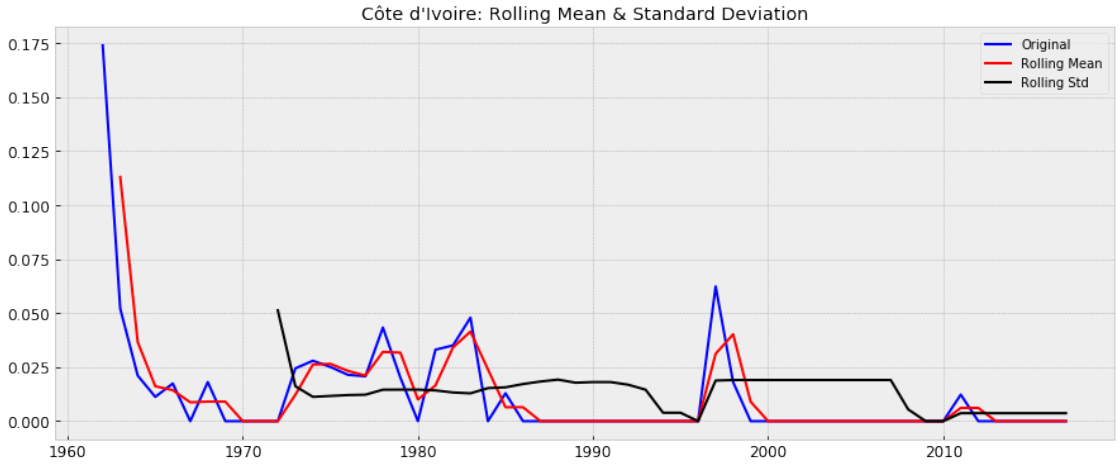

Croatia: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.10
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

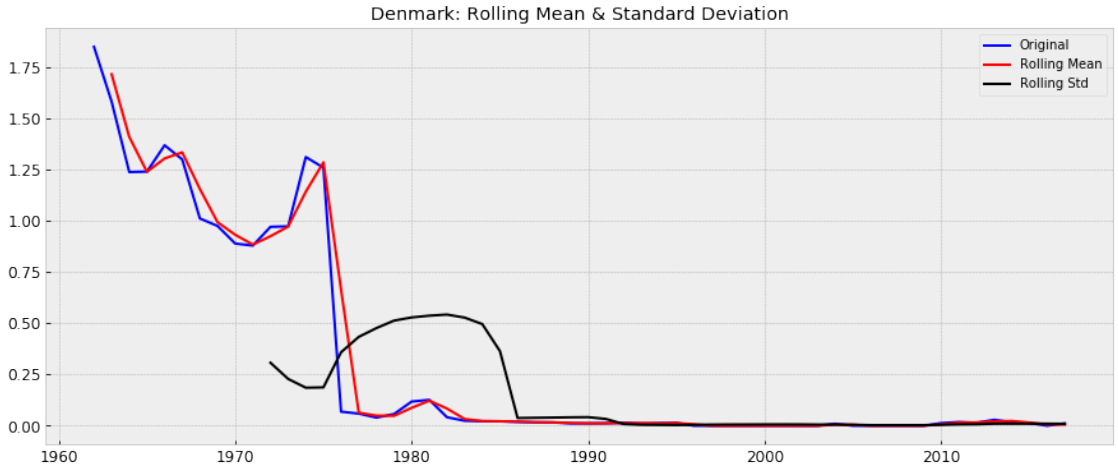

Cuba: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
1960
1970
1980
1990
2000
2010

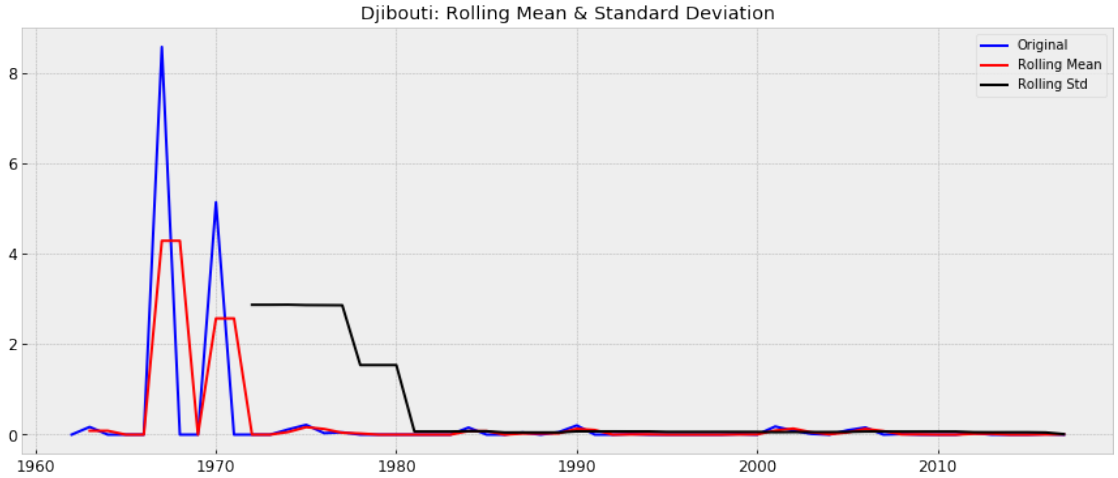

Cyprus: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010

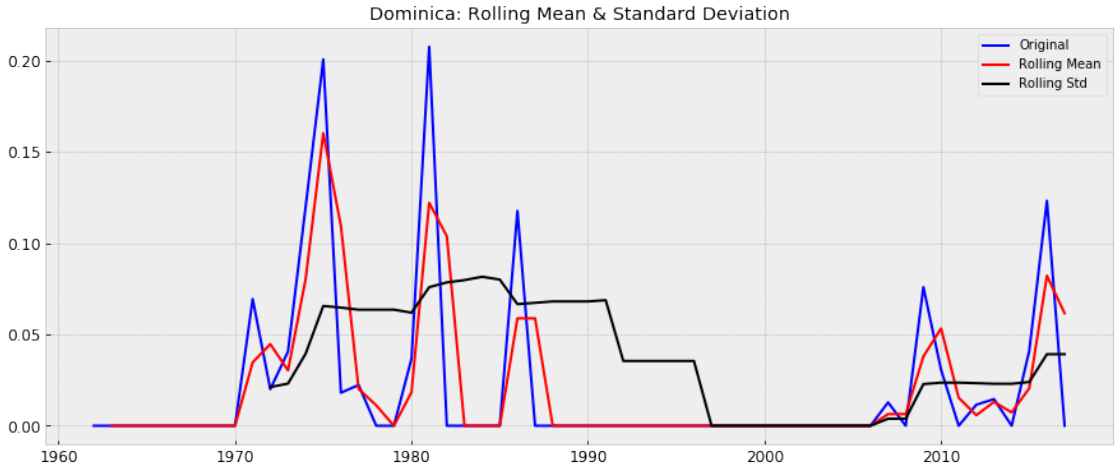

Czech Republic: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1960
1970
1980
1990
2000
2010

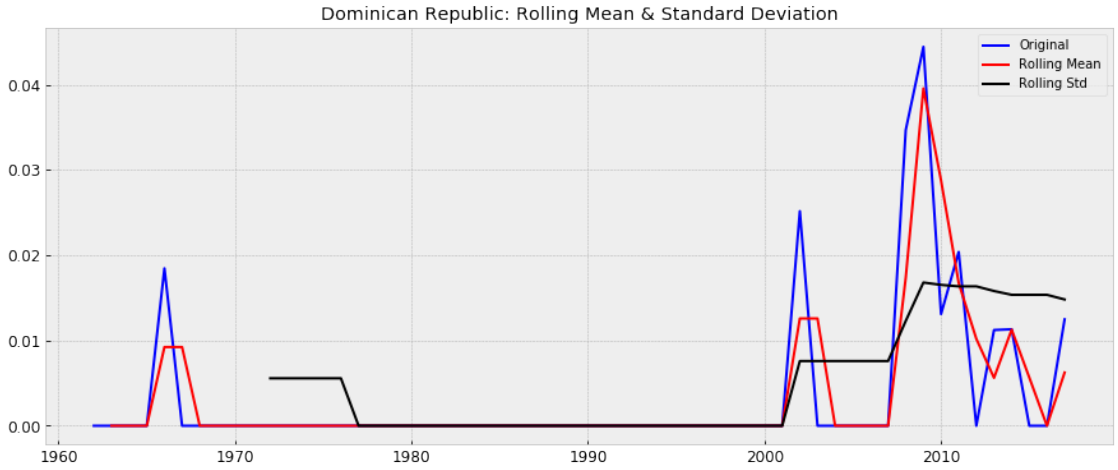

Czechoslovakia: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1960
1970
1980
1990
2000
2010

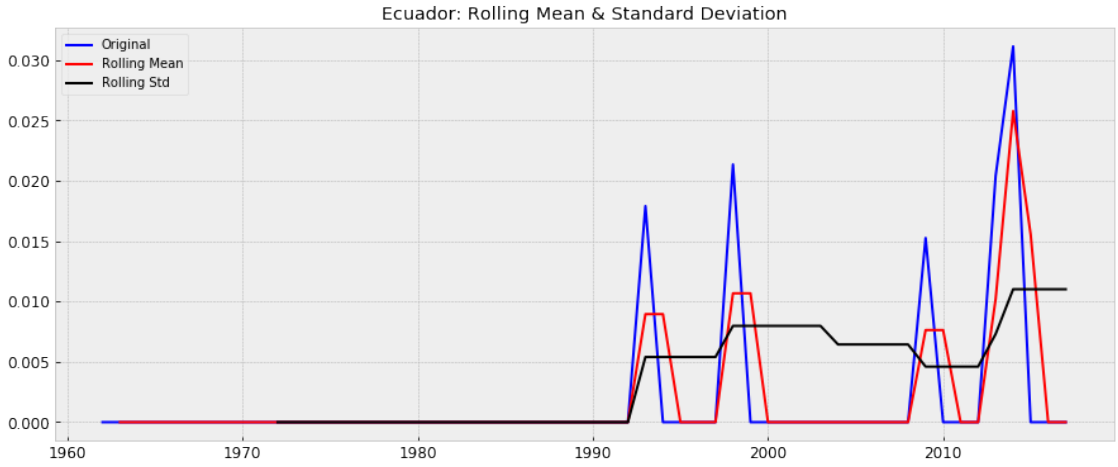

Côte d'Ivoire: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
1960
1970
1980
1990
2000
2010

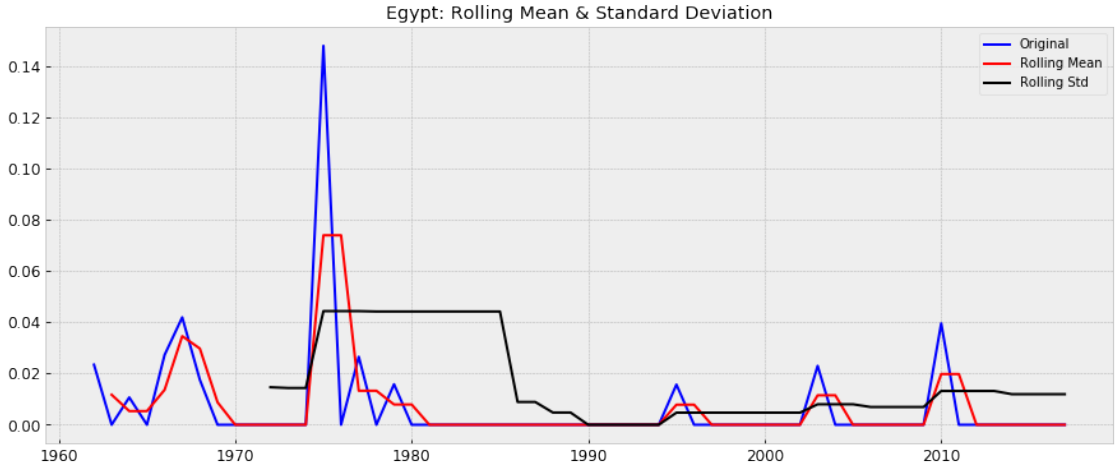

Denmark: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
1960
1970
1980
1990
2000
2010

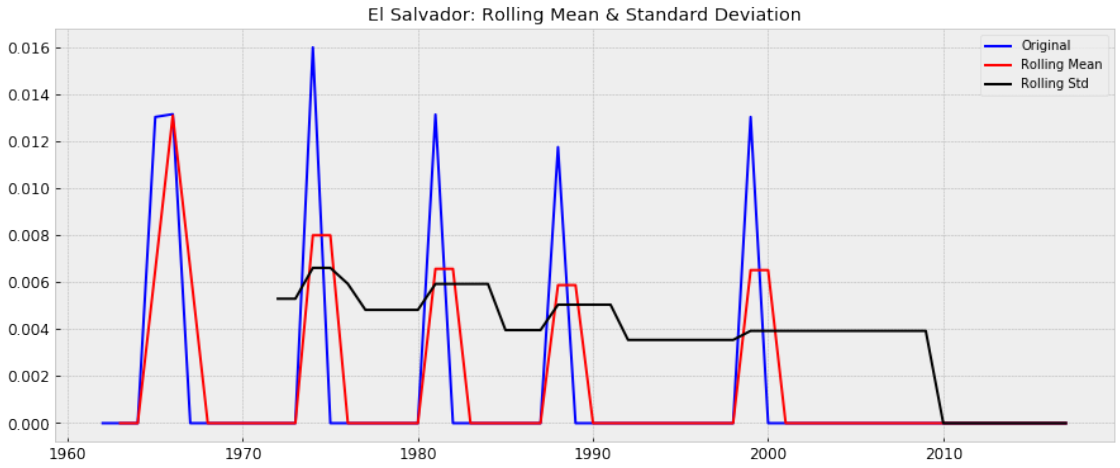

Djibouti: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
8
6
4
2
0
1960
1970
1980
1990
2000
2010

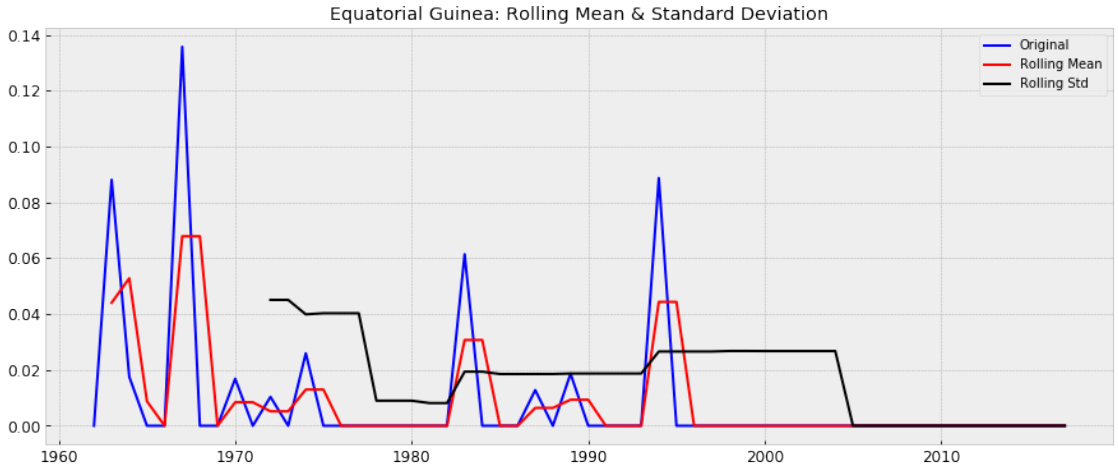

Dominica: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.20
0.15
0.10
0.05
0.00
1960
1970
1980
1990
2000
2010

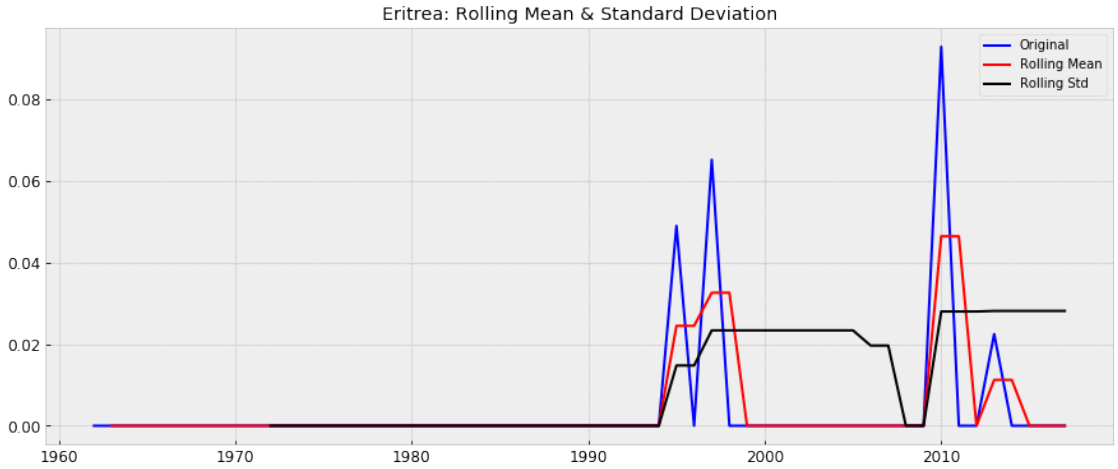

Dominican Republic: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.04
0.03
0.02
0.01
0.00
1960
1970
1980
1990
2000
2010

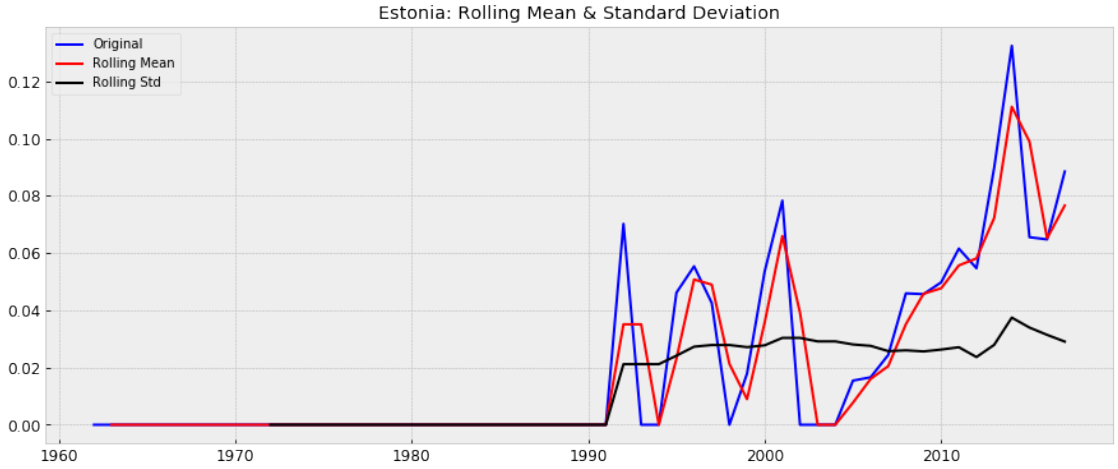

Ecuador: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.030
0.025
0.020
0.015
0.010
0.005
0.000
1960
1970
1980
1990
2000
2010

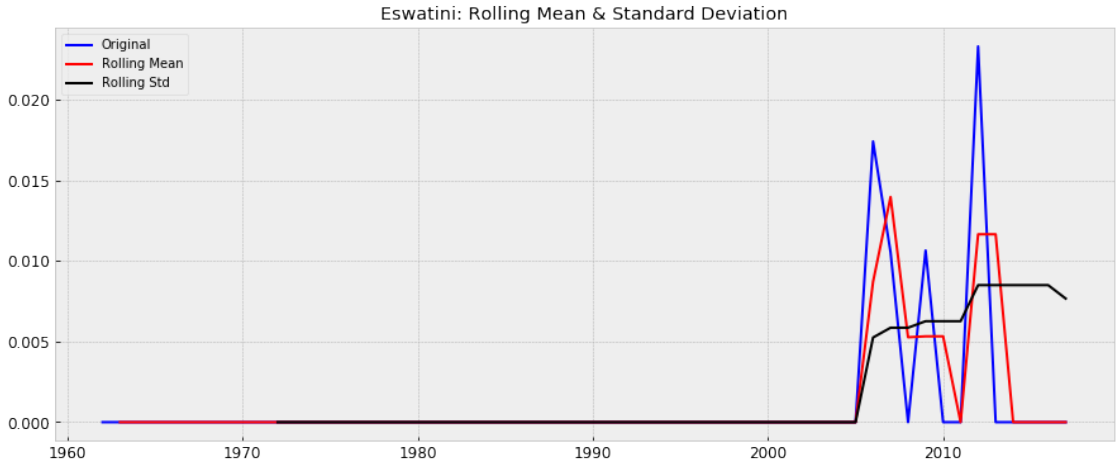

Egypt: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

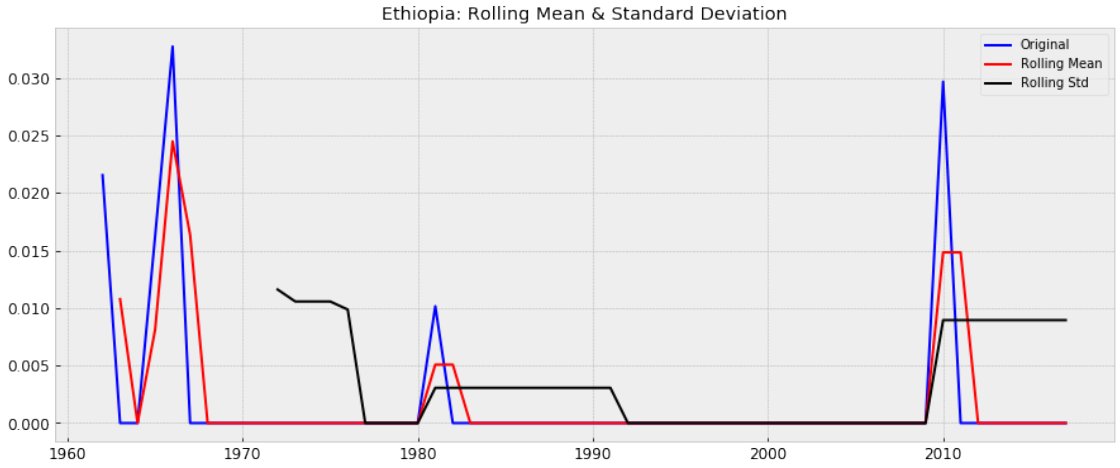

El Salvador: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
1960
1970
1980
1990
2000
2010

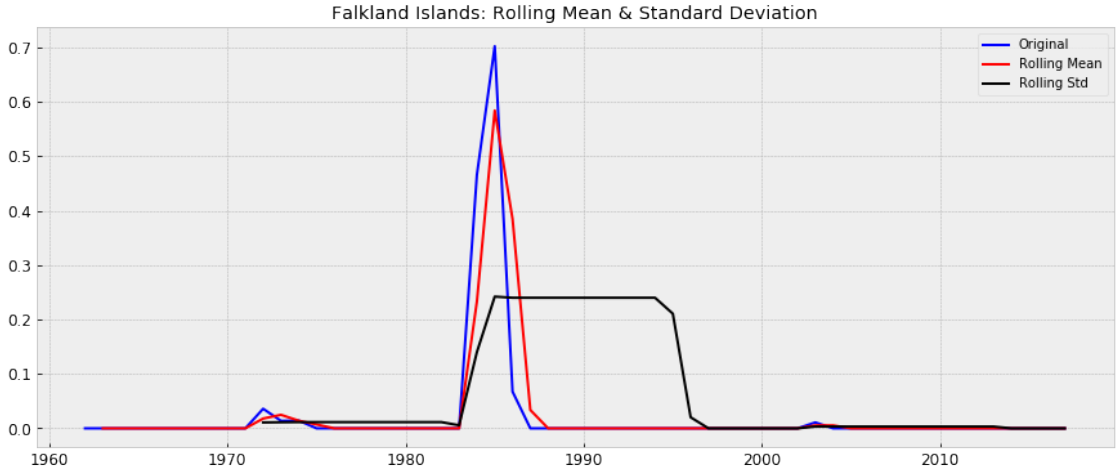

Equatorial Guinea: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

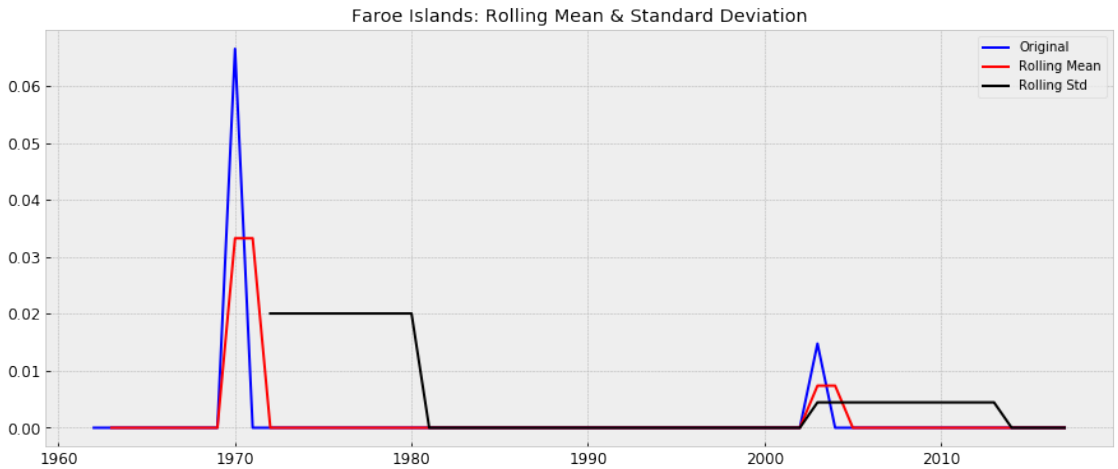

Eritrea: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

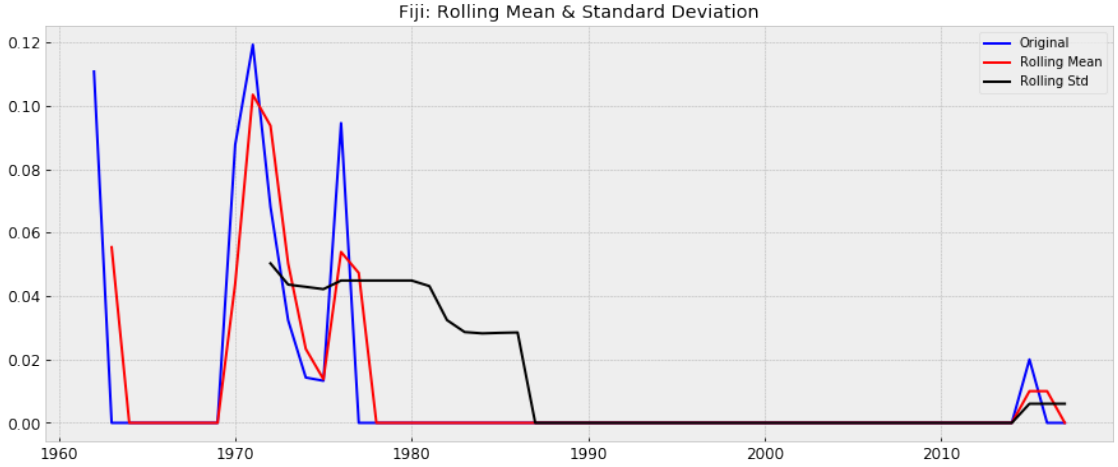

Estonia: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.12
0.10
0.08
0.06
0.04
0.02
0.00
1960
1970
1980
1990
2000
2010

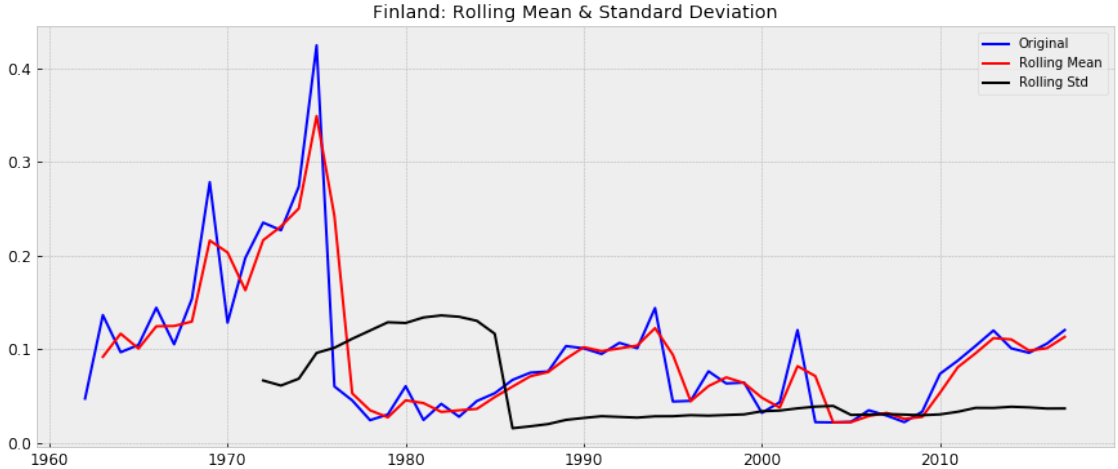

Eswatini: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.020
0.015
0.010
0.005
0.000
1960
1970
1980
1990
2000
2010


Ethiopia: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.030
0.025
0.020
0.015
0.010
0.005
0.000
1960
1970
1980
1990
2000
2010


Falkland Islands: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
1960
1970
1980
1990
2000
2010

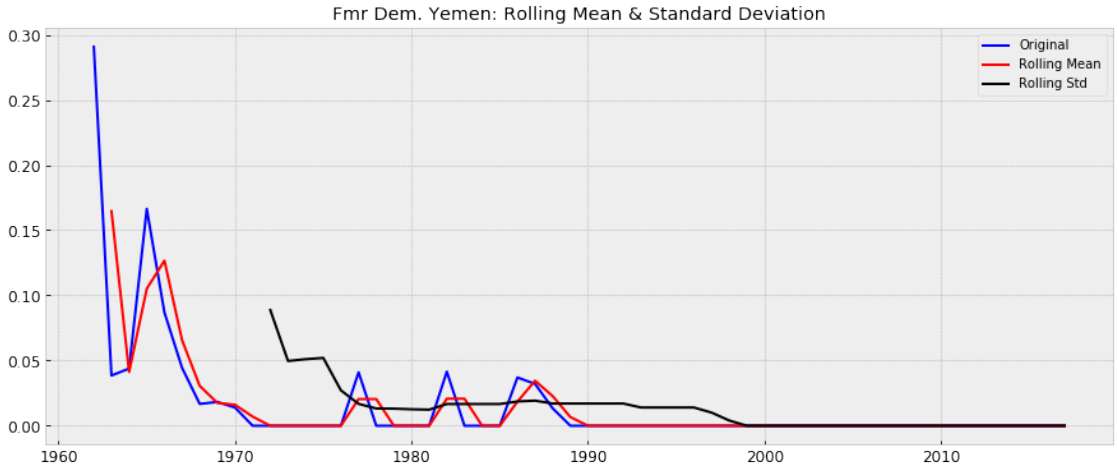

Faroe Islands: Rolling Mean & Standard Deviation
Original
Rolling Mean
Rolling Std
0.06
0.05
0.04
0.03
0.02
0.01
0.00
1960
1970
1980
1990
2000
2010






```
␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-64-008d5767e479> in <module>
        16     plt.legend(loc='best')
        17     plt.title('{}: Rolling Mean & Standard Deviation'.
↪format(df_grps[country].name))
   ---> 18     pyplot.savefig(str('{}_Rolling_Stats'.format(savefile)))
```

```
        19     pyplot.show(block=False)


       ~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args,␣
↪**kwargs)
       687 def savefig(*args, **kwargs):
       688     fig = gcf()
   --> 689     res = fig.savefig(*args, **kwargs)
       690     fig.canvas.draw_idle()   # need this if 'transparent=True' to␣
↪reset colors
       691     return res


       ~\Anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self,␣
↪fname, frameon, transparent, **kwargs)
      2092             self.set_frameon(frameon)
      2093
   -> 2094         self.canvas.print_figure(fname, **kwargs)
      2095
      2096         if frameon:


       ~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in␣
↪print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format,␣
↪bbox_inches, **kwargs)
      2004
      2005         # get canvas object and print method for format
   -> 2006         canvas = self._get_output_canvas(format)
      2007         print_method = getattr(canvas, 'print_%s' % format)
      2008


       ~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in␣
↪_get_output_canvas(self, fmt)
      1946         raise ValueError(
      1947             "Format {!r} is not supported (supported formats: {})"
   -> 1948             .format(fmt, ", ".join(sorted(self.
↪get_supported_filetypes()))))
      1949
      1950     def print_figure(self, filename, dpi=None, facecolor=None,␣
↪edgecolor=None,


       ValueError: Format ' yemen_rolling_stats' is not supported (supported␣
↪formats: eps, jpeg, jpg, pdf, pgf, png, ps, raw, rgba, svg, svgz, tif, tiff)
```



3.2.2 Augmented Dickey-Fuller (ADF) Test

- The intuition behind the test is that if the series is integrated then the lagged level of the series y(t-1) will provide no relevant information in predicting the change in y(t).

Null hypothesis (Ho): The time series is not stationary, it presents heterocedasticity. In other words, the time series depends on itself (i.e.: yt depends on yt-1, yt-1 depends on yt-2 - Rejecting the null hypothesis (i.e. a very low p-value) will indicate stationarity. Put simply, the t-test result is less than all critical values (1%,5%,10%)

Alternative hypothesis(Ha): the series is stationary

ADF results description

[85]:
```
from statsmodels.tsa.stattools import adfuller, kpss
```

[86]:
```
def ADF_Stationarity_Test(ts_groups):

    #countries = list(ts_groups.columns)
    adf_results = {}
    col_names=['Test Statistic','p_value','Lags used','Observations Used','1%␣
↪Critical value','5% Critical value','10% Critical value','IC_Best']
    data = pd.DataFrame(columns=col_names,index=countries)

    for country in countries:
        #Dickey-Fuller test
        adf_results[country] = tsa.adfuller(ts_groups[country],autolag='AIC')

    for country in adf_results:
        data.loc[country] = pd.Series({'Test Statistic':adf_results[country][0],
                                       'p_value':adf_results[country][1],
                                       'Lags used':adf_results[country][2],
```

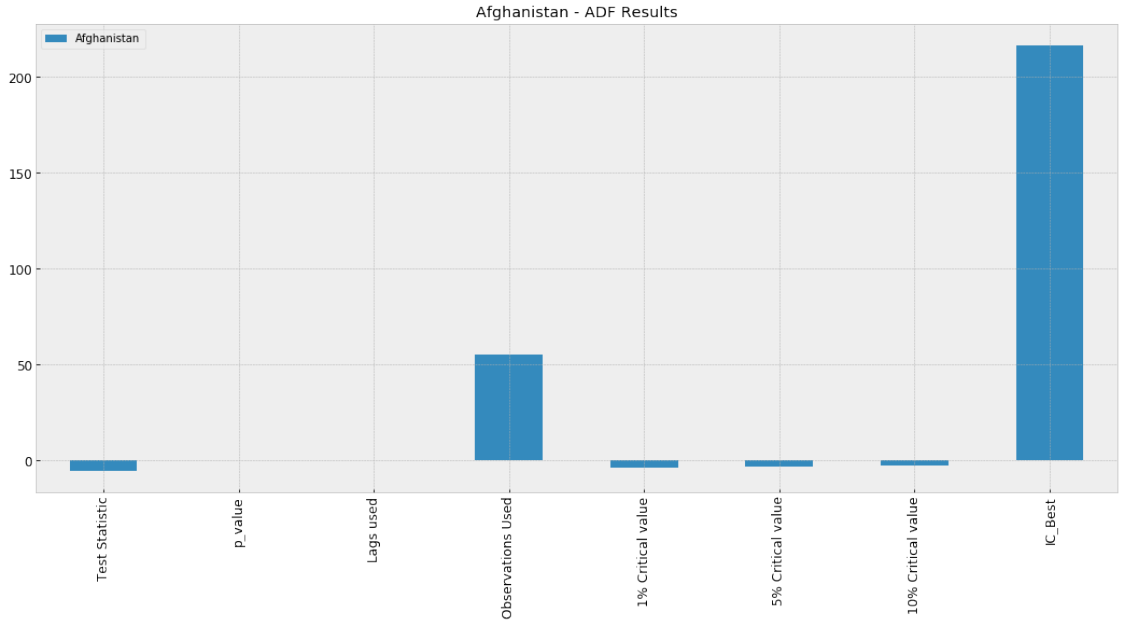
```
                                        'Observations Used':
↪adf_results[country][3],
                                        '1% Critical value':␣
↪adf_results[country][4]['1%'],
                                        '5% Critical value':␣
↪adf_results[country][4]['5%'],
                                        '10% Critical value':␣
↪adf_results[country][4]['10%'],
                                        'IC_Best': adf_results[country][5]
                                       })
    return data.transpose()
```

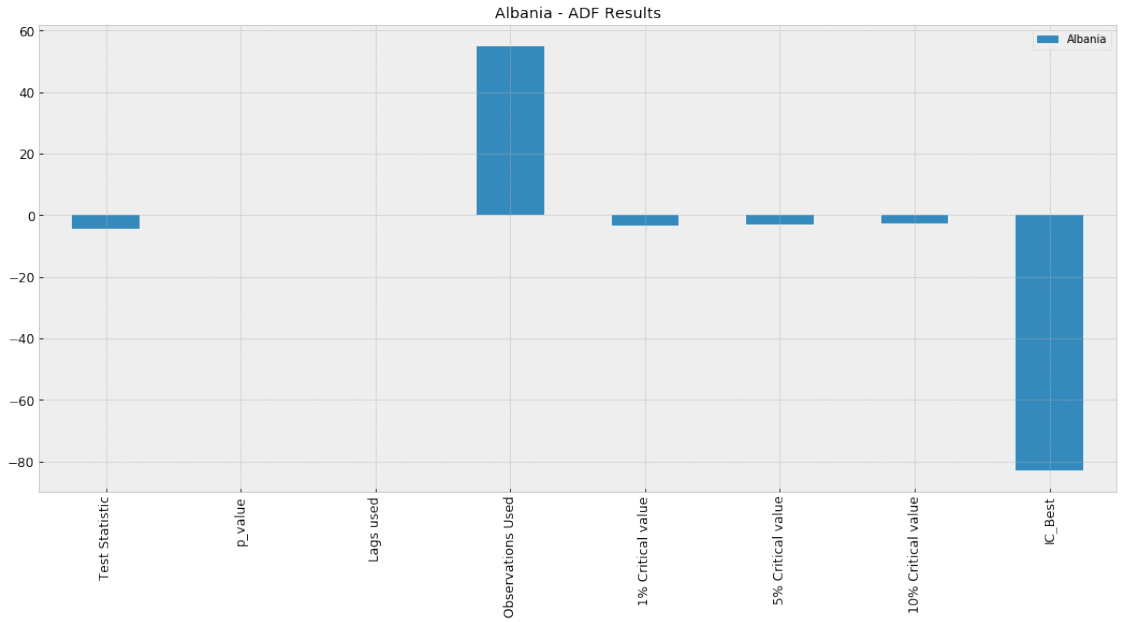
[87]:
```
adf_test = ADF_Stationarity_Test(df_grps)
adf_countries = adf_test.columns
```

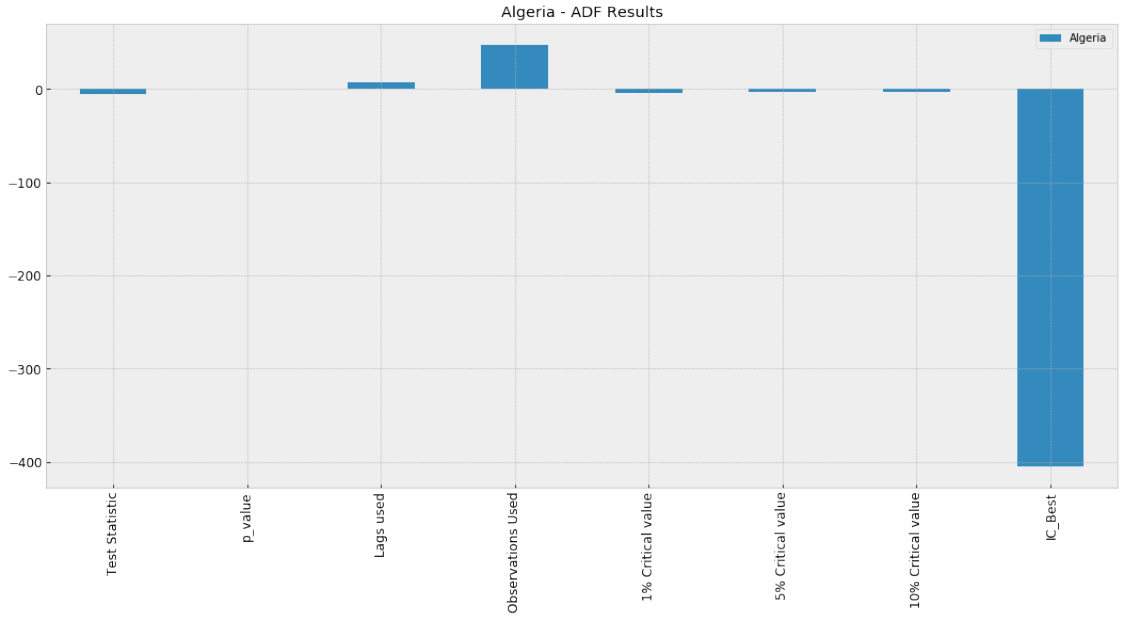
[88]:
```
adf_test['South Africa'].name
```

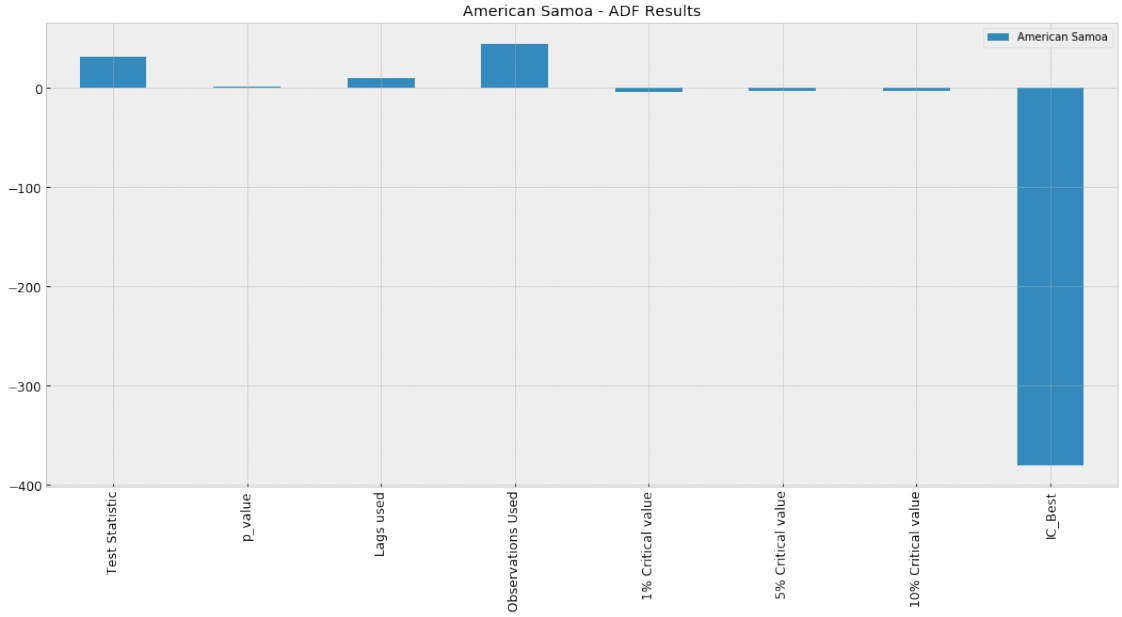
[88]:
```
'South Africa'
```

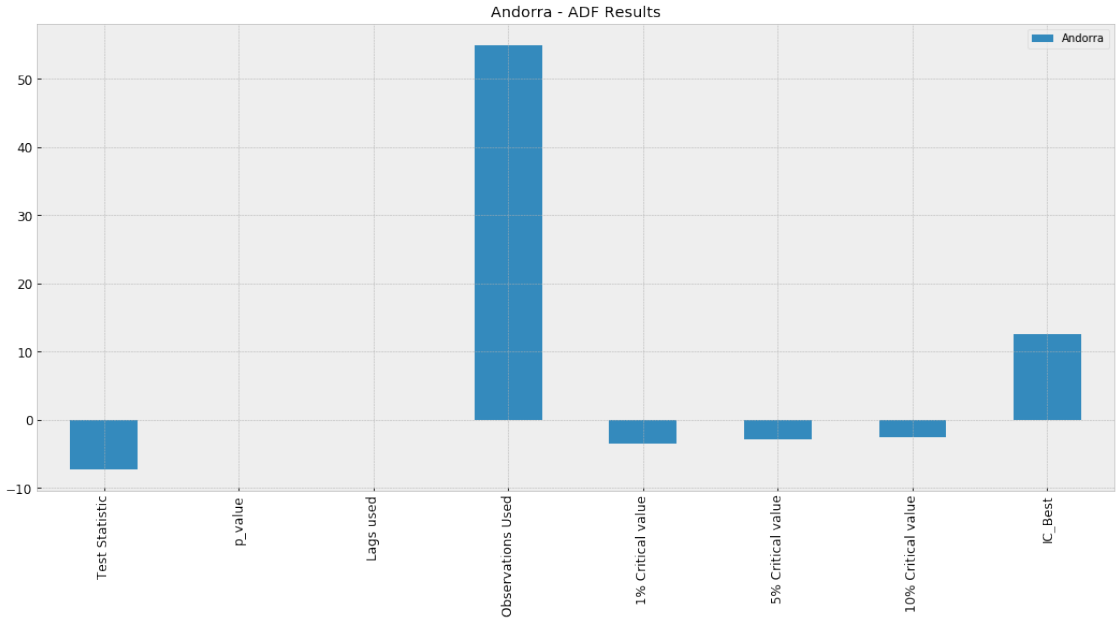
[117]:
```
for country in adf_test:
    savefile = save_images(adf_test,'adf_test')
    adf_test[country].plot.bar()
    plt.legend(loc='best')
    plt.title('{} - ADF Results'.format(adf_test[country].name))
    plt.savefig(str('{}_ADF.png'.format(savefile)))
    plt.show()


    #pyplot.figure(figsize=(15,6))

    #Plot rolling statistics:
    #adf_plot = plt.plot(adf_test[country])

    #plt.legend(loc='best')
    #plt.title('{}: ADF'.format(adf_test[country].name))
    #pyplot.savefig(str('{}adf_test.png'.format(savefile)))
    #pyplot.show(block=False)
```

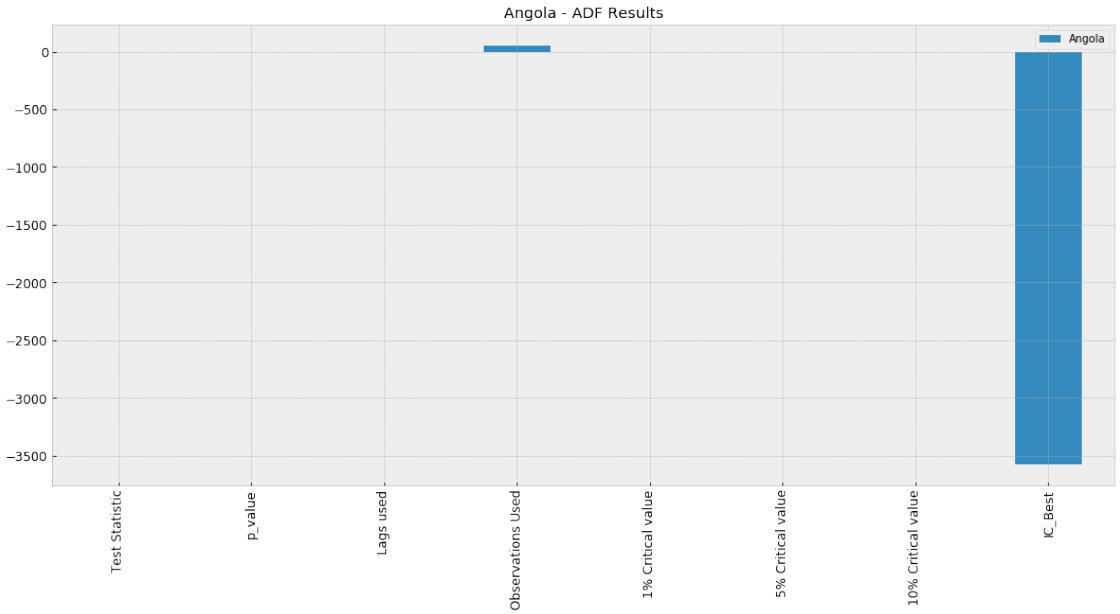

Afghanistan - ADF Results
Afghanistan
200
150
100
50
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

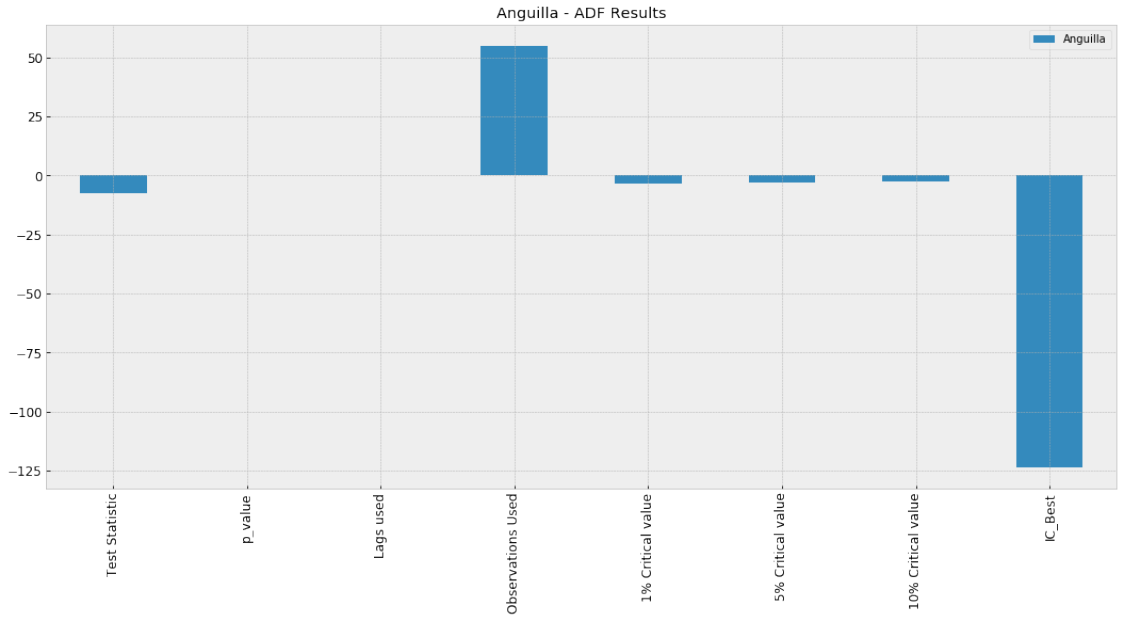

Albania - ADF Results
Albania
60
40
20
0
−20
−40
−60
−80
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

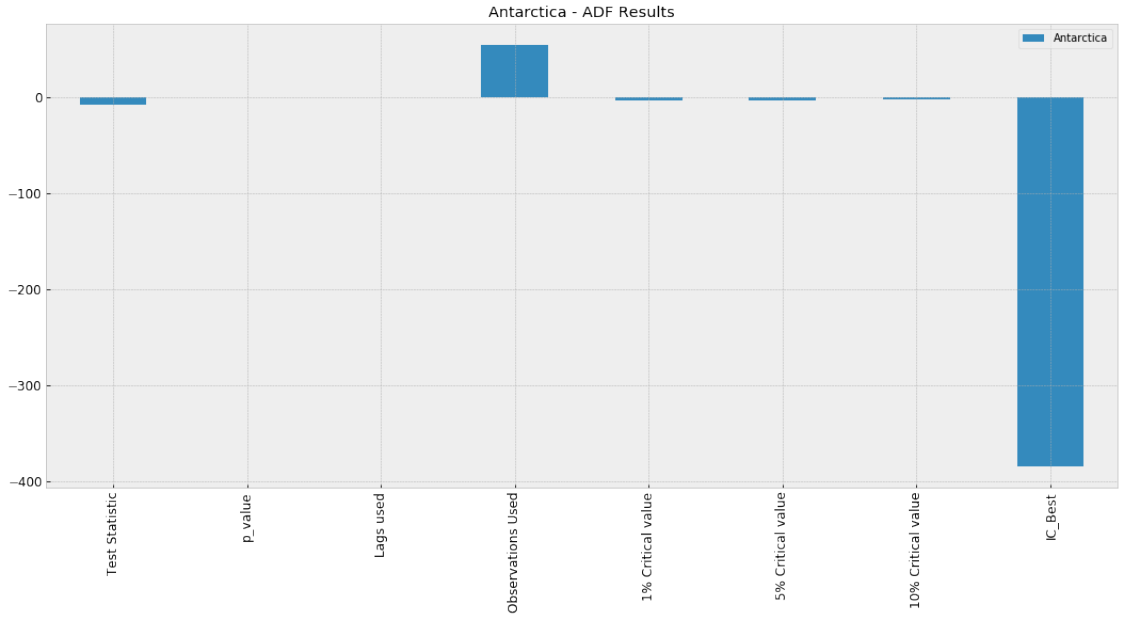

Algeria - ADF Results
Algeria
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

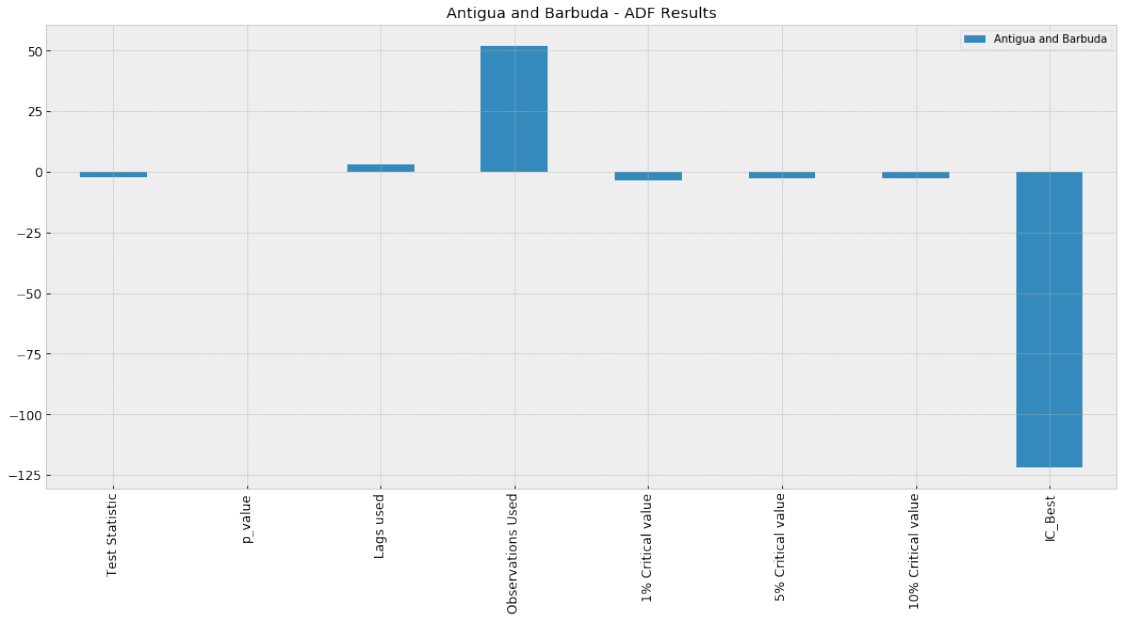

American Samoa - ADF Results
American Samoa
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

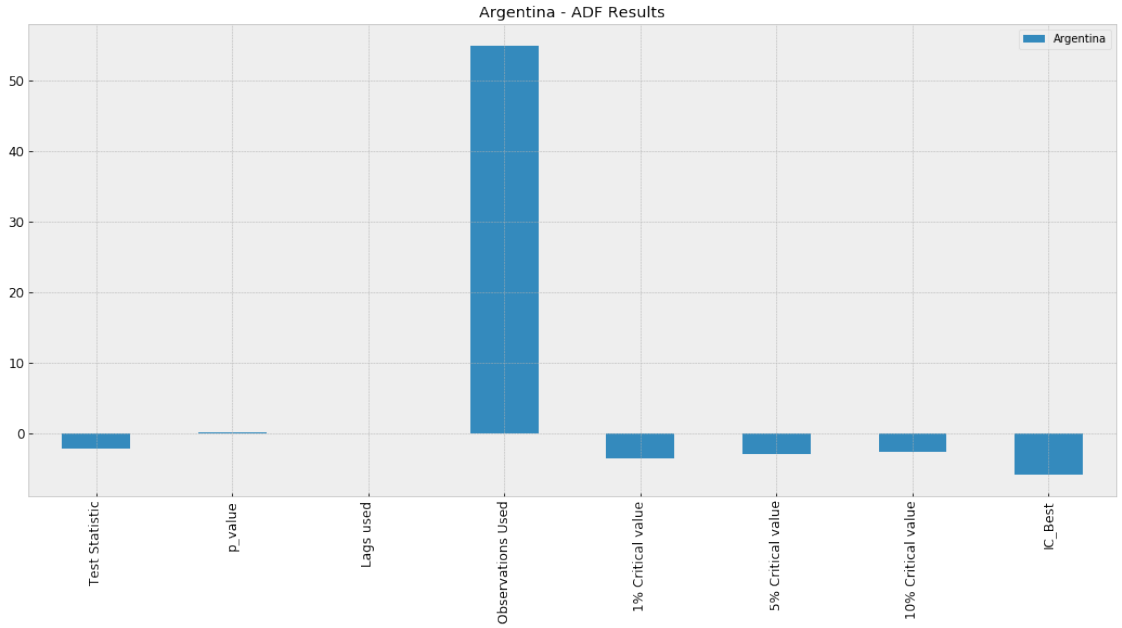

Andorra - ADF Results
Andorra
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

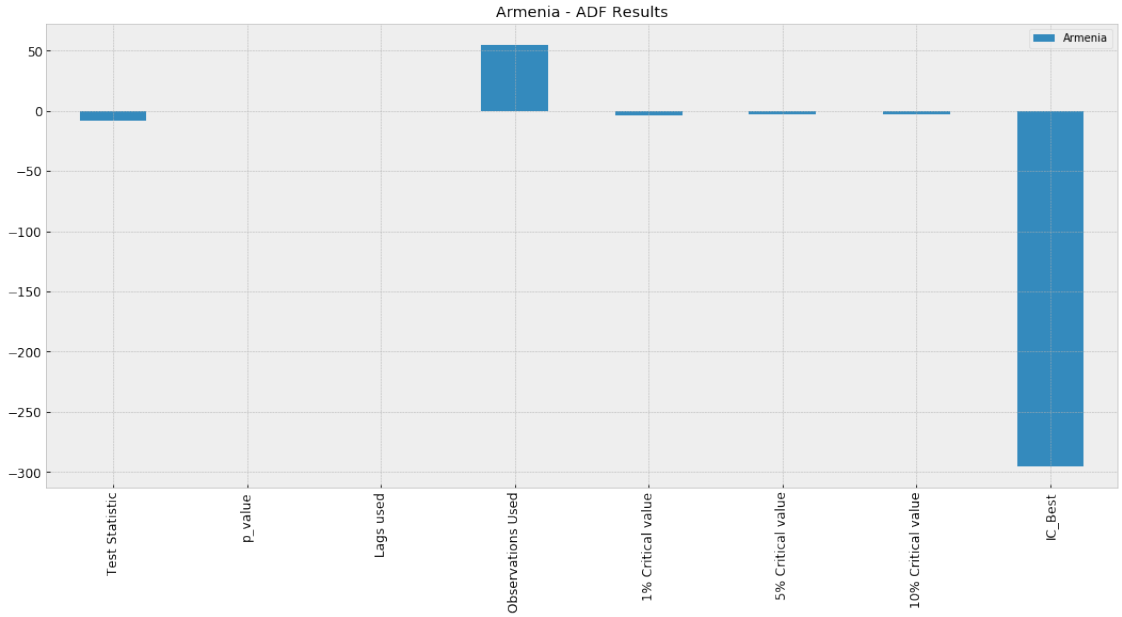

Angola - ADF Results
Angola
0
−500
−1000
−1500
−2000
−2500
−3000
−3500
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

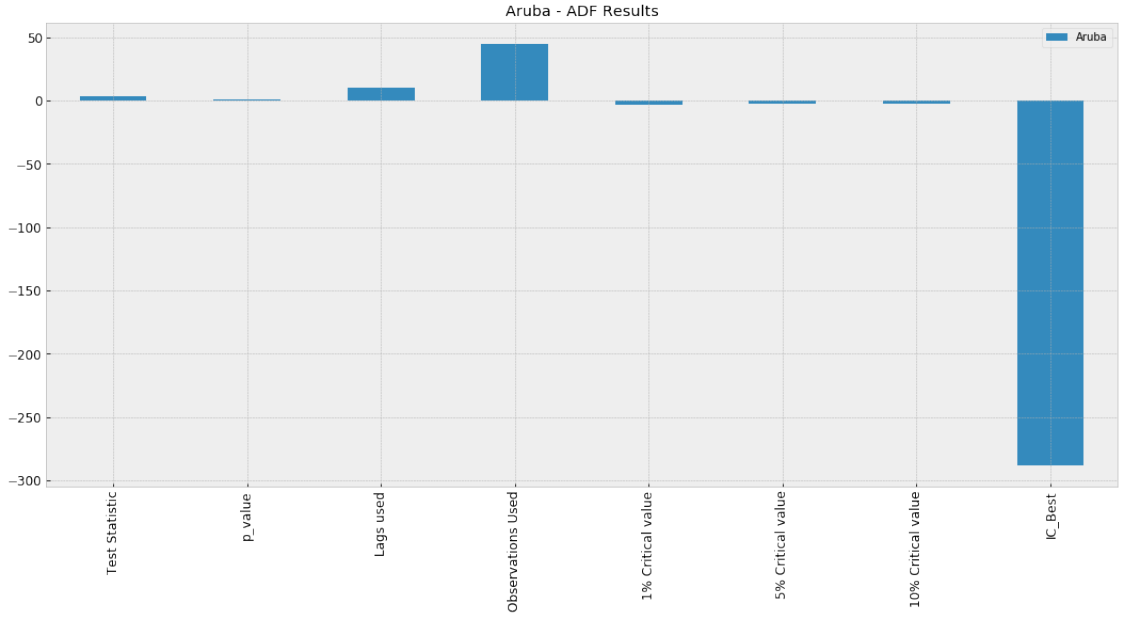

Anguilla - ADF Results
Anguilla
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

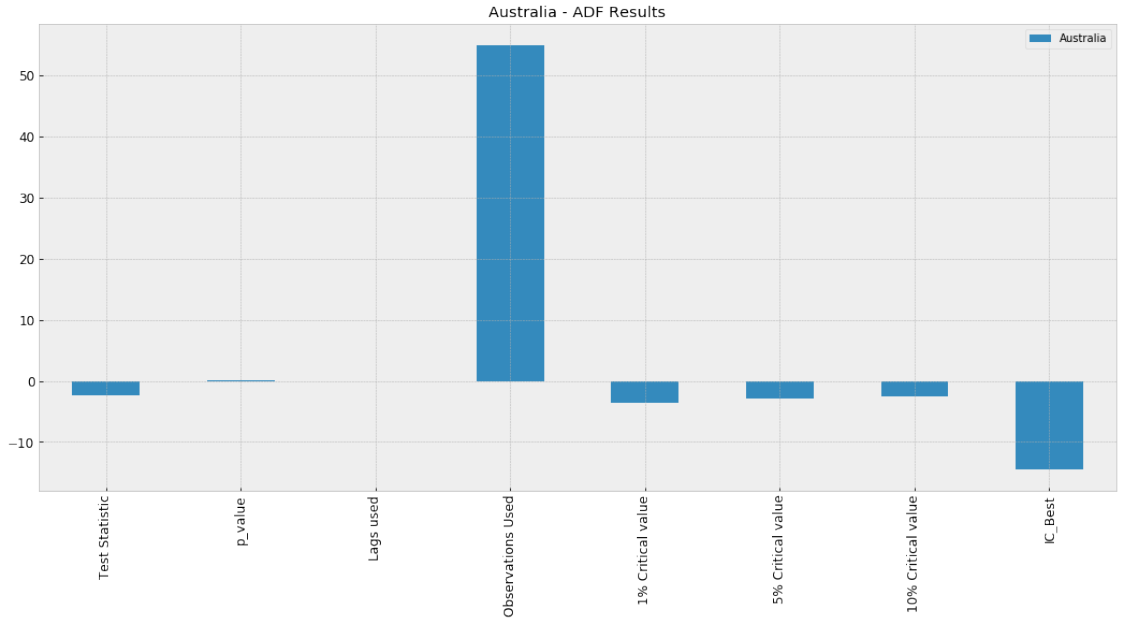

Antarctica - ADF Results
Antarctica
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

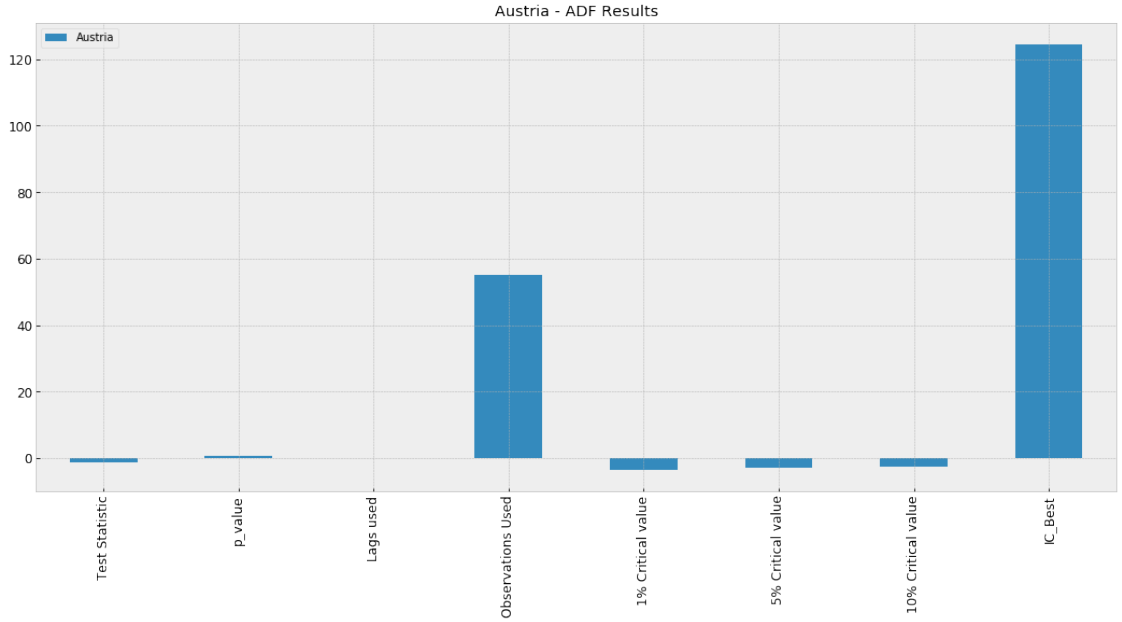

Antigua and Barbuda - ADF Results
Antigua and Barbuda
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

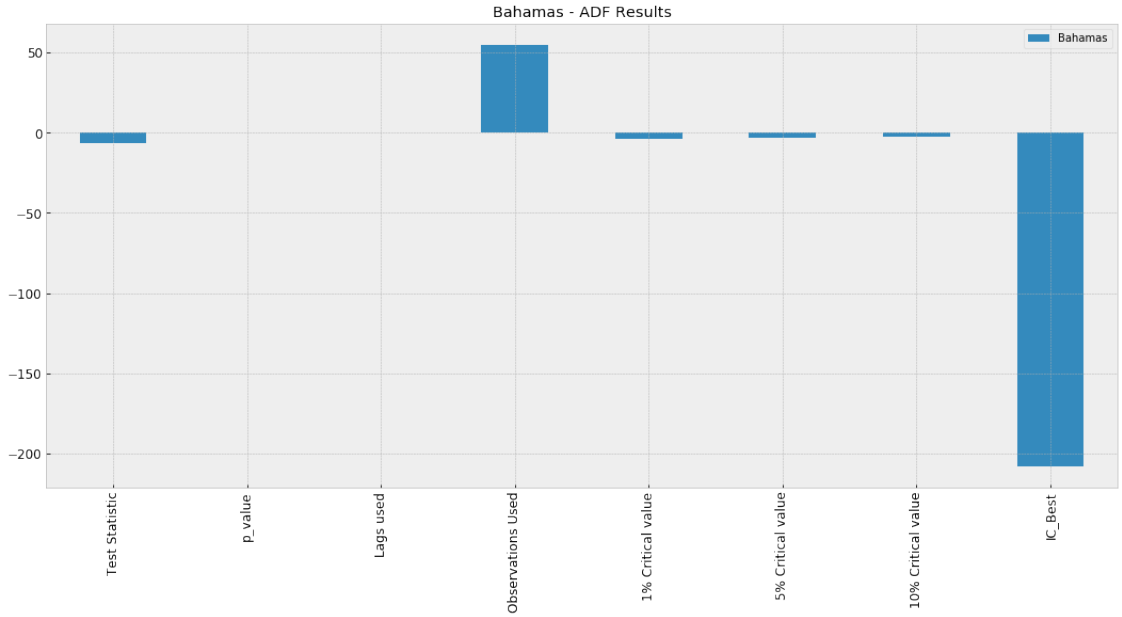

Argentina - ADF Results
Argentina
50
40
30
20
10
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

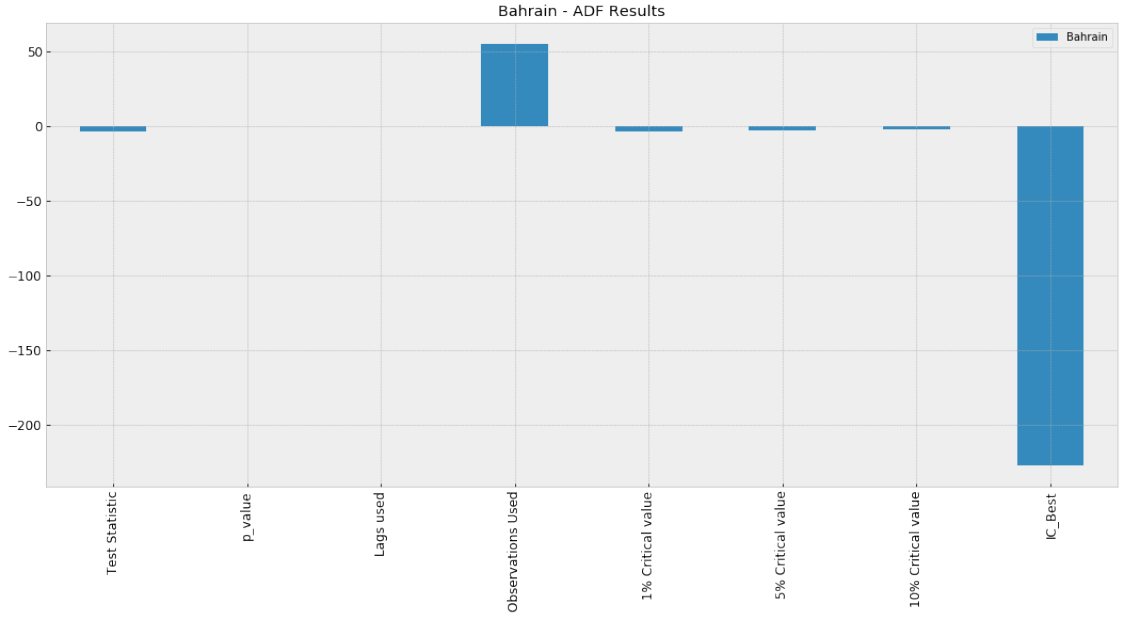

Armenia - ADF Results
Armenia
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

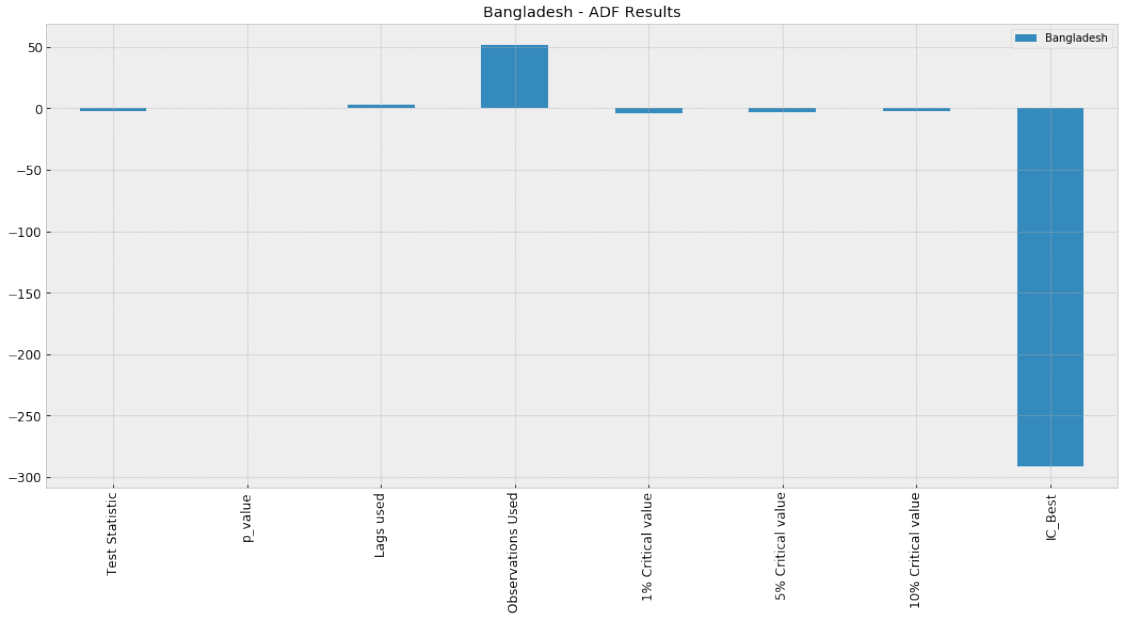

Aruba - ADF Results
Aruba
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

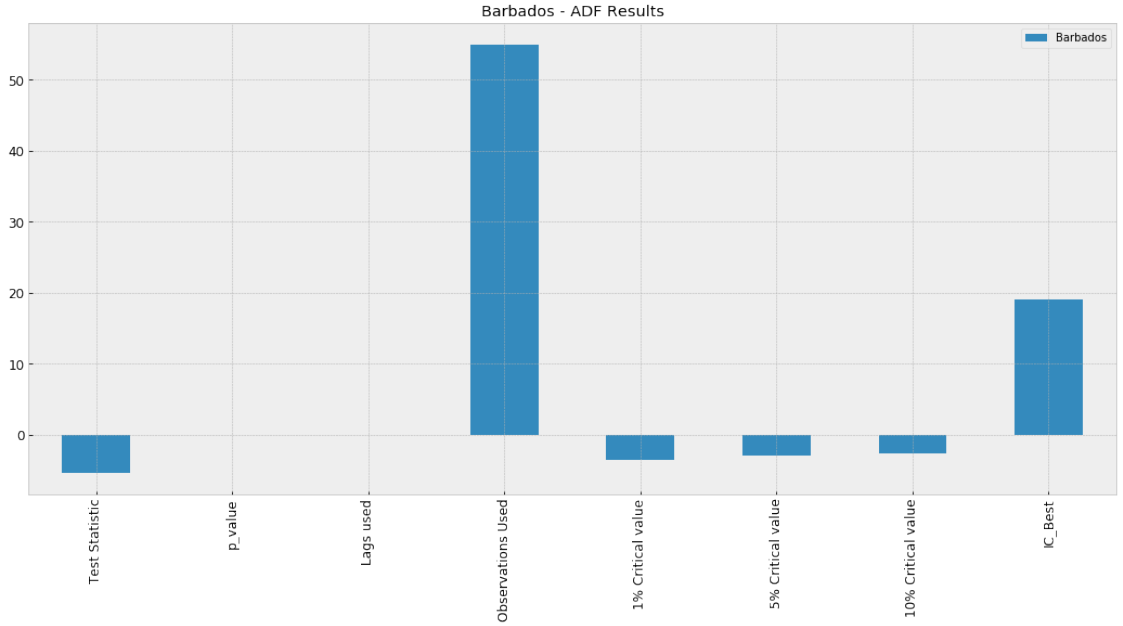

Australia - ADF Results
Australia
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

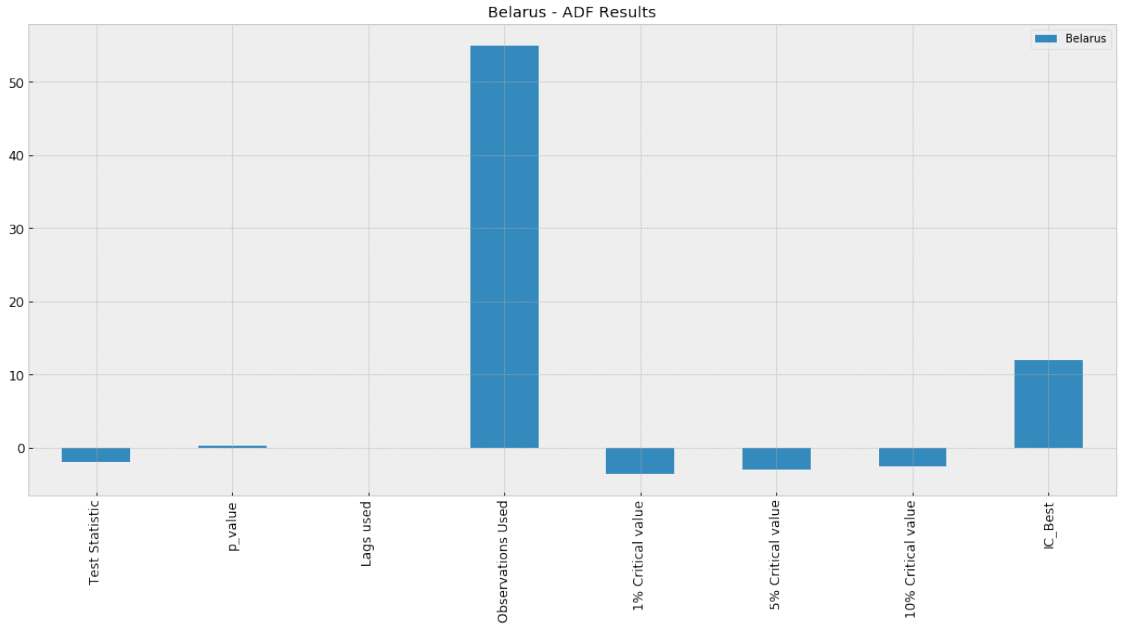

Austria - ADF Results
Austria
120
100
80
60
40
20
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

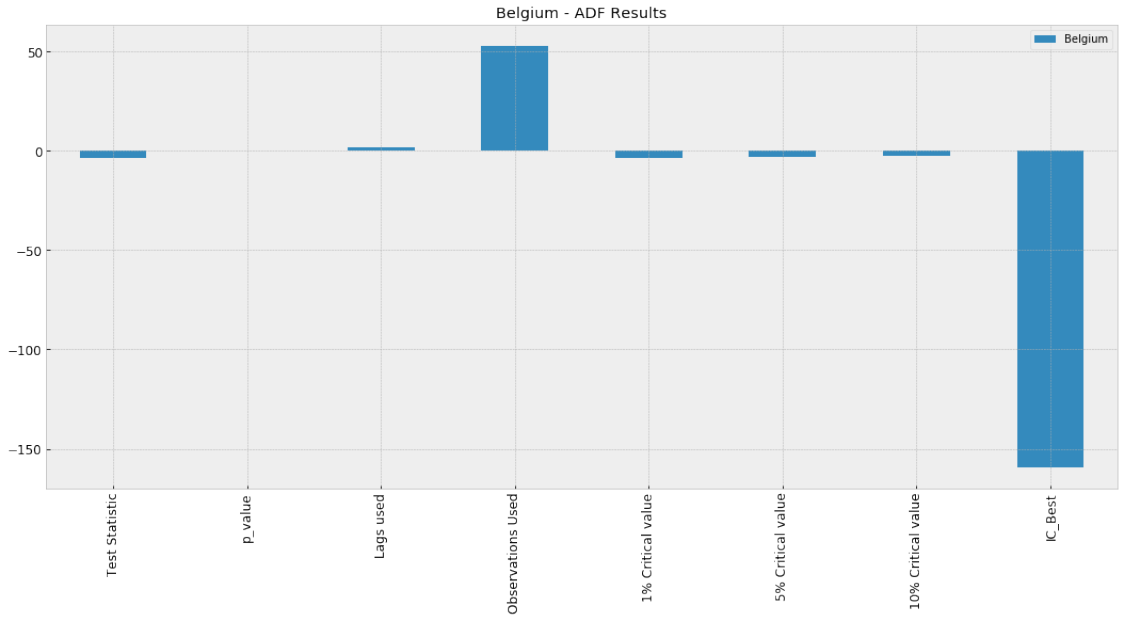

Bahamas - ADF Results
Bahamas
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

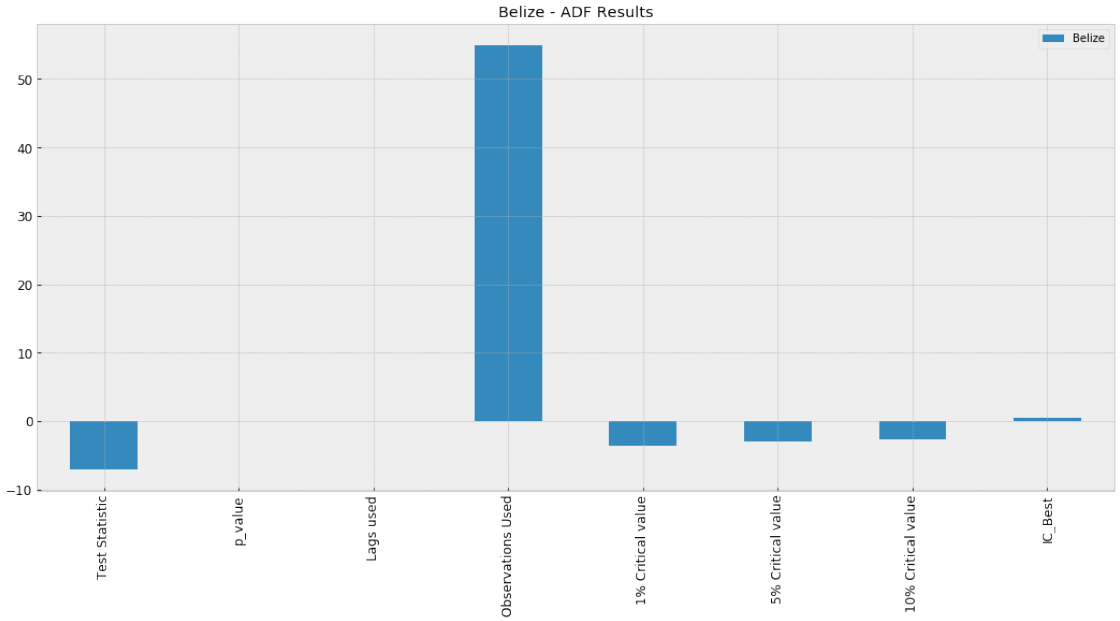

Bahrain - ADF Results
Bahrain
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

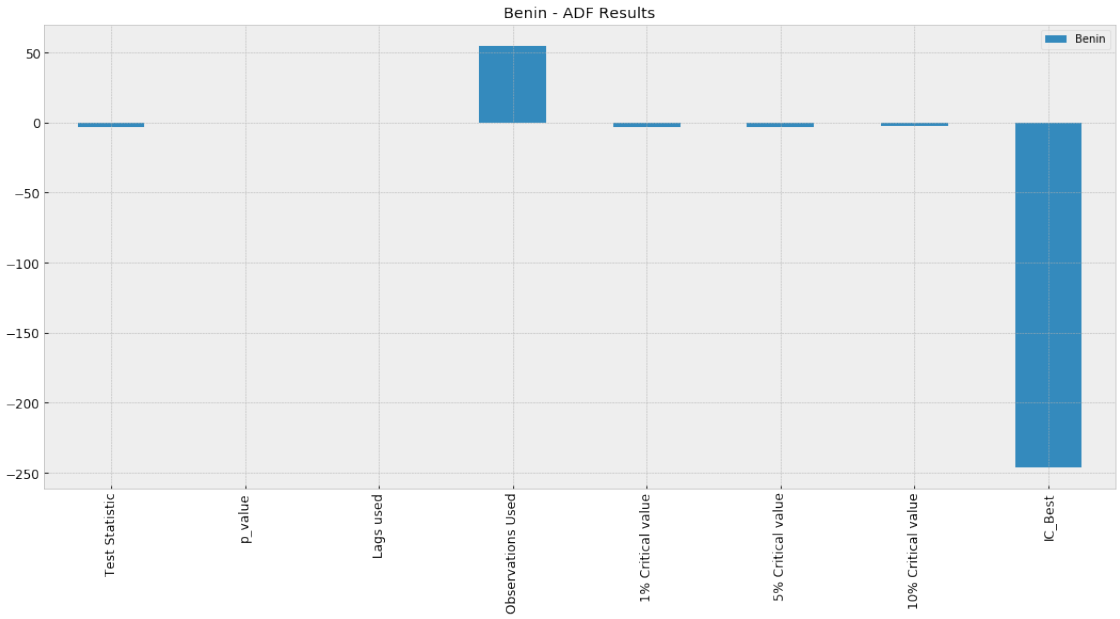

Bangladesh - ADF Results
Bangladesh
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

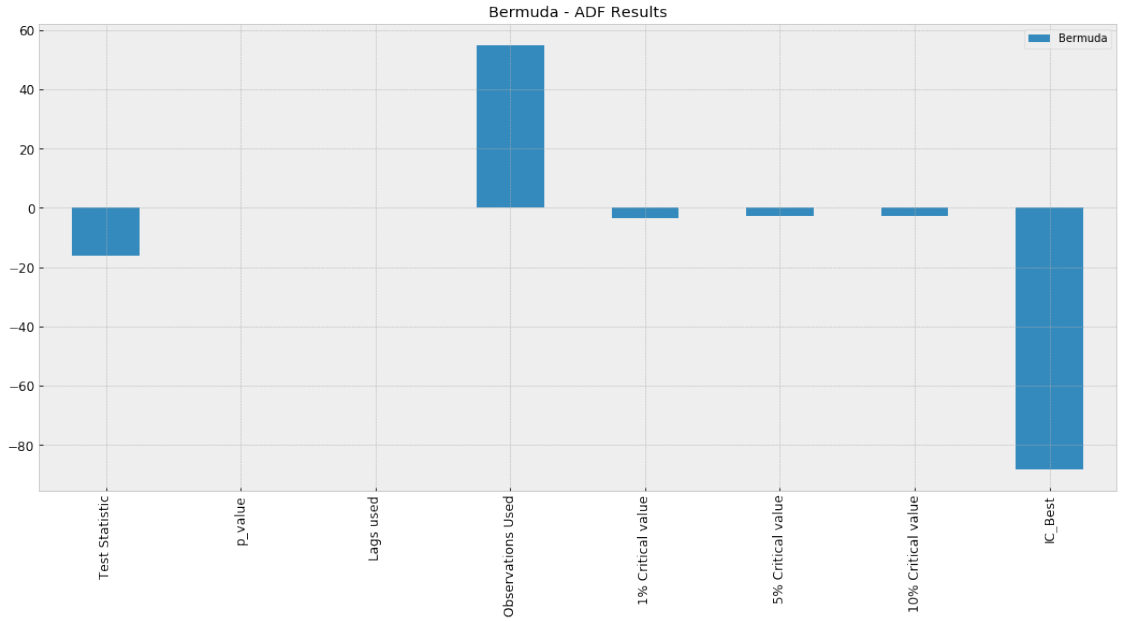

Barbados - ADF Results
Barbados
50
40
30
20
10
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

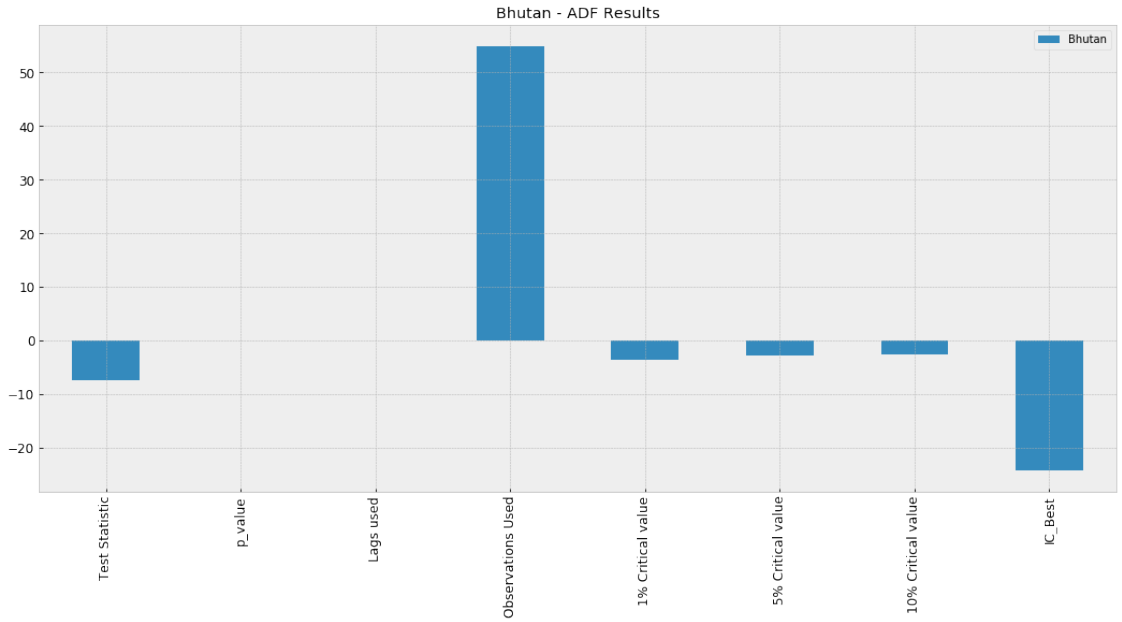

Belarus - ADF Results
Belarus
50
40
30
20
10
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

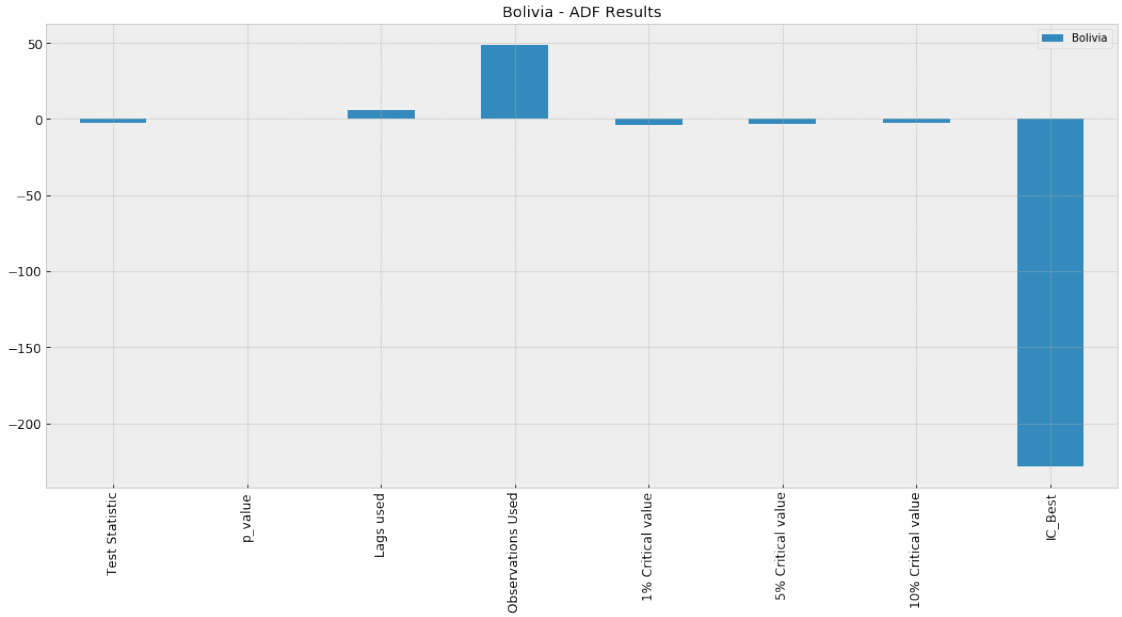

Belgium - ADF Results
Belgium
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

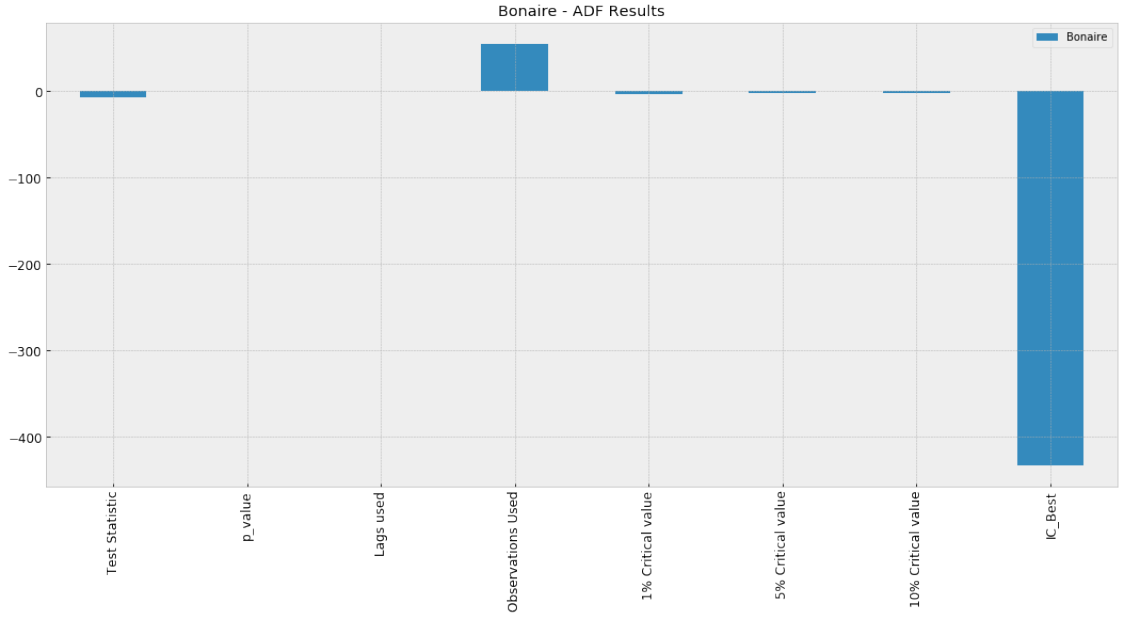

Belize - ADF Results
Belize
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

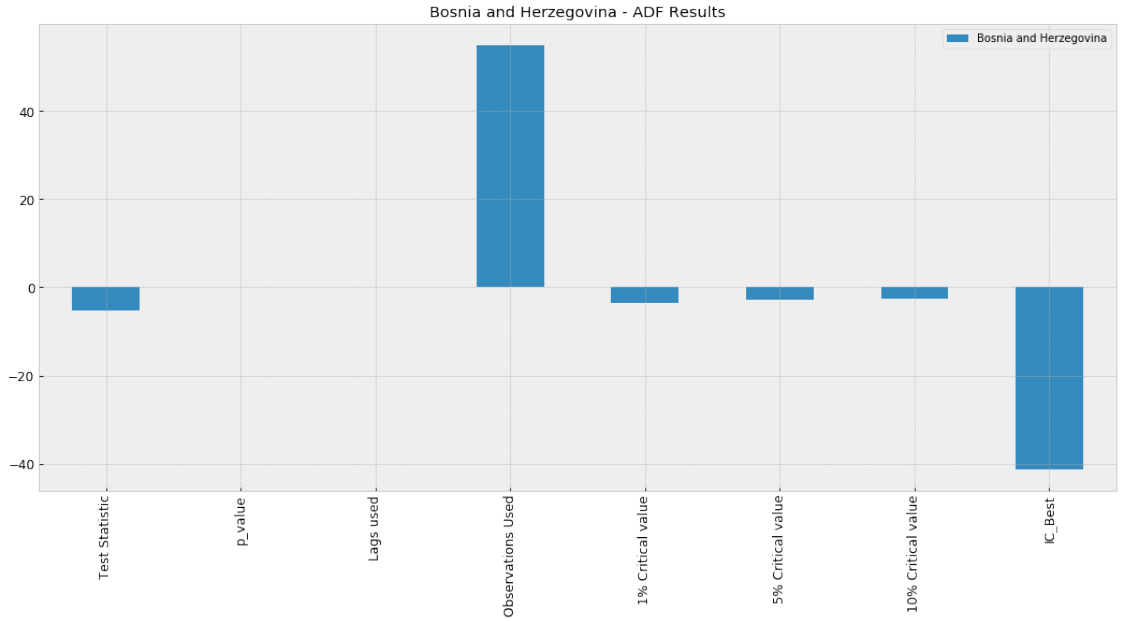

Benin - ADF Results
Benin
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

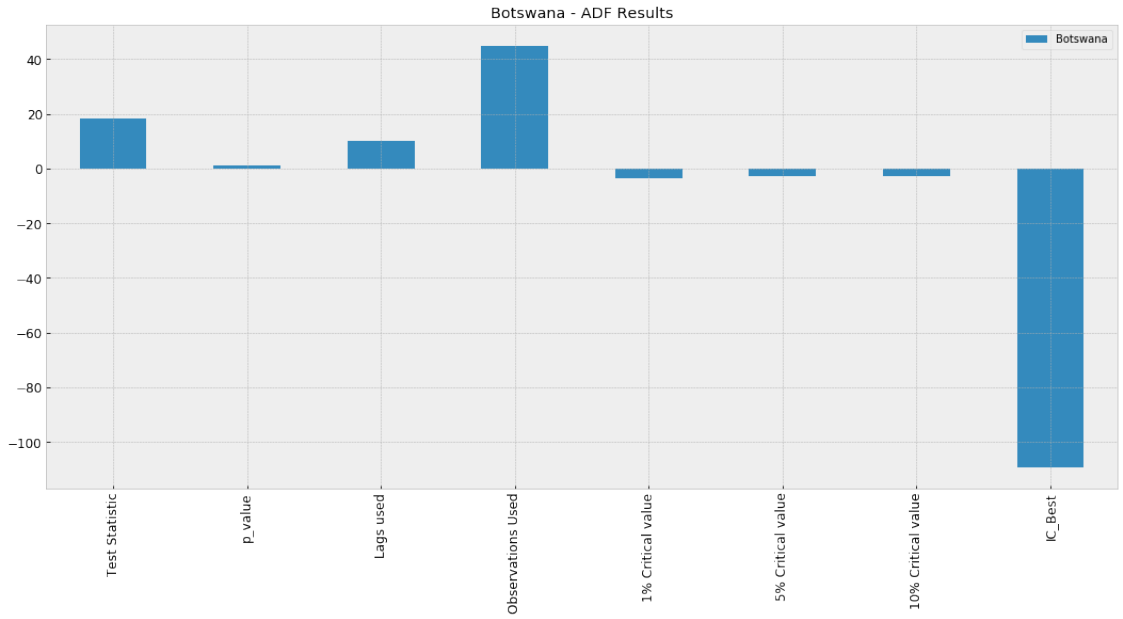

Bermuda - ADF Results
Bermuda
60
40
20
0
−20
−40
−60
−80
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

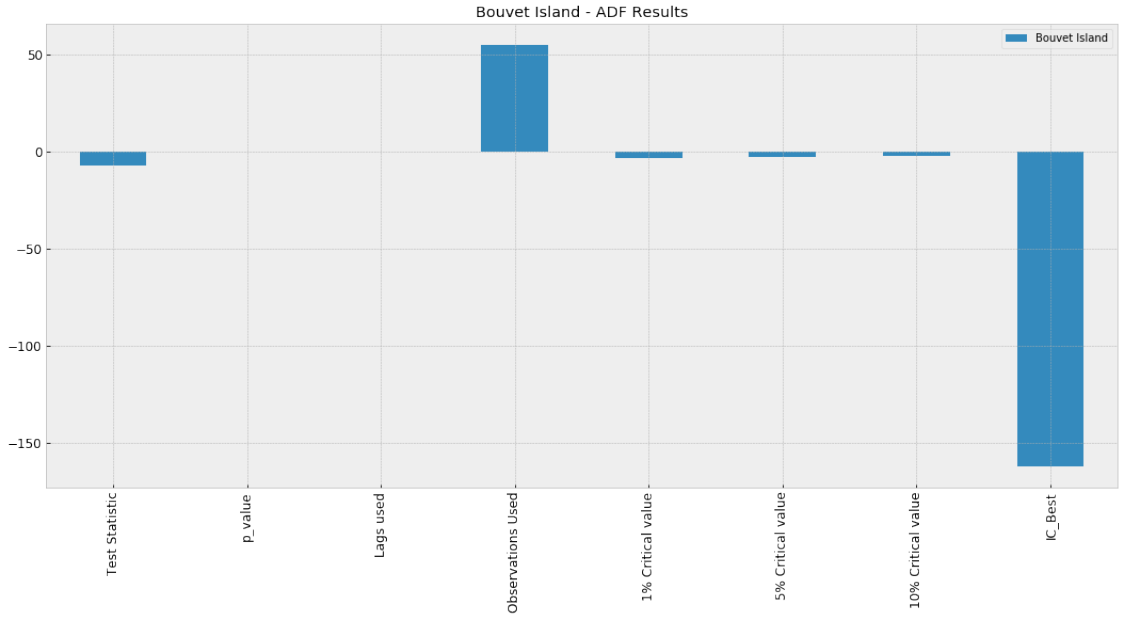

Bhutan - ADF Results
Bhutan
50
40
30
20
10
0
−10
−20
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

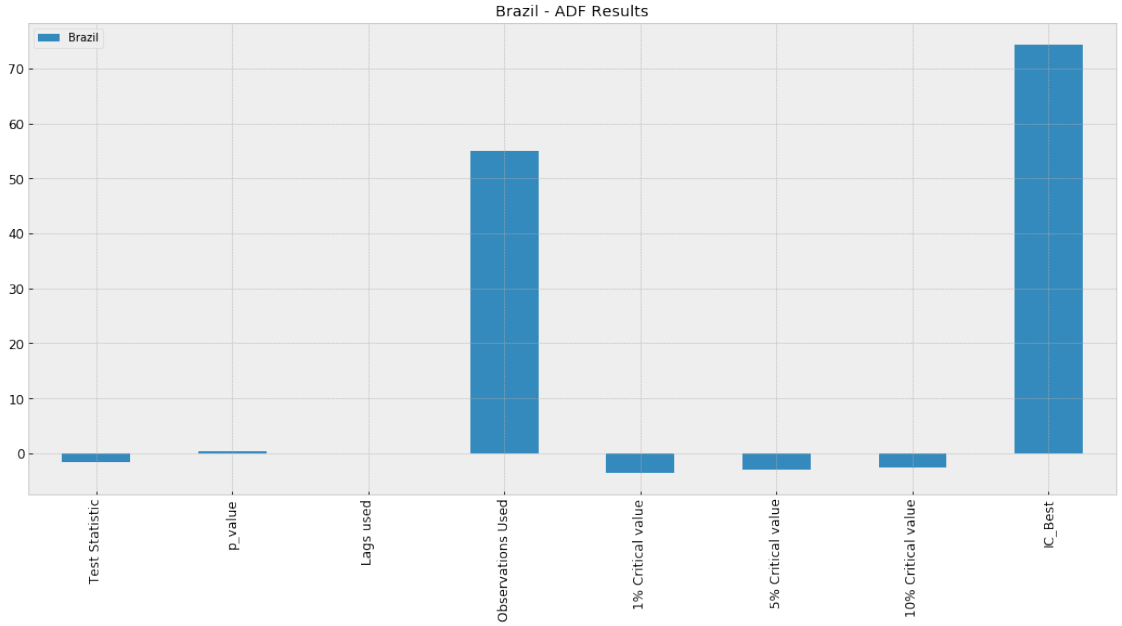

Bolivia - ADF Results
Bolivia
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

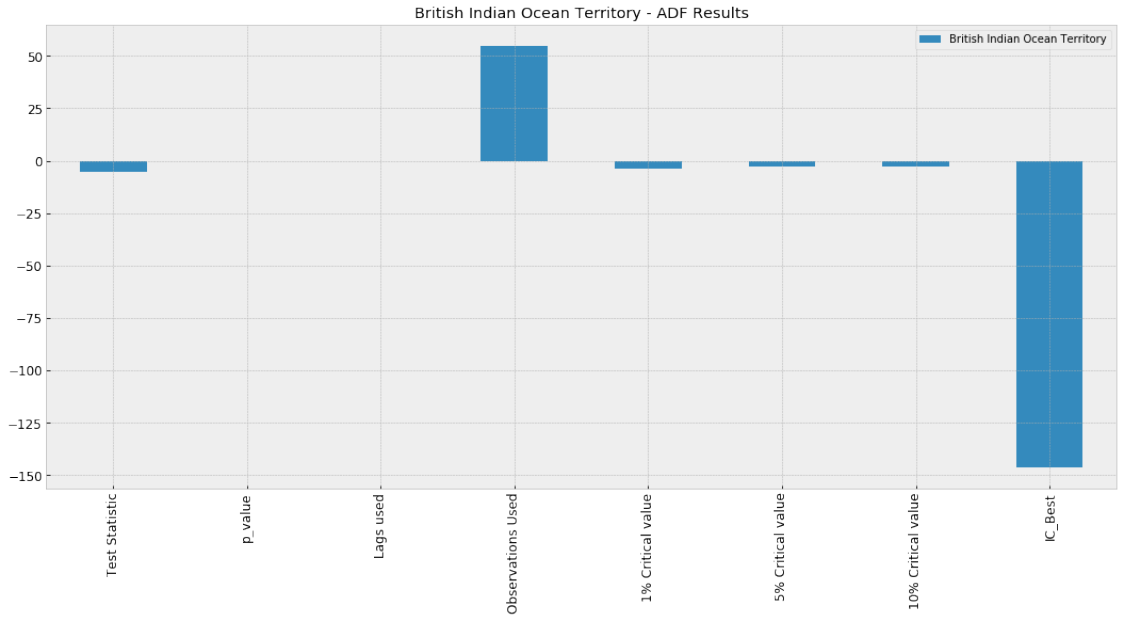

Bonaire - ADF Results
Bonaire
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

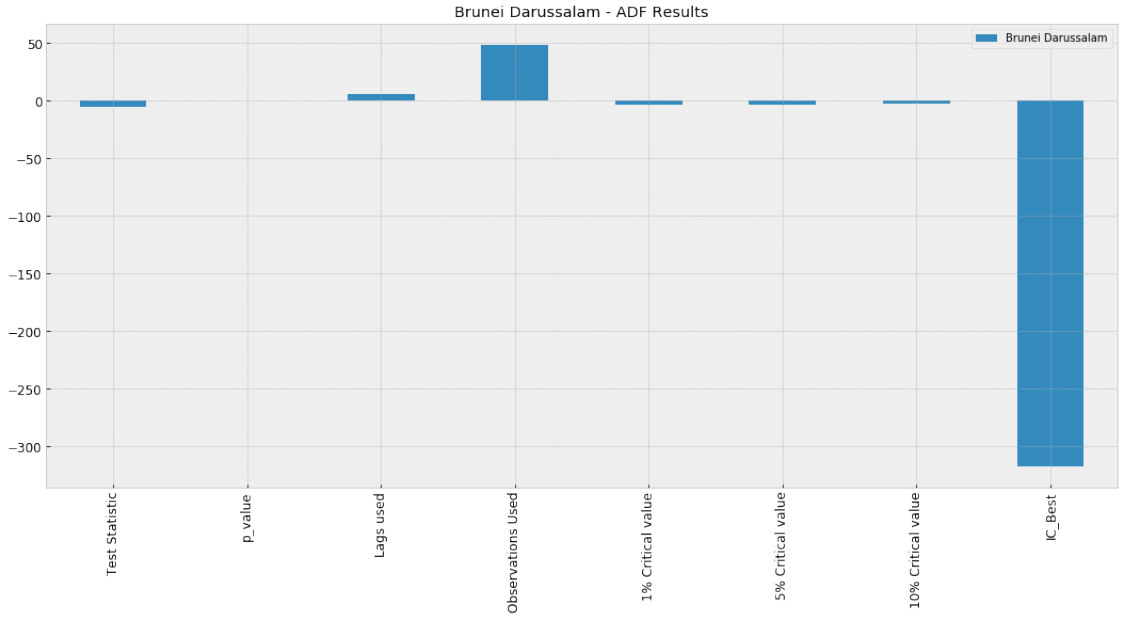

Bosnia and Herzegovina - ADF Results
Bosnia and Herzegovina
40
20
0
−20
−40
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

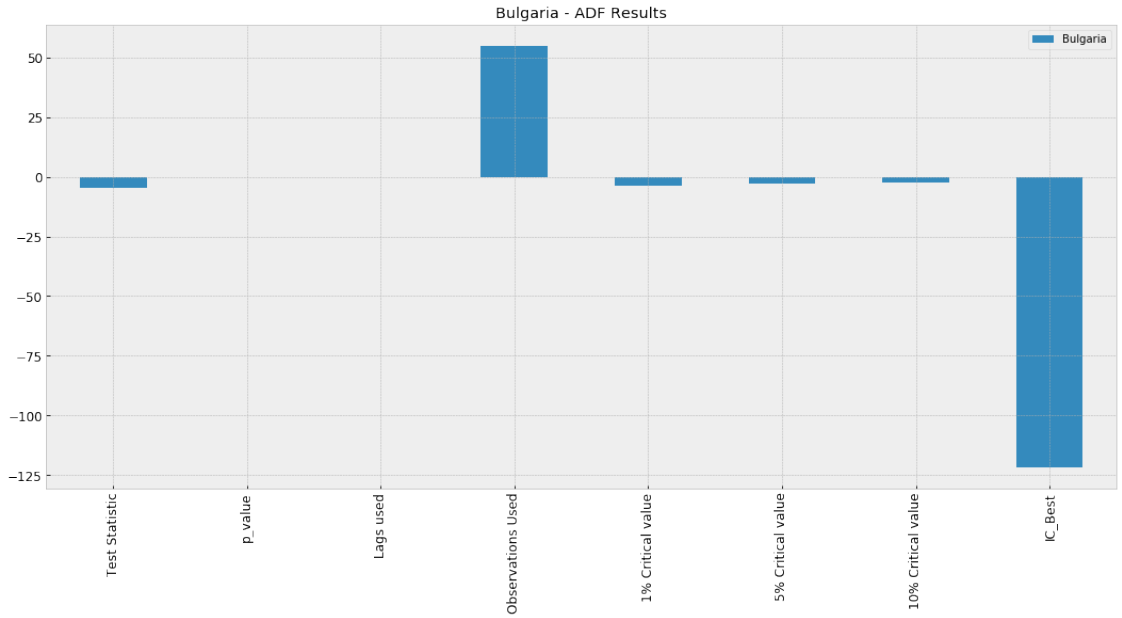

Botswana - ADF Results
Botswana
40
20
0
−20
−40
−60
−80
−100
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

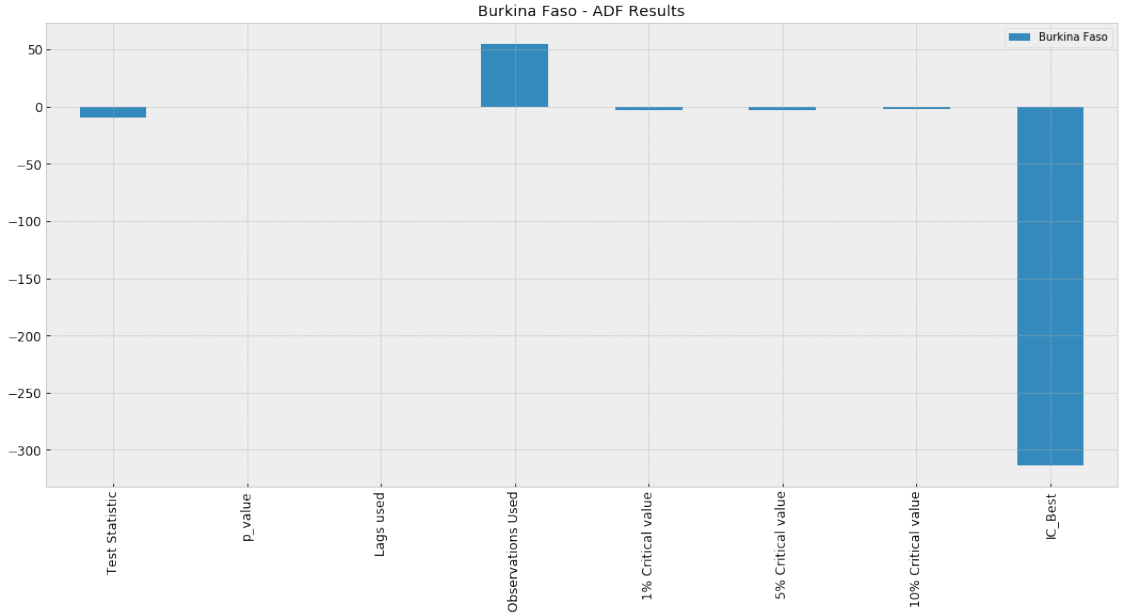

Bouvet Island - ADF Results
Bouvet Island
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

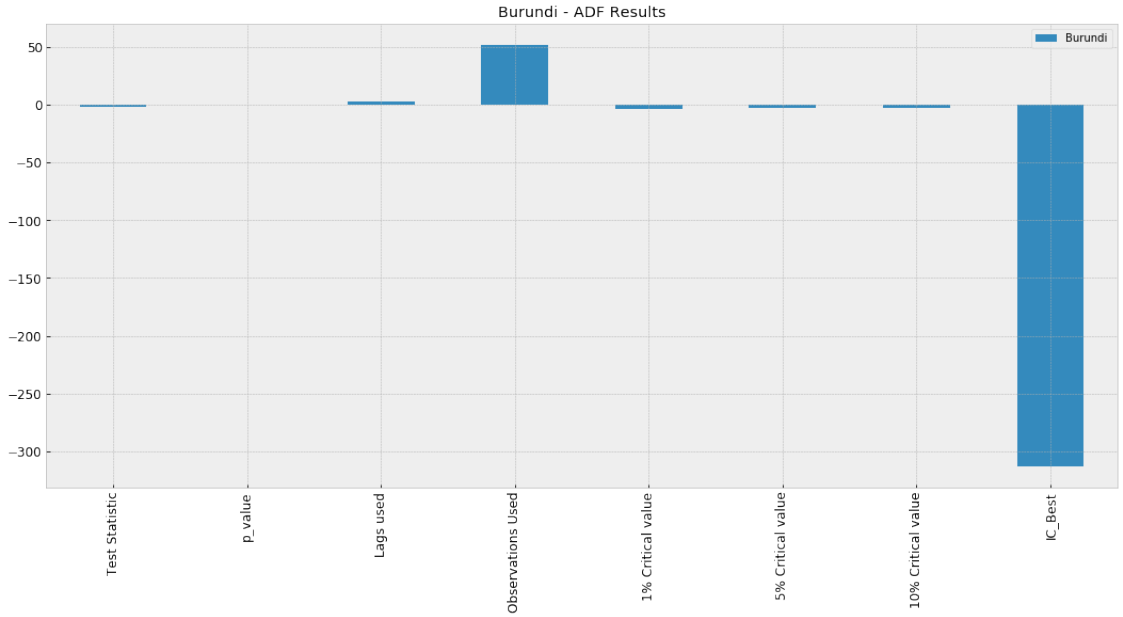

Brazil - ADF Results
Brazil
70
60
50
40
30
20
10
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

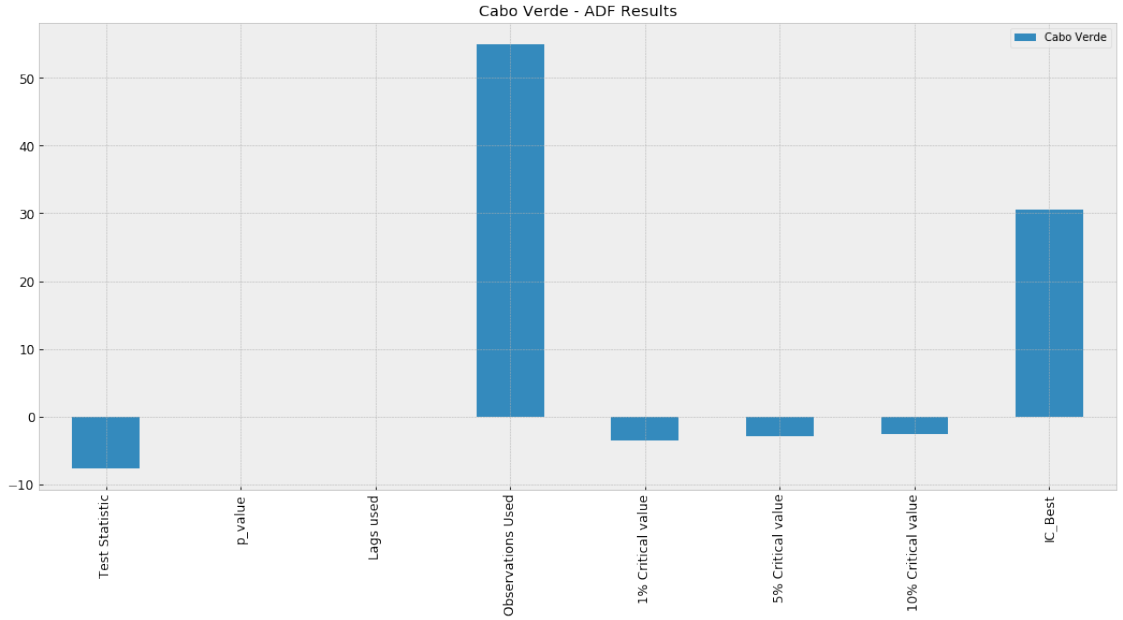

British Indian Ocean Territory - ADF Results
British Indian Ocean Territory
50
25
0
−25
−50
−75
−100
−125
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

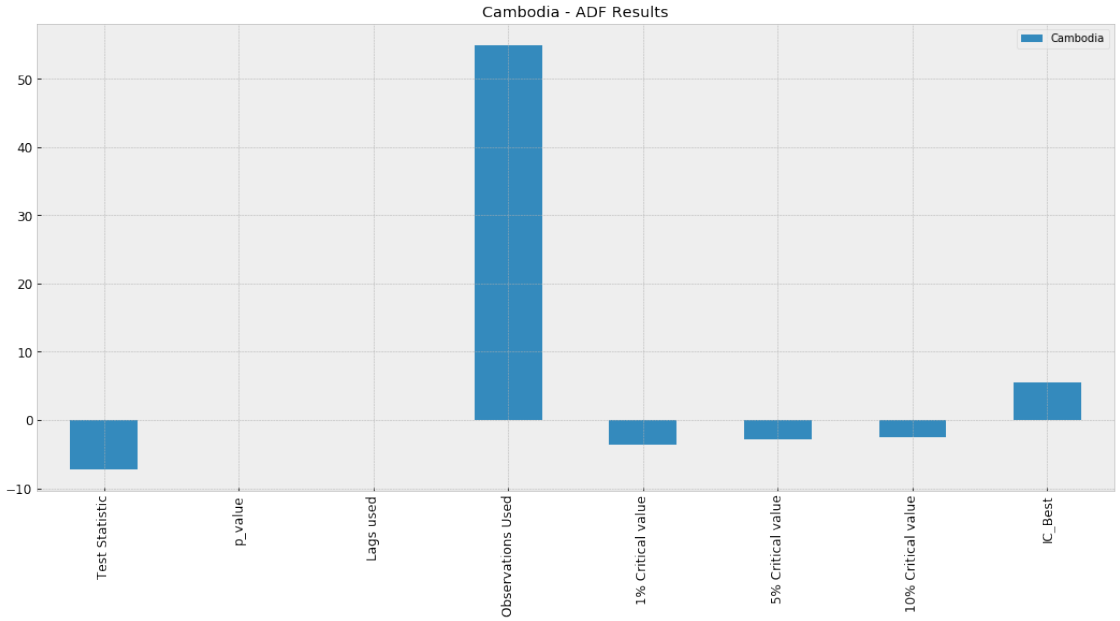

Brunei Darussalam - ADF Results
Brunei Darussalam
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

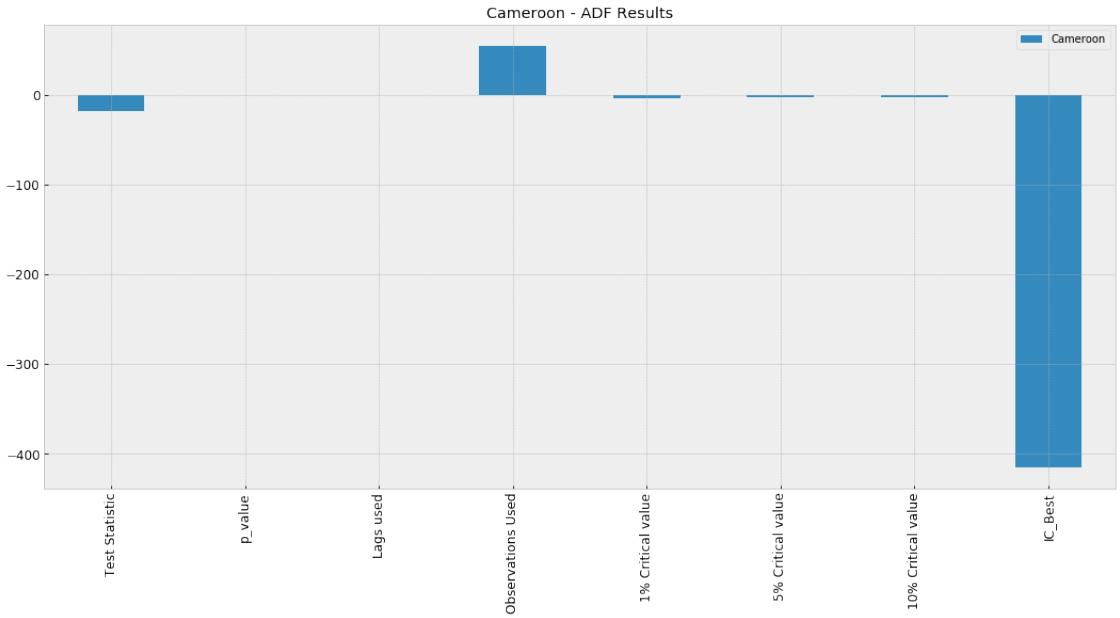

Bulgaria - ADF Results
Bulgaria
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

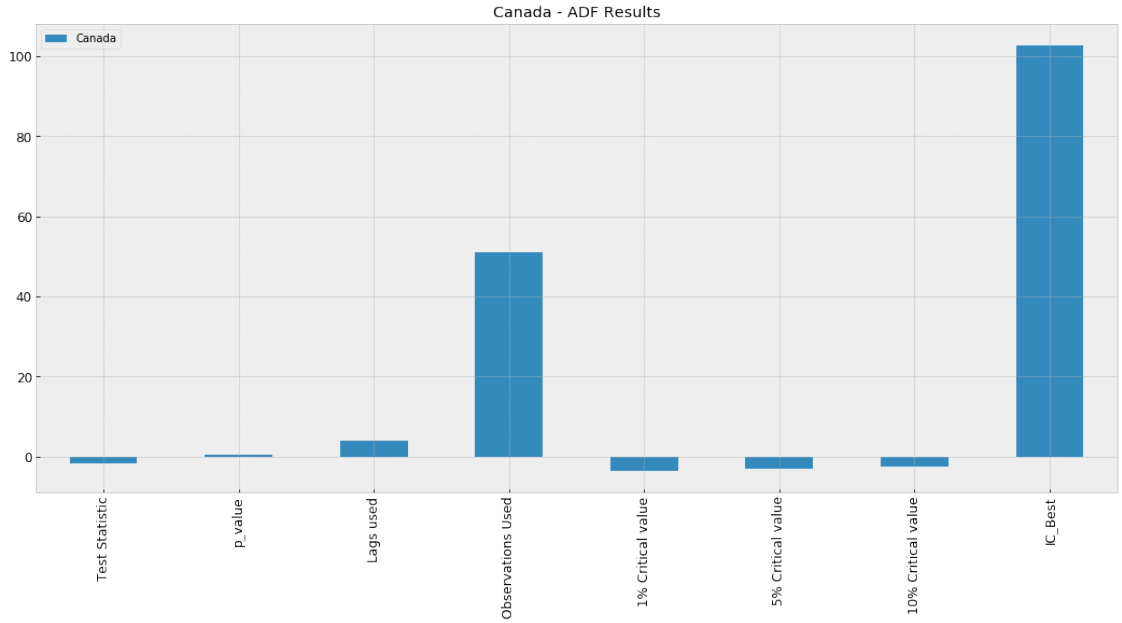

Burkina Faso - ADF Results
Burkina Faso
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

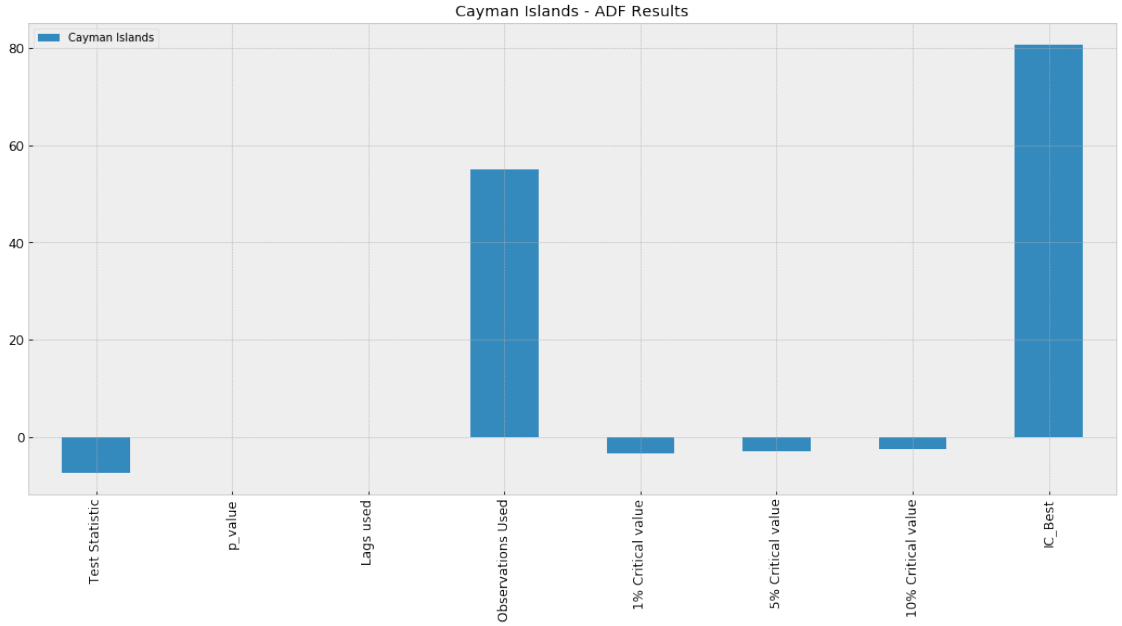

Burundi - ADF Results
Burundi
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

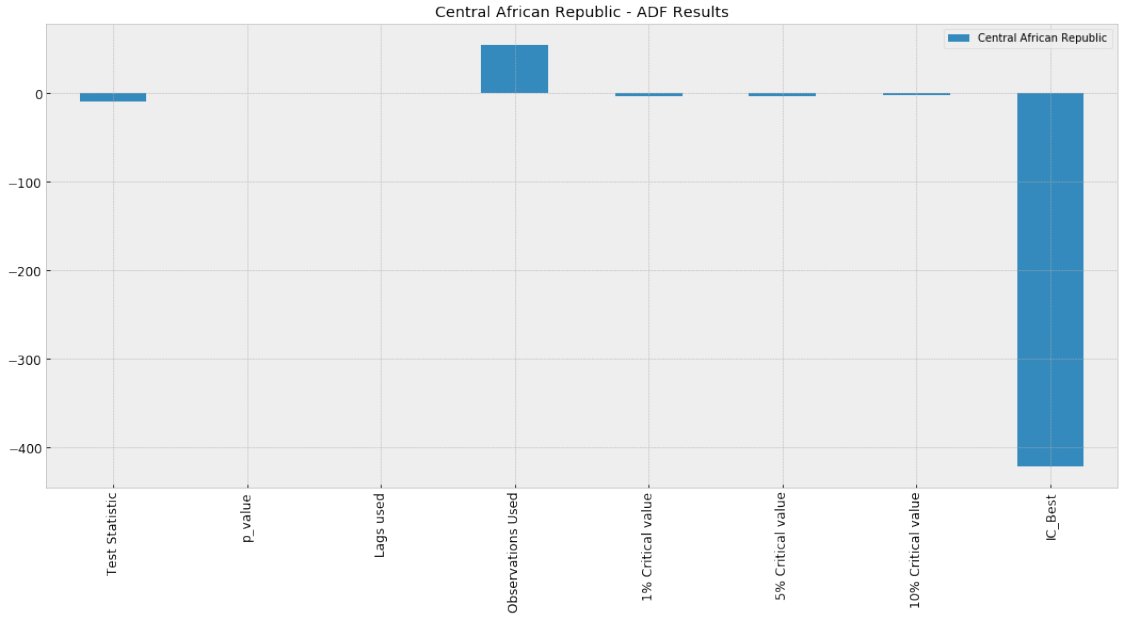

Cabo Verde - ADF Results
Cabo Verde
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

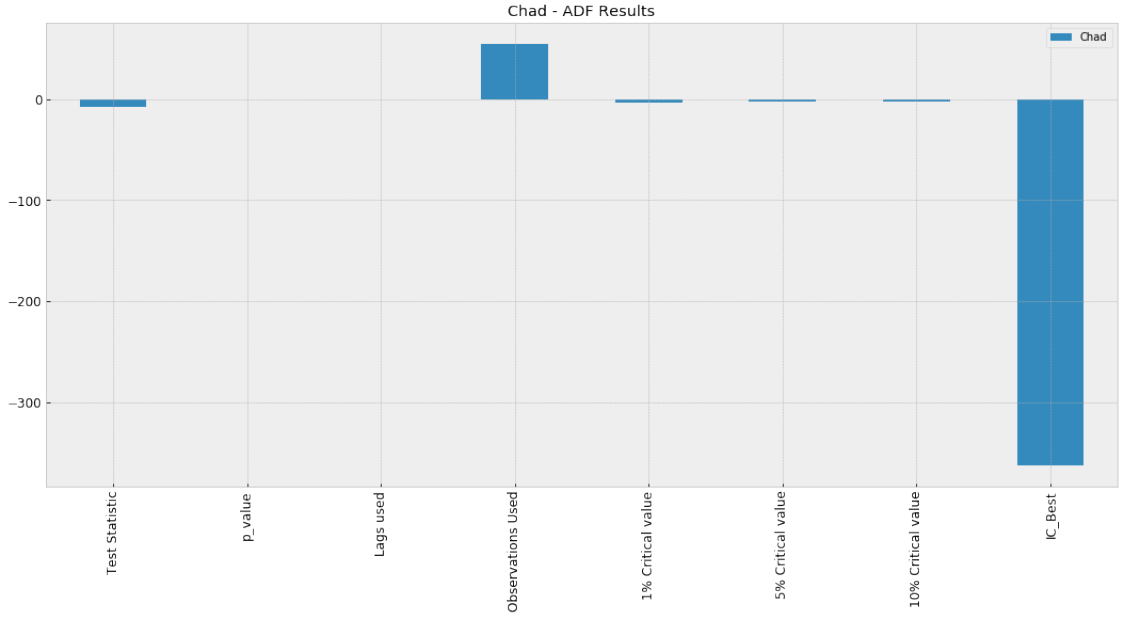

Cambodia - ADF Results
Cambodia
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

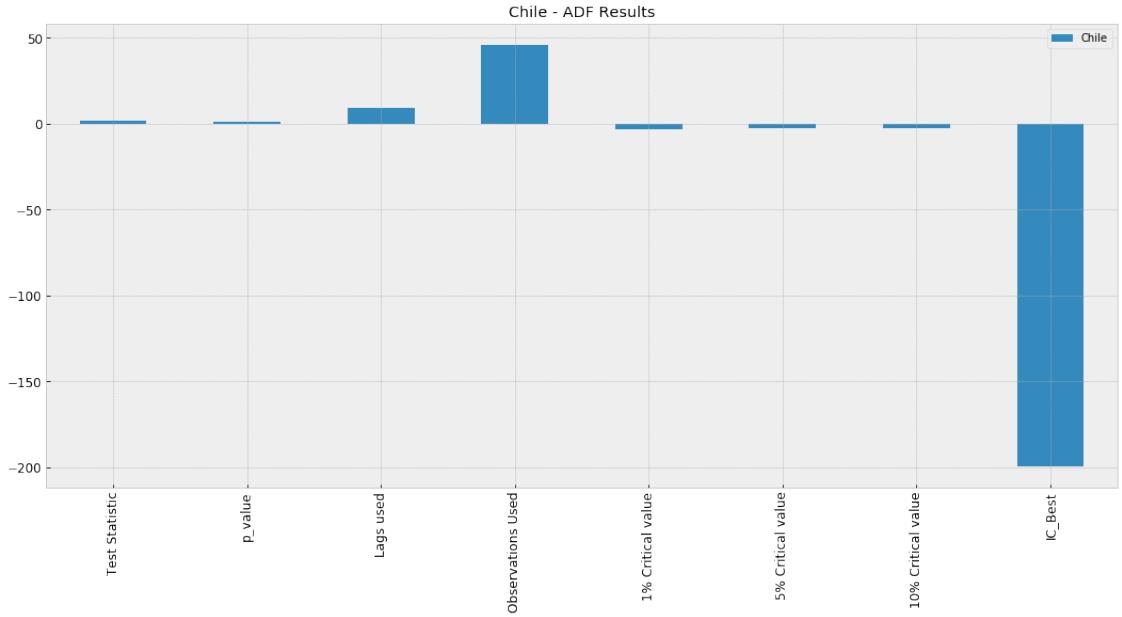

Cameroon - ADF Results
Cameroon
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

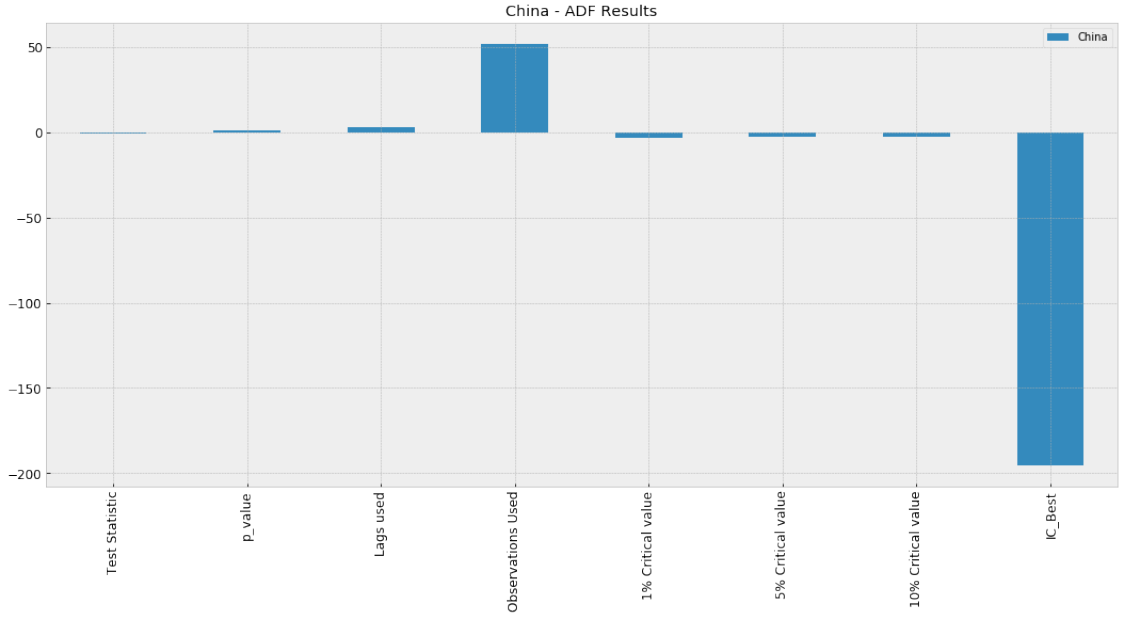

Canada - ADF Results
Canada
100
80
60
40
20
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

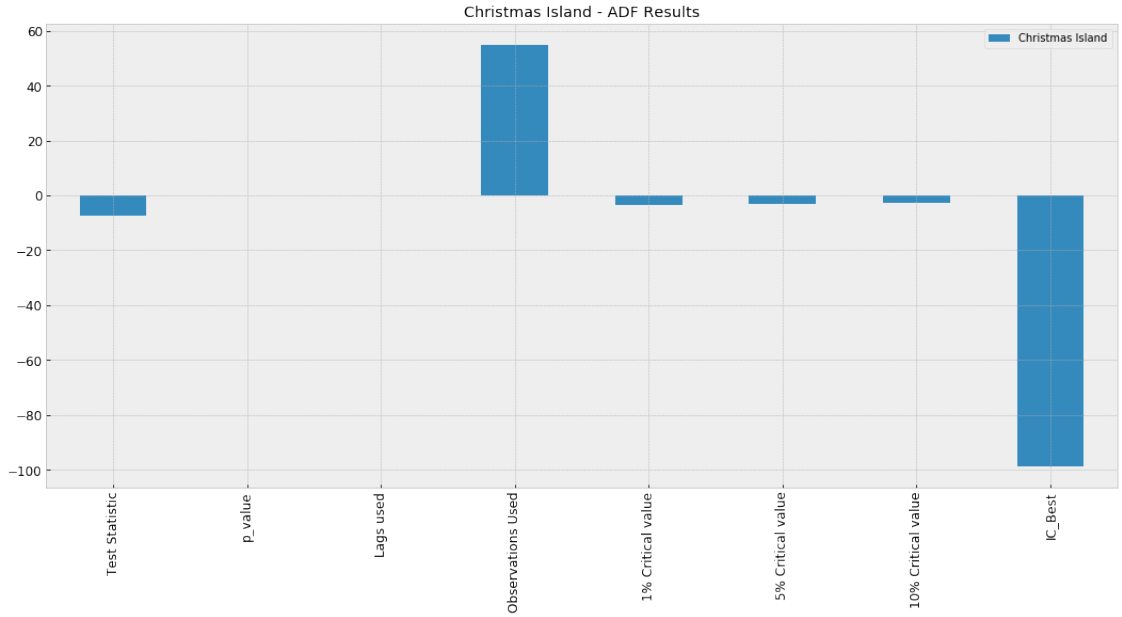

Cayman Islands - ADF Results
Cayman Islands
80
60
40
20
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

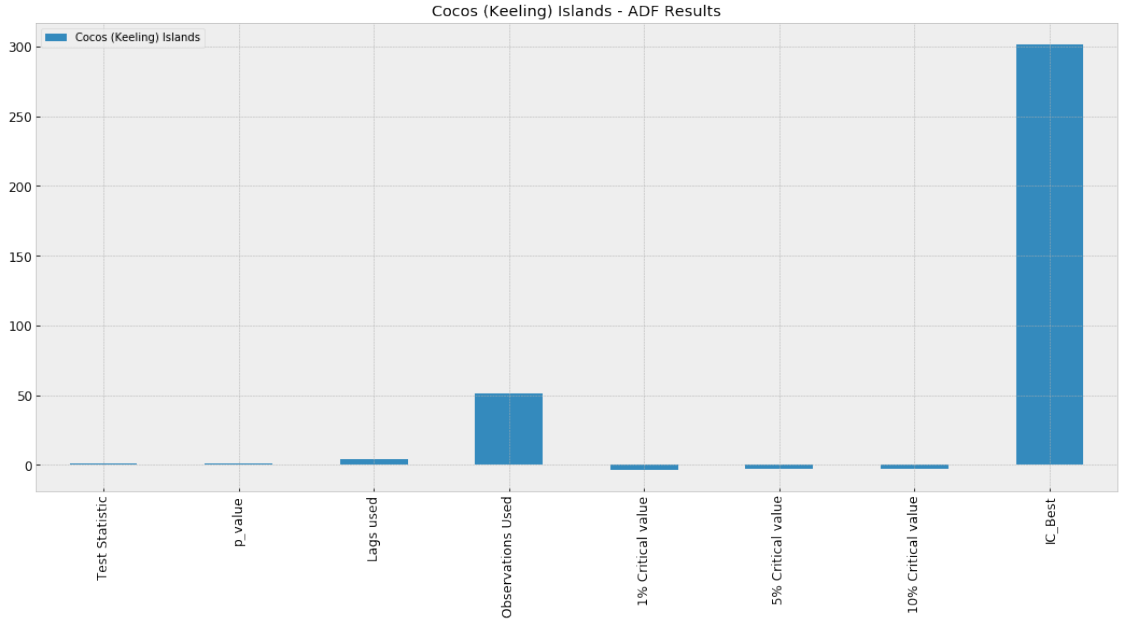

Central African Republic - ADF Results
Central African Republic
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

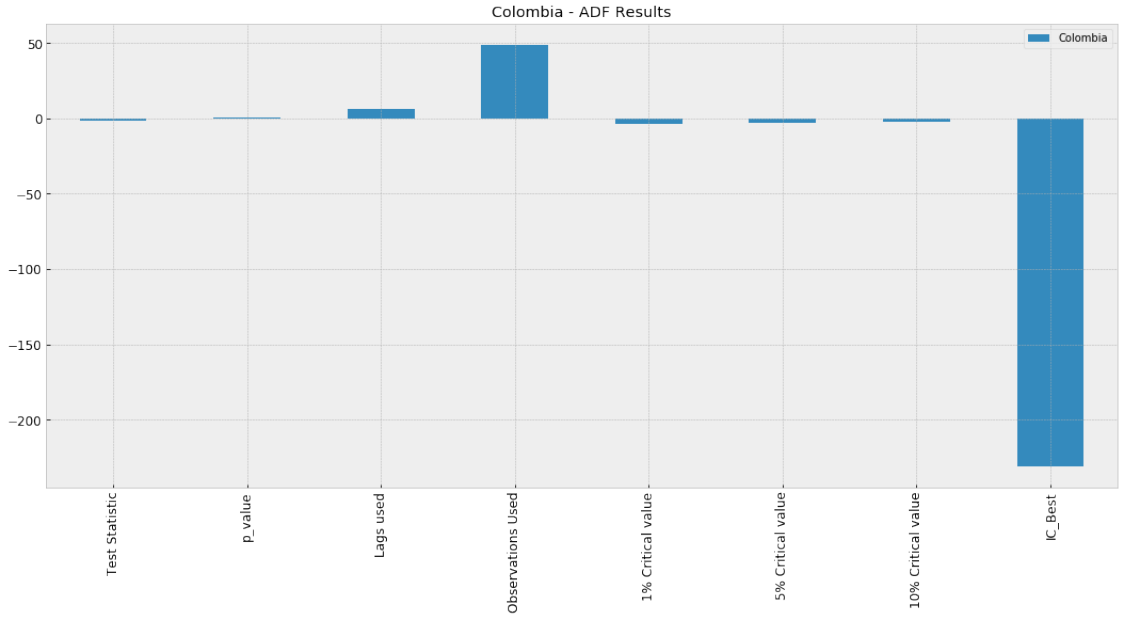

Chad - ADF Results
Chad
0
−100
−200
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

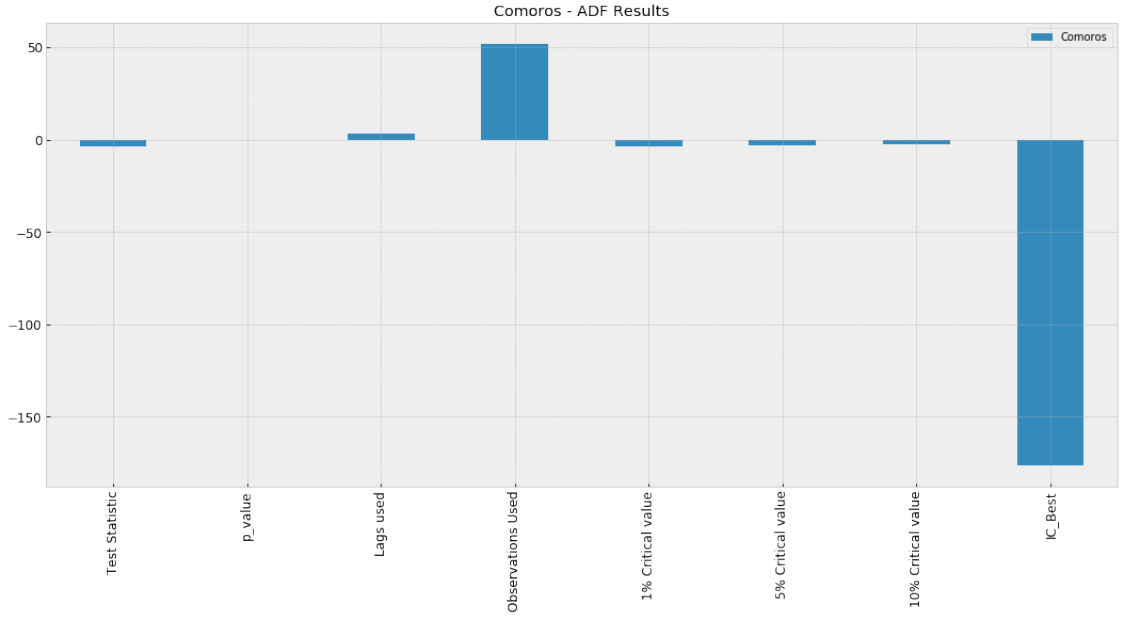

Chile - ADF Results
Chile
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

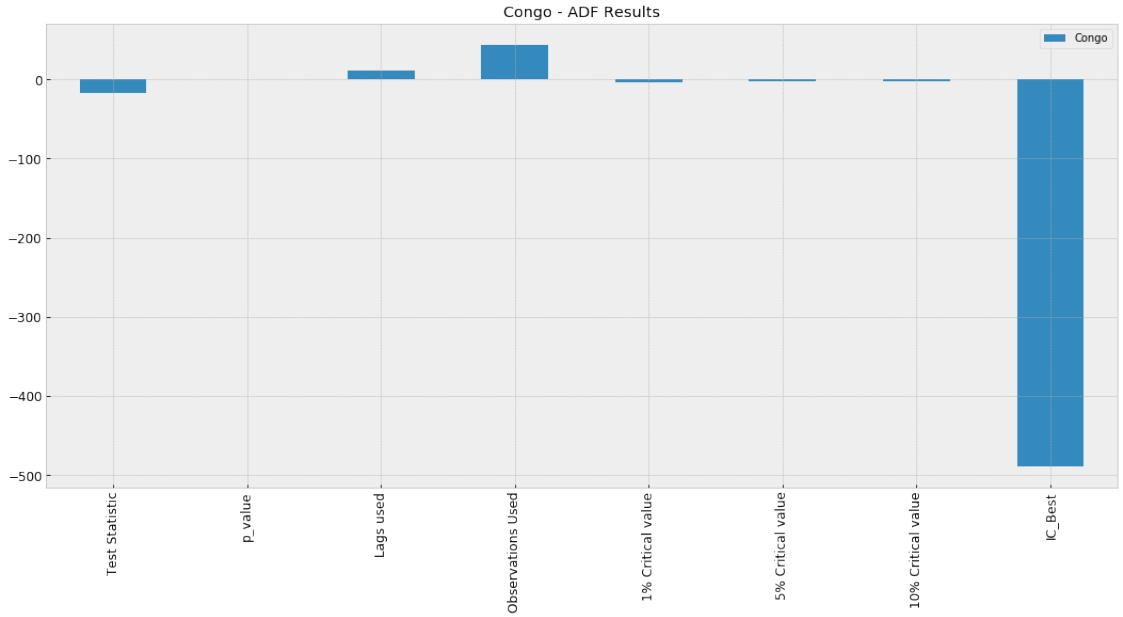

China - ADF Results
China
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

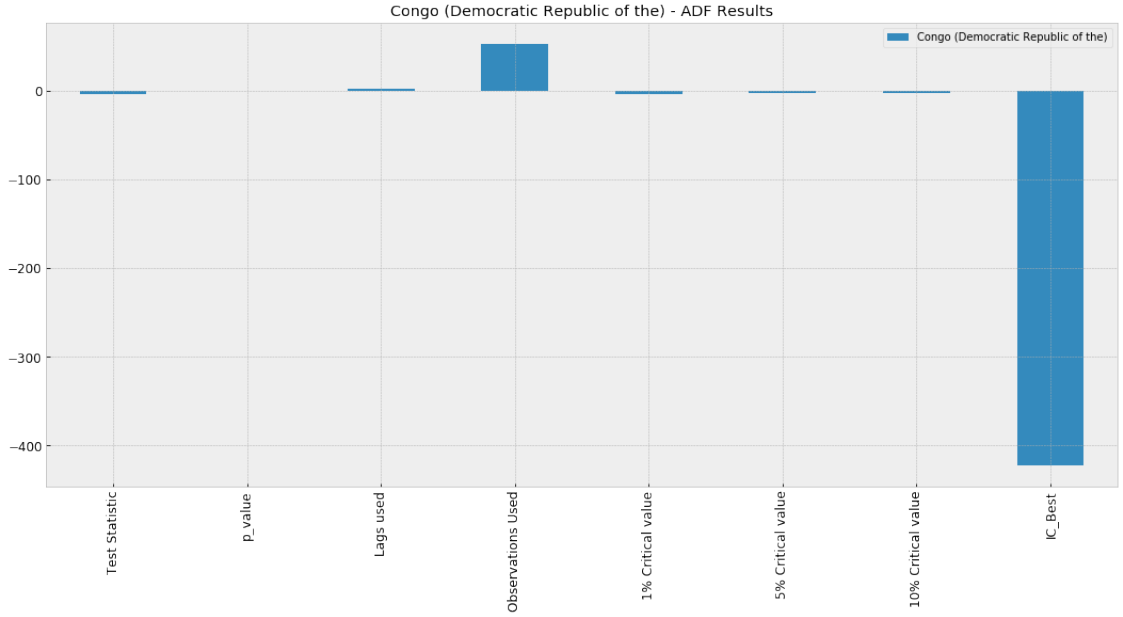

Christmas Island - ADF Results
Christmas Island
60
40
20
0
−20
−40
−60
−80
−100
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

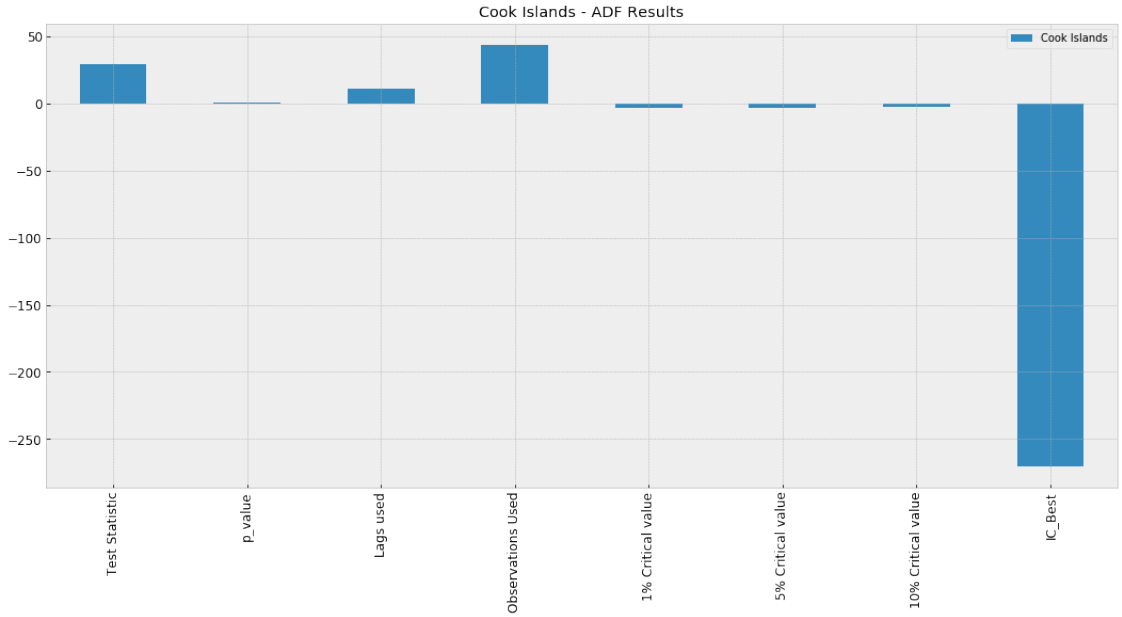

Cocos (Keeling) Islands - ADF Results
Cocos (Keeling) Islands
300
250
200
150
100
50
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

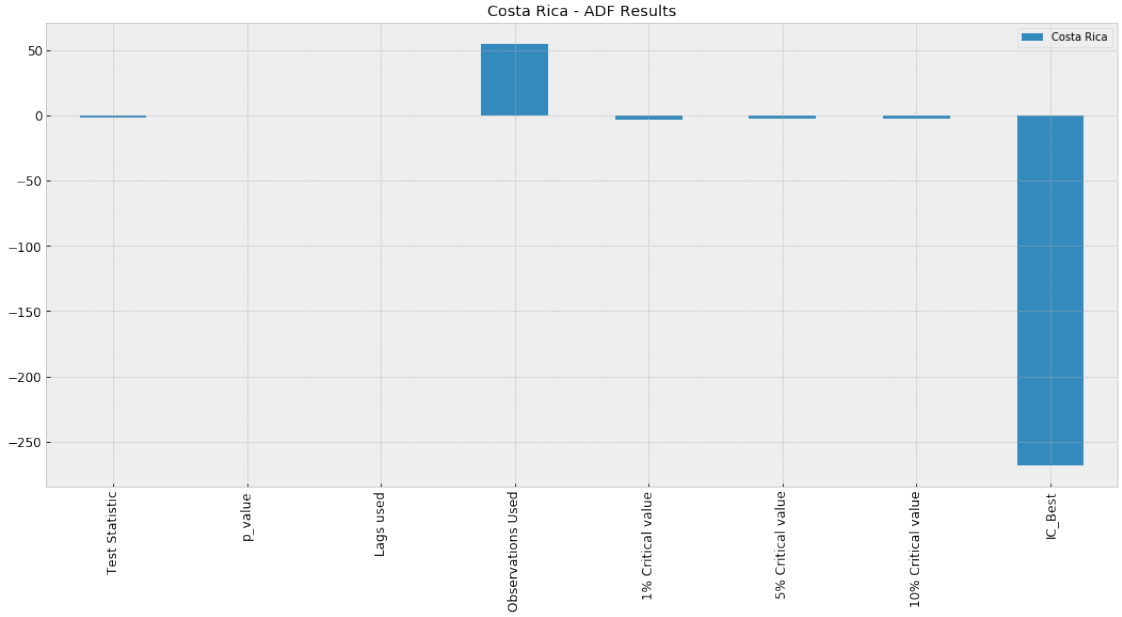

Colombia - ADF Results
Colombia
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

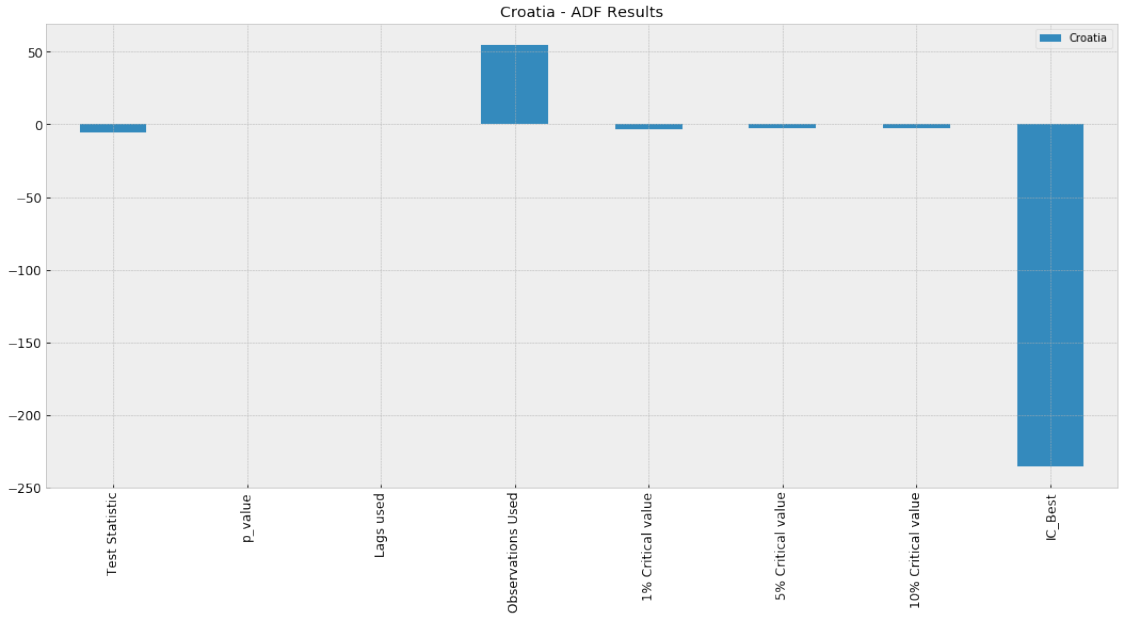

Comoros - ADF Results
Comoros
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

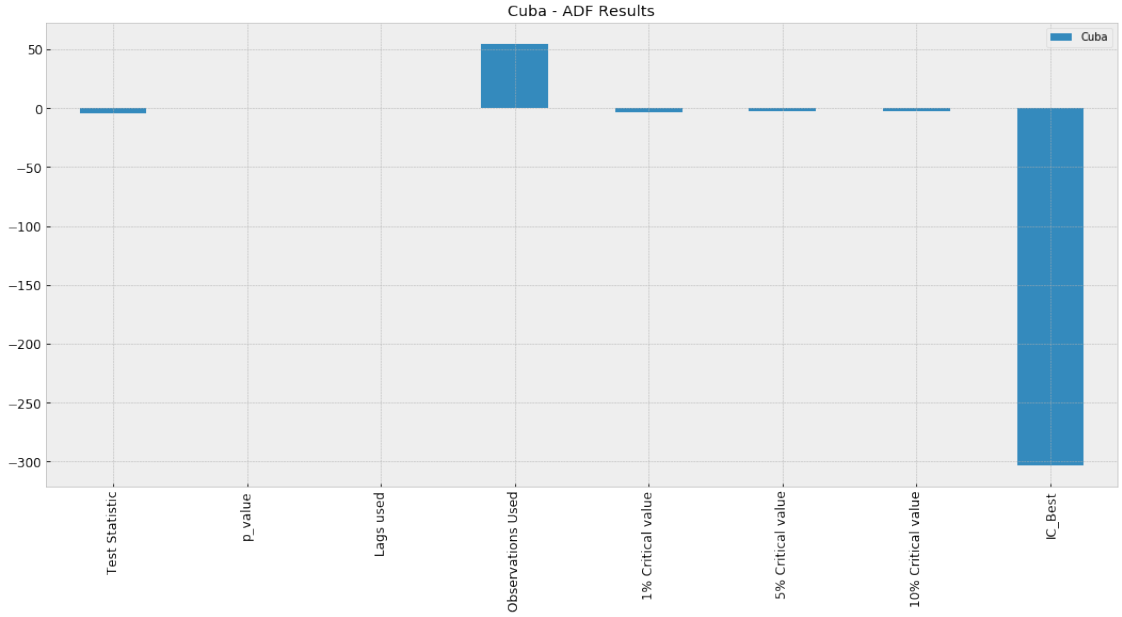

Congo - ADF Results
Congo
0
−100
−200
−300
−400
−500
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

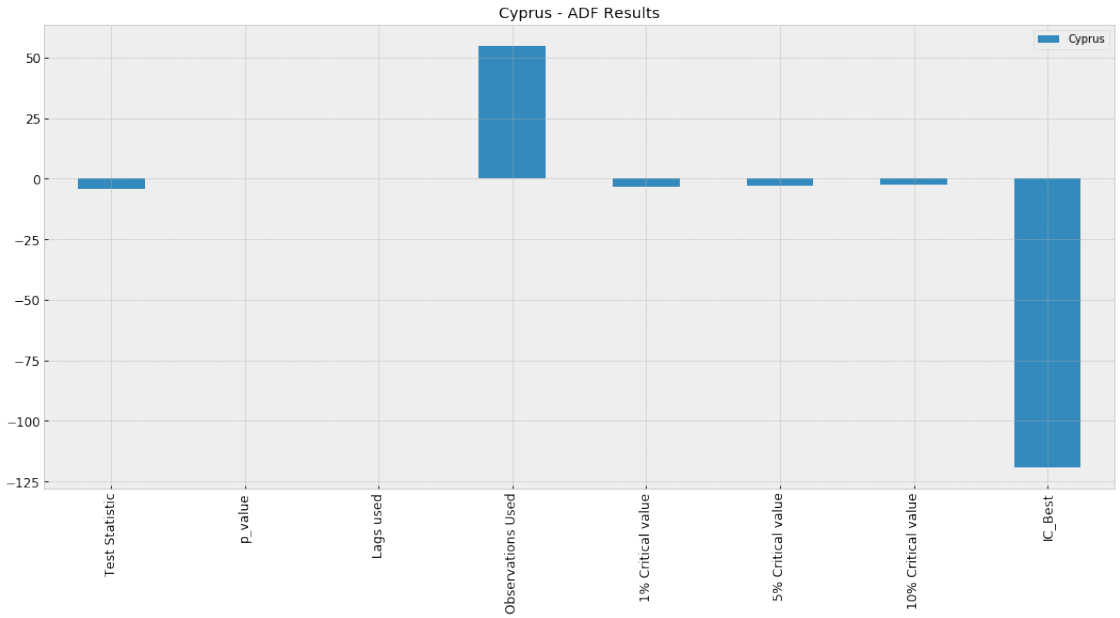

Congo (Democratic Republic of the) - ADF Results
Congo (Democratic Republic of the)
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

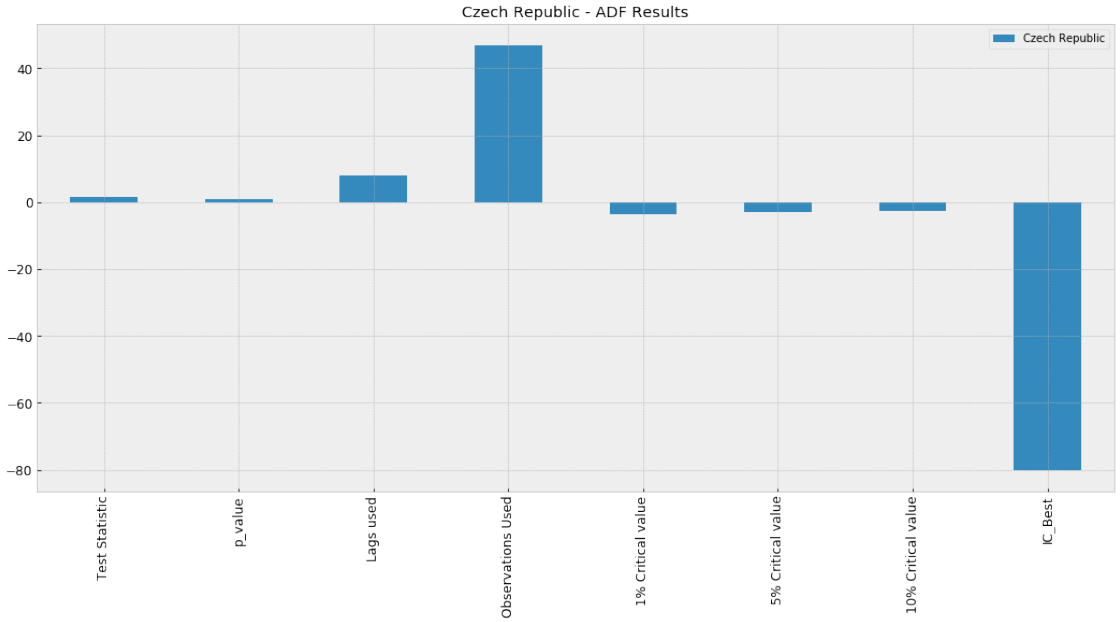

Cook Islands - ADF Results
Cook Islands
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

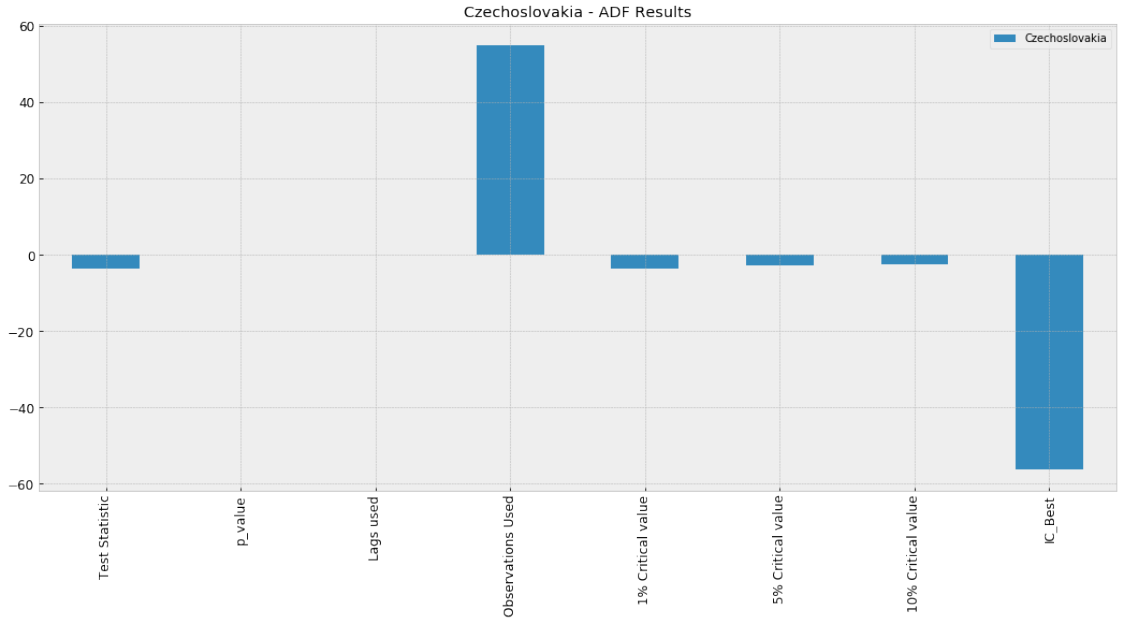

Costa Rica - ADF Results
Costa Rica
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

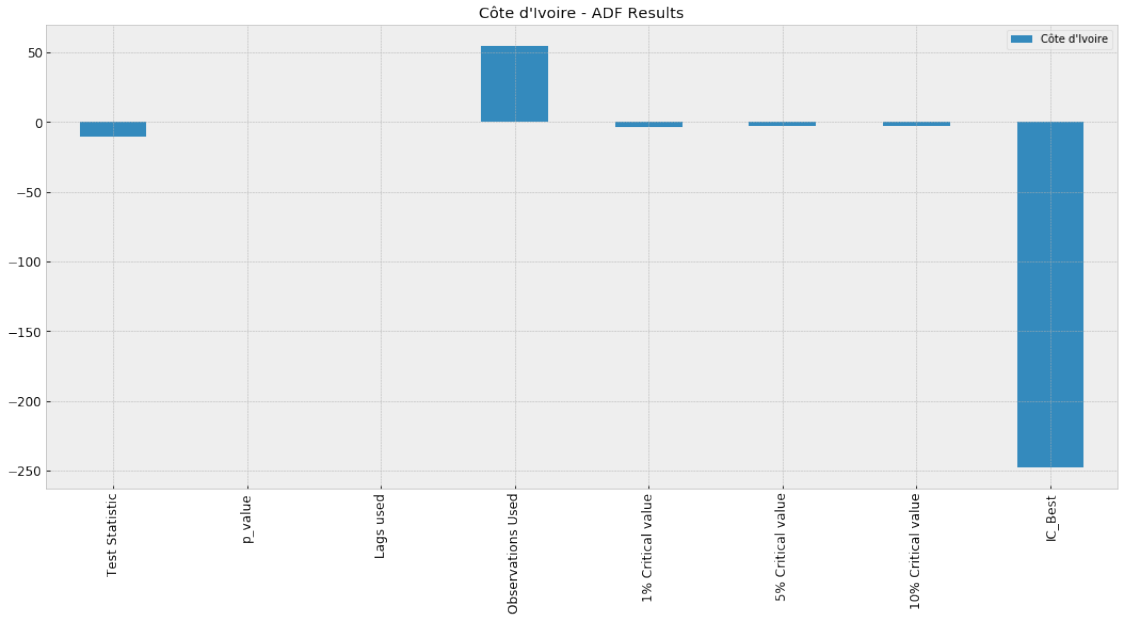

Croatia - ADF Results
Croatia
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

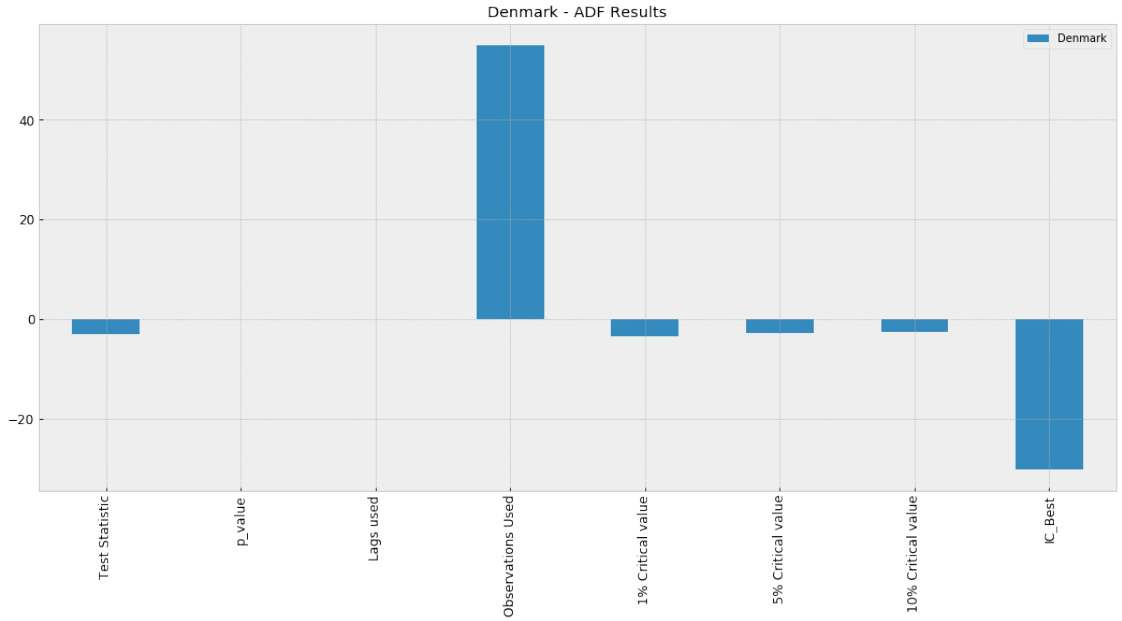

Cuba - ADF Results
Cuba
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

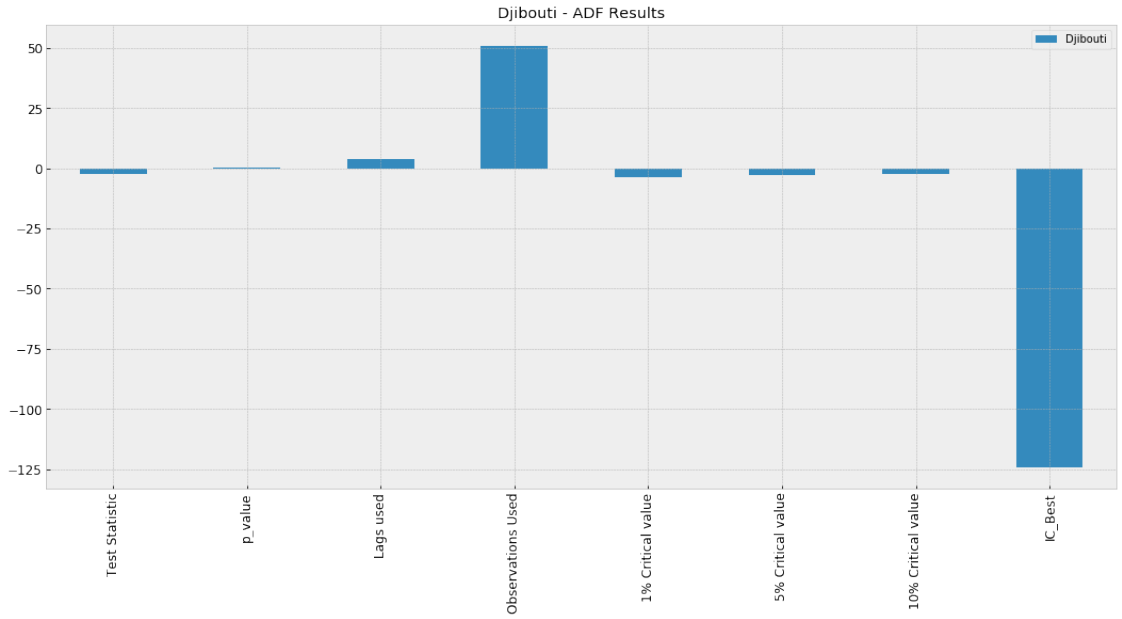

Cyprus - ADF Results
Cyprus
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

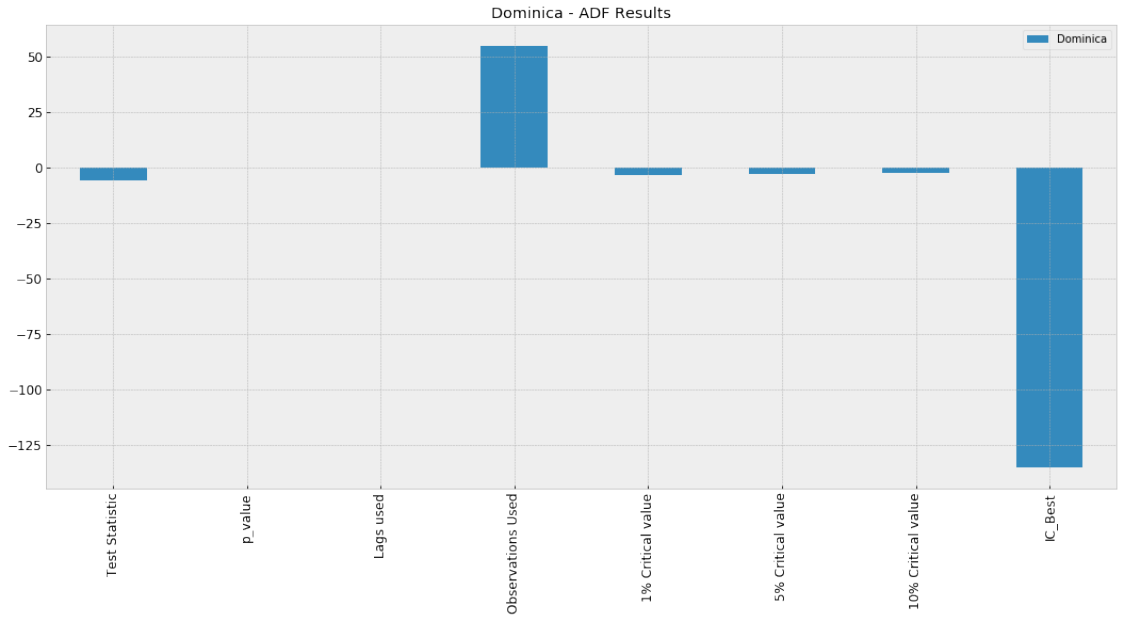

Czech Republic - ADF Results
Czech Republic
40
20
0
−20
−40
−60
−80
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

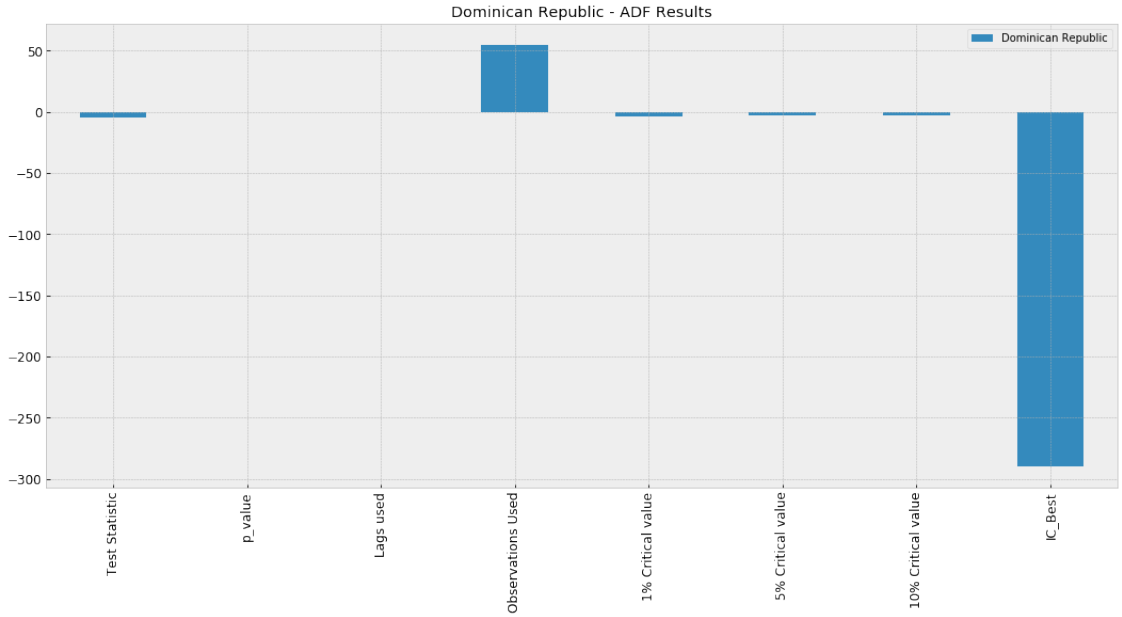

Czechoslovakia - ADF Results
Czechoslovakia
60
40
20
0
−20
−40
−60
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

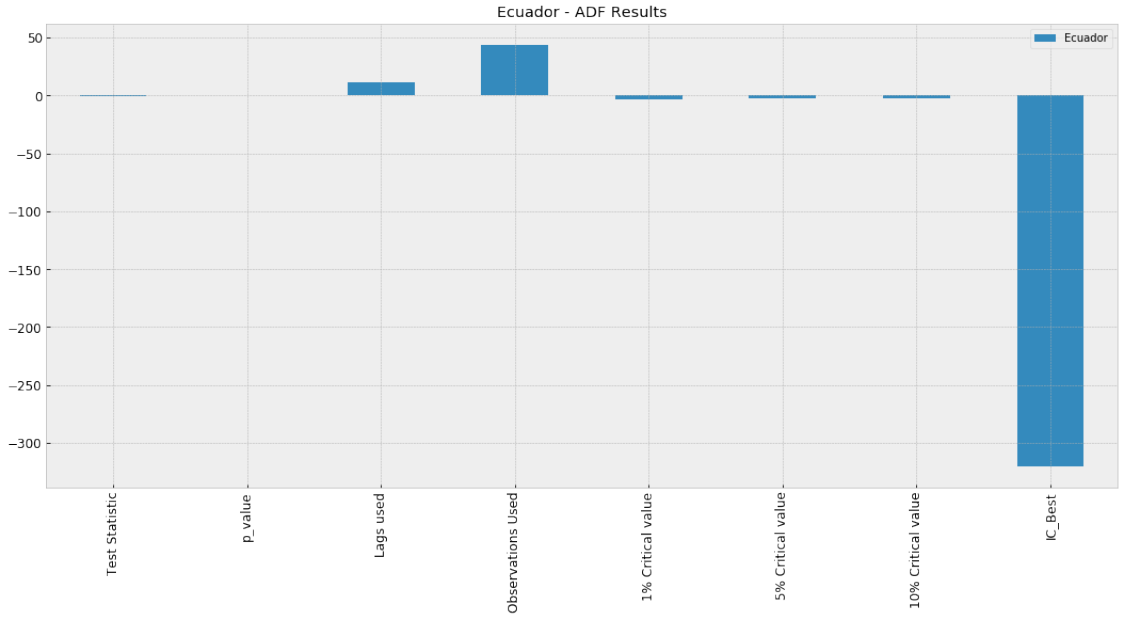

Côte d'Ivoire - ADF Results
Côte d'Ivoire
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

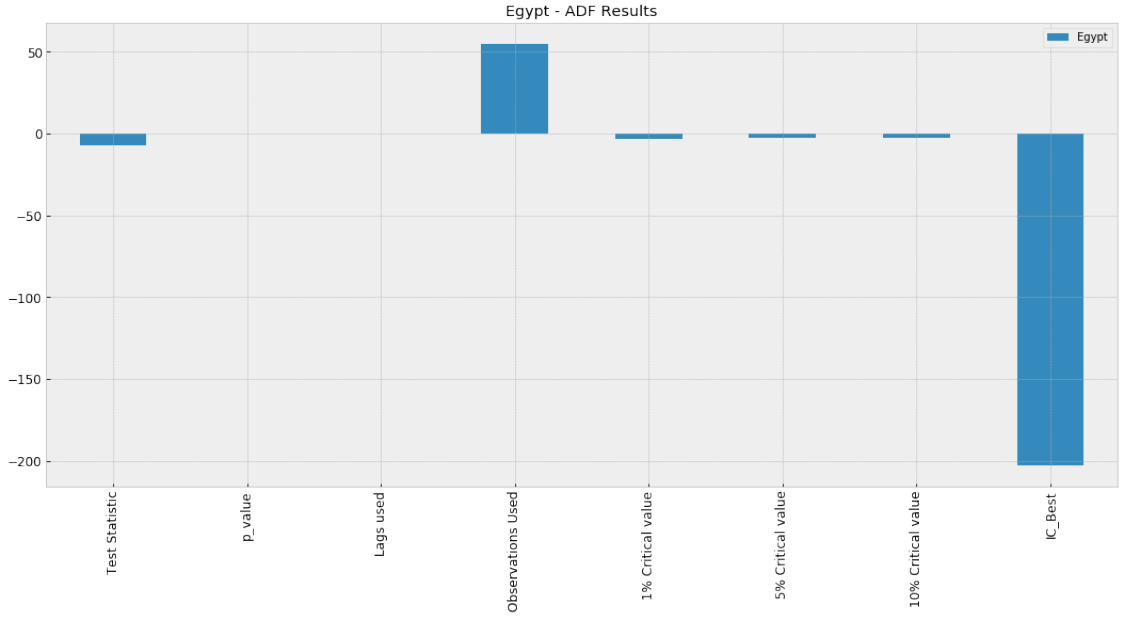

Denmark - ADF Results
Denmark
40
20
0
−20
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

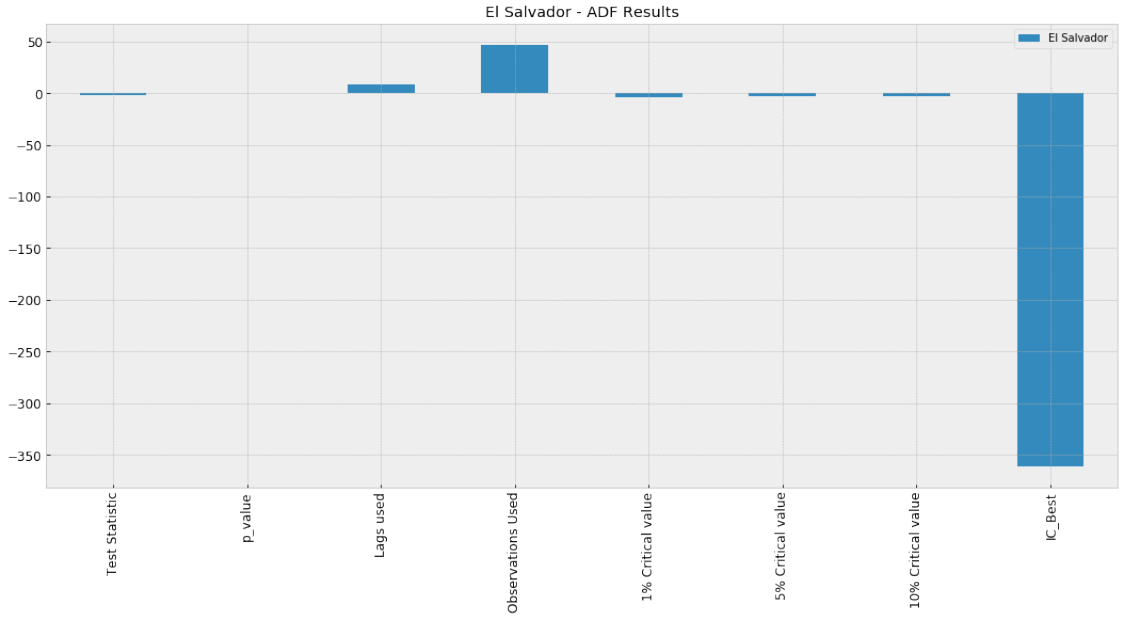

Djibouti - ADF Results
Djibouti
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

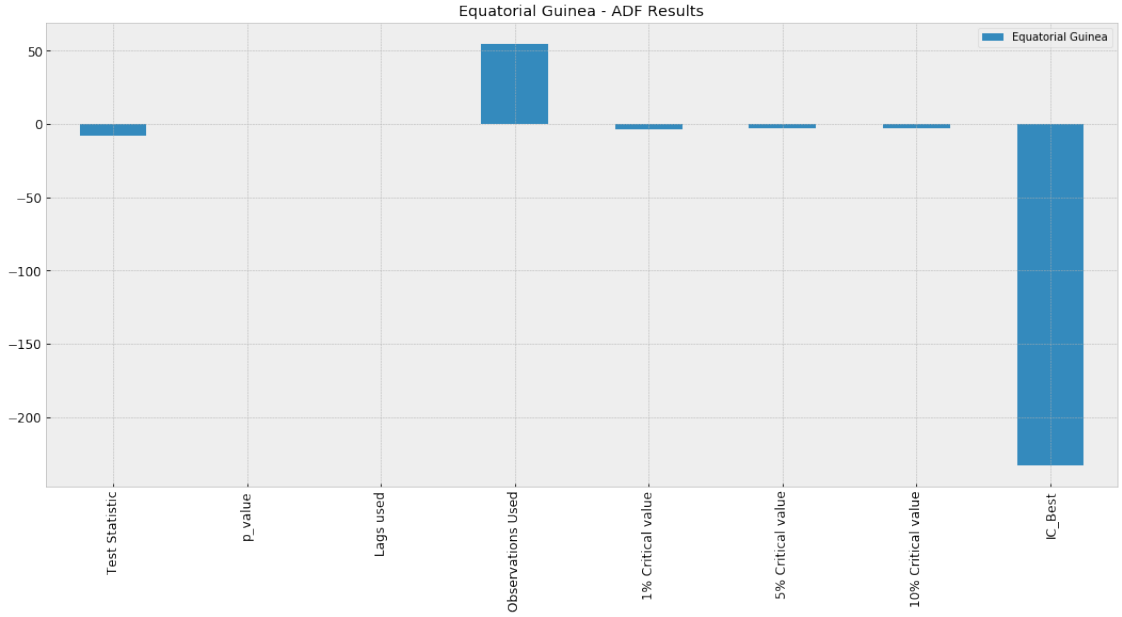

Dominica - ADF Results
Dominica
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

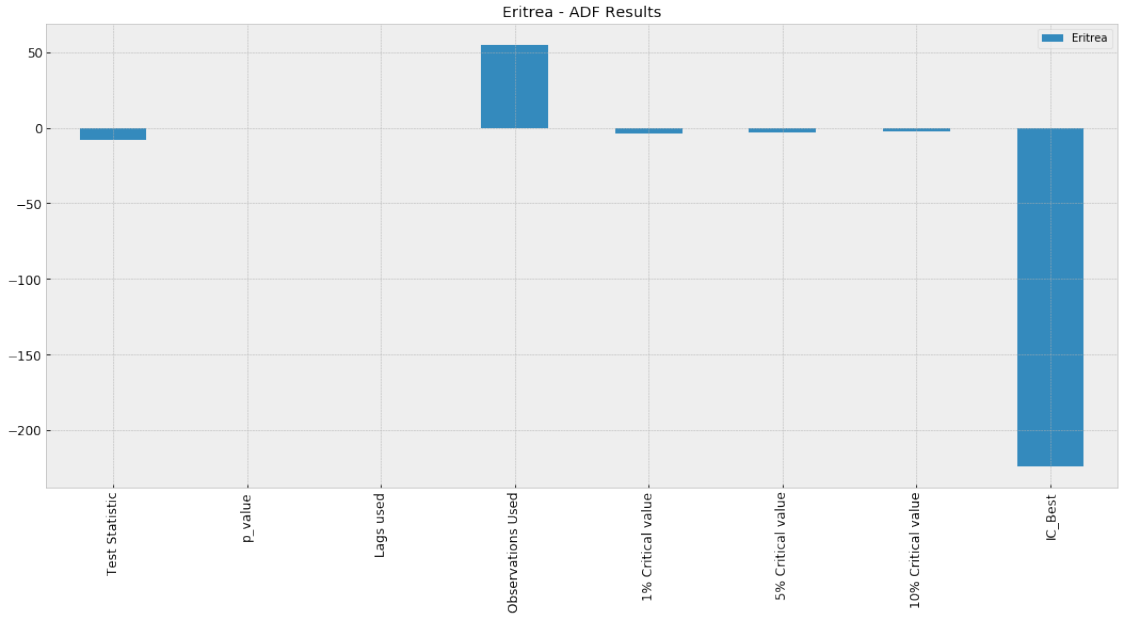

Dominican Republic - ADF Results
Dominican Republic
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

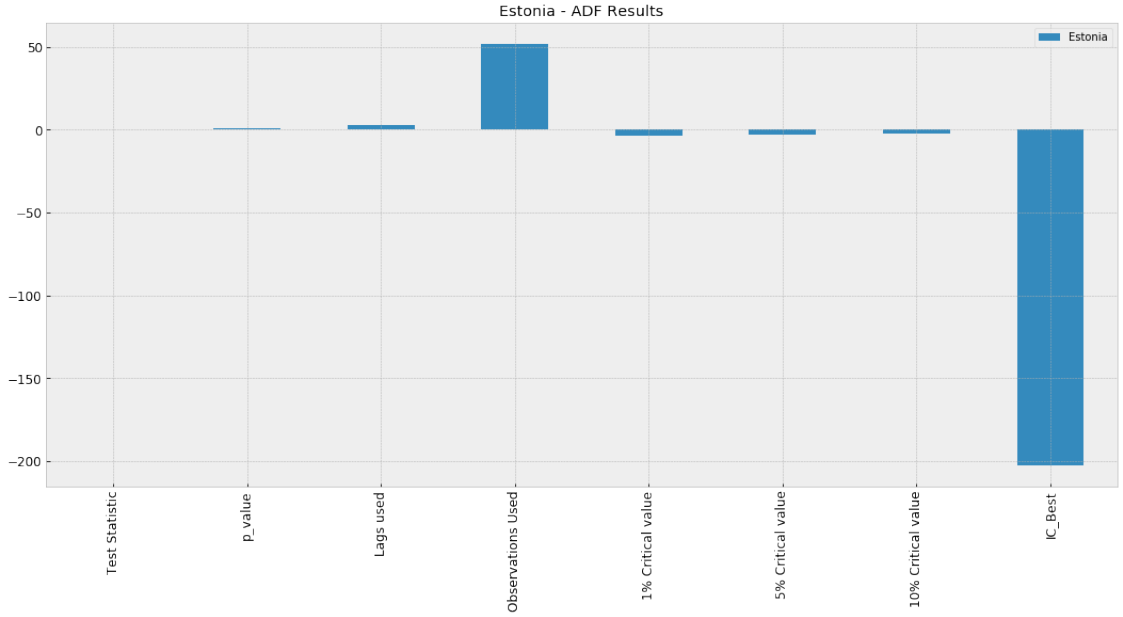

Ecuador - ADF Results
Ecuador
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

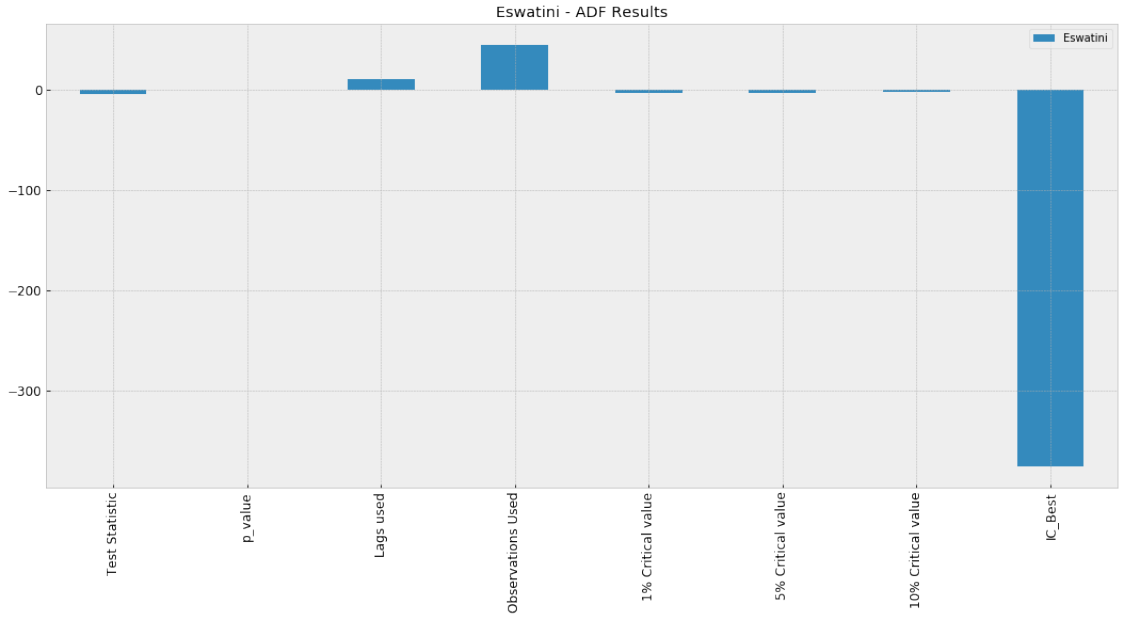

Egypt - ADF Results
Egypt
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

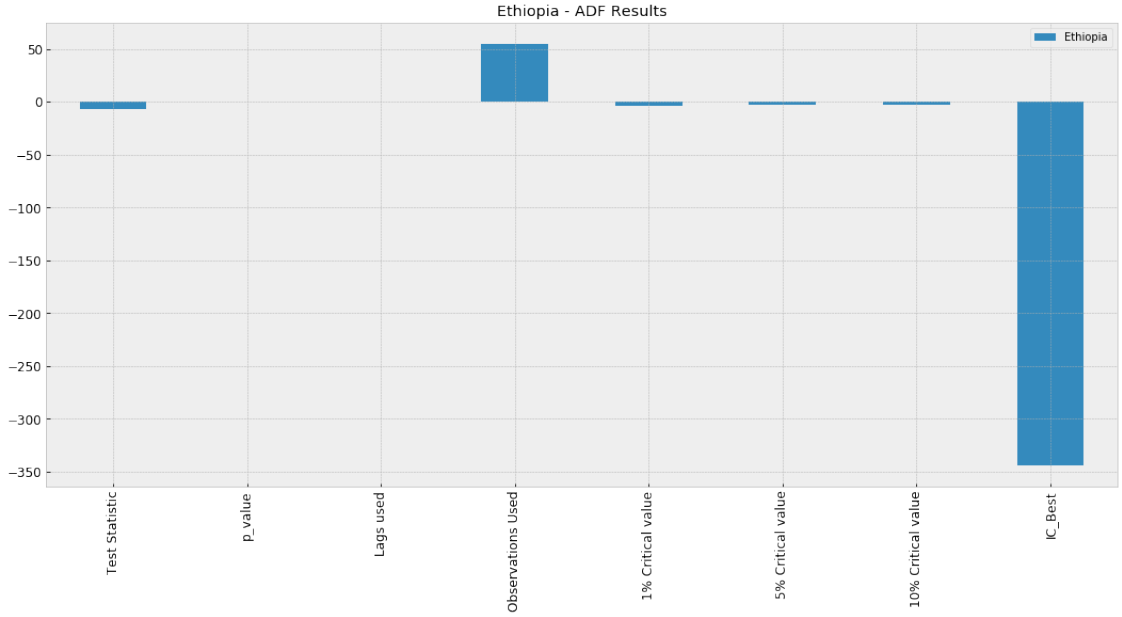

El Salvador - ADF Results
El Salvador
50
0
−50
−100
−150
−200
−250
−300
−350
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

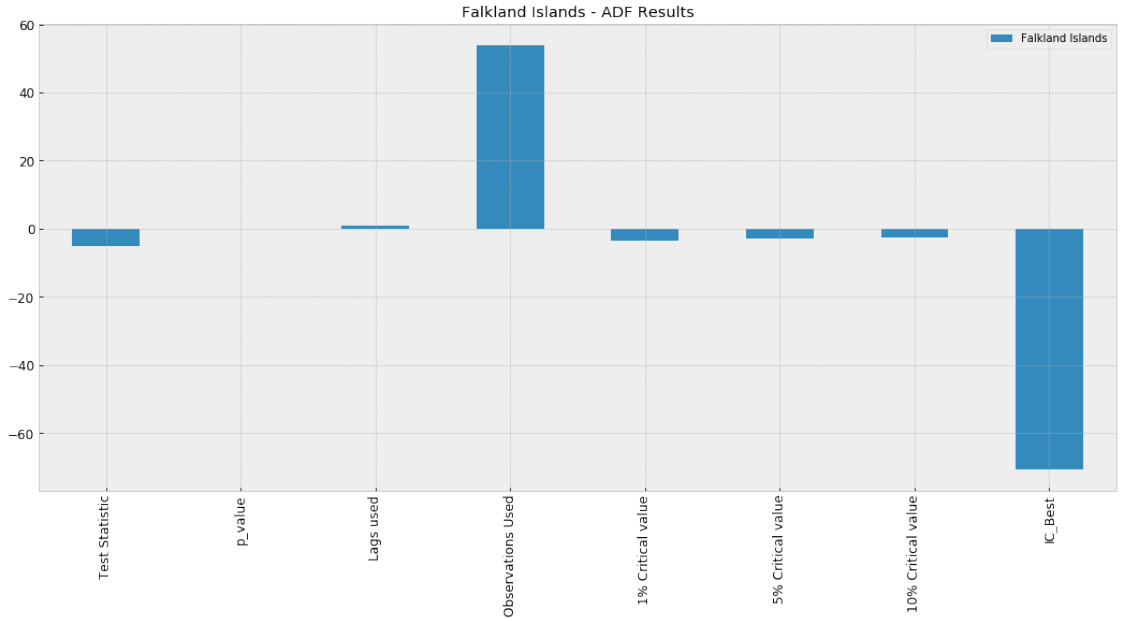

Equatorial Guinea - ADF Results
Equatorial Guinea
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

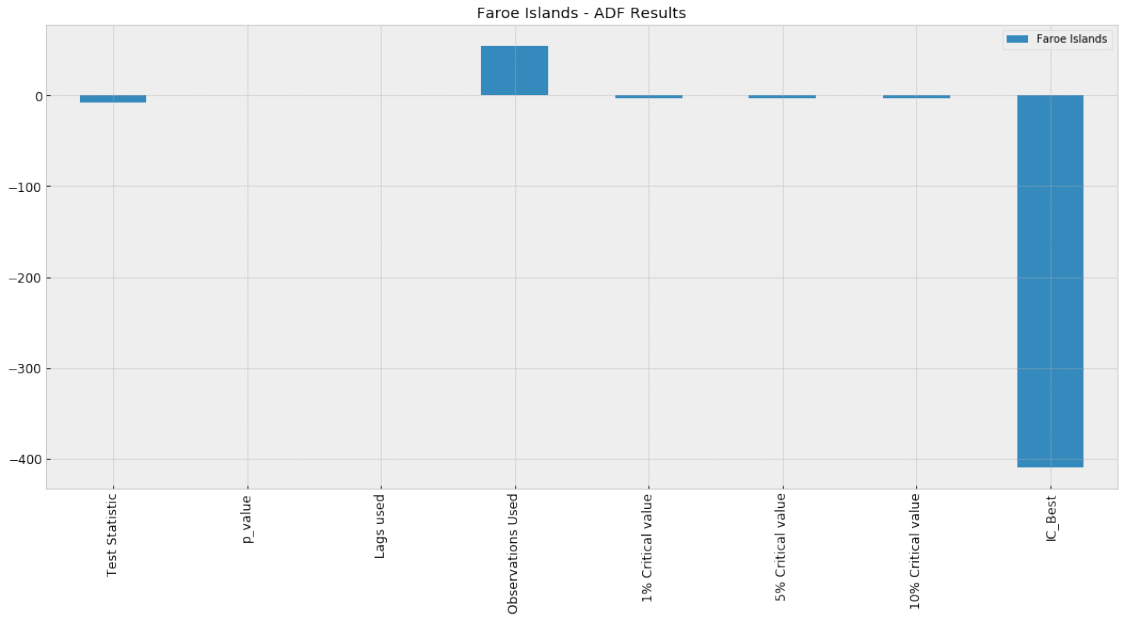

Eritrea - ADF Results
Eritrea
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

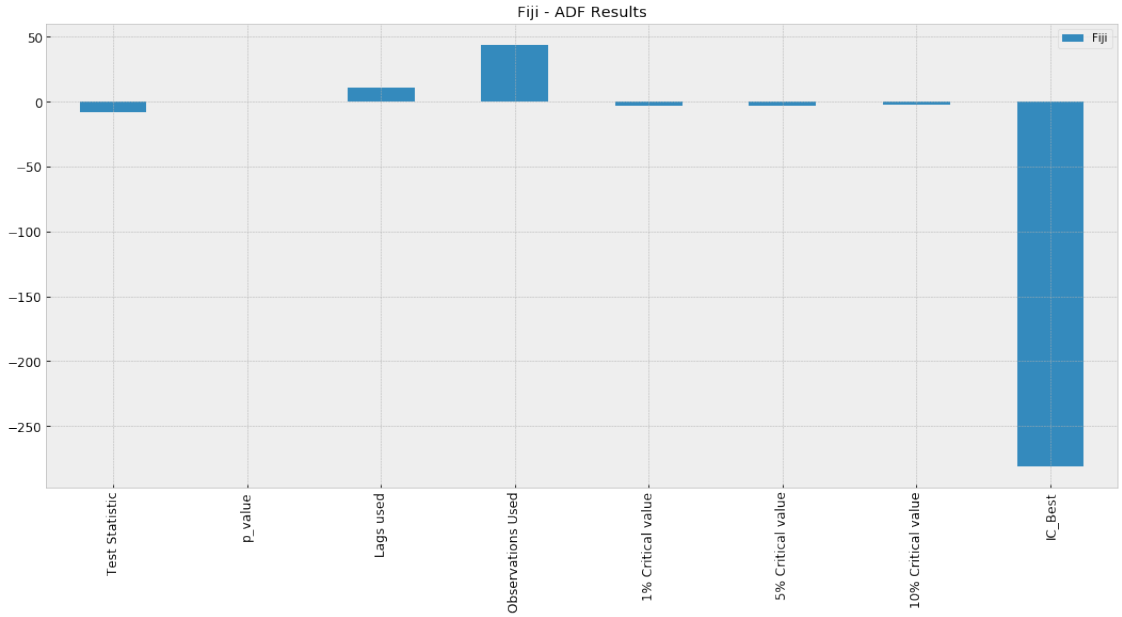

Estonia - ADF Results
Estonia
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

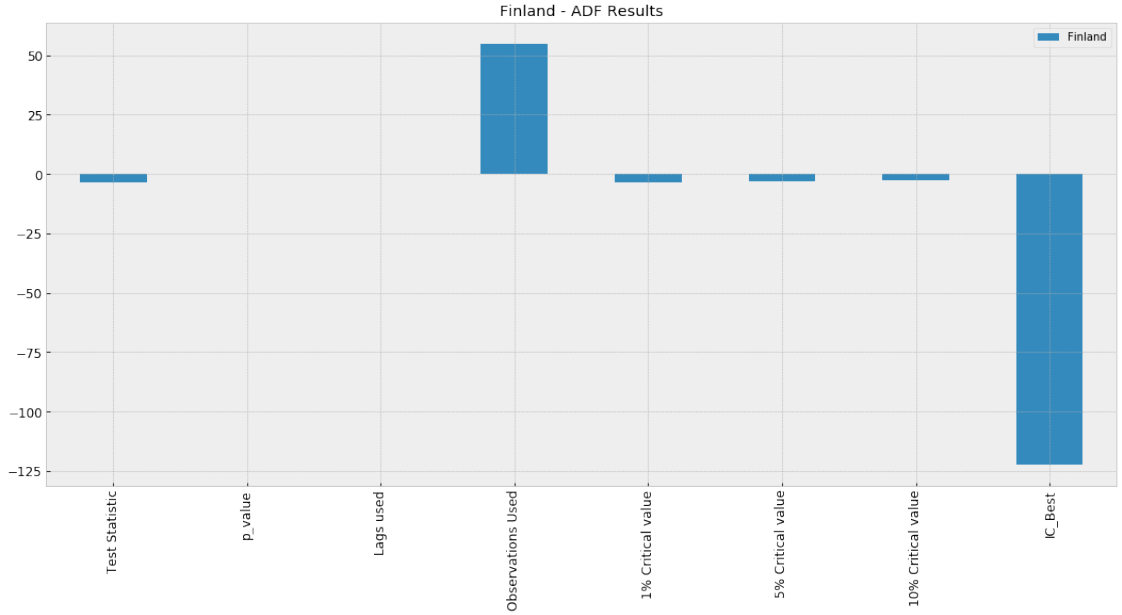

Eswatini - ADF Results
Eswatini
0
−100
−200
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

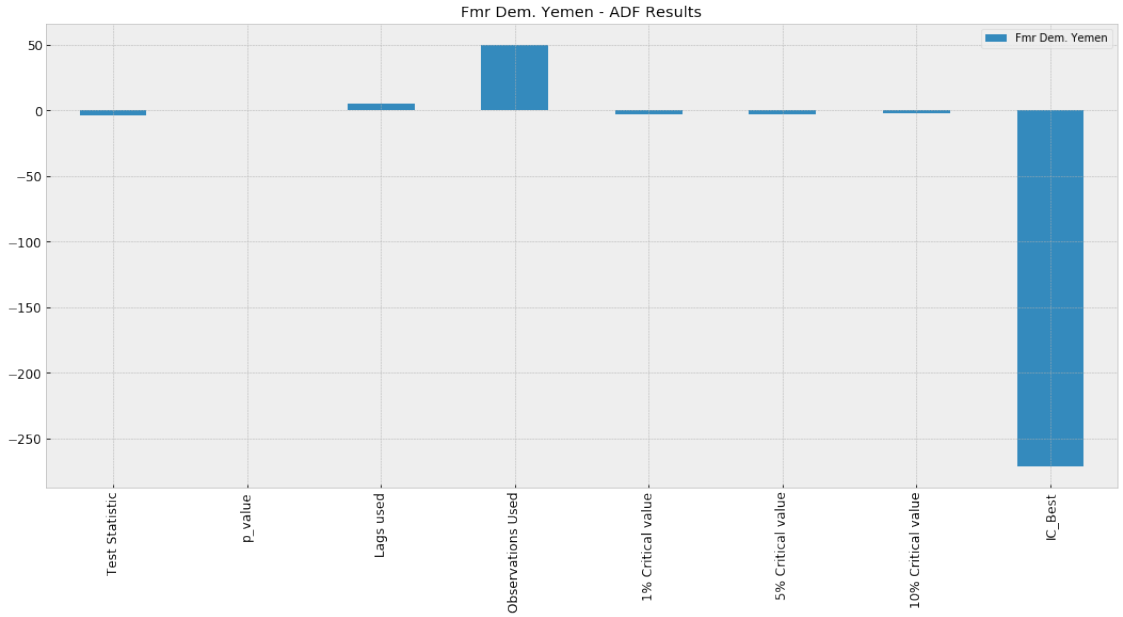

Ethiopia - ADF Results
Ethiopia
50
0
−50
−100
−150
−200
−250
−300
−350
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

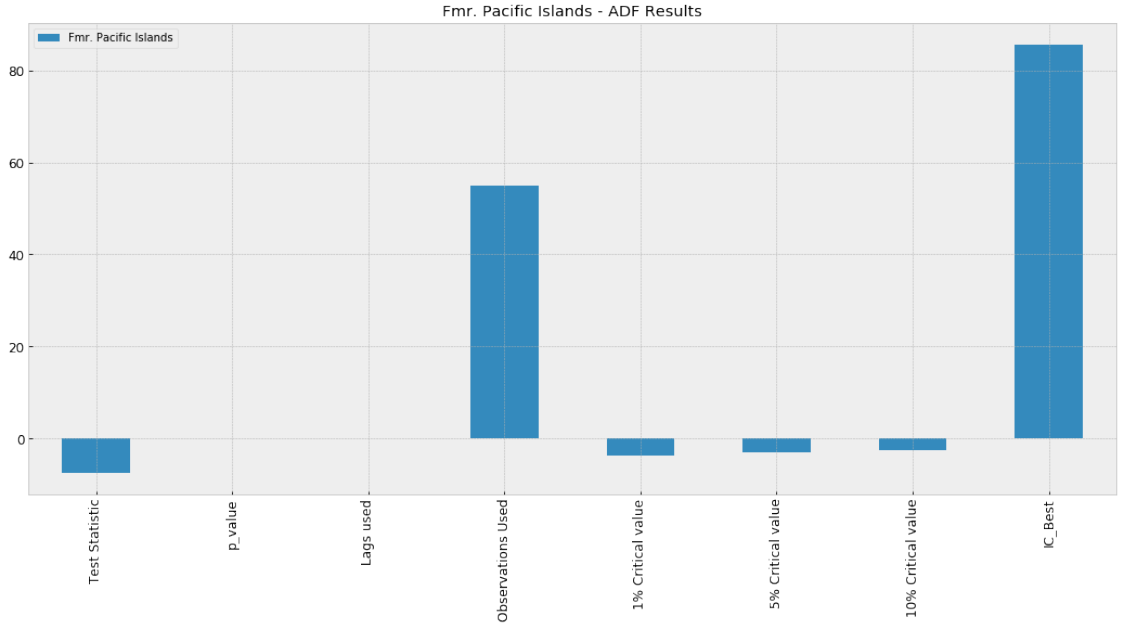

Falkland Islands - ADF Results
Falkland Islands
60
40
20
0
−20
−40
−60
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

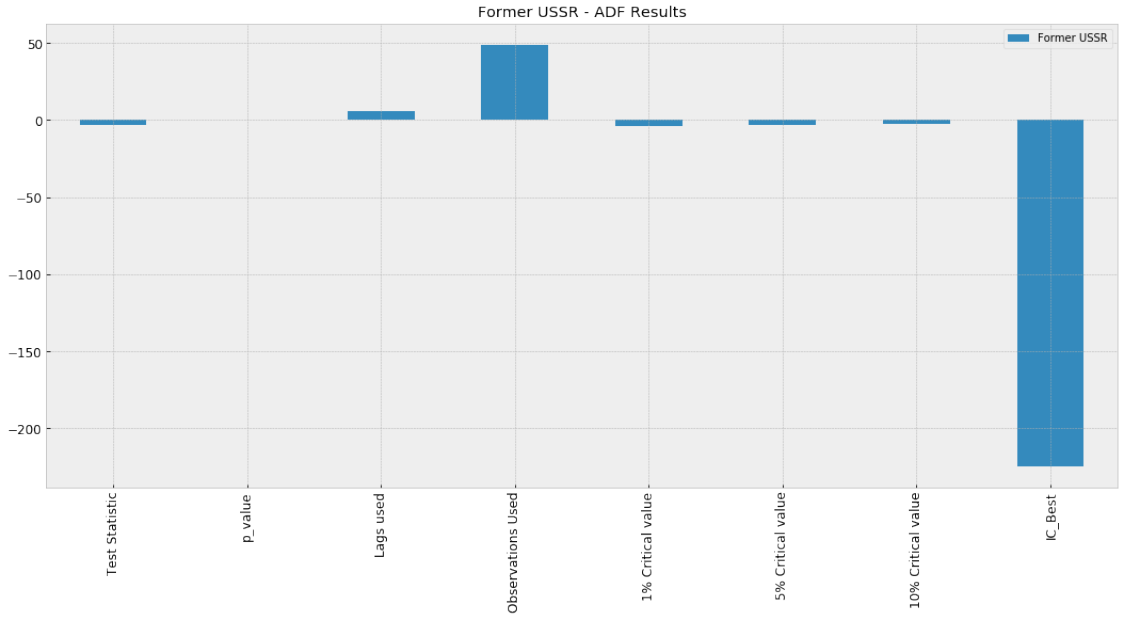

Faroe Islands - ADF Results
Faroe Islands
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

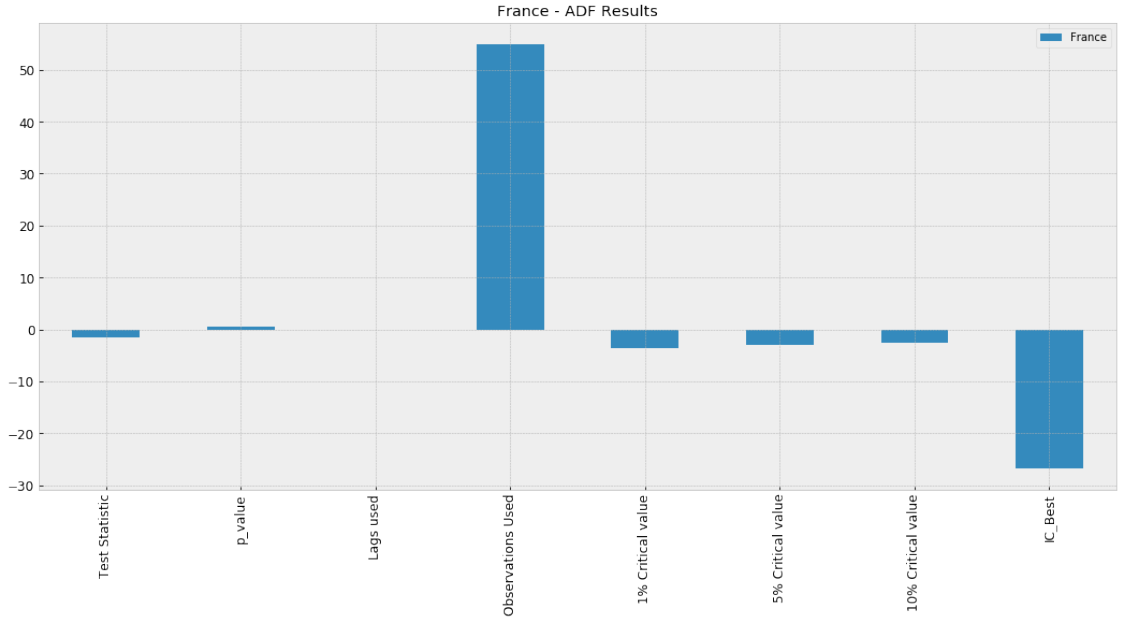

Fiji - ADF Results
Fiji
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

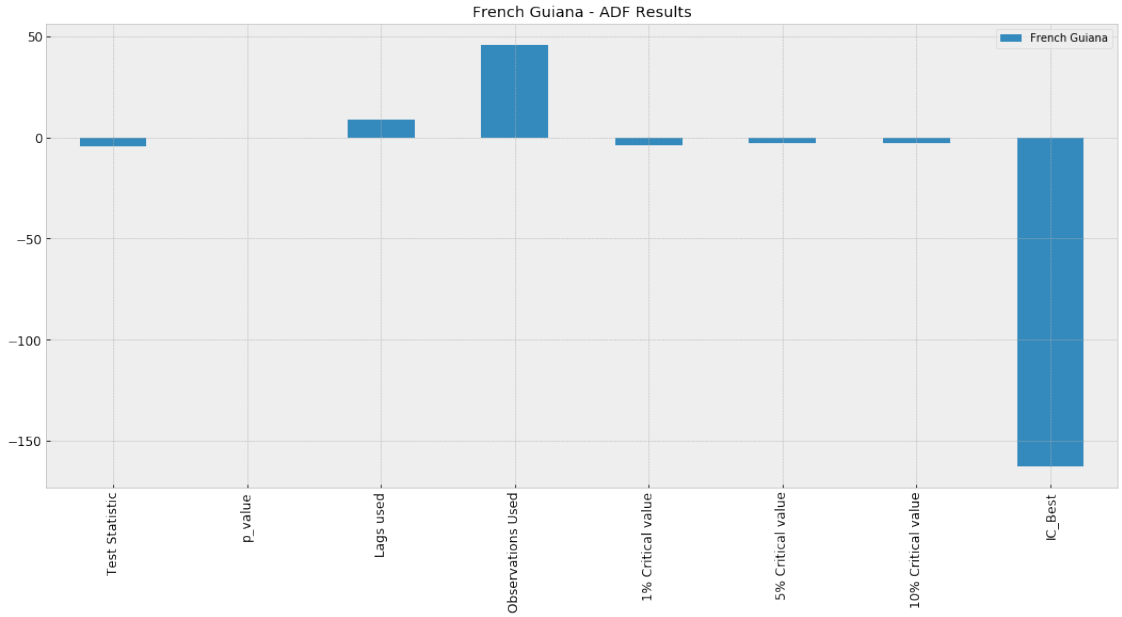

Finland - ADF Results
Finland
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

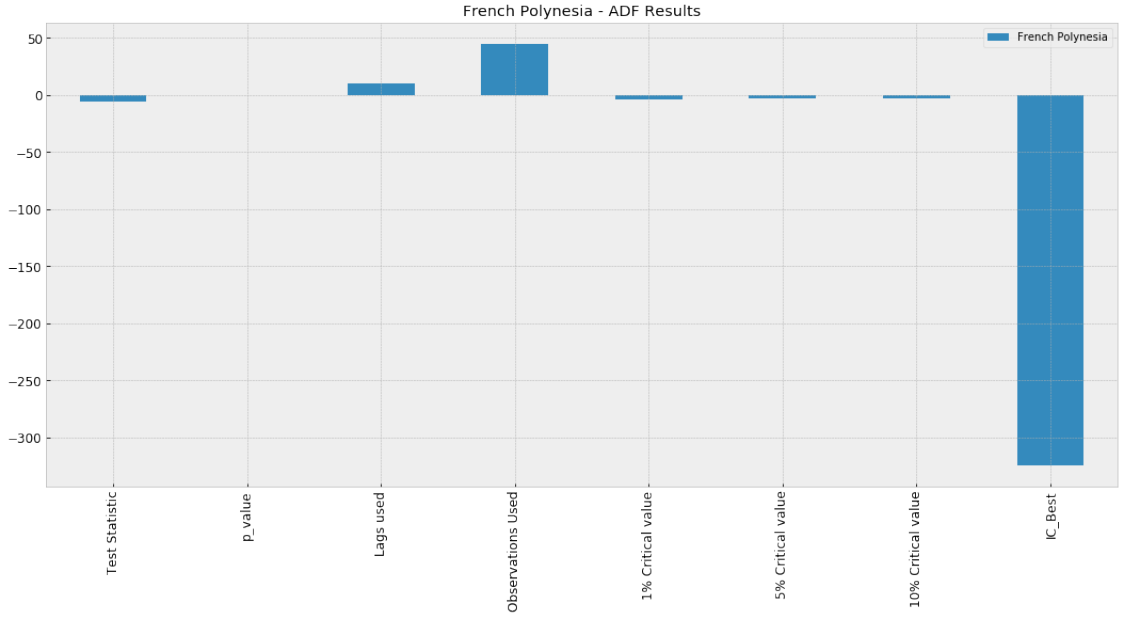

Fmr Dem. Yemen - ADF Results
Fmr Dem. Yemen
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

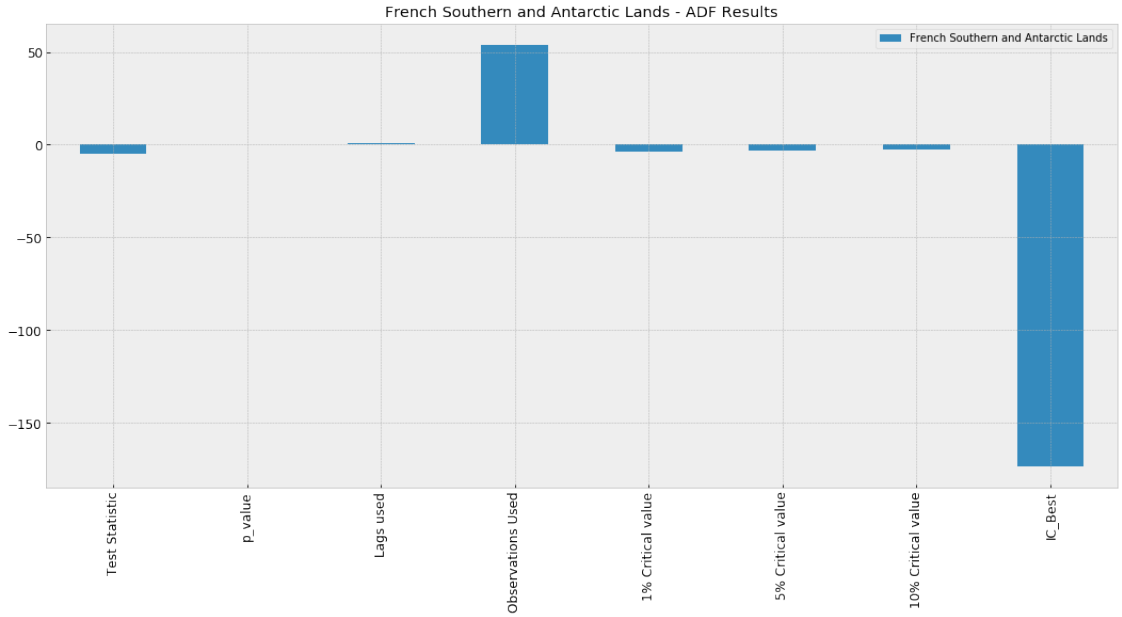

Fmr. Pacific Islands - ADF Results
Fmr. Pacific Islands
80
60
40
20
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

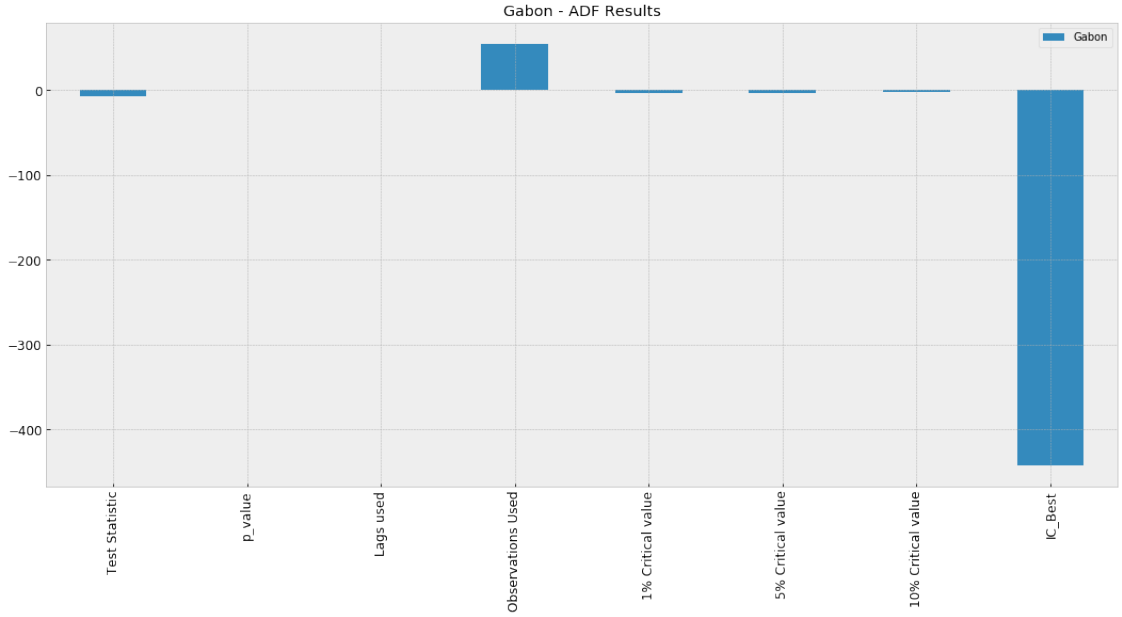

Former USSR - ADF Results
Former USSR
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

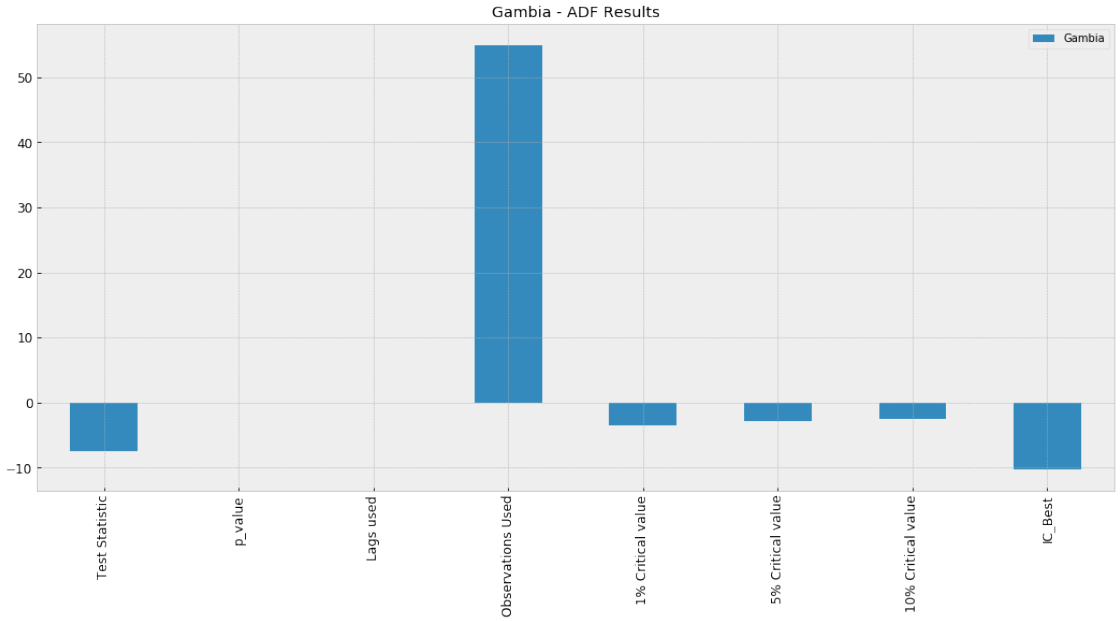

France - ADF Results
France
50
40
30
20
10
0
−10
−20
−30
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

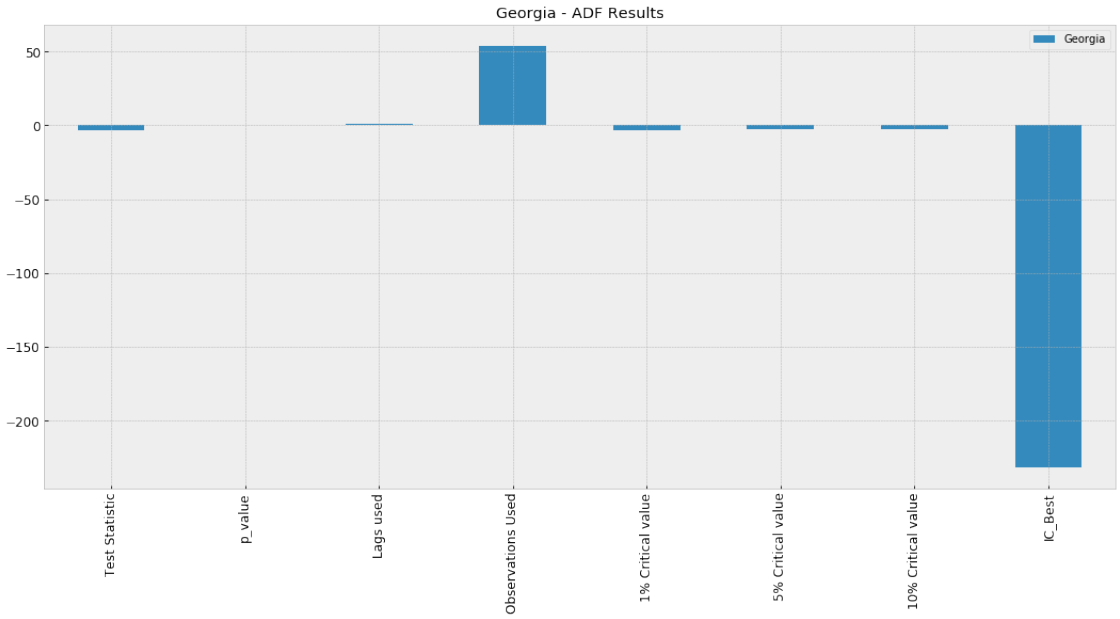

French Guiana - ADF Results
French Guiana
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

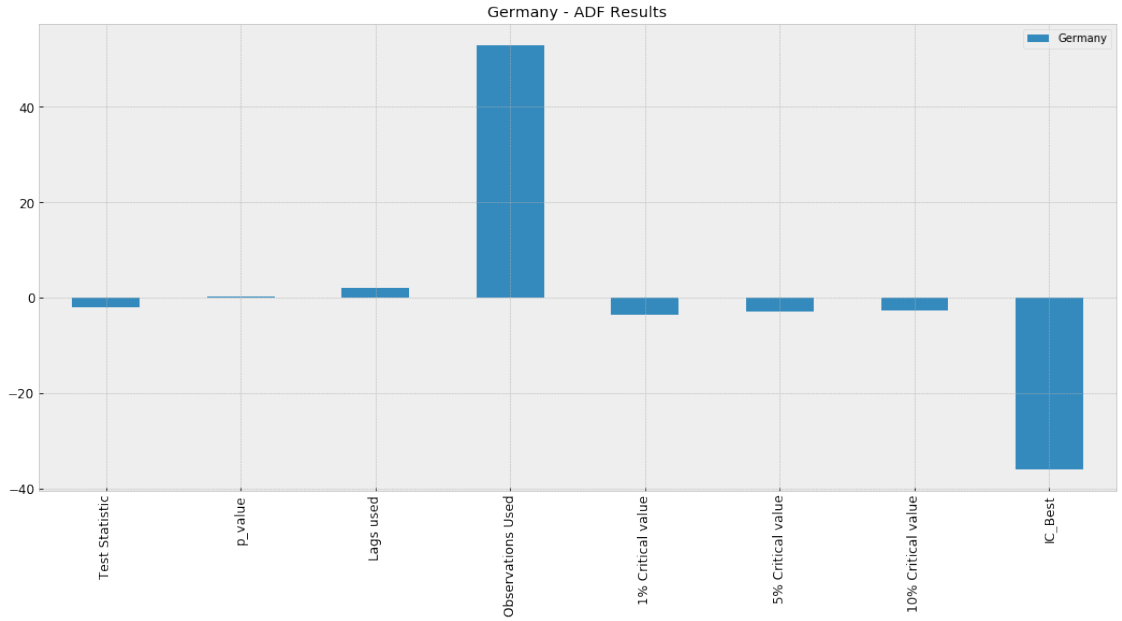

French Polynesia - ADF Results
French Polynesia
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

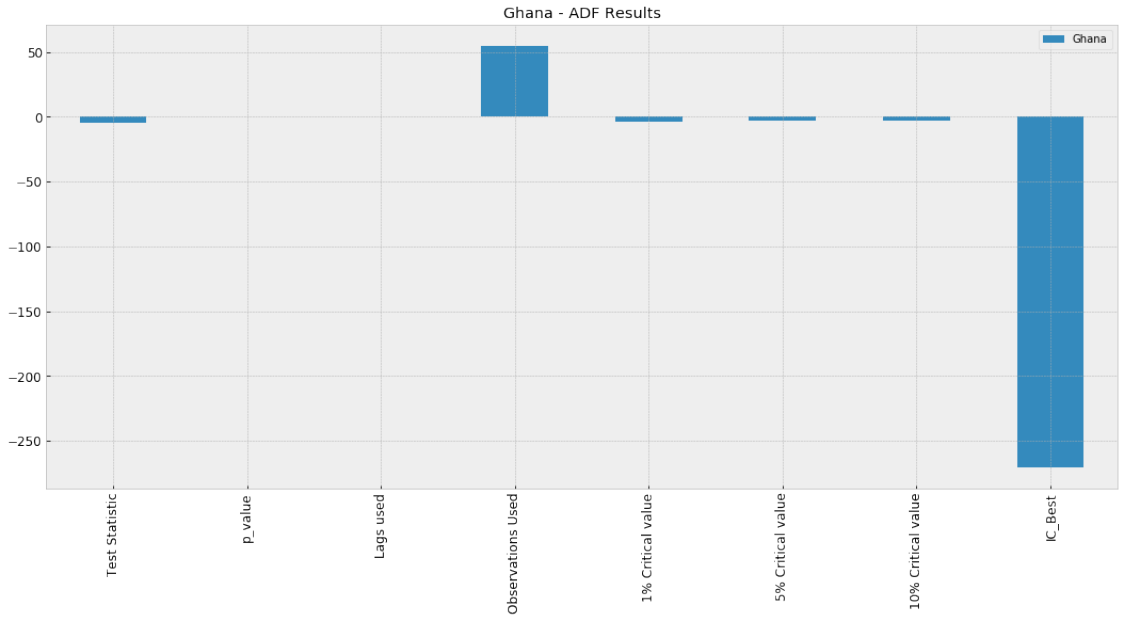

French Southern and Antarctic Lands - ADF Results
French Southern and Antarctic Lands
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

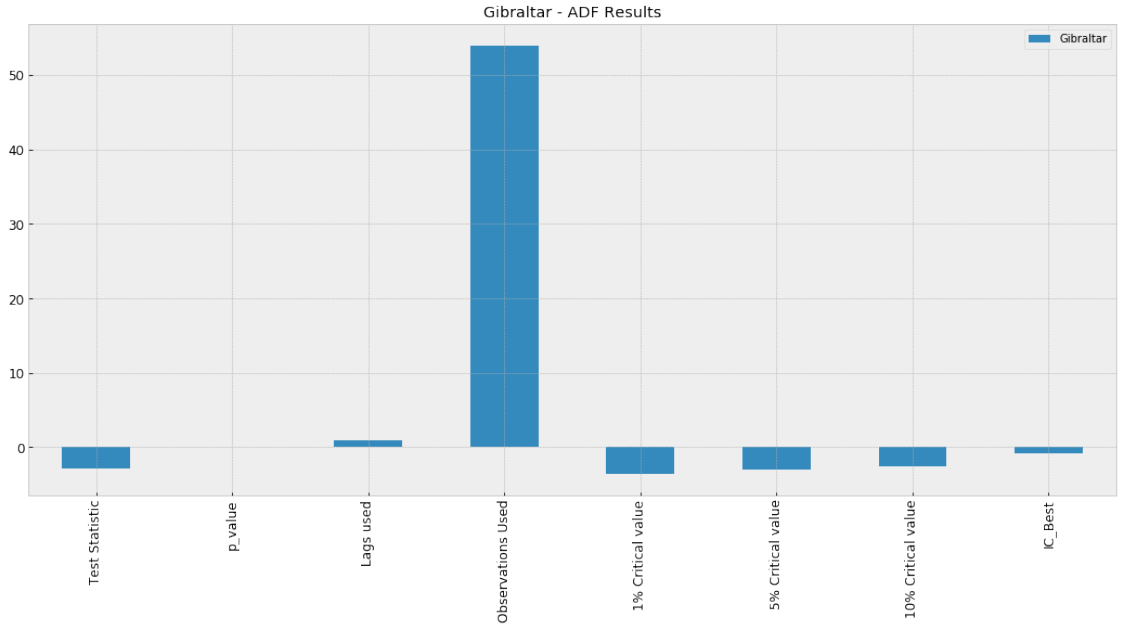

Gabon - ADF Results
Gabon
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

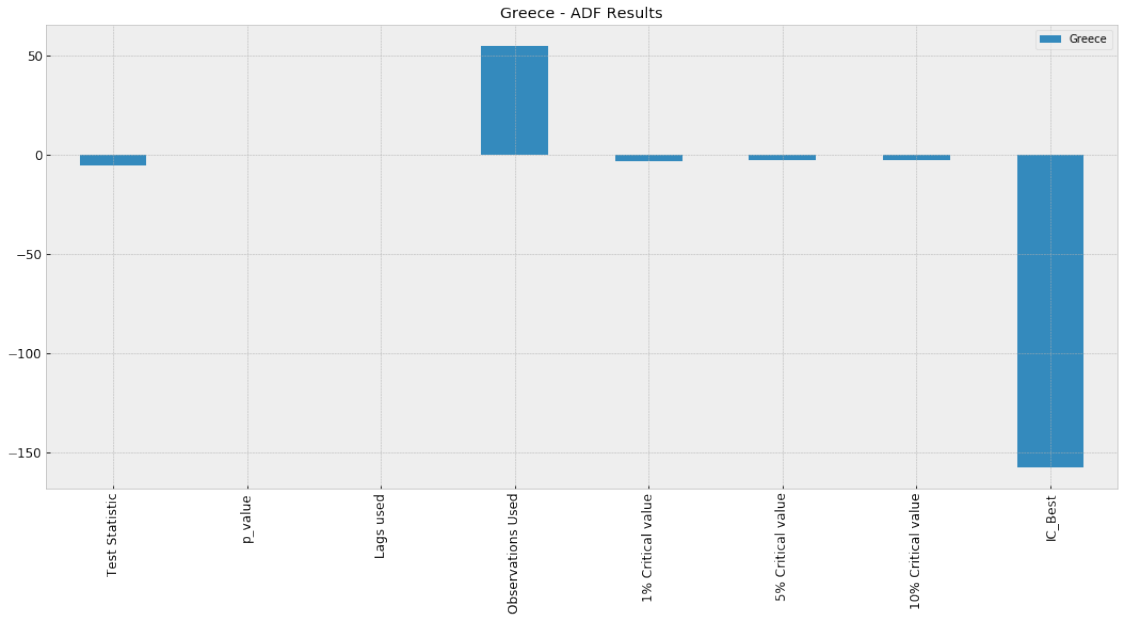

Gambia - ADF Results
Gambia
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

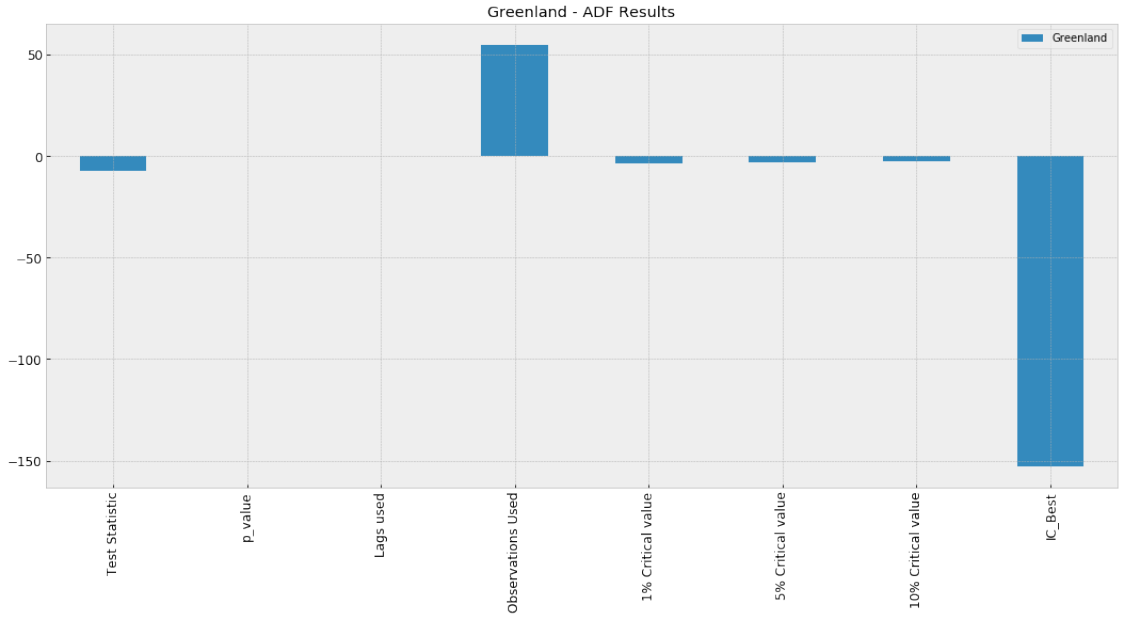

Georgia - ADF Results
Georgia
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

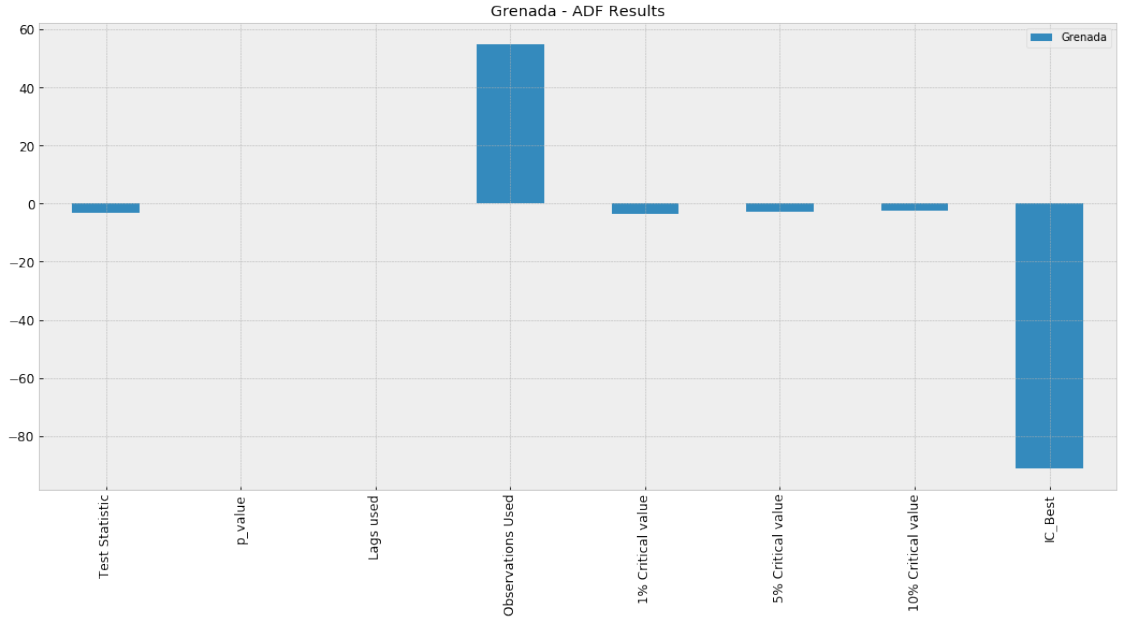

Germany - ADF Results
Germany
40
20
0
−20
−40
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

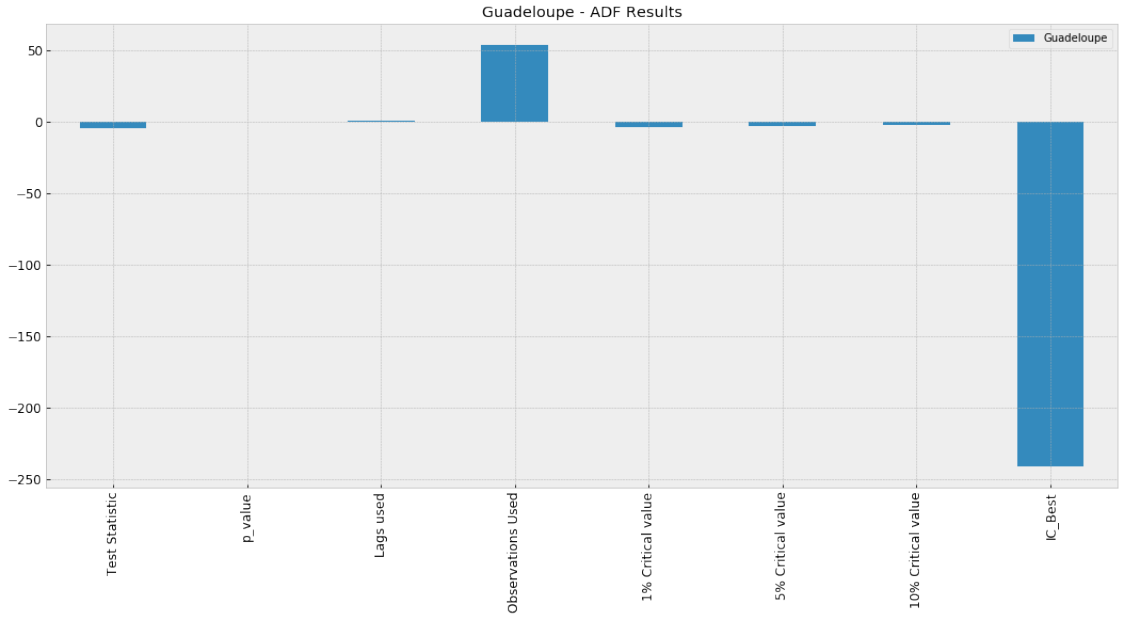

Ghana - ADF Results
Ghana
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

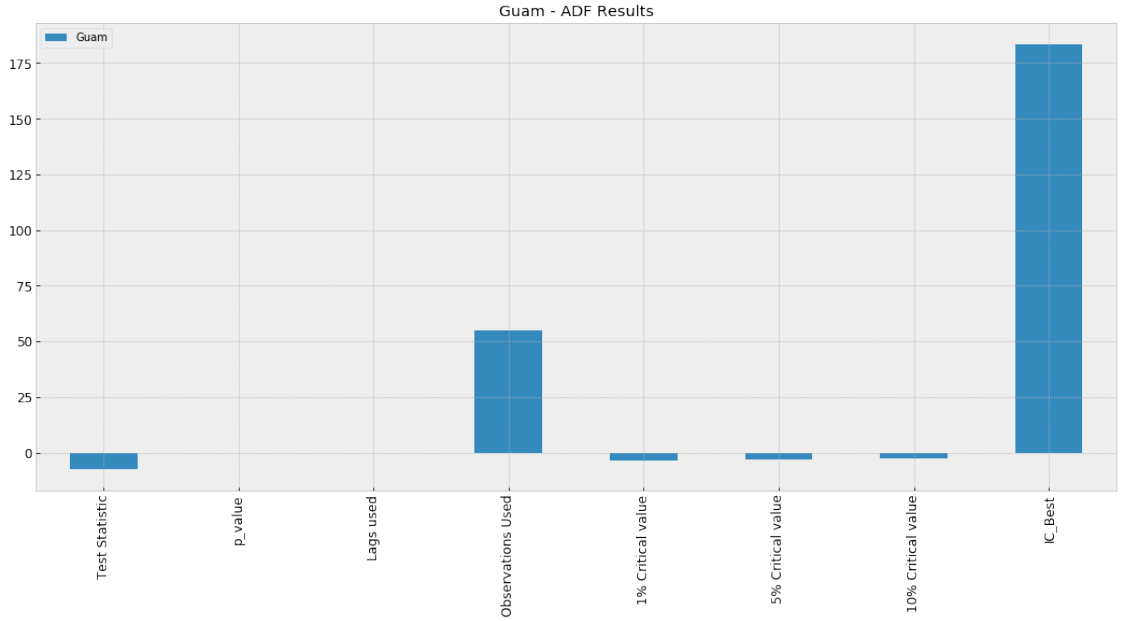

Gibraltar - ADF Results
Gibraltar
50
40
30
20
10
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

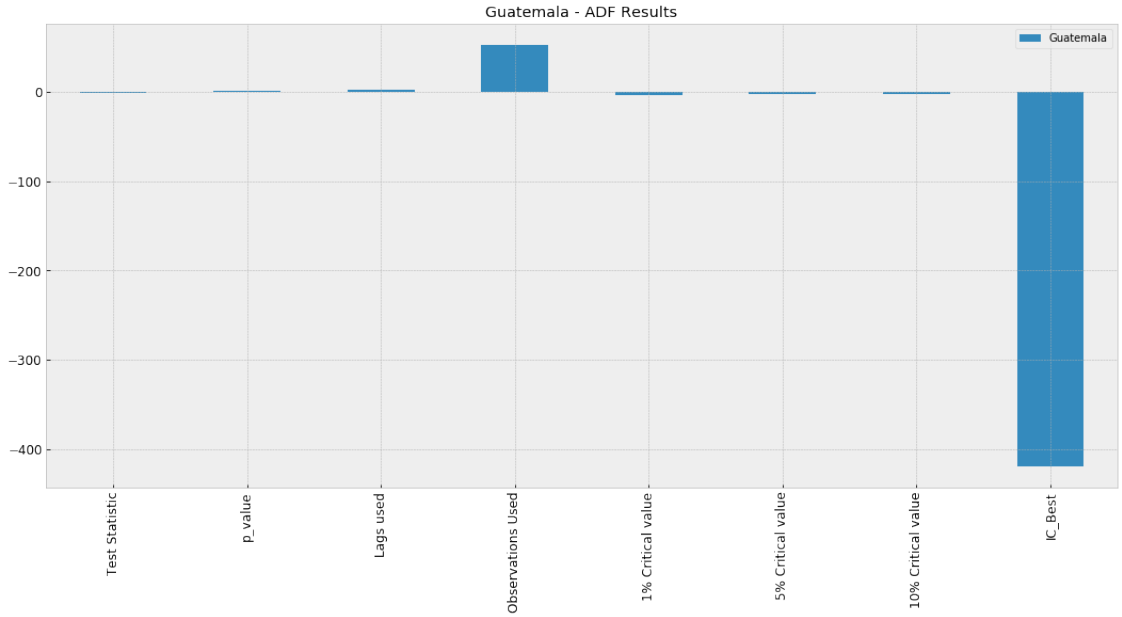

Greece - ADF Results
Greece
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

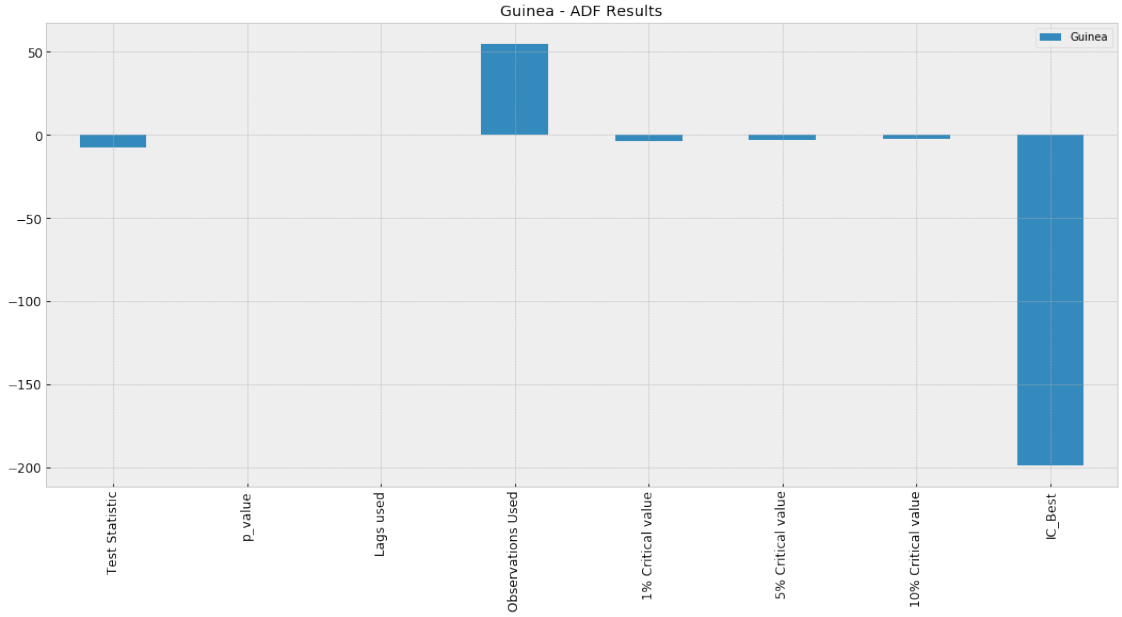

Greenland - ADF Results
Greenland
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

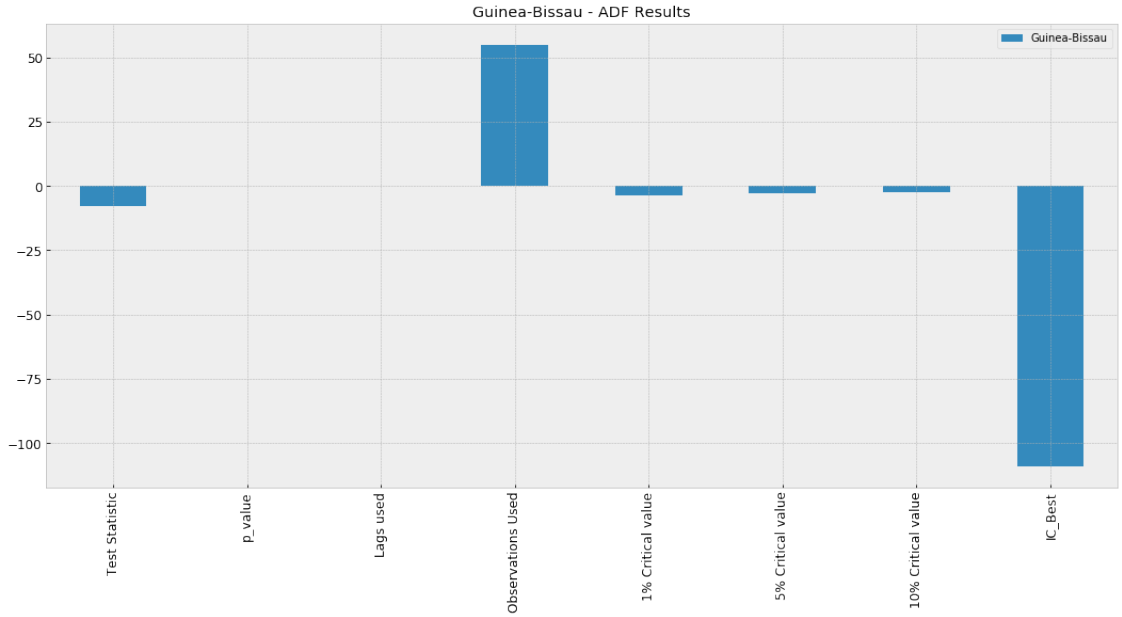

Grenada - ADF Results
Grenada
60
40
20
0
−20
−40
−60
−80
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

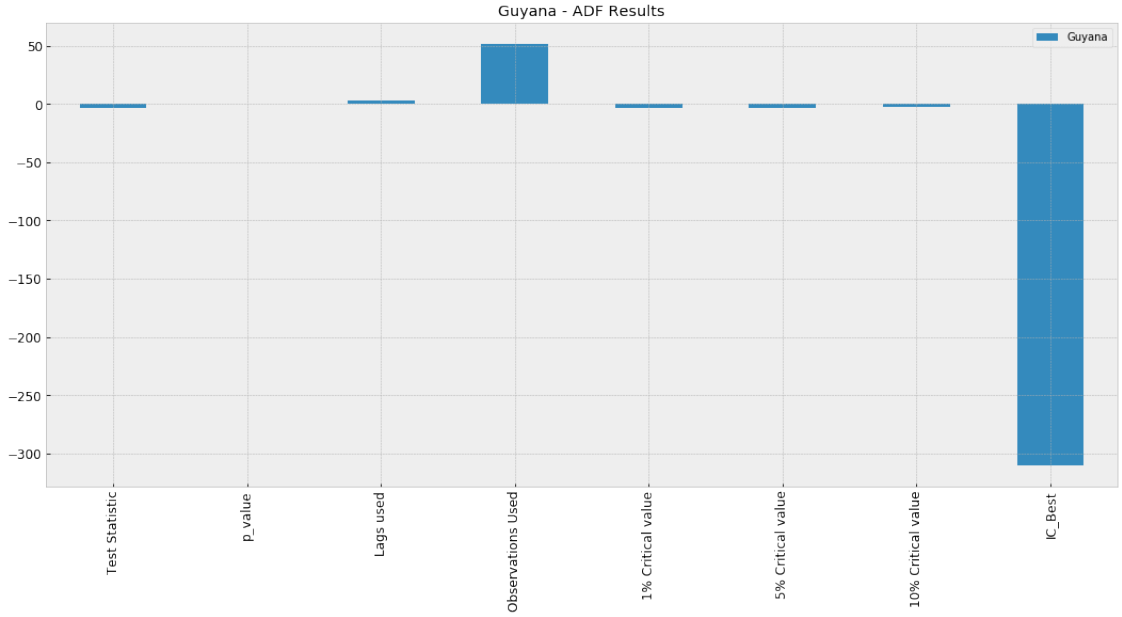

Guadeloupe - ADF Results
Guadeloupe
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

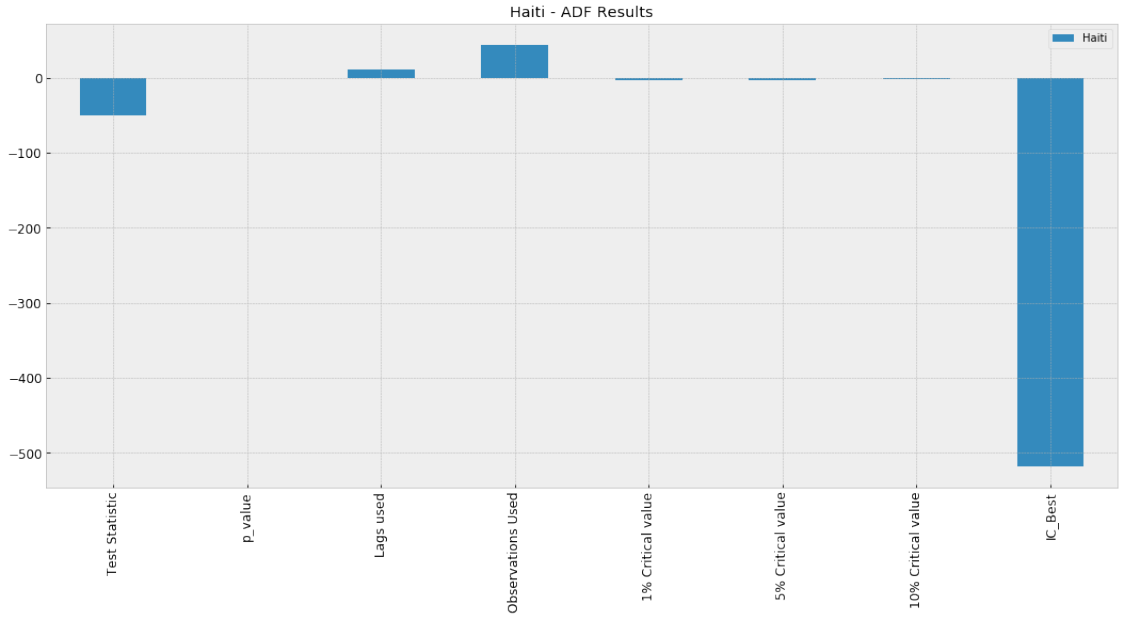

Guam - ADF Results
Guam
175
150
125
100
75
50
25
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

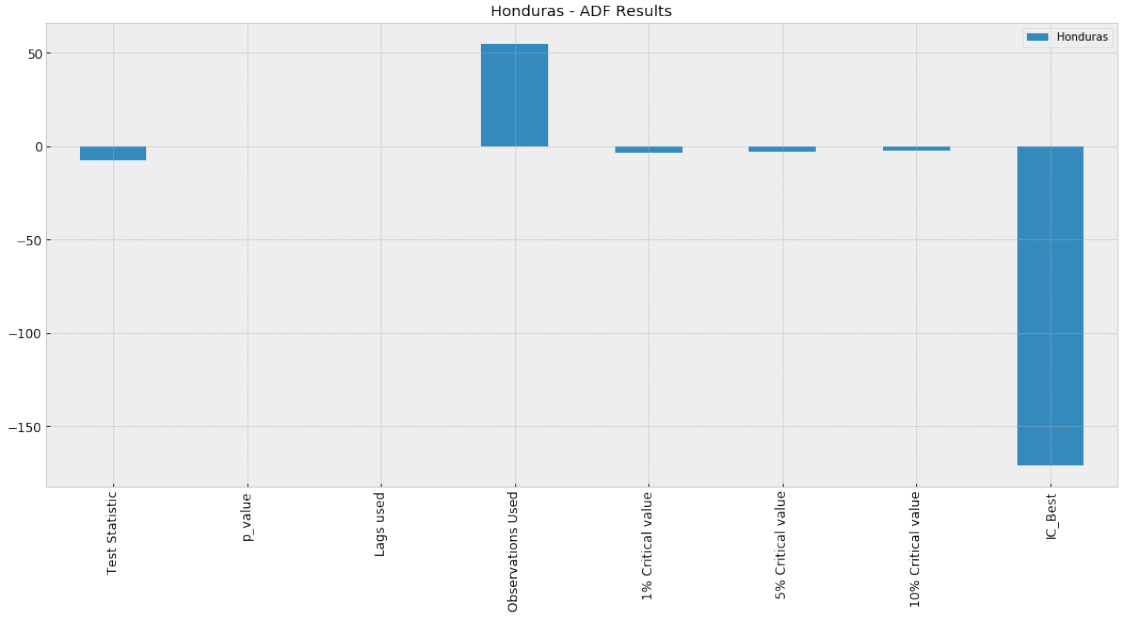

Guatemala - ADF Results
Guatemala
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

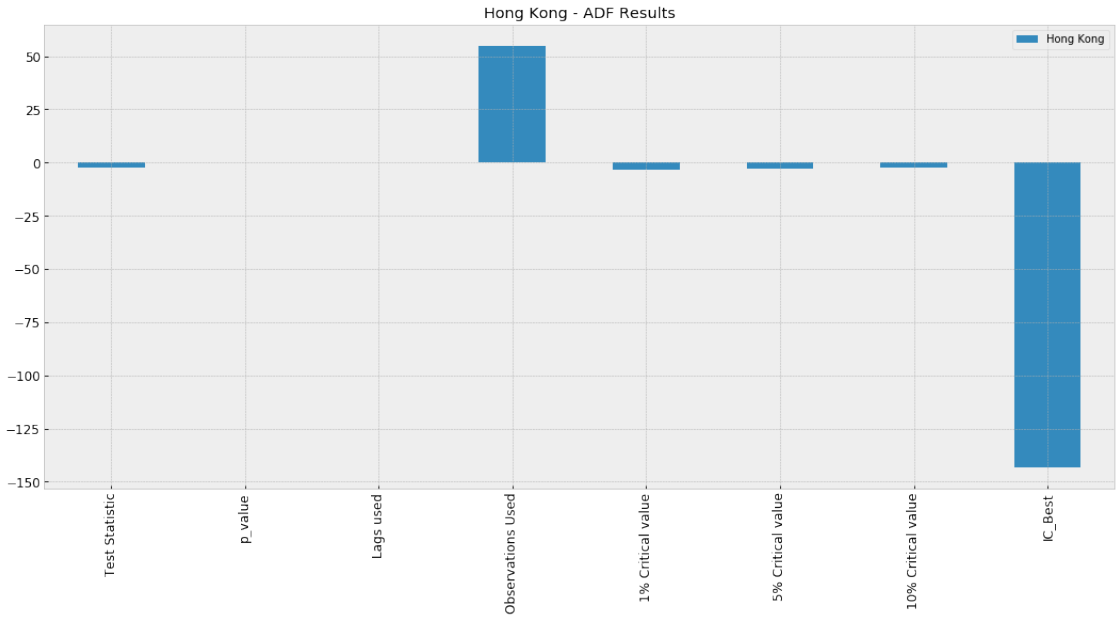

Guinea - ADF Results
Guinea
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

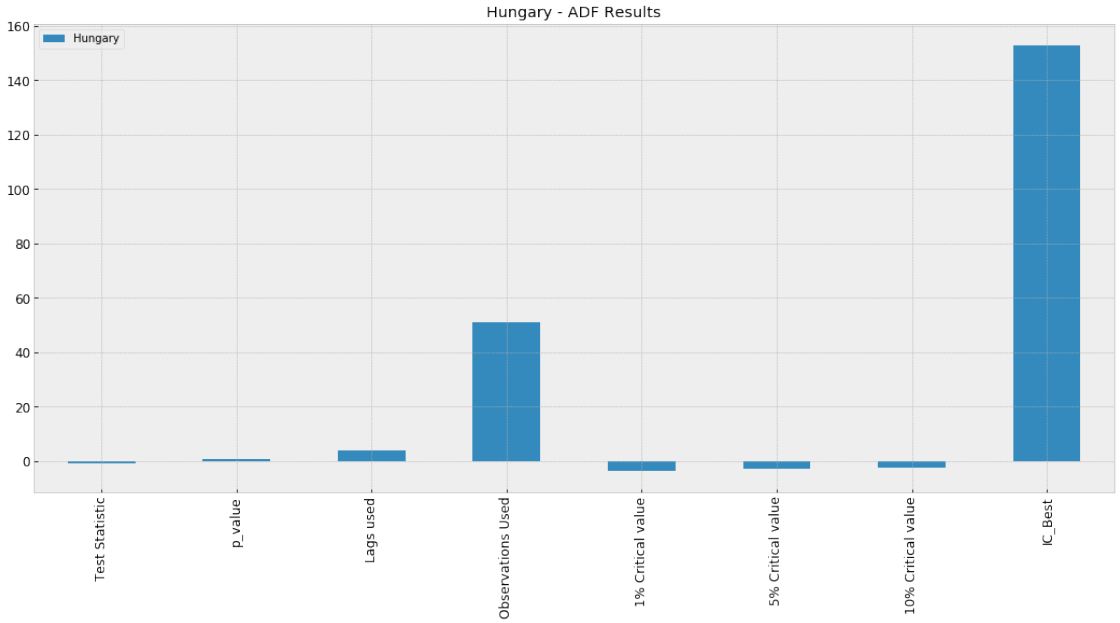

Guinea-Bissau - ADF Results
Guinea-Bissau
50
25
0
−25
−50
−75
−100
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

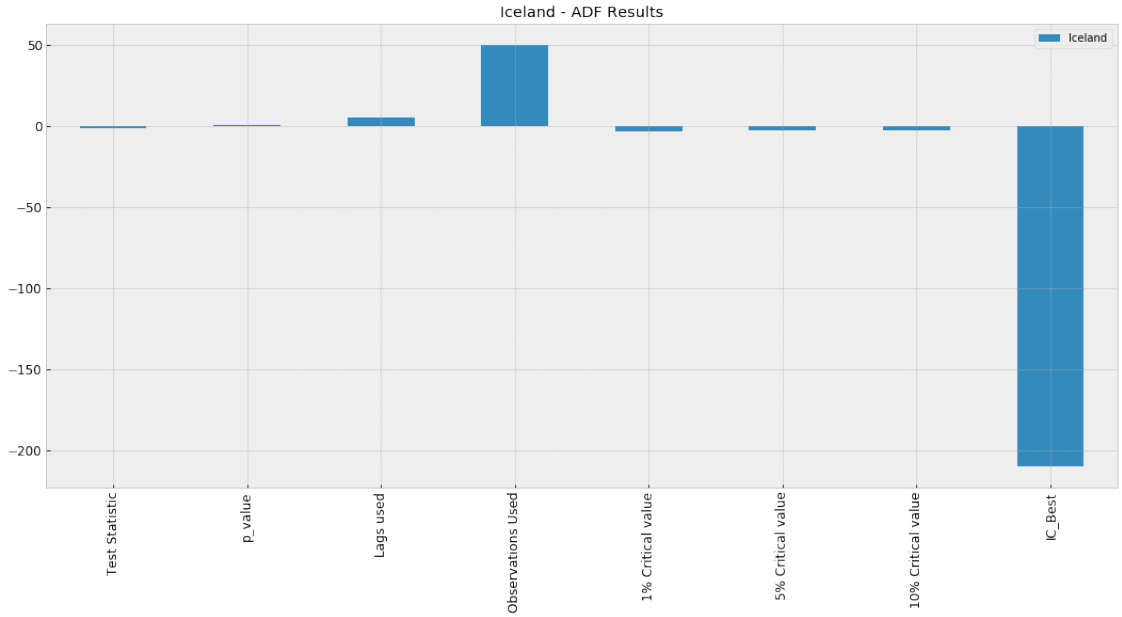

Guyana - ADF Results
Guyana
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

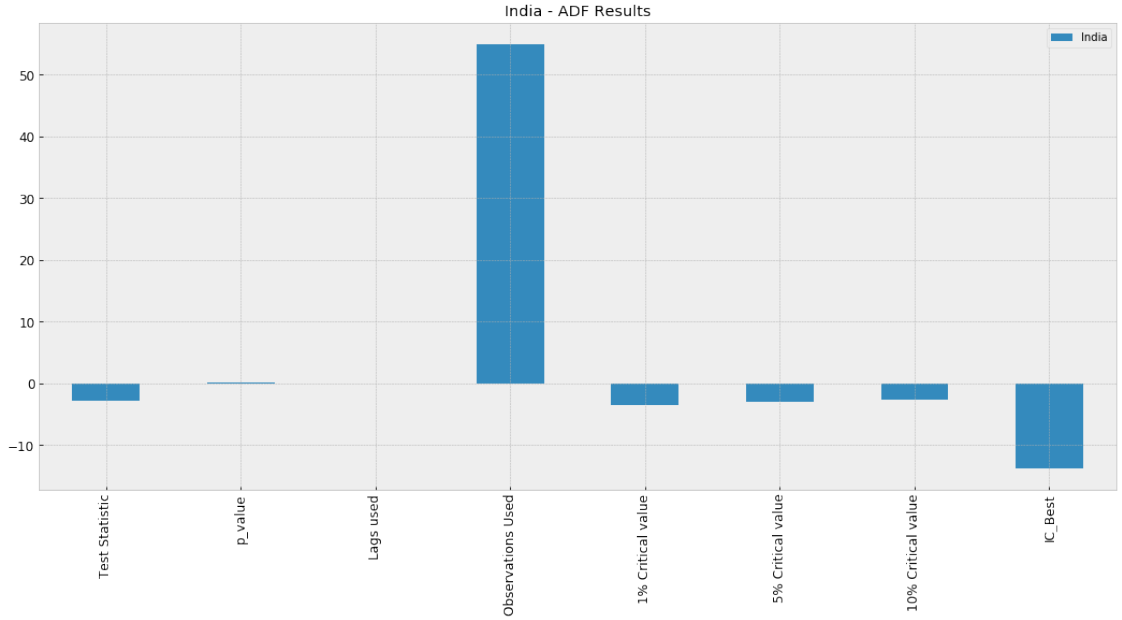

Haiti - ADF Results
Haiti
0
−100
−200
−300
−400
−500
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

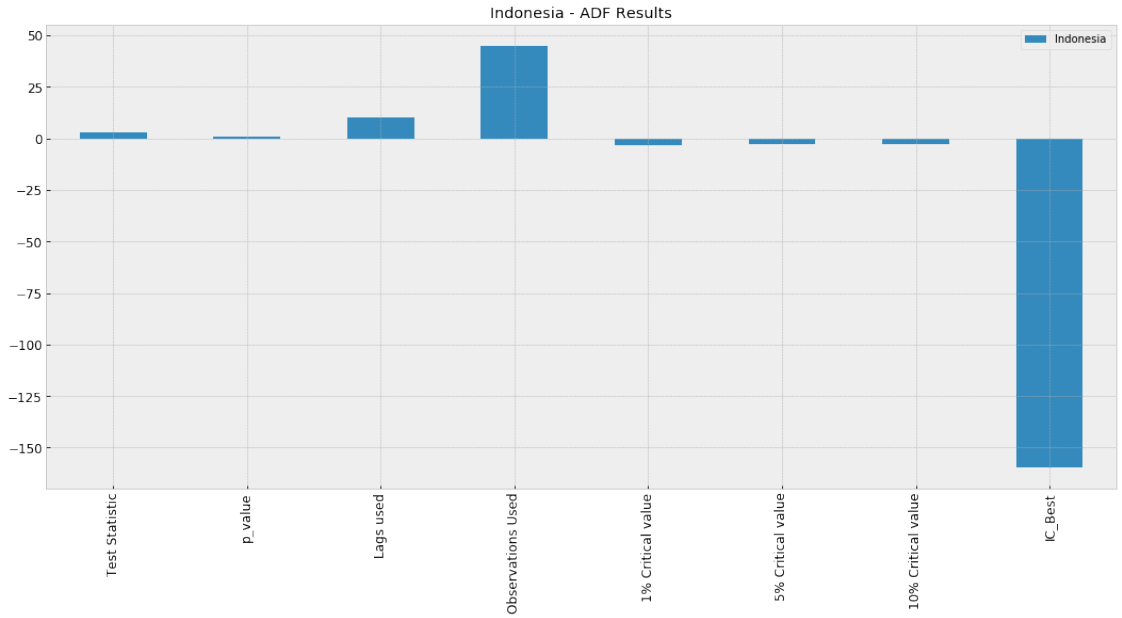

Honduras - ADF Results
Honduras
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

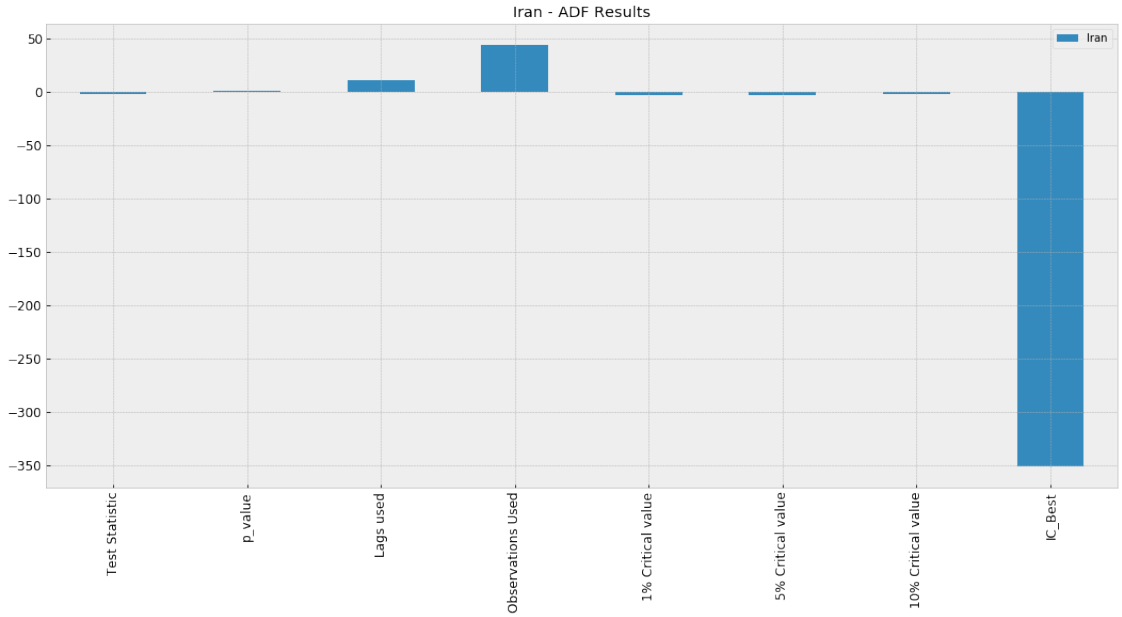

Hong Kong - ADF Results
Hong Kong
50
25
0
−25
−50
−75
−100
−125
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

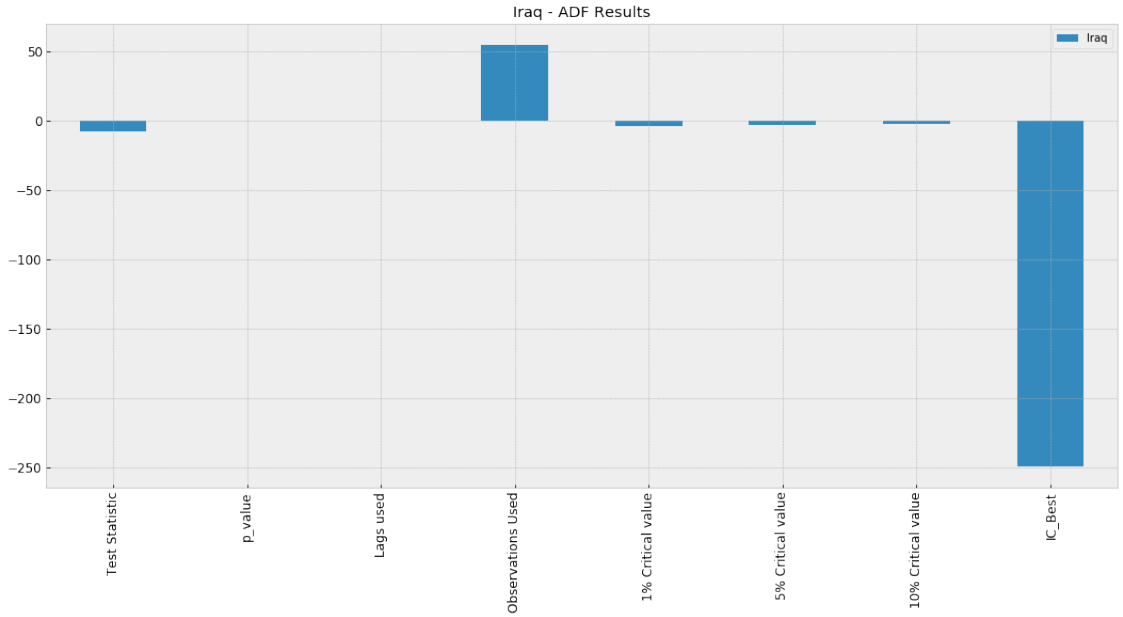

Hungary - ADF Results
Hungary
160
140
120
100
80
60
40
20
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

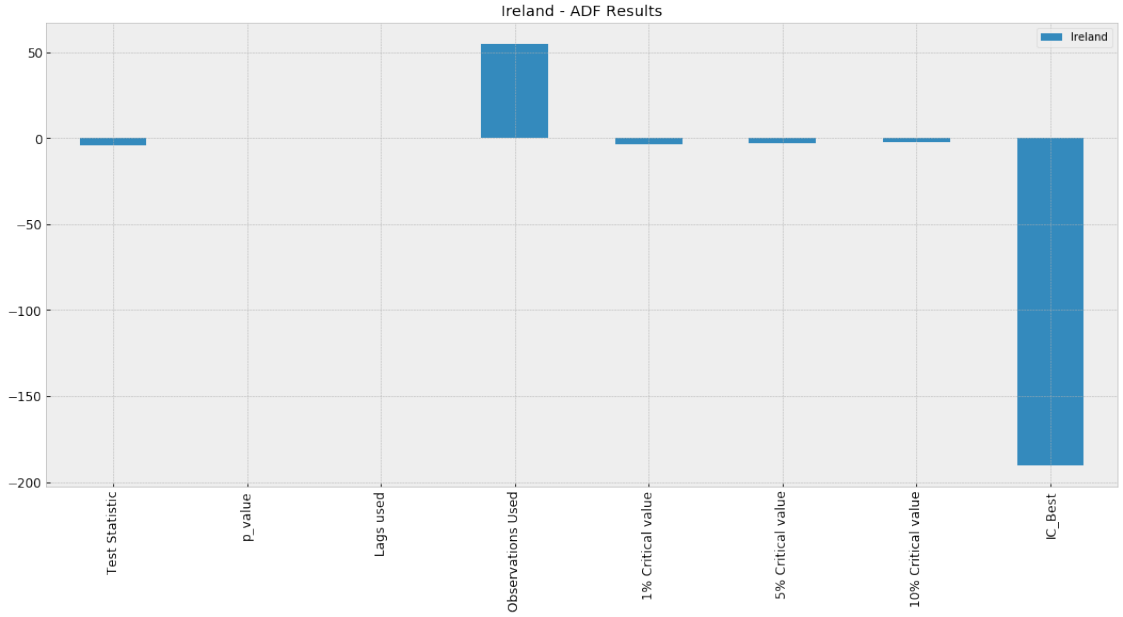

Iceland - ADF Results
Iceland
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

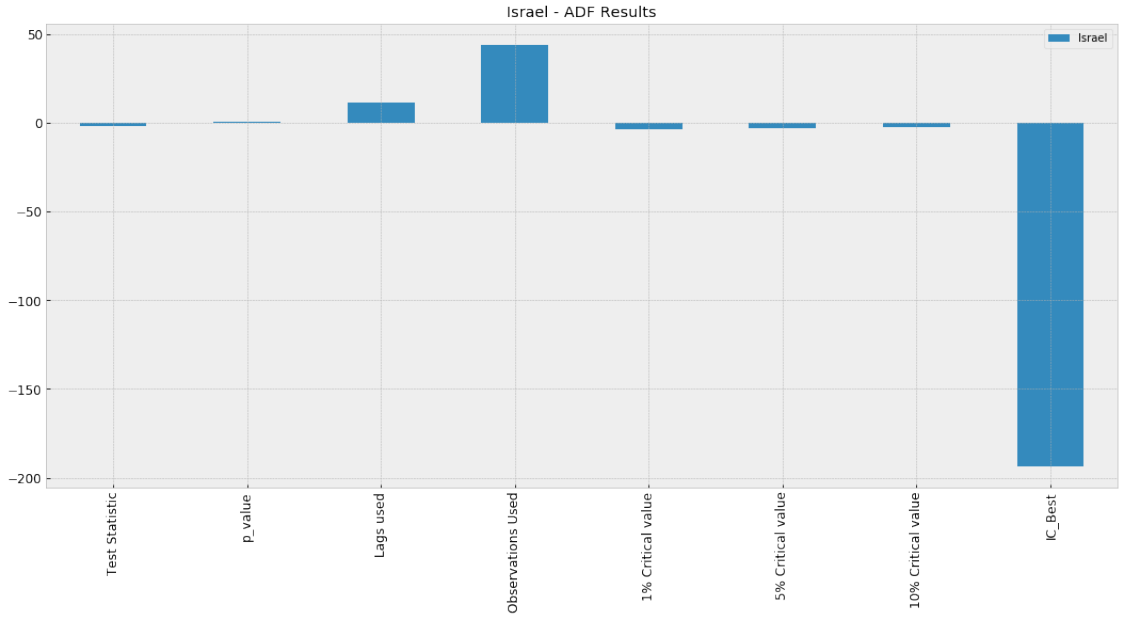

India - ADF Results
India
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

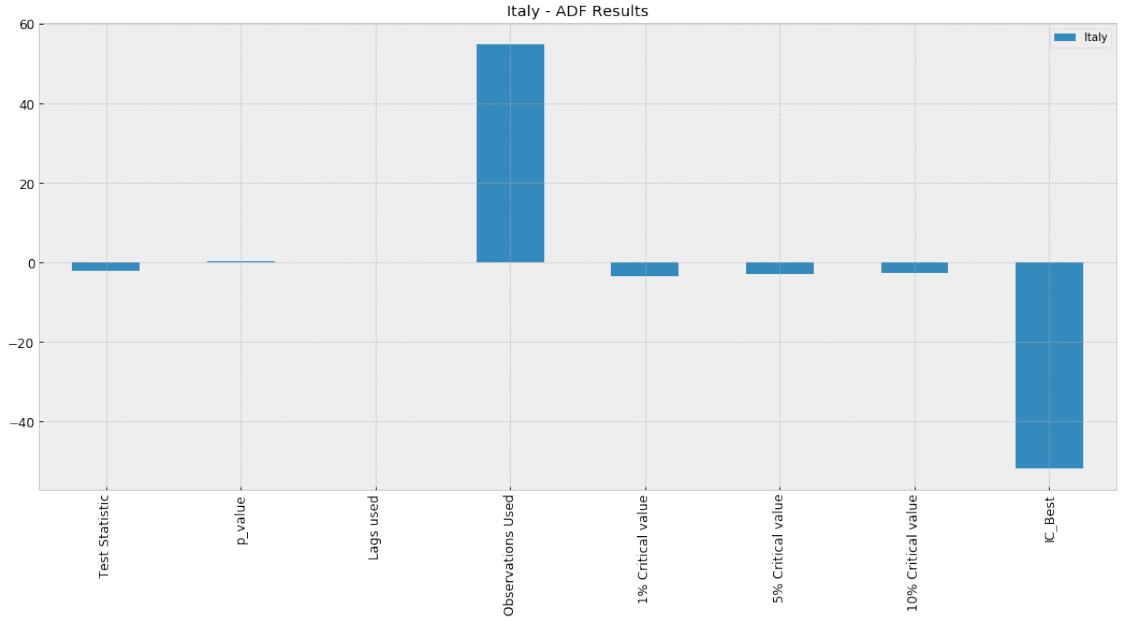

Indonesia - ADF Results
Indonesia
50
25
0
−25
−50
−75
−100
−125
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

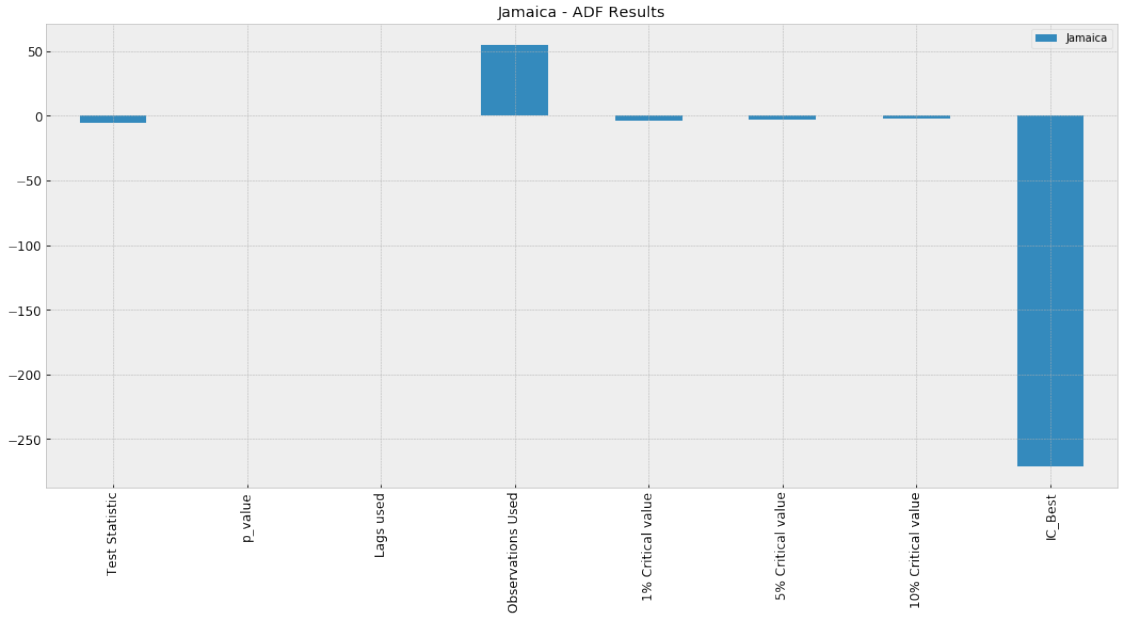

Iran - ADF Results
Iran
50
0
−50
−100
−150
−200
−250
−300
−350
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

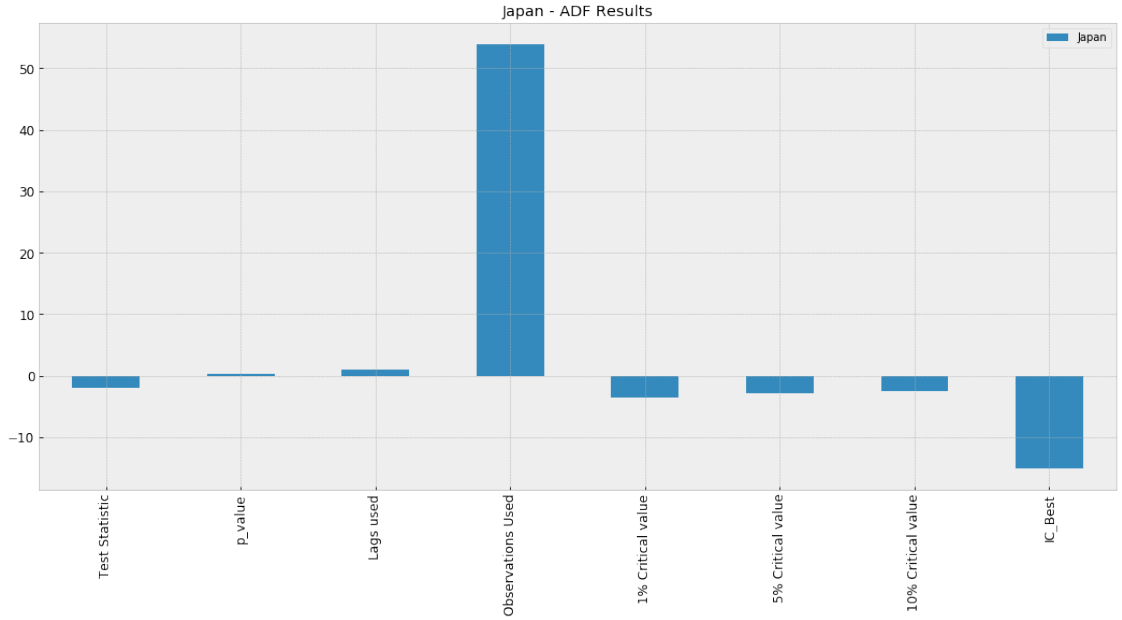

Iraq - ADF Results
Iraq
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

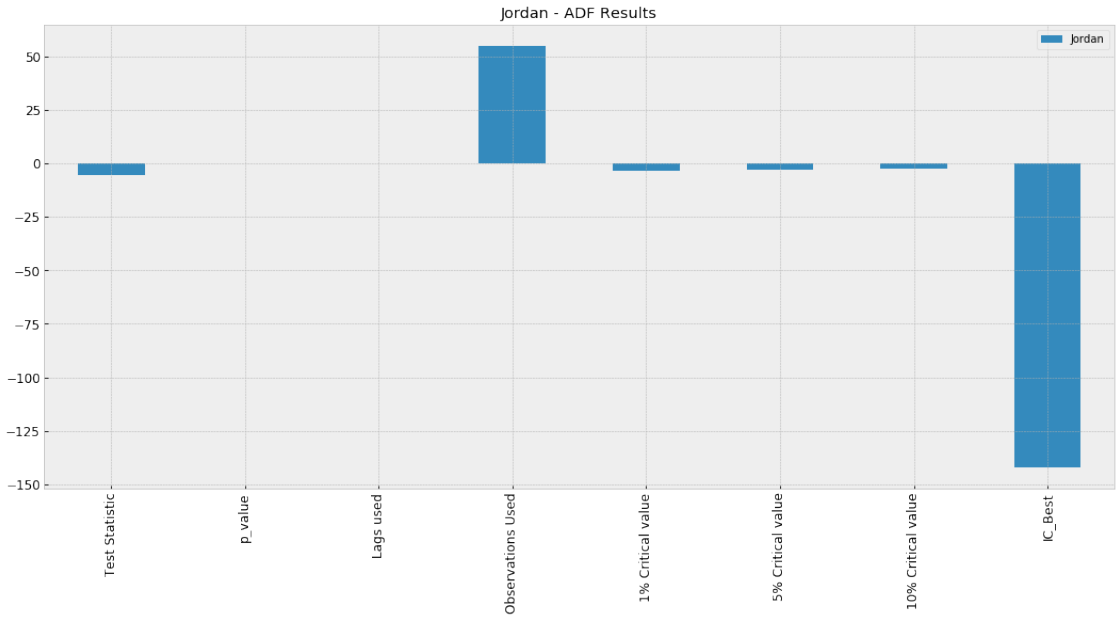

Ireland - ADF Results
Ireland
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

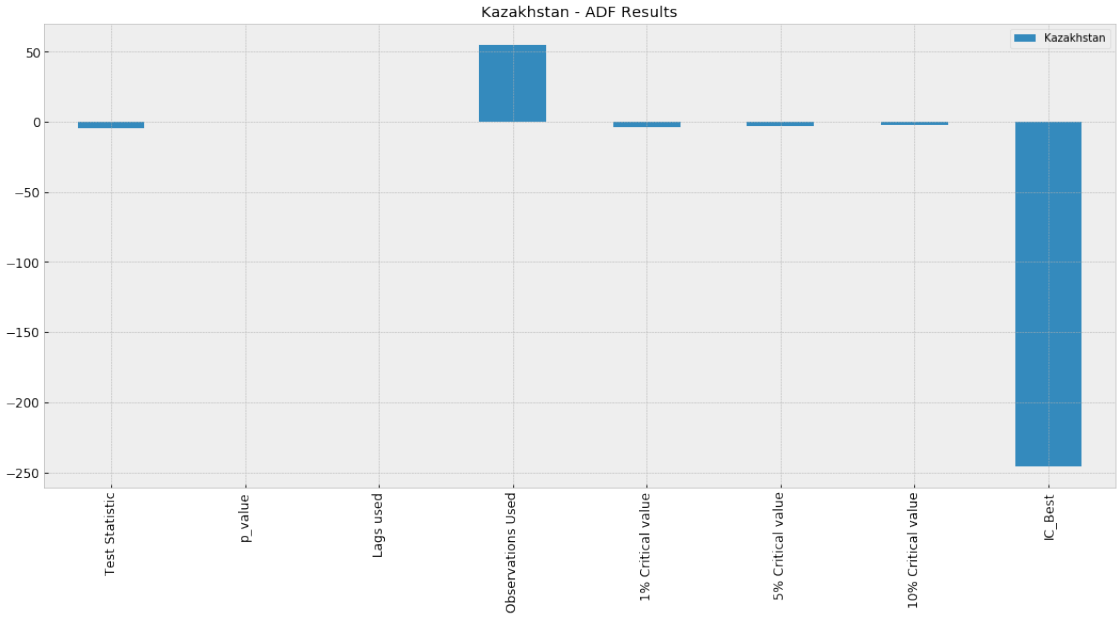

Israel - ADF Results
Israel
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

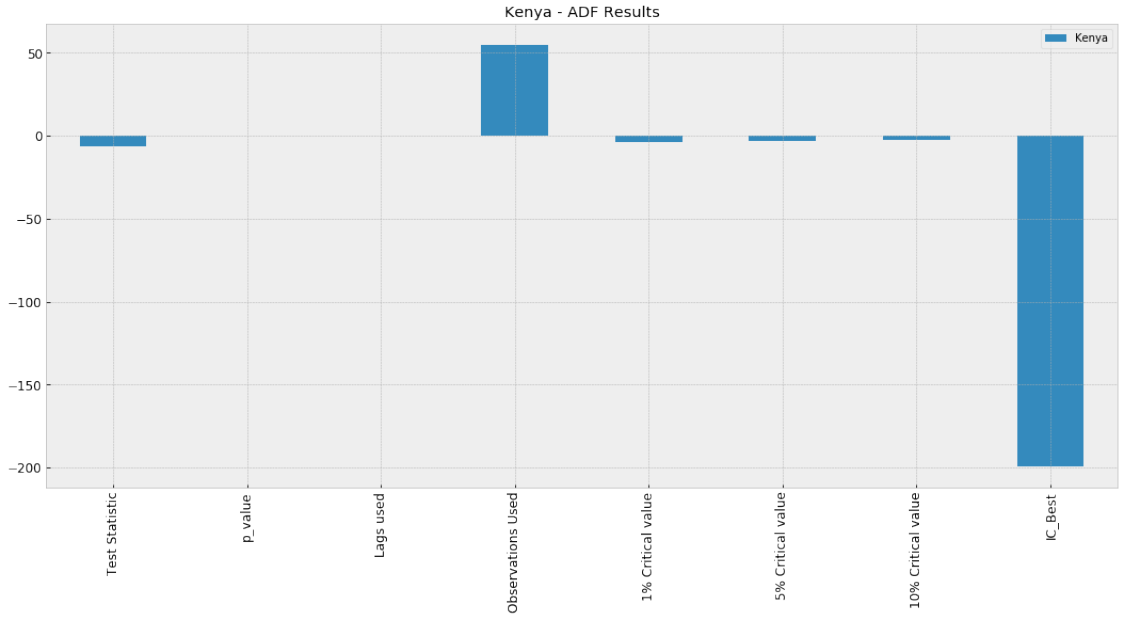

Italy - ADF Results
Italy
60
40
20
0
−20
−40
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

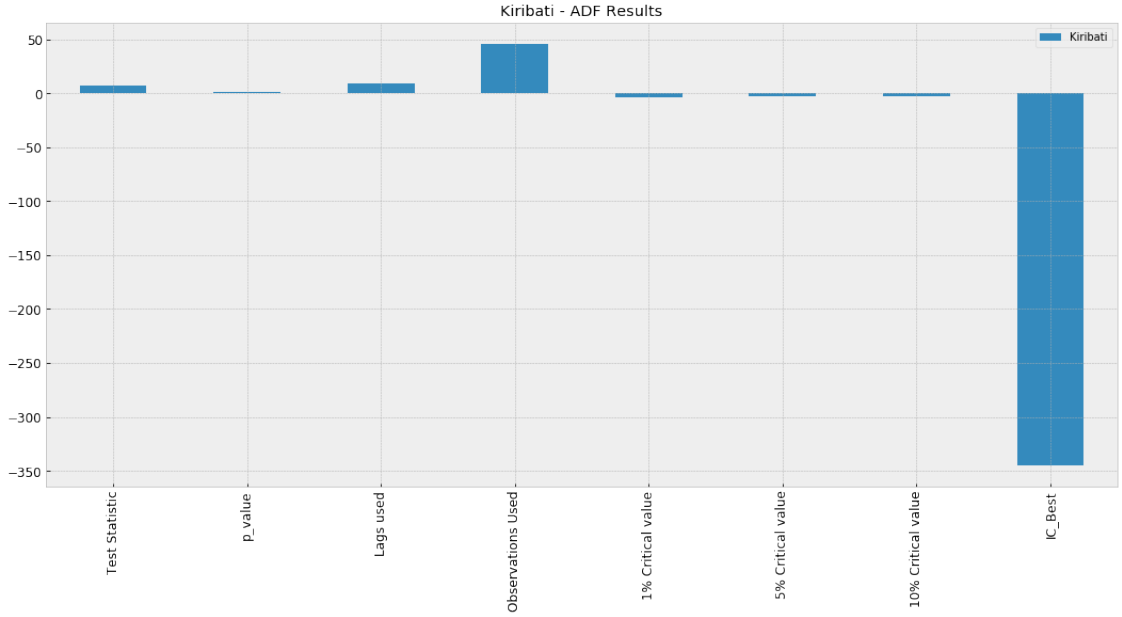

Jamaica - ADF Results
Jamaica
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

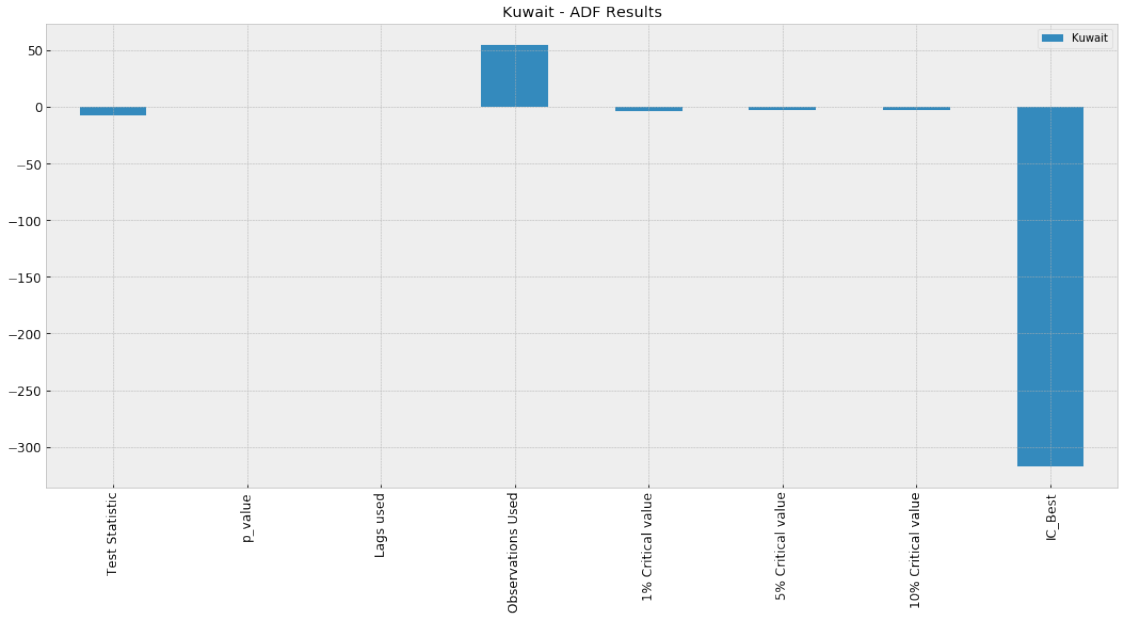

Japan - ADF Results
Japan
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

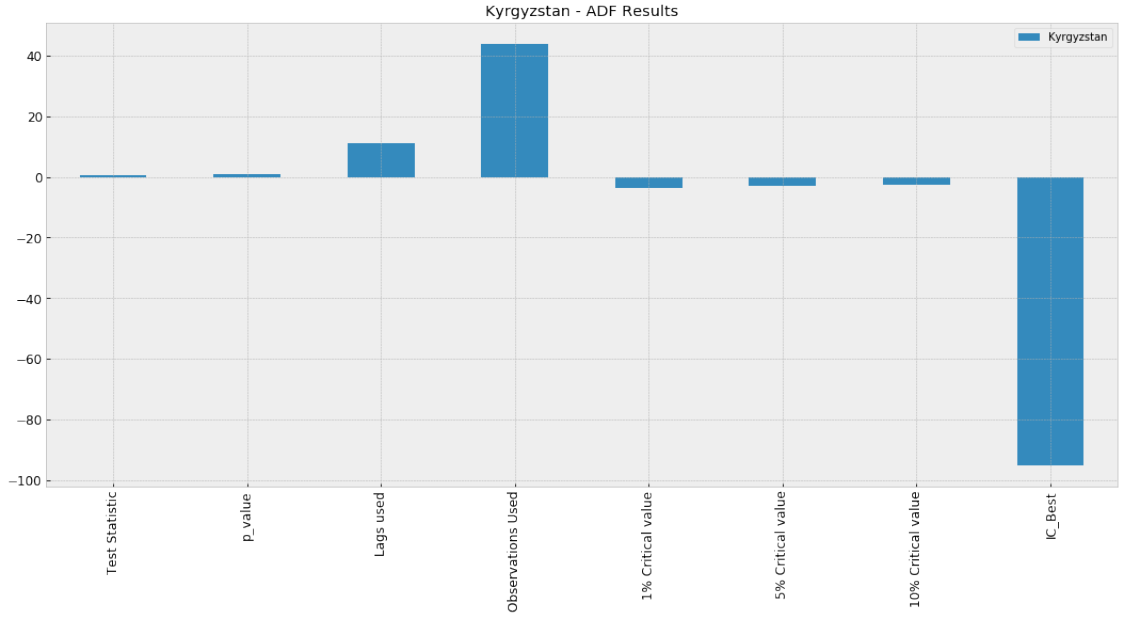

Jordan - ADF Results
Jordan
50
25
0
−25
−50
−75
−100
−125
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

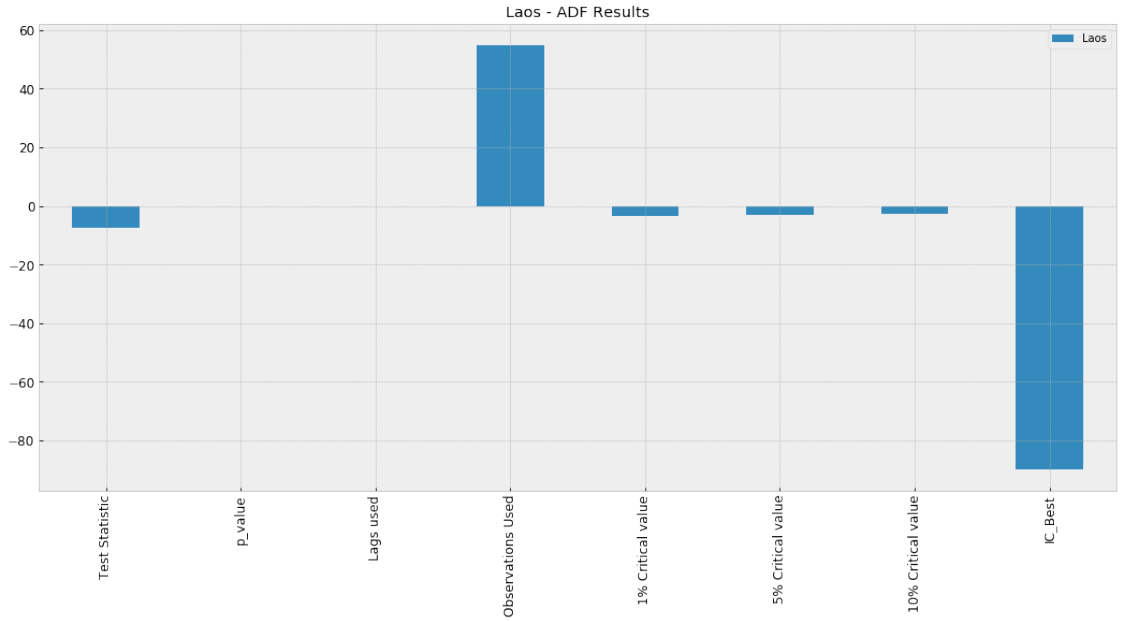

Kazakhstan - ADF Results
Kazakhstan
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

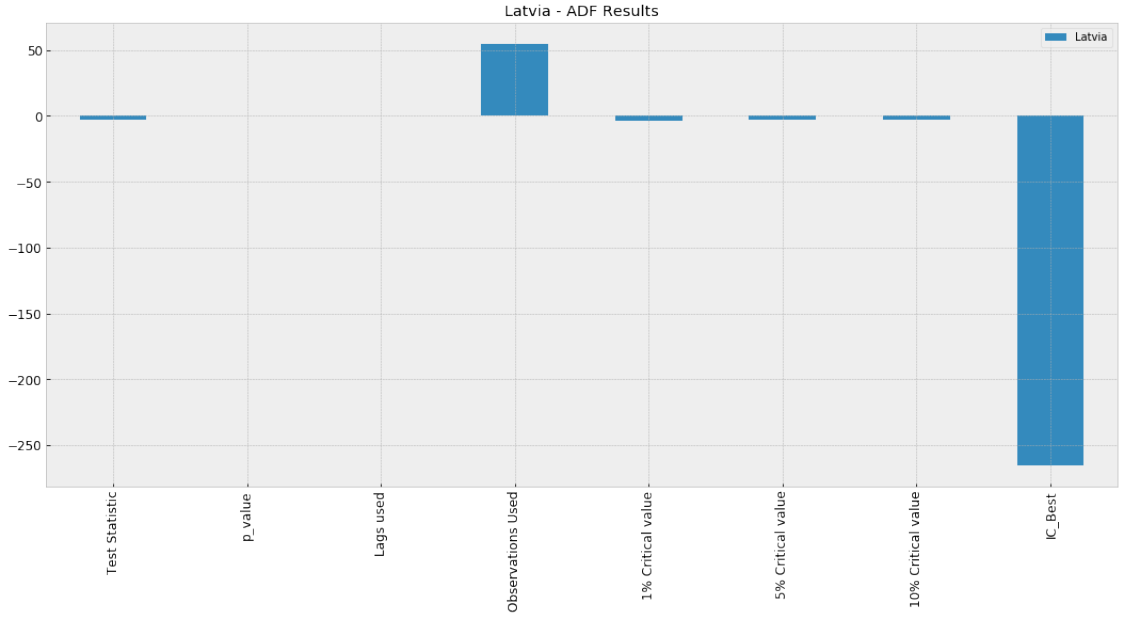

Kenya - ADF Results
Kenya
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

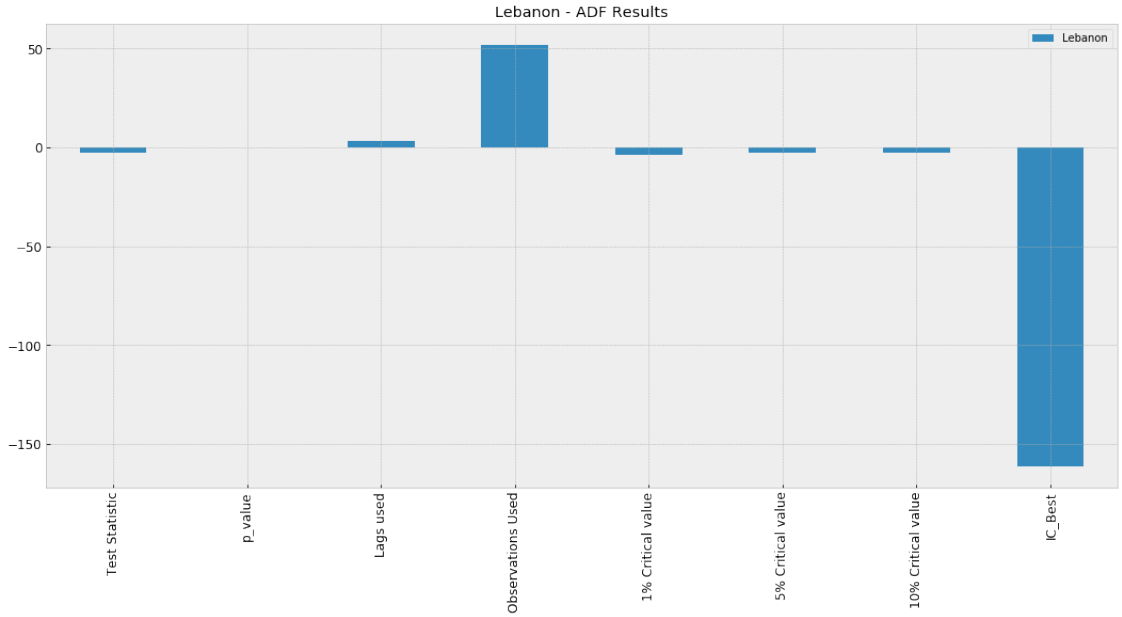

Kiribati - ADF Results
Kiribati
50
0
−50
−100
−150
−200
−250
−300
−350
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

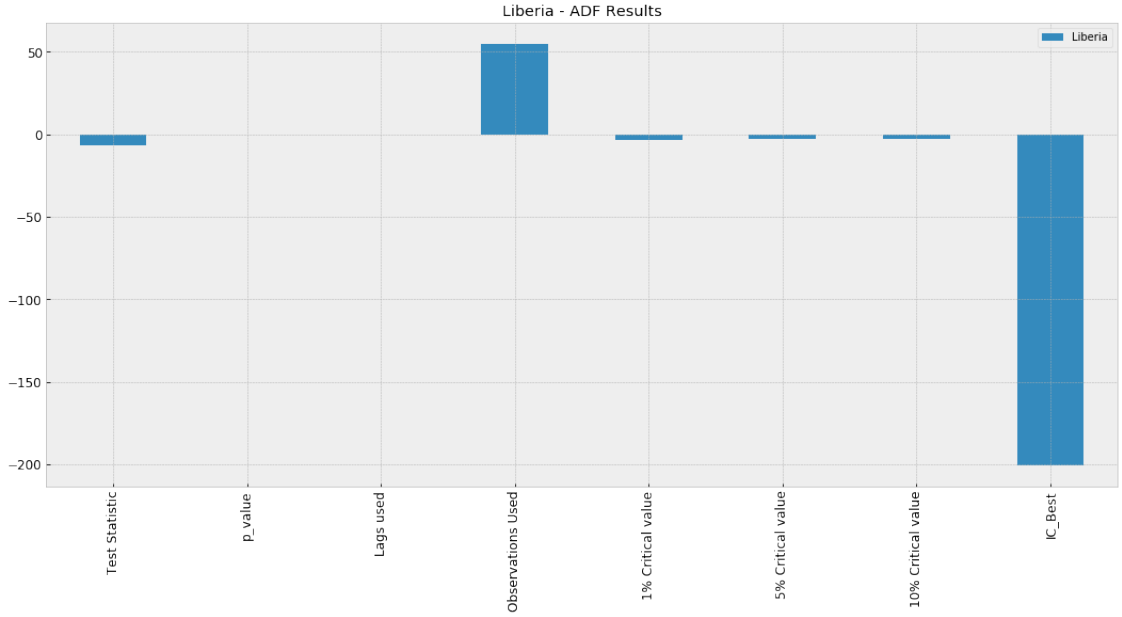

Kuwait - ADF Results
Kuwait
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

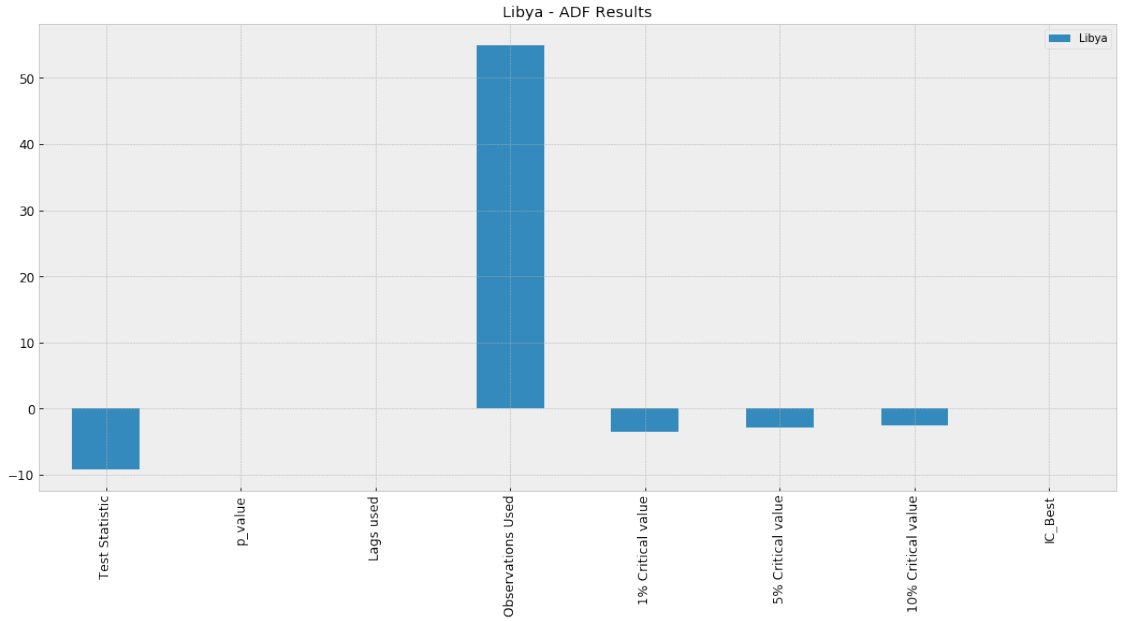

Kyrgyzstan - ADF Results
Kyrgyzstan
40
20
0
−20
−40
−60
−80
−100
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

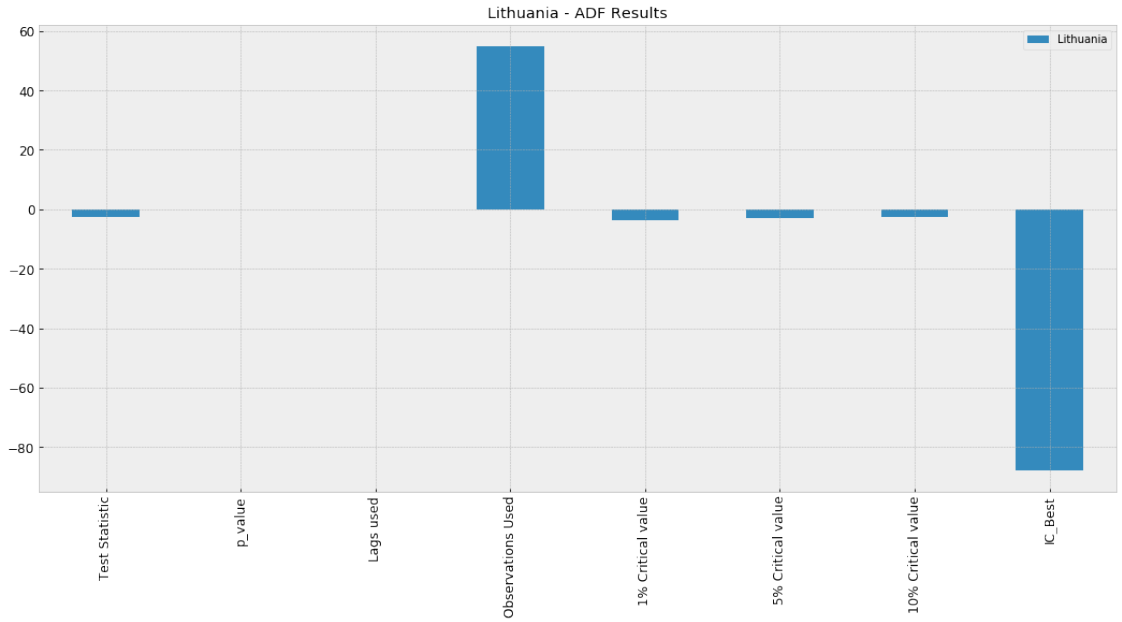

Laos - ADF Results
Laos
60
40
20
0
−20
−40
−60
−80
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

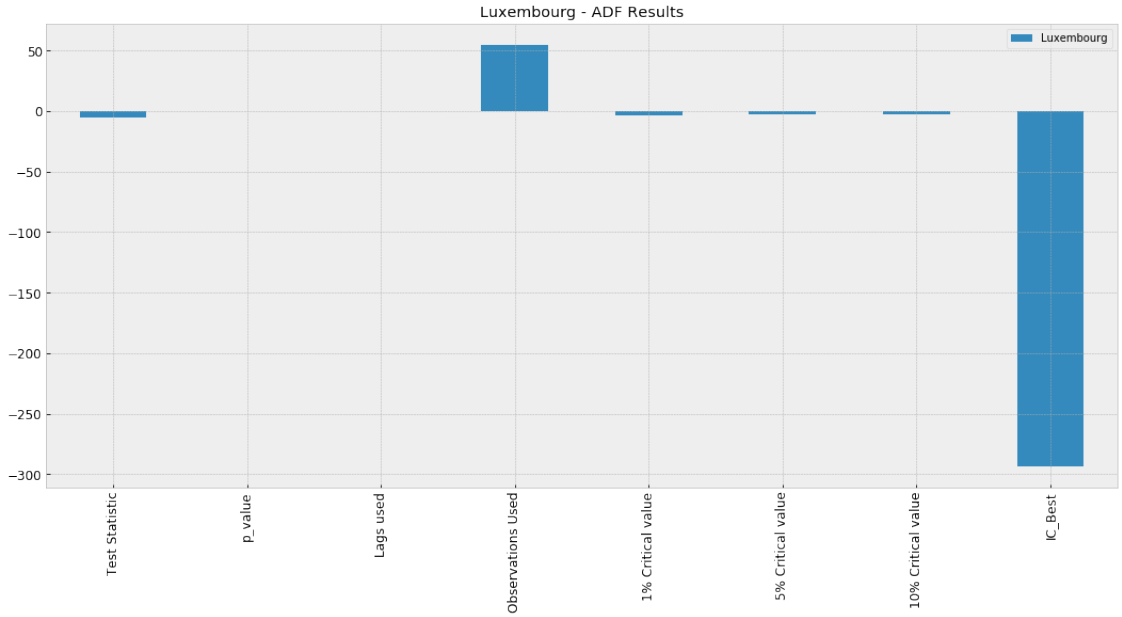

Latvia - ADF Results
Latvia
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

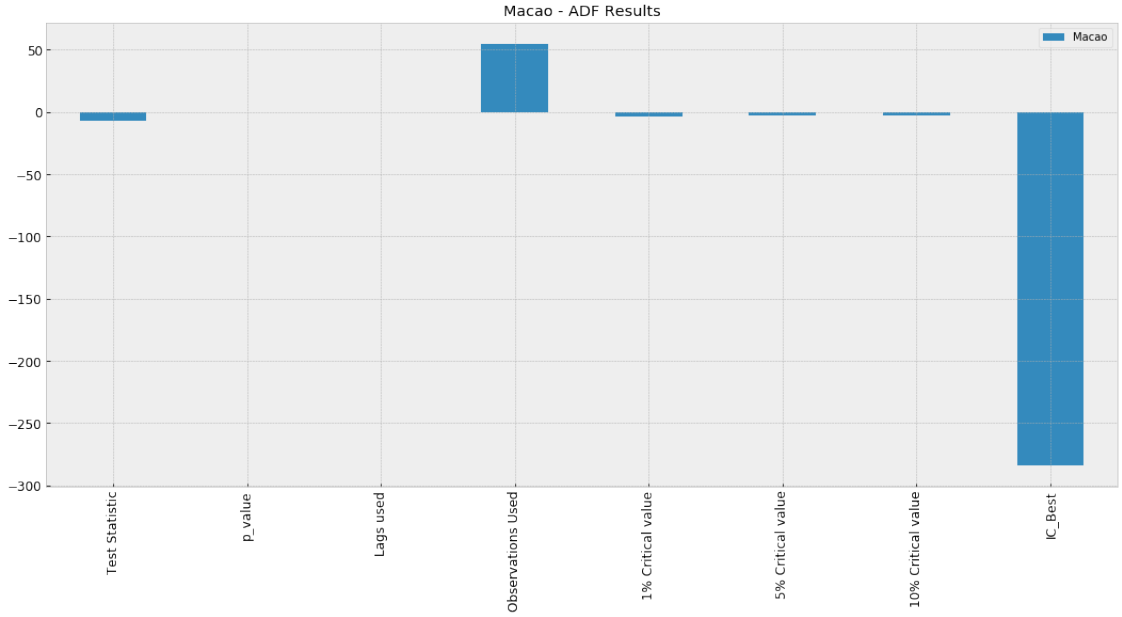

Lebanon - ADF Results
Lebanon
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

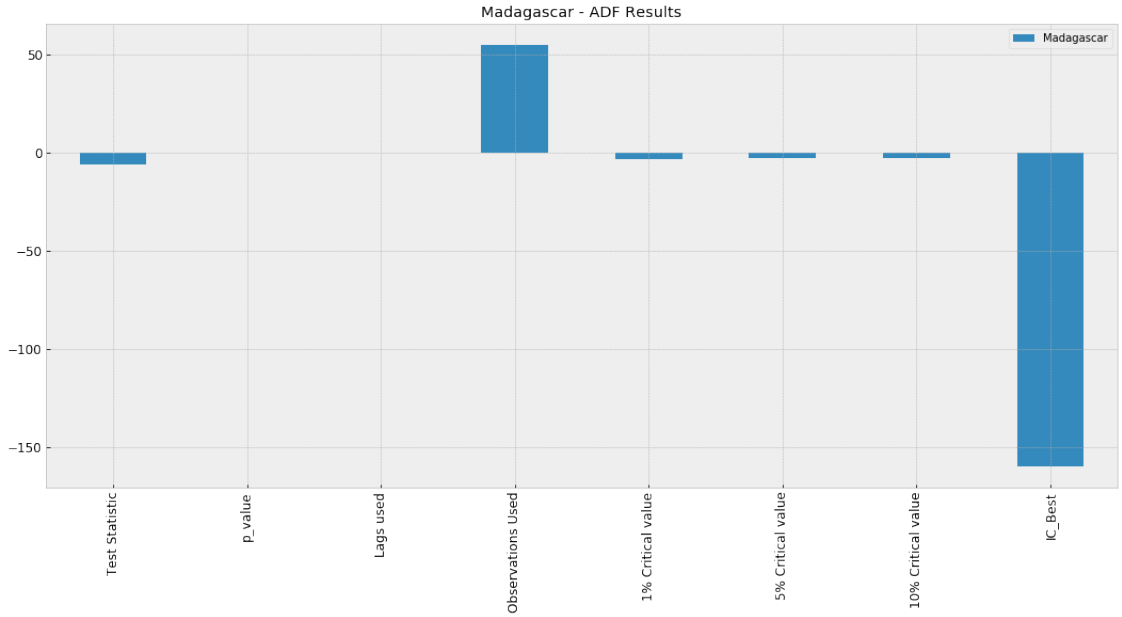

Liberia - ADF Results
Liberia
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

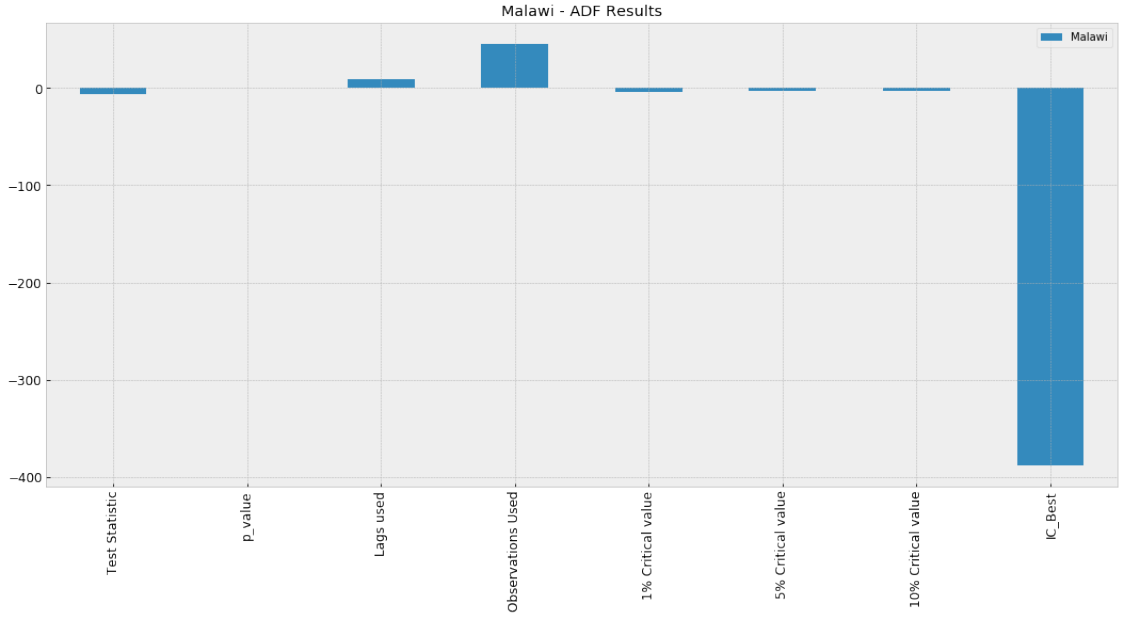

Libya - ADF Results
Libya
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

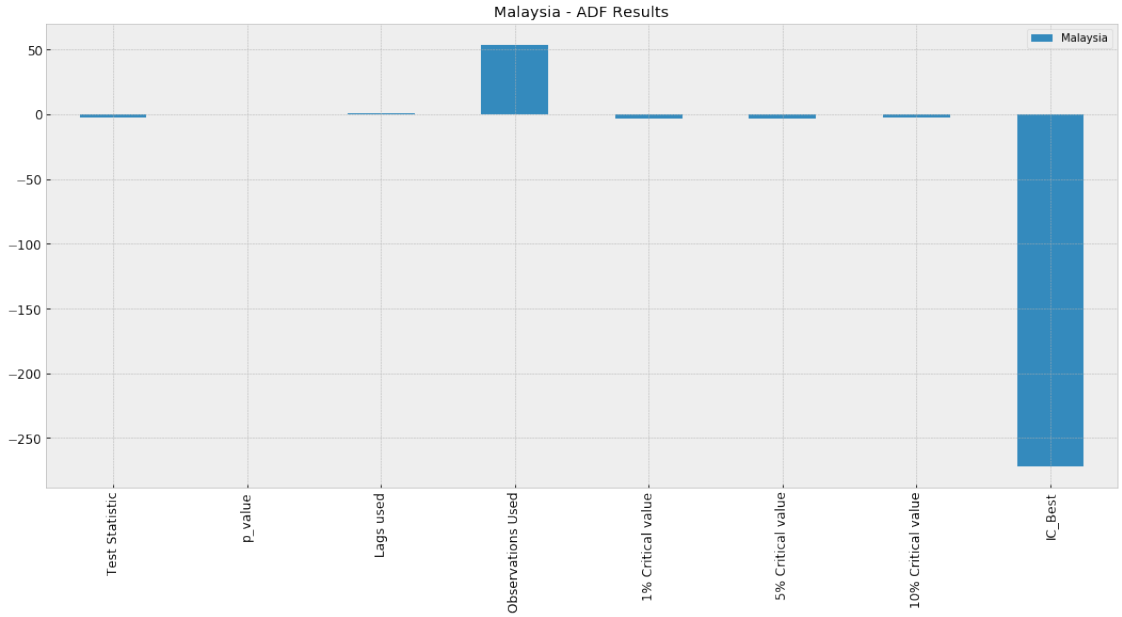

Lithuania - ADF Results
Lithuania
60
40
20
0
−20
−40
−60
−80
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

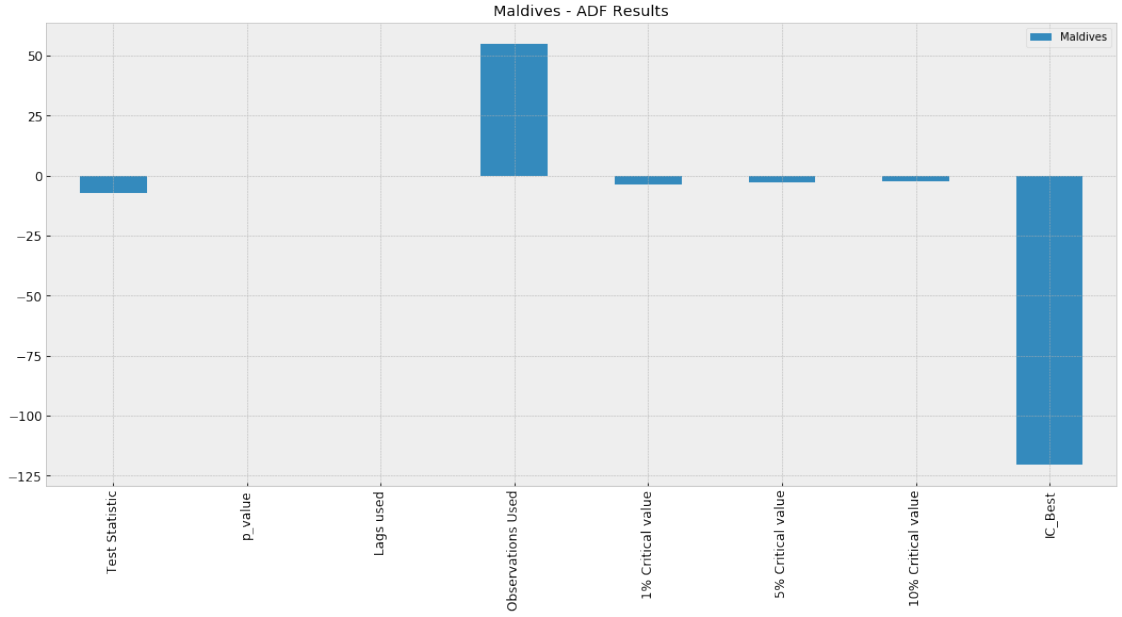

Luxembourg - ADF Results
Luxembourg
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

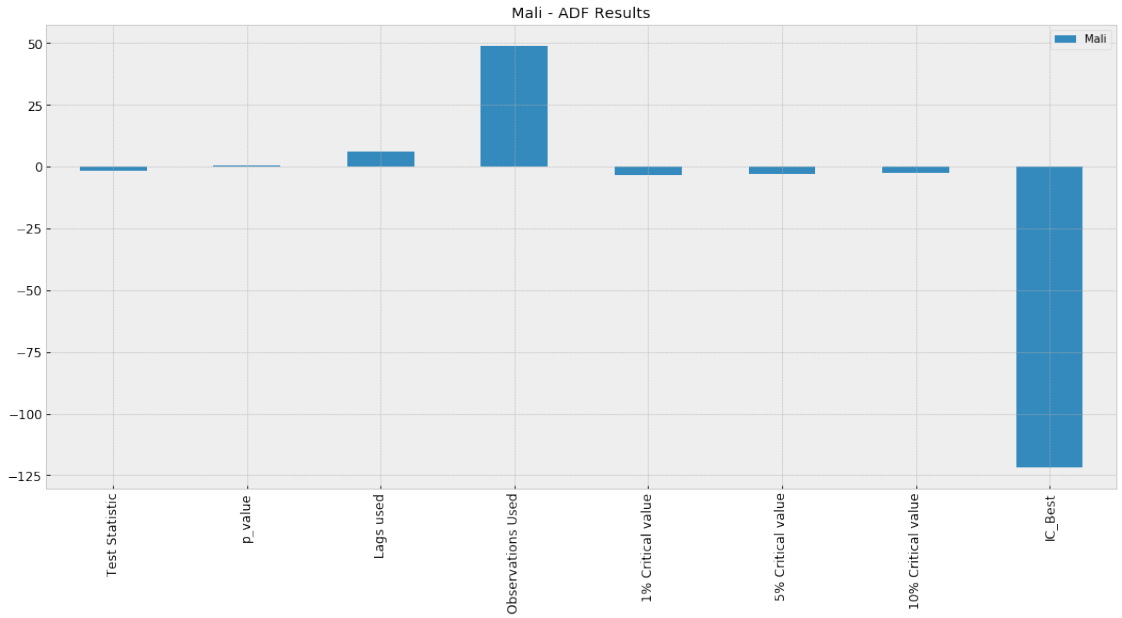

Macao - ADF Results
Macao
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

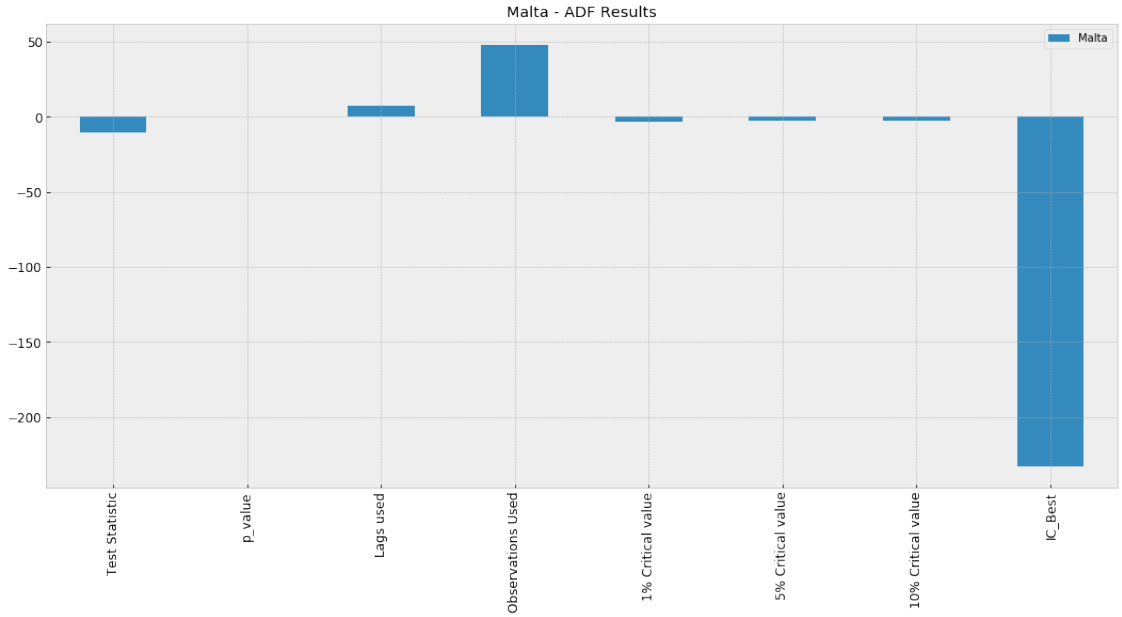

Madagascar - ADF Results
Madagascar
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

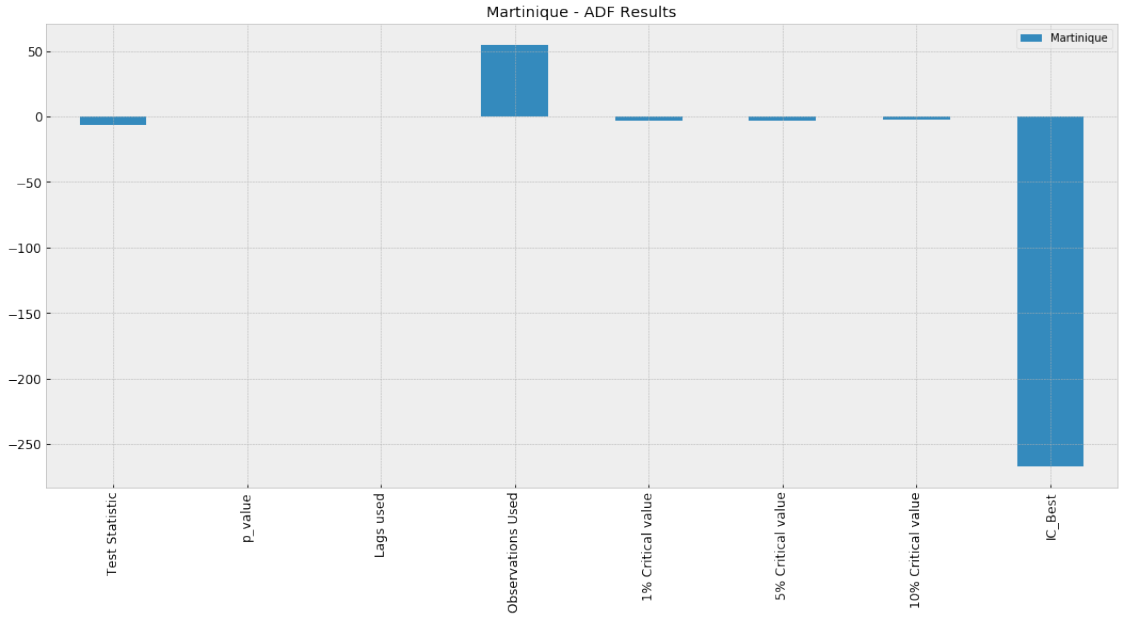

Malawi - ADF Results
Malawi
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

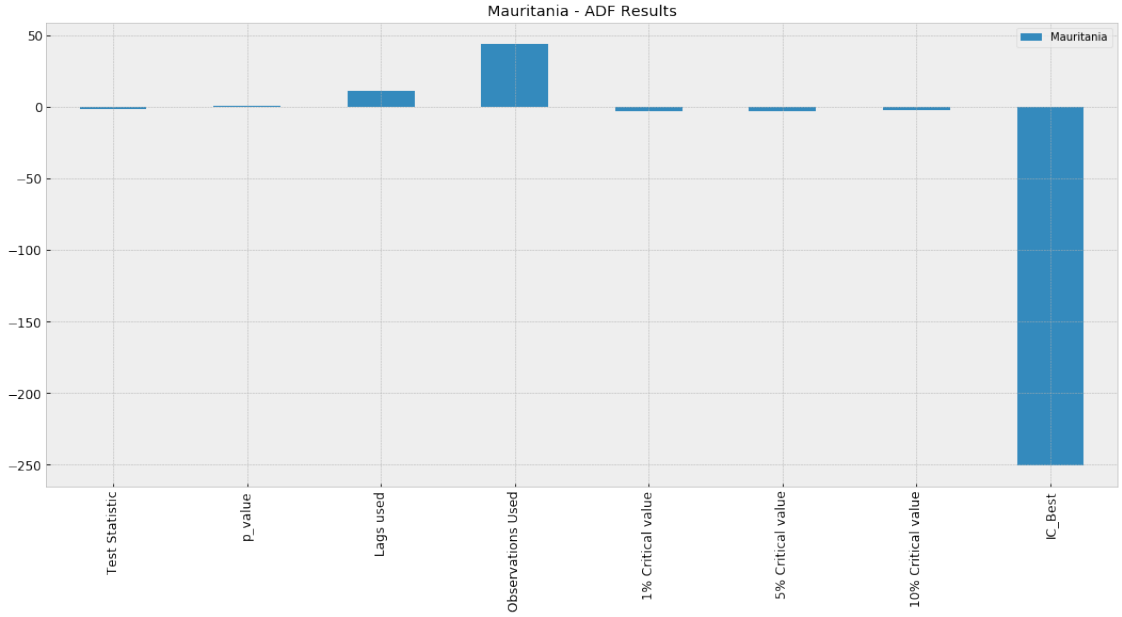

Malaysia - ADF Results
Malaysia
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

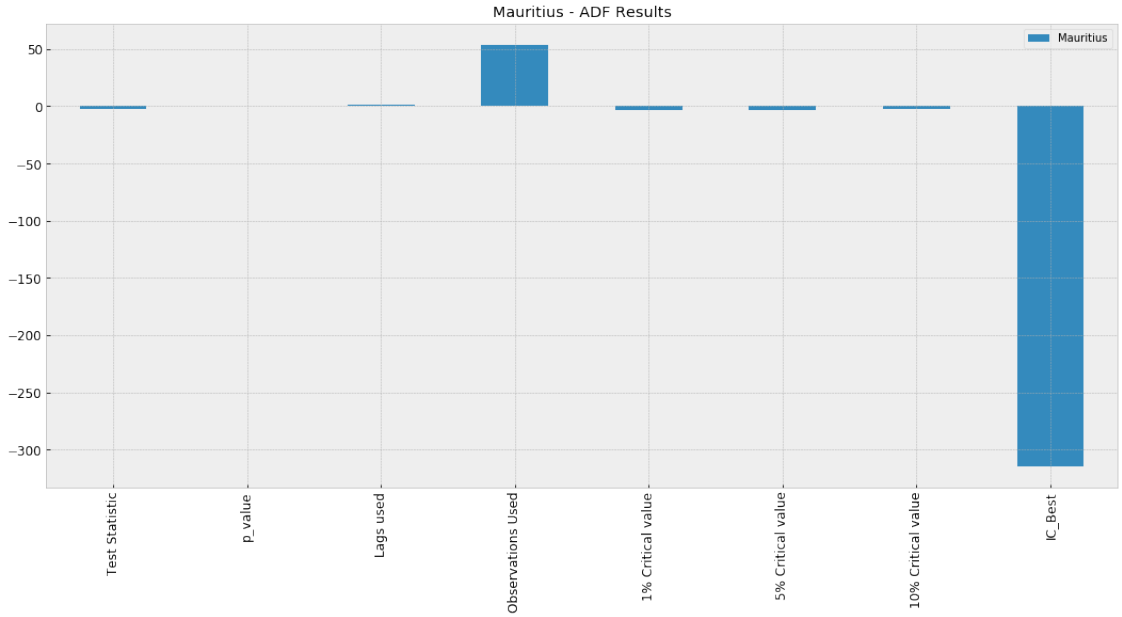

Maldives - ADF Results
Maldives
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

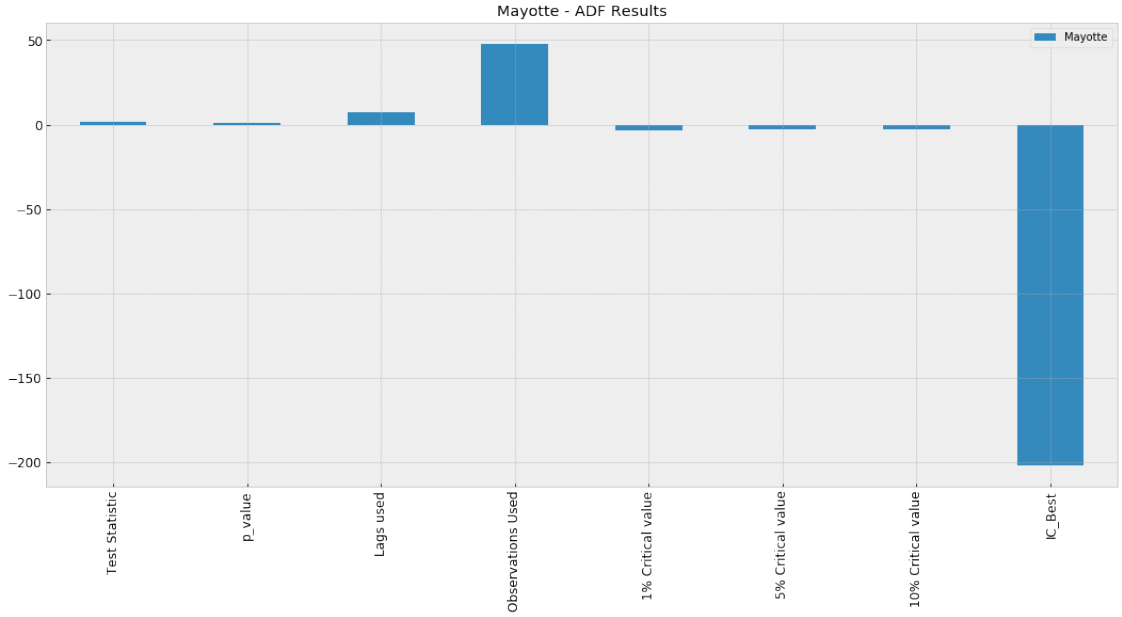

Mali - ADF Results
Mali
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

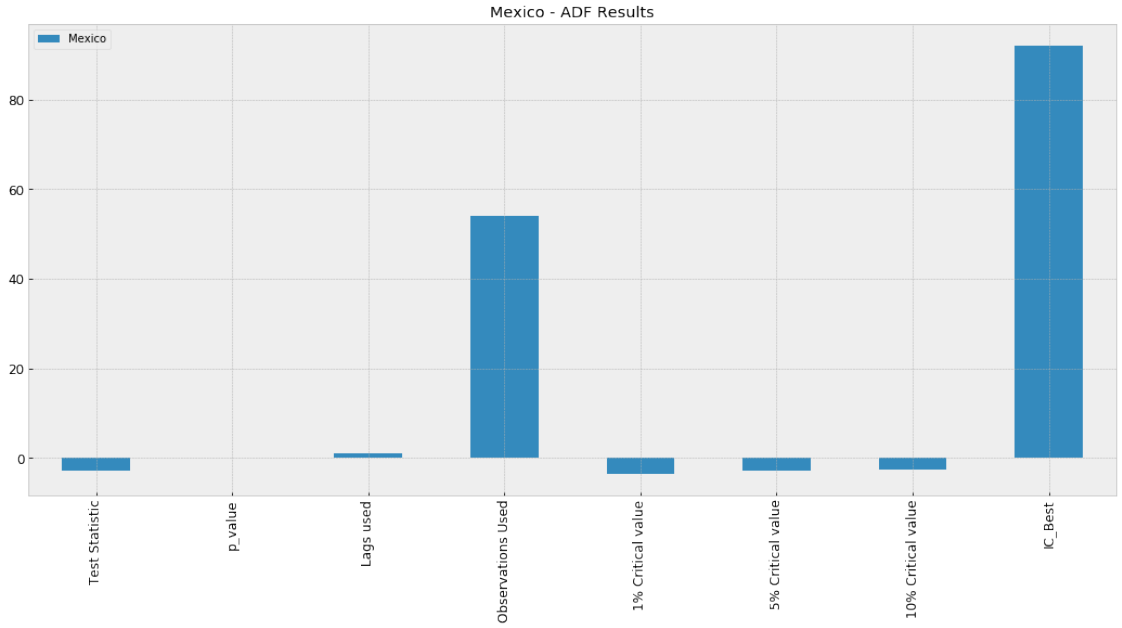

Malta - ADF Results
Malta
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

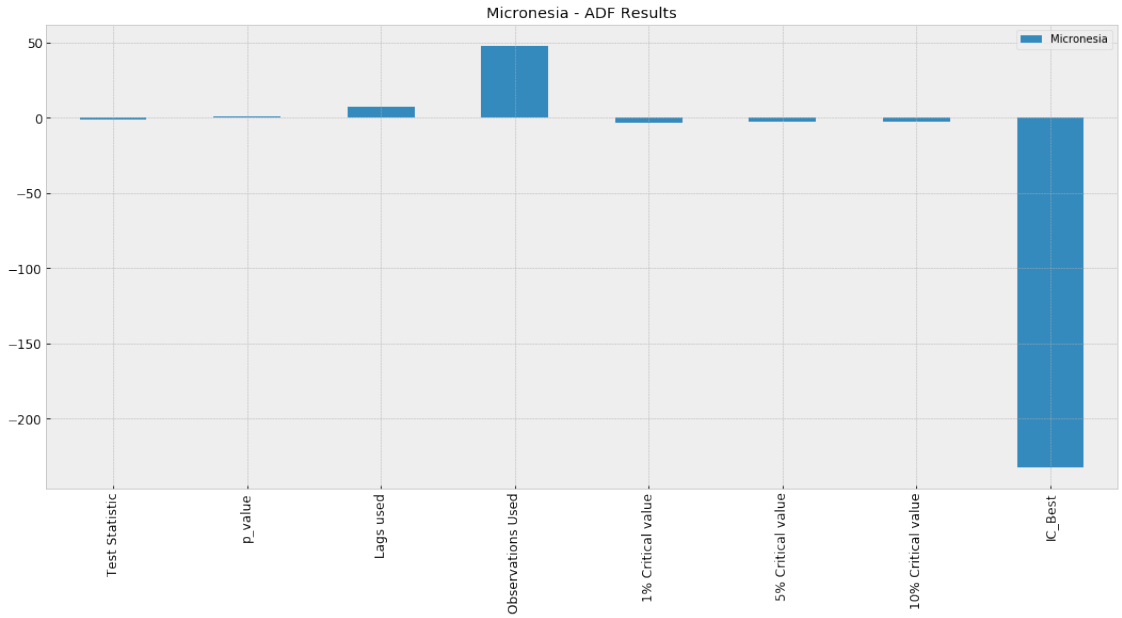

Martinique - ADF Results
Martinique
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

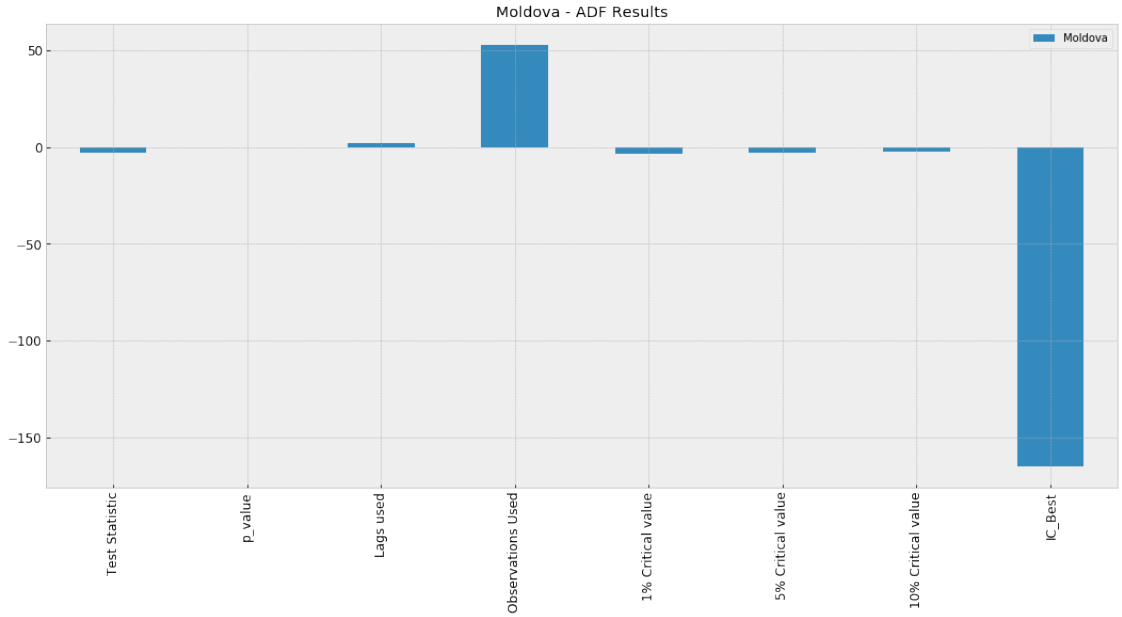

Mauritania - ADF Results
Mauritania
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

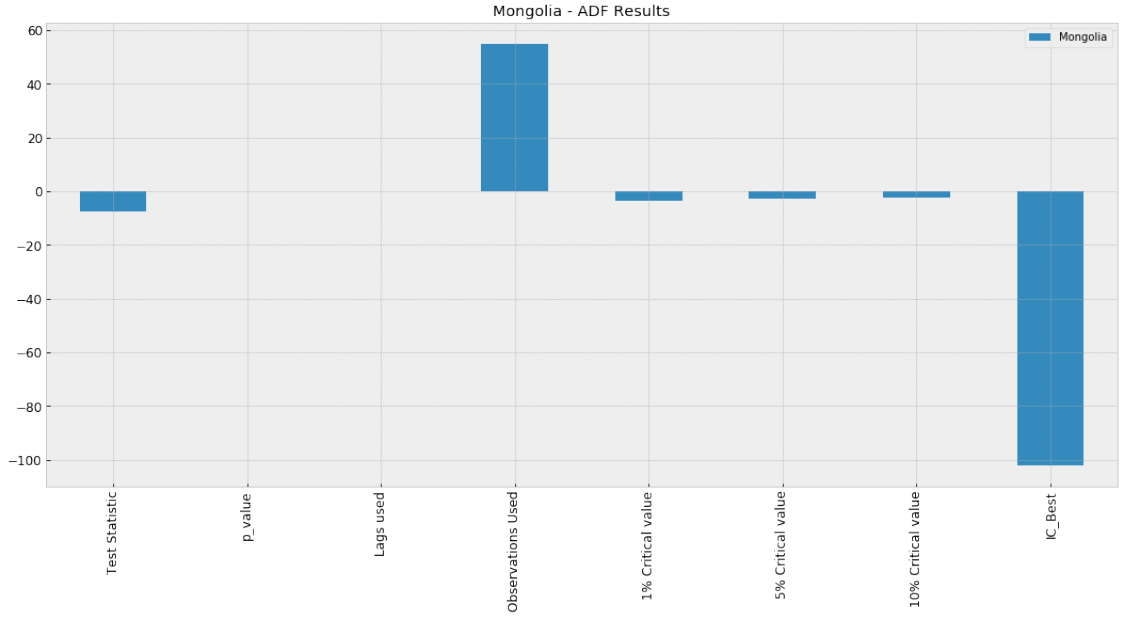

Mauritius - ADF Results
Mauritius
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

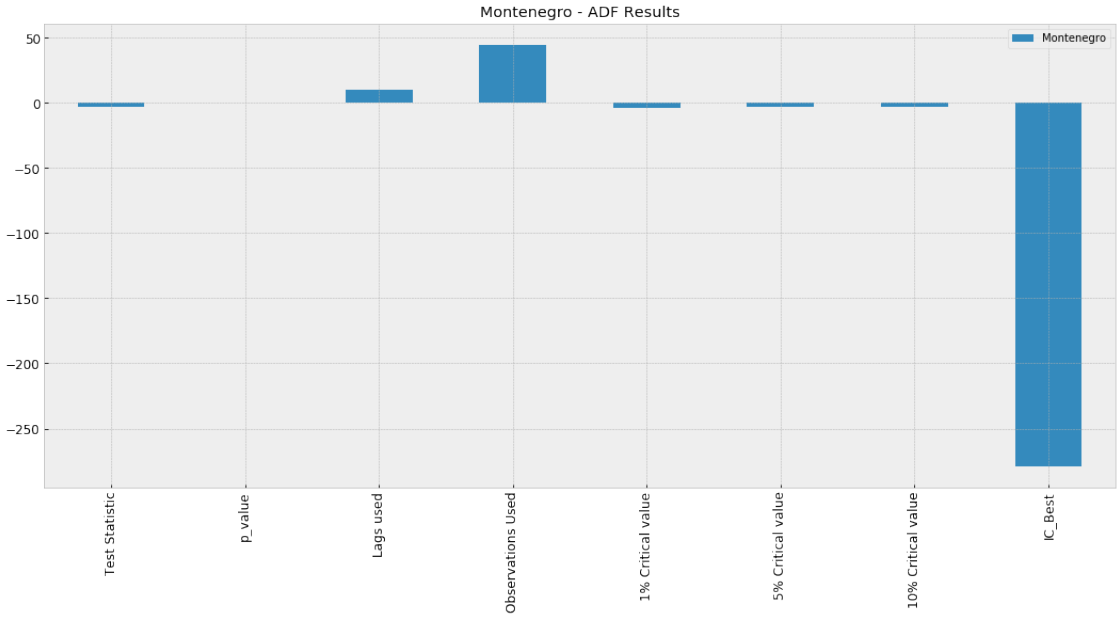

Mayotte - ADF Results
Mayotte
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

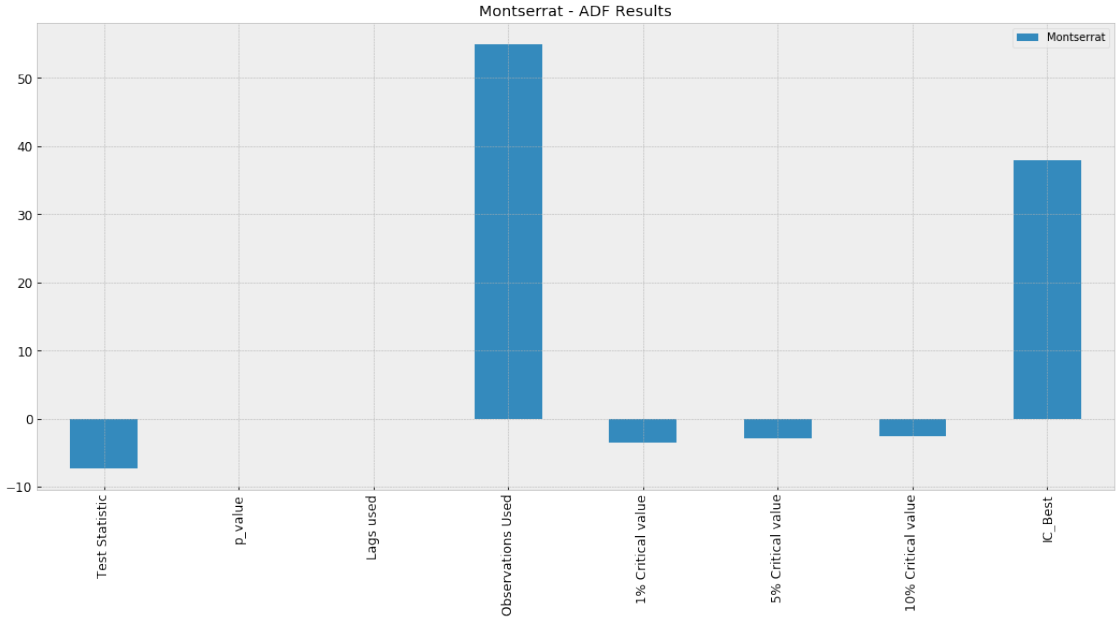

Mexico - ADF Results
Mexico
80
60
40
20
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

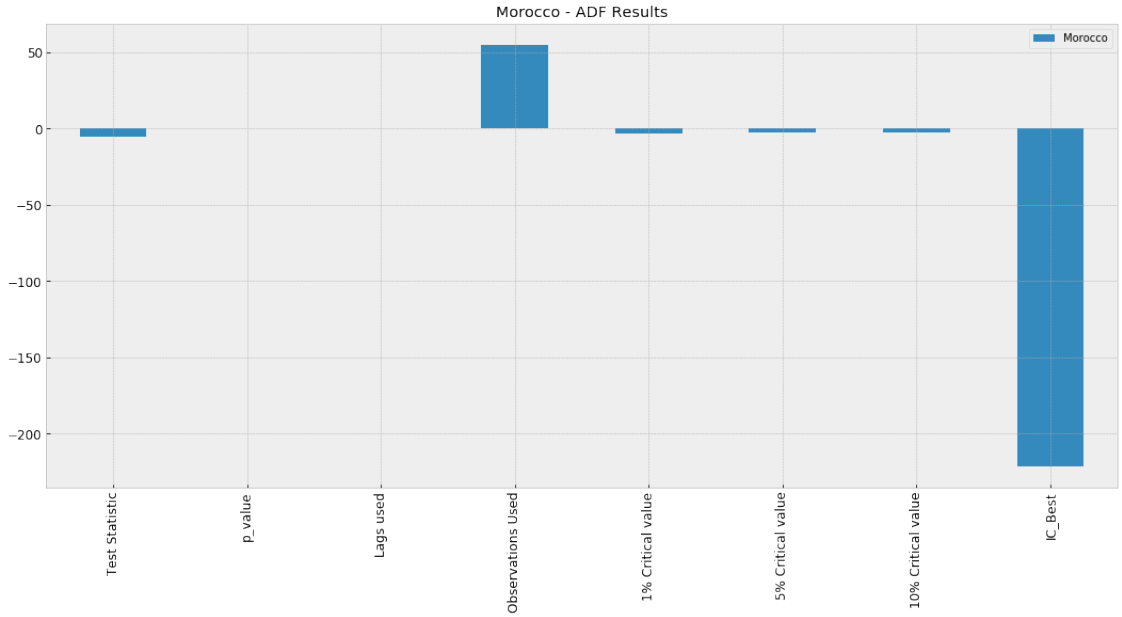

Micronesia - ADF Results
Micronesia
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

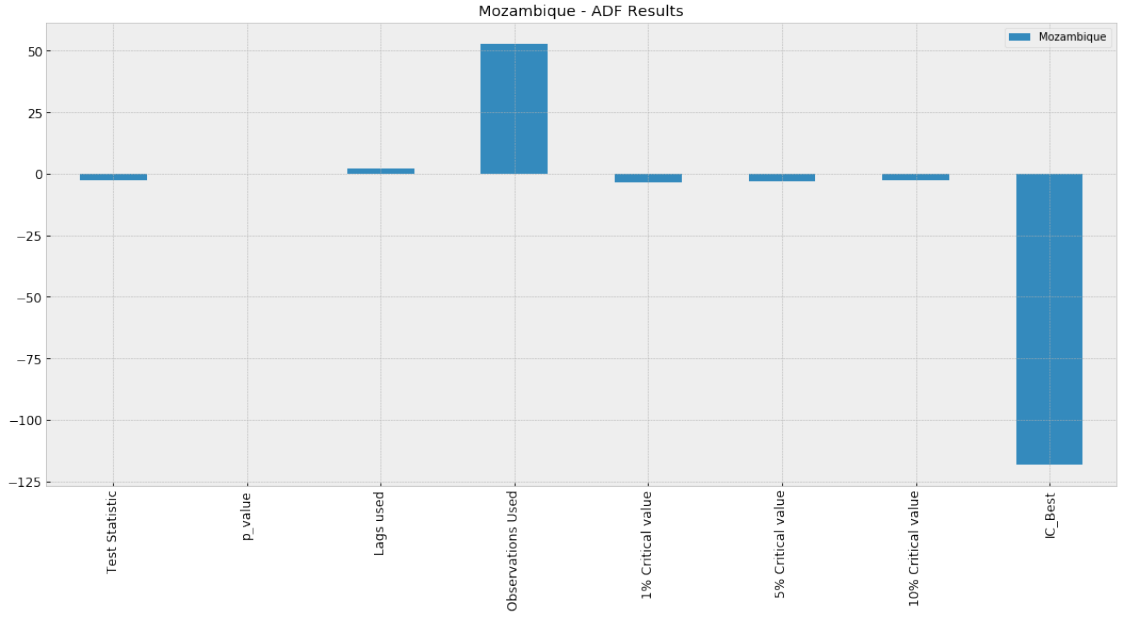

Moldova - ADF Results
Moldova
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

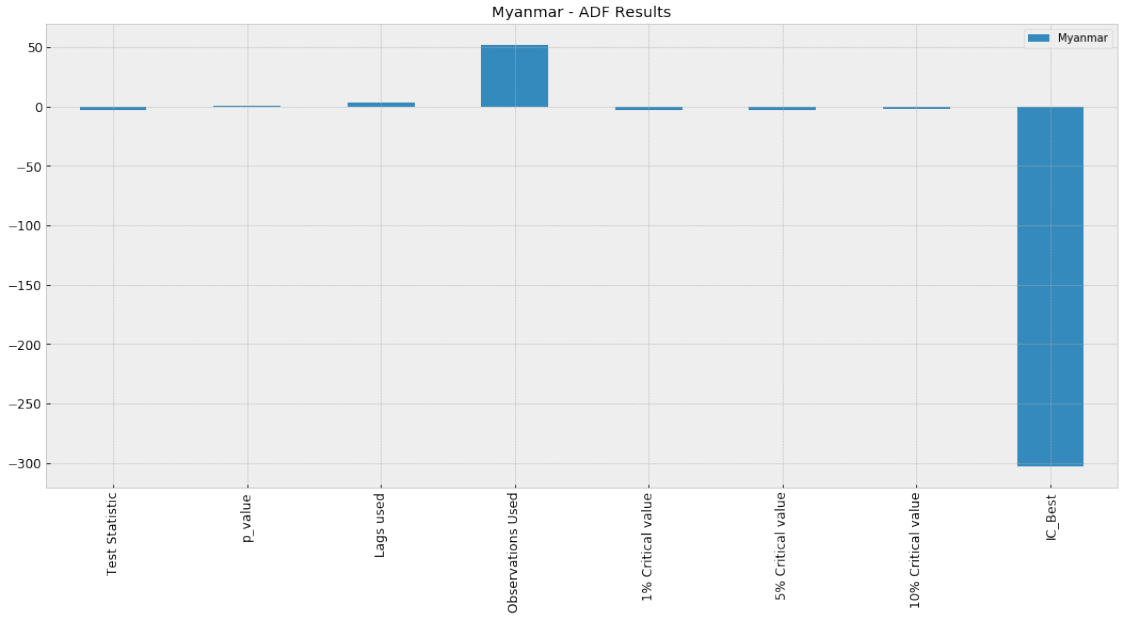

Mongolia - ADF Results
Mongolia
60
40
20
0
−20
−40
−60
−80
−100
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

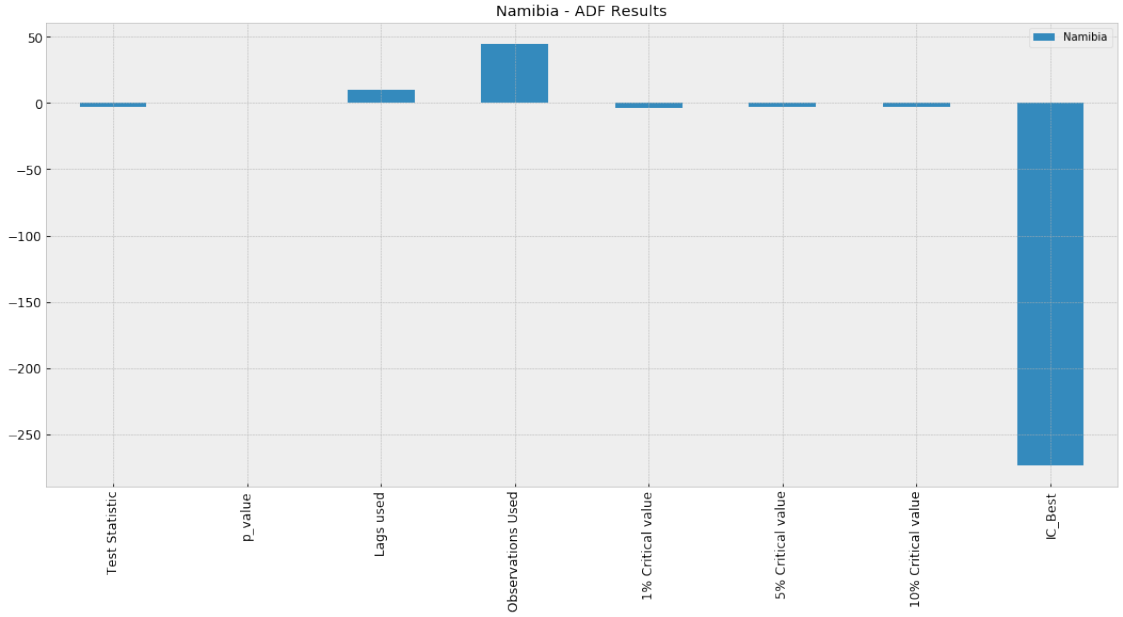

Montenegro - ADF Results
Montenegro
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

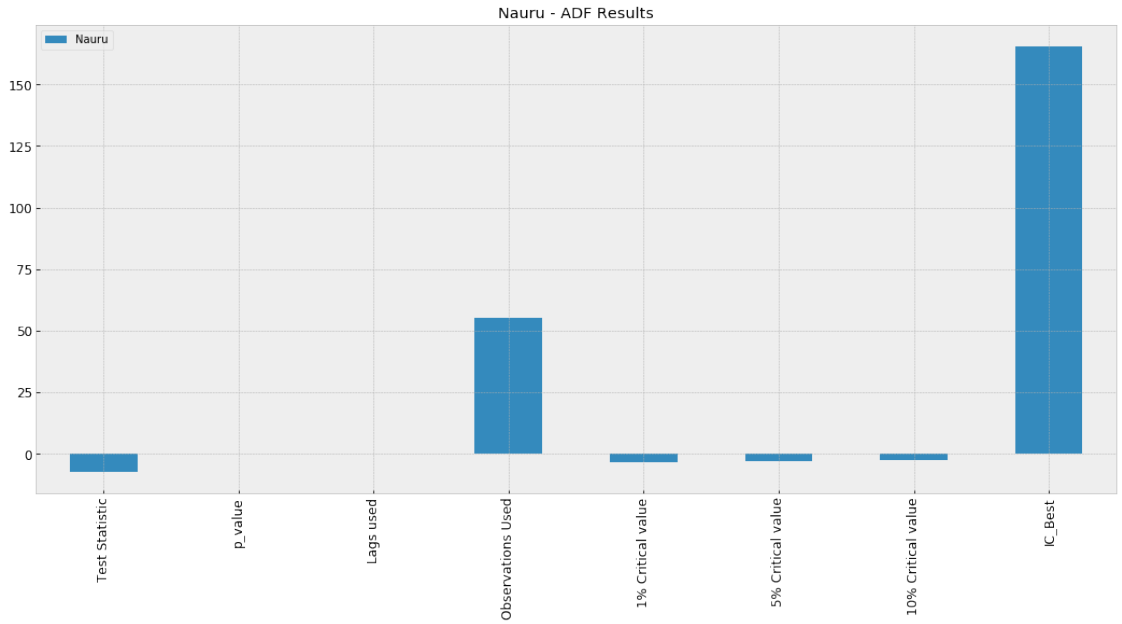

Montserrat - ADF Results
Montserrat
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

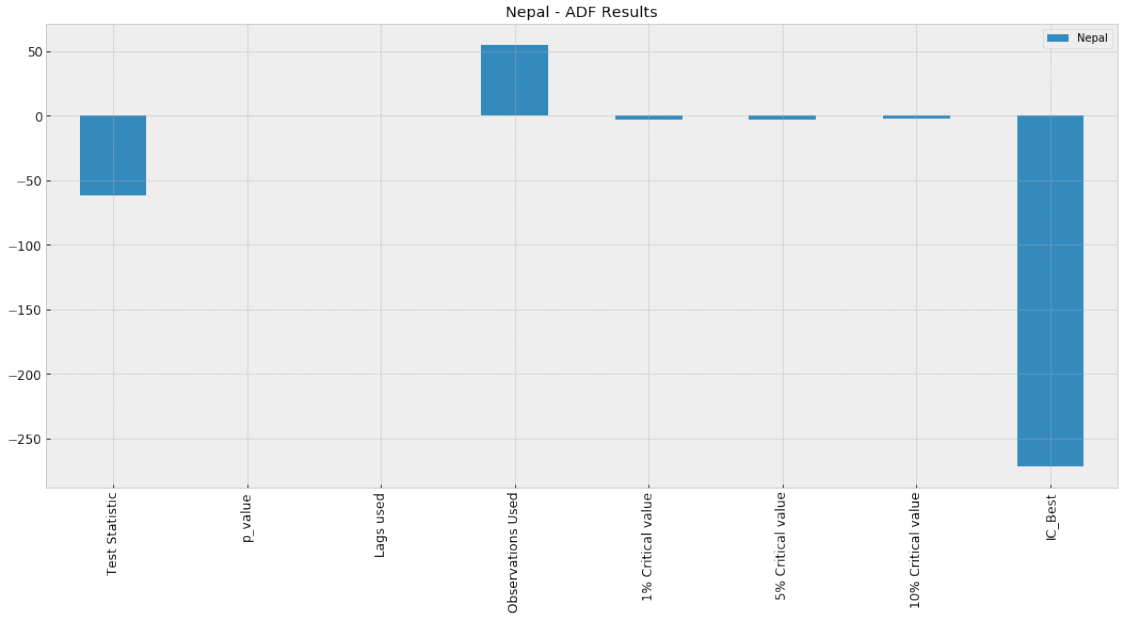

Morocco - ADF Results
Morocco
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

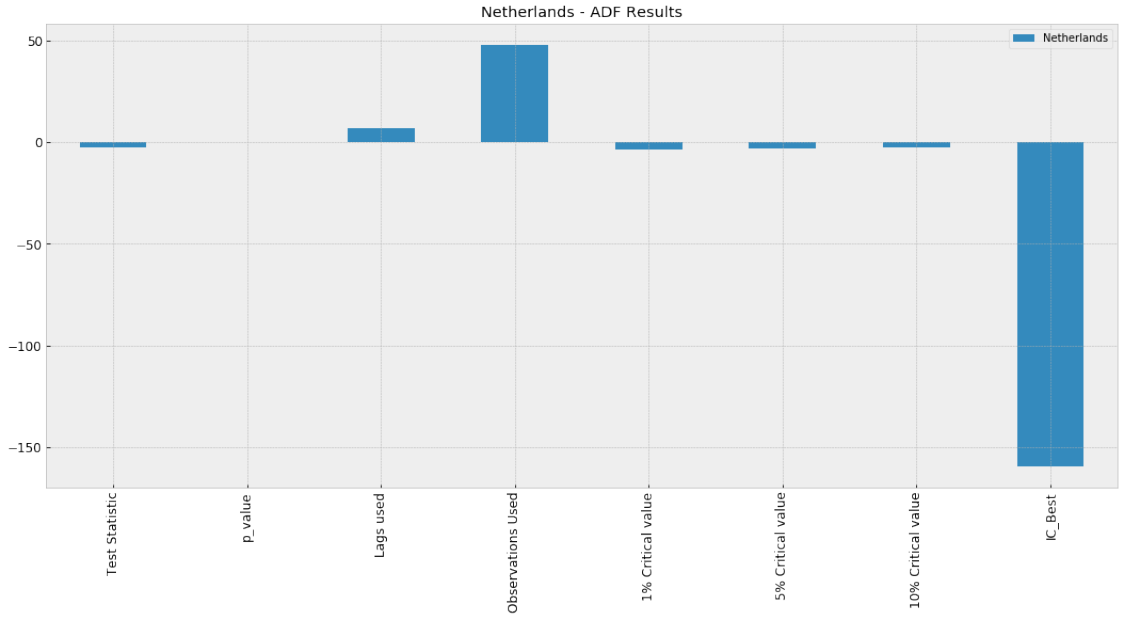

Mozambique - ADF Results
Mozambique
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

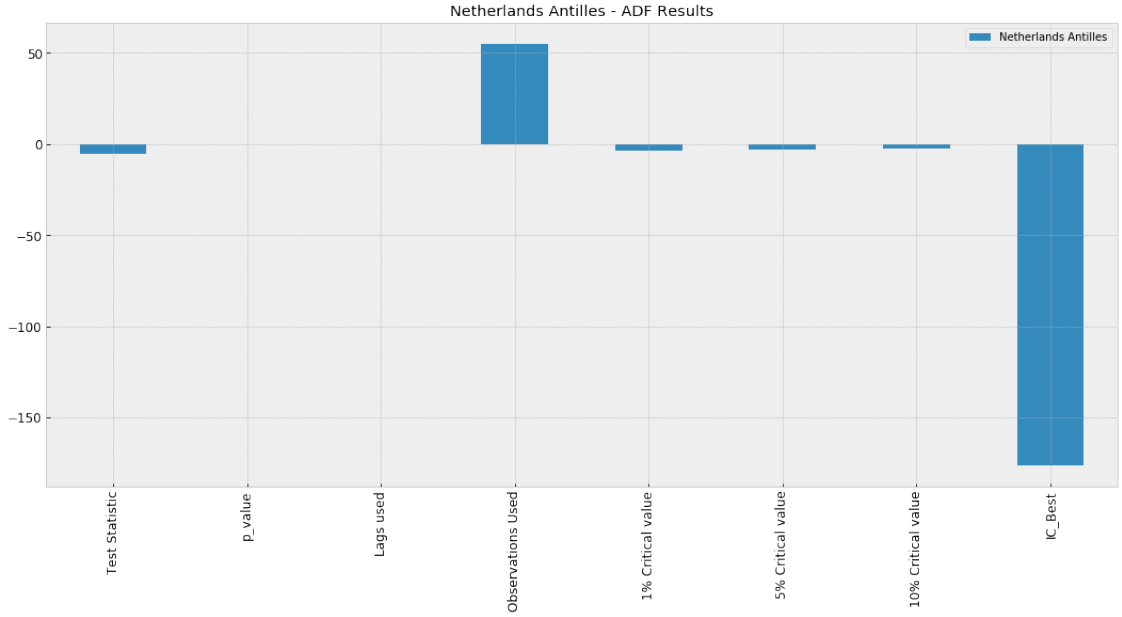

Myanmar - ADF Results
Myanmar
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

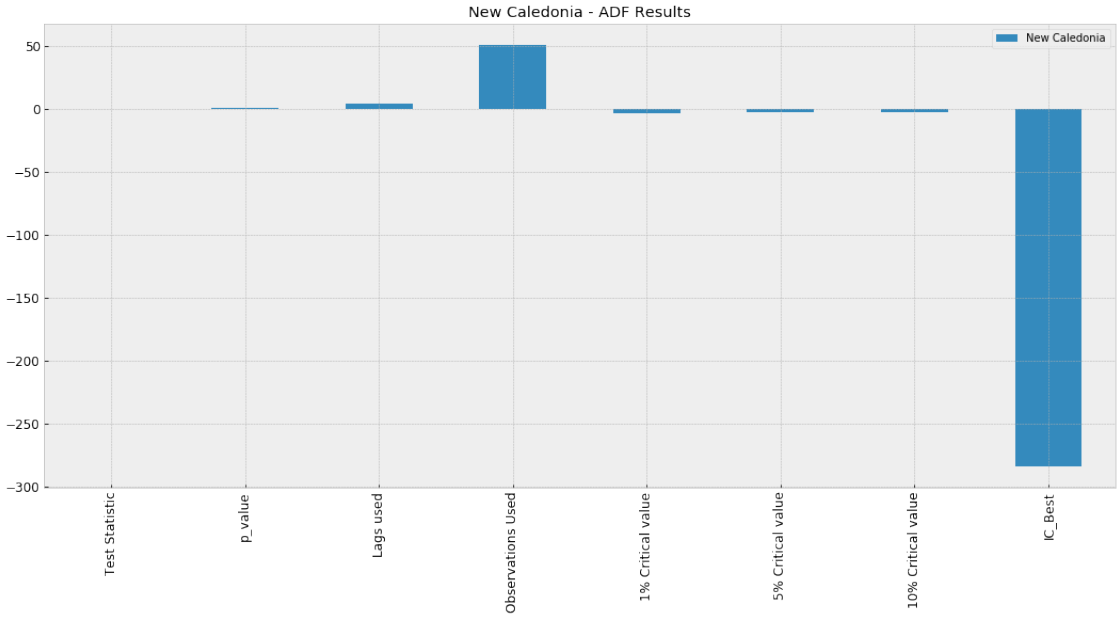

Namibia - ADF Results
Namibia
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

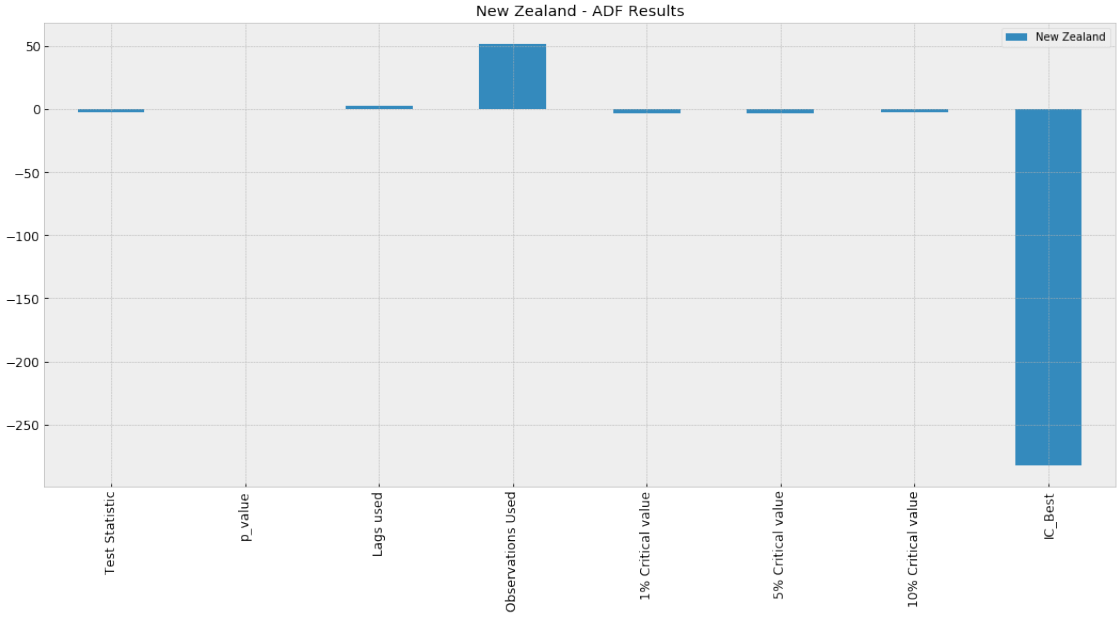

Nauru - ADF Results
Nauru
150
125
100
75
50
25
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

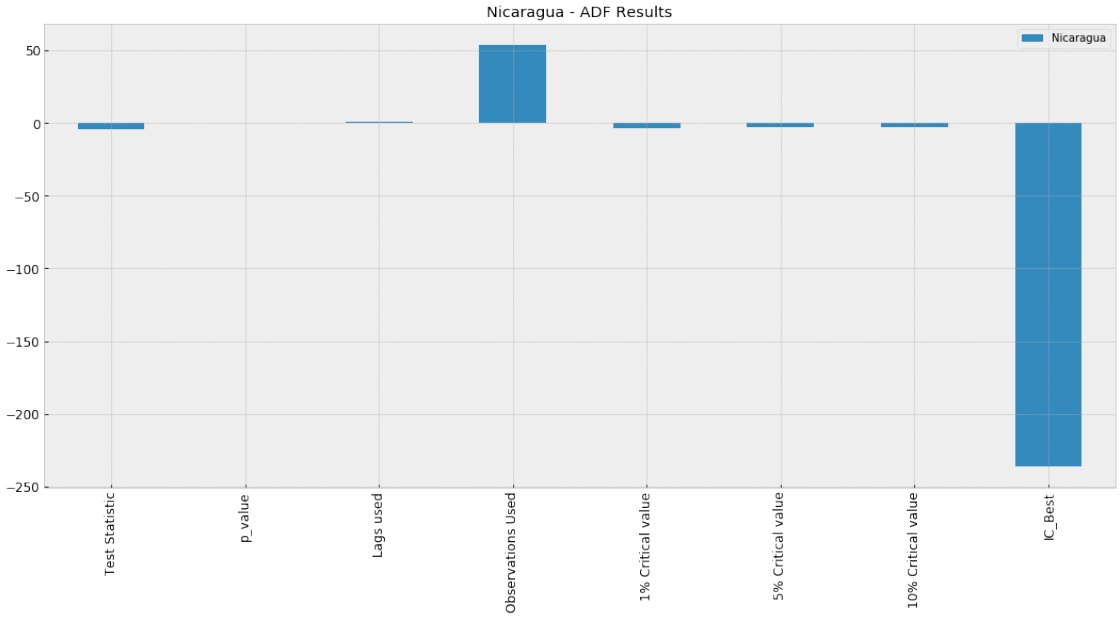

Nepal - ADF Results
Nepal
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

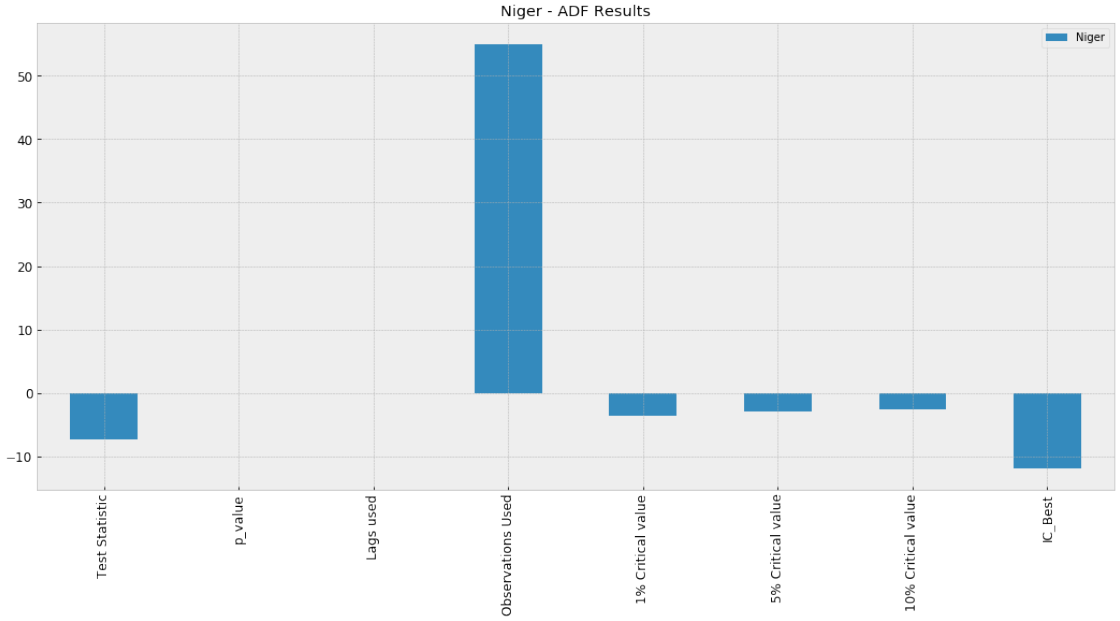

Netherlands - ADF Results
Netherlands
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

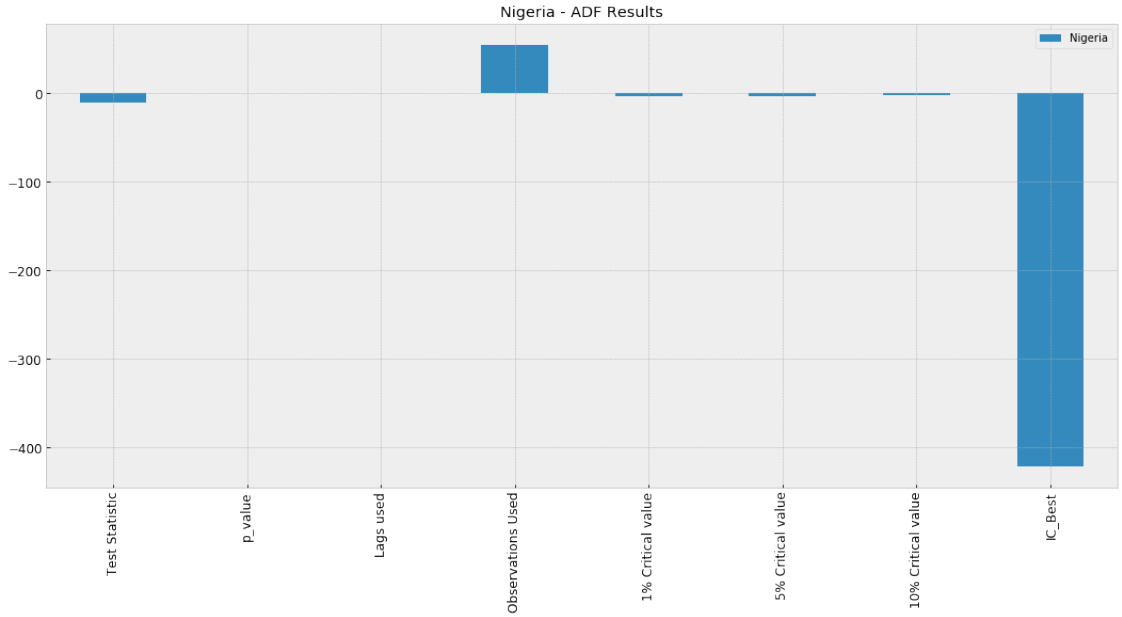

Netherlands Antilles - ADF Results
Netherlands Antilles
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

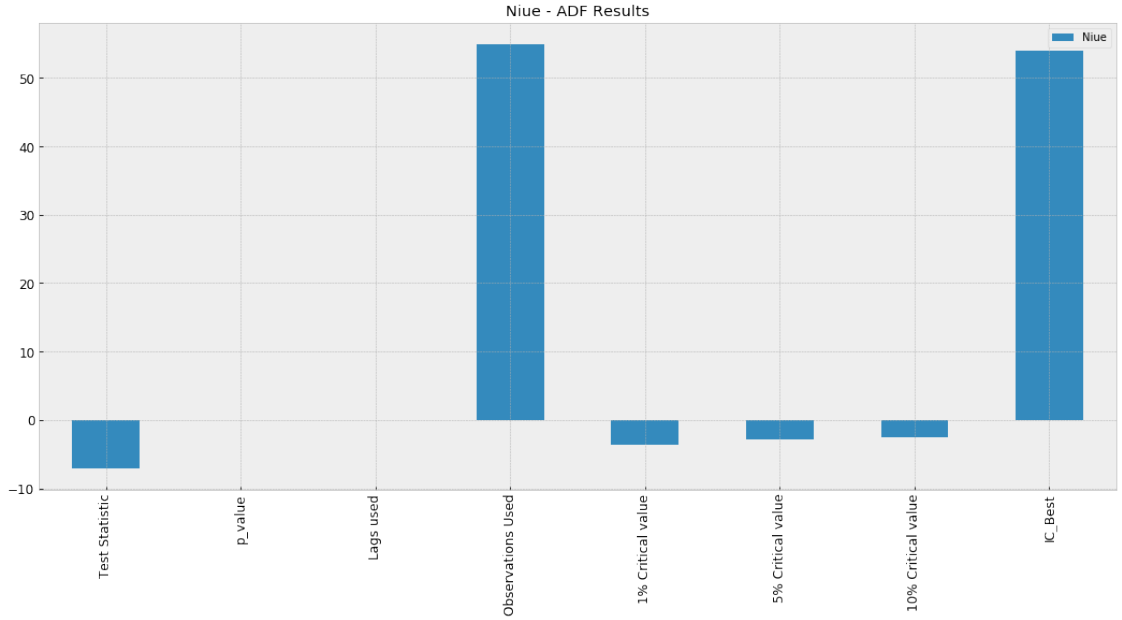

New Caledonia - ADF Results
New Caledonia
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

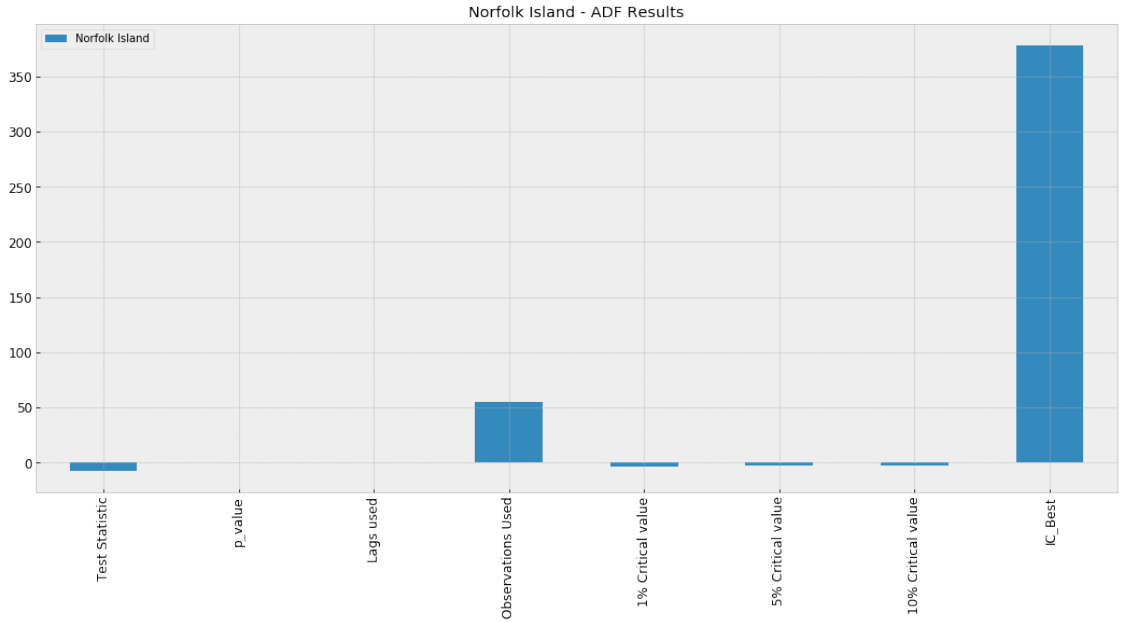

New Zealand - ADF Results
New Zealand
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

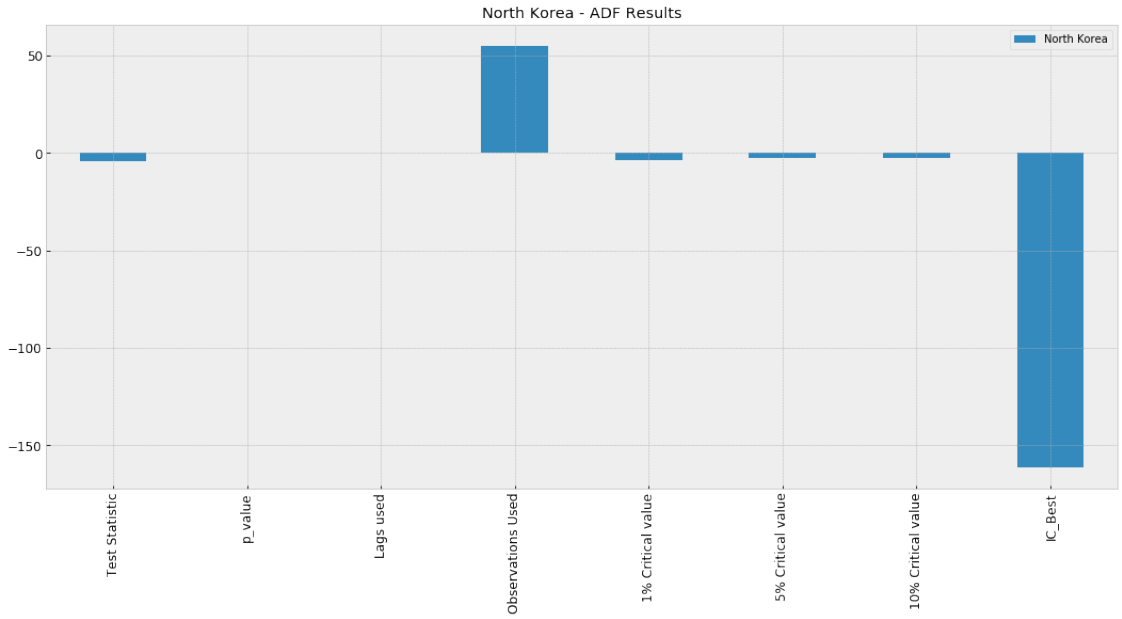

Nicaragua - ADF Results
Nicaragua
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

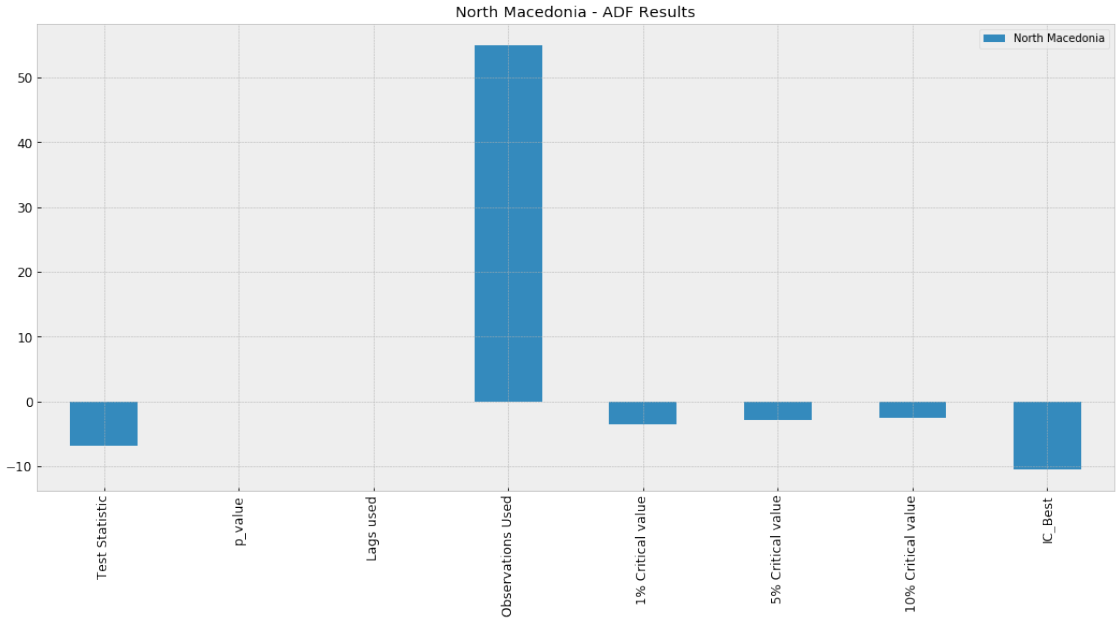

Niger - ADF Results
Niger
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

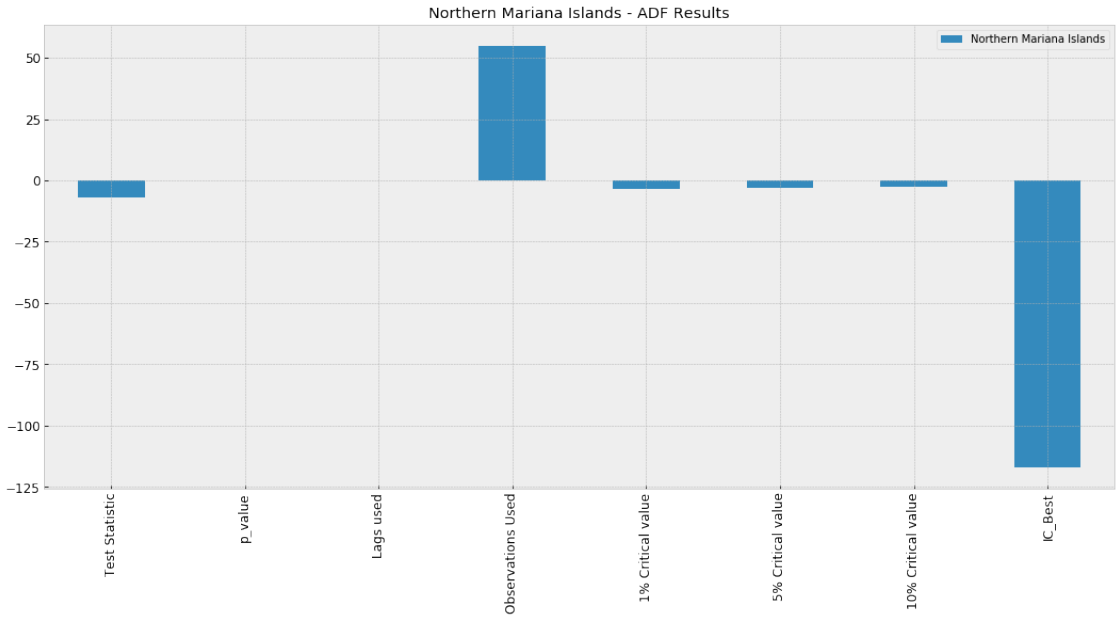

Nigeria - ADF Results
Nigeria
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

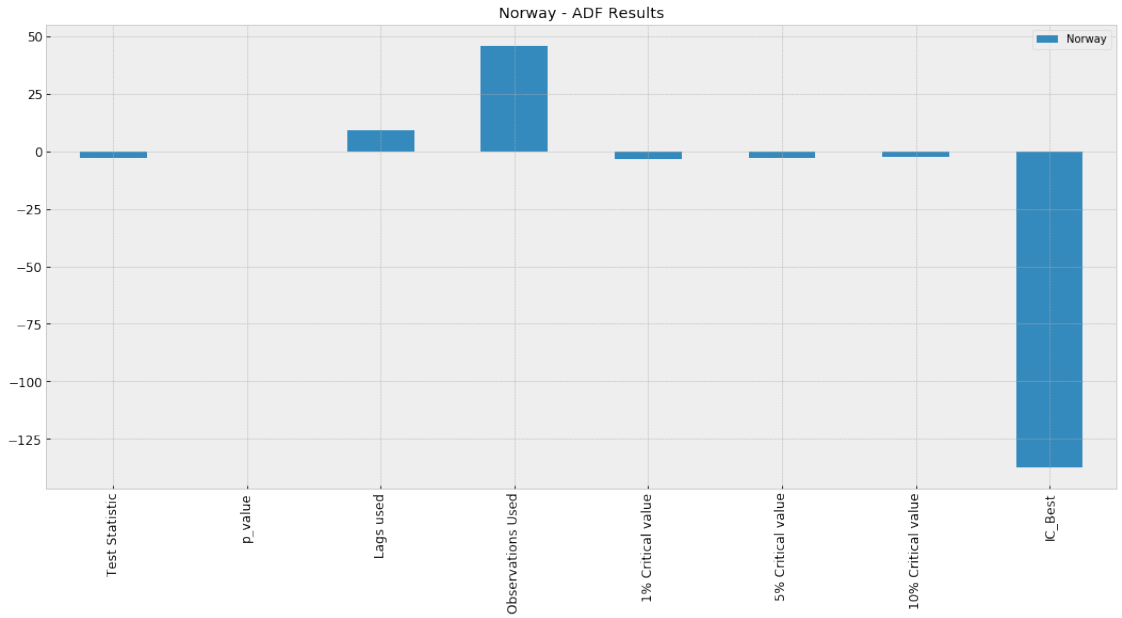

Niue - ADF Results
Niue
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

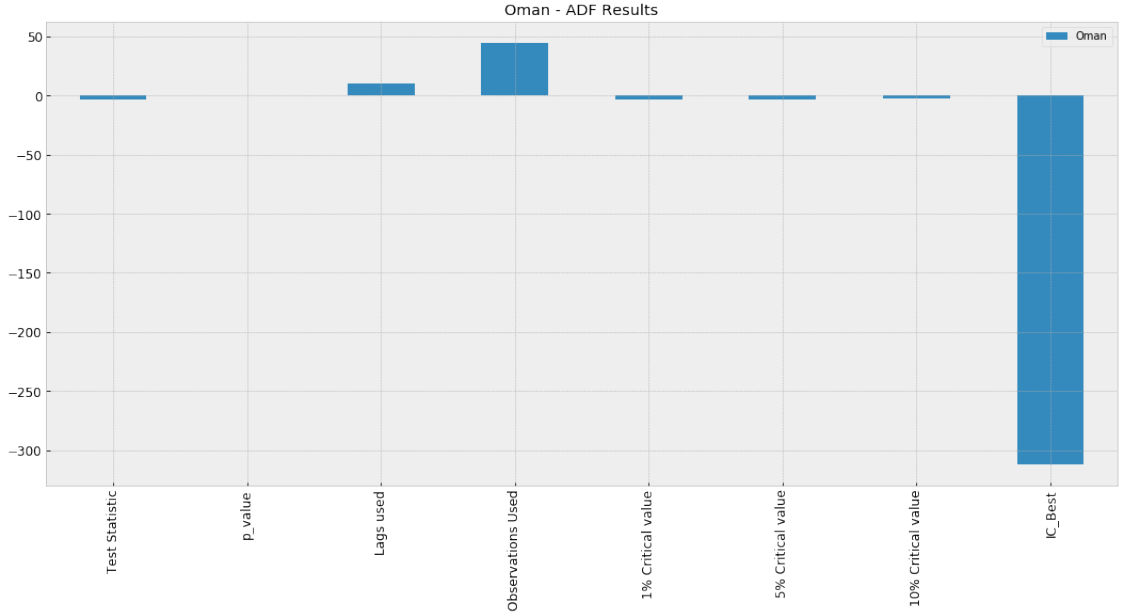

Norfolk Island - ADF Results
Norfolk Island
350
300
250
200
150
100
50
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

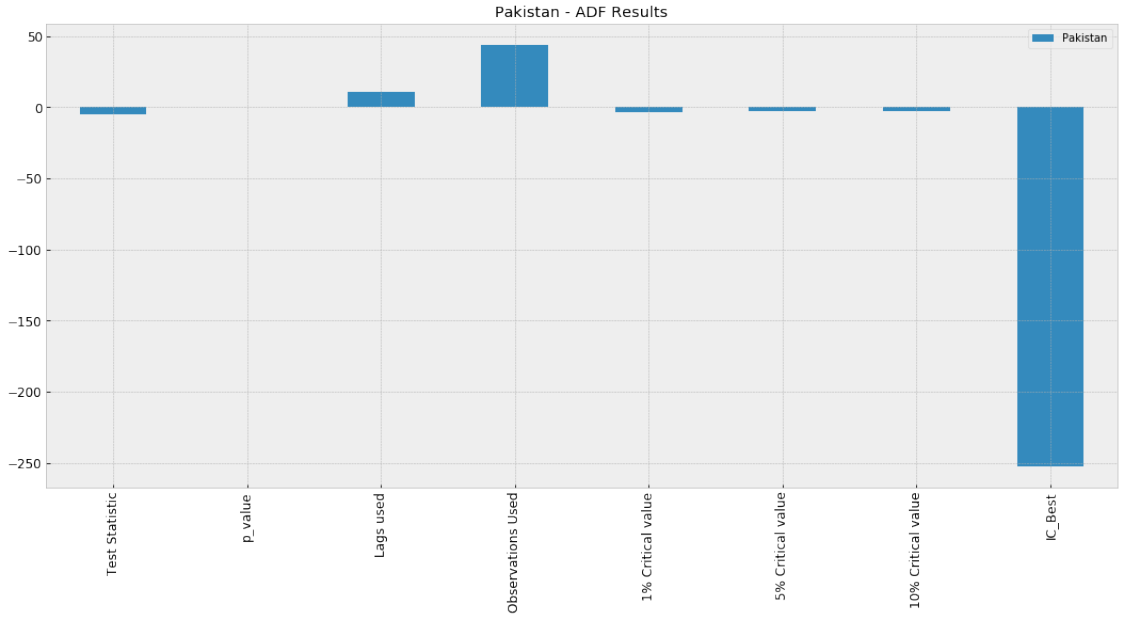

North Korea - ADF Results
North Korea
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

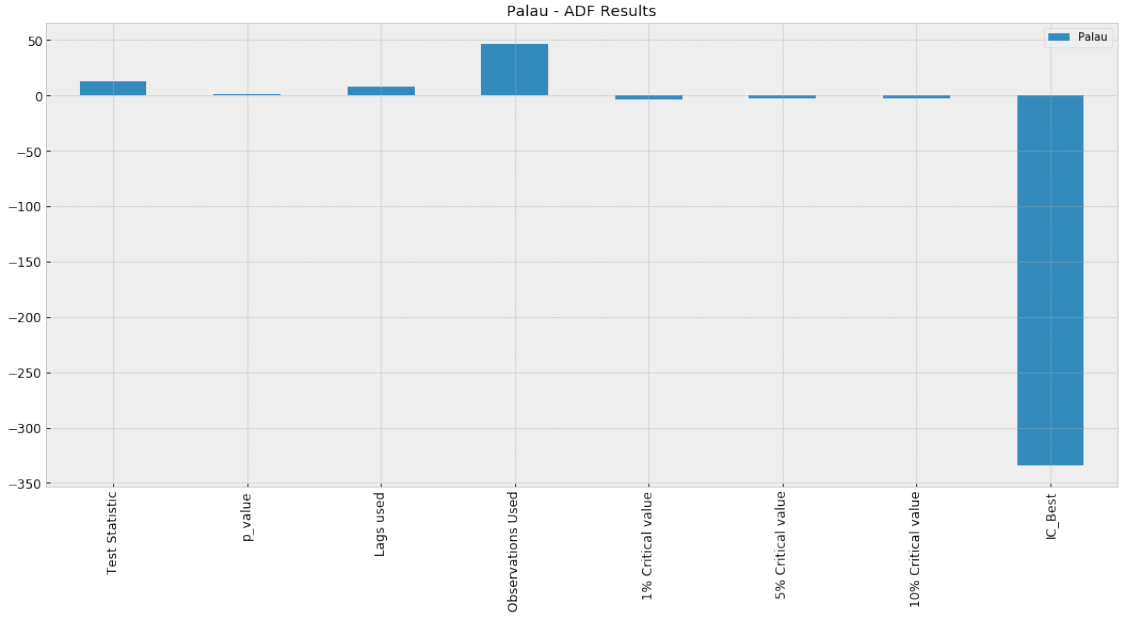

North Macedonia - ADF Results
North Macedonia
50
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

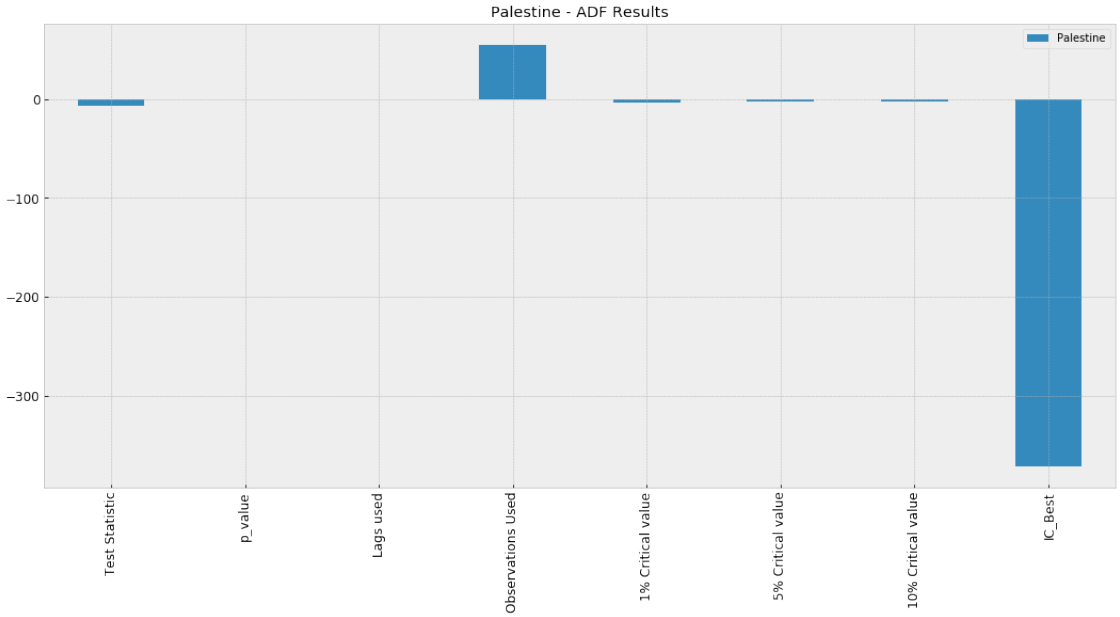

Northern Mariana Islands - ADF Results
Northern Mariana Islands
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

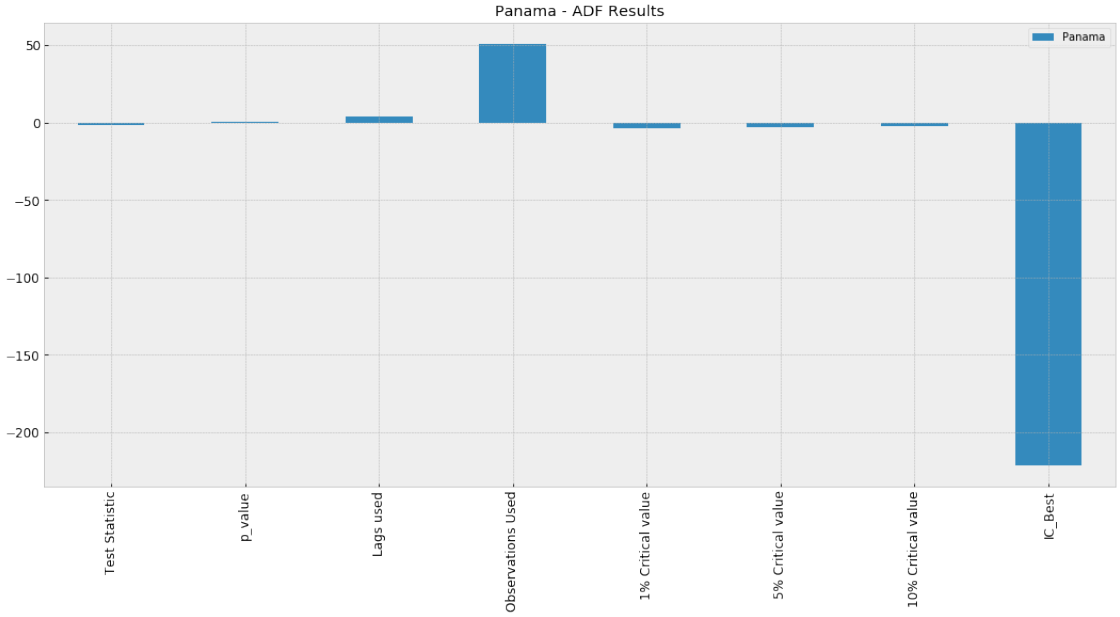

Norway - ADF Results
Norway
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

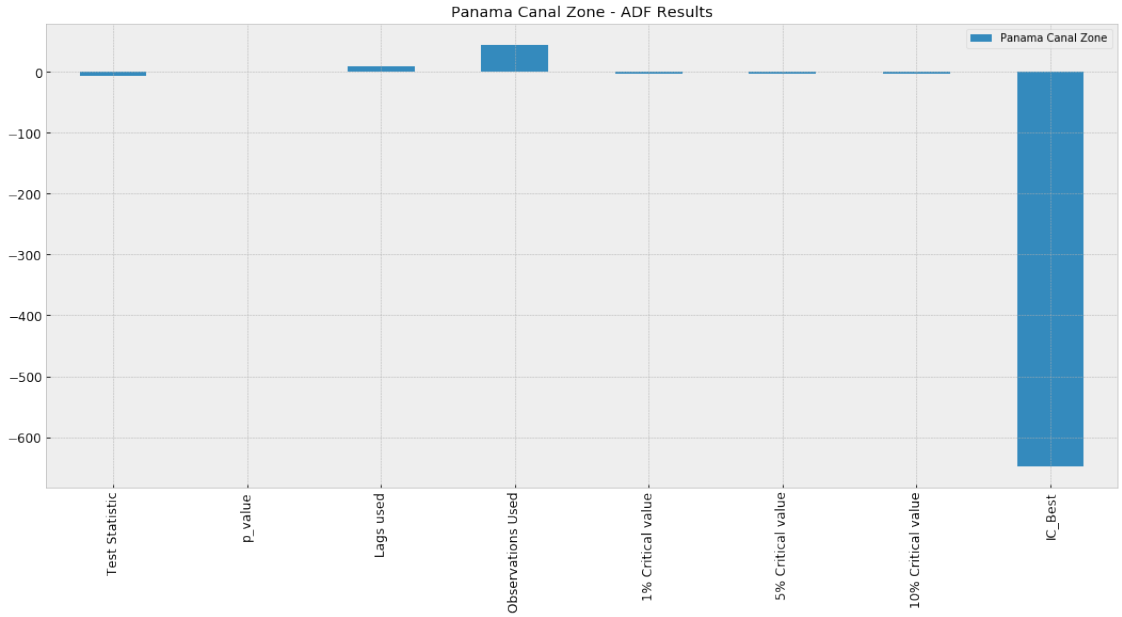

Oman - ADF Results
Oman
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

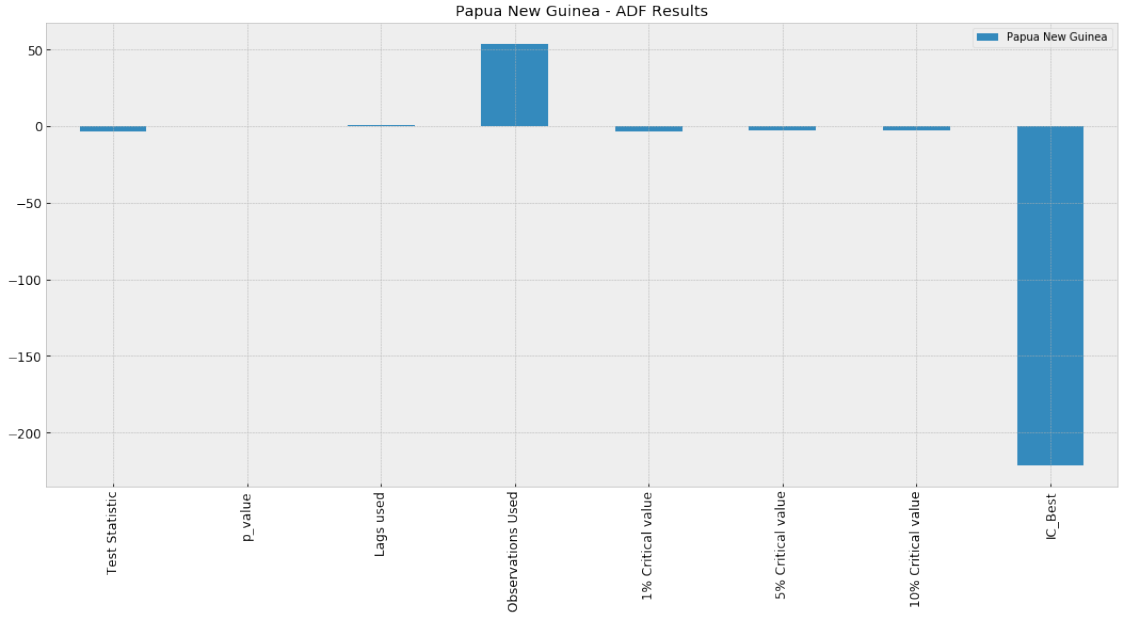

Pakistan - ADF Results
Pakistan
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

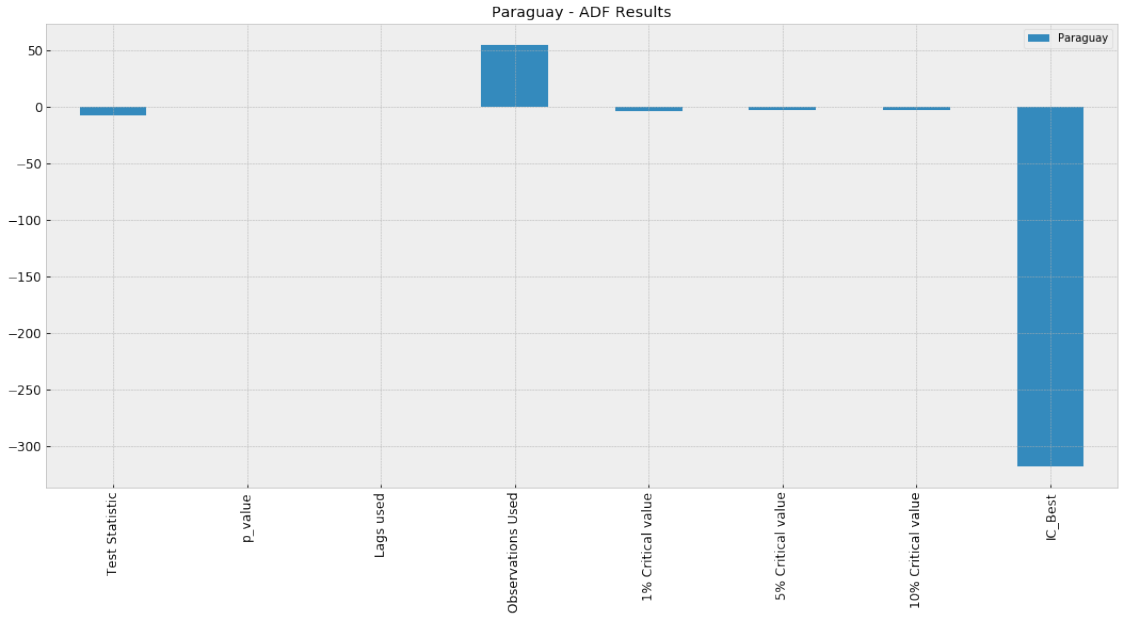

Palau - ADF Results
Palau
50
0
−50
−100
−150
−200
−250
−300
−350
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

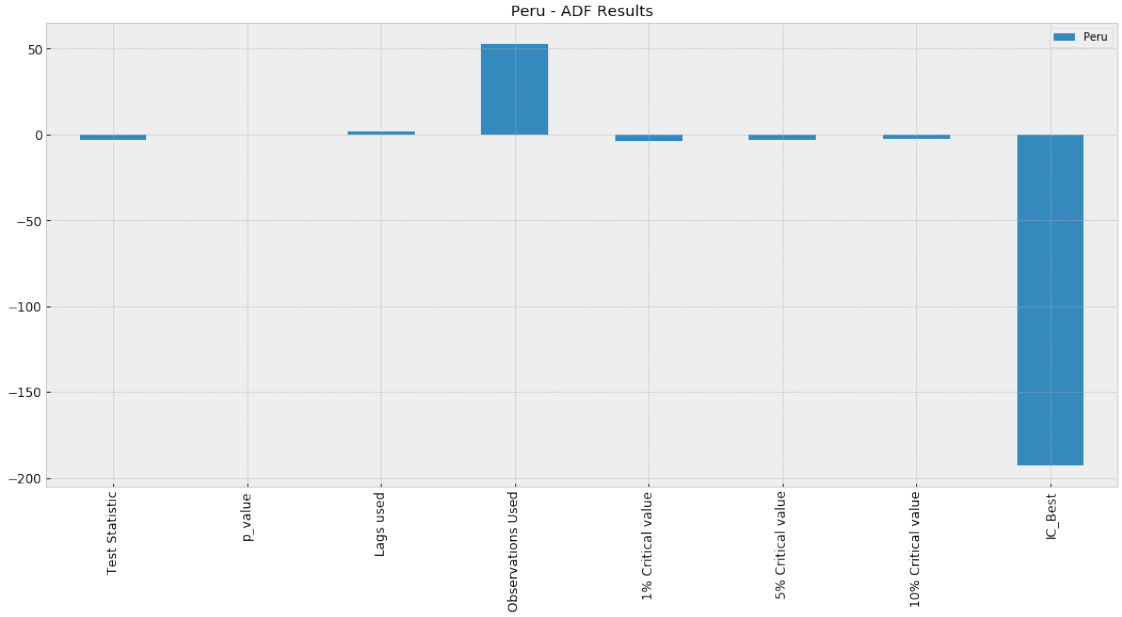

Palestine - ADF Results
Palestine
0
−100
−200
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

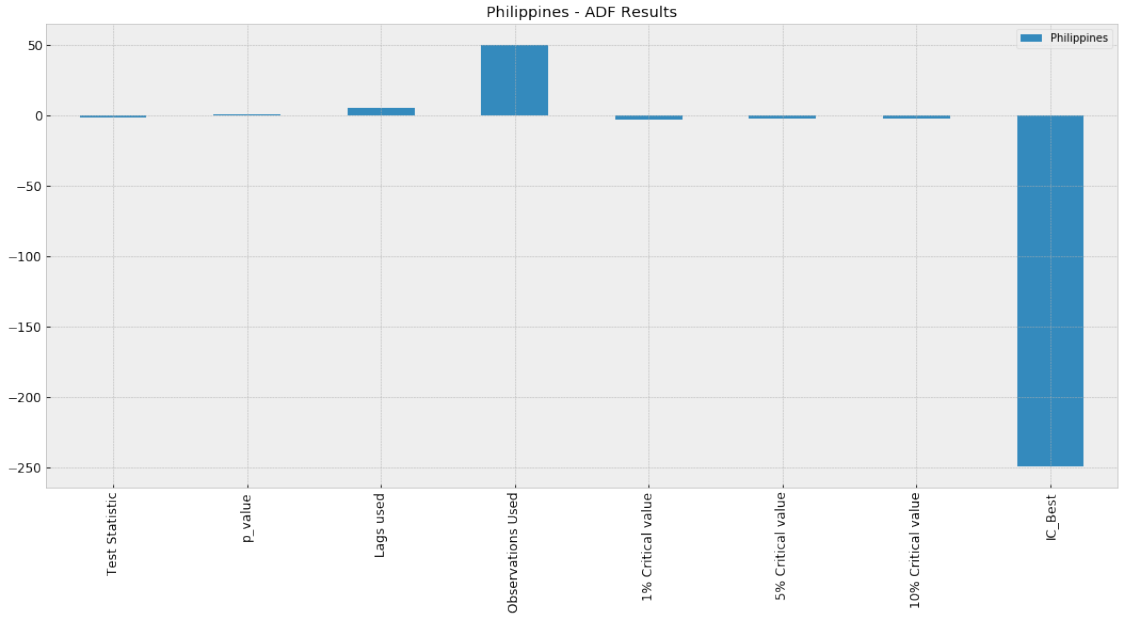

Panama - ADF Results
Panama
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

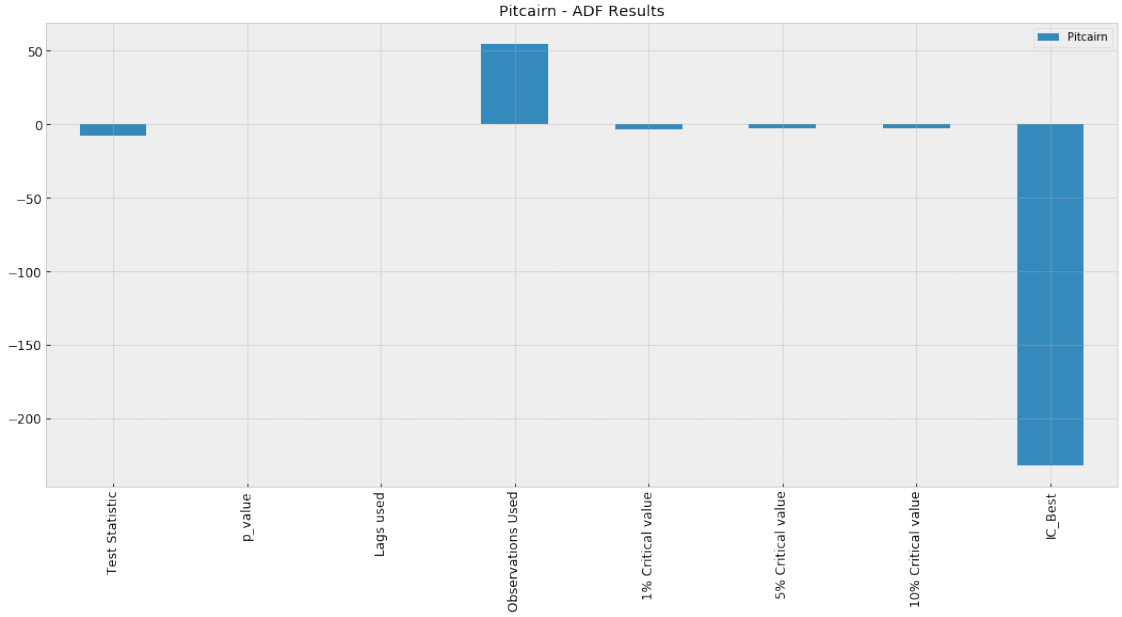

Panama Canal Zone - ADF Results
Panama Canal Zone
0
−100
−200
−300
−400
−500
−600
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

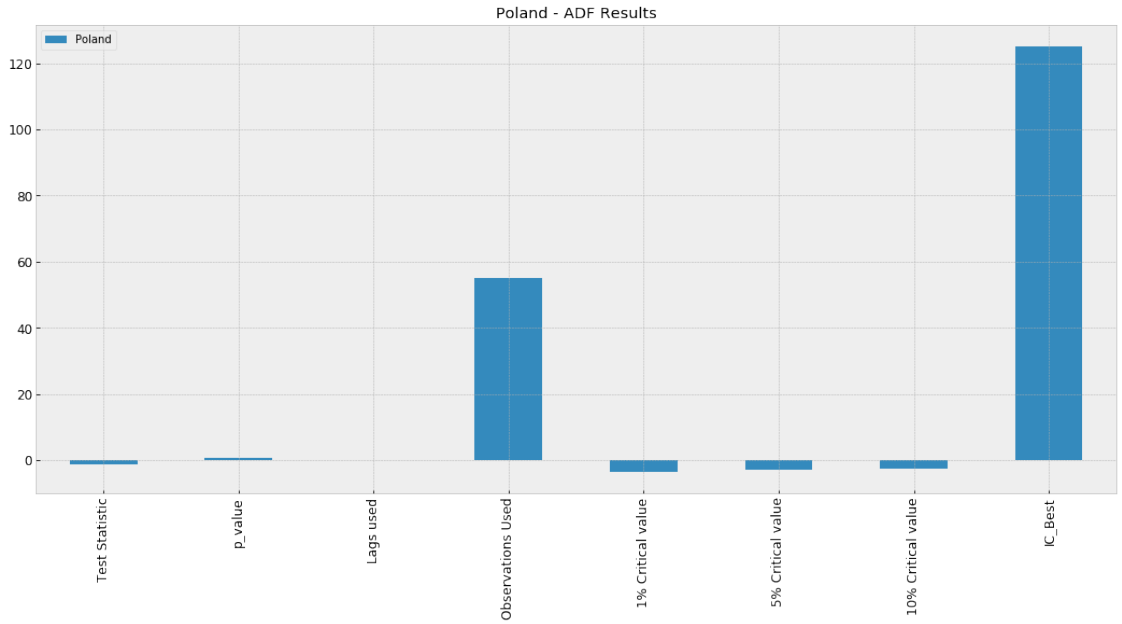

Papua New Guinea - ADF Results
Papua New Guinea
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

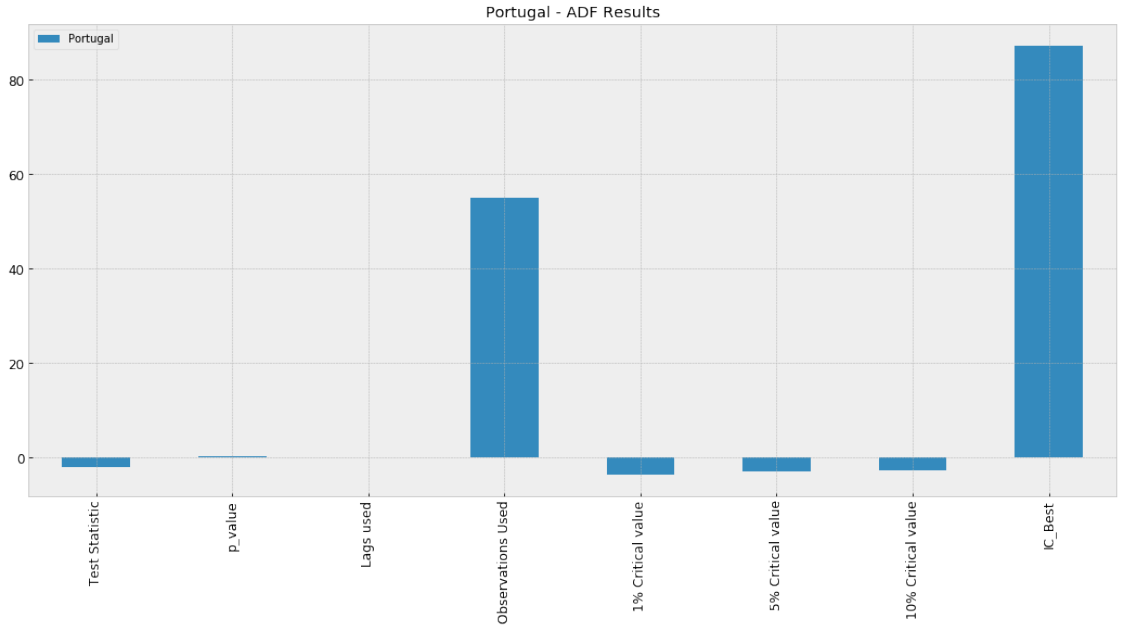

Paraguay - ADF Results
Paraguay
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

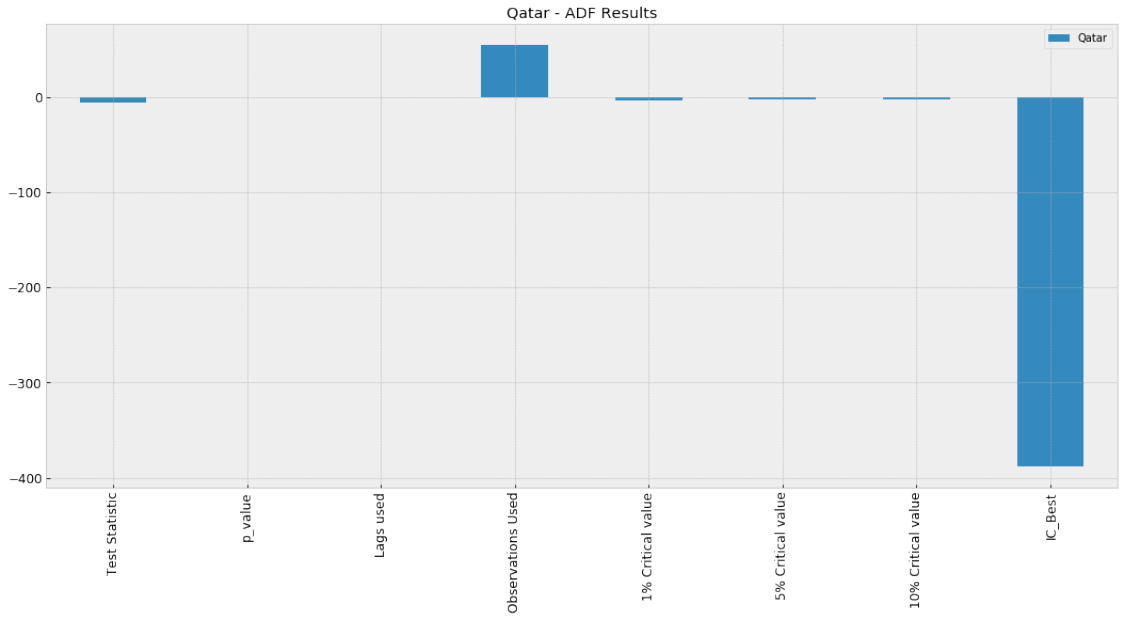

Peru - ADF Results
Peru
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

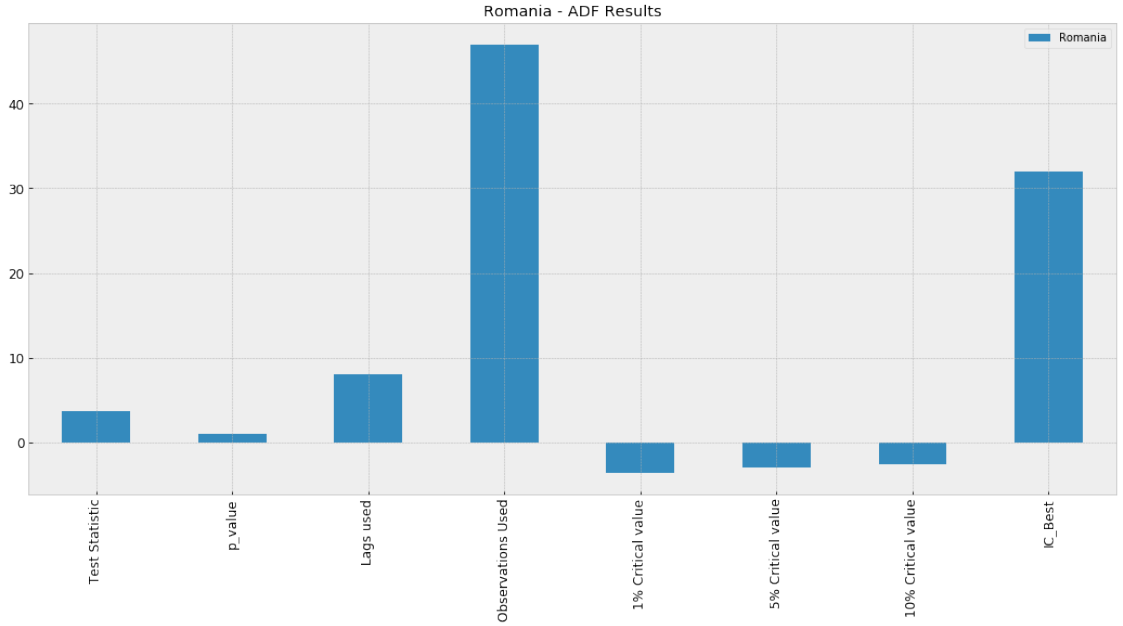

Philippines - ADF Results
Philippines
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

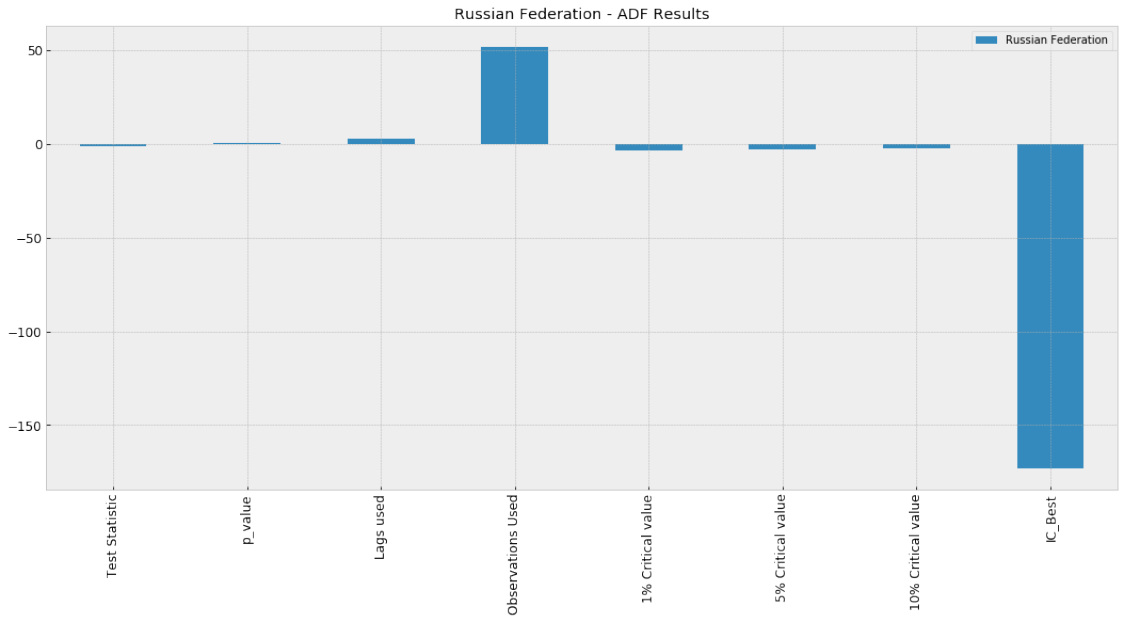

Pitcairn - ADF Results
Pitcairn
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

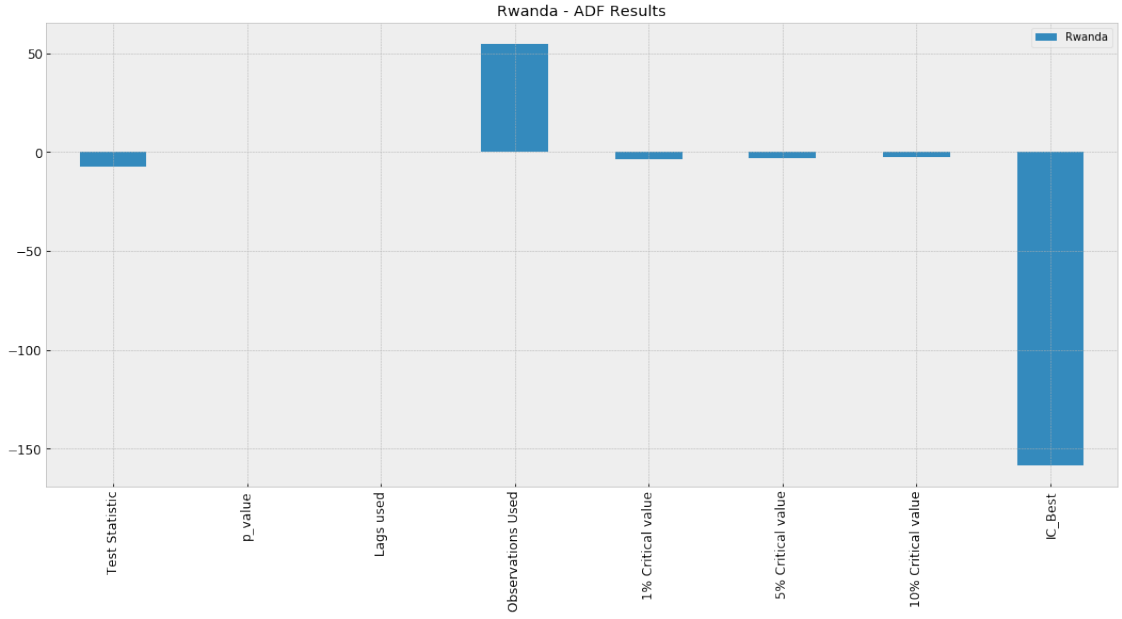

Poland - ADF Results
Poland
120
100
80
60
40
20
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

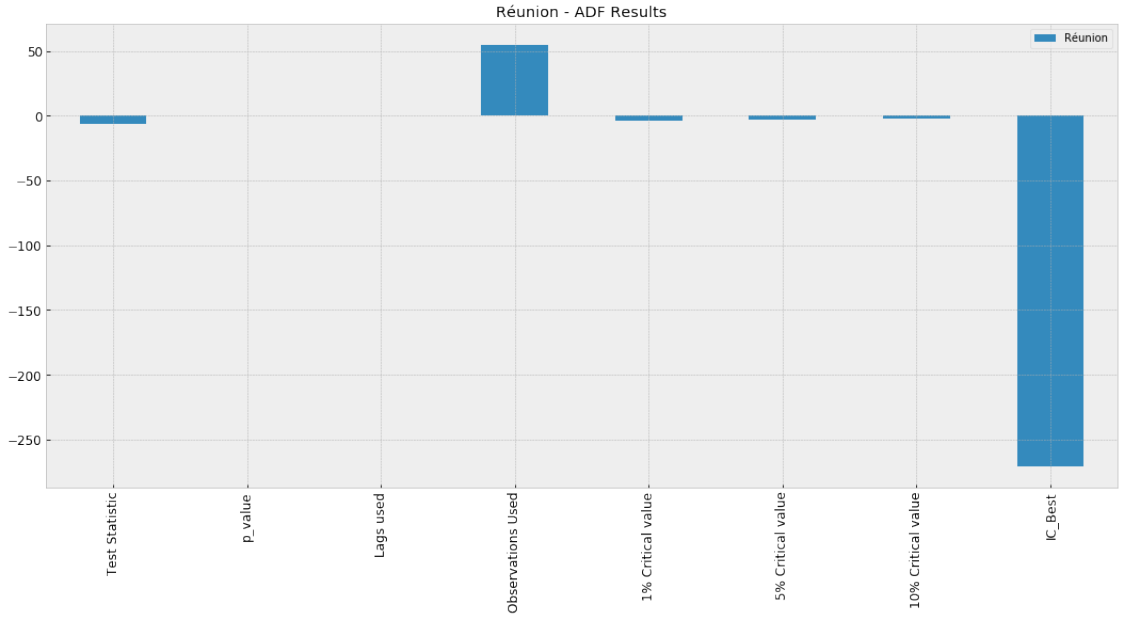

Portugal - ADF Results
Portugal
80
60
40
20
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

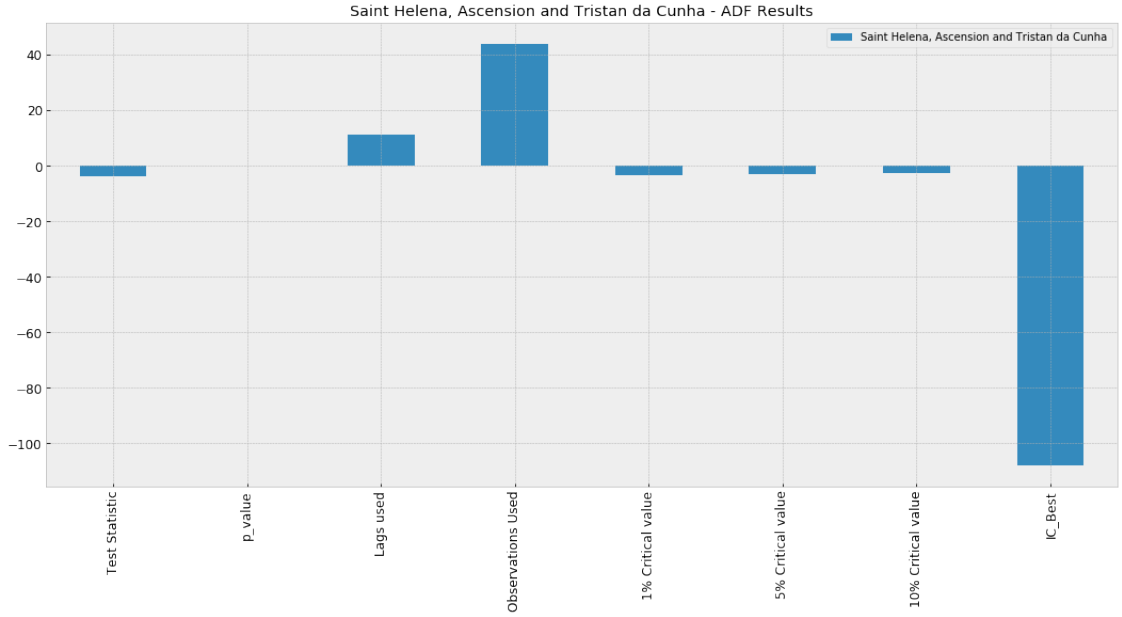

Qatar - ADF Results
Qatar
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

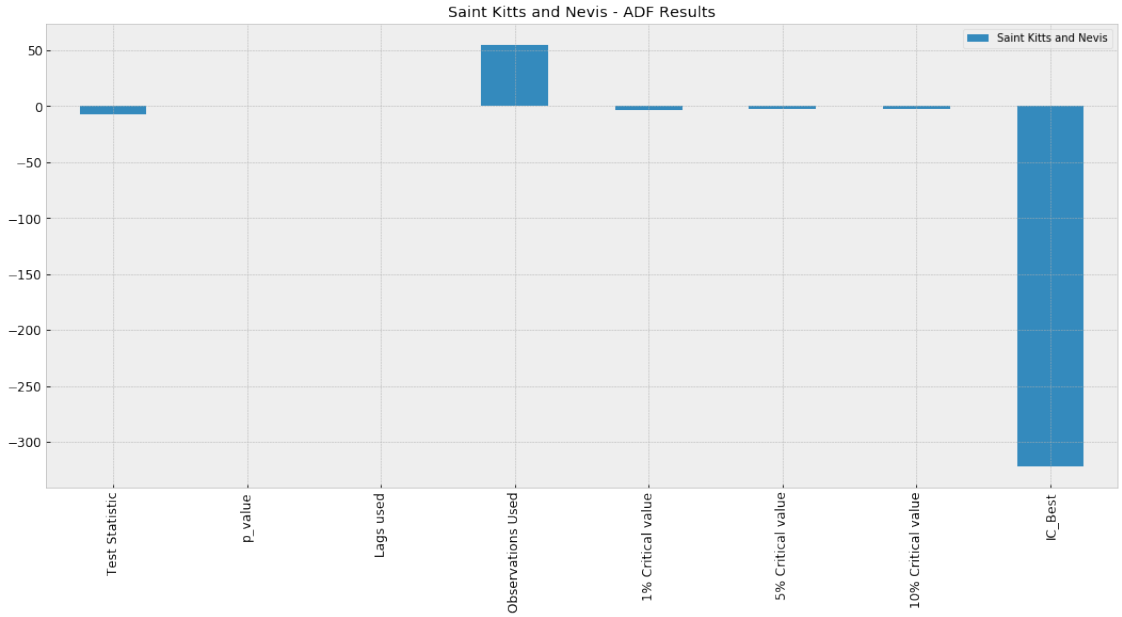

Romania - ADF Results
Romania
40
30
20
10
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

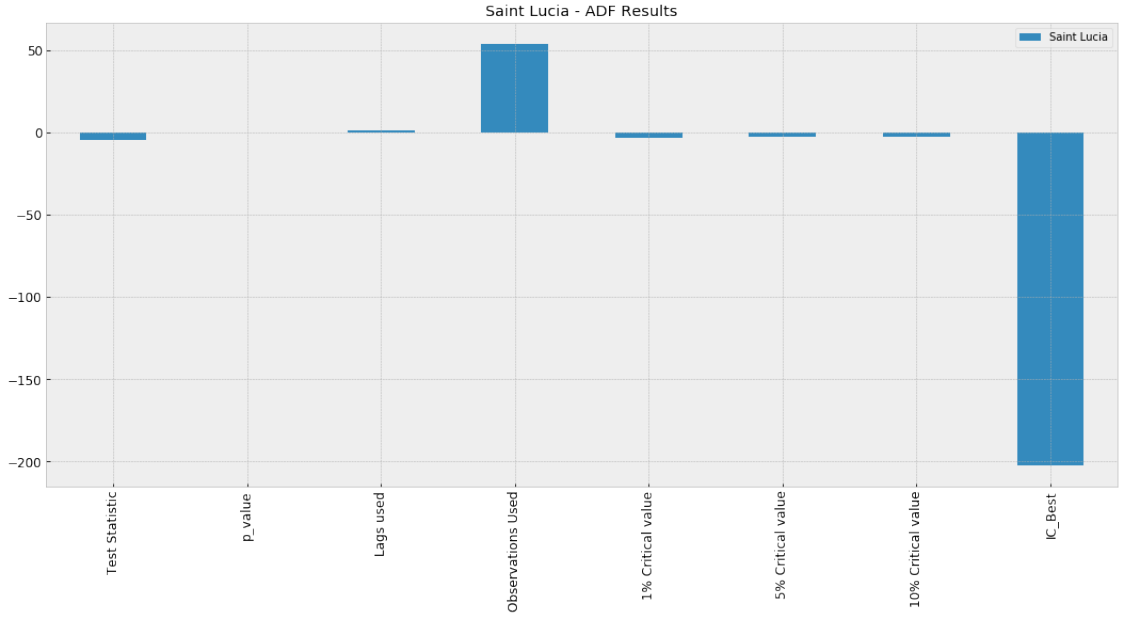

Russian Federation - ADF Results
Russian Federation
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

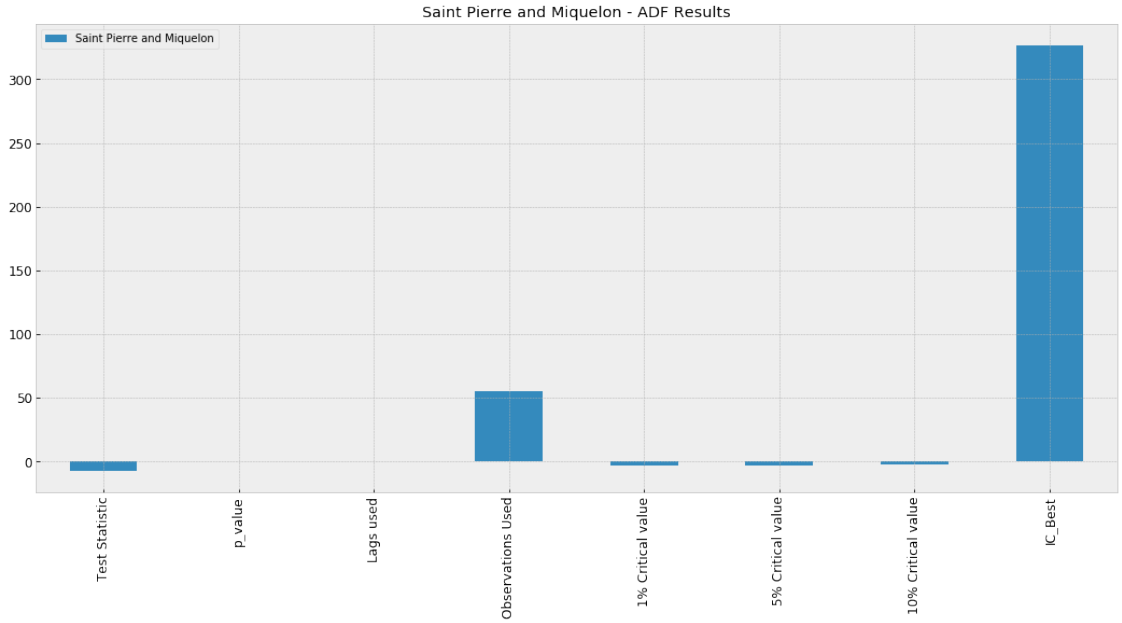

Rwanda - ADF Results
Rwanda
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

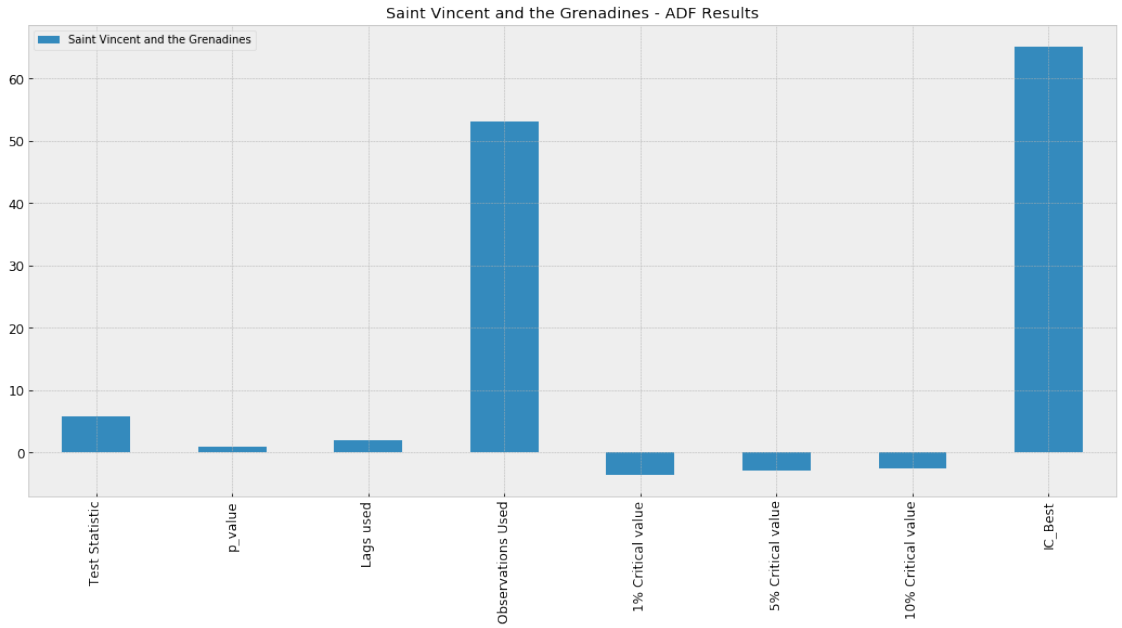

Réunion - ADF Results
Réunion
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

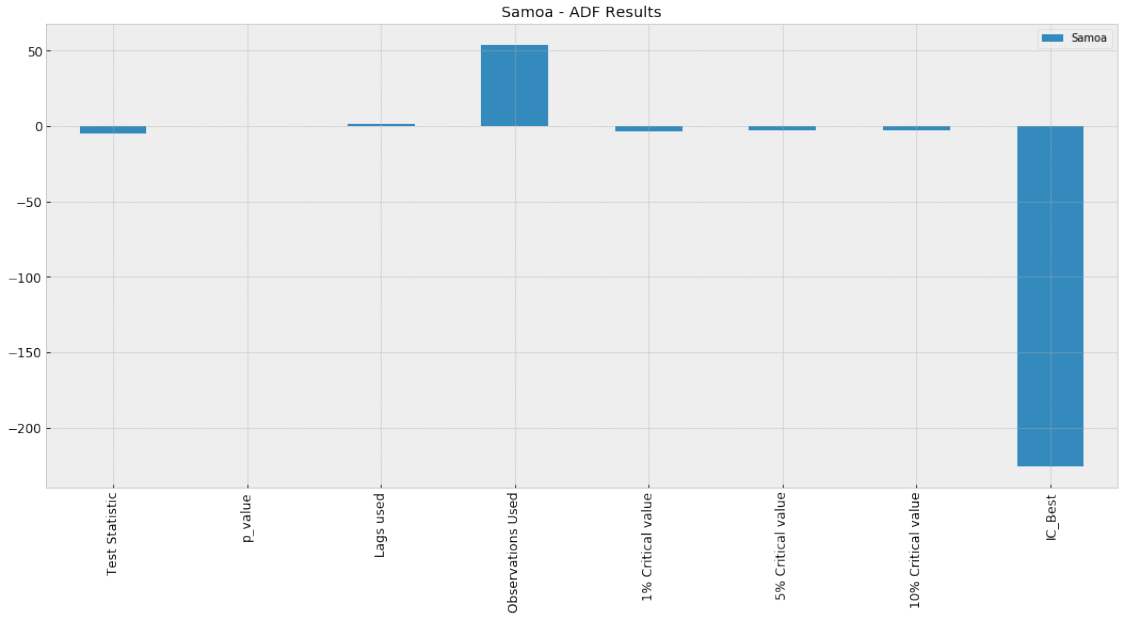

Saint Helena, Ascension and Tristan da Cunha - ADF Results
Saint Helena, Ascension and Tristan da Cunha
40
20
0
−20
−40
−60
−80
−100
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

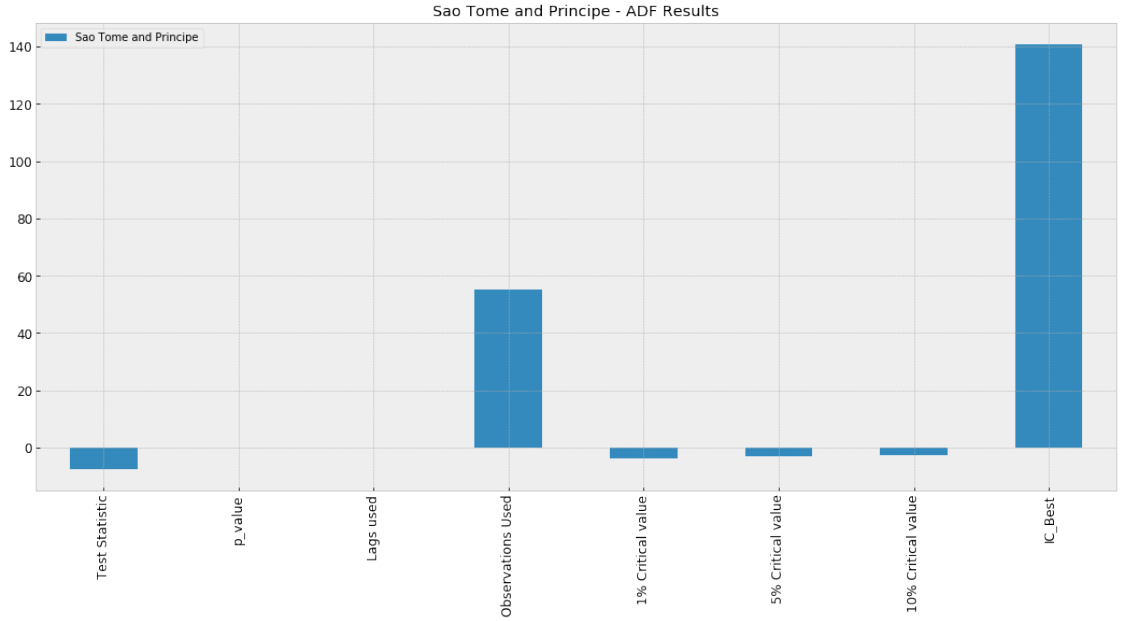

Saint Kitts and Nevis - ADF Results
Saint Kitts and Nevis
50
0
−50
−100
−150
−200
−250
−300
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

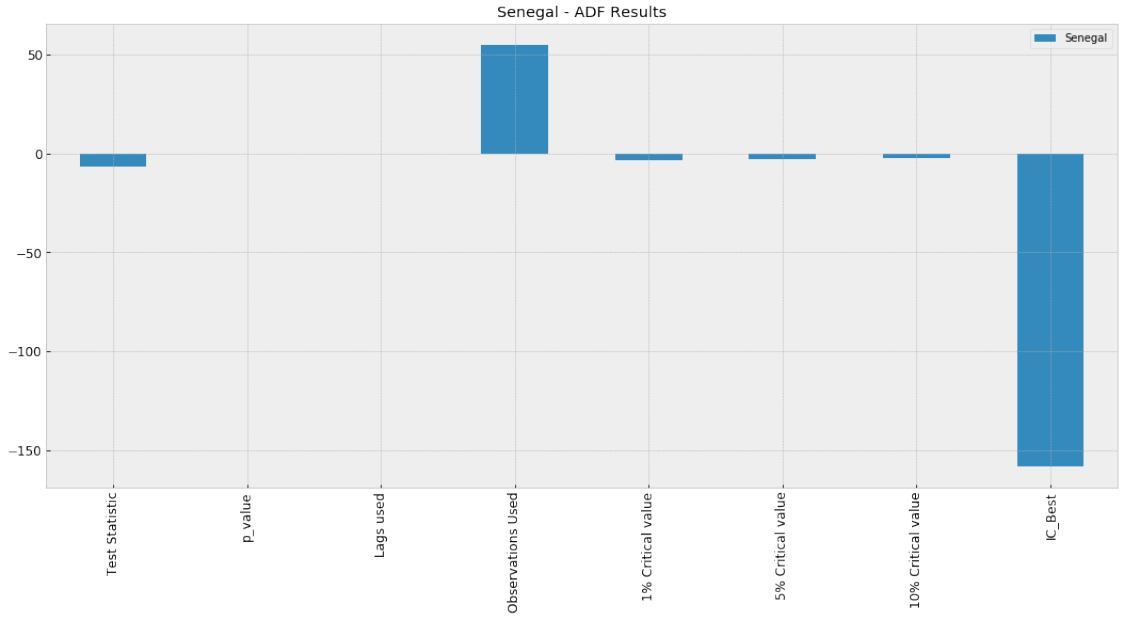

Saint Lucia - ADF Results
Saint Lucia
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

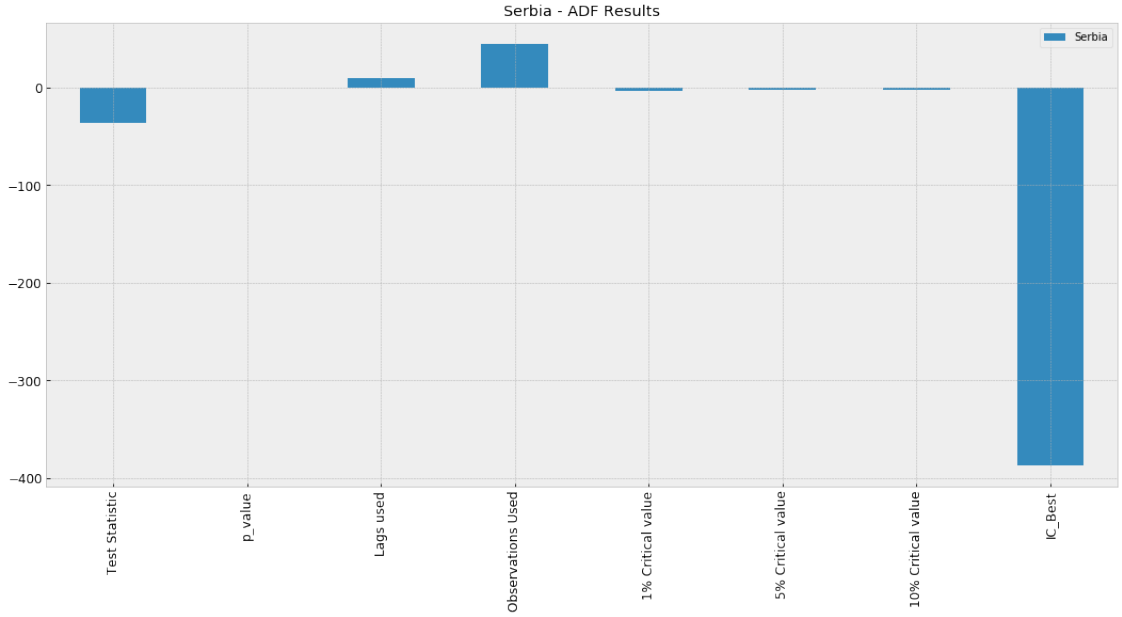

Saint Pierre and Miquelon - ADF Results
Saint Pierre and Miquelon
300
250
200
150
100
50
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

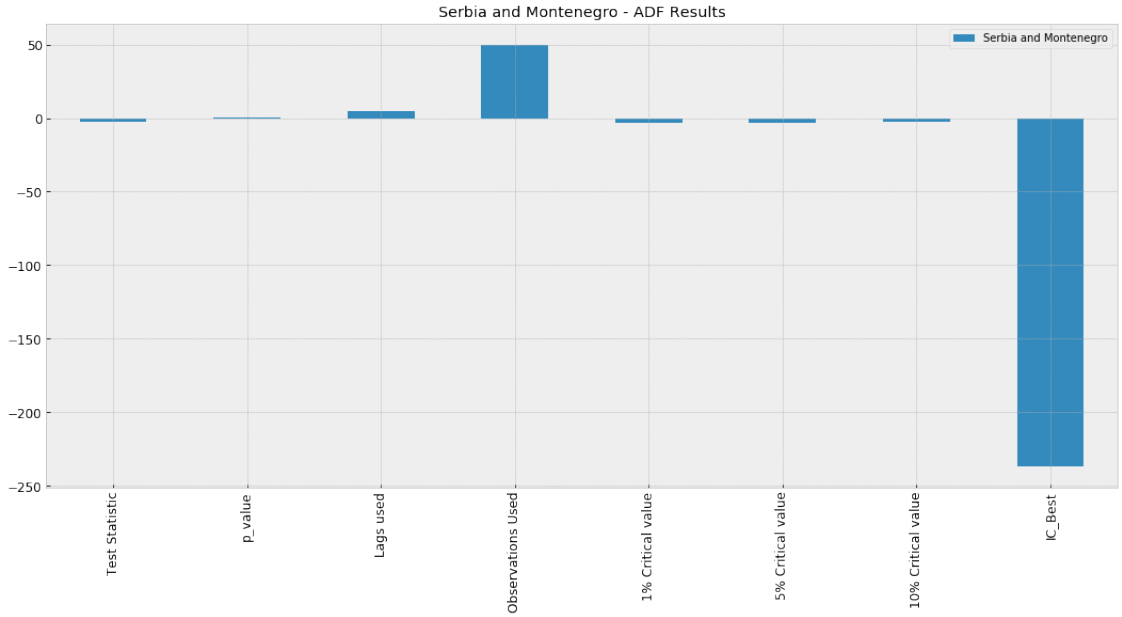

Saint Vincent and the Grenadines - ADF Results
Saint Vincent and the Grenadines
60
50
40
30
20
10
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

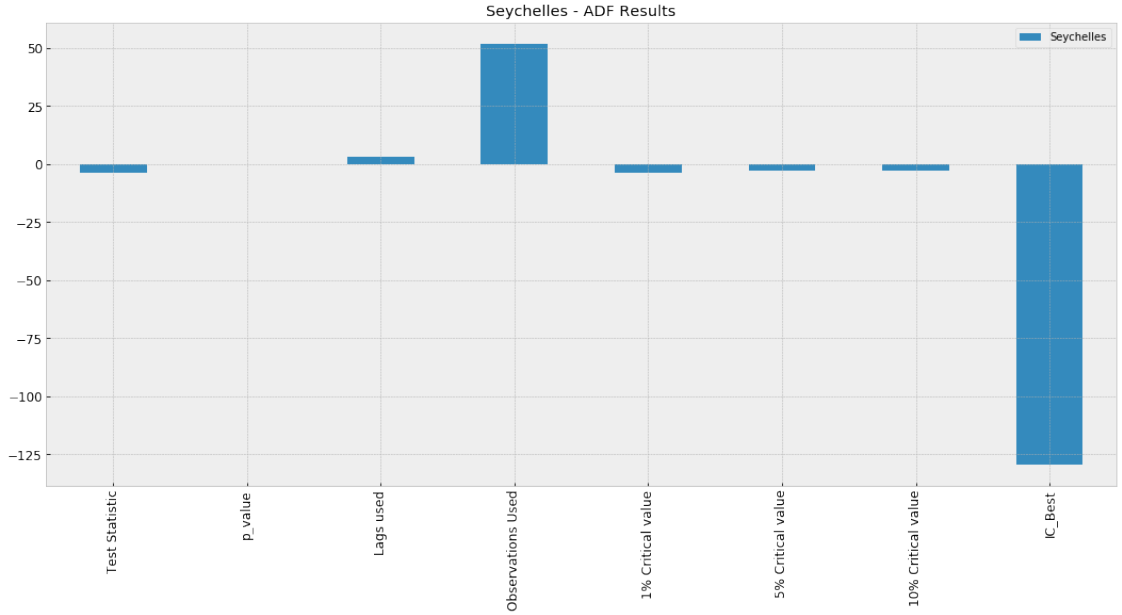

Samoa - ADF Results
Samoa
50
0
−50
−100
−150
−200
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

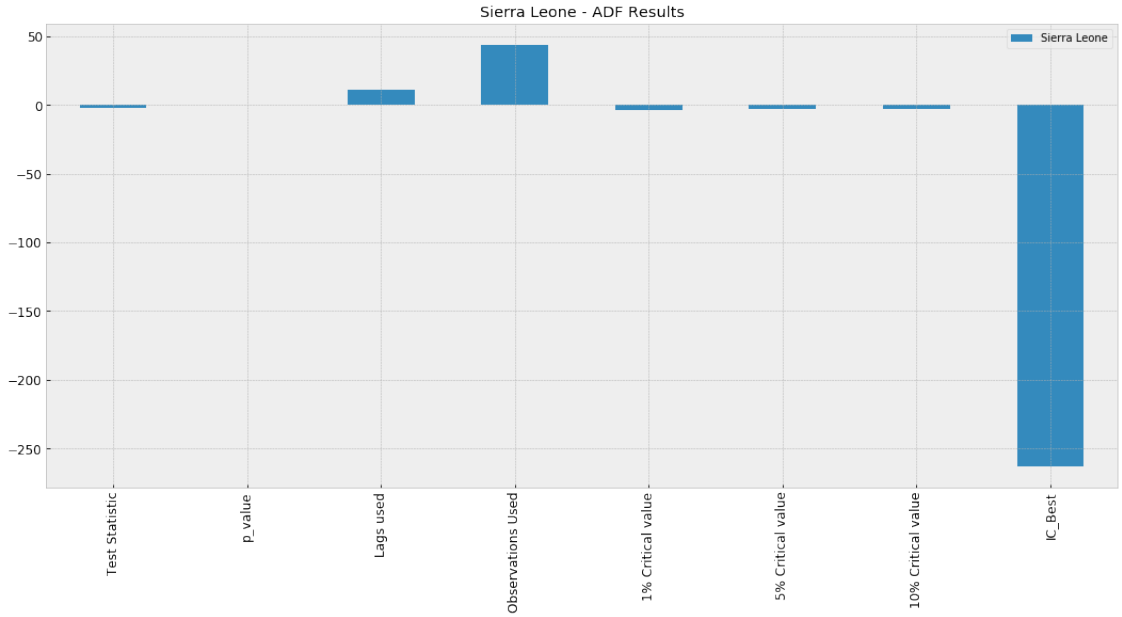

Sao Tome and Principe - ADF Results
Sao Tome and Principe
140
120
100
80
60
40
20
0
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

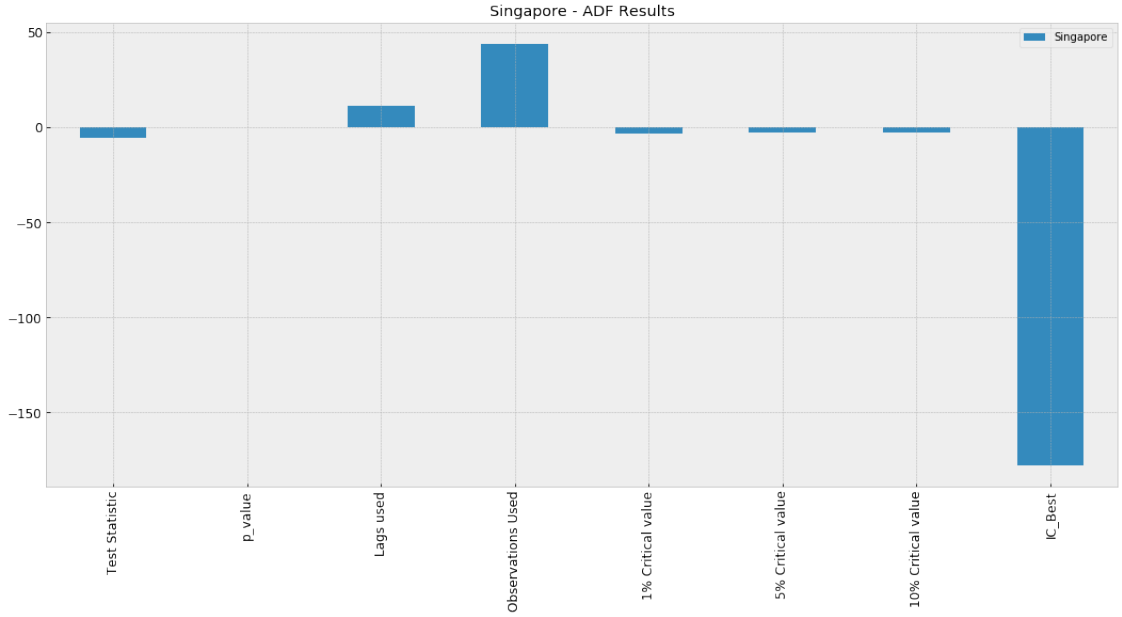

Senegal - ADF Results
Senegal
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

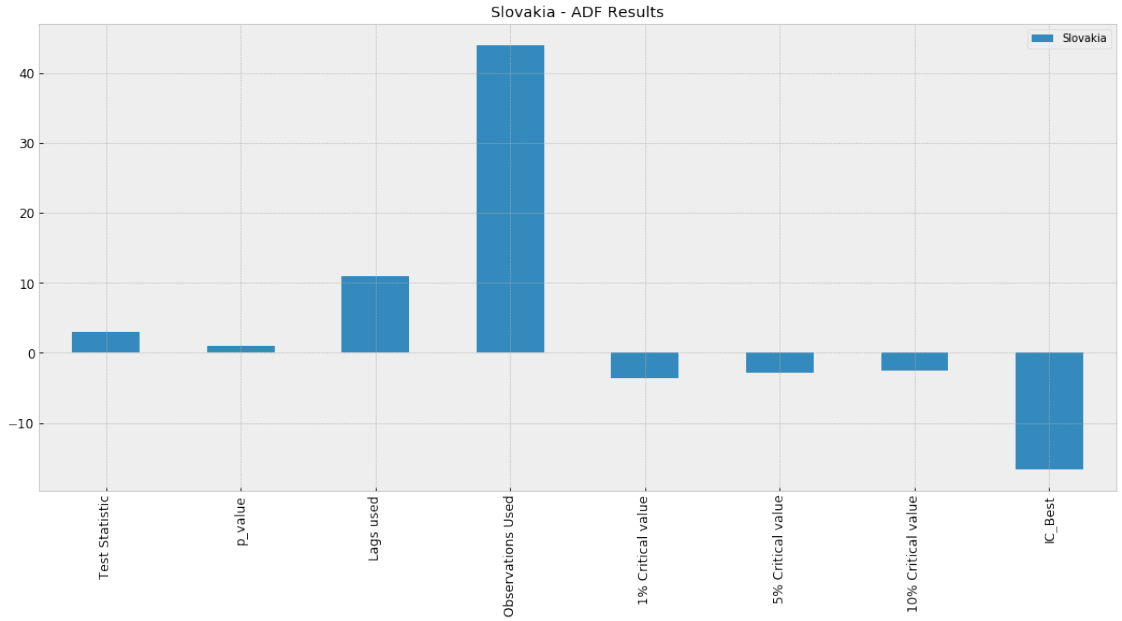

Serbia - ADF Results
Serbia
0
−100
−200
−300
−400
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

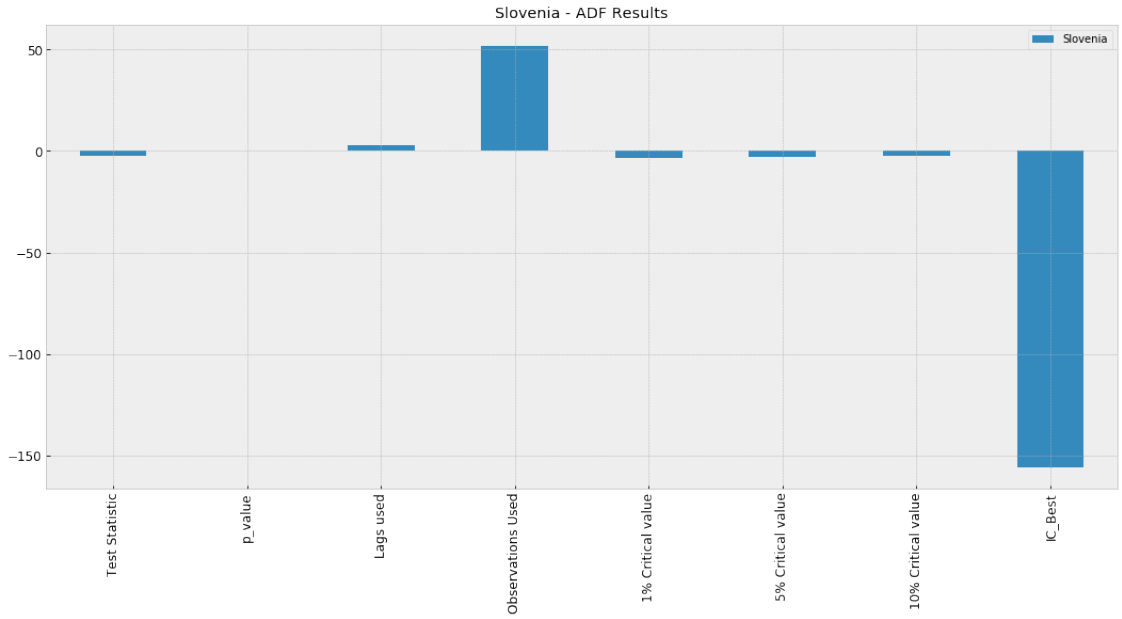

Serbia and Montenegro - ADF Results
Serbia and Montenegro
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

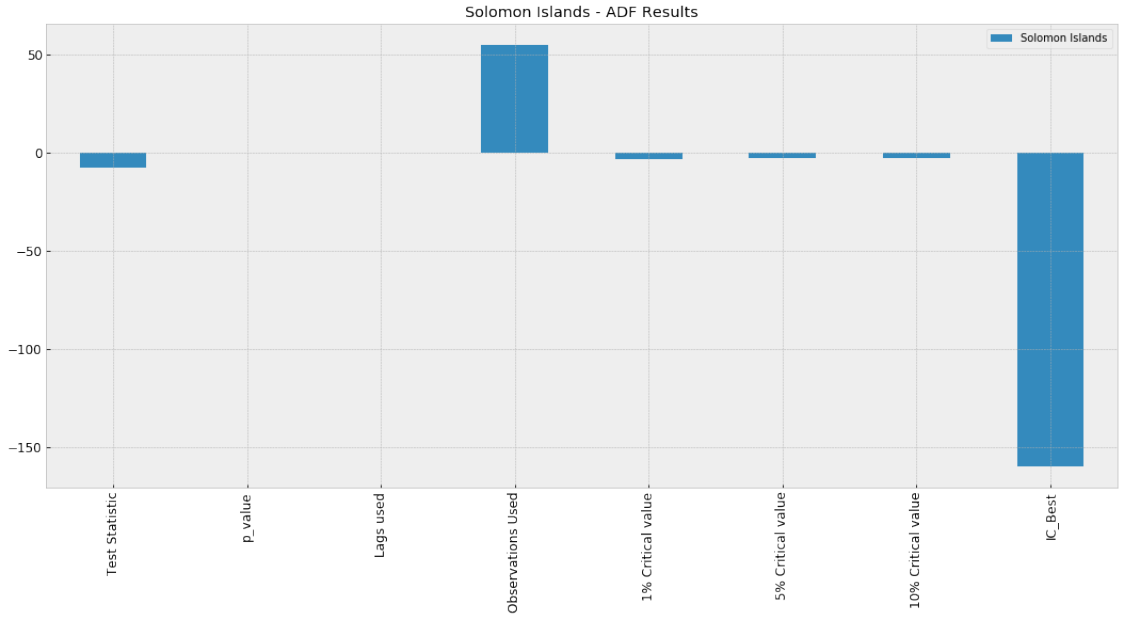

Seychelles - ADF Results
Seychelles
50
25
0
−25
−50
−75
−100
−125
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

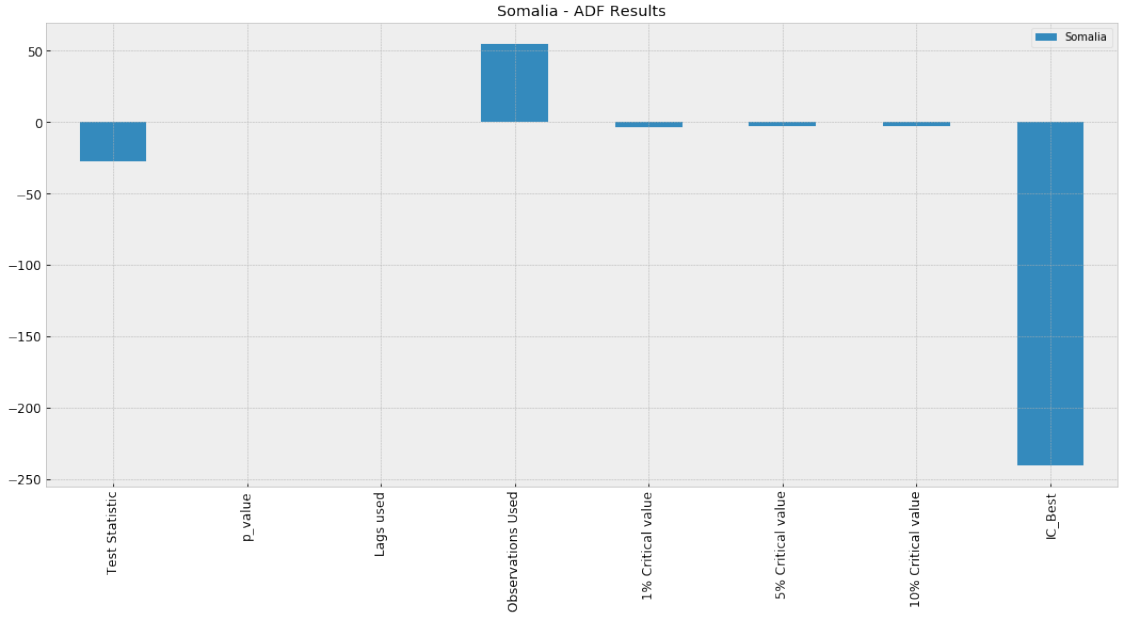

Sierra Leone - ADF Results
Sierra Leone
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

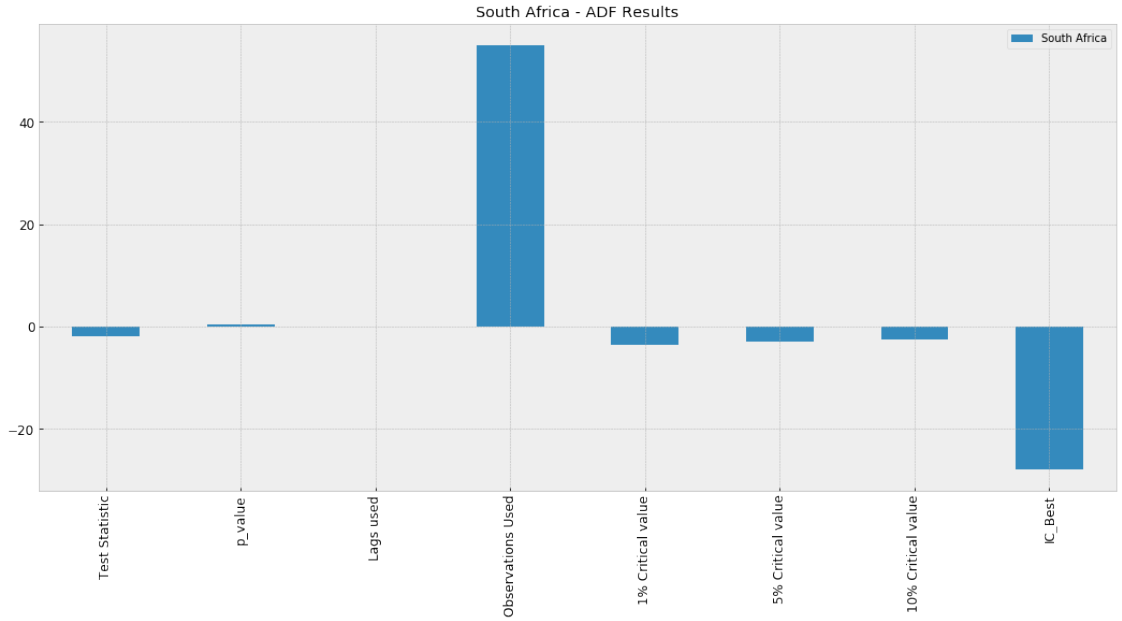

Singapore - ADF Results
Singapore
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

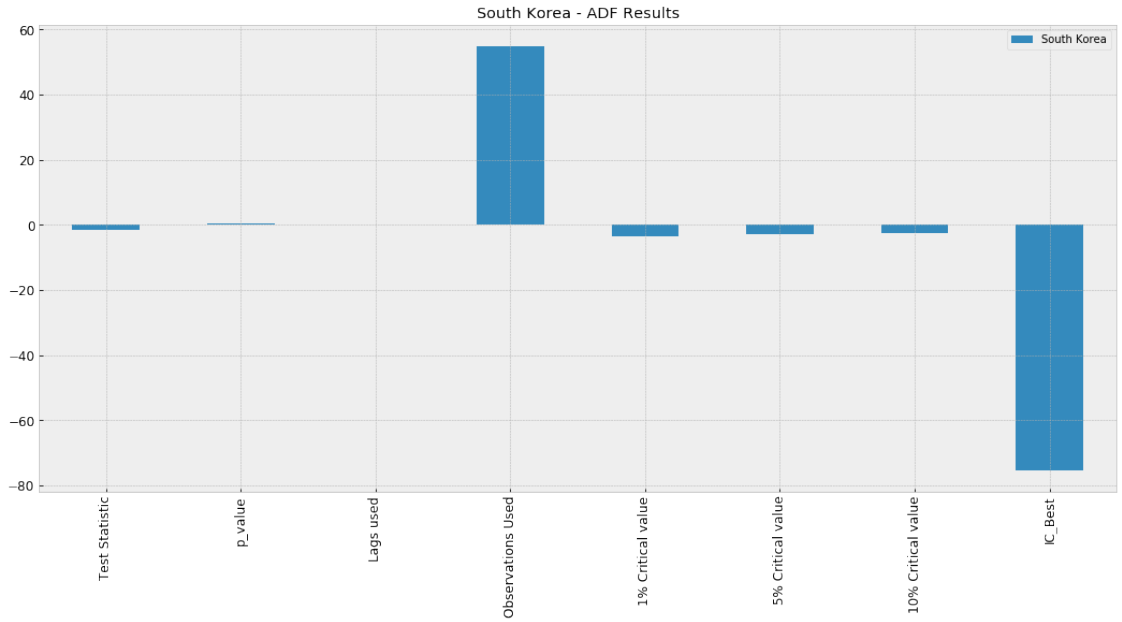

Slovakia - ADF Results
Slovakia
40
30
20
10
0
−10
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

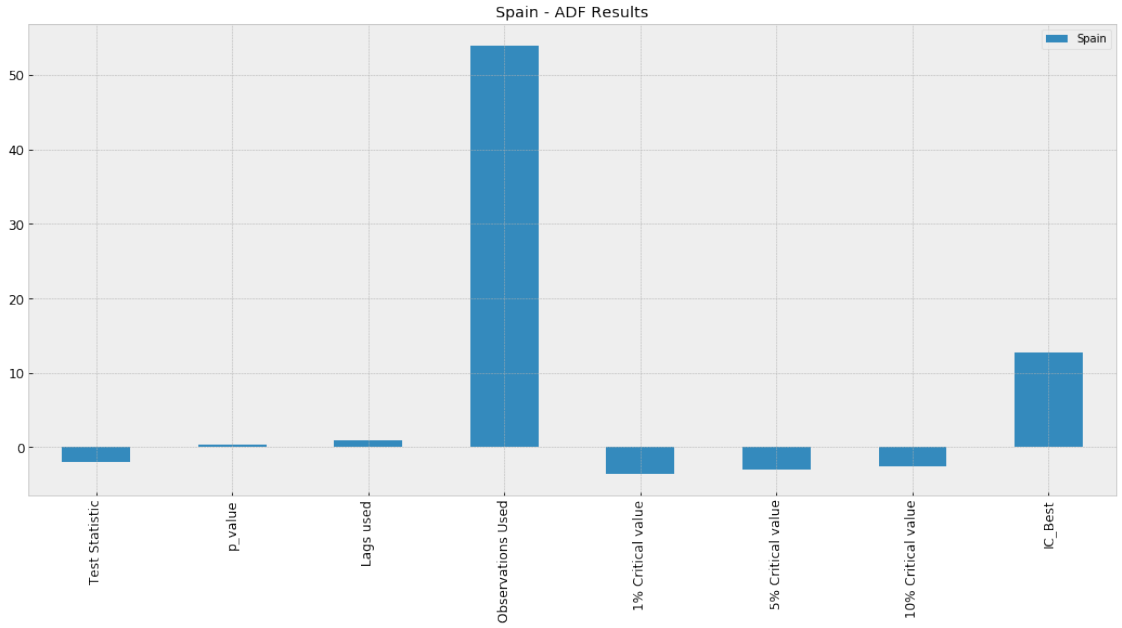

Slovenia - ADF Results
Slovenia
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

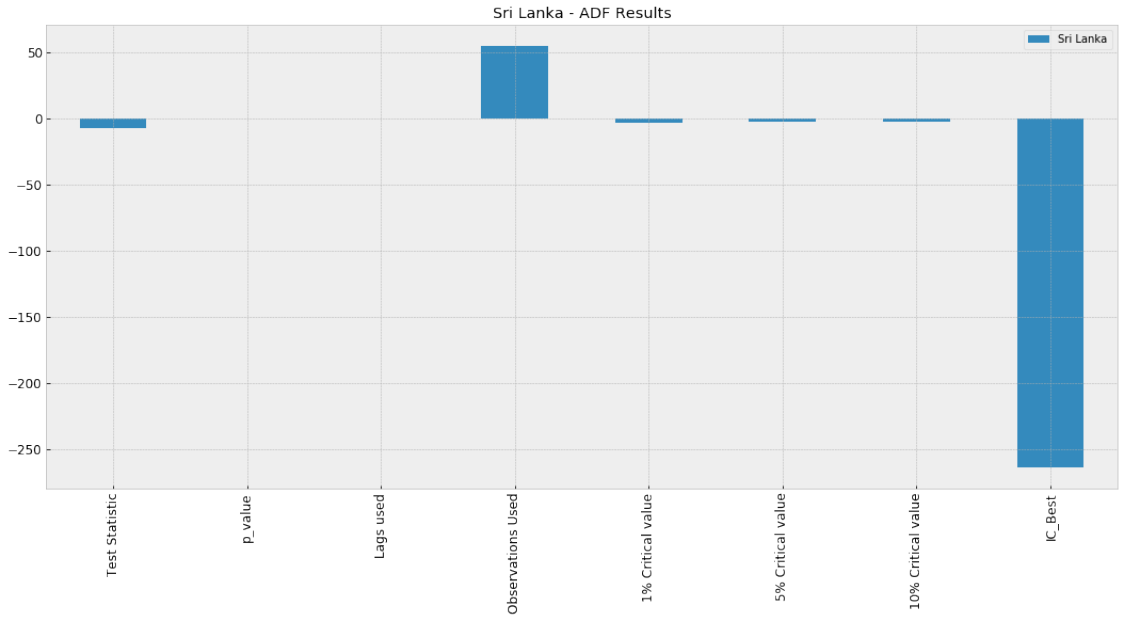

Solomon Islands - ADF Results
Solomon Islands
50
0
−50
−100
−150
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best


Somalia - ADF Results
Somalia
50
0
−50
−100
−150
−200
−250
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best


South Africa - ADF Results
South Africa
40
20
0
−20
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best


South Korea - ADF Results
South Korea
60
40
20
0
−20
−40
−60
−80
Test Statistic
p_value
Lags used
Observations Used
1% Critical value
5% Critical value
10% Critical value
IC_Best

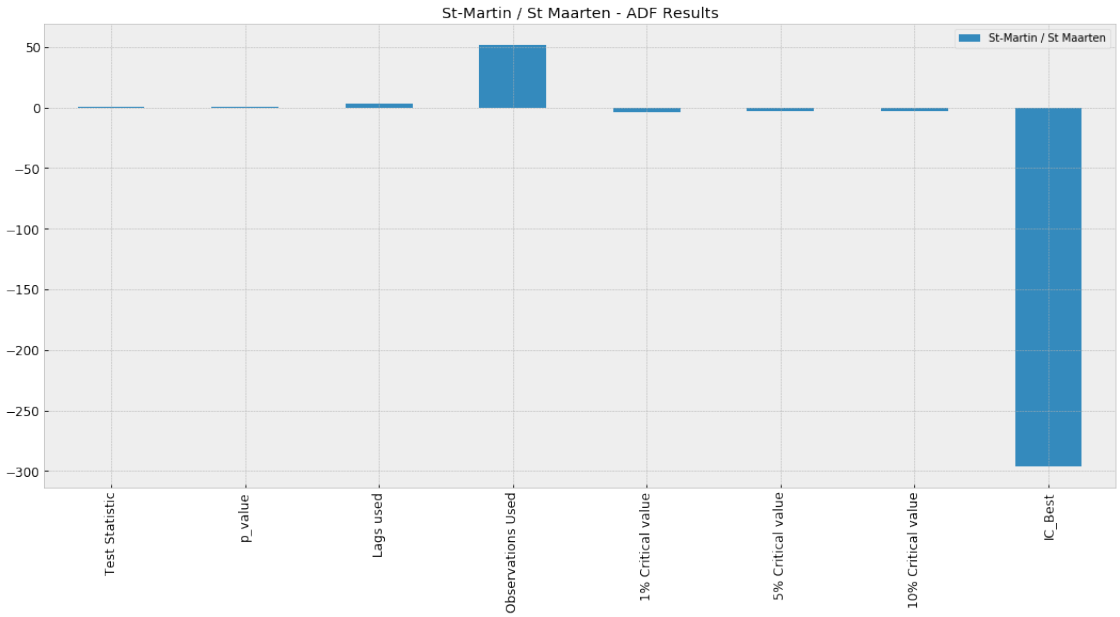





```
↪---------------------------------------------------------------------------

FileNotFoundError                         Traceback (most recent call␣
↪last)
```

```
      <ipython-input-117-9d60975f7a33> in <module>
        4     plt.legend(loc='best')
        5     plt.title('{} - ADF Results'.format(adf_test[country].name))
   ----> 6     plt.savefig(str('{}_ADF.png'.format(savefile)))
        7     plt.show()
        8


      ~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args,␣
↪**kwargs)
      687 def savefig(*args, **kwargs):
      688     fig = gcf()
   --> 689     res = fig.savefig(*args, **kwargs)
      690     fig.canvas.draw_idle()   # need this if 'transparent=True' to␣
↪reset colors
      691     return res


      ~\Anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self,␣
↪fname, frameon, transparent, **kwargs)
     2092             self.set_frameon(frameon)
     2093
   -> 2094         self.canvas.print_figure(fname, **kwargs)
     2095
     2096         if frameon:


      ~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in␣
↪print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format,␣
↪bbox_inches, **kwargs)
     2073                     orientation=orientation,
     2074                     bbox_inches_restore=_bbox_inches_restore,
   -> 2075                     **kwargs)
     2076             finally:
     2077                 if bbox_inches and restore_bbox:


      ~\Anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in␣
↪print_png(self, filename_or_obj, *args, **kwargs)
      519
      520         with cbook._setattr_cm(renderer, dpi=self.figure.dpi), \
   --> 521                 cbook.open_file_cm(filename_or_obj, "wb") as fh:
      522             _png.write_png(renderer._renderer, fh,
      523                            self.figure.dpi, metadata=metadata)
```

```
    ~\Anaconda3\lib\contextlib.py in __enter__(self)
    110         del self.args, self.kwds, self.func
    111         try:
--> 112             return next(self.gen)
    113         except StopIteration:
    114             raise RuntimeError("generator didn't yield") from None


    ~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in␣
↪open_file_cm(path_or_file, mode, encoding)
    405 def open_file_cm(path_or_file, mode="r", encoding=None):
    406     r"""Pass through file objects and context-manage `.PathLike`\s.
↪"""
--> 407     fh, opened = to_filehandle(path_or_file, mode, True, encoding)
    408     if opened:
    409         with fh:


    ~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in␣
↪to_filehandle(fname, flag, return_opened, encoding)
    390             fh = bz2.BZ2File(fname, flag)
    391         else:
--> 392             fh = open(fname, flag, encoding=encoding)
    393         opened = True
    394     elif hasattr(fname, 'seek'):


    FileNotFoundError: [Errno 2] No such file or directory: 'C:
↪\\Users\\16353\\Documents\\ThaW\\PhD\\time_series_trade\\images\\engine_parts\\rca_tests\\ad
↪/ St Maarten_ADF.png'
```



Automattion: Try this to automate p-value rejection for non-stationary time series

3.3 Making Time Series Stationary

There are 2 major reasons behind non-stationaruty of a TS:

1. **Trend** – varying mean over time. For eg, in this case we saw that on average, the number of passengers was growing over time.
2. **Seasonality** – variations at specific time-frames. eg people might have a tendency to buy cars in a particular month because of pay increment or festivals.

Other sources to check

3.3.1 Transformations

- We can apply transformation which penalize higher values more than smaller values
- These can be taking a log, square root, cube root

Log Scale Transformation

3.3.2 Techniques to remove Trend - Smoothing

- Smoothing is taking rolling averages over windows of time

[ ]:
```
ts_log = np.log(df_grps)
plt.plot(ts_log)
plt.show()
```

[ ]:
```
ts_log
```

Moving Average

- We take average of 'k' consecutive values depending on the frequency of time series.
- Here we can take the average over the past 1 year, i.e. last 12 values.
- A drawback in this particular approach is that the time-period has to be strictly defined.

[ ]:
```
moving_avg = ts_log.rolling(5).mean()
plt.plot(ts_log)
plt.plot(moving_avg)
plt.show()
```

[ ]:
```
ma = []
for col in ts_log:
    ts_log_moving_avg_diff = ts_log[col] - moving_avg[col]
    #ma.append(ts_log_moving_avg_diff)
ts_log_moving_avg_diff.head(10)
```

[ ]:
```
ts_log_moving_avg_diff.dropna(inplace=True)
test_stationarity(ts_log_moving_avg_diff)
```

Exponentially weighted moving average:

- To overcome the problem of choosing a defined window in moving average, we can use exponential weighted moving average
- We take a 'weighted moving average' where more recent values are given a higher weight.
- There can be many technique for assigning weights. A popular one is exponentially weighted moving average where weights are assigned to all the previous values with a decay factor.

[ ]:
```
expwighted_avg = ts_log.ewm(com=0.5).mean()
#df.ewm(com=0.5).mean()
plt.plot(ts_log)
plt.plot(expwighted_avg)
plt.show()
```

[ ]:
```
ts_log_ewma_diff = ts_log- expwighted_avg
ts_log_ewma_diff
```

3.3.3 Further Techniques to remove Seasonality and Trend

- The simple trend reduction techniques discussed before don't work in all cases, particularly the ones with high seasonality.

Differencing

- In this technique, we take the difference of the observation at a particular instant with that at the previous instant.
- First order differencing in Pandas

[ ]:
```
ts_log_diff = ts_log - ts_log.shift()
plt.plot(ts_log_diff)
plt.show()
```

[ ]:
```
ts_log_diff.dropna(inplace=True)
test_stationarity(ts_log_diff)
```

Decomposition

- In this approach, both trend and seasonality are modeled separately and the remaining part of the series is returned.

[ ]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(ts_log)

trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

plt.subplot(411)
plt.plot(ts_log, label='Original')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(trend, label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(seasonal,label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(residual, label='Residuals')
plt.legend(loc='best')
plt.tight_layout()
```

[ ]:
```
ts_log_decompose = residual
ts_log_decompose.dropna(inplace=True)
test_stationarity(ts_log_decompose)
```

[ ]:
```

```

[ ]:
```

```