1 

from dateutil.parser import parse

2

import matplotlib as mpl

3

import matplotlib.pyplot as plt

4

import seaborn as sns

5

import numpy as np

6

import pandas as pd

7

plt.rcParams.update({'figure.figsize': (10, 7), 'figure.dpi': 120})

8

9

# Import as Dataframe

10

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date'])

11

df.head()

**date value**

**0** 1991-07-01 3.526591

**1** 1991-08-01 3.180891

**2** 1991-09-01 3.252221

**3** 1991-10-01 3.611003

**4** 1991-11-01 3.565869

1 

ser = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date'], index\_col='date')

2

ser.head()

**value**

**date**

**1991-07-01** 3.526591

**1991-08-01** 3.180891

**1991-09-01** 3.252221

**1991-10-01** 3.611003

**1991-11-01** 3.565869

# 

1

# dataset source: https://github.com/rouseguy

2

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/MarketArrivals.csv')

3

df = df.loc[df.market=='MUMBAI', :]

4

df.head()

**market month year quantity priceMin priceMax priceMod state city date**

**6654** MUMBAI January 2004 267100 719 971 849 MS MUMBAI January-2004

**6655** MUMBAI January 2005 275845 261 513 387 MS MUMBAI January-2005

**6656** MUMBAI January 2006 228000 315 488 402 MS MUMBAI January-2006

**6657** MUMBAI January 2007 205200 866 1136 997 MS MUMBAI January-2007

**6658** MUMBAI January 2008 267550 348 550 448 MS MUMBAI January-2008

1 

#

# Time series data source: fpp pacakge in R.

2

import matplotlib.pyplot as plt

3

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date'], index\_col='date')

4

5

# Draw Plot

6

def plot\_df(df, x, y, title="", xlabel='Date', ylabel='Value', dpi=100):

7

    plt.figure(figsize=(16,5), dpi=dpi)

8

    plt.plot(x, y, color='tab:red')

9

    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)

10

    plt.show()

11

12

plot\_df(df, x=df.index, y=df.value, title='Monthly anti-diabetic drug sales in Australia from 1992 to 2008.')

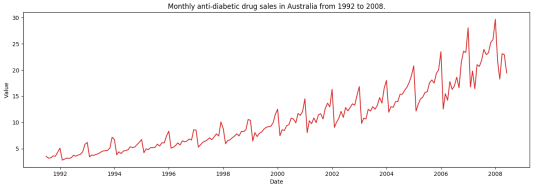
/usr/local/lib/python3.6/dist-packages/pandas/plotting/\_converter.py:129: FutureWarning: Using an implicitly registered datetime converter for a matplotlib plotting method. The converter was registere

To register the converters:

>>> from pandas.plotting import register\_matplotlib\_converters

>>> register\_matplotlib\_converters()

warnings.warn(msg, FutureWarning)



#

1

# Import data

2

df = pd.read\_csv('/content/AirPassengers.csv', parse\_dates=['Month'])

3

x = df['Month'].values

4

y1 = df['#Passengers']

5

6

# Plot

7

fig, ax = plt.subplots(1, 1, figsize=(16,5), dpi= 120)

8

plt.fill\_between(x, y1=y1, y2=-y1, alpha=0.5, linewidth=2, color='seagreen')

9

plt.ylim(-800, 800)

10

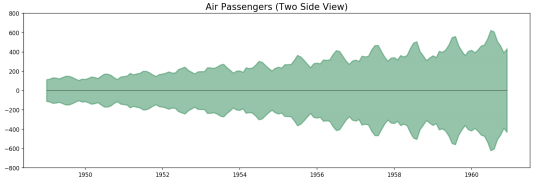
plt.title('Air Passengers (Two Side View)', fontsize=16)

11

plt.hlines(y=0, xmin=np.min(df.Month), xmax=np.max(df.Month), linewidth=.5)

12

plt.show()

1 

# Import Data

#

2

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date'], index\_col='date')

3

df.reset\_index(inplace=True)

4

5

# Prepare data

6

df['year'] = [d.year for d in df.date]

7

df['month'] = [d.strftime('%b') for d in df.date]

8

years = df['year'].unique()

9

10

# Prep Colors

11

np.random.seed(100)

12

mycolors = np.random.choice(list(mpl.colors.XKCD\_COLORS.keys()), len(years), replace=False)

13

14

# Draw Plot

15

plt.figure(figsize=(16,12), dpi= 80)

16

for i, y in enumerate(years):

17

    if i > 0:

18

        plt.plot('month', 'value', data=df.loc[df.year==y, :], color=mycolors[i], label=y)

19

        plt.text(df.loc[df.year==y, :].shape[0]-.9, df.loc[df.year==y, 'value'][-1:].values[0], y, fontsize=12, color=mycolors[i]) 20

21

# Decoration

22

plt.gca().set(xlim=(-0.3, 11), ylim=(2, 30), ylabel='$Drug Sales$', xlabel='$Month$')

23

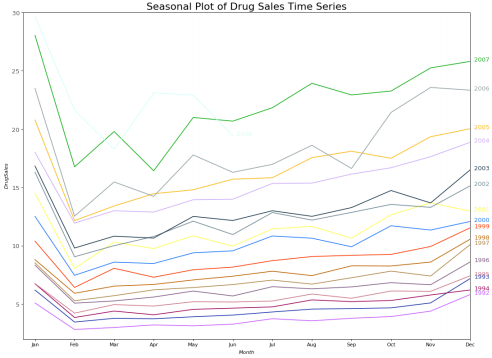
plt.yticks(fontsize=12, alpha=.7)

24

plt.title("Seasonal Plot of Drug Sales Time Series", fontsize=20)

25

plt.show()



1

# Import Data

#

2

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date'], index\_col='date')

3

df.reset\_index(inplace=True)

4

5

# Prepare data

6

df['year'] = [d.year for d in df.date]

7

df['month'] = [d.strftime('%b') for d in df.date]

8

years = df['year'].unique()

9

10

# Draw Plot

11

fig, axes = plt.subplots(1, 2, figsize=(20,7), dpi= 80)

12

sns.boxplot(x='year', y='value', data=df, ax=axes[0])

13

sns.boxplot(x='month', y='value', data=df.loc[~df.year.isin([1991, 2008]), :])

14

15

# Set Title

16

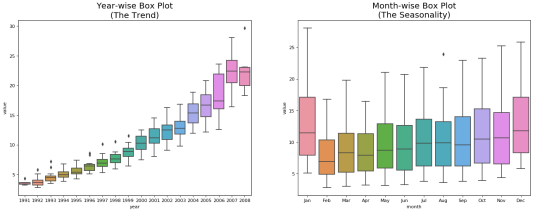
axes[0].set\_title('Year-wise Box Plot\n(The Trend)', fontsize=18);

17

axes[1].set\_title('Month-wise Box Plot\n(The Seasonality)', fontsize=18)

18

plt.show()



1

fig, axes = plt.subplots(1,3, figsize=(20,4), dpi=100)

fi

2

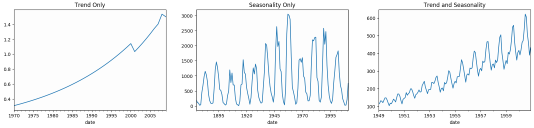
pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/guinearice.csv', parse\_dates=['date'], index\_col='date').plot(title='Trend Only', legend=Fal 3

4

pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/sunspotarea.csv', parse\_dates=['date'], index\_col='date').plot(title='Seasonality Only', leg 5

6 pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/AirPassengers.csv', parse\_dates=['date'], index\_col='date').plot(title='Trend and Seasonalit

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fb130d222b0>



1 

from statsmodels.tsa.seasonal import seasonal\_decompose

2

from dateutil.parser import parse

3

4

# Import Data

5

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date'], index\_col='date') 6

7

# Multiplicative Decomposition

8

result\_mul = seasonal\_decompose(df['value'], model='multiplicative', extrapolate\_trend='freq')

9

10

# Additive Decomposition

11

result\_add = seasonal\_decompose(df['value'], model='additive', extrapolate\_trend='freq')

12

13

# Plot

14

plt.rcParams.update({'figure.figsize': (10,10)})

15

result\_mul.plot().suptitle('Multiplicative Decompose', fontsize=22)

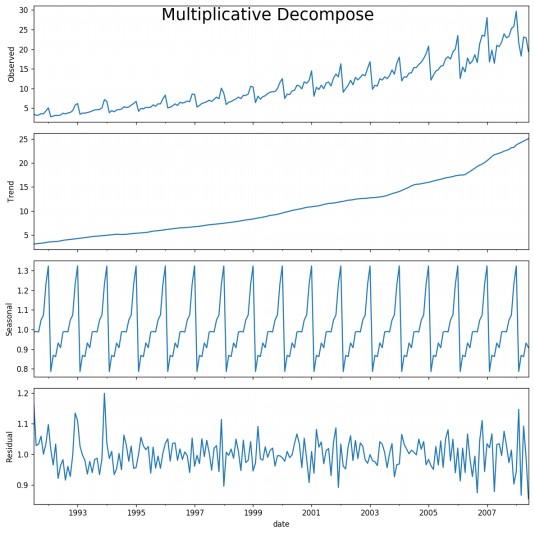
16

result\_add.plot().suptitle('Additive Decompose', fontsize=22)

17

plt.show()

18

# 

1

# Extract the Components ----

2

# Actual Values = Product of (Seasonal \* Trend \* Resid)

3

df\_reconstructed = pd.concat([result\_mul.seasonal, result\_mul.trend, result\_mul.resid, result\_mul.observed], axis=1)

4

df\_reconstructed.columns = ['seas', 'trend', 'resid', 'actual\_values']

5

df\_reconstructed.head()

**seas trend resid actual\_values**

**date**

**1991-07-01** 0.987845 3.060085 1.166629 3.526591

**1991-08-01** 0.990481 3.124765 1.027745 3.180891

**1991-09-01** 0.987476 3.189445 1.032615 3.252221

**1991-10-01** 1.048329 3.254125 1.058513 3.611003

**1991-11-01** 1.074527 3.318805 0.999923 3.565869

1 

from statsmodels.tsa.stattools import adfuller, kpss

2

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date'])

3

4

# ADF Test

5

result = adfuller(df.value.values, autolag='AIC')

6

print(f'ADF Statistic: {result[0]}')

7

print(f'p-value: {result[1]}')

8

for key, value in result[4].items():

9 

    print('Critial Values:')

10

    print(f'   {key}, {value}')

11

12

# KPSS Test

13

result = kpss(df.value.values, regression='c')

14

print('\nKPSS Statistic: %f' % result[0])

15

print('p-value: %f' % result[1])

16

for key, value in result[3].items():

17

    print('Critial Values:')

18

    print(f'   {key}, {value}')

ADF Statistic: 3.14518568930674

p-value: 1.0

Critial Values:

1%, -3.465620397124192

Critial Values:

5%, -2.8770397560752436

Critial Values:

10%, -2.5750324547306476

KPSS Statistic: 1.313675

p-value: 0.010000

Critial Values:

10%, 0.347

Critial Values:

5%, 0.463

Critial Values:

2.5%, 0.574

Critial Values:

1%, 0.739

/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/stattools.py:1685: FutureWarning: The behavior of using lags=None will change in the next release. Currently lags=None is the same as lags='legac warn(msg, FutureWarning)

/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/stattools.py:1708: InterpolationWarning: p-value is smaller than the indicated p-value

warn("p-value is smaller than the indicated p-value", InterpolationWarning)

r

1

randvals = np.random.randn(1000)

2

pd.Series(randvals).plot(title='Random White Noise', color='k')

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fb131003c18>

1 

# Using scipy: Subtract the line of best fit

#

2

from scipy import signal

3

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date']) 4

detrended = signal.detrend(df.value.values)

5

plt.plot(detrended)

6

plt.title('Drug Sales detrended by subtracting the least squares fit', fontsize=16)

7

Text(0.5, 1.0, 'Drug Sales detrended by subtracting the least squares fit')



1

# Using statmodels: Subtracting the Trend Component.

#

2

from statsmodels.tsa.seasonal import seasonal\_decompose

3

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date'], index\_col='date') 4

result\_mul = seasonal\_decompose(df['value'], model='multiplicative', extrapolate\_trend='freq')

5

detrended = df.value.values - result\_mul.trend

6

plt.plot(detrended)

7

plt.title('Drug Sales detrended by subtracting the trend component', fontsize=16)

Text(0.5, 1.0, 'Drug Sales detrended by subtracting the trend component')

# 

1

# Subtracting the Trend Component.

2

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date'], index\_col='date') 3

4

# Time Series Decomposition

5

result\_mul = seasonal\_decompose(df['value'], model='multiplicative', extrapolate\_trend='freq')

6

7

# Deseasonalize

8

deseasonalized = df.value.values / result\_mul.seasonal

9

10

# Plot

11

plt.plot(deseasonalized)

12

plt.title('Drug Sales Deseasonalized', fontsize=16)

13

plt.plot()

[]



1

from pandas.plotting import autocorrelation\_plot

2

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv')

3

4

# Draw Plot

5

plt.rcParams.update({'figure.figsize':(9,5), 'figure.dpi':120})

6

autocorrelation\_plot(df.value.tolist())

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fb12ddefba8>



1

from statsmodels.tsa.stattools import acf, pacf

2

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

3

4

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv')

5

6

# Calculate ACF and PACF upto 50 lags

7

# acf\_50 = acf(df.value, nlags=50)

8

# pacf\_50 = pacf(df.value, nlags=50)

9

10

# Draw Plot

11

fig, axes = plt.subplots(1,2,figsize=(16,3), dpi= 100)

12

plot\_acf(df.value.tolist(), lags=50, ax=axes[0])

13

plot\_pacf(df.value.tolist(), lags=50, ax=axes[1])



1

from pandas.plotting import lag\_plot

2

plt.rcParams.update({'ytick.left' : False, 'axes.titlepad':10})

3

4

# Import

5

ss = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/sunspotarea.csv')

6

a10 = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv')

7

8

# Plot

9

fig, axes = plt.subplots(1, 4, figsize=(10,3), sharex=True, sharey=True, dpi=100)

10

for i, ax in enumerate(axes.flatten()[:4]):

11

    lag\_plot(ss.value, lag=i+1, ax=ax, c='firebrick')

12

    ax.set\_title('Lag ' + str(i+1))

13

14

fig.suptitle('Lag Plots of Sun Spots Area \n(Points get wide and scattered with increasing lag -> lesser correlation)\n', y=1.15)     15

16

fig, axes = plt.subplots(1, 4, figsize=(10,3), sharex=True, sharey=True, dpi=100)

17

for i, ax in enumerate(axes.flatten()[:4]):

18

    lag\_plot(a10.value, lag=i+1, ax=ax, c='firebrick')

19

    ax.set\_title('Lag ' + str(i+1))

20

21

fig.suptitle('Lag Plots of Drug Sales', y=1.05)

22

plt.show()



1

# https://en.wikipedia.org/wiki/Approximate\_entropy

#

2

ss = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/sunspotarea.csv')

3

a10 = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv')

4

rand\_small = np.random.randint(0, 100, size=36)

5

rand\_big = np.random.randint(0, 100, size=136)

6

7

def ApEn(U, m, r):

8

    """Compute Aproximate entropy"""

9

    def \_maxdist(x\_i, x\_j):

10

        return max([abs(ua - va) for ua, va in zip(x\_i, x\_j)])

11

12

    def \_phi(m):

13

        x = [[U[j] for j in range(i, i + m - 1 + 1)] for i in range(N - m + 1)]

14

        C = [len([1 for x\_j in x if \_maxdist(x\_i, x\_j) <= r]) / (N - m + 1.0) for x\_i in x]

15

        return (N - m + 1.0)\*\*(-1) \* sum(np.log(C))

16

17

    N = len(U)

18

    return abs(\_phi(m+1) - \_phi(m))

19

20

print(ApEn(ss.value, m=2, r=0.2\*np.std(ss.value)))     # 0.651

21

print(ApEn(a10.value, m=2, r=0.2\*np.std(a10.value)))   # 0.537

22

print(ApEn(rand\_small, m=2, r=0.2\*np.std(rand\_small))) # 0.143

23

print(ApEn(rand\_big, m=2, r=0.2\*np.std(rand\_big)))     # 0.716

0.6514704970333534

0.5374775224973489

0.0898376940798844

0.6725953850207098

#

1

# https://en.wikipedia.org/wiki/Sample\_entropy

2

def SampEn(U, m, r):

3

    """Compute Sample entropy"""

4

    def \_maxdist(x\_i, x\_j):

5

        return max([abs(ua - va) for ua, va in zip(x\_i, x\_j)])

6

7

    def \_phi(m):

8

        x = [[U[j] for j in range(i, i + m - 1 + 1)] for i in range(N - m + 1)]

9

C [l ([1 f j i (l ( )) if i ! j d di t( [i] [j]) < ]) f i i (l ( ))]

9

        C = [len([1 for j in range(len(x)) if i != j and \_maxdist(x[i], x[j]) <= r]) for i in range(len(x))] 10

        return sum(C)

11

12

    N = len(U)

13

    return -np.log(\_phi(m+1) / \_phi(m))

14

15

print(SampEn(ss.value, m=2, r=0.2\*np.std(ss.value)))      # 0.78

16

print(SampEn(a10.value, m=2, r=0.2\*np.std(a10.value)))    # 0.41

17

print(SampEn(rand\_small, m=2, r=0.2\*np.std(rand\_small)))  # 1.79

18

print(SampEn(rand\_big, m=2, r=0.2\*np.std(rand\_big)))      # 2.42

0.7853311366380039

0.41887013457621214

inf

2.2721258855093374

/usr/local/lib/python3.6/dist-packages/ipykernel\_launcher.py:12: RuntimeWarning: divide by zero encountered in log if sys.path[0] == '':

1 

from statsmodels.tsa.stattools import grangercausalitytests

2

df = pd.read\_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse\_dates=['date']) 3

df['month'] = df.date.dt.month

4

grangercausalitytests(df[['value', 'month']], maxlag=2)

Granger Causality

number of lags (no zero) 1

ssr based F test: F=54.7797 , p=0.0000 , df\_denom=200, df\_num=1

ssr based chi2 test: chi2=55.6014 , p=0.0000 , df=1

likelihood ratio test: chi2=49.1426 , p=0.0000 , df=1

parameter F test: F=54.7797 , p=0.0000 , df\_denom=200, df\_num=1

Granger Causality

number of lags (no zero) 2

ssr based F test: F=162.6989, p=0.0000 , df\_denom=197, df\_num=2

ssr based chi2 test: chi2=333.6567, p=0.0000 , df=2

likelihood ratio test: chi2=196.9956, p=0.0000 , df=2

parameter F test: F=162.6989, p=0.0000 , df\_denom=197, df\_num=2

{1: ({'lrtest': (49.14260233004984, 2.38014300604565e-12, 1),

'params\_ftest': (54.77967483557335, 3.661425871353419e-12, 200.0, 1.0),

'ssr\_chi2test': (55.6013699581072, 8.876175235021508e-14, 1),

'ssr\_ftest': (54.7796748355736, 3.661425871353102e-12, 200.0, 1)},

[<statsmodels.regression.linear\_model.RegressionResultsWrapper at 0x7fb12bcd05c0>,

<statsmodels.regression.linear\_model.RegressionResultsWrapper at 0x7fb12bcd06a0>,

array([[0., 1., 0.]])]),

2: ({'lrtest': (196.99559277182186, 1.6709003499116746e-43, 2),

'params\_ftest': (162.69891799873227, 1.9133235086857535e-42, 197.0, 2.0),

'ssr\_chi2test': (333.65666432227346, 3.5267600881280646e-73, 2),

'ssr\_ftest': (162.69891799873236, 1.9133235086857257e-42, 197.0, 2)},

[<statsmodels.regression.linear\_model.RegressionResultsWrapper at 0x7fb12bdf22b0>,

<statsmodels.regression.linear\_model.RegressionResultsWrapper at 0x7fb12bdf2eb8>,

array([[0., 0., 1., 0., 0.],

[0., 0., 0., 1., 0.]])])}