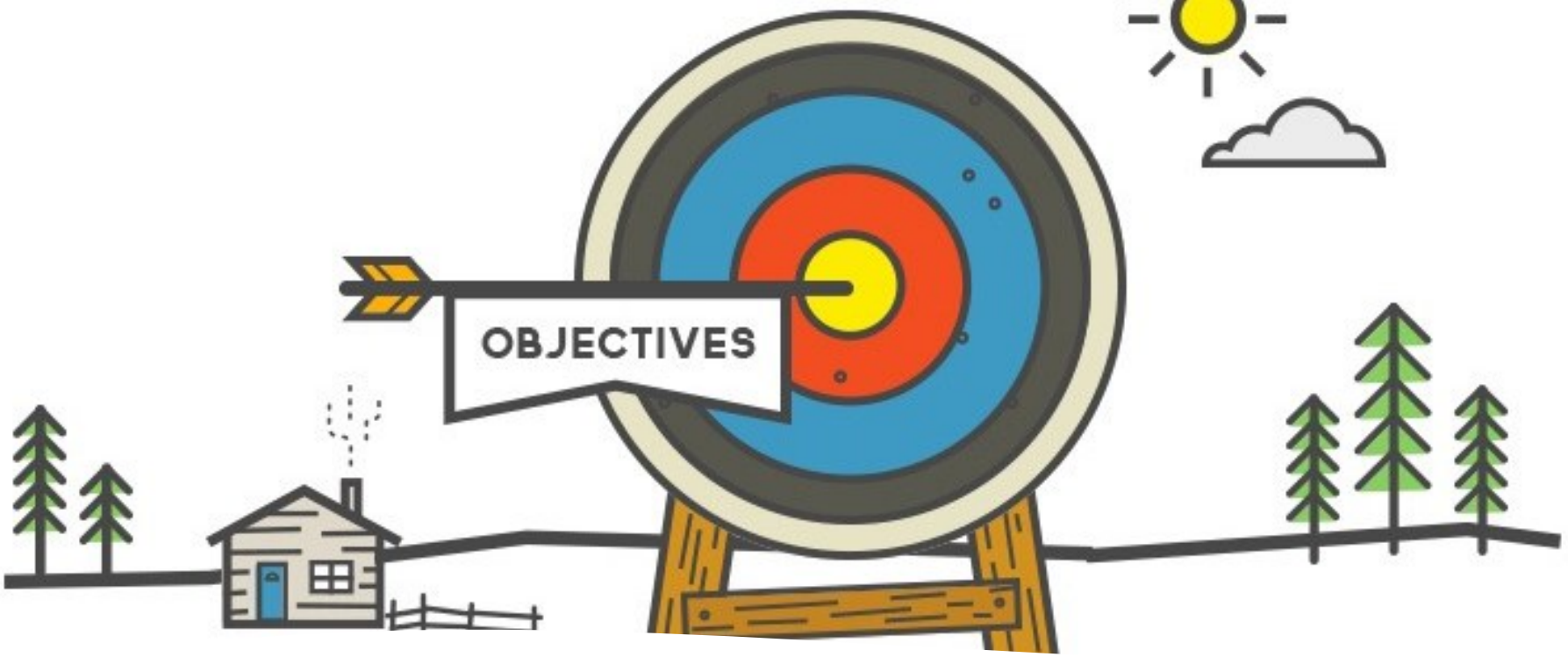
Lisbon School of Economics & Management
Universidade de Lisboa

U LISBOA
UNIVERSIDADE DE LISBOA

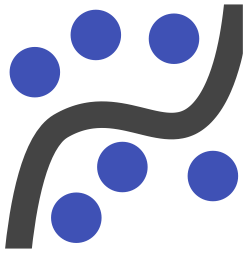
Time Series

Prof. Carlos J. Costa, PhD



Learning Goals

- Use Python libraries to decompose time series
- Use Python libraries to analyse series
- Use Python libraries to predict

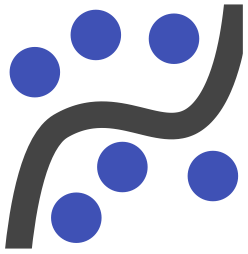


statsmodels

https://www.statsmodels.org/stable/api.html#statistics-and-tests

Time Serie Analysis API

- Statistics and Tests
- Univariate Time-Series Analysis
- Exponential Smoothing
- Multivariate Time Series Models
- Filters and Decompositions
- Markov Regime Switching Models
- Forecasting
- Time-Series Tools



statsmodels

Time Serie Analysis API

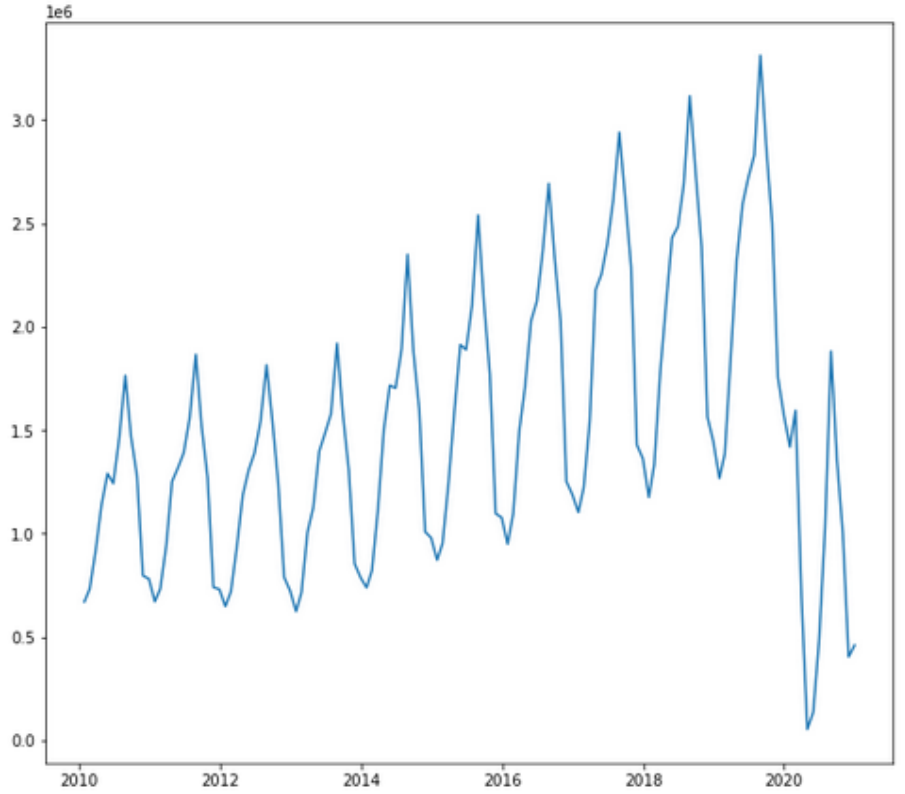
- https://www.statsmodels.org/stable/api.html#statistics-and-tests
- Part of the statsmodels API is
- Time-series models and methods.
- Canonically imported using:

```
import statsmodels.tsa.api as tsa
```

Suppose that we have a time Serie...

- What is the purpose?
- Prediction....

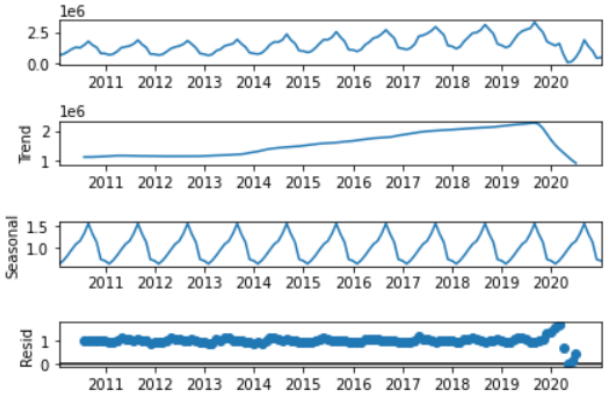
```
import pandas as pd
import matplotlib.pyplot as plt
# read file and create dataframe
df = pd.read_csv('tourismPortugal.csv',sep=";")
df['month'] = pd.to_datetime(df['month'])
df=df.set_index(df['month'])
# create a serie
series = pd.Series(df['tourists'])
# Plot the serie
plt.plot(series)
plt.show()
```



Decompose Time Serie

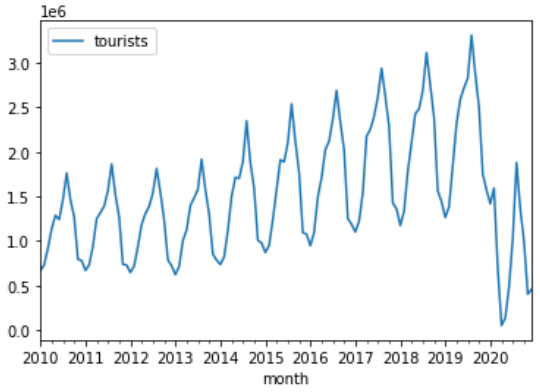
- Parameters:
  - Data (Time Serie)
  - Model
  - Periods
- Output:
  - Observed Serie
  - Trend
  - Seasonal
  - Residual

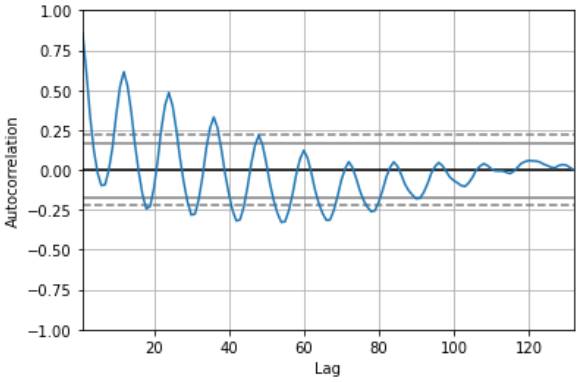
```
import statsmodels.tsa.api as tsa
import pandas as pd

# read file
df = pd.read_csv('tourismPortugal.csv',sep=";")
df['month'] = pd.to_datetime(df['month'])
df=df.set_index(df['month'])
df=df.drop(['month'],axis=1)

result = tsa.seasonal_decompose(df, model='multiplicative', period=12)

figure=result.plot()
```



- Autocorrelation Plot

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import autocorrelation_plot

df = pd.read_csv('tourismPortugal.csv',sep=";")
df['month'] = pd.to_datetime(df['month'])
serie=df[['tourists','month']]
serie=serie.set_index('month')
serie.plot()
plt.show()
autocorrelation_plot(serie)
plt.show()
```





ARIMA

- Preparing the time serie
- Setting lags number, differencing degree, size of moving average
- Create Model
- Fit Model

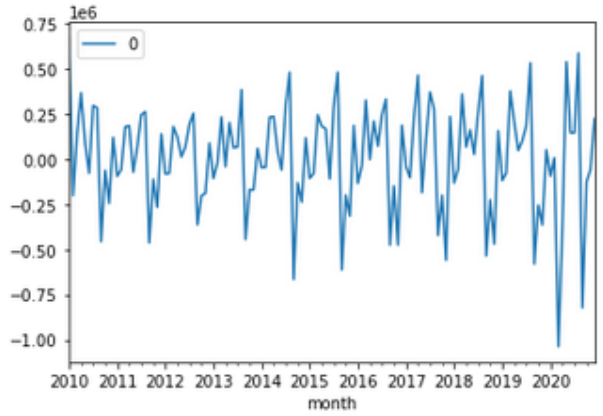
```
import statsmodels.tsa.api as tsa
import pandas as pd

df = pd.read_csv('tourismPortugal.csv',sep=";")
df['month'] = pd.to_datetime(df['month'])
serie=df[['tourists','month']]
serie=serie.set_index('month')
serie.index = serie.index.to_period('M')

#### fit model
p = 4 # number of lags
d =1  #  degree of differencing.
q =0 # size of the moving average window

model = tsa.arima.ARIMA(serie, order=(p,d,q))
result = model.fit()

# summary of fit model
print(result.summary())
```

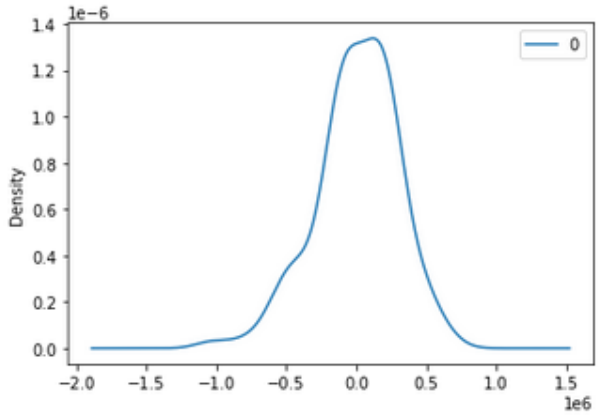
```
                               SARIMAX Results
==============================================================================
Dep. Variable:                tourists   No. Observations:                  132
Model:                 ARIMA(4, 1, 0)   Log Likelihood               -1832.341
Date:                Wed, 03 Mar 2021   AIC                           3674.682
Time:                        20:43:08   BIC                           3689.058
Sample:                    01-31-2010   HQIC                          3680.524
                         - 12-31-2020
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.3783      0.088      4.304      0.000       0.206       0.551
ar.L2          0.1787      0.116      1.541      0.123      -0.049       0.406
ar.L3         -0.3248      0.113     -2.877      0.004      -0.546      -0.104
ar.L4         -0.0495      0.076     -0.648      0.517      -0.199       0.100
sigma2      8.34e+10   3.81e-13   2.19e+23      0.000    8.34e+10    8.34e+10
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 13.51
Prob(Q):                              0.95   Prob(JB):                          0.00
Heteroskedasticity (H):               2.77   Skew:                             -0.69
Prob(H) (two-sided):                  0.00   Kurtosis:                          3.76
===================================================================================
```

Residuals

- Still seasonal information?
- Errors are Gaussian and centered on zero
- All Time Serie used
- Train
- Test

```
# line plot of residuals
residuals = pd.DataFrame(result.resid)
residuals.plot()
plt.show()
# density plot of residuals
residuals.plot(kind='kde')
plt.show()
# summary stats of residuals
print(residuals.describe())
```

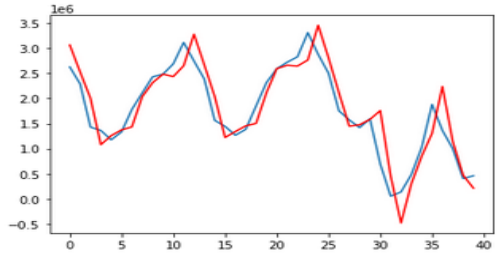





```
                0
count  1.320000e+02
mean   5.720243e+03
std    2.929744e+05
min   -1.037078e+06
25%   -1.331688e+05
50%    4.510200e+04
75%    2.037719e+05
max    6.692650e+05
```

```python
import statsmodels.tsa.api as tsa
from math import sqrt
from sklearn.metrics import mean_squared_error

# split into train and test sets
X = serie.values
size = int(len(X) * 0.70)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()

# walk-forward validation
for t in range(len(test)):
    model = tsa.arima.ARIMA(history, order=(p,d,q))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))

# evaluate forecasts
rmse = sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)

# plot forecasts against actual outcomes
plt.plot(test)
plt.plot(predictions, color='red')
plt.show()
```

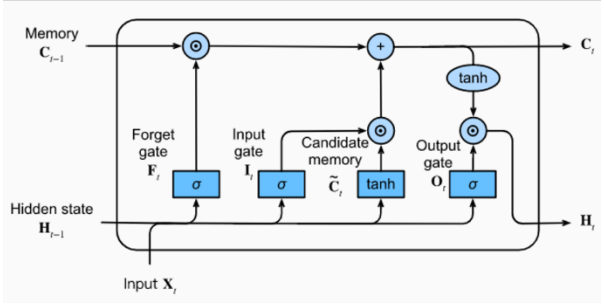
```
predicted=3063604.080269, expected=2626644.000000
predicted=2525982.250892, expected=2288037.000000
predicted=2002943.846006, expected=1428340.000000
predicted=1078516.806844, expected=1361468.000000
predicted=1258560.779588, expected=1175192.000000
predicted=1372904.473562, expected=1332691.000000
predicted=1432217.143661, expected=1781196.000000
predicted=2035578.781393, expected=2108530.000000
predicted=2312727.094421, expected=2430238.000000
predicted=2483461.262802, expected=2482333.000000
predicted=2435286.970732, expected=2688691.000000
predicted=2650288.112961, expected=3114972.000000
predicted=3279049.270532, expected=2744670.000000
predicted=2666538.577637, expected=2384716.000000
predicted=2041275.064636, expected=1563689.000000
predicted=1221972.214214, expected=1442986.000000
predicted=1340725.993823, expected=1266569.000000
predicted=1453534.813600, expected=1384281.000000
predicted=1502523.096499, expected=1851804.000000
predicted=2114363.277998, expected=2315302.000000
predicted=2593929.206090, expected=2591650.000000
predicted=2660734.007920, expected=2721267.000000
predicted=2644743.817489, expected=2829319.000000
predicted=2765675.848207, expected=3310953.000000
predicted=3457691.264943, expected=2876341.000000
predicted=2806680.256163, expected=2500312.000000
predicted=2115370.675047, expected=1754086.000000
predicted=1448101.195276, expected=1572013.000000
predicted=1474614.230027, expected=1418459.000000
predicted=1582339.862212, expected=1594530.000000
predicted=1757101.018797, expected=692691.000000
predicted=456786.428852, expected=53326.000000
predicted=-480050.526637, expected=136493.000000
predicted=300477.997097, expected=482523.000000
predicted=847710.594265, expected=1024811.000000
predicted=1302801.549562, expected=1880926.000000
predicted=2237415.473255, expected=1362664.000000
predicted=1146468.537631, expected=998811.000000
predicted=471767.345690, expected=403446.000000
predicted=211485.252493, expected=459386.000000
Test RMSE: 379005.383
```

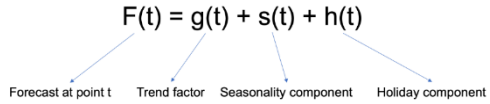


Other Approaches

- LSTM
- Prophet (https://facebook.github.io/prophet/)
- Garch

- https://neptune.ai/blog/arima-vs-prophet-vs-lstm





$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2$$