Taller - Sesion 3 - Series de Tiempo y Python IIE - UNAM

Luis E. Ascencio G.
CIMAT
luis.ascencio@cimat.mx

Abstract Este Notebook incluye una introduccion al manejo de Series de Tiempo con Python

```
import numpy as np # Libreria Matematica basica
import pandas as pd # Libreria para manejo, manipulacion y visualizacion de
datos
from pandas import read_excel # funcion para leer archivos de excel


import matplotlib as mpl # Libreria para visualizacion de datos y graficas
import matplotlib.pyplot as plt   # Funcion para graficar
import seaborn as sns             # Libreria para visualizacion de datos
from pandas.plotting import autocorrelation_plot

from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima_process import ArmaProcess

import session_info
```

```
df = pd.read_csv('AirPassengers.csv')
df1 = read_excel('ClayBricks.xls')
df2 = read_excel('Electricity.xls')
df3 = read_excel('MilkProduction.xls')
df4 = read_excel('JapaneseCars.xls')
df5 = read_excel('HouseSales.xls')
```

Simulacion de Series de Tiempo

Definimos la funcion para graficar series de Tiempo

```
def plot_df(df, x, y, title="", xlabel='Fecha', ylabel='Numero de Pasajeros',
colores="", dpi=100):
    plt.figure(figsize=(15,4), dpi=dpi)
    plt.plot(x, y, color=colores)
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()
```

Programa para simular modelos MA(q)

```
def Simul_TS_MA(Q,T):
    print("TS MA de Orden:",len(Q))
    t0=np.random.rand(len(Q))
    E=list(t0)
    X=[]
    x=0
    for i in range(T):
        e=np.random.normal(0,1)
        x=e
        for j in range(len(Q)):
            x=x+Q[j]*E[-j-1]
        X.append(x)
        E.append(e)
        x=0
        e=0
    return(X)
```

Programa para simular modelos AR(p)

```
def Simul_TS_AR(P,T):
    print("TS AR de Orden:",len(P))
    t0=np.random.rand(len(P))
    E=np.random.rand(len(P))
    X=list(t0)
    x=0
    for i in range(T):
        x=np.random.normal(0,1)
        for j in range(len(P)):
            x=x+P[j]*X[-j-1]
        X.append(x)
        x=0
    return(X)
```

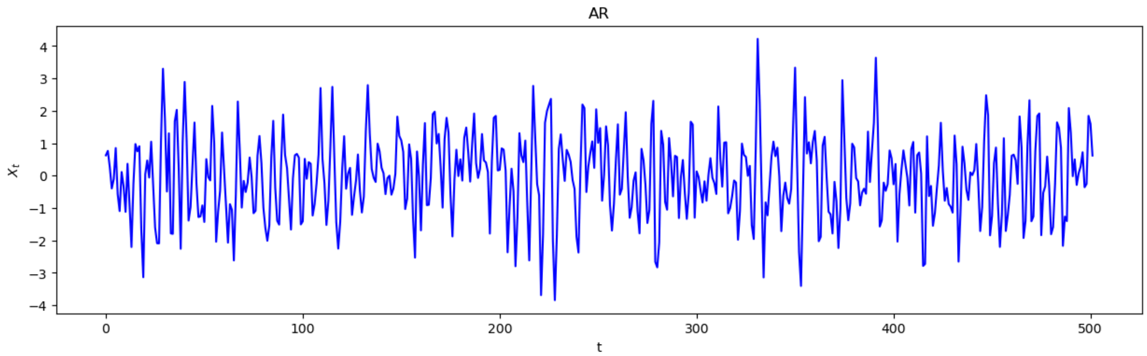
Programa para simular modelos ARMA(p,q)

```python
def Simul_TS_ARMA(P,Q,T):
    print("TS ARMA de Orden: p="+str(len(P))+', q='+str(len(Q)))
    t0=np.random.rand(len(P))
    E=np.random.rand(len(P))
    X=list(t0)
    x=0
    for i in range(T):
        e=np.random.normal(0,1)
        x=e
        for j in range(len(P)):
            x=x+P[j]*X[-j-1]
            x=x+Q[j]*E[-j-1]
        X.append(x)
        x=0
    return(X)
```

Simulamos un modelo AR

```python
C=[1/2,-1/2]
T=500
AR3_1=Simul_TS_AR(C,T)
plot_df(AR3_1, range(T+len(C)), AR3_1, title="AR", xlabel='t', ylabel='$X_t$',
colores="blue", dpi=100)
```

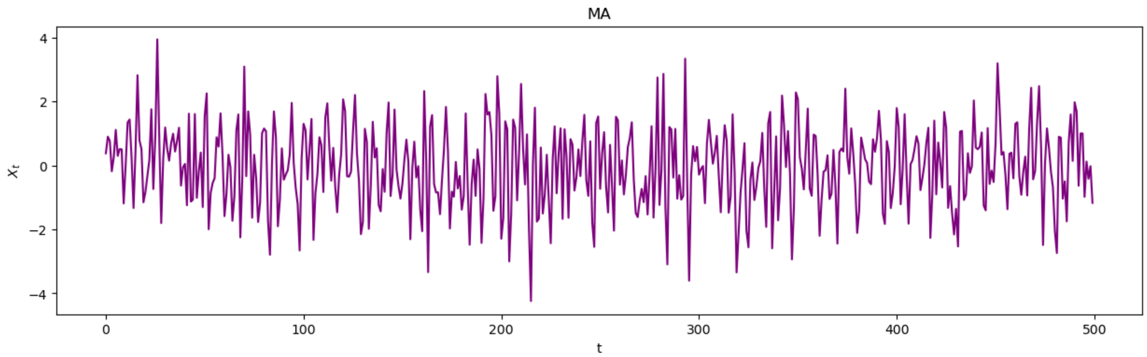
```
TS AR de Orden: 2
```



Simulamos un modelo MA(q)

```python
C=[1/2,-1/2]
MA3_1=Simul_TS_MA(C,T)
plot_df(MA3_1, range(T), MA3_1, title="MA", xlabel='t', ylabel='$X_t$',
colores="purple", dpi=100)
```

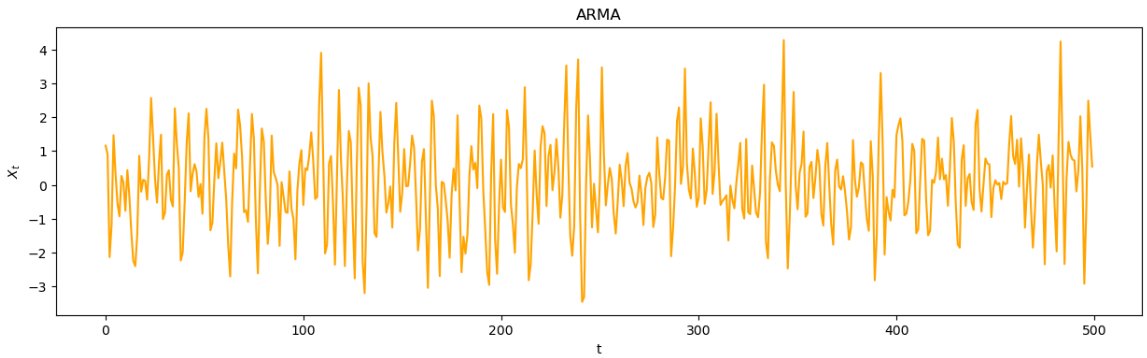
```
TS MA de Orden: 2
```



Simulamos un modelo ARMA(p,q)

```
P=[1/2,-1/2]
Q=[1/2,-1/2]
ARMA3_1=Simul_TS_ARMA(P,Q,T)
plot_df(ARMA3_1[len(P):], range(T), ARMA3_1[len(P):], title="ARMA",
xlabel='t', ylabel='$X_t$', colores="orange", dpi=100)
```

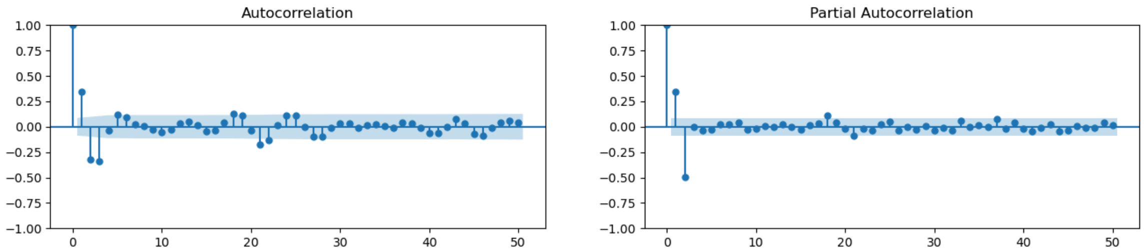
```
TS ARMA de Orden: p=2, q=2
```

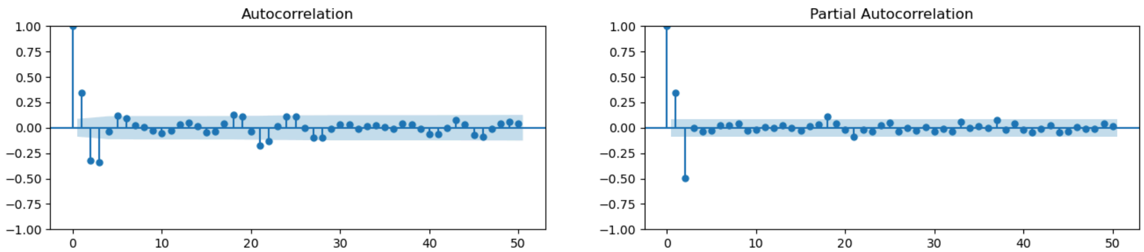


Descripcion de modelos mediante ACF y PACF

ACF y PACF del modelo AR

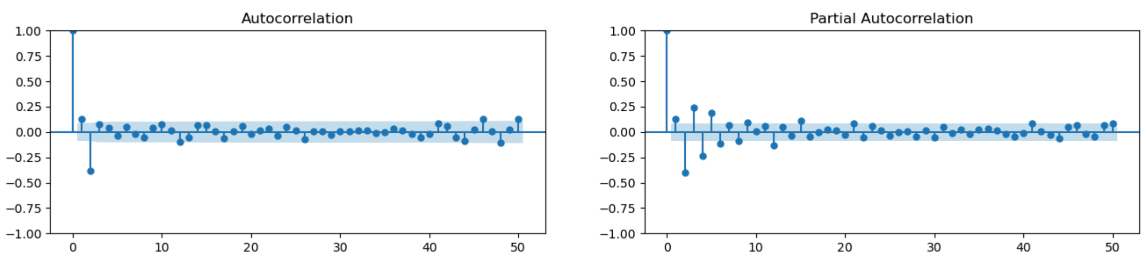
```
# Graficas
fig, axes = plt.subplots(1,2,figsize=(16,3), dpi= 100)
plot_acf(AR3_1, lags=50, ax=axes[0])
plot_pacf(AR3_1, lags=50, ax=axes[1])
```

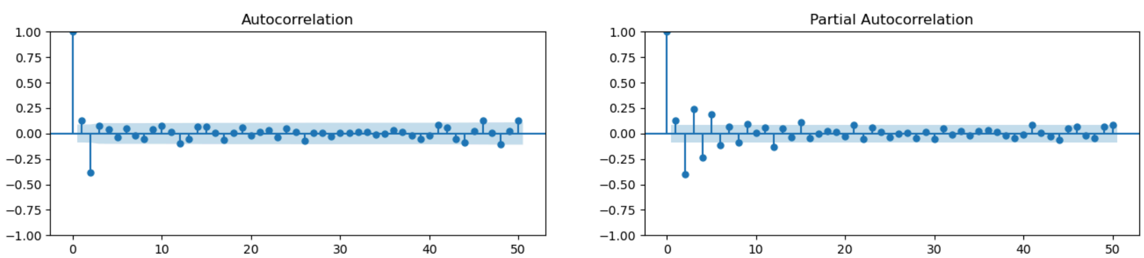




ACF y PACF del modelo MA

```
# Graficas
fig, axes = plt.subplots(1,2,figsize=(16,3), dpi= 100)
plot_acf(MA3_1, lags=50, ax=axes[0])
plot_pacf(MA3_1, lags=50, ax=axes[1])
```

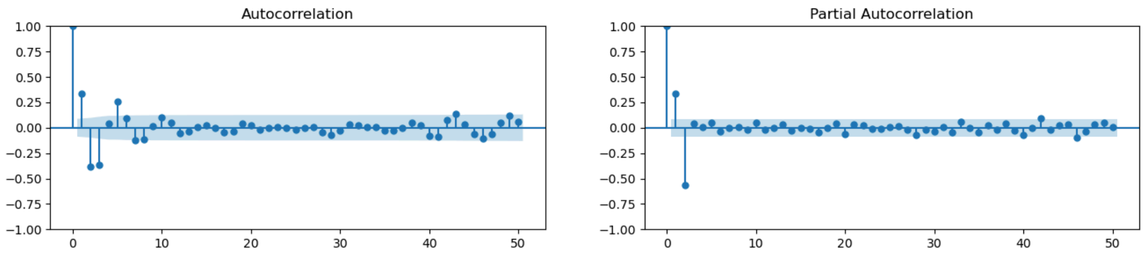


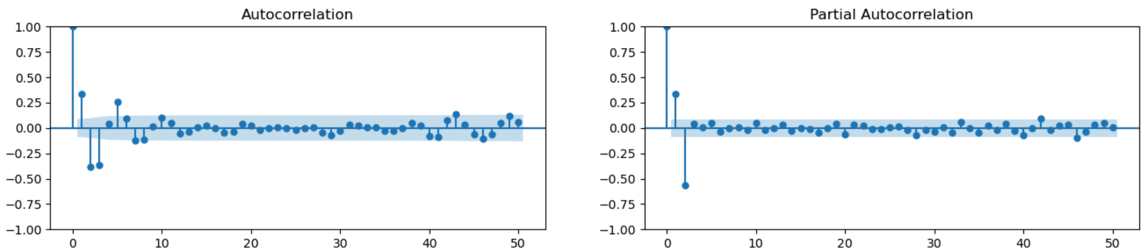


ACF y PACF del modelo ARMA

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

```python
plot_acf(ARMA3_1, lags=50, ax=axes[0])
plot_pacf(ARMA3_1, lags=50, ax=axes[1])
```





Pruebas de Estacionariedad

```python
adf1 = adfuller(AR3_1)
print(f'p-value: {adf1[1]}')
```

```
p-value: 0.0
```

```python
adf2 = adfuller(MA3_1)
print(f'p-value: {adf2[1]}')
```

```
p-value: 7.152051192222167e-06
```

```python
adf3 = adfuller(ARMA3_1)
print(f'p-value: {adf3[1]}')
```

```
p-value: 0.0
```

Estacionariedad de los ejemplos de TS

```python
Data=[df["#Passengers"],df1["Bricks"],df2["Kwh"],df3["Monthly Milk Production
per Cow"],df4["Price"],df5["HouseSales"]]
```

```
w=0
for k in Data:
    w=w+1
    adf = adfuller(k)
    print("P-value de la serie #"+str(w))
    print(f'p-value: {adf[1]}')
```

```
P-value de la serie #1
p-value: 0.991880243437641
P-value de la serie #2
p-value: 0.236826218261678
P-value de la serie #3
p-value: 0.994097902491198
P-value de la serie #4
p-value: 0.6274267086030311
P-value de la serie #5
p-value: 0.16387564674048022
P-value de la serie #6
p-value: 0.03722371625292226
```

Simulacion con StatsModels

```
# Simulate AR(1) process
ar_params = [1, -0.7]  # AR(1) with phi=0.7
ma_params = [1]  # No MA component
ns=500
ar_process = ArmaProcess(ar_params, ma_params)
ar_data = ar_process.generate_sample(nsample=ns)
```

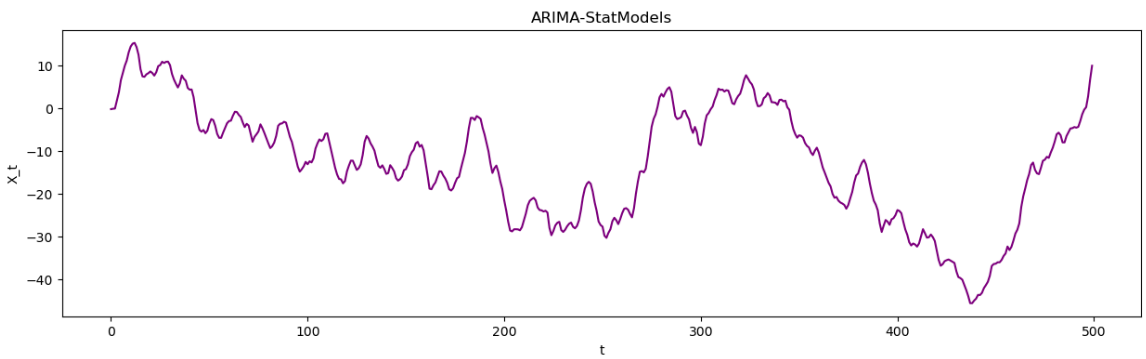
```
plot_df(ar_data,range(ns),ar_data, title="AR-StatModels", xlabel='t',
ylabel='X_t', colores="Blue", dpi=100)
```

```
# Simulate MA(1) process
ar_params = [1]  # No AR component
ma_params = [1, 0.5]  # MA(1) with theta=0.5
ma_process = ArmaProcess(ar_params, ma_params)
ma_data = ma_process.generate_sample(nsample=ns)
```

```
plot_df(ma_data,range(ns),ma_data, title="MA-StatModels", xlabel='t',
ylabel='X_t', colores="purple", dpi=100)
```


```
# Simulate ARIMA(1,1,1) process
ar_params = [1, -0.5]  # AR(1) with phi=0.5
ma_params = [1, 0.4]   # MA(1) with theta=0.4
arima_process = ArmaProcess(ar_params, ma_params)
arima_data = np.cumsum(arima_process.generate_sample(nsample=ns))
```

```
plot_df(arima_data,range(ns),arima_data, title="ARIMA-StatModels", xlabel='t',
ylabel='X_t', colores="purple", dpi=100)
```



Estimacion de Series de Tiempo, Modelos ARIMA

Estimacion modelo AR

```
estim1 = ARIMA(AR3_1, order=(2, 0, 0))
res1 = estim1.fit()
print(res1.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  502
Model:                 ARIMA(2, 0, 0)   Log Likelihood                -730.560
Date:                Mon, 20 Oct 2025   AIC                           1469.120
Time:                        03:32:17   BIC                           1485.995
Sample:                             0   HQIC                          1475.741
                                - 502
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0290      0.047     -0.615      0.538      -0.121       0.063
ar.L1          0.5107      0.039     12.974      0.000       0.434       0.588
ar.L2         -0.4960      0.037    -13.575      0.000      -0.568      -0.424
sigma2         1.0739      0.066     16.374      0.000       0.945       1.203
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
1.39
Prob(Q):                              0.95   Prob(JB):
0.50
Heteroskedasticity (H):               1.19   Skew:
-0.10
Prob(H) (two-sided):                  0.27   Kurtosis:
3.15
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```

Estimacion modelo MA

```
estim2 = ARIMA(MA3_1, order=(0, 0, 2))
res2 = estim2.fit()
print(res2.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  500
Model:                 ARIMA(0, 0, 2)   Log Likelihood                -723.636
Date:                Mon, 20 Oct 2025   AIC                           1455.271
```

```
Time:                        03:32:32   BIC                           1472.130
Sample:                             0   HQIC                          1461.886
                                - 500
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0422      0.045     -0.932      0.351      -0.131       0.047
ma.L1          0.4837      0.040     12.217      0.000       0.406       0.561
ma.L2         -0.5006      0.041    -12.120      0.000      -0.582      -0.420
sigma2         1.0512      0.069     15.141      0.000       0.915       1.187
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
1.39
Prob(Q):                              0.98   Prob(JB):
0.50
Heteroskedasticity (H):               0.98   Skew:
-0.10
Prob(H) (two-sided):                  0.90   Kurtosis:
2.85
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```

Estimacion modelo ARMA

```
estim3 = ARIMA(ARMA3_1, order=(2, 0, 2))
res3 = estim3.fit()
print(res3.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  502
Model:                 ARIMA(2, 0, 2)   Log Likelihood                -722.971
Date:                Mon, 20 Oct 2025   AIC                           1457.942
Time:                        03:32:55   BIC                           1483.253
Sample:                             0   HQIC                          1467.872
                                - 502
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0825      0.046      1.788      0.074      -0.008       0.173
ar.L1          0.4727      0.077      6.120      0.000       0.321       0.624
ar.L2         -0.5448      0.060     -9.111      0.000      -0.662      -0.428
```

```
ma.L1          0.0728      0.090      0.812      0.417      -0.103       0.248
ma.L2          0.0039      0.090      0.043      0.965      -0.172       0.180
sigma2         1.0416      0.066     15.825      0.000       0.913       1.171
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
0.53
Prob(Q):                              0.98   Prob(JB):
0.77
Heteroskedasticity (H):               0.93   Skew:
0.07
Prob(H) (two-sided):                  0.64   Kurtosis:
3.06
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```

```
session_info.show(html=False)
```

```
-----
matplotlib          3.10.6
numpy               2.3.3
pandas              2.3.3
seaborn             0.13.2
session_info        v1.0.1
statsmodels         0.14.5
-----
IPython             9.6.0
jupyter_client      8.6.3
jupyter_core        5.8.1
jupyterlab          4.4.9
notebook            7.4.7
-----
Python 3.13.5 | packaged by conda-forge | (main, Jun 16 2025, 08:27:50) [GCC
13.3.0]
Linux-6.8.0-85-generic-x86_64-with-glibc2.39
-----
Session information updated at 2025-10-20 03:31
```