SERIES DE TIEMPO EJERCICIOS DE PROCESOS ESTACIONARIOS EN R, PYTHON Y MATLAB

VEGA CARHUAZ ITALO
Profesor: Rafael Caparó .

SOLUCION DE EJERCICIOS EN R

VEGA CARHUAZ ITALO
Profesor: Rafael Caparó .

1. PROBLEMA 1

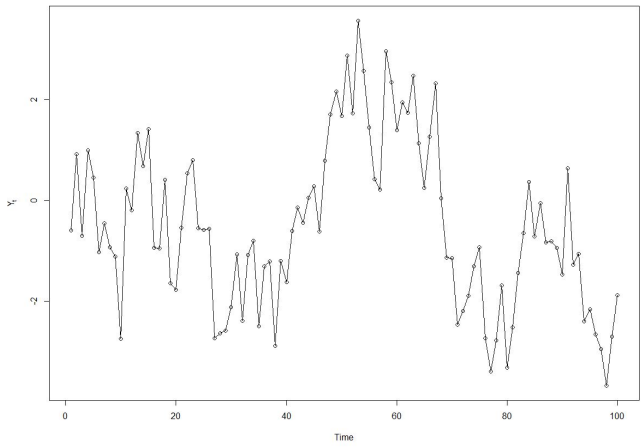
1.1. Simulemos un proceso AR(2), de la forma:

$$y_t = 0{,}5y_{t-1} + 0{,}3y_{t-2} + \varepsilon_t, \quad con \;\; 100 \;\; observaciones$$

```
#El procesos anterior es estacionario, para la demostracion utilizaremos la
    funcion "arima.sim" y  visualizaremos sus graficos de correlacion y de
    estacionariedad.

ar.sim<- arima.sim(model = list(ar=c(.5,.3)),n=100)
ar.sim
#Funcion de AR(2)

win.graph(width = 4, height = 3, pointsize = 8)
plot(ar.sim, ylab=expression(Y[t]),type='o')
```




```
#Funcion de autocorrelación

ar.acf<-acf(ar.sim,type = "correlation",plot=T)
ar.acf
```

```
Autocorrelations of series 'ar.sim', by lag

     1      2      3      4      5      6      7      8
 0.759  0.661  0.574  0.489  0.418  0.330  0.318  0.319
     9     10     11     12     13     14     15     16
 0.277  0.189  0.123  0.017 -0.062 -0.108 -0.153 -0.111
    17     18     19     20
-0.123 -0.134 -0.148 -0.209
```

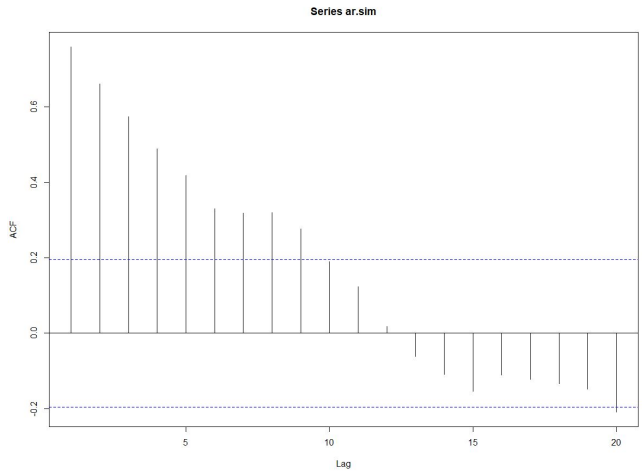


Figura 1: Función autocorrelacion

2. PROBLEMA 2

2.1. Simulemos un MA(2) de la forma

$$y_t = -0{,}7_{t-1} + 0{,}1\varepsilon_{t-2} + \varepsilon_t \quad con \quad 100 \quad observaciones$$

```
#Tenemos un modelo de media movil, con la ayuda del comando "arima.sim" y
    las graficas verificaremos su estacionariedad.

ma.sim<- arima.sim(model=list(ma=c(-.7,.1)),n=100)
ma.sim

# La gráfica del proceso anterior es :

win.graph(width = 4.875, height = 3, pointsize = 8)
plot(ma.sim,ylab=expression((e[t])),type='o')
```

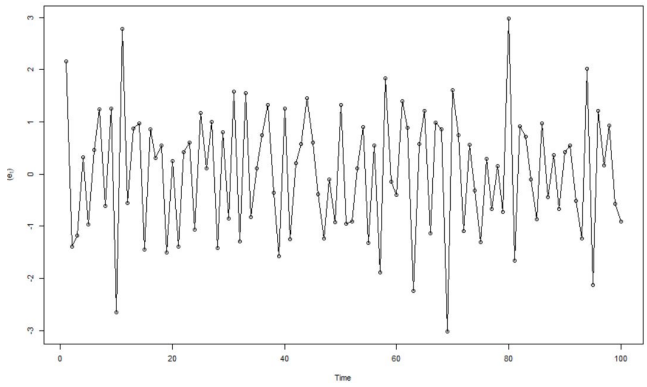


Figura 2: Proceso MA(2)

```
#La función de autocorrelacion simple.
ma.acf<-acf(ma.sim,type="correlation",plot=T)
ma.acf
```

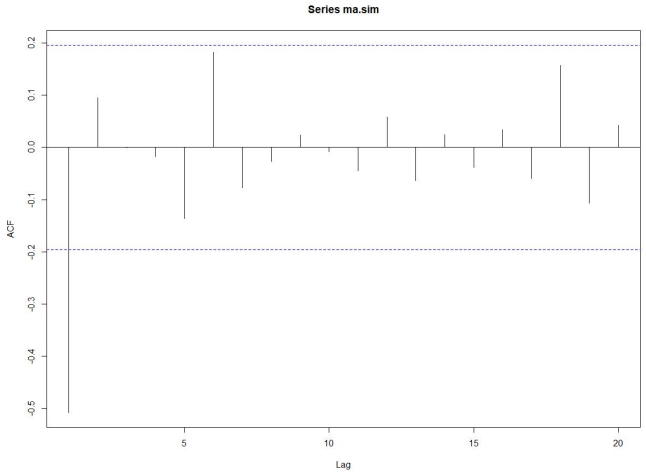


Figura 3: Funcion de autocorrelacion simple

```
#La función de autocorrelacion parcial.
ma.pacf<-acf(ma.sim,type="partial",plot=T)
ma.acf
```

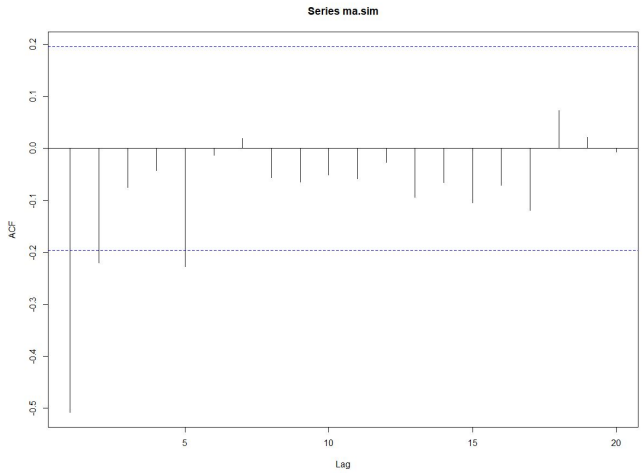


Figura 4: Funcion de autocorrelacion parcial

3. PROBLEMA 3

3.1. Ahora simulemos un proceso ARMA(2,2) de la forma:

$$y_t = 0{,}5y_{t-1} - 0{,}2y_{t-2} + \varepsilon_t - 0{,}4\varepsilon_{t-1} + 0{,}3\varepsilon_{t-2}, \quad con \;\; 100 \;\; observaciones$$

```
#En el proceso anterior, mostraremos su estacionariedad a traves del comando
    "arima.sim" y sus graficos .
arma.sim<- arima.sim(model = list(ar=c(.5,-.2),ma=c(-.4,.3)),n=100)
arma.sim
```

Obs.
Para realizar el siguiente proceso necesitaremos activar el paquete "tseries"

```
library("tseries")
win.graph(width = 4.875, height = 3, pointsize = 8)
ts.plot(arma.sim)
```

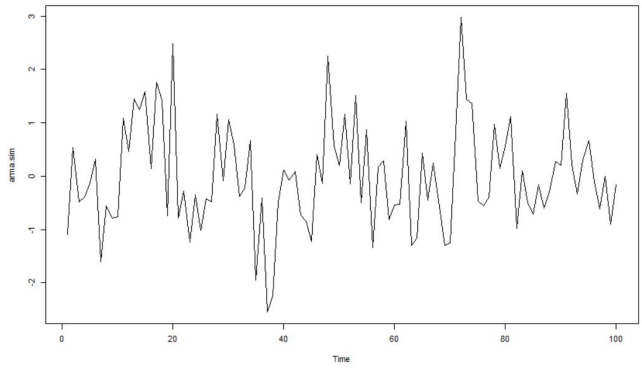


Figura 5: Proceso ARMA(2,2)

```
#La función de autocorrelacion simple.
arma.acf<-acf(arma.sim,type="correlation",plot = T)
arma.acf
```

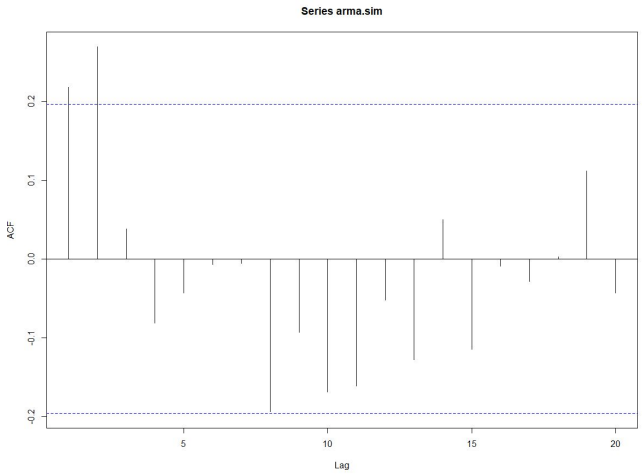


Figura 6: Funcion de autocorrelacion simple

```
#La función de autocorrelacion parcial.
arma.pacf<-acf(arma.sim,type = "partial",plot = T)
arma.pacf
```

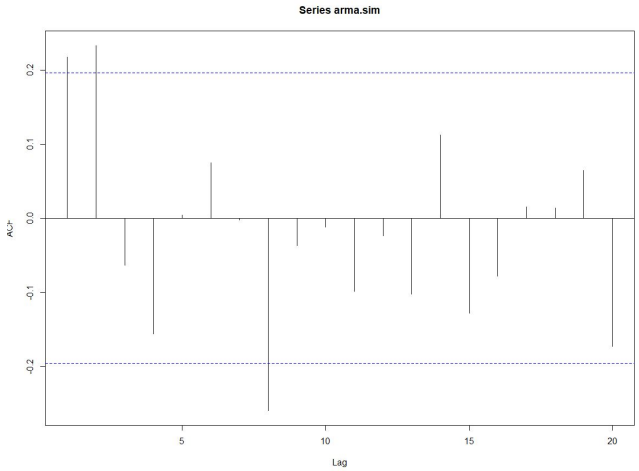


Figura 7: Funcion de autocorrelacion parcial

SOLUCION DE EJERCICIOS EN PYTHON

VEGA CARHUAZ ITALO
Profesor: Rafael Caparó .

4. PROBLEMA 1

OBSERVACION:
Para el problema que presentaremos, necesitaremos ayuda de Yahoo Finance.

```
#Ejemplo de serie de tiempo con Panda
#Creando una serie de tiempo de las acciones de WFT desde yahoo finance

wft = web.DataReader("WFT", 'yahoo', '2016-1-1', '2016-9-30')
wft.head(5)
```

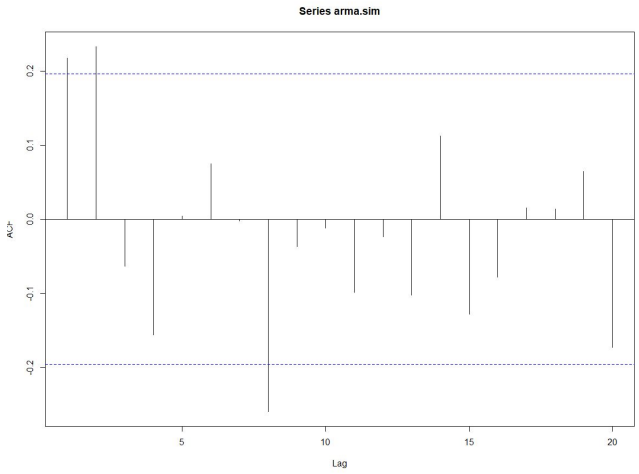


Figura 8: Precios importados desde Yahoo Finance

```
#Filtraremos sólo del periodo 2016-02-04 al 2016-02-18
wft['2016-02-04':'2016-02-18'])
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2016-02-04	7.10	7.82	6.99	7.39	34474500	7.39
2016-02-05	7.37	7.52	6.87	6.94	27775700	6.94
2016-02-08	6.68	6.79	6.41	6.74	17611300	6.74
2016-02-09	6.60	6.72	6.07	6.34	13741100	6.34
2016-02-10	6.28	6.59	6.11	6.24	8623900	6.24
2016-02-11	6.02	6.27	5.74	6.06	17133900	6.06
2016-02-12	6.14	6.66	6.06	6.47	13498600	6.47
2016-02-16	6.66	6.74	6.33	6.62	11453500	6.62
2016-02-17	6.70	7.13	6.55	6.72	29061300	6.72
2016-02-18	6.95	6.96	6.22	6.51	13587900	6.51

Figura 9: Precios filtrados

Nuestro ejercicio sera desplazar los valores hacia atrás y adelante en el tiempo, como por ejemplo para calcular el cambio porcentual de una muestra a otra. En Pandas podemos utilizar el método .shift()

```
#Desplazaremos el 1 dia el valor de cierre
desplazado = wft['Adj Close'].shift(1)
desplazado[:5]
```

```
Date
2016-01-04     NaN
2016-01-05    8.64
2016-01-06    8.26
2016-01-07    7.91
2016-01-08    7.34
Name: Adj Close, dtype: float64
```

Figura 10: Valor de cierre

```
#Calculamos el porcentaje de variación del día
variacion_diaria = wft['Adj Close'] / wft['Adj Close'].shift(1) - 1
wft['var_diaria'] = variacion_diaria
wft['var_diaria'][:5]
```

```
Date
2016-01-04       NaN
2016-01-05   -0.0440
2016-01-06   -0.0424
2016-01-07   -0.0721
2016-01-08   -0.0504
Name: var_diaria, dtype: float64
```

Figura 11: Valor de cierre

```
#Calculamos el rendimiento acumulado diario
rendimientodiario = (1 + wft['Adj Close'].pct_change()).cumprod()
wft['rend_diario'] = rendimiento_diario
wft['rend_diario'][:5]
```

```
Date
2016-01-04       NaN
2016-01-05    0.9560
2016-01-06    0.9155
2016-01-07    0.8495
2016-01-08    0.8067
Name: rend_diario, dtype: float64
```

Figura 12: Rendimiento diario

Una operación fundamental para entender el comportamiento de una serie de tiempo y poder determinar si se trata de una serie estacionaria o no; es realizar gráficos de la misma. En Pandas esto lo podemos realizar en forma muy sencilla con el método .plot().

```
# Graficamos el precio de cierre ajustado Adj Close
plot = wft['Adj Close'].plot(figsize=(10, 8))
```

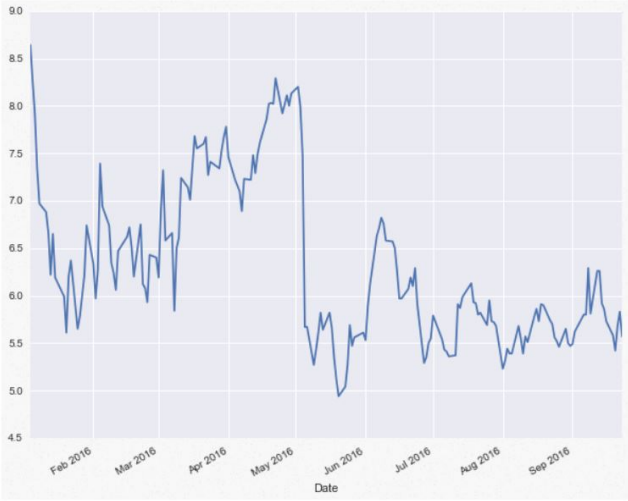


Figura 13: Precio Ajustado

```
# Aplicamos el filtro Hodrick-Prescott para separar en tendencia y
# componente ciclico.
wft_ciclo, wft_tend = sm.tsa.filters.hpfilter(wft['Adj Close'])
wft['tend'] = wft_tend
# Graficamos la variacion del precio real con la tendencia.
wft[['Adj Close', 'tend']].plot(figsize=(10, 8), fontsize=12);
legend = plt.legend()
legend.prop.set_size(14)
```

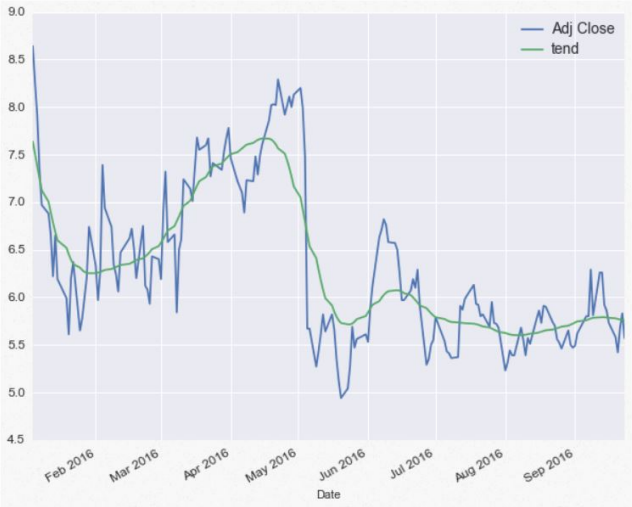


Figura 14: Serie descompuesta

```
#Graficamos el rendimiento diario
plot = wft['var_diaria'].plot(figsize=(10, 8))
```

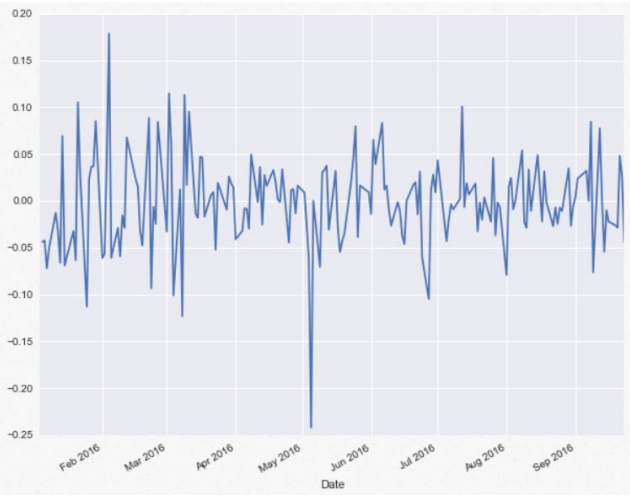


Figura 15: Rendimientos

5. PROBLEMA 2

5.1. Promedios móviles y descomposición

Otra técnica interesante que podemos intentar también es la descomposición. Esta es una técnica que trata de descomponer una serie de tiempo en su tendencia, su estacionalidad y sus factores residuales. Statsmodels

viene con una función de descomposición que nos facilita en sobremanera el trabajo, para el ejercicio anterior tenemos lo siguiente :

```
# Calculando promedios móviles cada 5 días
wft_ma = pd.rolling_mean(wft['Adj Close'], 5)
wft['prod_mov'] = wft_ma
plot = wft[['Adj Close', 'prod_mov']].plot(figsize=(10, 8), fontsize=12)
```

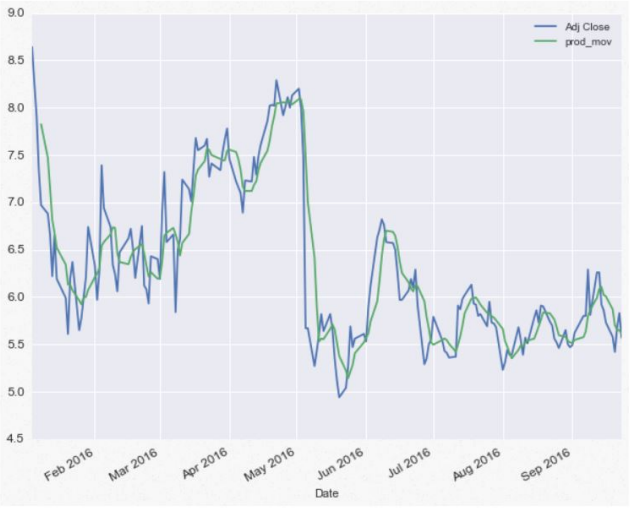


Figura 16: Promedios Moviles

```
# Ejemplo de descomposición de serie de tiempo
descomposicion = sm.tsa.seasonal_decompose(wft['Adj Close'],model='additive'
    , freq=30)
fig = descomposicion.plot()
```

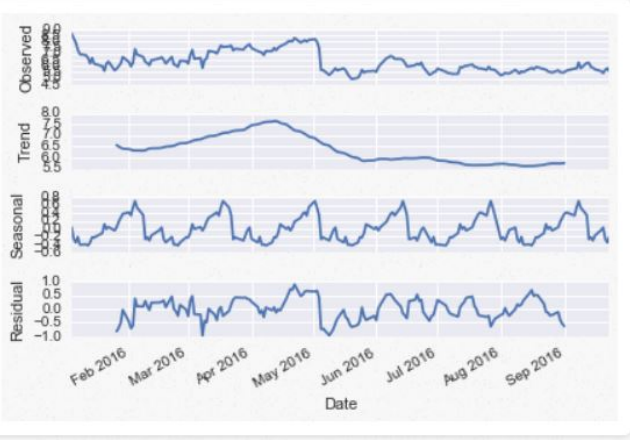


Figura 17: Promedios Moviles

6. PROBLEMA 3

6.1. Pronosticaremos la serie con ARIMA

Partiremos del ejercicio anterior y como podemos observar en los gráficos que realizamos, el comportamiento de la serie de tiempo con la que estamos trabajando parece ser totalmente aleatorio y las medidas móviles que calculamos tampoco parecen ser de mucha utilidad para acercar la serie a un comportamiento estacionario. De todas formas podemos intentar aplicar un modelo ARIMA sobre la serie y ver que tan bien nos va con el pronostico del modelo. El modelo ARIMA es similar a una regresión estadística pero aplicando los conceptos de las series de tiempo; por tanto, los pronósticos del modelo vienen explicadas por los datos del pasado y no por variables independientes.

```
# Modelo ARIMA sobre el valor de cierre de la acción.
modelo = sm.tsa.ARIMA(wft['Adj Close'].iloc[1:], order=(1, 0, 0))
resultados = modelo.fit(disp=-1)
wft['pronostico'] = resultados.fittedvalues
plot = wft[['Adj Close', 'pronostico']].plot(figsize=(10, 8))
```

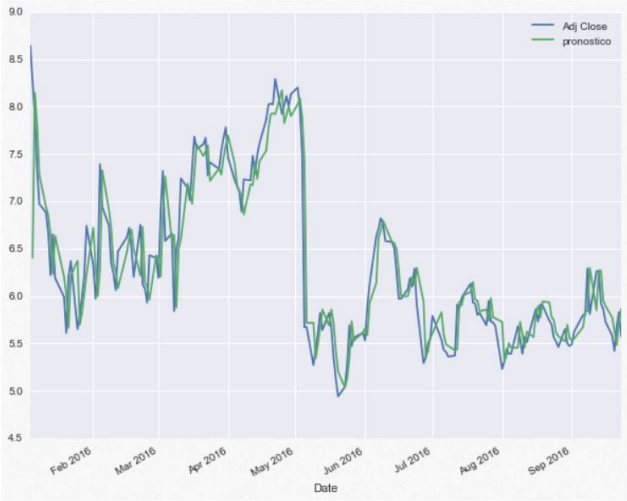


Figura 18: Modelo ARIMA

Aquí el modelo parece ser bastante efectivo, las líneas en el gráfico son muy similares. Pero para armar el modelo hemos utilizado el valor de cierre de la acción, lo que realmente nos interesa predecir es la variación diaria del precio de la acción, por lo tanto deberíamos armar el modelo utilizando la columna de variación diaria que calculamos previamente

```
# Modelo ARIMA sobre variación diaria
modelo = sm.tsa.ARIMA(wft['var_diaria'].iloc[1:], order=(1, 0, 0))
resultados = modelo.fit(disp=-1)
wft['pronostico'] = resultados.fittedvalues
plot = wft[['var_diaria', 'pronostico']].plot(figsize=(10, 8))
```

Figura 19: Modelo ARIMA

SOLUCION DE EJERCICIOS EN MATLAB

VEGA CARHUAZ ITALO
Profesor: Rafael Caparó .

7. PROBLEMA 1

7.1. Mostrar la grafica de la siguiente serie

$$y_t = 0{,}4 + 0{,}25 y_{t-1} + \varepsilon_t,$$

```
phi0=0.4
phi1=0.25
%epsilon=randn(1,1)
y=zeros(100,1)
for t=2:100
y(t)=phi0+phi1*y(t-1)+randn(1,1)
end
plot(y)
```

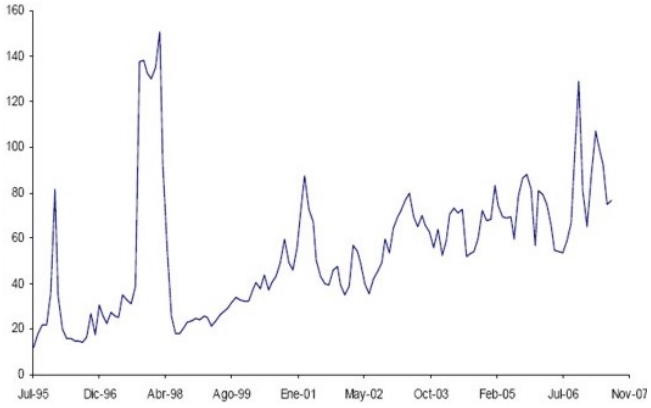


Figura 20: Modelo AR(1) en Matlab