

Clase 3

⚙️ Status	In progress
👤 Assign	🤖 Sharop

▼ 2.3 Modelos ARMA, SARIMA.

▼ Intuición

▼ Ruido Blanco

Una serie $(w_t, t \in \mathbb{Z})$

se dice que es Ruido Blanco si cumple

- $E(w_t) = 0$ (media cero)
- $Var(w_t) = \sigma^2$ (varianza constante)
- $Cov(w_t, w_{t+k}) = 0$ (sin correlación)

Si además cumple que $w_t \sim N(0, \sigma^2)$ se dice que w_t es Ruido Blanco Gaussiano (RBG).

```
import numpy as np

mu, sigma = 0, 3 # media y desviacion estandar
s = np.random.normal(mu, sigma, 200) # Generamos 200 muestras de una distribucion normal
```

El ruido blanco se puede detectar si se analiza la función de autocorrelación muestral.

Es una función (`acf` en `R`) que para instante t , y cada entero k toma un valor $\hat{\rho}_k(t)$

$$\hat{\rho}_k(t) = \frac{\hat{\gamma}_k(t)}{\hat{\gamma}_0}$$

donde k identifica el rezago Y_{t-k}

$$\hat{\gamma}_k = \frac{\sum (Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{n}$$

$$\hat{\gamma}_0 = \frac{\sum (Y_t - \bar{Y})^2}{n}$$

Se asume que $\hat{\rho}_k \sim N(0, 1/\sqrt{n})$. Es decir, si:

$$\hat{\rho}_k = \frac{\sum_{t=k+1}^T (Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=1}^T (Y_t - \bar{Y})^2}$$

presenta valores fuera de las bandas $\pm 2(\frac{1}{\sqrt{n}})$ (ver líneas entre cortadas en gráfico), es evidencia de la presencia de autocorrelación.

La función de autocorrelación muestral en Python se puede calcular usando la función `sm.tsa.acf(x)`. Esta función toma dos parámetros: los datos de entrada y el tipo de correlación a utilizar. Por ejemplo, para calcular la autocorrelación muestral de una serie temporal usando Python, podemos usar el siguiente código:

```
import numpy as np
import statsmodels.api as sm
from statsmodels.graphics import tsaplots
import matplotlib.pyplot as plt

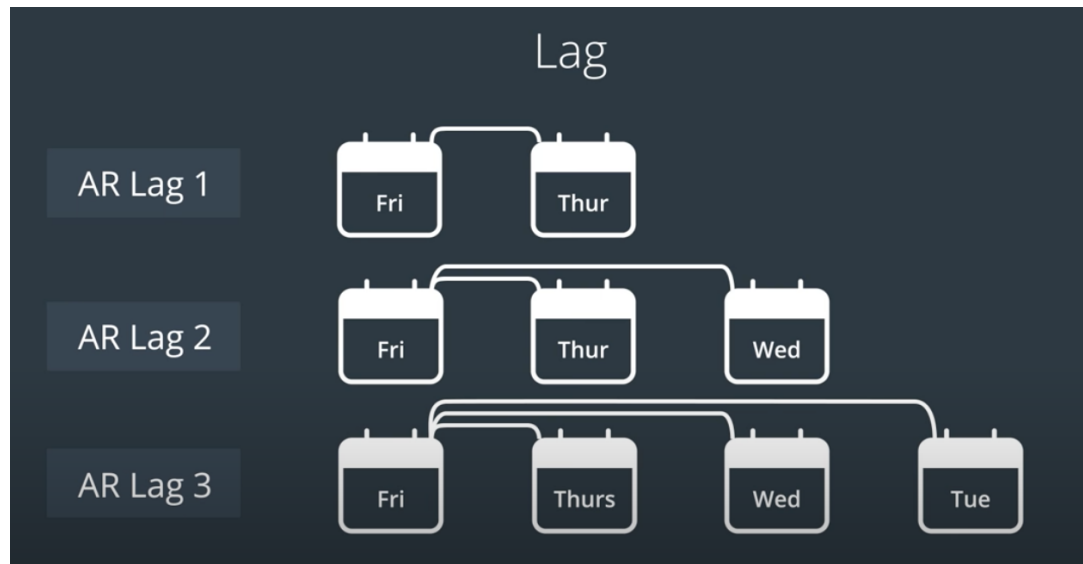
import numpy as np

mu, sigma = 0, 3 # media y desviacion estandar
s = np.random.normal(mu, sigma, 200) # Generamos 200 muestras de una distribucion normal

#calculate autocorrelations
sm.tsa.acf(s)
sm.tsa.acf(s, nlags=5)

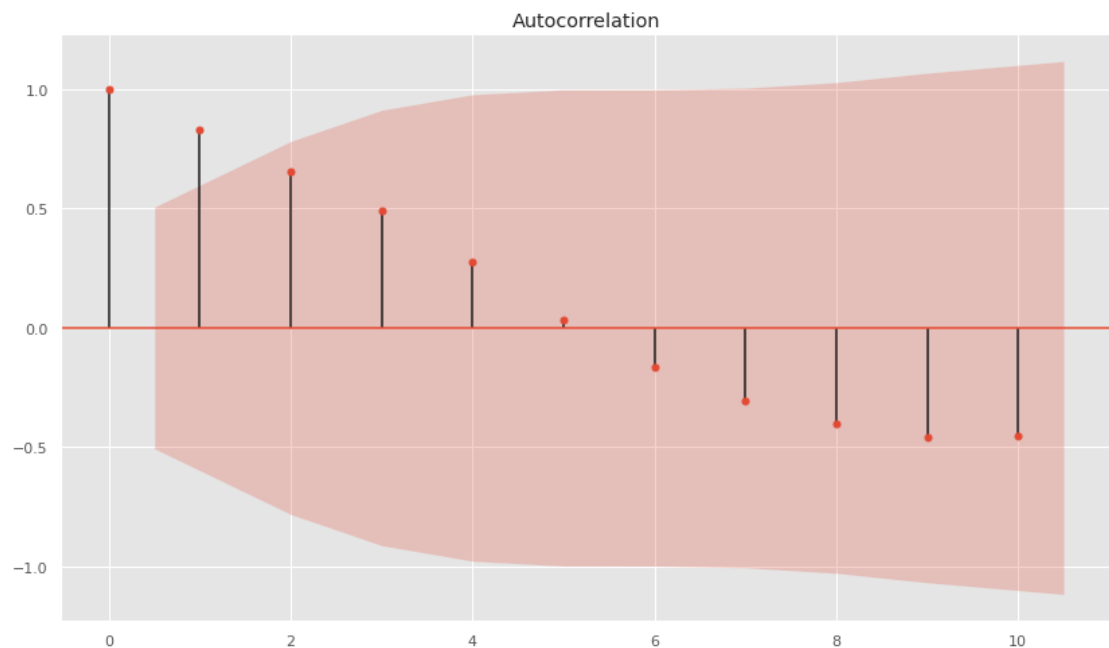
#Graficando la función de autocorrelación.
fig = tsaplots.plot_acf(x, lags=10)
plt.show()
```

El lag, son los elementos previos que utilizamos para calcular el valor actual.



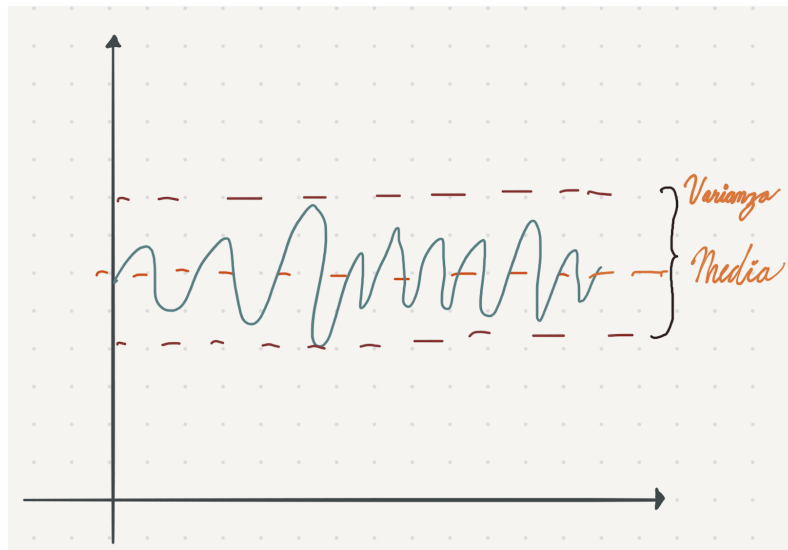
Referencia de imagen: udacity - AI fro trading

Si el valor de la función de autocorrelación se sale de las franjas, si hay correlación significativa y no hay ruido blanco.



▼ Serie estacionaria

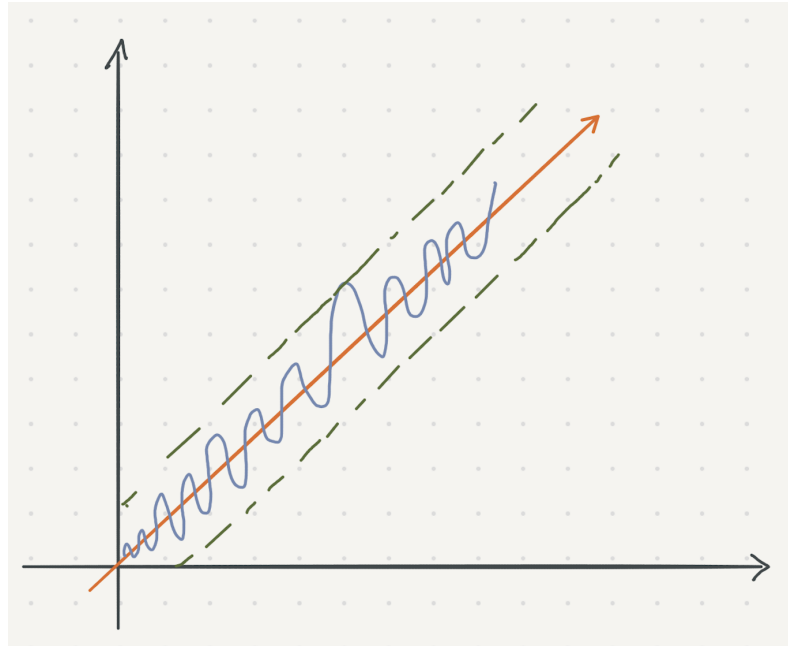
- Las series estacionarias y_t cumplen tres propiedades:
 - La esperanza matemática de y_t es constante a lo largo del tiempo, no importa que este rezagado k periodos.
 - $E(y_t) = E(y_{t-k})$
 - La varianza es constante a lo largo de todas las series
 - $Var(y_t) = \sigma^2$
 - La covarianza entre dos valores diferentes, será la misma siempre que este separada en k periodos.
- Serie estacionaria en medias
 - Una serie temporal estacionaria se refiere a una serie temporal cuyo valor medio se mantiene constante a lo largo del tiempo. Esto significa que los valores fluctúan alrededor de un valor fijo que es su media, con una varianza como intervalo de fluctuación.



Estacionaria en medias

- Serie estacionaria en tendencia

- Una serie es estacionaria en tendencia si su incremento o disminución fluctúa alrededor de una línea de tendencia y cuya varianza también permanece constante.



Estacionaria en tendencia

- La forma de una serie estacionaria es de la siguiente forma:

"donde el valor de"

" \leftarrow resolviendo esta ecuación tenemos.

" \leftarrow Sustituyendo el valor de"

" \leftarrow resolviendo

"

Si este procedimiento se repite sucesivas ocasiones, tenemos

"

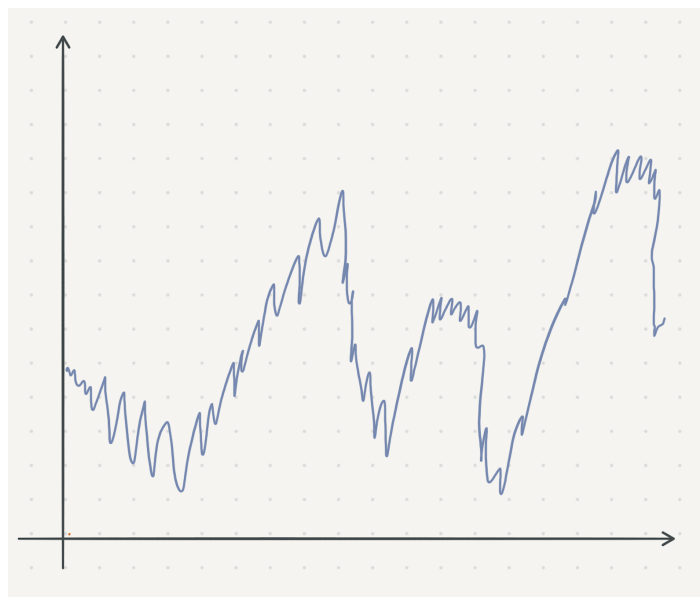
Como $\phi < 1 \Rightarrow \phi^{k+1} \rightarrow 0$ cuando $k \rightarrow \infty$, podemos reescribir de la siguiente forma:

»

Podemos concluir que los errores (shocks) tienen memoria de corto plazo y que los errores pasados tienen menos importancias que los mas recientes.

▼ Serie no estacionaria (Random Walk)

- Una serie no estacionaria es aquella que no cumple con las tres condiciones necesarias para ser estacionaria: la esperanza matemática debe fluctuar con el tiempo, la varianza debe variar a lo largo de la serie temporal y la covarianza entre dos valores diferentes debe ser diferente en función de un periodo de tiempo determinado. Esto significa que los valores de la serie temporal no se mantienen alrededor de una línea de tendencia media, y la varianza fluctúa a lo largo de la serie temporal.



Proceso estocástico no estacionario

- La forma de una serie no estacionaria es la siguiente:

»

Esto quiere decir que la serie viene explicada por el pasado mas un rezago.

»

¿Cual seria el valor de Y_{t-1} ?

»

»

entonces.

»

Sustituyendo tenemos

»

Hasta encontrar que

»

Una serie de este tipo al final es la que va acumulando los errores (shocks) a lo largo del tiempo. Entonces las series estacionarias resultan del acumulado de los errores.

- Podemos ver que los Shock o errores tienen memoria infinita, ya que es una acumulación de todos los errores en nuestra serie de tiempo. La importancia de un evento ocurrido $t-1$, $t-100$ tiene la misma importancia. Esto quiere decir que todos los errores tienen el mismo peso al momento de explicar a nuestra serie temporal.

▼ Ljung-Box

El test Ljung-Box es una prueba estadística utilizada en análisis de series de tiempo para determinar si existe autocorrelación en una serie de tiempo. El nombre del test proviene de los estadísticos Greta Ljung y Harald Box, quienes lo propusieron en un artículo en 1978.

La hipótesis nula del test es que no existe autocorrelación en la serie de tiempo, mientras que la hipótesis alternativa es que hay autocorrelación. El test se basa en una prueba de hipótesis de chi-cuadrado y se utiliza para evaluar la significancia de la autocorrelación en varios retrasos de la serie de tiempo.

Para utilizar el test Ljung-Box en Python, podemos utilizar la biblioteca statsmodels. El siguiente ejemplo muestra cómo utilizar el test para evaluar la autocorrelación en una serie de tiempo.

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

# Generamos una serie de tiempo aleatoria
np.random.seed(123)
x = np.random.normal(0, 1, 100)

# Calculamos los estadísticos Q y p-valor del test Ljung-Box para los primeros 10 retrasos
lbvalue, pvalue = sm.stats.acorr_ljungbox(x, lags=10)

# Imprimimos los resultados del test
for i in range(len(lbvalue)):
    print("Ljung-Box test statistic for lag {}: {:.2f} with p-value: {:.2f}".format(i+1, lbvalue[i], pvalue[i]))
```

En este ejemplo, se genera una serie de tiempo aleatoria utilizando la función `np.random.normal` de NumPy. Luego, se utiliza la función `sm.stats.acorr_ljungbox` de statsmodels para calcular los estadísticos Q y los p-valores del test Ljung-Box para los primeros 10 retrasos de la serie de tiempo.

El resultado impreso en la consola muestra el estadístico Q y el p-valor para cada retraso en la serie de tiempo. Si el p-valor es menor que el nivel de significancia, podemos rechazar la hipótesis nula y concluir que existe autocorrelación en la serie de tiempo.

▼ Estacionalidad o Estacionario -

La estacionalidad se refiere a patrones repetitivos o ciclos que ocurren en un conjunto de datos en diferentes momentos del tiempo, como estaciones del año, días de la semana o meses. Por otro lado, un proceso estacionario es aquel que tiene propiedades estadísticas constantes a lo largo del tiempo, es decir, su media, varianza y autocorrelación son constantes.

La principal diferencia entre ambos conceptos radica en que la estacionalidad es un patrón que ocurre en momentos específicos del tiempo, mientras que un proceso estacionario se mantiene constante en el tiempo.

Ejemplo de código en Python para un proceso estacional:


```
import numpy as np
import matplotlib.pyplot as plt

# Generar datos aleatorios estacionarios
np.random.seed(0)
serie_estacionaria = np.random.normal(0, 1, 1000)

# Graficar los datos
plt.plot(serie_estacionaria)
plt.title("Ejemplo de proceso estacionario")
plt.xlabel("Tiempo")
plt.ylabel("Valores")
plt.show()
```

En este ejemplo, se genera una serie de datos aleatorios normalmente distribuidos con media cero y desviación estándar de uno. Como se trata de un proceso estacionario, los datos no presentan patrones cíclicos ni tendencias claras, y su media y varianza se mantienen constantes a lo largo del tiempo.

Ejemplo de código en Python para una serie con estacionalidad:

```
import numpy as np
import matplotlib.pyplot as plt

# Generar datos aleatorios con estacionalidad
np.random.seed(0)
serie_estacional = np.sin(np.arange(1000)/10) + np.random.normal(0, 0.1, 1000)

# Graficar los datos
plt.plot(serie_estacional)
plt.title("Ejemplo de serie con estacionalidad")
plt.xlabel("Tiempo")
plt.ylabel("Valores")
plt.show()
```

En este caso, se genera una serie de datos que incluye un patrón cíclico estacional, en este caso una función sinusoidal, más un componente de ruido aleatorio. Los datos presentan un patrón cíclico que se repite a lo largo del tiempo y, por lo tanto, no son estacionarios en el sentido estadístico.


▼ Resumen

Las series estacionarias tienen memoria de corto plazo, mientras las series no estacionarias, tienen memoria de largo plazo.

▼ Herramientas

<http://personal.cimat.mx:8181/~jortega/MaterialDidactico/ST2013/STClase4-5.pdf>

Time Series analysis tsa — statsmodels

 <https://www.statsmodels.org/devel/tsa.html>

▼ ¿Que son los modelos ARMA y SARIMA?

Objetivo

Los modelos ARMA y SARIMA son modelos estadísticos para series de tiempo que combinan características tanto del modelo autoregresivo (AR) como del modelo de promedio móvil (MA). Estos modelos se usan para capturar la dependencia entre los puntos de datos en una serie temporal. El objetivo de estos modelos es predecir los valores futuros de una serie temporal dada. Además, los modelos ARMA y SARIMA pueden usarse para explicar los patrones subyacentes en una serie temporal. Esto se logra al usar los modelos para capturar la dependencia entre los precios pasados y futuros del activo. Los modelos ARMA y SARIMA se usan principalmente para pronosticar datos financieros, meteorológicos, económicos y demográficos.

Por lo tanto, estos modelos pueden usarse para pronosticar y explicar la información de una serie temporal.

La idea de estos modelos es que los valores actuales de la serie X_t dependen de los p valores previos: X_{t-1}, \dots, X_{t-p} .

Definición. Un modelo autoregresivo de orden p , denotado por $AR(p)$, es de la forma

»

donde X_t es estacionario, $\phi_1, \phi_2, \dots, \phi_p$ son constantes ($\phi_p \neq 0$) y w_n es un ruido blanco con media 0 y varianza σ_w^2 .

Definición. Operador autoregresivo se define por:

»

Consideramos el modelo autoregresivo de primer orden:

$$X_t = \phi X_{t-1} + w_t$$

en el cual iterando hacia el pasado obtenemos:

»

Continuando el proceso siempre que $|\phi| < 1$ y X_t sea estacionario, podemos representar $AR(1)$ como un proceso lineal:

$$X_t = \sum_{j=0}^{\infty} \phi^j w_{t-j}$$

este proceso es estacionario con media

$$E(X_t) = \sum_{j=0}^{\infty} \phi^j E(w_{t-j})$$

y función de covarianza

$$\gamma(h) = Cov(X_{t+h}, X_t) = \frac{\sigma_w^2 \phi^h}{1 - \phi^2}, h \geq 0$$

Aquí está un ejemplo de un modelo AR(1) en Python con una gráfica para ayudar a entender mejor el concepto:

```
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.graphics import tsaplots

# Definimos nuestros parámetros
phi = 0.5 # coeficiente AR
sigma = 0.1 # varianza

# Definimos una matriz para almacenar nuestros valores generados
X = np.empty(100)

# Generamos valores aleatorios para nuestro ruido blanco
e = np.random.normal(0, sigma, 100)
```

```

# Generamos los valores de la serie temporal usando la ecuación AR(1)
for t in range(1, len(X)):
    X[t] = phi*X[t-1] + e[t]

# Graficamos nuestra serie temporal
plt.plot(X)
plt.show()

#Calculando autocorrelaciones
sm.tsa.acf(X)
sm.tsa.acf(X, nlags=5)

#Graficando la función de autocorrelación.
fig = tsaplots.plot_acf(X, lags=10)
plt.show()

```

Definición. Un modelo de media móvil q (MA(q)) es un modelo estadístico para series de tiempo que considera el ruido blanco como una media móvil. Esto significa que los valores de la serie de tiempo se explican en términos de los ruidos blancos recientes.

$MA(q)$, es de la forma:

»

donde X_t es estacionario, $\theta_1, \theta_2, \dots, \theta_q$ son constantes ($\theta_q \neq 0$) y w_n es un ruido blanco con media 0 y varianza σ_w^2 .

El proceso se puede reescribir de la siguiente forma:

$$X_t = \theta(B)w_t$$

Donde B es el parámetro de retardo.

Definición: El operador de promedio móvil es

$$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \dots + \theta_q B^q$$

El proceso de promedio móvil es estacionario para cualesquiera de los parámetros $\theta_1, \dots, \theta_q$, a diferencia del proceso autoregresivo.

Aquí está un ejemplo de un modelo MA(1) en Python con una gráfica para ayudar a entender mejor el concepto:

```
import numpy as np
import matplotlib.pyplot as plt

# Definimos nuestros parámetros
theta = 0.5 # coeficiente MA
sigma = 0.1 # varianza

# Definimos una matriz para almacenar nuestros valores generados
X = np.empty(100)

# Generamos valores aleatorios para nuestro ruido blanco
e = np.random.normal(0, sigma, 100)

# Generamos los valores de la serie temporal usando la ecuación MA(1)
for t in range(1, len(X)):
    X[t] = theta*e[t-1] + e[t]

# Graficamos nuestra serie temporal
plt.plot(X)
plt.show()
```

Teoría

▼ Modelos ARMA

$$y_t = \underbrace{\sum^p \phi y_{t-i}}_{\text{AR}} + \underbrace{\sum^q \beta_i \epsilon_{t-i}}_{\text{MA}} + \epsilon_t$$

Por ejemplo un proceso AR de orden uno AR(1).

Definición. El modelo autoregresivo-movimiento promedio (ARMA) es un modelo estadístico para series de tiempo que combina características de un modelo autoregresivo (AR) y un modelo de promedio móvil (MA).

$$ARMA(P, Q) : y \sim AR(p) + MA(q)$$

Estos modelos se usan para capturar la dependencia entre los puntos de datos en una serie temporal. Un modelo $ARMA(p, q)$ es un proceso estacionario, donde dado un proceso estocástico discreto (PED), tenemos que $PED = y_1, y_2, y_3, \dots, y_\infty$ obtenemos y_t, y_{t-k} de la siguiente forma:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \dots + \phi_p X_{t-p} + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \dots + \theta_q w_{t-q} + w_t$$

donde X_t es estacionario, $\phi_1, \phi_2, \dots, \phi_p$ y $\theta_1, \theta_2, \dots, \theta_q$ son constantes ($\phi_p \neq 0$ y $\theta_q \neq 0$), y w_n es un ruido blanco con media 0 y varianza σ_w^2 .

Los parámetros p y q se conocen como los órdenes *autoregresivo* y de promedio móvil, respectivamente. Si X_t tiene media μ distinta de 0, ponemos $\alpha = \mu(1 - \phi_1 - \phi_2 - \dots - \phi_p)$ y escribimos el modelo como

»

Diremos que este es un proceso estacionario si sus propiedades dependen de la distancia que los separa k y no del proceso temporal t .

Tenemos dos tipos de estacionariedad.

▼ Estacionariedad estricta y débil

Como se menciona una serie es estacionaria si su media y varianza son constantes en el tiempo y si el valor de la covarianza entre 2 periodos depende sólo de la distancia o rezago entre los tiempos.

Sea X_t una serie de tiempo con las siguientes propiedades se dice que es debilmente estacionaria o simplemente estacionaria:

- $E(Y_t) = E(Y_{t+k}) = \mu$
- $Var(Y_t) = Var(Y_{t+k}) = \sigma^2$
- $Cov(Y_t, Y_{t+k}) = \gamma_k$

En la práctica esta condición es imposible de verificar, pues requeriría conocer todas las distribuciones finito-dimensionales

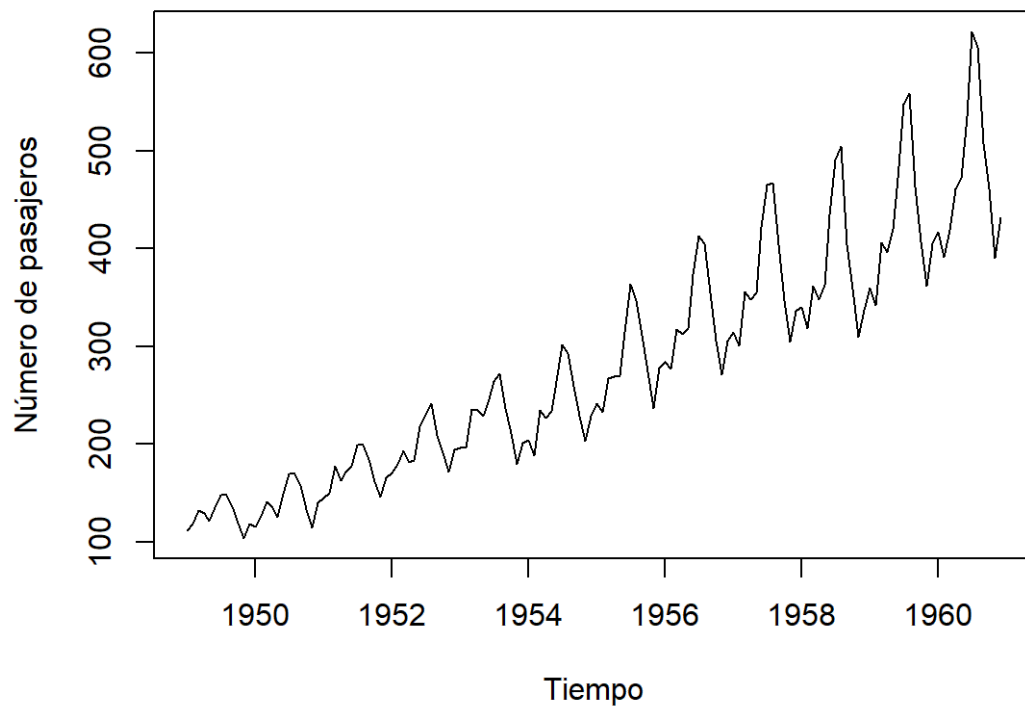
del proceso. Por ello se usa una condición más débil, que sólo pone condiciones en los dos primeros momentos de las variables que forman el proceso.

La condición adicional para que una serie de tiempo sea fuertemente estacionaria es que la distribución conjunta $(X_{t_1}, \dots, X_{t_k})$ es la misma que la de $(X_{t_1+h}, \dots, X_{t_k+h})$ es decir, la distribución sólo depende de la diferencia de tiempo h y no del tiempo (t_1, \dots, t_k)

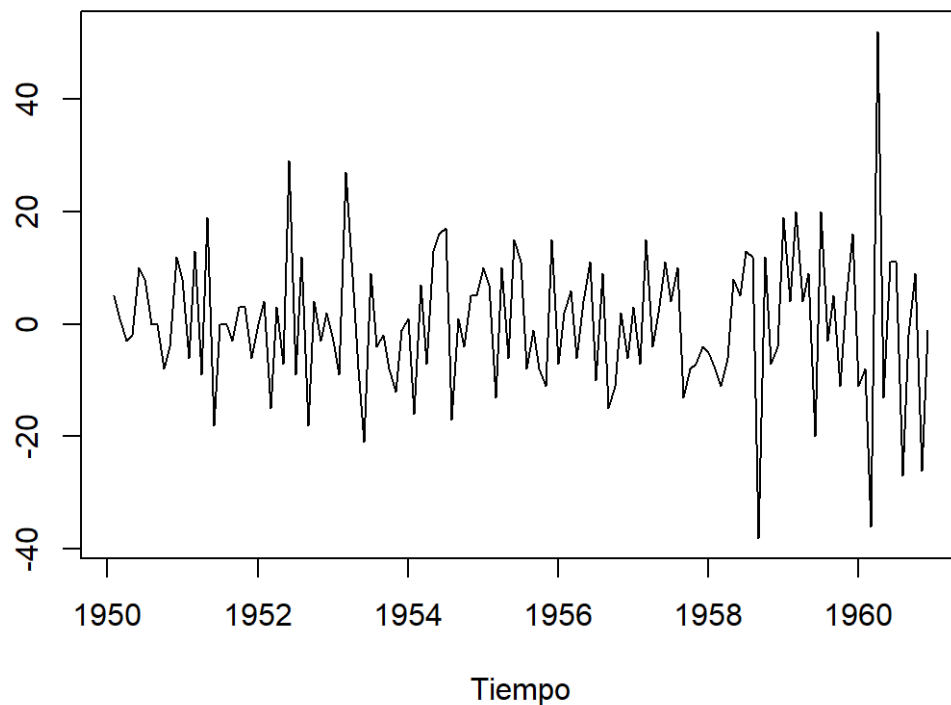
- $f_{yt}(Z) = f_{y-k}(Z)$

Es importante mencionar que Fuertemente estacionaria no implica debilmente estacionaria ya que en la fuerte no se usa el supuesto de que la varianza sea constante en el tiempo.

Se muestran las gráficas de una serie de tiempo que modela el número de pasajeros mensualmente de una aerolínea de 1949 a 1960.



Serie de tiempo no estacionaria.



Serie de tiempo estacionaria

▼ Ejemplo

Un buen ejemplo de una serie de tiempo **estacionaria débil** es una serie de tiempo que contiene una tendencia lineal o cíclica. Estas series de tiempo cambian a lo largo de un periodo de tiempo, pero su media permanece constante.

Por ejemplo, consideremos la siguiente serie de tiempo con una tendencia lineal creciente:

$$y_t = \mu + \beta t + \epsilon_t$$

Donde t representa el periodo de tiempo y ϵ_t es un ruido blanco.

Esta serie de tiempo es débilmente estacionaria, ya que su media (μ) es constante, pero la varianza y la covarianza cambian a lo largo del tiempo.

Un ejemplo de una serie de tiempo estrictamente estacionaria es una serie de tiempo que contiene un patrón de media constante, varianza constante y covarianza constante. Esto significa que los valores de la serie temporal permanecen alrededor de una línea de tendencia media y la varianza no cambia a lo largo de la serie temporal.

Por ejemplo, una serie de tiempo ARMA (2,2) se considera estrictamente estacionaria, ya que cumple con los tres criterios de estacionariedad.

Las series de tiempo que son estacionarias, se pueden modelar a travez de especificaciones ARMA(P,Q).

Ejemplo de un modelo ARMA(1,1) en Python con una gráfica para ayudar a entender mejor el concepto:

```
import numpy as np
import matplotlib.pyplot as plt

# Definimos nuestros parámetros
phi = 0.5 # coeficiente AR
theta = 0.5 # coeficiente MA
sigma = 0.1 # varianza

# Definimos una matriz para almacenar nuestros valores generados
X = np.empty(100)

# Generamos valores aleatorios para nuestro ruido blanco
e = np.random.normal(0, sigma, 100)

# Generamos los valores de la serie temporal usando la ecuación ARMA(1,1)
for t in range(1, len(X)):
    X[t] = phi*X[t-1] + theta*e[t-1] + e[t]

# Graficamos nuestra serie temporal
plt.plot(X)
plt.show()
```

¿Como saber si un X_t es estacionario?

Para saber el orden del modelo ARMA(p,q) podemos utilizar la siguiente:

la función de autocorrelación simple es la siguiente:

$$P_k = \frac{\text{cov}(X_t - X_{t-k})}{\sqrt{\text{var}(X_t)\text{var}(X_{t-k})}} = \frac{\gamma_k}{\gamma_0}$$

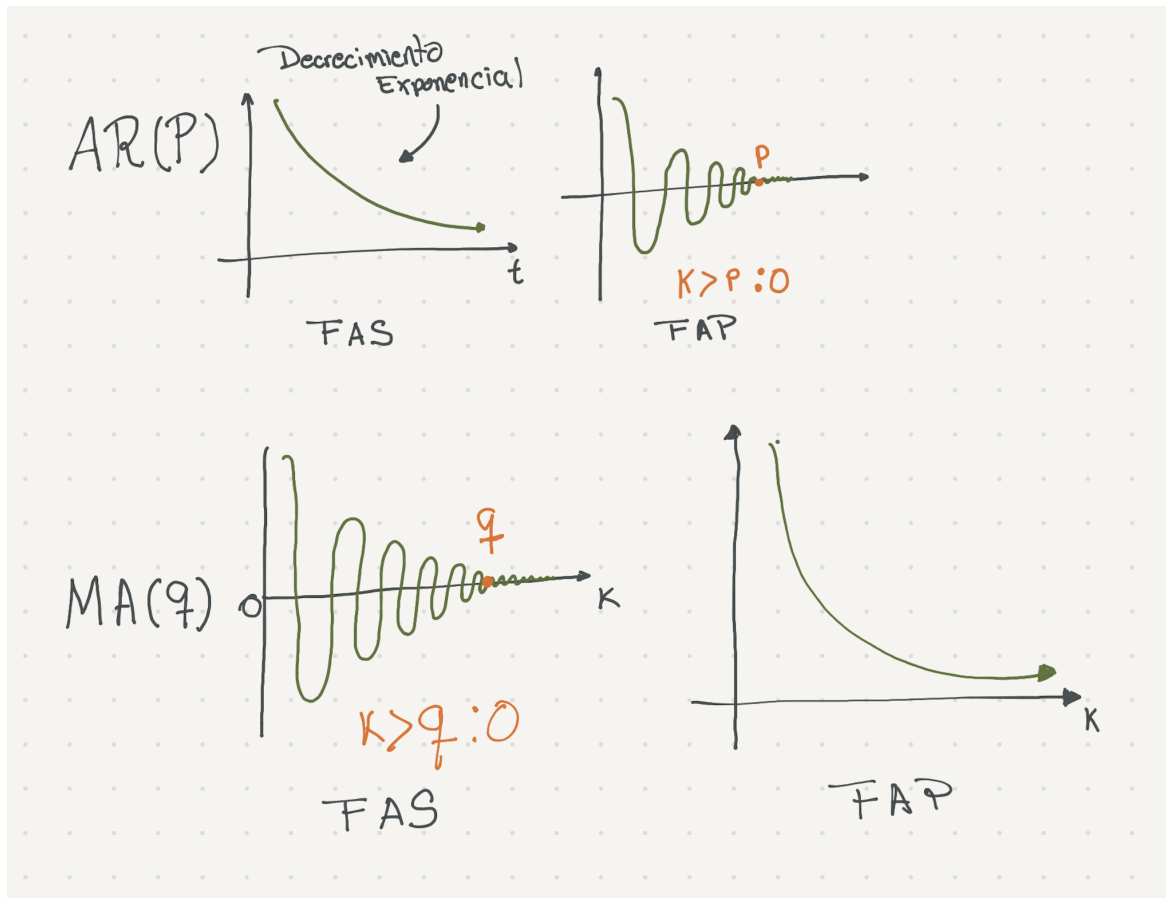
k son los lags(rezagos).

La función de autocorrelación parcial es la siguiente:

$$\phi_k = X_t = aX_{t-1} + bX_{t-2} + \dots + \phi_k X_{t-k} + w_t$$

con $a \neq \phi_1$ y $b \neq \phi_2$ que son diferentes a los coeficientes de la función parcial de primer orden . Mientras que el coeficiente de segundo orden es el coeficiente temporal rezagado en dos periodos.

¿Como saber si X_t sigue un proceso AR o MA... 🤔?



La autocorrelación parcial determina el orden de AR y La autocorrelación simple determina el orden de MA.

Un modelo autoregresivo de orden p , denotado por $AR(p)$, tiene la siguiente forma:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \dots + \phi_p X_{t-p} + w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \dots + \theta_q w_{t-q}$$

donde X_t es estacionario, $\phi_1, \phi_2, \dots, \phi_p$ y $\theta_1, \theta_2, \dots, \theta_q$ son constantes y w_n es un ruido blanco con media 0 y varianza σ_w^2 .

Los modelos ARMA son útiles para la predicción de series de tiempo, es decir, para generar pronósticos de la serie de tiempo basados en los datos históricos de la misma.

Esto implica que los modelos ARMA son útiles para predecir el futuro de una serie de tiempo, así como también para explicar los patrones subyacentes en una serie temporal.

Por ejemplo, un modelo ARMA se puede usar para predecir los precios futuros de un activo a partir de los precios pasados del mismo. Esto se logra al usar los modelos para capturar la dependencia entre los precios pasados y futuros del activo.

A continuación se muestra un ejemplo en Python de una serie ARMA. Se crea una serie de tiempo con la función `arma_generate_sample` de `statsmodels` que genera una serie de tiempo de ruido blanco con un modelo ARMA de orden (2,2).

```
import numpy as np
from statsmodels.tsa.arima_process import arma_generate_sample
np.random.seed(0)
arparams = np.array([0.75, -0.25])
maparams = np.array([0.65, 0.35])
ar = np.r_[1, -arparams] # add zero-lag and negate
ma = np.r_[1, maparams] # add zero-lag
y = arma_generate_sample(ar, ma, 100)
```

El modelo ARMA es un proceso estacionario.

▼ Modelos SARIMA

SARIMA

Definición. El modelo SARIMA (o modelo estacional autorregresivo integrado de media móvil) es un modelo estadístico para series de tiempo que combina características de un modelo autorregresivo (AR), un modelo de promedio móvil (MA) y una componente estacional.

Estos modelos se usan para capturar la dependencia entre los puntos de datos en una serie temporal. Un modelo $SARIMA(p, d, q)$ es un proceso estacionario, donde dado un proceso estocástico discreto (PED), tenemos que $PED = y_1, y_2, y_3, \dots, y_\infty$ obtenemos y_t, y_{t-k} de la siguiente forma:

»

donde X_t es estacionario, $\phi_1, \phi_2, \dots, \phi_p$ y $\theta_1, \theta_2, \dots, \theta_q$ son constantes ($\phi_p \neq 0$ y $\theta_q \neq 0$), $\phi_{s1}, \phi_{s2}, \dots, \phi_{sp}$ y $\theta_{s1}, \theta_{s2}, \dots, \theta_{sq}$ son constantes estacionales ($\phi_{sp} \neq 0$ y $\theta_{sq} \neq 0$), y w_n es un ruido blanco con media 0 y varianza σ_w^2 .

A continuación se muestra un ejemplo en Python de una serie SARIMA.

Se crea una serie de tiempo con la función `sarimax` de `statsmodels` que genera una serie de tiempo de ruido blanco con un modelo SARIMA de orden (2,1,2)(1,1,1,12).

```
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Definimos nuestros parámetros
arparams = np.array([0.75, -0.25])
maparams = np.array([0.65, 0.35])
sarp = np.array([0.25, 0.15])
samap = np.array([0.45, 0.35])

# Definimos una matriz para almacenar nuestros valores generados
X = np.empty(100)

# Generamos valores aleatorios para nuestro ruido blanco
e = np.random.normal(0, 0.1, 100)

# Generamos los valores de la serie temporal usando la ecuación SARIMA(2,1,2)(1,1,1,12)
sarimax_model = SARIMAX(X, order=(2,1,2), seasonal_order=(1,1,1,12))
sarimax_model_fit = sarimax_model.fit(dispatch=False)

# Graficamos nuestra serie temporal
plt.plot(X)
plt.show()
```

Aquí está un ejemplo de un modelo SARIMA(2,1,2)(1,1,1,12) en Python con una gráfica para ayudar a entender mejor el concepto:

Una serie temporal estacional es aquella que presenta una variación entre sus valores periódicamente, por ejemplo, una serie de tiempo que representa las temperaturas durante un año tendrá un patrón estacional con valores más altos en verano y más bajos en invierno.

Aquí está un ejemplo en Python de una serie estacional. Se crea una serie de tiempo con la función `seasonal_decompose` de `statsmodels` que genera una serie de tiempo con un patrón estacional.

```
import numpy as np
from statsmodels.tsa.seasonal import seasonal_decompose

# Definimos una matriz para almacenar nuestros valores generados
X = np.empty(100)

# Generamos los valores de la serie temporal con patrón estacional
for t in range(len(X)):
    X[t] = np.sin(2 * np.pi * t / 12)

#Aplicamos la descomposición estacional
result = seasonal_decompose(X, model='additive')

# Graficamos nuestra serie temporal
plt.plot(X)
plt.show()
```

El resultado de la descomposición estacional es una serie temporal compuesta de tres componentes: tendencia, estacional y ruido. Los valores de los componentes se pueden ver en la siguiente figura:

Descomposición estacional

Esta descomposición se puede usar para eliminar el patrón estacional de una serie temporal. Esto es útil cuando se desea ajustar un modelo a una serie temporal para predecir valores futuros, ya que se puede eliminar el patrón estacional de forma que sólo se estén considerando los patrones no estacionales.

- Ejemplos

Modelo SARIMA

El modelo SARIMA (Seasonal Autoregressive Integrated Moving Average) es un modelo utilizado en el análisis de series de tiempo para modelar la estacionalidad, tendencias y comportamientos aleatorios en una serie de tiempo.

En este ejemplo, utilizaremos la serie de tiempo de demanda de energía eléctrica de un conjunto de datos público.

Primero, importamos las bibliotecas necesarias y cargamos la serie de tiempo:

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Cargar la serie de tiempo
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv')
df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%d')
df.set_index('date', inplace=True)
```

Luego, dividimos la serie de tiempo en un conjunto de entrenamiento y uno de prueba:

```
# Dividir la serie de tiempo en entrenamiento y prueba
train_data = df[:'1994-12-01']
test_data = df['1995-01-01':]
```

A continuación, ajustamos un modelo SARIMA al conjunto de entrenamiento y hacemos predicciones para el conjunto de prueba:

```
# Ajustar un modelo SARIMA al conjunto de entrenamiento
model = SARIMAX(train_data, order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
model_fit = model.fit()

# Hacer predicciones en el conjunto de prueba
predictions = model_fit.predict(start='1995-01-01', end='1996-01-01')
```

Finalmente, graficamos las predicciones junto con los datos de prueba para evaluar el rendimiento del modelo:

```
# Graficar las predicciones y los datos de prueba
plt.plot(test_data)
```

```
plt.plot(predictions, color='red')
plt.show()
```

Modelo VAR

- Actividad ARMA y SARIMA

▼ 2.4 Pronósticos con modelos de vectores auto-regresivos (VAR)

Modelo VAR

El modelo VAR (Vector Autoregression) es un modelo utilizado en el análisis de series de tiempo multivariadas para modelar la dependencia entre varias series de tiempo.

Veamos el modelo estructural dinámico (Novales (2011)):

$$\begin{aligned}y_{1t} &= \alpha_{10} + \alpha_{11}y_{2t} + \alpha_{12}y_{1t-1} + \alpha_{13}y_{2t-1} + \gamma_1'z_t + \varepsilon_{1t} \\y_{2t} &= \alpha_{20} + \alpha_{21}y_{1t} + \alpha_{22}y_{1t-1} + \alpha_{23}y_{2t-1} + \gamma_2'z_t + \varepsilon_{2t}\end{aligned}\tag{5.2}$$

donde y_{1t} , y_{2t} , son **variables estacionarias**, y ε_{1t} , ε_{2t} son procesos ruido blanco con esperanza cero y varianzas $\sigma_{\varepsilon_1}^2$, $\sigma_{\varepsilon_2}^2$ y covarianza σ_{12} .

El modelo (5.2) es de ecuaciones simultáneas con **dos variables endógenas** y_{1t} y y_{2t} y un vector z_t de **variables exógenas**.

Un *shock* sobre y_{2t} , en la forma de un valor no nulo de la **innovación** estructural ε_{2t} , afecta directamente a y_{2t} , pero también influye a y_{1t} a través de la presencia de y_{2t} como variable explicativa en la primera ecuación.

Además, este **efecto se propaga en el tiempo**, debido a la presencia de los valores rezagados de ambas variables como variables explicativas.

Las variables explicativas exógenas z_t **también pueden aparecer con rezagos en el modelo**. Por ejemplo, z_t podría ser una tendencia determinista o que recoja la estacionalidad. z_t también puede representar variables tal que $E(z_{t-s}\varepsilon_{1t}) = E(z_{t-s}\varepsilon_{2t}) = 0 \forall s$. Por ejemplo, el precio de barril de petróleo que se determina en mercados internacionales mientras y_{1t} y y_{2t} son variables de la macroeconomía ecuatoriana.

En este ejemplo, utilizaremos un conjunto de datos que contiene información sobre el índice de precios al consumidor (IPC) y la tasa de interés de los bonos del Tesoro de EE.UU.

Primero, importamos las bibliotecas necesarias y cargamos el conjunto de datos:

```
import pandas as pd
from statsmodels.tsa.vector_ar.var_model import VAR

# Cargar el conjunto de datos
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Raotbl6.csv')
df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%d')
df.set_index('date', inplace=True)
```

Luego, dividimos el conjunto de datos en un conjunto de entrenamiento y uno de prueba:

```
# Dividir el conjunto de datos en entrenamiento y prueba
train_data = df['2014']
test_data = df['2015:']
```

A continuación, ajustamos un modelo VAR al conjunto de entrenamiento y hacemos predicciones para el conjunto de prueba:

```
# Ajustar un modelo VAR al conjunto de entrenamiento
model = VAR(train_data)
model_fit = model.fit()
```

▼ Extras

Un filtro de Kalman es un algoritmo de estimación utilizado para estimar el estado de un sistema dinámico basado en mediciones observadas. Utiliza un modelo matemático para representar el sistema para el que se realizan las mediciones, y un proceso de estimación para combinar las mediciones observadas con la información previa para calcular una

estimación actualizada del estado del sistema. El filtro de Kalman es una herramienta útil para estimar el estado de un sistema cuando hay ruido en las mediciones.

https://youtu.be/ZYexI6_zUMk

Notas adicionales

Time Series Analysis, Regression and Forecasting – With tutorials in Python

(6) Introducción a las Series de Tiempo Estacionarias - YouTube

3 Modelización de Series Univariantes: (S)ARIMA | Series de Tiempo