



Series Temporales y Minería de Flujos de Datos

Departamento de Ciencias de la Computación e
Inteligencia Artificial

Universidad de Granada



UNIVERSIDAD
DE GRANADA

**Prácticas de Series Temporales y
Minería de Flujos de Datos:**

Series Temporales

Prof. Manuel Pegalajar Cuéllar



01

Introducción

02

Predicción de series temporales

03

Análisis de series temporales

04

Modelo ARIMA

Introducción



Introducción

Definición

Una serie temporal $X(t) = \{x(1), x(2), x(3), \dots, x(t)\}$ es una secuencia de observaciones de un fenómeno dado, muestreadas periódicamente y indexadas en el tiempo.



Ejemplos :

Precios del mercado de valores
Uso de electricidad

Históricos meteorológicos
Monitorización de actividades

Frecuencia cardíaca

Etc.



Predicción de series temporales



Predicción de series temporales

■ El problema de la predicción (forecasting)

Sea $X(t) = \{x(1), x(2), \dots, x(t)\}$ una secuencia de datos históricos de un fenómeno dado, muestreados periódicamente e indexados en el tiempo. El problema de la predicción (forecasting) trata de predecir $x(t+1)$ como:

$$x(t+1) = f(x(t), x(t-1), \dots, x(t-T), w) + \varepsilon(t+1)$$

Donde:

- $f()$ es un modelo (hipótesis)
- T es un horizonte temporal (pasado)
- $w=(w_1, \dots, w_n)$ son parámetros del modelo
- $\varepsilon(t+1)$ es un error de aproximación para $x(t+1)$ usando $f()$

■ Ejemplo

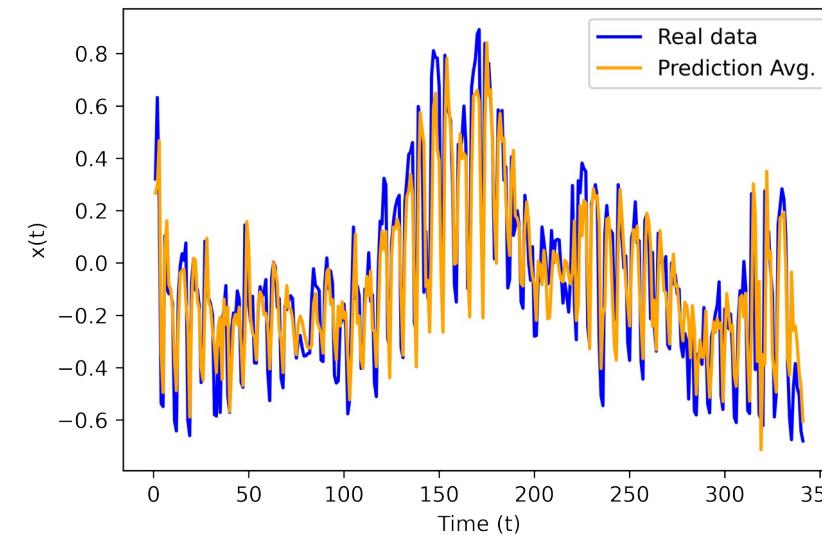
Predecir el consumo energético de un edificio para el próximo lunes, teniendo como datos el consumo diario de energía de la semana anterior ($T=7$).



Predicción de series temporales

■ Predicción (prediction) vs. Predicción (forecasting)

- **Prediction** : Tratar de « adivinar » un valor de una variable objetivo (no necesariamente dependiente del tiempo). Por ejemplo : Predecir la clase de un dígito escrito dada una imagen de entrada, predecir presencia/ausencia de cáncer dados atributos médicos de entrada, etc.
- **Forecasting** : Terminología de series temporales. Se trata de una predicción sobre el futuro.



Predicción de series temporales

■ Análisis de regresión vs. Forecasting

- **Análisis de regresión** : Dada una secuencia de observaciones $X(t) = \{x(1), x(2), \dots, x(t)\}$, $x(i) \in \mathbb{R}$, y una hipótesis de modelo $f: \mathbb{N} \rightarrow \mathbb{R}$ parametrizado por $w = (w_1, w_2, \dots, w_n)$, el problema consiste en encontrar los parámetros w óptimos que minimicen una función de coste sobre $f(t, w)$ y $x(t)$ (Por ejemplo la normal L2).

$$\min_w \sum_{i=1}^t (f(i, w) - x(i))^2$$

Estos parámetros óptimos se obtienen usualmente utilizando un procedimiento de optimización numérica estándar, como optimización por mínimos cuadrados.



Predicción de series temporales

■ Análisis de regresión vs. Forecasting

- **Ejemplo de análisis de regresión:** Dada la secuencia $\{x(1), x(2), x(3), x(4), x(5)\} = \{3.1, 3.98, 5.2, 5.8, 7.1\}$, y la hipótesis de modelo lineal :

$$f(t) = w_1 * t + w_2,$$

el análisis de regresión trata de aproximar $x(t) = f(t) + \varepsilon(t)$, donde $\varepsilon(t)$ es el error de estimación del valor $x(t)$.

En el ejemplo, una aproximación de los parámetros óptimos sería $w_1 = 1$, $w_2 = 2$.

El análisis de regresión no considera la dependencia existente sobre los datos históricos pasados (anteriores).



Predictión de series temporales

Análisis de regresión vs. Forecasting

- Ejemplo de análisis de regresión en Python (notebook [RegressionVsTimeSeries](#)):

Data example generation

We create an example time series $x(t)$ containing N samples. The synthetic data is just a linear model $x(t) = a * t + b + \epsilon(t)$, where $\epsilon(t)$ is gaussian noise with standard deviation σ_{error} .

```
N= 10 # Number of sampled data
sigmaError= 0.2 # To include noise in data from a normal N(0, sigmaError)

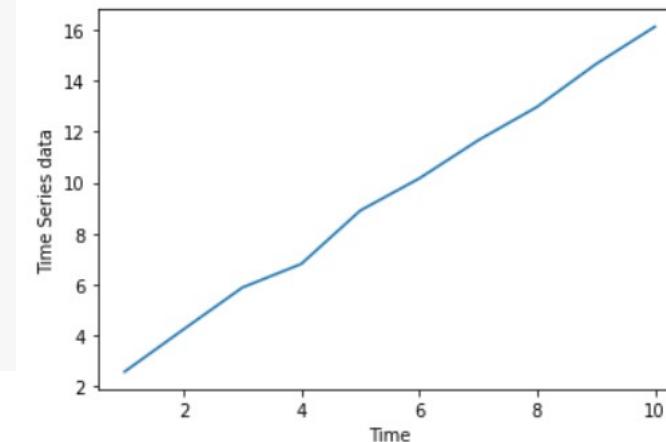
# Data model: Linear with slope and intercept
Slope= 1.5 # Slope of the synthetic data
Intercept= 1 # Intercept of the synthetic data

# Synthetic generation of data + noise
t= np.linspace(1.0, 10.0, num=N) # Sampling times
t= np.array(range(1, N+1), dtype=float) # Create time indices

# Create the dataset
x= Slope*t + Intercept

# Add noise
x+= sigmaError*np.random.randn(N)

# plot time series data
plt.xlabel('Time')
plt.ylabel('Time Series data')
plt.plot(t, x)
```



Predictión de series temporales

Análisis de regresión vs. Forecasting

- Ejemplo de análisis de regresión en Python (notebook [RegressionVsTimeSeries](#)):

Regression analysis

In this case, we use a model hypothesis $x(t) = f(t, w) + \epsilon(t)$, with $f(t, w) = w_1 * t + w_2$.

```
# First, define our model hypothesis to predict x(t)= f(t, w1, w2)
def LinearModel(t, w1, w2):
    return w1*t+w2

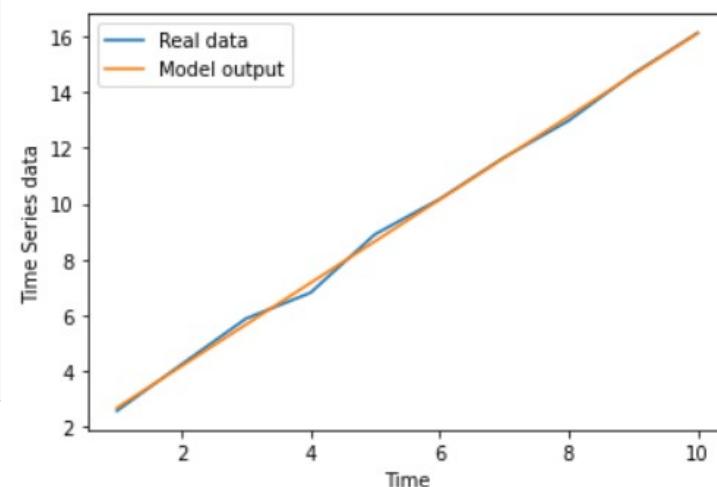
# Then, optimize the model parameters of LinearModel by least squares optimization
#   using t as input and x(t) as output
[w1, w2], cov = curve_fit(LinearModel, t, x) # Returns w: parameters and cov: Covariance matrix
print('Values of Optimized parameters. w1={:.3f}; w2={:.3f}'.format(w1, w2))

# plot time series data
plt.xlabel('Time')
plt.ylabel('Time Series data')
plt.plot(t, x)

# plot generated model
f_ra= LinearModel(t, w1, w2)
plt.plot(t, f_ra)
plt.legend(['Real data', 'Model output'])

# Find MSE approximation error
MSE_ra= np.sum( (f_ra-x)**2 )/N
print('The Mean Squared Error is : {:.3f}'.format(MSE_ra))
```

Values of Optimized parameters. w1=1.491; w2=1.201
The Mean Squared Error is : 0.027



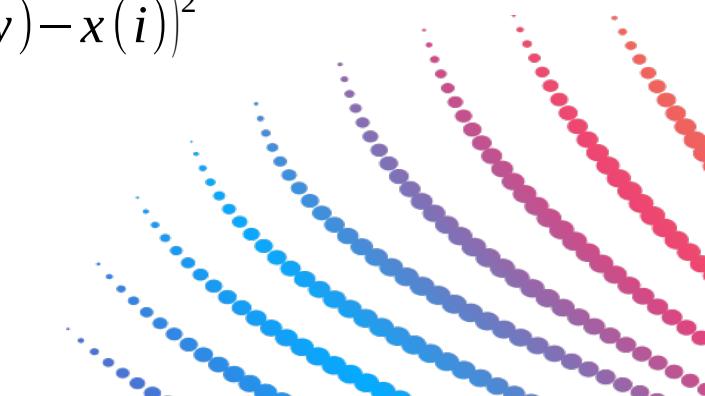
Predictión de series temporales

■ Análisis de regresión vs. Forecasting

- **Predictión (forecasting) de series temporales :** Dada una secuencia de observaciones $X(t)=\{x(1), x(2), \dots, x(t)\}$, $x(i) \in \mathbb{R}$, un **horizonte temporal T** , y una hipótesis de modelo $f: \mathbb{R}^T \rightarrow \mathbb{R}$ parametrizado por $w=(w_1, w_2, \dots, w_n)$, el problema consiste en encontrar los parámetros óptimos w que minimicen la función de coste :
 - $f(x(t-1), x(t-2), \dots, x(t-T), w)$ and $x(t)$.

Como por ejemplo :

$$\min_w \sum_{i=T+1}^t (f(x(i-1), x(i-2), \dots, x(i-T), w) - x(i))^2$$



Predicción de series temporales

■ Análisis de regresión vs. Forecasting

- **Ejemplo de predicción de series temporales:** Dada la secuencia de observaciones $\{x(1), x(2), x(3), x(4), x(5)\} = \{3.1, 3.98, 5.2, 5.8, 7.1\}$, una hipótesis de modelo lineal con horizonte temporal $T=2$ y parámetros $w=(w_1, w_2, w_3)$:

$$f(x(t-1), x(t-2), w) = w_1 + w_2 x(t-1) + w_3 x(t-2)$$

- La predicción de series temporales trata de aproximar $x(t) = f(x(t-1), x(t-2), w) + \varepsilon(t)$, donde $\varepsilon(t)$ es el error de estimación del valor $x(t)$.

En el ejemplo, algunos posibles valores de aproximación al óptimo serían :

$$w_1 = 1, w_2 = 1, w_3 = 0$$

Pero también :

$$w_1 = 2, w_2 = 0, w_3 = 1$$



Predictión de series temporales

Análisis de regresión vs. Forecasting

- Predicción de series temporales en Python (notebook [RegressionVsTimeSeries](#)):

Time Series

In this case, we use a model hypothesis $x(t) = f(x(t-1), x(t-2), w) + \epsilon(t)$, with $f(x(t-1), x(t-2), w) = w_1 + w_2x(t-1) + w_3x(t-2)$.

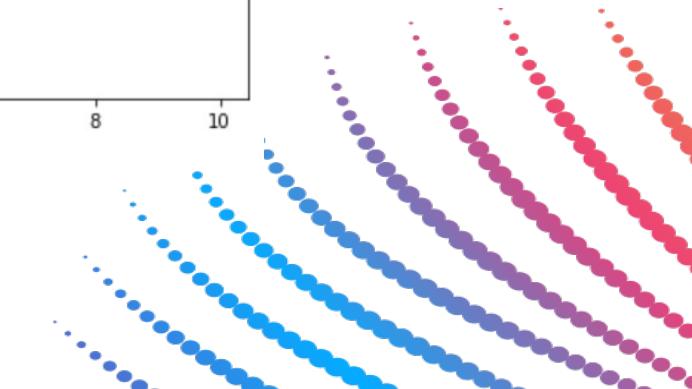
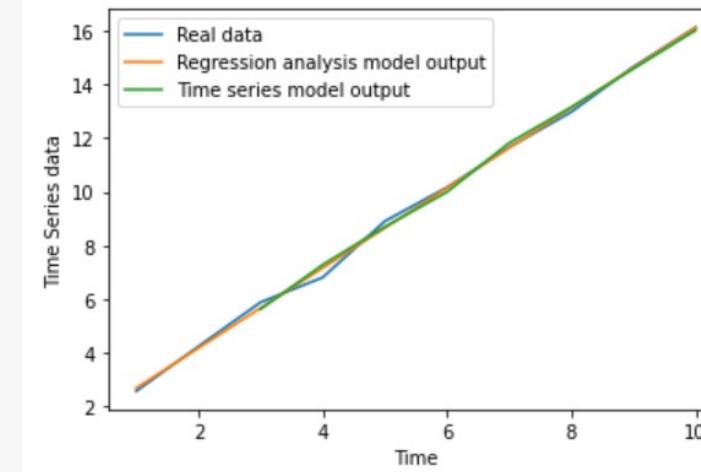
```
# First, define our model hypothesis to predict x(t)= f(x(t-1), ..., x(t-T), w1, w2, w3)
# Inputs x= array of size N-T x T . x[t,:]=[x(t-1), x(t-2), ..., x(t-T)]
def TimeSeriesLinearModel(x, w1, w2, w3):
    return w1+w2*x[:, 0]+w3*x[:, 1]

T= 2 # History time horizon to predict x(t)= f(x(t-1), ..., x(t-T), w)

# Format time series input/output data
# Total of T+1 columns: (x(t-T), x(t-T+1), ..., x(t-1), target x(t))
tsx= []
for i in range(T+1):
    if i != T:
        tsx.append(x[i:(-T+i)])
    else:
        tsx.append(x[i:])
tsx= np.stack(tsx).T

print('Original Time Series data:\n', x)
print('Time series formatted dataset:\n', tsx)
print('Time series input data {x(t-1), x(t-2)}:\n', tsx[:, :T])
print('Time series output data {x(t)}:\n', tsx[:, T])

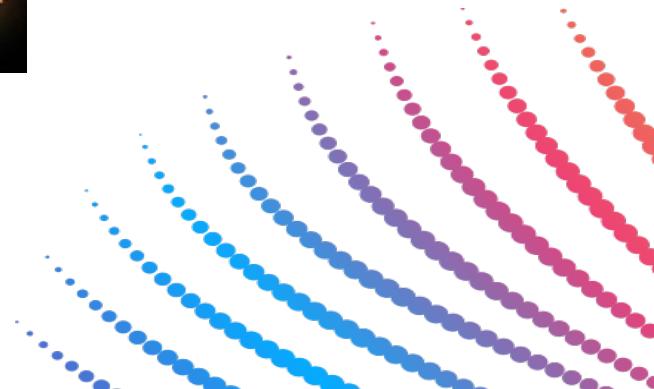
# Then, optimize the model parameters of TimeSeriesLinearModel by least squares optimization
# using x(t-1) and x(t-2) as input and x(t) as output
w, cov = curve_fit(TimeSeriesLinearModel, tsx[:, :T], tsx[:, T])
print('Values of Optimized parameters. w={}'.format(w))
```



Predicción de series temporales

■ ¿Se puede predecir cualquier cosa ?

- El grado de predicción sobre un fenómeno depende de... :
 - ✓ La cantidad de datos disponibles
 - ✓ La calidad de los datos (ruidos, valores perdidos...)
 - ✓ El conocimiento experto de los eventos/atributos que influyen sobre el fenómeno.
 - ✓ Si el conocimiento que tenemos sobre la predicción puede a su vez alterar el futuro.



Predicción de series temporales

■ Tipos de predicción basados en el horizonte temporal :

- **Short-term forecasting** (Predecir ventas del próximo mes)
- **Mid-term forecasting** (Estimar las ventas para los 6 próximos meses)
- **Long-term forecasting** (Estimar las ventas para los 3 próximos años)

■ Otros aspectos a considerar

- **Datos exógenos**
- **Series de datos multi-variantes**



Predictión de series temporales

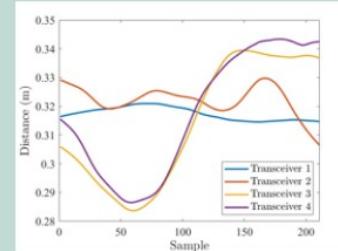
■ Series de datos multivariantes

Una observación sobre un fenómeno en un instante t , $x(t)$, no es sólo un valor escalar, sino una tupla de valores $x(t) = (x_1(t), x_2(t), \dots, x_k(t))$.

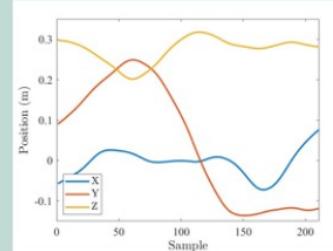
■ Ejemplo

- **Reconocimiento de gestos usando 4 sensores de ultrasonidos.**
 - Serie temporal : Secuencia de valores de distancia de cada sensor
$$x(t) = (d_1(t), d_2(t), d_3(t), d_4(t))$$

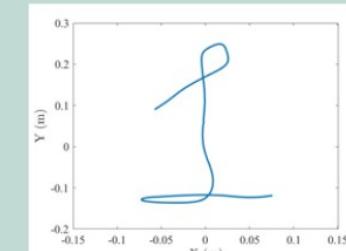
ToF and Distances calculation



3D coordinates calculation



Track definition and filtering



Predicción de series temporales

■ Series temporales con datos exógenos

Se usa una o más serie de datos adicionales $A(t)$, $B(t)$, ..., cuya información externa (exógena) no depende de la variable de estudio $X(t)$:

$$x(t+1) = f(X(t), A(t), B(t), \dots, w) + \varepsilon(t+1)$$

■ Ejemplo

- **Predicción del consumo energético de un edificio considerando :**
 - Datos históricos de consumo energético del edificio $X(t)$
 - Ocupación esperada del edificio $A(t)$ (día laboral, vacaciones...)
 - Información meteorológica (temperatura, ...) $B(t)$

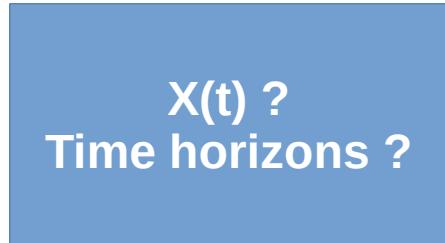


Predictión de series temporales

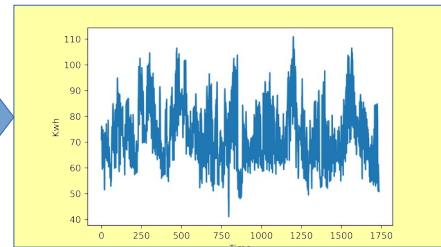
Flujo de procesamiento para predicción de series temporales

- 1) Planteamiento del problema
- 2) Obtención de datos y preprocesamiento
- 3) Análisis de datos exploratorio
- 4) Selección de modelo de predicción
- 5) Entrenamiento y validación

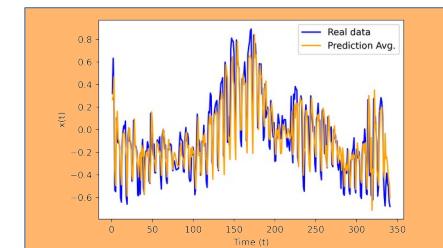
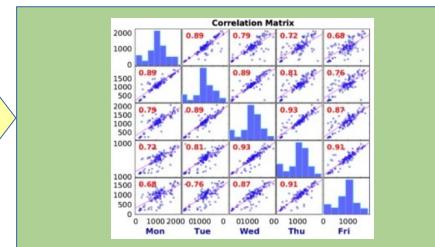
Problem statement



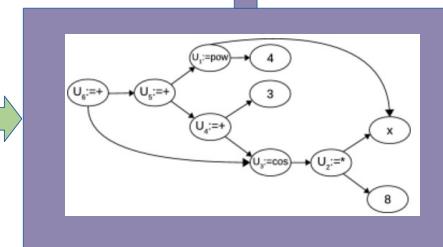
Data gathering



Explore data



Train & test



Model selection

Predictión de series temporales

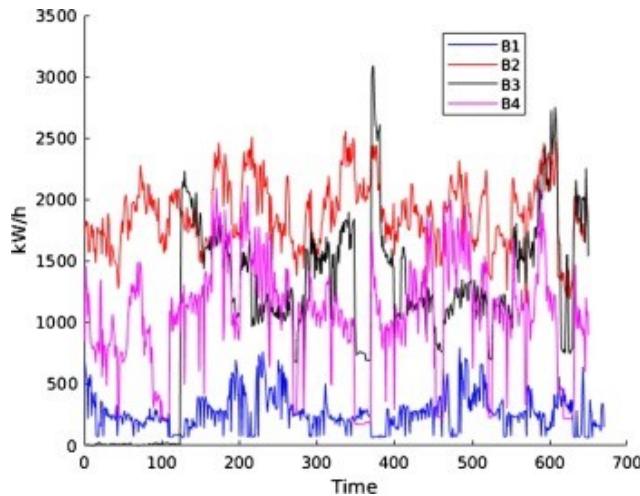
■ Habilidades del analista de datos :

- Conocimiento sobre técnicas de análisis exploratorio de datos
- Conocimiento sobre técnicas de preprocessamiento de datos
- Conocimiento de diversos modelos de predicción de series temporales
- Conocimiento sobre técnicas para entrenamiento de modelos
- Conocimiento sobre técnicas de validación y análisis de resultados

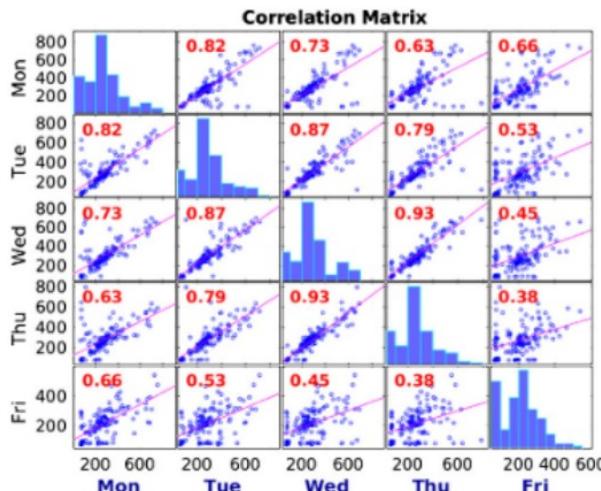


Predictión de series temporales

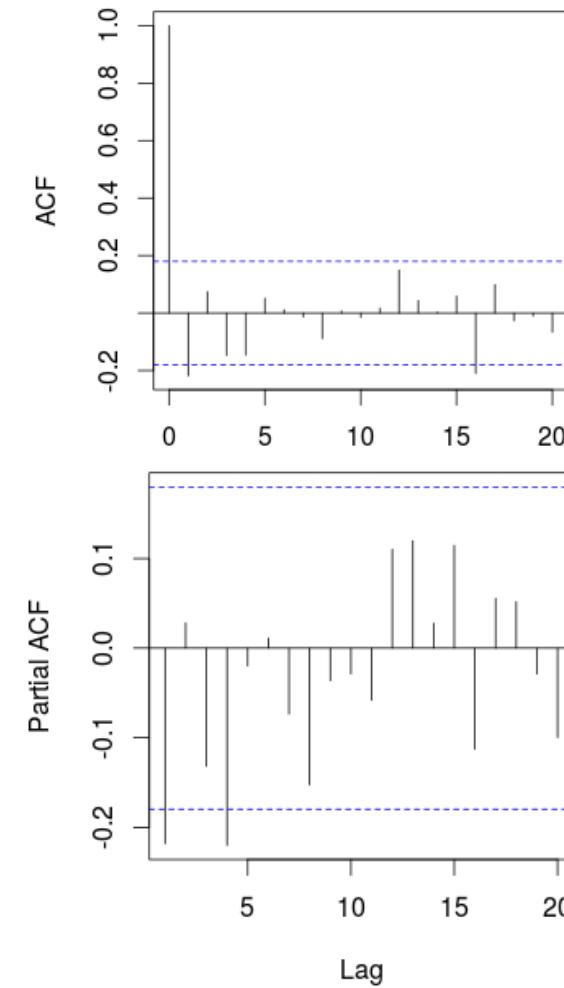
Herramientas de análisis exploratorio :



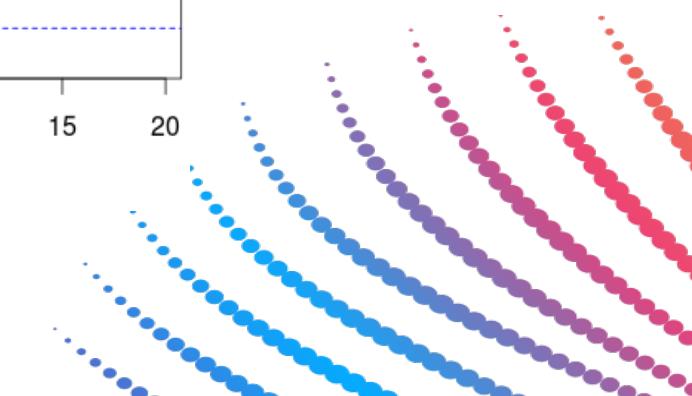
Time Series plot



Correlation plots
& Histograms



Auto-correlation
and Partial Auto-
correlation
functions



Predicción de series temporales

■ Métricas para dar soporte al análisis exploratorio :

- **Correlation coefficient r :**
$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}$$

- **Autocorrelation r_k :**
$$r_k = \frac{\sum_{t=k+1}^T (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2}$$

- **Partial Autocorrelation :**
$$\rho_k = \frac{\rho_1 - r_1 \frac{Cov[x_t, x_{t-k} | x_{t-1}, \dots, x_{t-k+1}]}{Var[x_t | x_{t-1}, \dots, x_{t-k}] Var[x_{t-k} | x_{t-1}, \dots, x_{t-k+1}]}}{\sqrt{Var[x_t | x_{t-1}, \dots, x_{t-k}] Var[x_{t-k} | x_{t-1}, \dots, x_{t-k+1}]}}$$

- **Además: estadísticos comunes como la media, desviación estándar...**



Predicción de series temporales

■ Herramientas de preprocesamiento de datos :

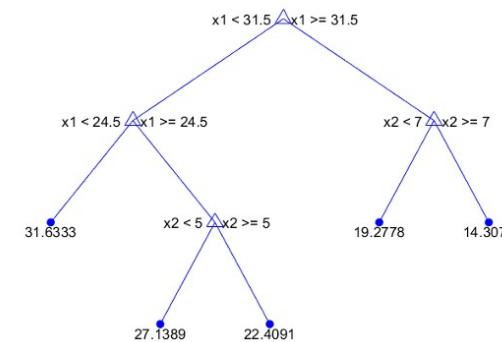
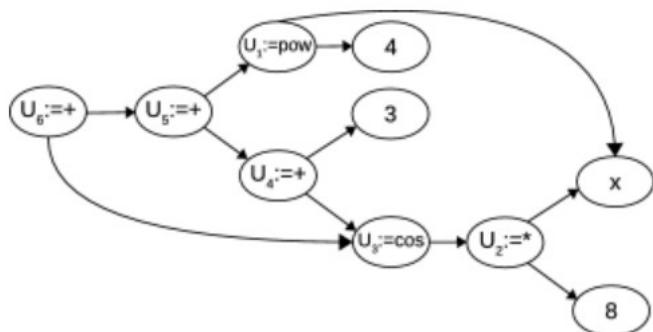
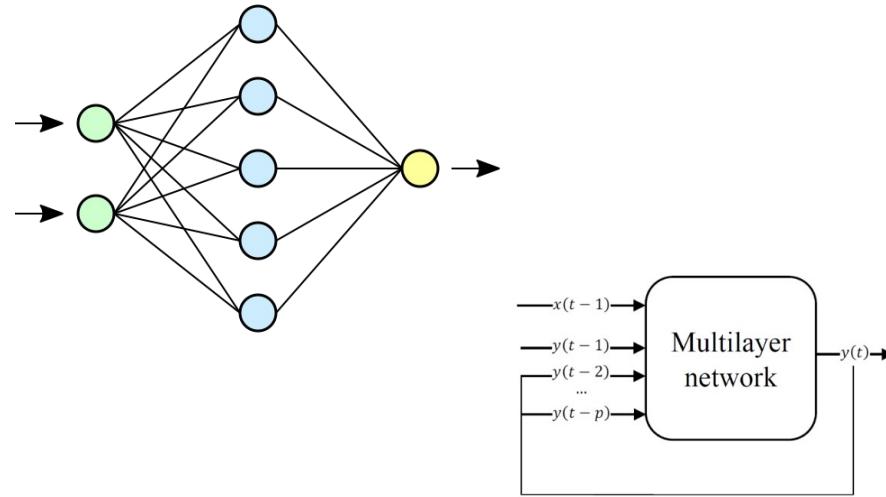
- Tratamiento de valores perdidos (averaging, regression...)
- Técnicas de transformación de datos :
 - Transformaciones de datos (logarítmicas, exponenciales, ...)
 - Transformaciones de espacios (Fourier, Laplace, wavelets...)
- Agregación de datos :
 - Agregación por calendario : diario, semanal, mensual... (promedio, máximo, mínimo...)



Predictión de series temporales

■ Modelos de predicción :

- Auto-regression (ARIMA)
- Feedforward/recurrent Neural networks
- Support Vector Machines
- Regression forests/decision trees
- Symbolic regression



Predictión de series temporales

■ Evaluación de modelos :

- **Precisión :**
 - Mean Square Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentual Error (MAPE), Percentage of error...
- **Rendimiento del modelo (bias, robustness):**
 - Análisis de residuos (random residuals, normal probability distribution, ...)

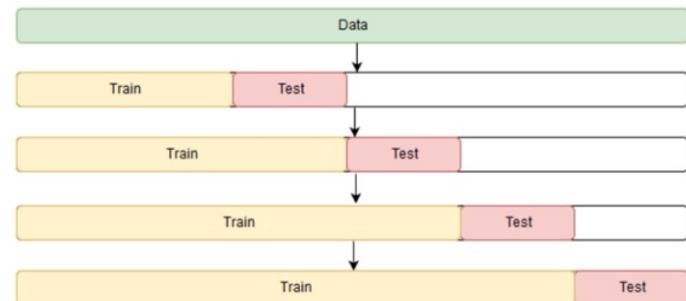


Predictión de series temporales

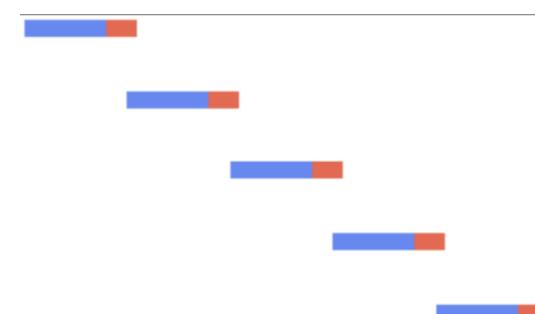
Métodos de validación :

- Time Series Cross-Validation
- Blocked Cross-Validation
- Last Block Validation

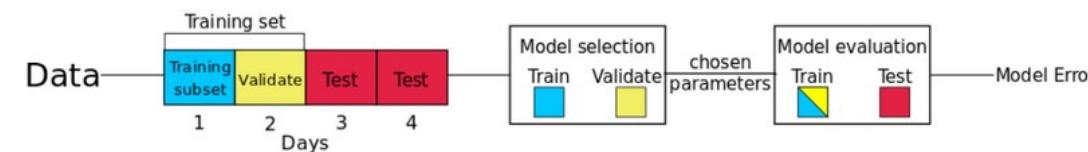
Cross-validation



Blocked cross-validation



Last Block validation



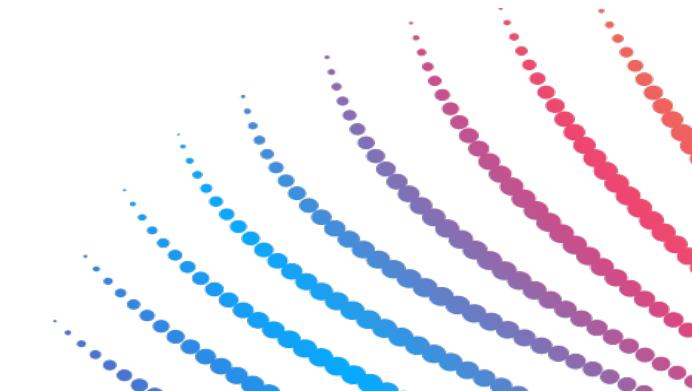
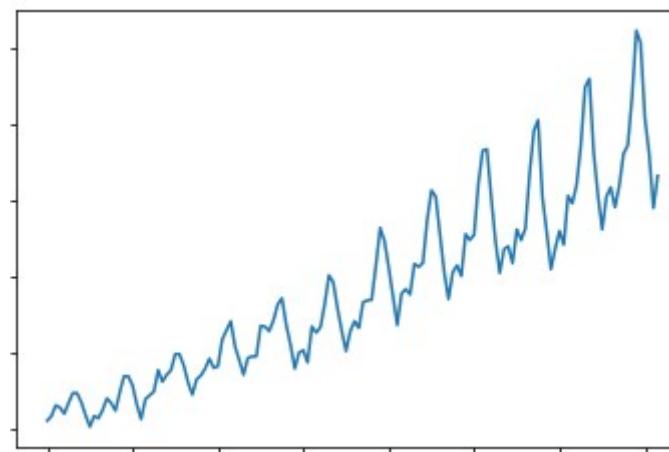
Análisis de series temporales



Análisis de series temporales

Ejercicio guiado :

- 1) **Dataset:** Número mensual de pasajeros de avión (en miles) desde 1949 hasta 1960. (**fichero AirPassengers.csv**).
- 2) **Objetivo :** Calcular el número de pasajeros en 1961.



Análisis de series temporales

■ Patrones usuales en el problema de Predicción:

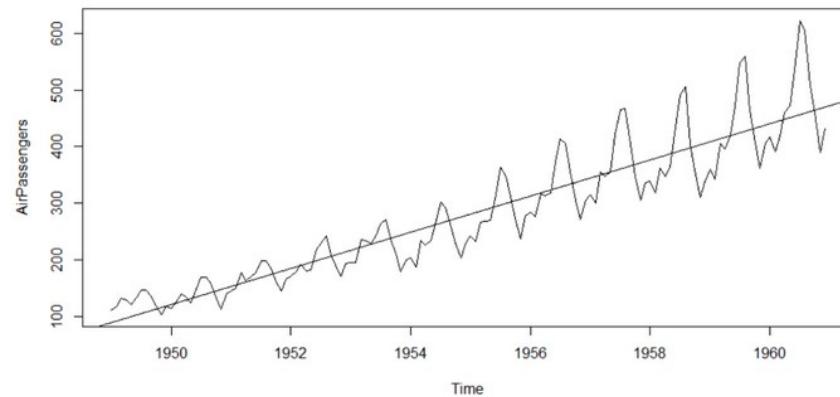
- **Trend (tendencia)** : Incremento/decremento de los valores de la serie, prolongado en el tiempo a largo plazo
- **Seasonality (estacionalidad)**: Los datos de la serie exhiben un comportamiento periódico.
- **Cycles (ciclos)** : Dentro de períodos de tiempo acotados, la serie exhibe un mismo comportamiento. En períodos de tiempo distintos, el comportamiento es distinto. **No confundir con estacionalidad** (en ciclos no tiene porqué existir periodicidad).



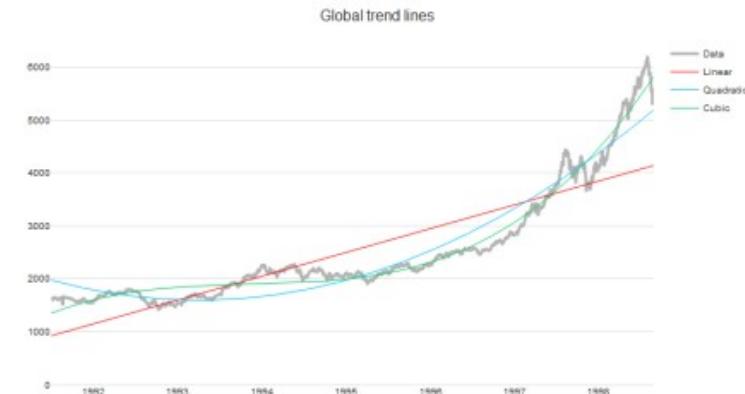
Análisis de series temporales

Ejemplos de tendencia

Linear Trend

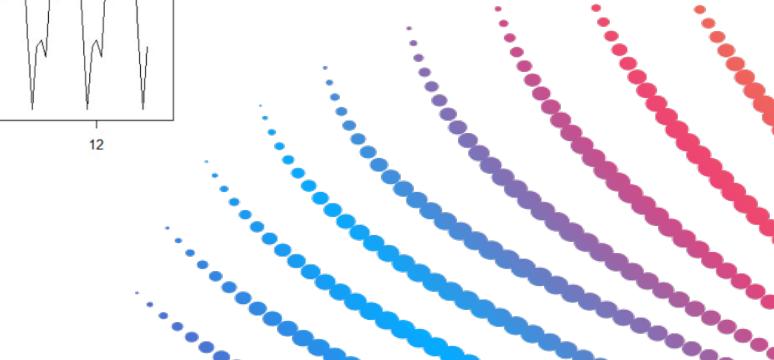
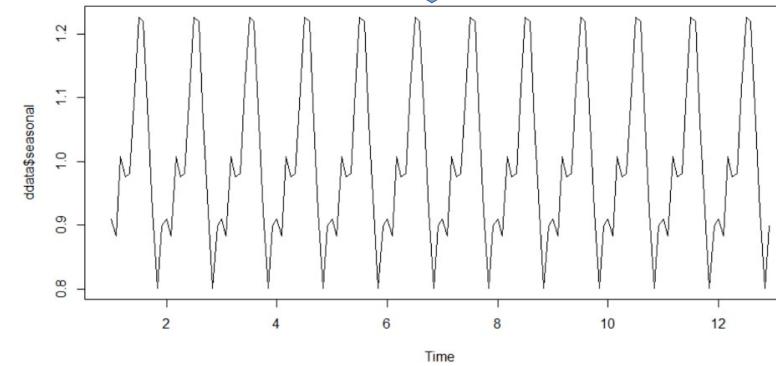
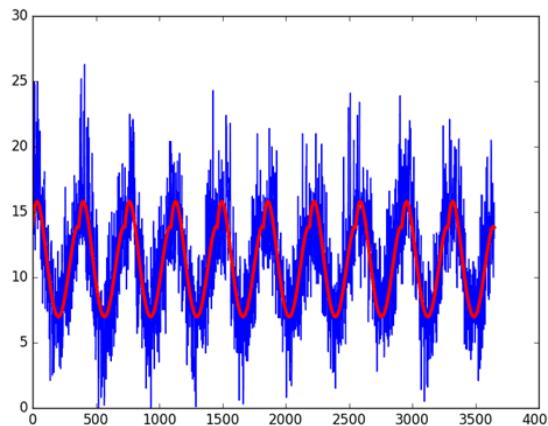
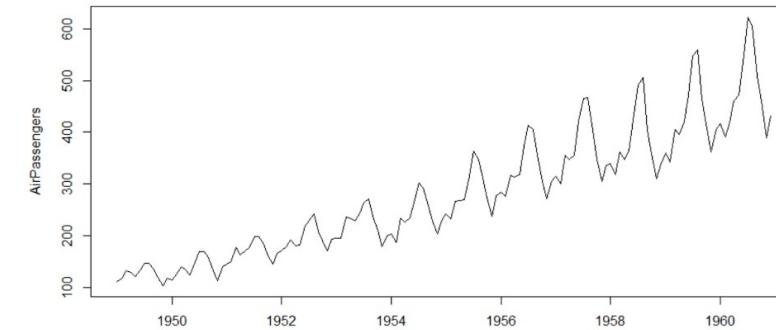
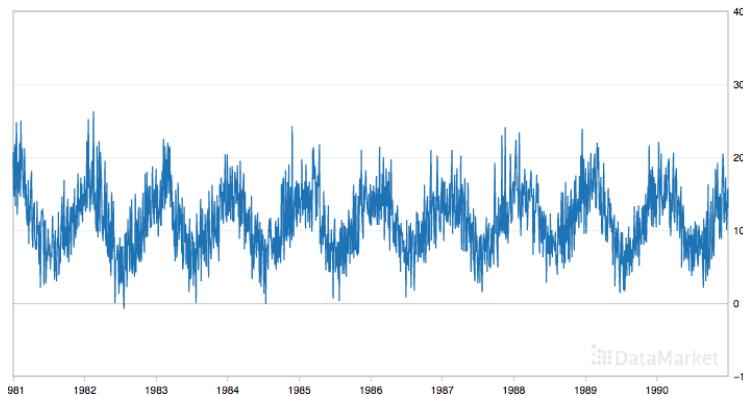


Polynomial Trend



Análisis de series temporales

Ejemplos de estacionalidad



Análisis de series temporales

Ejemplos de ciclos

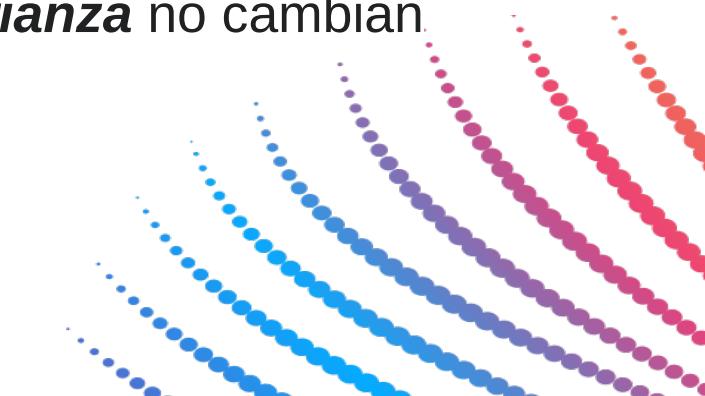


Análisis de series temporales

■ La metodología Box-Jenkins

Es una metodología de análisis de las más extendidas.

- **Componentes** : Asume que una serie temporal se corresponde con la agregación de 3 componentes básicas : Trend ($T(t)$), Seasonality ($S(t)$), y una componente irregular ($E(t)$)
- **Modelo aditivo** : $X(t) = T(t) + S(t) + E(t)$
- **Hipótesis de estacionariedad** : Los estadísticos *media* y *varianza* no cambian significativamente a lo largo del tiempo.



Análisis de series temporales

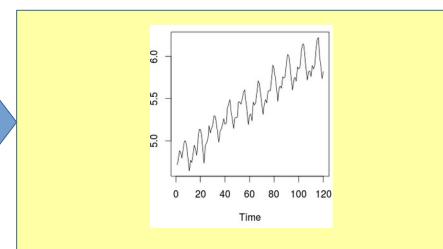
■ La metodología Box-Jenkins

Proceso de análisis con la metodología Box-Jenkins

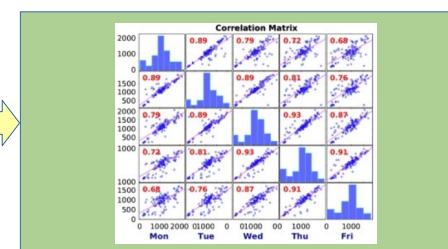
Problem statement



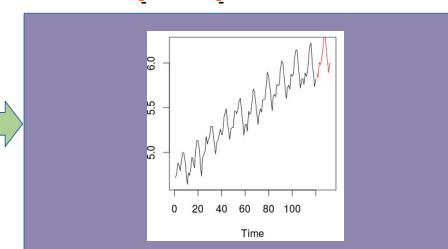
Data gathering



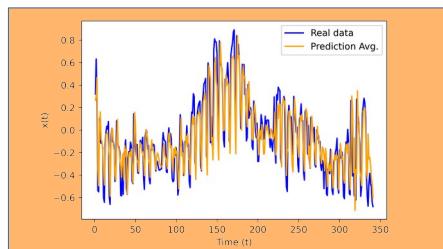
Explore data



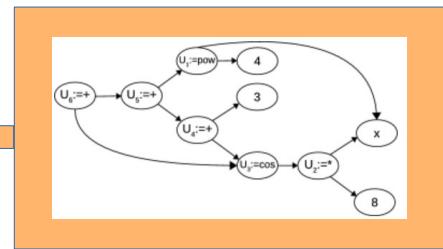
Data preprocessing



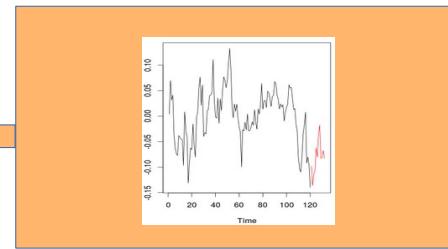
Model validation



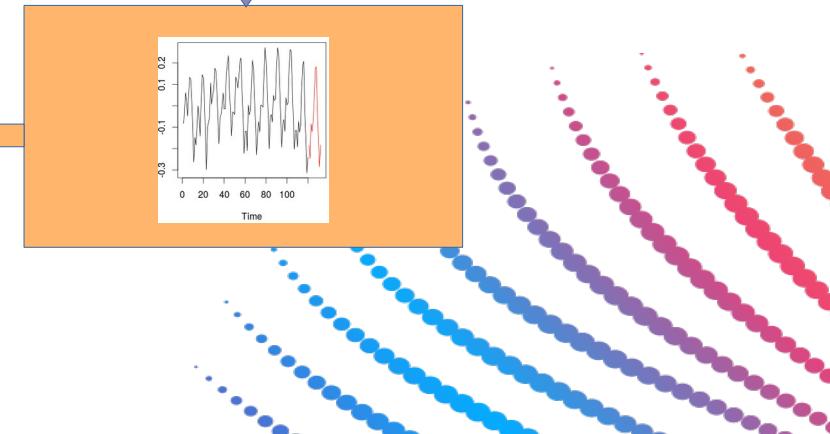
Model selection



Remove seasonality



Remove Trend



Análisis de series temporales

■ Modelos de tendencia

- Se requiere modelar la tendencia de la serie (**de existir**)
- Las técnicas de regresión sobre series temporales son un mecanismo simple y útil en muchos casos:
 - Linear regressors
 - Polynomial regressors
 - ...
 - Etc.

ARIMA también propone diferenciar la serie un número de veces, hasta que la media es constante.



Análisis de series temporales

Cómo modelar la tendencia :

- Air Passenger dataset (**notebook *TrendAnalysis***) :

Linear and polynomial regressors

We assume that a time series $x(t) = T(t) + S(t) + E(t)$. The trend model is $T(t) = f(t, w) + \epsilon(t)$, where f is the regressor.

```
# Linear model hypothesis to estimate trend T(t)
def LinearModel(t, w0, w1):
    return w1*t+w0

# Quadratic polynomial model hypothesis to estimate trend T(t)
def QuadraticModel(t, w0, w1, w2):
    return w2*(t**2) + w1*t + w0
```

Example for AirPassengers

Get data

We get the time index into variable **t** and time series data into variable **x_ts**.

```
df = pd.read_csv('./data/AirPassengers.csv')
df['Month'] = pd.to_datetime(df['Month']) # Transform "month" column to date datatype

x_ts= df['#Passengers'].to_numpy()
t= np.array(range(len(x_ts)))

plt.plot(t, x_ts)
```



Análisis de series temporales

Cómo modelar la tendencia :

- Air Passenger dataset (notebook *TrendAnalysis*) :

Test linear and quadratic models

```
# Try a linear model
w_linear, _ = curve_fit(LinearModel, t, x_ts) # Returns w: parameters and cov: Covariance matrix

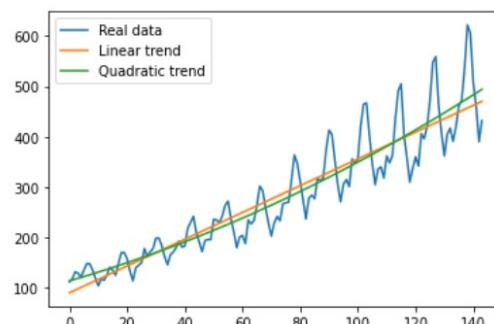
# Try a quadratic model
w_quad, _ = curve_fit(QuadraticModel, t, x_ts) # Returns w: parameters and cov: Covariance matrix

# Get linear trend estimation
T_linear= LinearModel(t, *w_linear)

# Get quadratic trend estimation
T_quad= QuadraticModel(t, *w_quad)

plt.plot(t, x_ts) # plot real data
plt.plot(t, T_linear) # plot linear model estimation
plt.plot(t, T_quad) # plot quadratic model estimation
plt.legend(['Real data', 'Linear trend', 'Quadratic trend'])
```

<matplotlib.legend.Legend at 0x7f1dab8fe8b0>



Análisis de series temporales

■ Cómo modelar la tendencia :

- ¡ Cuidado ! La hipótesis o requisito necesario de muchos modelos de predicción es la estacionariedad.

Si la serie no es estacionaria, podemos forzar la estacionariedad a través de :

- Diferenciación de la serie temporal
- Preprocesamiento (transformaciones)



Análisis de series temporales

■ Cómo modelar la tendencia :

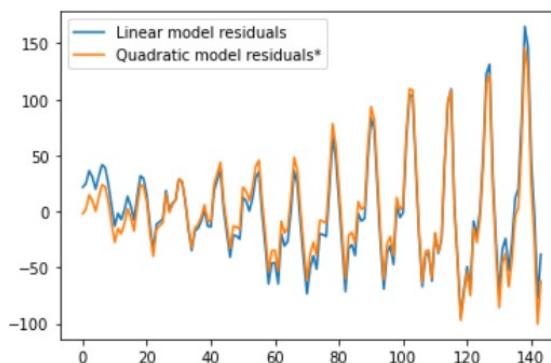
- Air Passenger dataset ([notebook *TrendAnalysis*](#)) :

En nuestro caso, un análisis exploratorio de datos temprano nos puede alertar de falta de estacionariedad.

Test residuals

```
T_linear_res= x_ts - T_linear # Get residuals of linear trend model  
T_quad_res= x_ts - T_quad # Get residuals of quadratic trend model  
  
# Plot residuals  
plt.plot(t, T_linear_res)  
plt.plot(t, T_quad_res)  
plt.legend(['Linear model residuals', 'Quadratic model residuals*'])  
print('UOPSSS!!: Variance increases over time!! -> Non-stationary')
```

UOPSSS!!: Variance increases over time!! -> Non-stationary



Análisis de series temporales

Cómo modelar la tendencia :

- Air Passenger dataset (notebook *TrendAnalysis*) :

Variance reduction

In this case, we decide to log-transform the initial time series. Then, re-do all trend estimation.

```
# Make a log-transformation to reduce the effect of variance increase
x_ts_log= np.log(x_ts)

# Try a linear model
w_linear, _ = curve_fit(LinearModel, t, x_ts_log) # Returns w: parameters and cov: Covariance matrix

# Try a quadratic model
w_quad, _ = curve_fit(QuadraticModel, t, x_ts_log) # Returns w: parameters and cov: Covariance matrix

# Get linear trend estimation
T_linear= LinearModel(t, *w_linear)

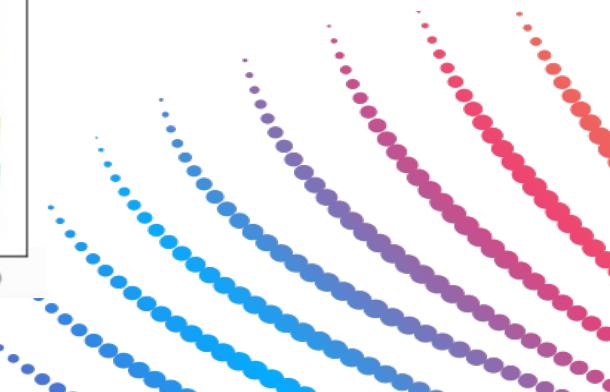
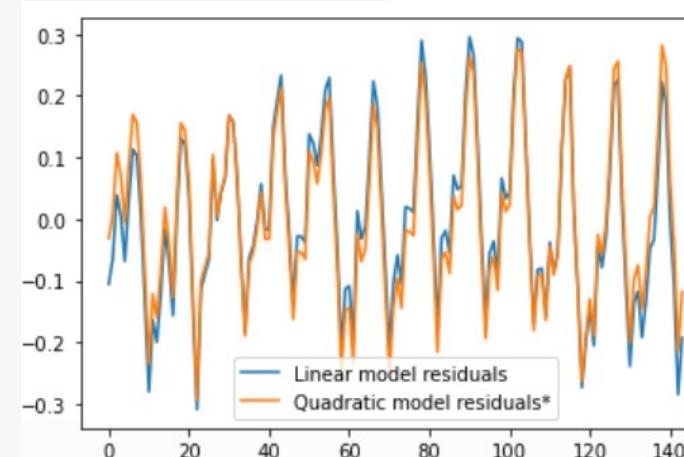
# Get quadratic trend estimation
T_quad= QuadraticModel(t, *w_quad)

plt.plot(t, x_ts_log) # plot real data
plt.plot(t, T_linear) # plot linear model estimation
plt.plot(t, T_quad) # plot quadratic model estimation
plt.legend(['Real data', 'Linear trend', 'Quadratic trend'])

T_linear_res= x_ts_log - T_linear # Get residuals of linear trend model
T_quad_res= x_ts_log - T_quad # Get residuals of quadratic trend model

# Calculate MSE for both hypotheses
MSE_linear= np.sum(T_linear_res*T_linear_res)/len(T_linear_res)
MSE_quad= np.sum(T_quad_res*T_quad_res)/len(T_quad_res)

print('MSE of residuals with linear model: {:.3f}'.format(MSE_linear))
print('MSE of residuals with quadratic model: {:.3f}'.format(MSE_quad))
```



Análisis de series temporales

■ Cómo modelar la tendencia :

- **¿ Cuál es el mejor modelo de tendencia ?**

Una propiedad deseable es que los residuos (errores) del modelo sigan una distribución normal de media 0.

Los tests estadísticos de normalidad pueden ayudar a decidir si el modelo de tendencia está sesgado o no.

Ejemplos de tests de normalidad :

- Shapiro-Wilk
- Jarque-Bera



Análisis de series temporales

■ Cómo modelar la tendencia :

- Air Passenger dataset (notebook *TrendAnalysis*) :

```
# Plot residuals
plt.figure()
plt.plot(t, T_linear_res)
plt.plot(t, T_quad_res)
plt.legend(['Linear model residuals', 'Quadratic model residuals'])

pval_linear= shapiro(T_linear_res).pvalue
pval_quad= shapiro(T_quad_res).pvalue
print('pvalue of linear model (Shapiro-Wilk normality test): {}'.format(pval_linear))
print('pvalue of quadratic model (Shapiro-Wilk normality test): {}'.format(pval_quad))
print('\nRESIDUALS IN QUADRATIC MODEL ARE NOT NORMAL --> DISCARD THIS HYPOTHESIS')

print('\nLinear trend model is selected')

MSE of residuals with linear model: 0.019
MSE of residuals with quadratic model: 0.018
pvalue of linear model (Shapiro-Wilk normality test): 0.09458756446838379
pvalue of quadratic model (Shapiro-Wilk normality test): 0.04592949151992798
```



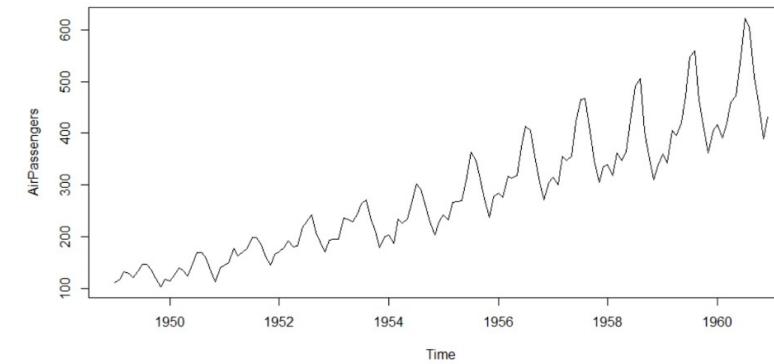
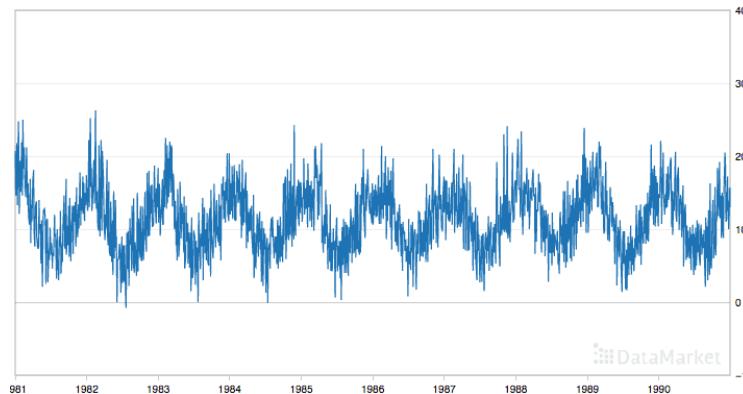
Análisis de series temporales

■ Cómo modelar la estacionalidad:

La estacionalidad ocurre cuando los datos muestran un comportamiento periódico.

Herramientas para detectar estacionalidad :

- Análisis de datos exploratorio(ACF, Time Series plot)
- Transformaciones de espacio de datos (Fourier, Laplace...)



Análisis de series temporales

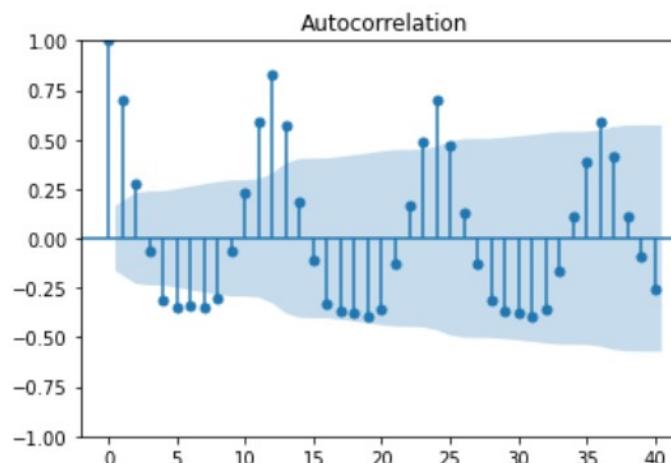
■ Cómo modelar la estacionalidad con la gráfica de la función ACF

La gráfica de la ACF puede ayudar a determinar el periodo (S) de una estacionalidad. En este caso, la forma del ACF es sinusoidal (**notebook Seasonality**).

Plot ACF to find seasonality

In this case, we decide to use exploratory data analysis using the ACF to check if there is seasonality

```
from statsmodels.graphics.tsaplots import plot_acf # AutoCorrelation Function  
plot_acf(x_ts, lags=40)
```



Análisis de series temporales

■ Cómo modelar la estacionalidad con la gráfica de la función ACF

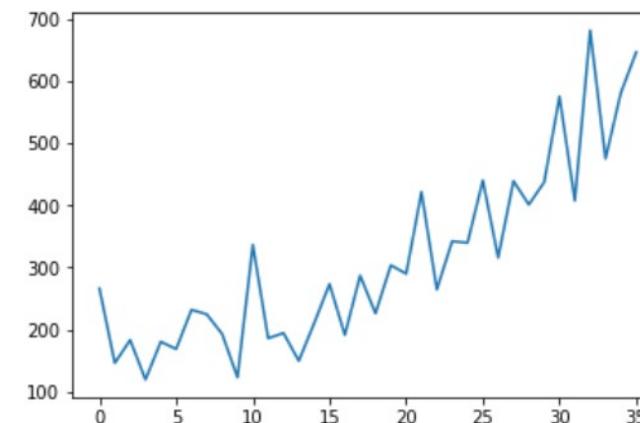
Dataset : Ventas mensuales de una marca de champú a lo largo de 3 años.

Ejemplo de serie de datos sin estacionalidad

Load dataset

```
df = pd.read_csv('./data/shampoo.csv')
x_ts= df['Sales'].to_numpy()

t= np.array(range(len(x_ts)))
plt.plot(x_ts)
```



Análisis de series temporales

■ Cómo modelar la estacionalidad con la gráfica de la función ACF

Remove trend (assume quadratic model)

```
# Quadratic polynomial model hypothesis to estimate trend T(t)
def QuadraticModel(t, w0, w1, w2):
    return w2*(t**2) + w1*t + w0

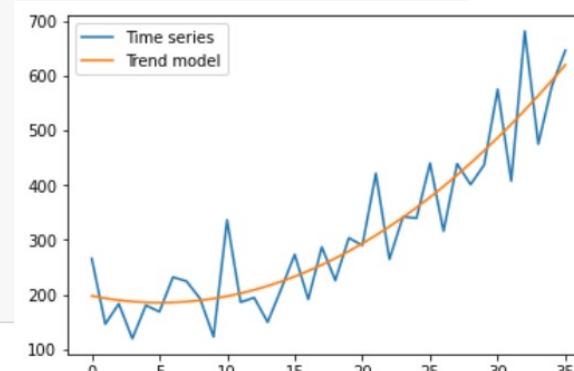
# Fit a linear trend model
w_quad, _ = curve_fit(QuadraticModel, t, x_ts) # Returns w: parameters and cov: Covariance matrix

# Get linear trend estimation
T_quad= QuadraticModel(t, *w_quad)

# Plot trend model
plt.plot(t, x_ts)
plt.plot(t, T_quad)
plt.legend(['Time series', 'Trend model'])

# Remove trend
x_ts= x_ts - T_quad # Get time series without trend

plt.figure()
plt.plot(t, x_ts)
```



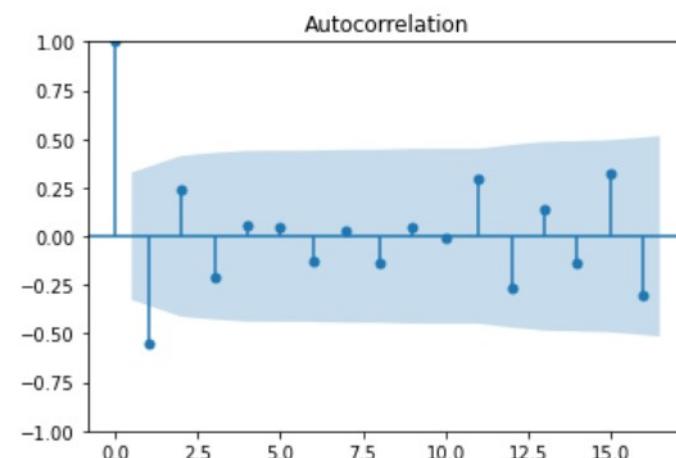
Análisis de series temporales

■ Cómo modelar la estacionalidad con la gráfica de la función ACF

La gráfica de ACF sugiere que la serie temporal no tiene componente estacional (o lo que es lo mismo, $S(t)= 0$).

Plot ACF to find seasonality

```
plot_acf(x_ts)
```



Análisis de series temporales

■ Cómo modelar la estacionalidad:

Necesitamos un modelo de estacionalidad, y suponemos que ya tenemos el periodo $S>0$ (por ejemplo, a través del ACF).

- Un modelo de estacionalidad es una descripción de cómo se van repitiendo los valores a lo largo de la serie temporal.
- Ejemplo:
 - 1) Calcular la serie temporal promedio cada S valores.
 - 2) Restar la serie resultante de la original.



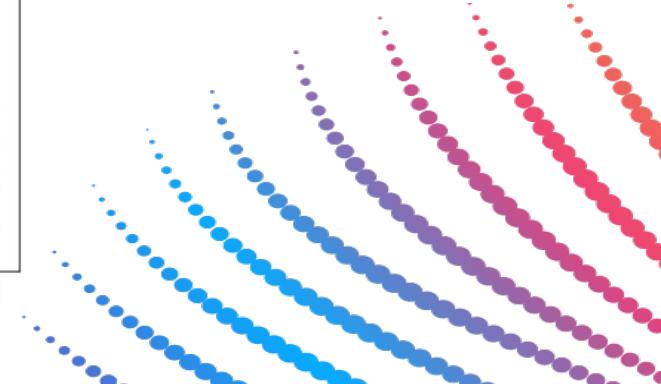
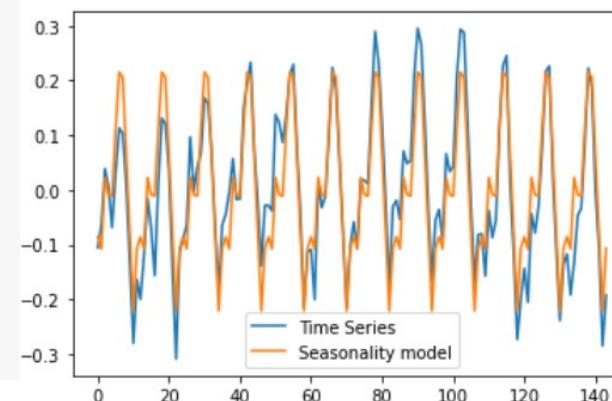
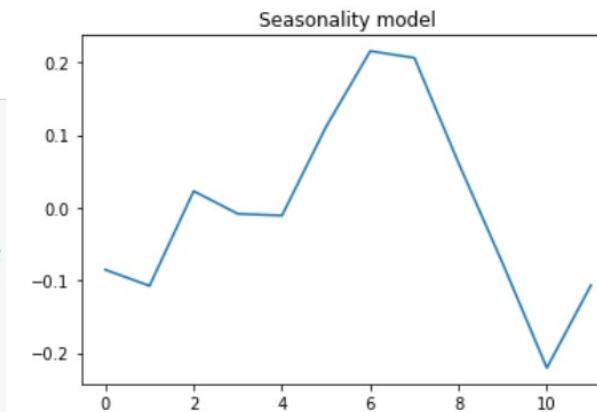
Análisis de series temporales

Cómo modelar la estacionalidad:

AirPassengers data series (notebook *Seasonality*)

Remove seasonality

```
S= 12 # Set seasonality period  
  
# Find seasonality model by averaging time series values with period S  
Season= np.zeros(S)  
for i in range(S):  
    Season[i]= np.mean(x_ts[i::S]) # Take all time series values from i, in steps of S  
  
# Plot seasonal model  
plt.plot(Season)  
plt.title('Seasonality model')  
  
# plot Seasonal model over the time series  
NumSeasons= int(np.ceil(len(x_ts)/len(Season)))  
TiledSeason= np.tile(Season, NumSeasons) # Tile seasonal model on time series  
TiledSeason= TiledSeason[:len(x_ts)]  
plt.figure()  
plt.plot(t, x_ts)  
plt.plot(t, TiledSeason)  
plt.legend(['Time Series', 'Seasonality model'])  
  
# Remove seasonality  
x_ts= x_ts - TiledSeason  
  
plt.figure()  
plt.plot(t, x_ts)  
plt.title('Time Series with no seasonality')
```



Análisis de series temporales

■ Condición de estacionariedad

La condición de estacionariedad es necesaria por muchos modelos de predicción de series temporales. Es necesario comprobar dicha estacionariedad.

¿ Cómo ?

- A través del análisis exploratorio de datos (función ACF)
- A través de tests estadísticos (Augmented Dickey-Fuller)

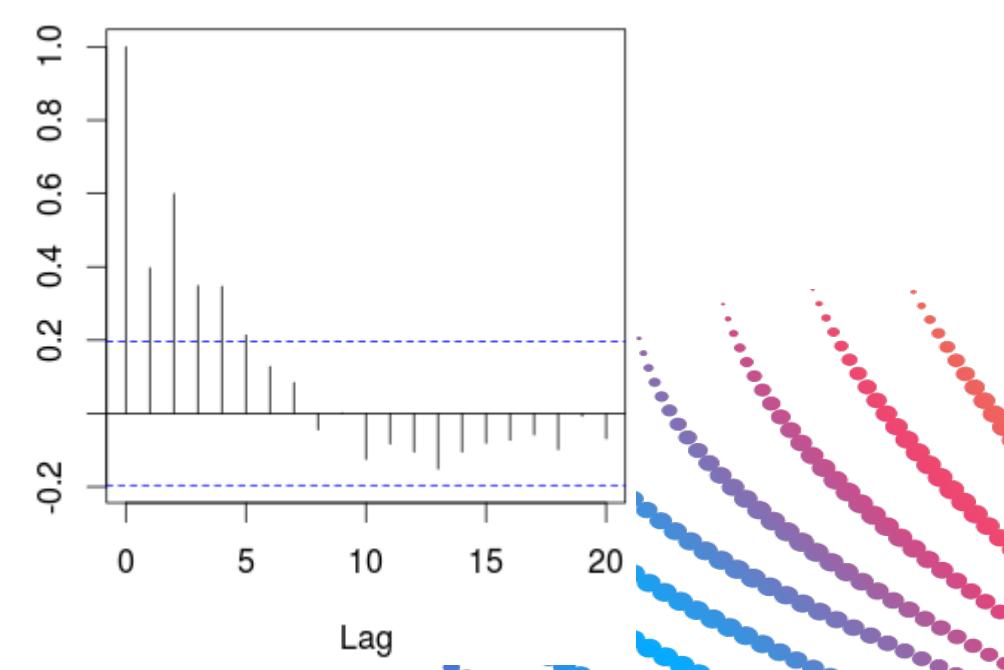
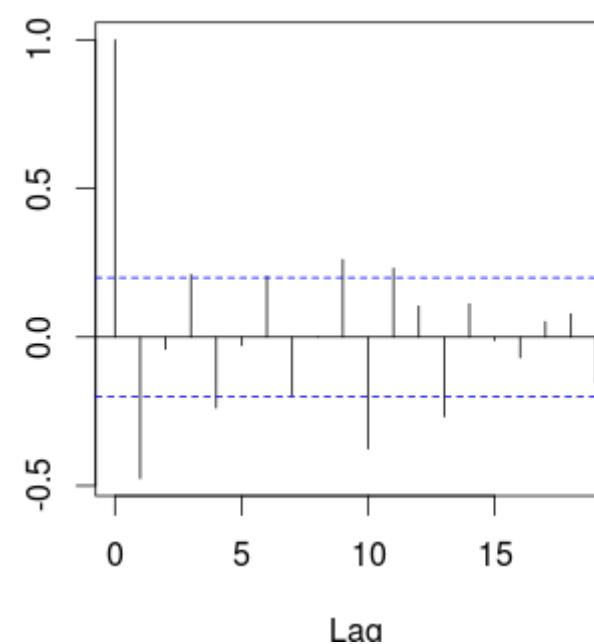
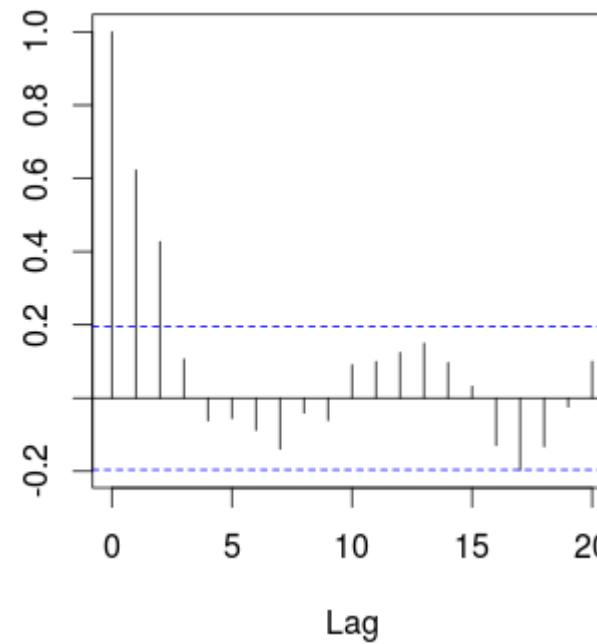


Análisis de series temporales

■ Análisis exploratorio de datos para comprobar estacionariedad

Se utiliza la gráfica del ACF. Si esta decrece « *rápidamente* » a 0, es señal de estacionariedad. En otro caso, es señal de no estacionariedad.

Ejemplos de gráficas de ACF de series estacionarias :

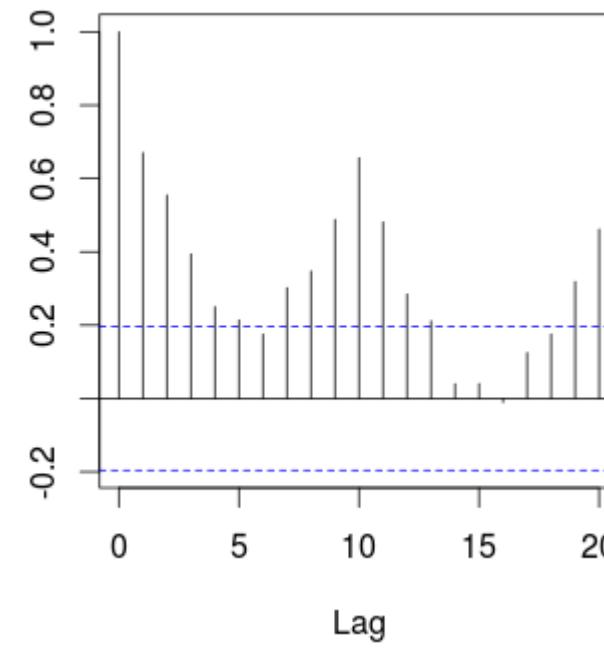
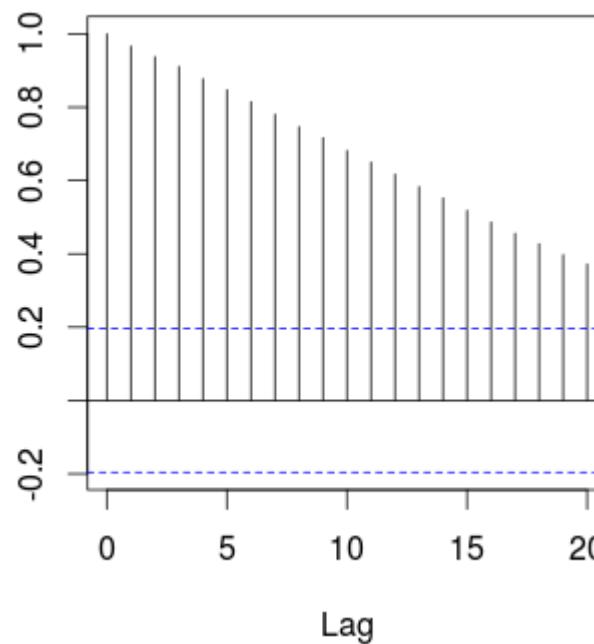


Análisis de series temporales

■ Análisis exploratorio de datos para comprobar estacionariedad

Se utiliza la gráfica del ACF. Si esta decrece « *rápidamente* » a 0, es señal de estacionariedad. En otro caso, es señal de no estacionariedad.

Ejemplos de gráficas de ACF de series **NO estacionarias :**



Análisis de series temporales

Ejemplo de comprobación de estacionariedad

AirPassengers data series (notebook [Stationarity](#))

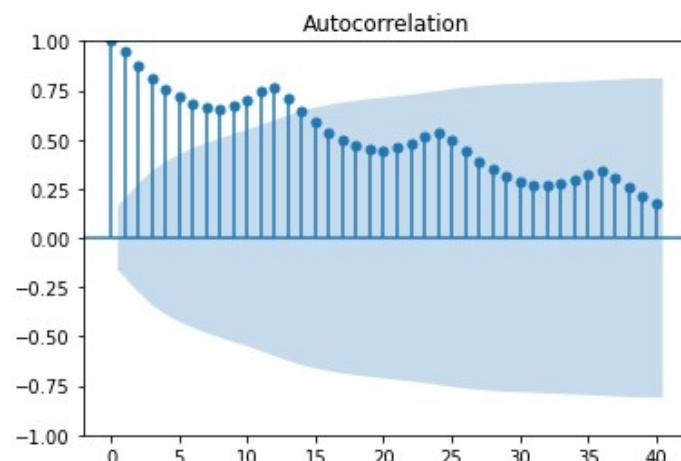
Check Stationarity (I)

```
plot_acf(x_ts, lags=40)
print('Not stationary based on ACF')

results= adfuller(x_ts)
pvalue= results[1]
print('With p-value={:.3f}, the Augmented Dickey-Fuller test DISCARDS stationarity'.format(pvalue))
```

Not stationary based on ACF

With p-value=0.992, the Augmented Dickey-Fuller test DISCARDS stationarity



Análisis de series temporales

Ejemplo de comprobación de estacionariedad (continuación)

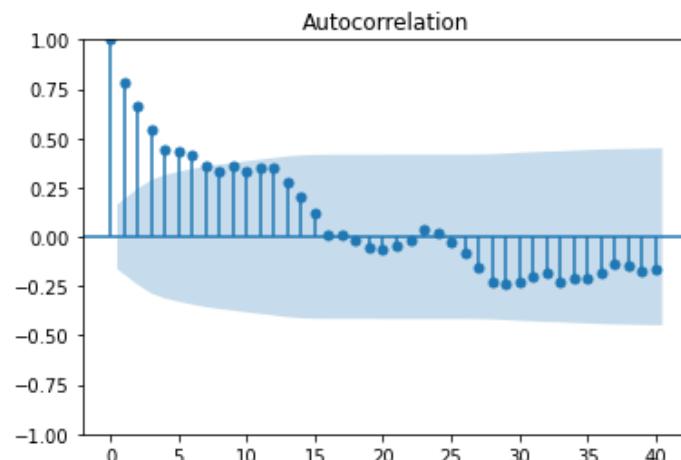
AirPassengers data series (notebook [Stationarity](#))

Check Stationarity (III)

```
plot_acf(x_ts, lags=40)
print('Stationary (maybe?) time series based on ACF')

results= adfuller(x_ts)
pvalue= results[1]
print('With p-value={:.3f}, the Augmented Dickey-Fuller test FAILS stationarity'.format(pvalue))
```

Stationary time series based on ACF
With p-value=0.441, the Augmented Dickey-Fuller test FAILS stationarity



Análisis de series temporales

Ejemplo de comprobación de estacionariedad (continuación)

AirPassengers data series (notebook [Stationarity](#))

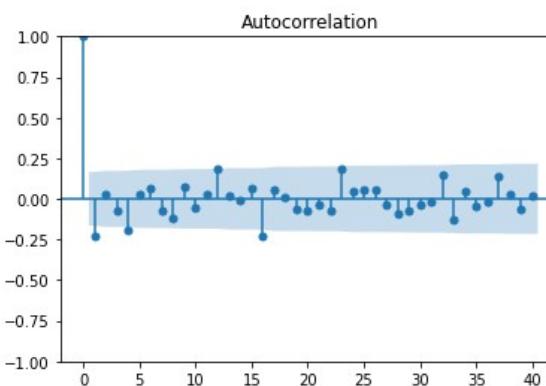
Differentiate time series

```
# Differentiate Time Series  
x_ts_d = x_ts[1:]-x_ts[:-1]
```

Check Stationarity (IV)

```
plot_acf(x_ts_d, lags=40)  
print('Stationary time series based on ACF')  
  
results= adfuller(x_ts_d)  
pvalue= results[1]  
print('With p-value={:.3f}, the Augmented Dickey-Fuller test ACCEPTS stationarity'.format(pvalue))
```

Stationary time series based on ACF
With p-value=0.000, the Augmented Dickey-Fuller test ACCEPTS stationarity



Modelo ARIMA



Modelo ARIMA

■ ARIMA : Auto-Regressive Integrated Moving Average

EL modelo **ARIMA(p,d,q)** asume regresión lineal sobre :

- **Valores históricos (Modelo autorregresivo de orden p, AR(p)) :**

$$X(t) = \mu + \sum_{i=1}^p \lambda_i X(t - i) + \xi(t)$$

- **Errores anteriores de estimación (Moving Average de orden q, MA(q)) :**

$$X(t) = \xi(t) - \sum_{i=1}^q \theta_i \xi(t - i)$$

- **Número de veces que hay que diferenciar la serie para hacerla estacionaria**

Orden de integración d



Modelo ARIMA

■ ARIMA(p,d,q) : Cómo encontrar el parámetro d

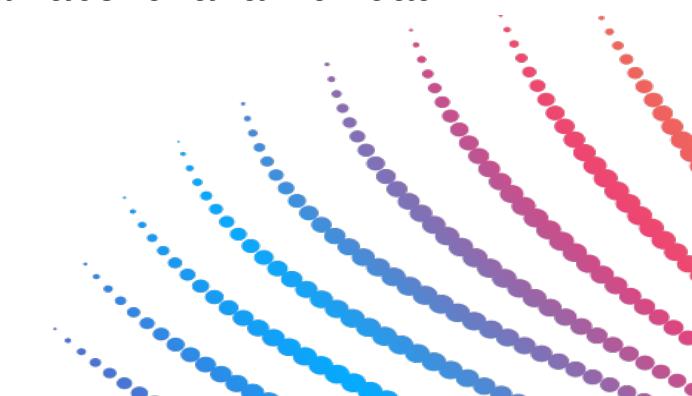
- **Regla 1:** Si la serie tiene autocorrelaciones positivas por un gran número de lags (digamos, 10 por ejemplo), entonces probablemente no sea estacionaria y requiera diferenciación.
- **Regla 2:** Si el lag 1 de autocorrelación es 0 o negativo, o las autocorrelaciones son pequeñas y sin patrón, entonces no se necesita diferenciar. De hecho, si el lag 1 de autocorrelación es -0.5 o aún menor, es posible que la serie esté sobre-diferenciada.
- **Regla 3:** El orden óptimo de diferenciación es a menudo el orden al que la desviación estandar es mínima.
- **Regla 4:** Un modelo sin orden de diferenciación ($d=0$) asume que la serie original es estacionaria.
- **Regla 5:** Un modelo sin orden de diferenciación normalmente incluye un término constante (media de la serie distinta de 0). Con dos órdenes de diferenciación, normalmente no hay un término constante.



Modelo ARIMA

■ ARIMA(p,d,q) : Cómo encontrar los parámetros p,q

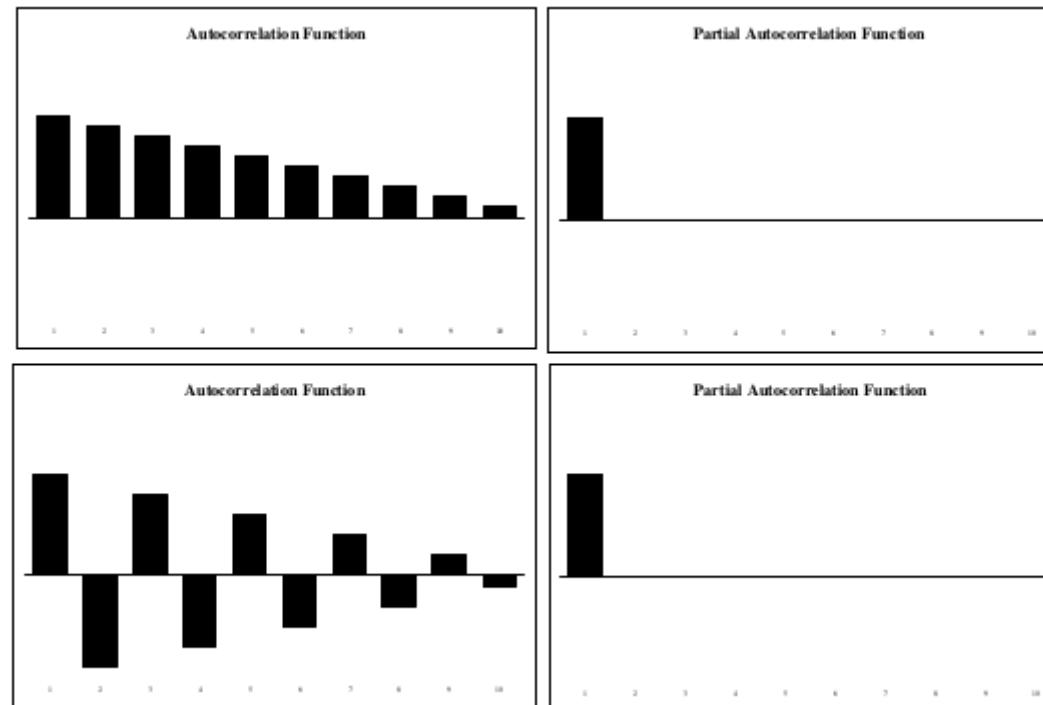
- **Regla 1:** Si la PACF de la serie ya diferenciada decrece bruscamente y/o el lag 1 de autocorrelación es positivo, puede ser significativo de requerir más términos AR(p). Normalmente, el último lag con PACF por encima del umbral es el valor de p.
- **Regla 2:** Si la ACF de la serie ya diferenciada decrece rápidamente, y/o el lag 1 de autocorrelación es negativo, podría ser significativo de un modelo MA(q). El último lag con ACF por encima del umbral suele indicar el valor de q.
- **Regla 3:** Es posible que un modelo AR y un modelo MA se cancelen entre sí. Por lo que si un modelo mixto ARMA(p,q) parece modelar los datos, se debería probar un modelo AR con valor inferior de p y un modelo MA con un valor inferior de q.
- **Regla 4:** Si hay una raíz unidad en la parte del AR (la suma de los coeficientes de AR es casi 1), se debe reducir el número de órdenes en p e incrementar el valor d.
- **Regla 5:** Si hay una raíz unidad en la parte MA (la suma de los coeficientes de MA es casi igual a 1), se debe reducir el orden q y el valor de diferenciación d.
- **Regla 6:** Si las predicciones a largo plazo parecen erráticas o inestables, podría haber una raíz unidad en la parte AR o en la parte MA.



Modelo ARIMA

■ ARIMA(p,d,q) : Ejemplos

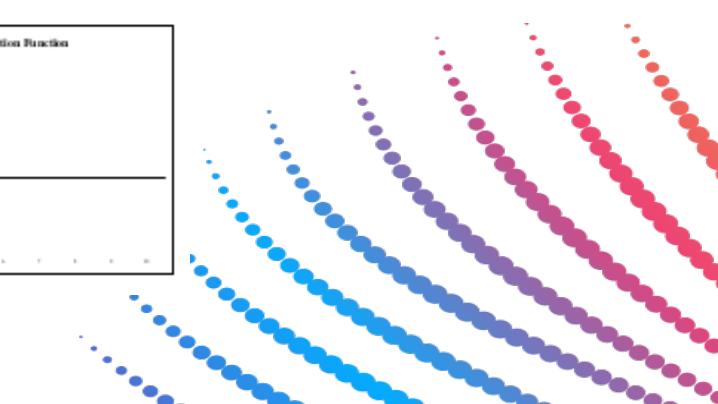
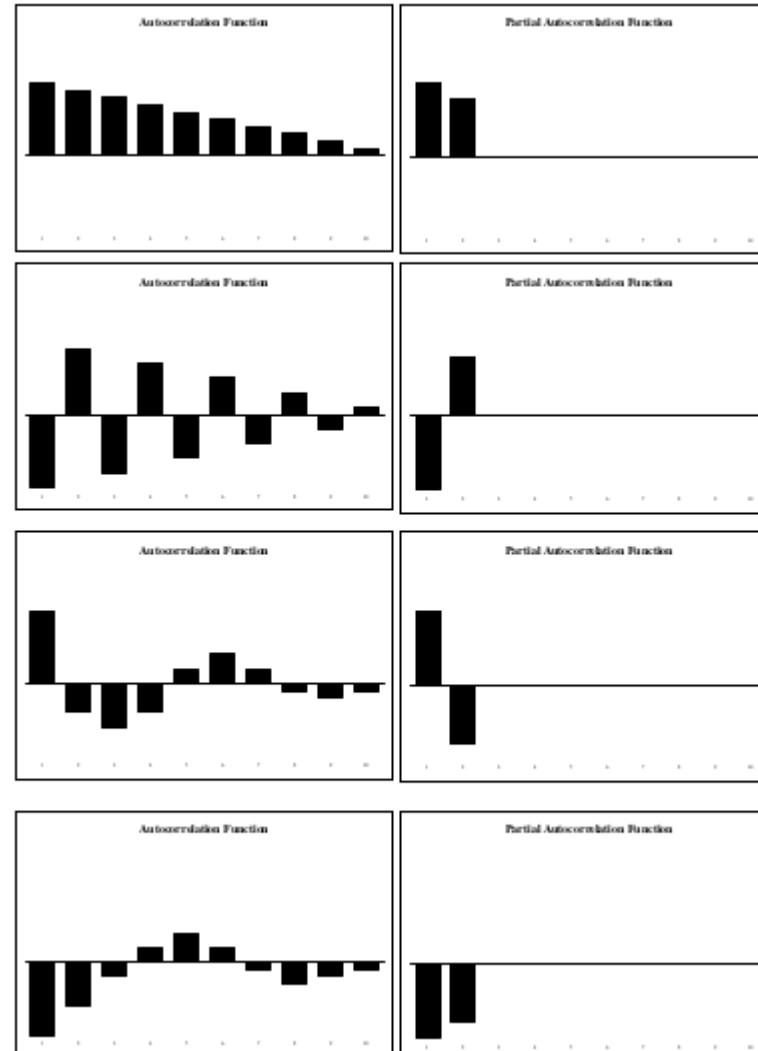
- AR(1) ACF/PACF :



Modelo ARIMA

■ ARIMA(p,d,q) : Ejemplos

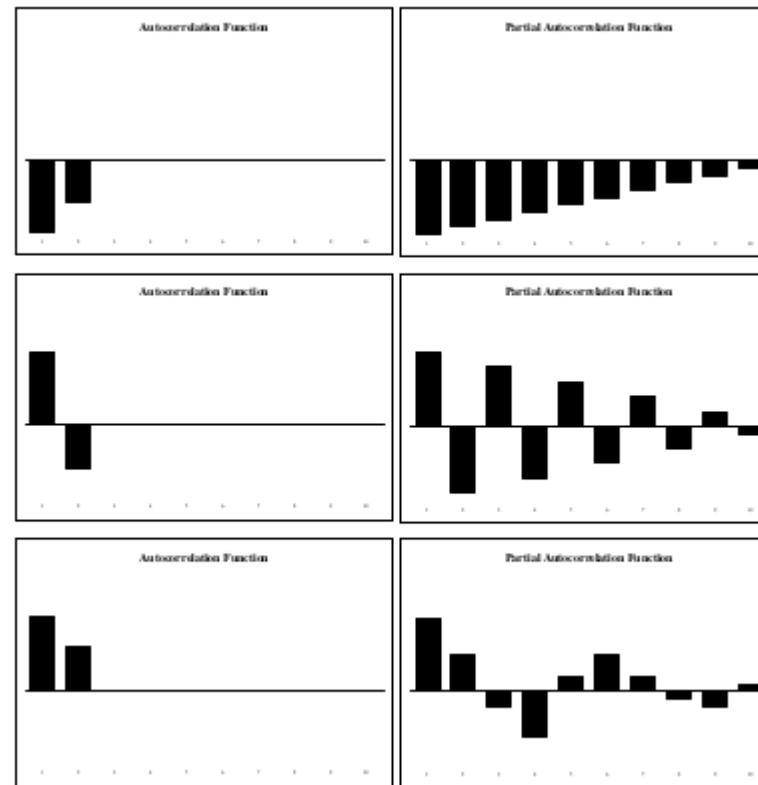
- AR(2) ACF/PACF :



Modelo ARIMA

■ ARIMA(p,d,q) : Ejemplos

- MA(2) ACF/PACF :



Modelo ARIMA

■ Selección del mejor modelo ARIMA(p,d,q)

Normalmente, disponemos de varias hipótesis de modelos de predicción. ¿Cuál es el correcto ?

- ¿El que mejor ajusta los datos de entrenamiento ? → Overfitting !!
- ¿El que mejor ajusta los datos de test ? → ... ¿quizá?
- Criteria to consider a balance between model complexity and performance

El criterio de Akaike (**AIC**) es uno de los criterios más usados para seleccionar un modelo ARIMA. Considera tanto la precisión como la complejidad del modelo :

$$AIC = 2k + n \operatorname{Log}(RSS/n)$$

k= Número de parámetros

n= Número de datos

RSS= Residual Sum of Squares



Modelo ARIMA

■ ARIMA(p,d,q) en Python

Comenzamos desde una serie sintética generada ([notebook ArimaExample](#)) :

Synthetic data generation for ARMA(2,2) model

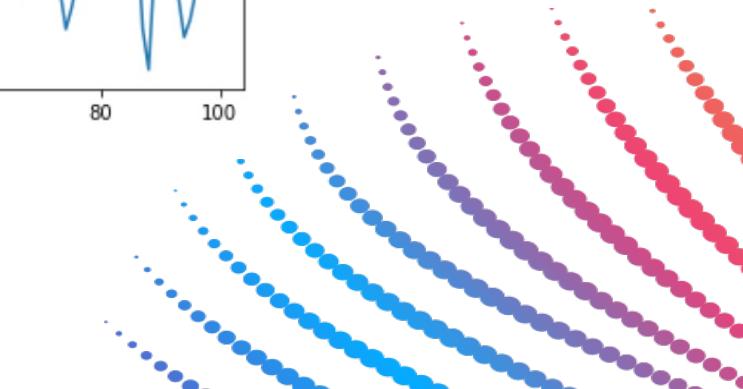
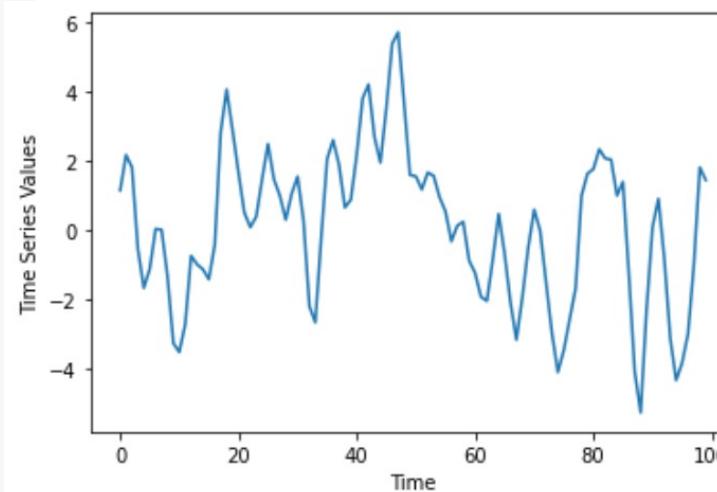
```
N= 100 # Number of data in the time series

# ARMA(p,q) parameters to generate synthetic data
arparams = np.array([.75, -.25])
maparams = np.array([.65, .35])

ar = np.r_[1, -arparams] # add zero-lag and negate
ma = np.r_[1, maparams] # add zero-lag
x = arma_generate_sample(ar, ma, nsample=N)
t= np.arange(len(x))

df= pd.DataFrame()
df[ 'Time']= t
df[ 'Values']= x

# Plot generated time series
plt.plot(x)
plt.xlabel('Time')
plt.ylabel('Time Series Values')
```



Modelo ARIMA

■ ARIMA(p,d,q) en Python

Comprobamos estacionariedad con el test Augmented Dickey-Fuller y la ACF
(notebook *ArimaExample*) :

Augmented Dickey-Fulle test

=====

Test Statistic

p-value

#Lags Used

Number of Observations Used

Critical Value (1%)

Critical Value (5%)

Critical Value (10%)

-2.323637
0.164458

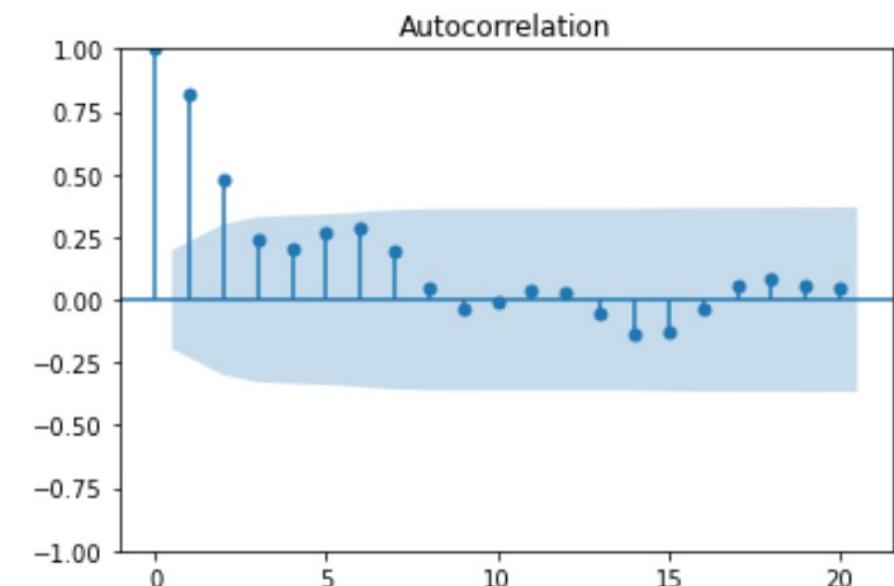
3.000000

96.000000

-3.500379

-2.892152

-2.583100



Modelo ARIMA

■ ARIMA(p,d,q) en Python

Entrenamiento ([notebook ArimaExample](#)) :

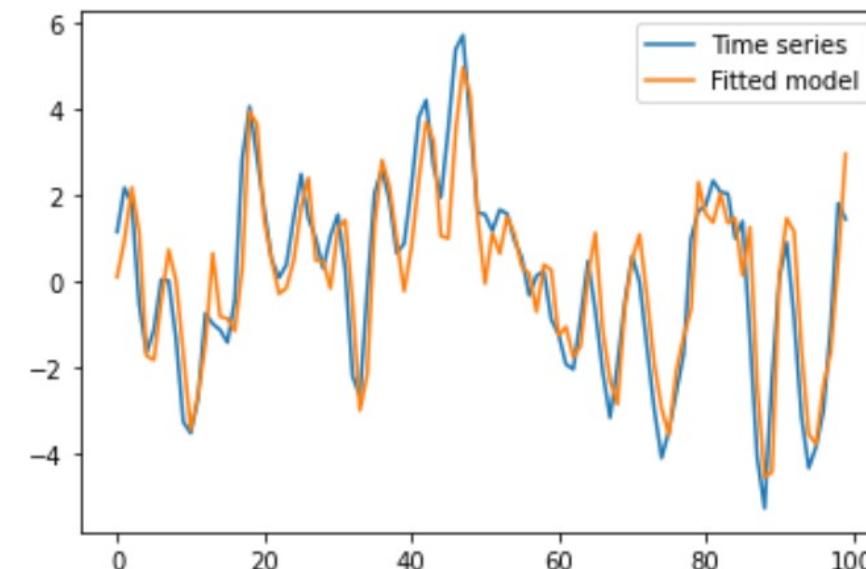
ARIMA(p,d,q) model fitting

We hypothesize ARMA(2,2) (we are cheating because we know the actual time series).

```
# Create model
model = ARIMA(df['Values'], order=(2,0,0))

# Fit model
fitted_model = model.fit()

# Show fitted values
plt.figure()
plt.plot(df['Values'])
plt.plot(fitted_model.fittedvalues)
plt.legend(['Time series', 'Fitted model'])
```



Modelo ARIMA

■ ARIMA(p,d,q) en Python

Diagnóstico ([notebook ArimaExample](#)) :

ARIMA model diagnosis with AIC

```
# Calculate AIC
print('The Akaike model value is: {}'.format(fitted_model.aic))

# Model summary
fitted_model.summary()
```

The Akaike model value is: 296.4841470459956

SARIMAX Results

Dep. Variable:	Values	No. Observations:	100			
Model:	ARIMA(2, 0, 0)	Log Likelihood	-144.242			
Date:	Thu, 12 May 2022	AIC	296.484			
Time:	10:24:02	BIC	306.905			
Sample:	0	HQIC	300.702			
	- 100					
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
const	0.0914	0.357	0.256	0.798	-0.609	0.792
ar.L1	1.2988	0.083	15.697	0.000	1.137	1.461
ar.L2	-0.5839	0.085	-6.846	0.000	-0.751	-0.417
sigma2	1.0278	0.154	6.685	0.000	0.727	1.329
Ljung-Box (L1) (Q):	6.83	Jarque-Bera (JB):	0.31			
Prob(Q):	0.01	Prob(JB):	0.86			
Heteroskedasticity (H):	1.34	Skew:	0.11			
Prob(H) (two-sided):	0.40	Kurtosis:	2.83			

Modelo ARIMA

■ ARIMA(p,d,q) en Python

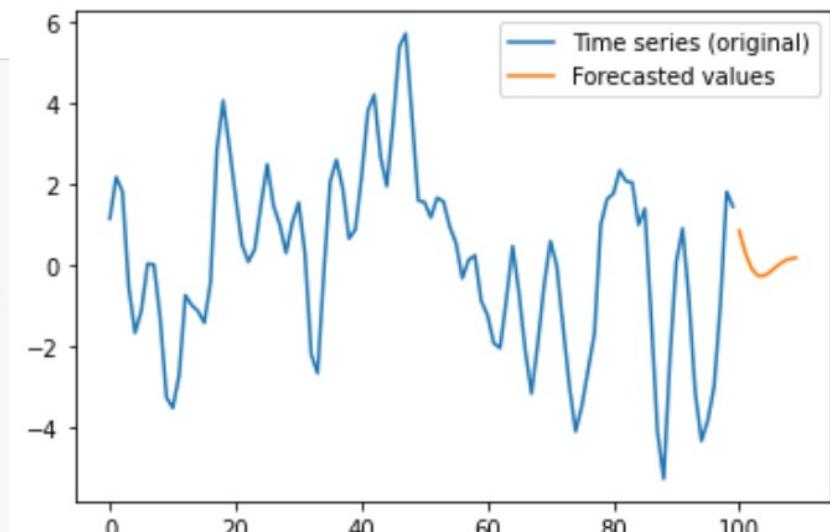
Predicción ([notebook ArimaExample](#)) :

ARIMA forecasting

```
# Forecasting horizon (10 next values)
fh= 10

# Predict next fh values with ARIMA
predictions= fitted_model.get_prediction(start=N, end=N+fh-1).predicted_mean
t_future= np.arange(N, N+fh)

# print predictions
plt.plot(df['Time'], df['Values'])
plt.plot(t_future, predictions)
plt.legend(['Time series (original)', 'Forecasted values'])
```



Modelo ARIMA

ARIMA es un modelo clásico, pero muy extendido (no siempre el mejor).

Algunas extensiones :

- **SARIMA** : Incluye tratamiento de estacionalidad
- **SARIMAX** : Incluye también tratamiento de variables exógenas.
- **Auto-ARIMA** : Ampliamente implementado con posibilidad de ARIMA/SARIMA/SARIMAX. Se realiza selección automática de los parámetros ARIMA(p,d,q) con técnicas heurísticas (greedy, grid search, extensive search...). Muchas bibliotecas de series temporales incluyen una implementación de Auto-ARIMA.



Modelo ARIMA

■ Ejercicio (*Solución en ARIMAAirPassengers*)

- Aplicar la metodología ARIMA en la serie de datos *AirPassengers.csv*.

Realizar:

- 1) Análisis exploratorio de datos.
- 2) Comprobación de estacionariedad (cuando proceda) y preprocessamiento.
- 3) Modelado y eliminación de tendencia (si procede).
- 4) Modelado y eliminación de estacionalidad (si procede).
- 5) Experimentación para encontrar los parámetros (p,d,q) óptimos.
- 6) Validación y selección del modelo.
- 7) Predicción.
- 8) Incorporar a la predicción la componente estacional.
- 9) Incorporar a la predicción la componente de tendencia.
- 10) Deshacer preprocessamiento.
- 11) Devolver resultado final.



Modelo ARIMA

ARIMA is a very extended model due to its simplicity and tradition in some science fields, but it has some limitations :

- In practice, it is not a well-suited model for mid/long-term forecasting (only 1/2 values are predicted with -sometimes- acceptable accuracy).
- **Not suitable when there are non-linear dependencies among data.**
- Very dependent on trend and seasonal models.
- Unit roots avoidance (ADF test).

