

```
import os, sys
from google.colab import drive
drive.mount('/content/mnt', force_remount=True)
nb_path = '/content/notebooks'
os.symlink('/content/mnt/My·Drive/Colab Notebooks/Librerias', nb_path)
#sys.path.insert(0, nb_path) # or append(nb_path)
sys.path.append(nb_path) # or append(nb_path)
```

Mounted at /content/mnt

```
import warnings
warnings.filterwarnings('ignore')
```

▼ Predicción de Series Temporales

Bibliografía

- Hyndman, R. J., A. B. Koehler, J. K. Ord and R. D. Snyder (2008). Forecasting with exponential smoothing: the state space approach. Berlin: Springer-Verlag.

- Hyndman, Rob y George Athanasopoulos (2015). "Forecasting: principles and practice". Otexts. <https://www.otexts.org/fpp>
- Gardner Jr, E. S. (1985). Exponential smoothing: The state of the art. Journal of Forecasting 4(1), 1–28.
- Gardner Jr, E. S. (2006). Exponential smoothing: The state of the art—Part II. International Journal of Forecasting 22(4), 637–666.

Una **serie temporal** es una sucesión de observaciones de una variable tomadas en varios instantes de tiempo:

- Interesa estudiar los cambios en esa variable con respeto al tiempo.
- Predecir sus valores futuros.

Problema:

Estas observaciones provienen de una distribución que puede ser diferente en cada instante del tiempo.

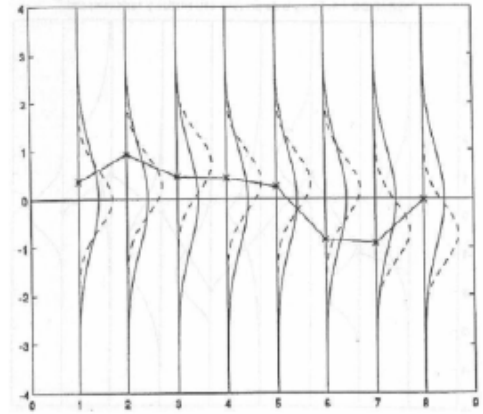
Estacionariedad:

- Una serie **es estacionaria** si la media y la variabilidad se mantienen constantes a lo largo del tiempo.

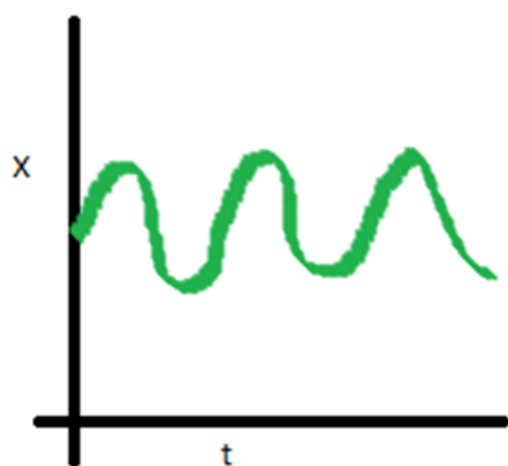
- Una serie **es no estacionaria** si la media y/o la variabilidad cambian a lo largo del tiempo.:
 - Series no estacionarias pueden mostrar cambios de varianza.
 - Series no estacionarias pueden mostrar una tendencia, es decir que la media crece o baja a lo largo del tiempo.
 - Además, pueden presentar efectos estacionales, es decir que el comportamiento de la serie es parecido en ciertos tiempos periódicos en el tiempo.

Definición 3. *Una serie temporal es **estacionaria** en sentido amplio si:*

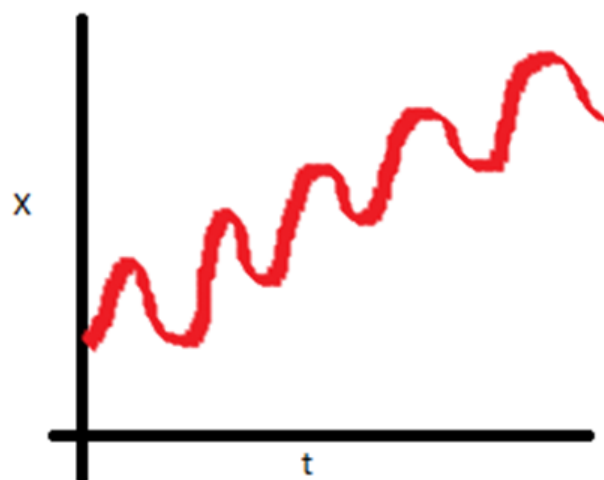
- $E[X_t] = \mu$ para todo t
- $Var(X_t) = \sigma^2$ para todo t .
- $Cov(X_t, X_{t+k}) = \gamma_k$ para todo t y k .



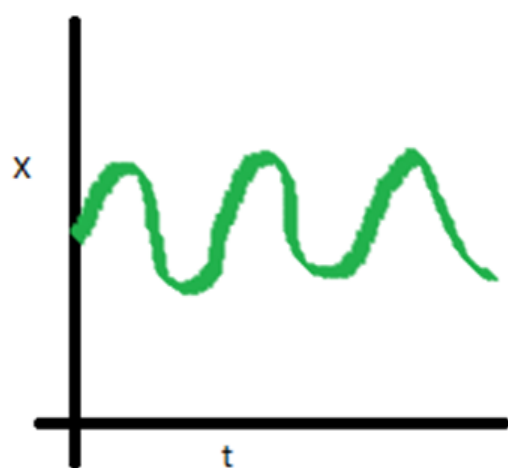
El ejemplo más simple es el RUIDO BLANCO, cuando la media y la covarianza son siempre cero.



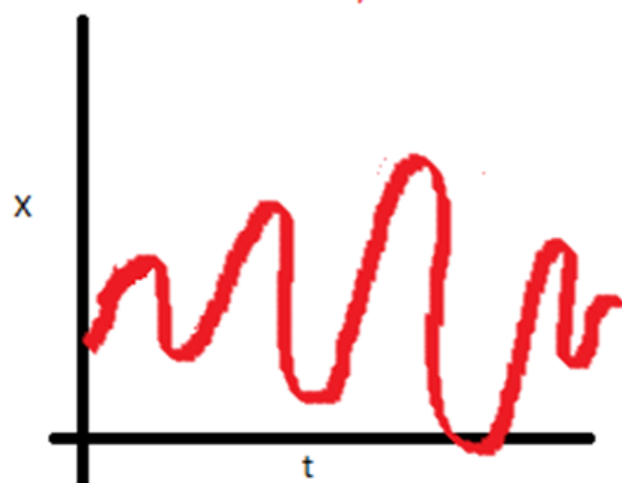
Stationary series



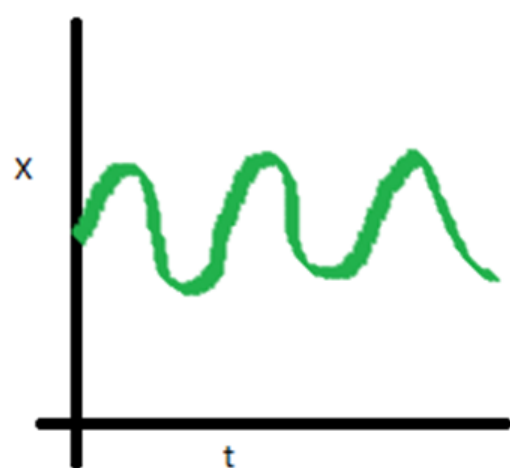
Non-Stationary series



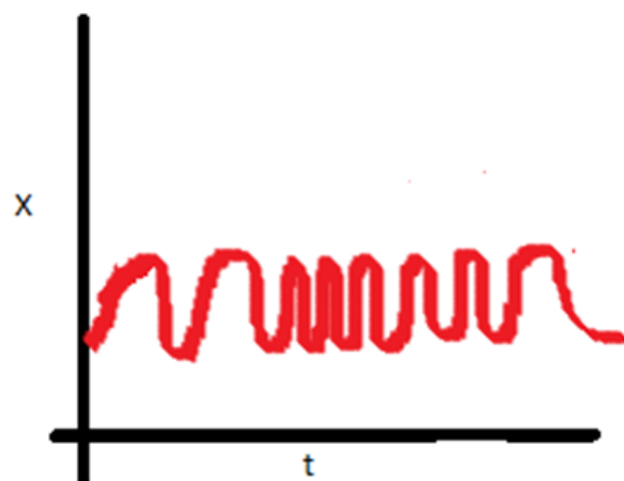
Stationary series



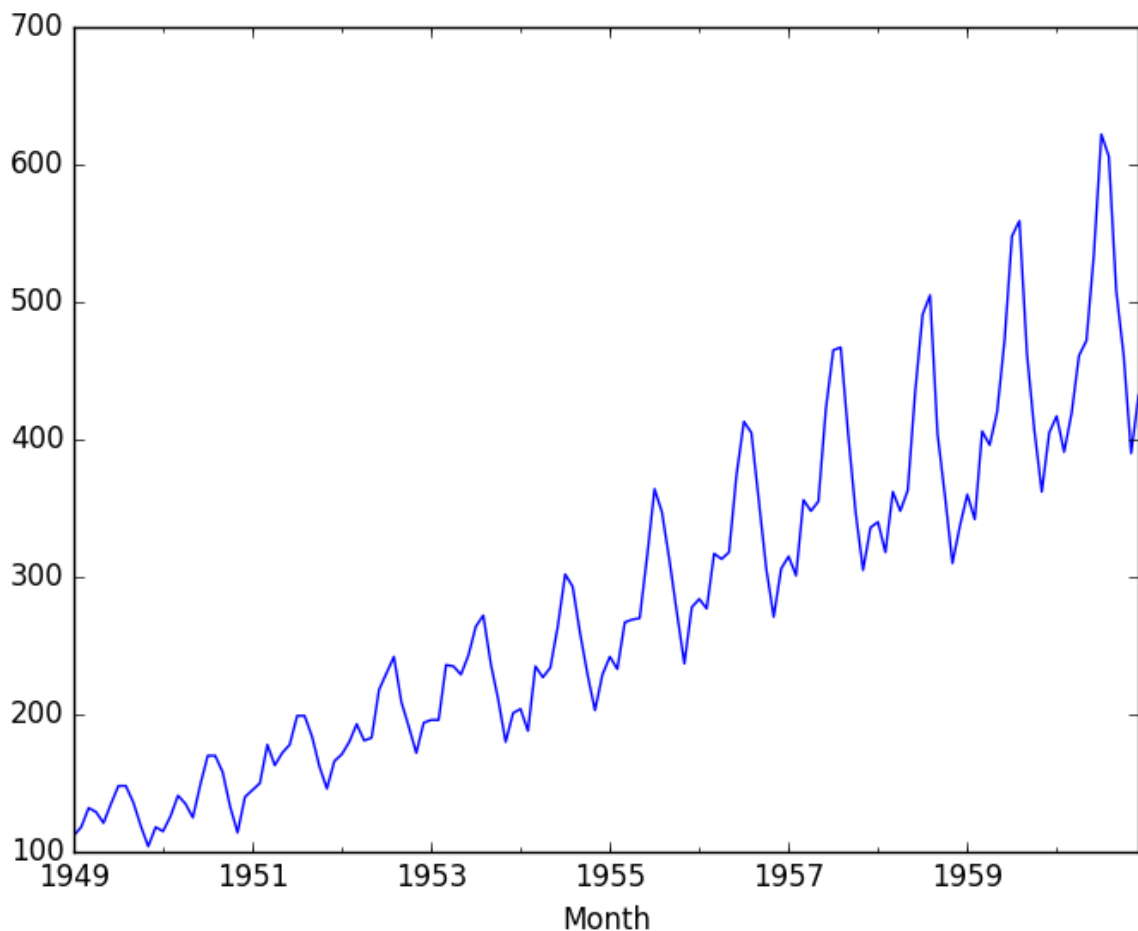
Non-Stationary series



Stationary series



Non-Stationary series



▼ Series Temporales

¿Por qué es bueno que las series sean estacionarias?

- Con series estacionarias podemos obtener predicciones fácilmente.
- Como la media es constante, podemos estimarla con todos los datos, y utilizar este valor para predecir una nueva observación.
- También se pueden obtener intervalos de predicción (confianza) para las predicciones asumiendo que X_t sigue una distribución conocida, por ejemplo, normal.

Series temporales: Componentes

Componentes de una serie temporal

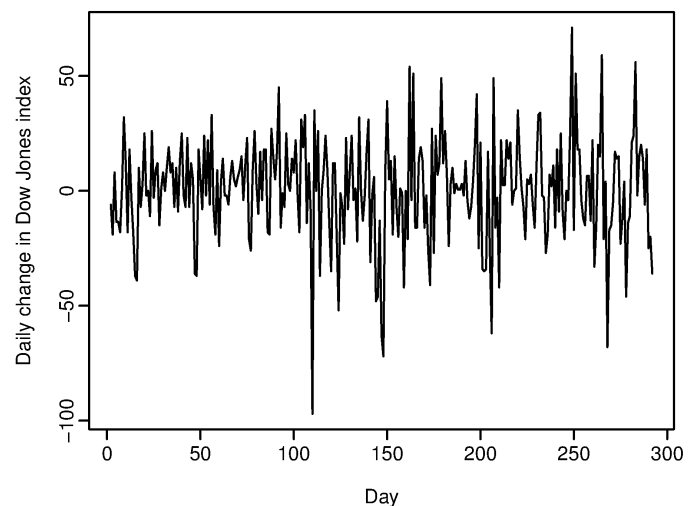
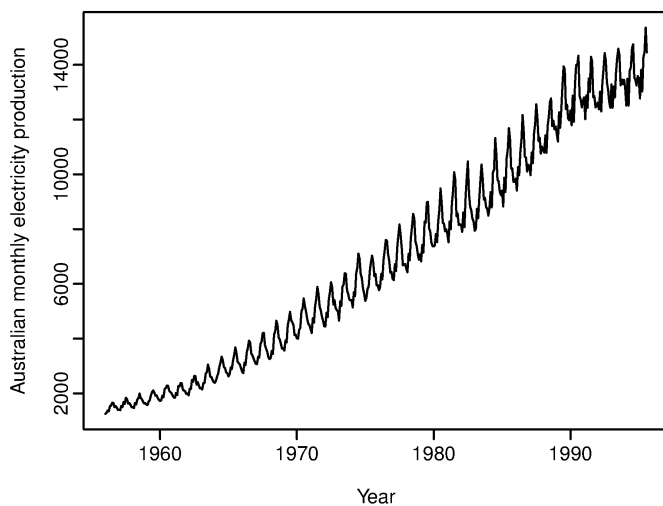
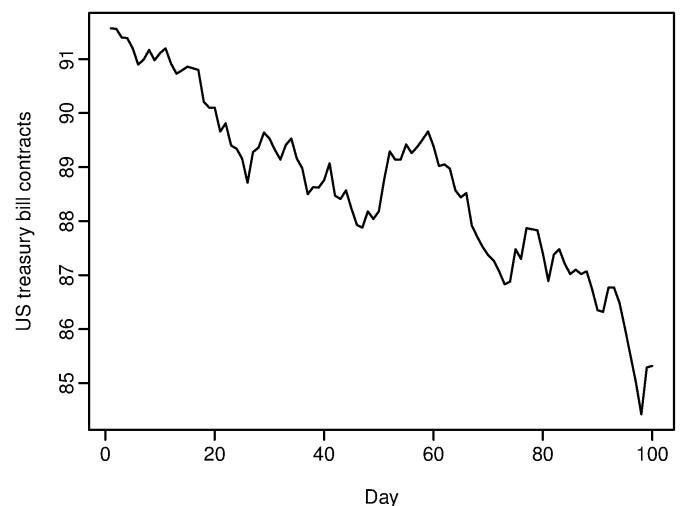
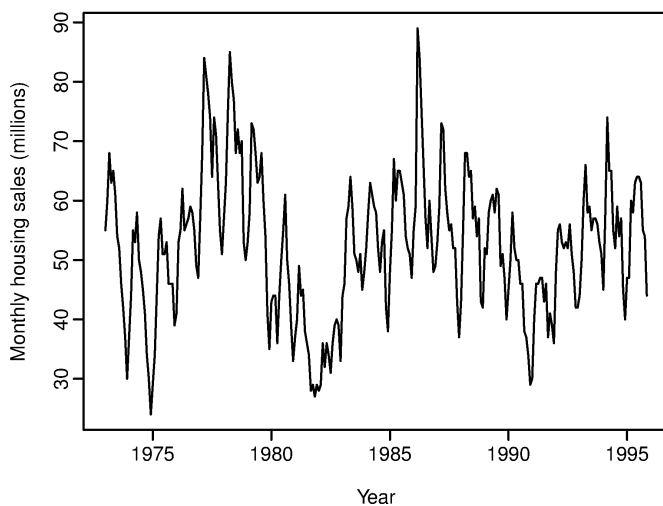
- En muchos casos, se supone que la serie temporal es la suma de varias componentes:

$$Y_t = T_t + S_t + E_t$$

Valor observado = Tendencia + Estacionalidad + Irregular

- **Tendencia:** comportamiento o movimiento suave de la serie a largo plazo. Puede ser creciente o decreciente y no tiene por que ser lineal. A veces incluye el ciclo (tendencia_ciclo).
- **Estacionalidad:** movimientos de oscilación dentro del año (trimestral, mensual, diario). La estacionalidad siempre es de un periodo fijo y conocido. No confundir con los ciclos que no tienen duración fija y como mínimo son de dos años.
- **Irregular:** variaciones aleatorias alrededor de los componentes anteriores.

▼ Componentes



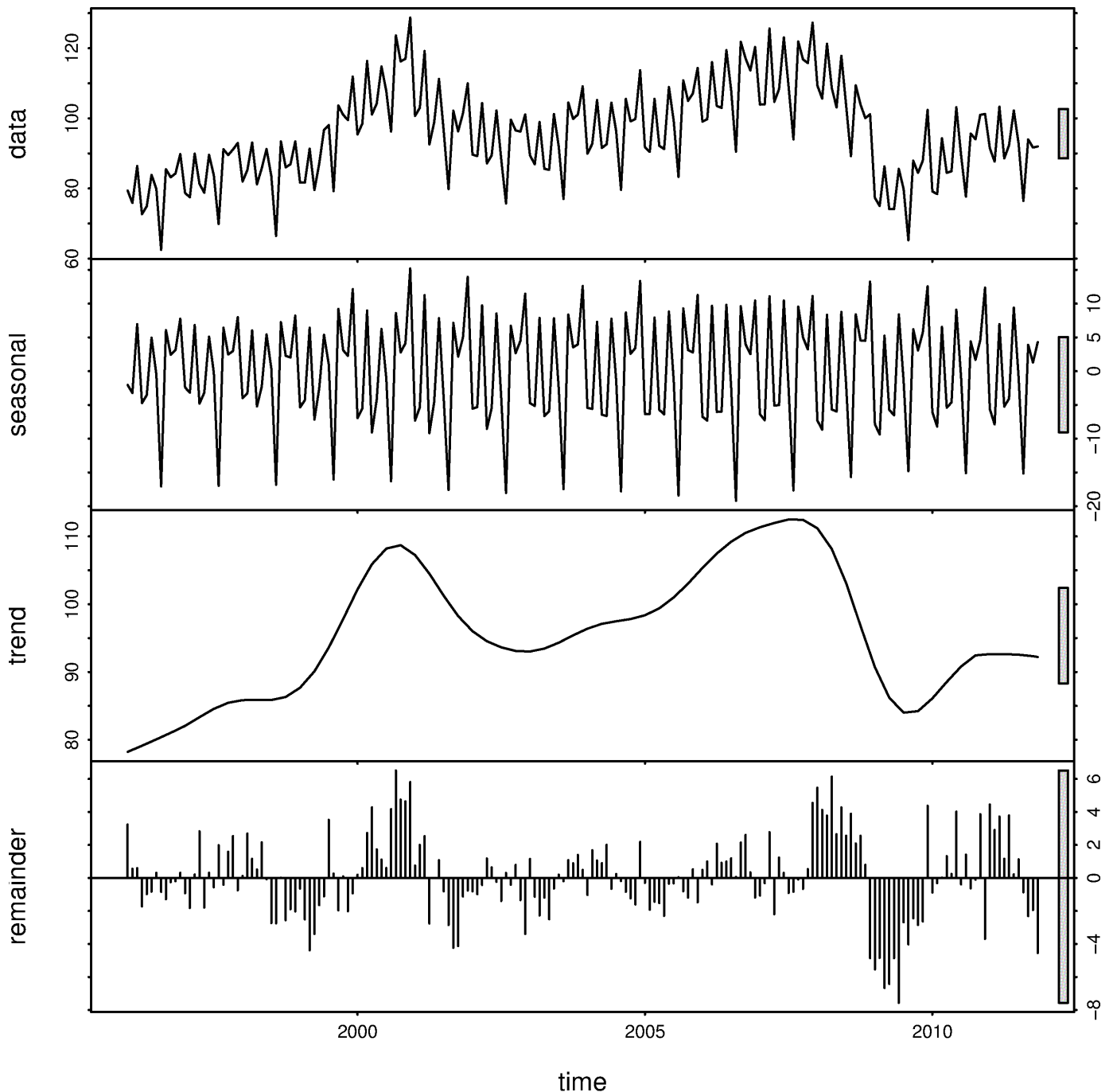
- Las ventas de casas muestra un componente estacional y uno cíclico. No parece que exista tendencia.
- Los t-bill no son estacionales pero tiene tendencia decreciente.
- La producción de electricidad tiene una fuerte tendencia y una fuerte estacionalidad.
- Los rendimientos del Dow Jones no tiene ni tendencia ni componentes estacional.

Modelo Multiplicativo

- El modelo también se puede expresar en forma multiplicativa cuando las variaciones alrededor de la tendencia son proporcionales al nivel de la serie temporal.
- Para ello se debe realizar una transformación logarítmica para que la variación de la serie sea estable:

$$Y_t = T_t \times S_t \times E_t$$
$$\ln Y_t = \ln T_t + \ln S_t + \ln E_t$$

Ejemplo Descomposición:



Suavizado Exponencial



Modelos de Suavizado Exponencial

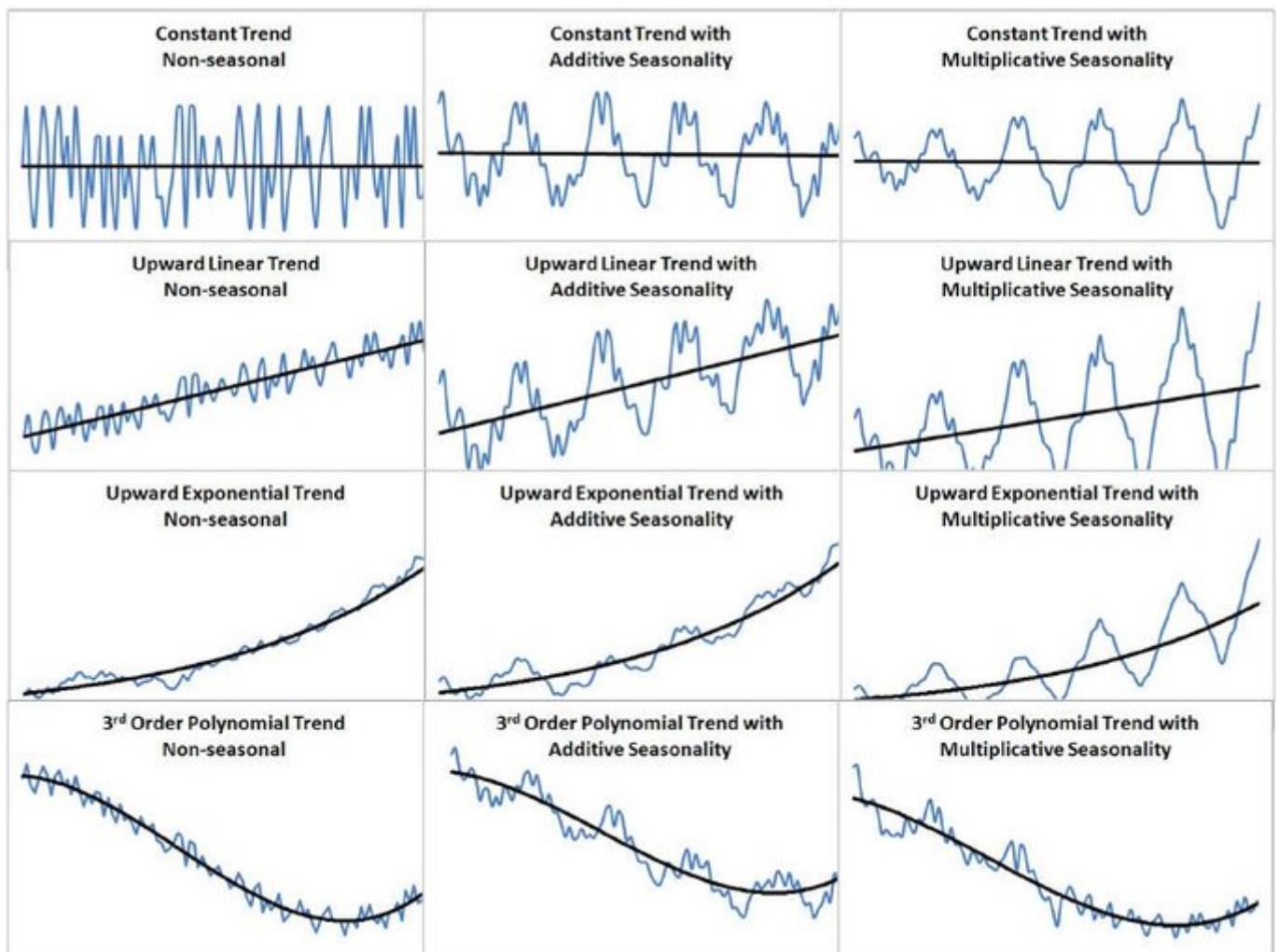
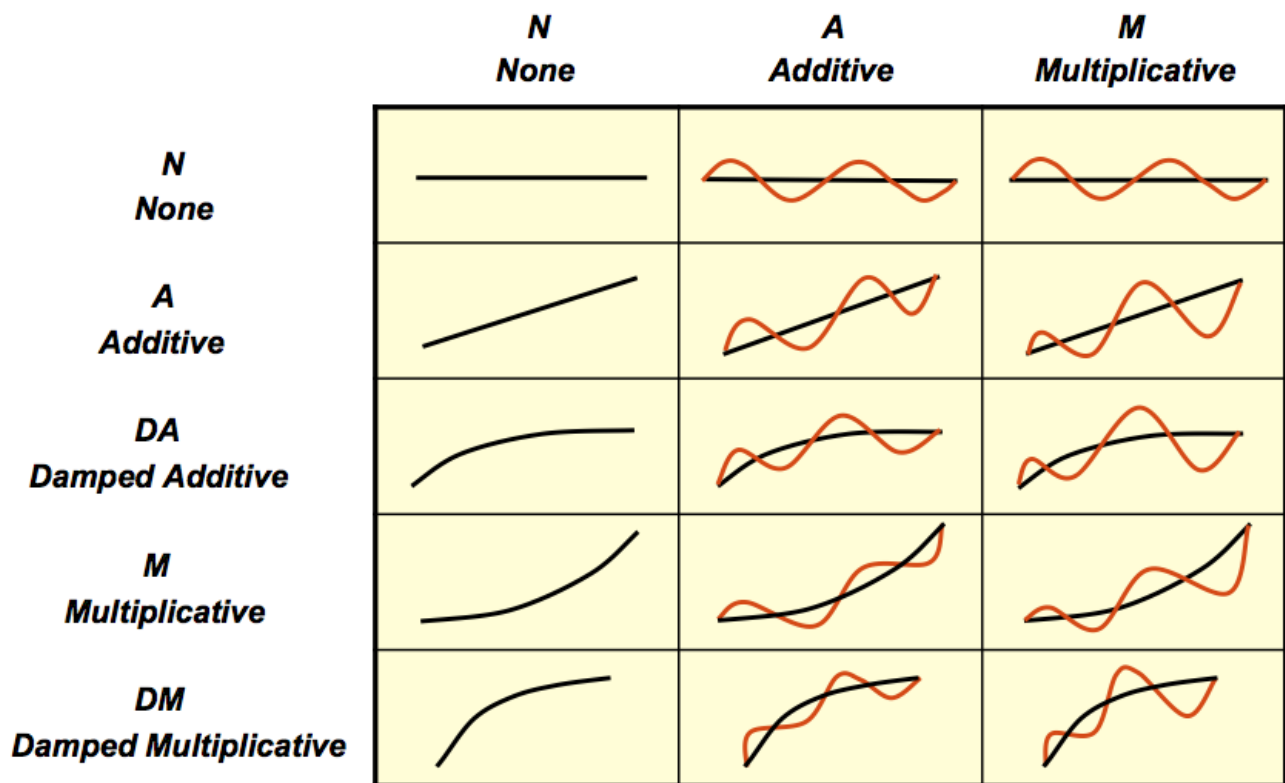
- Se emplean fundamentalmente para predecir nuevos valores de la serie.
- Se basan en modelos paramétricos deterministas que se ajustan a la evolución de la serie.
- Las observaciones más recientes tienen más peso en la predicción que las más alejadas.
- Se resuelven por métodos recursivos.

▼ Modelos ETS

$$Y_t = T_t + S_t + E_t \rightarrow \text{Modelo Aditivo}$$

$$Y_t = T_t \times S_t \times E_t \rightarrow \text{Modelo Multiplicativo}$$

Trend Component		Seasonal Component		
		N (None)	A  (Additive)	M  (Multiplicative)
N	(None)	N,N	N,A	N,M
A	(Additive)	A,N	A,A	A,M
A _d	(Additive damped)	A _d ,N	A _d ,A	A _d ,M
M	(Multiplicative)	M,N	M,A	M,M
M _d	(Multiplicative damped)	M _d ,N	M _d ,A	M _d ,M



(N,N)	=	simple exponential smoothing
(A,N)	=	Holts linear method
(M,N)	=	Exponential trend method
(A _d ,N)	=	additive damped trend method
(M _d ,N)	=	multiplicative damped trend method
(A,A)	=	additive Holt-Winters method
(A,M)	=	multiplicative Holt-Winters method
(A _d ,M)	=	Holt-Winters damped method



Selección de Modelos

¿Cómo seleccionar un modelo de entre varios candidatos?

- Utilizando los **Criterios de Información (IC)**:
 - Cuantifican la información residual. Cuanto más información tienen los residuos del modelo mayor es el IC, por lo tanto peor es el modelo.
 - Suponemos que tenemos k modelos alternativos, $M_1, M_2, \dots, M_j, \dots, M_k$, Se elige el j que **minimice**:

$$IC(j) = \ln \hat{\sigma}_j^2 + \frac{C(T)}{T}$$
 donde $\hat{\sigma}_j^2$ es la varianza residual del modelo, T es el tamaño muestral y $C(T)$ es el término de penalización (penalty).
 - AIC (Akaike Information Criterium): $C(T)=2$
 - BIC (Bayesian Information Criterium): $C(T)=\ln T$
 - HQ (Hannan-Quin Information Criterium): $C(T)=2 \cdot \ln(\ln T)$

Precisión de las Predicciones

Precisión de las Predicciones

- Error de predicción:

$$e_t = y_t - \hat{y}_t \quad t = 1 \dots h$$

- Medidas Absolutas (dependen de las unidades de medida):
 - Mean absolute error (MAE) = $\sum \frac{|e_t|}{h}$
 - Mean squared error (MSE) = $\sum \frac{e_t^2}{h}$
 - Root mean squared error (RMSE) = \sqrt{MSE}

Precisión de las Predicciones

- Medidas Relativas (expresadas en %, y no dependen de las unidades de medida)
 - Porcentaje de error: $p_t = 100 \frac{e_t}{y_t}$
 - Mean absolute percentage error (MAPE) = $\sum \frac{|p_t|}{h}$

Precisión de las Predicciones

- Medidas Escaladas (se escalan los errores con respecto al modelo naïve)
 - Modelo naïve: $\hat{y}_t = y_{t-1}$
 - Error modelo naïve: $n_t = y_t - y_{t-1}$
 - Scaled error: $q_t = \frac{e_t}{\sum \frac{|n_t|}{h}}$
 - Mean absolute scaled error (MASE): $\sum \frac{|q_t|}{h}$

CP 01: Predicción Ingresos Coca-Cola (KO)


▼ Introducción

- El objetivo es predecir las ventas de Coca-Cola.
- Se realizan diferentes predicciones de las ventas de Coca-Cola.
- Se tienen datos trimestrales desde 1991-1T hasta 2021-2T.
- Se dejan fuera de la estimación los ocho últimos trimestres para seleccionar el mejor modelo.
- Se prueban todos los modelos de suavizados exponencial.

```
# Packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```


```
# Read Data
ko_df = pd.read_csv('/content/mnt/My Drive/Colab Notebooks/Series Temporales/S01/ko')
ko_df['Fecha'] = pd.to_datetime(ko_df['Fecha'], format="%Y%m%d")
ko_df = ko_df.set_index('Fecha')
```

```
ko_df.head()
```

	Ingresos 
Fecha	
2021-06-03	10129
2021-03-03	9020
2020-12-31	8611
2020-09-26	8652
2020-06-27	7150

Convertimos los datos en trimestrales

```
ko_ts=ko_df.resample("q").last()  
#ko_ts=ko_df['Ingresos'].astype('float64').to_period('Q')  
ko_ts.tail()
```

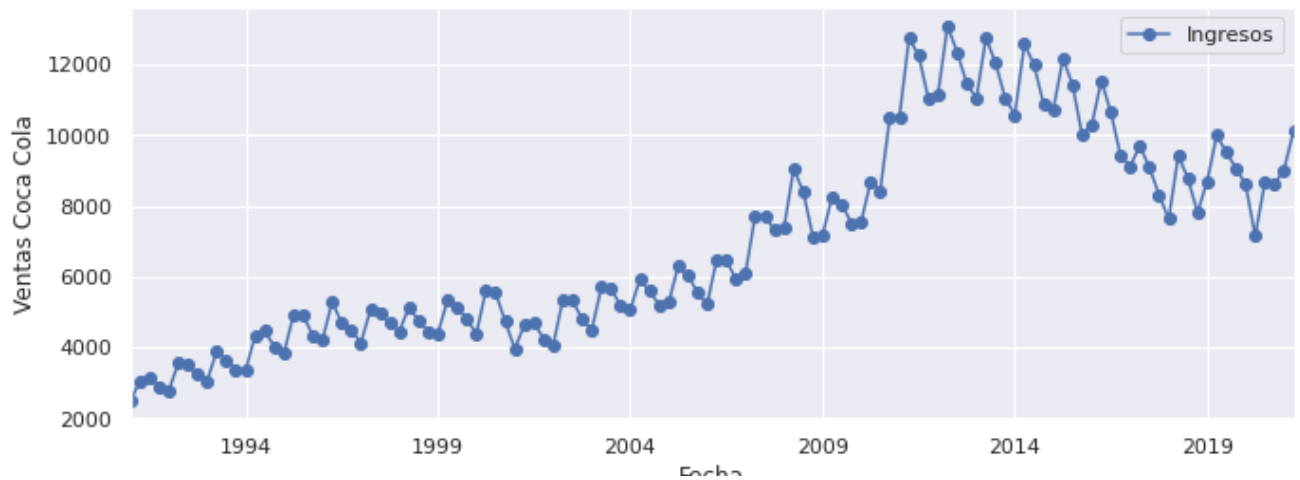
	Ingresos 
Fecha	
2020-06-30	7150
2020-09-30	8652
2020-12-31	8611
2021-03-31	9020
2021-06-30	10129

▼ Solución

Graficar los Ingresos

- Tendencia - Componente Estacional - Varianza no constante

```
import seaborn as sns  
# Use seaborn style defaults and set the default figure size  
sns.set(rc={'figure.figsize':(11, 4)})  
ax = ko_ts.plot(marker='o', linestyle='-')  
ax.set_ylabel('Ventas Coca Cola');
```



Analizamos la estacionalidad

- Componente estacional
- Las Ventas dependen del trimestre
- Más ventas en el 2T y en el 3T
- El Componente estacional no es estacionario

```
import statsmodels.api as sm
```

```
ax = plt.gca()
sm.graphics.tsa.quarter_plot(ko_ts['Ingresos'],ax=ax)
ax.set_title('Comportamiento Estacional')
```

```
Text(0.5, 1.0, 'Comportamiento Estacional')
```



▼ Modelos de Suavizado Exponencial

Vamos a separar la muestra en la parte de estimación (Training) y la parte de predicción/Verificación (Testing). Quitamos 8 trimestres.

Vamos a predecir 4 periodos (un año) (h=4)

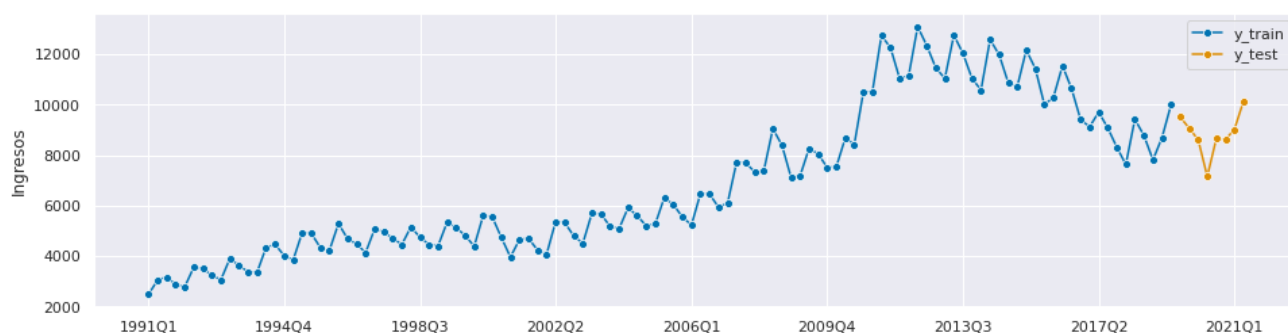
```
from sktime.forecasting.base import ForecastingHorizon
from sktime.utils.plotting import plot_series
from sktime.forecasting.model_selection import temporal_train_test_split
```

```
ko_ts['Ingresos'].astype('float64').to_period('Q')
```

```
Fecha
1991Q1    2480.0
1991Q2    3039.0
1991Q3    3172.0
1991Q4    2879.0
1992Q1    2772.0
...
2020Q2    7150.0
2020Q3    8652.0
2020Q4    8611.0
2021Q1    9020.0
2021Q2    10129.0
Freq: Q-DEC, Name: Ingresos, Length: 122, dtype: float64
```

```
y_train, y_test = temporal_train_test_split(y = ko_ts['Ingresos'].astype('float64'))
# we will try to forecast y_test from y_train
# plotting for illustration
plot_series(y_train, y_test, labels=["y_train", "y_test"])
print(y_train.shape[0], y_test.shape[0])
```

114 8



Se observa la necesidad de incluir componente estacional.

```
from sktime.forecasting.ets import AutoETS
```

```
# step 2: specifying forecasting horizon
fh = np.arange(1, 17)

# step 3: specifying the forecasting algorithm
ko_auto_model = AutoETS(auto=True, sp=4, n_jobs=-1)
```

```
ko_auto_model.fit(y_train)
```

```
AutoETS(auto=True, n_jobs=-1, sp=4)
```

```
print(ko_auto_model.summary())
```

```

                                ETS Results
=====
Dep. Variable:                  Ingresos      No. Observations:                  114
Model:                          ETS(MAM)      Log Likelihood                    -828.866
Date:                          Sun, 25 Sep 2022  AIC                        1677.731
Time:                          11:46:31      BIC                              1705.093
Sample:                        03-31-1991     HQIC                             1688.836
                                - 06-30-2019   Scale                           0.003
Covariance Type:                approx
=====
                                coef      std err          z      P>|z|      [0.025
-----
smoothing_level                0.9999      0.121      8.246      0.000      0.762
smoothing_trend                9.999e-05      nan      nan      nan      nan
smoothing_seasonal            4.239e-05      nan      nan      nan      nan
initial_level                 2902.6074      nan      nan      nan      nan
initial_trend                 73.3890      nan      nan      nan      nan
initial_seasonal.0            1.0390      nan      nan      nan      nan
initial_seasonal.1            1.1352      nan      nan      nan      nan
initial_seasonal.2            1.1936      nan      nan      nan      nan
initial_seasonal.3            1.0000      nan      nan      nan      nan
=====
Ljung-Box (Q):                  4.90      Jarque-Bera (JB):                  9
Prob(Q):                       0.77      Prob(JB):
Heteroskedasticity (H):        2.36      Skew:
Prob(H) (two-sided):           0.01      Kurtosis:
=====
```

```
Warnings:
```

```
[1] Covariance matrix calculated using numerical (complex-step) differentiatio
```

```
# step 5: querying predictions
ko_pred = ko_auto_model.predict(fh)
print(ko_pred)
```

```
2019Q3      9591.389872
2019Q4      8853.969701
2020Q1      8594.952101
```

```

2020Q2      10346.028127
2020Q3      9923.333013
2020Q4      9157.763414
2021Q1      8887.350349
2021Q2      10695.029335
2021Q3      10255.276153
2021Q4      9461.557128
2022Q1      9179.748597
2022Q2      11044.030543
2022Q3      10587.219293
2022Q4      9765.350841
2023Q1      9472.146845
2023Q2      11393.031750
Freq: Q-DEC, dtype: float64

```

```

ko_pred_ints = ko_auto_model.predict_interval(fh, coverage=0.9)
ko_pred_ints

```

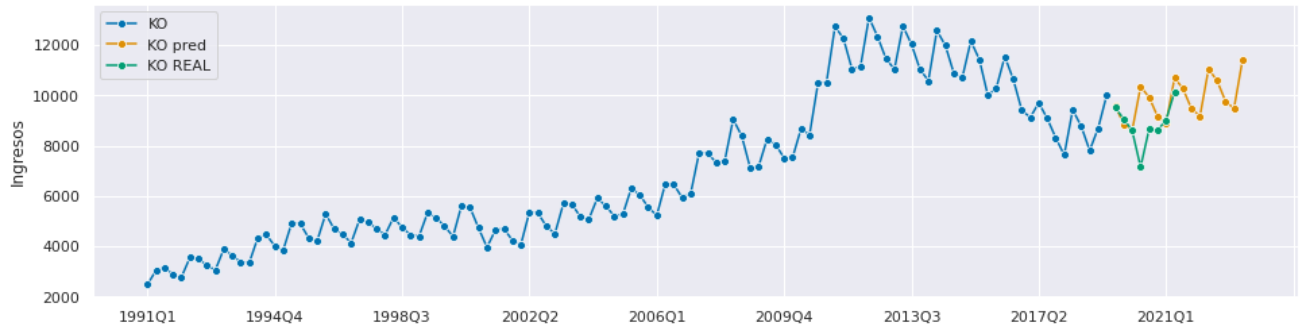
	Coverage	
	0.9	
	lower	upper
2019Q3	8705.280749	10436.243032
2019Q4	7697.298211	9946.818797
2020Q1	7247.372905	9958.127255
2020Q2	8510.784281	12288.077361
2020Q3	8071.813201	12093.502643
2020Q4	7283.716351	11190.941105
2021Q1	6904.223998	11125.933896
2021Q2	8292.387636	13555.588992
2021Q3	7735.655883	13124.229589
2021Q4	7041.407272	12265.083426
2022Q1	6739.015208	12033.671137
2022Q2	8010.033674	14609.324064
2022Q3	7597.340071	14415.696275
2022Q4	6897.289484	13241.737406
2023Q1	6511.050275	13029.030958
2023Q2	7895.518157	15941.590043

```

# optional: plotting predictions and past data
plot_series(y_train, ko_pred,y_test, labels=["KO", "KO pred", "KO REAL"])

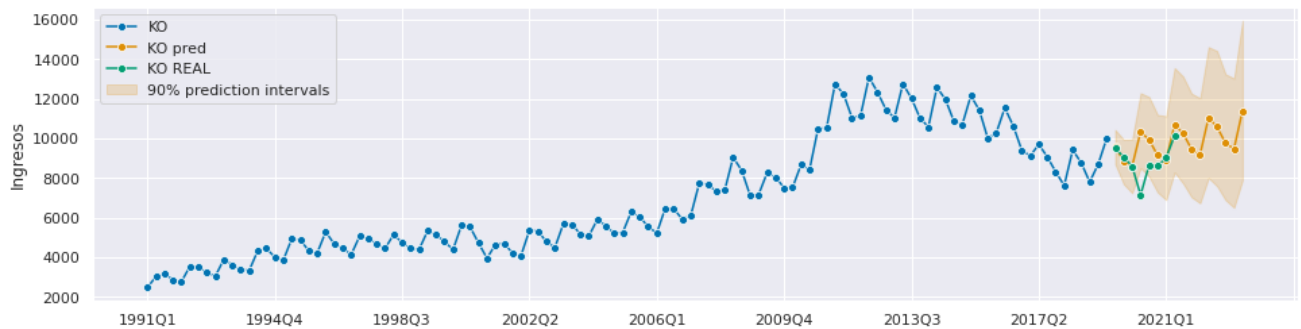
```

(<Figure size 1152x288 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f5e711fd410>)



```
fig, ax = plot_series(y_train, ko_pred, y_test, labels=["KO", "KO pred", "KO REAL"])
ax.fill_between(
    ax.get_lines()[-2].get_xdata(),
    ko_pred_ints[('Coverage', 0.9, 'lower')],
    ko_pred_ints[('Coverage', 0.9, 'upper')],
    alpha=0.2,
    color=ax.get_lines()[-2].get_c(),
    label=f"90% prediction intervals",
)
ax.legend(loc='upper left')
```

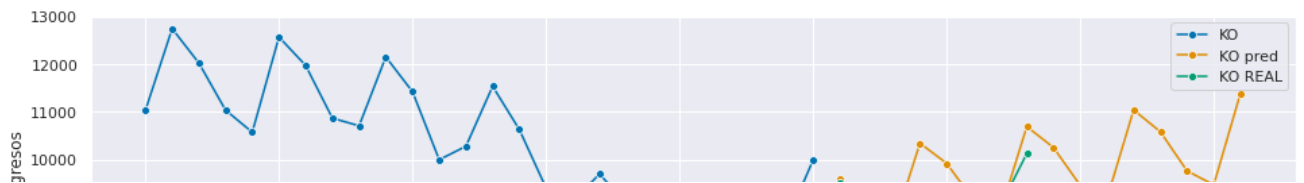
(<matplotlib.legend.Legend at 0x7f5e6d5c2850>



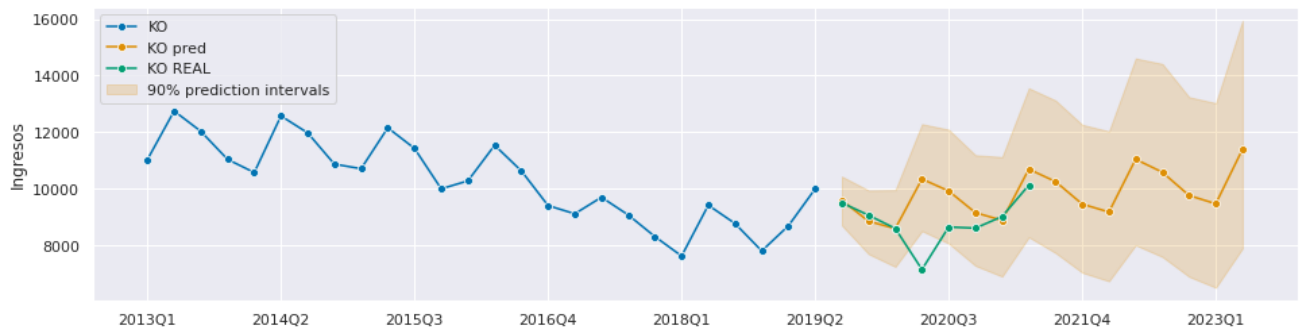
```
plot_series(y_train["2013":], ko_pred, y_test, labels=["KO", "KO pred", "KO REAL"])
```



```
(<Figure size 1152x288 with 1 Axes>,
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e708d4290>)
```



```
fig, ax = plot_series(y_train["2013":], ko_pred, y_test, labels=["KO", "KO pred", "
ax.fill_between(
    ax.get_lines()[-2].get_xdata(),
    ko_pred_ints[('Coverage', 0.9, 'lower')],
    ko_pred_ints[('Coverage', 0.9, 'upper')],
    alpha=0.2,
    color=ax.get_lines()[-2].get_c(),
    label=f"90% prediction intervals",
)
ax.legend(loc='upper left');
```



Comprobemos la precisión de las predicciones

```
from sktime.performance_metrics.forecasting import mean_absolute_percentage_error
# option 1: using the lean function interface
mean_absolute_percentage_error(y_test, ko_pred[0:8])
```

```
0.0951505761366288
```

```
from sktime.performance_metrics.forecasting import MeanSquaredError
mse = MeanSquaredError()
mse(y_test, ko_pred[0:8])
```

```
1565098.2481350685
```

```
rmse = MeanSquaredError(square_root=True)
rmse(y_test, ko_pred[0:8])
```

```
1251.038867555708
```

Prediccion para final de año 2021 y 2022

```
# step 2: specifying forecasting horizon
fh = np.arange(1, 7)

# step 3: specifying the forecasting algorithm
ko_auto_model = AutoETS(auto=True, sp=4, n_jobs=-1)

y = ko_ts['Ingresos'].astype('float64').to_period('Q')

ko_auto_model.fit(y)

print(ko_auto_model.summary())
```

```

=====
                        ETS Results
=====
Dep. Variable:          Ingresos      No. Observations:      122
Model:                  ETS(MAM)      Log Likelihood         -905.927
Date:                   Sun, 25 Sep 2022  AIC                  1831.853
Time:                   11:42:36        BIC                  1859.893
Sample:                 03-31-1991      HQIC                  1843.242
                        - 06-30-2021      Scale                  0.004
Covariance Type:        approx
=====

```

	coef	std err	z	P> z	[0.025
smoothing_level	0.8438	0.087	9.652	0.000	0.672
smoothing_trend	8.438e-05	nan	nan	nan	nan
smoothing_seasonal	0.1179	0.046	2.567	0.010	0.028
initial_level	2892.1665	nan	nan	nan	nan
initial_trend	53.4918	18.626	2.872	0.004	16.985
initial_seasonal.0	1.0491	nan	nan	nan	nan
initial_seasonal.1	1.1644	nan	nan	nan	nan
initial_seasonal.2	1.1958	nan	nan	nan	nan
initial_seasonal.3	1.0000	nan	nan	nan	nan

```
=====
Ljung-Box (Q):          4.02      Jarque-Bera (JB):          6
Prob(Q):                0.86      Prob(JB):
Heteroskedasticity (H):  2.15      Skew:
Prob(H) (two-sided):    0.02      Kurtosis:
=====
```

Warnings:

```
[1] Covariance matrix calculated using numerical (complex-step) differentiat
```

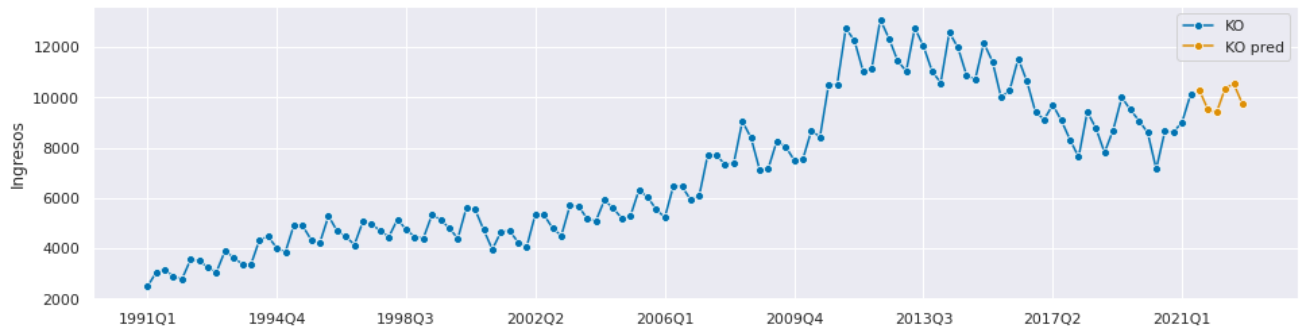
```
# step 5: querying predictions
ko_pred = ko_auto_model.predict(fh)
print(ko_pred)
```

```
2021Q3    10282.769738
2021Q4     9506.523609
2022Q1     9401.106815
2022Q2    10361.900677
2022Q3    10527.174373
```

```
2022Q4      9731.143430
Freq: Q-DEC, dtype: float64
```

```
plot_series(y, ko_pred, labels=["KO", "KO pred"])
```

```
(<Figure size 1152x288 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f5e6de44c50>)
```



Estimemos el modelo de forma manual

```
from sktime.forecasting.exp_smoothing import ExponentialSmoothing
forecaster = ExponentialSmoothing(trend='additive', seasonal='multiplicative', sp=4)
forecaster.fit(y)
```

```
ExponentialSmoothing(seasonal='multiplicative', sp=4, trend='additive')
```

```
y_pred = forecaster.predict(fh)
y_pred
```

```
2021Q3      10223.765652
2021Q4      9456.334854
2022Q1      9373.310362
2022Q2      10430.163683
2022Q3      10535.227084
2022Q4      9742.239445
Freq: Q-DEC, Name: Ingresos, dtype: float64
```

```
print(forecaster._fitted_forecaster.summary())
```

```

ExponentialSmoothing Model Results
=====
Dep. Variable:      Ingresos      No. Observations:      1
Model:      ExponentialSmoothing      SSE      29538439.8
Optimized:      True      AIC      1528.4
Trend:      Additive      BIC      1550.8
```

```

Seasonal:                      Multiplicative      AICC                      1530.4
Seasonal Periods:              4                   Date:                      Sun, 25 Sep 20
Box-Cox:                       False                Time:                      11:43:
Box-Cox Coeff.:                None
=====

```

	coeff	code	optimized
smoothing_level	0.8889286	alpha	T
smoothing_trend	0.0001000	beta	T
smoothing_seasonal	0.1110714	gamma	T
initial_level	2902.6083	l.0	T
initial_trend	73.389214	b.0	T
initial_seasons.0	0.9139023	s.0	T
initial_seasons.1	1.1063511	s.1	T
initial_seasons.2	1.0795056	s.2	T
initial_seasons.3	0.9970990	s.3	T

```
forecaster.get_fitted_params()
```

```

{'initial_level': 2902.6083250886268,
 'initial_slope': None,
 'initial_seasons': array([0.91390226, 1.10635106, 1.07950563, 0.99709896]),
 'aic': 1528.4562007472675,
 'bic': 1550.8883691051335,
 'aicc': 1530.4381827292495}

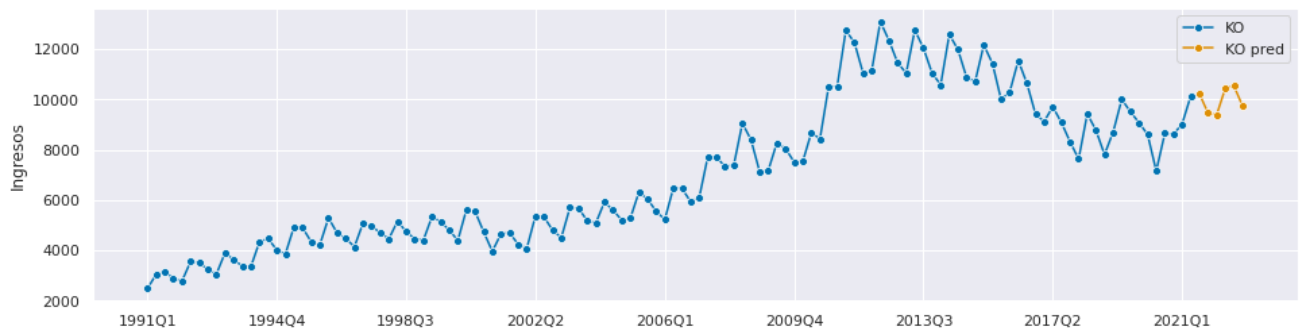
```

```
plot_series(y, y_pred, labels=["KO", "KO pred"])
```

```

(<Figure size 1152x288 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f5e6dee1550>)

```



Estimemos el modelo de forma manual sin componente estacional

```

forecaster = ExponentialSmoothing(trend='additive',seasonal=None, sp=4)
forecaster.fit(y)

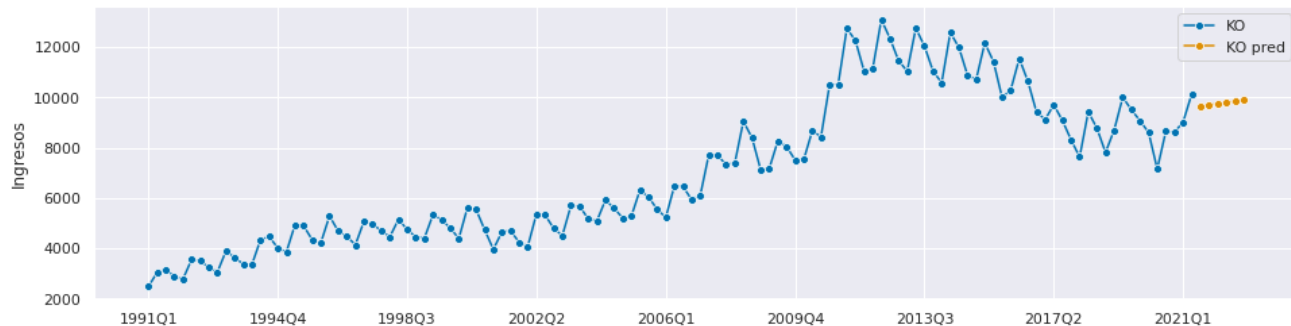
ExponentialSmoothing(sp=4, trend='additive')

```

```
y_pred = forecaster.predict(fh)
```

```
plot_series(y, y_pred, labels=["KO", "KO pred"])
```

```
(<Figure size 1152x288 with 1 Axes>,  
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e708592d0>)
```



Estimemos el modelo de forma manual sin componente estacional y tendencia multiplicativa

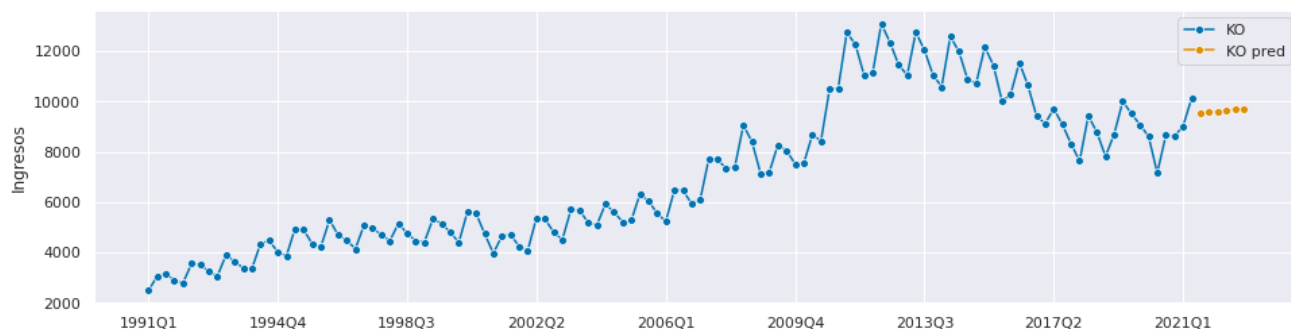
```
forecaster = ExponentialSmoothing(trend='mul',seasonal=None, sp=4)
```

```
forecaster.fit(y)
```

```
y_pred = forecaster.predict(fh)
```

```
plot_series(y, y_pred, labels=["KO", "KO pred"])
```

```
(<Figure size 1152x288 with 1 Axes>,  
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e6de0e090>)
```



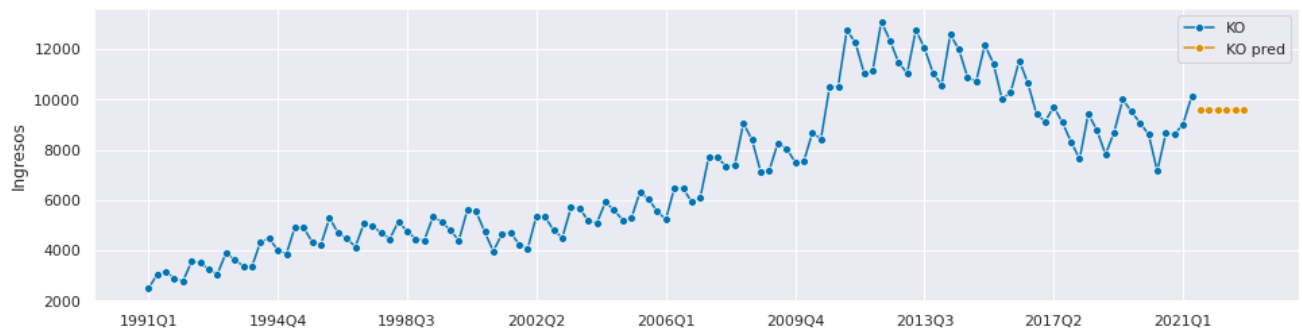
Estimemos el modelo de forma manual sin componente estacional y sin tendencia

```
forecaster = ExponentialSmoothing(trend=None,seasonal=None, sp=4)
```

```
forecaster.fit(y)
```

```
y_pred = forecaster.predict(fh)
plot_series(y, y_pred, labels=["KO", "KO pred"])
```

```
(<Figure size 1152x288 with 1 Axes>,
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e6dc99310>)
```

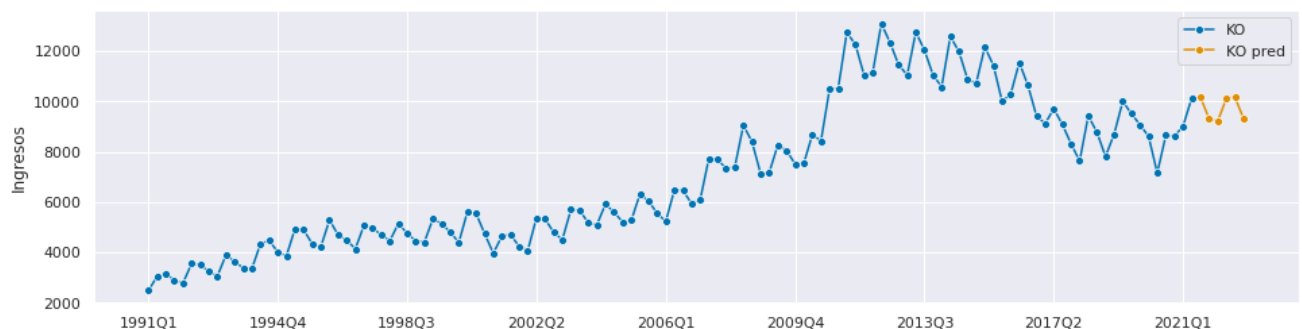


Estimemos el modelo de forma manual sin tendencia

Componente estacional multiplicativo

```
forecaster = ExponentialSmoothing(trend=None,seasonal="mul", sp=4)
forecaster.fit(y)
y_pred = forecaster.predict(fh)
plot_series(y, y_pred, labels=["KO", "KO pred"])
```

```
(<Figure size 1152x288 with 1 Axes>,
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e6dc3cb90>)
```



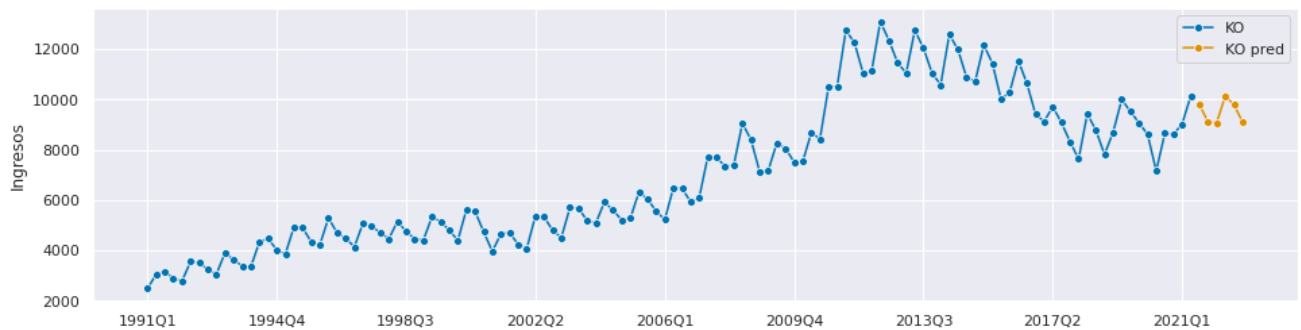
Estimemos el modelo de forma manual sin tendencia

Componente estacional aditivo

```
forecaster = ExponentialSmoothing(trend=None,seasonal="add", sp=4)
forecaster.fit(y)
```

```
y_pred = forecaster.predict(fh)
plot_series(y, y_pred, labels=["KO", "KO pred"])
```

```
(<Figure size 1152x288 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f5e6dc7f910>)
```



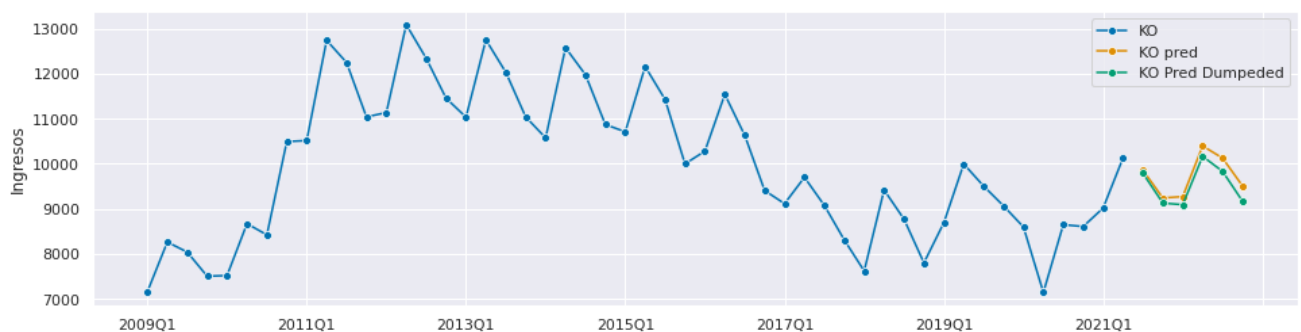
Comparar modelo con tendencia aditiva con y sin dumped

Componente estacional aditivo

```
forecaster = ExponentialSmoothing(trend="add",seasonal="add",damped_trend=False, sp=
forecaster.fit(y)
y_pred = forecaster.predict(fh)
```

```
forecaster = ExponentialSmoothing(trend="add",seasonal="add",damped_trend=True, sp=
forecaster.fit(y)
y_pred_dump = forecaster.predict(fh)
plot_series(y["2009":], y_pred, y_pred_dump,labels=["KO", "KO pred","KO Pred Dumped"])
```

```
(<Figure size 1152x288 with 1 Axes>,
 <matplotlib.axes._subplots.AxesSubplot at 0x7f5e6db40910>)
```



Productos de pago de Colab - Cancelar contratos

✓ 0 s completado a las 13:51

