Análisis de Series de Tiempo

Carrera de Especialización para Inteligencia Artificial

Clase 5

Ing. Magdalena Bouza, Ing. Carlos German Carreño Romano

Temas

- Modelos de suavizado exponencial
- Predicciones
- Análisis de intervenciones
- Análisis de outliers
- Segunda entrega TP

Modelos de suavizado exponencial

Suavizado exponencial

El método de suavizado exponencial se suele ajustar bien a series estacionales. Tiene la ventaja de que es simple, requiere menos datos que los modelos ARIMA para predecir y responde bien a cambios en tendencia y estacionalidad de corto plazo.

Los pronósticos obtenidos con este método son promedios ponderados de observaciones pasadas. Los **pesos decaen exponencialmente**, por lo tanto las últimas observaciones serán las de mayor peso.

Vamos a usar una notación de componentes para describir estos modelos.

Ref.: *"Tráfico de Internet: Una Aplicación de Datos Georreferenciados y Series de Tiempo", F. De Arca, C. G. Carreño Romano, C. Righetti, G. Carro, RPIC 2019, Bahia Blanca, Argentina.*

Suavizado exponencial simple

Si y_t es la serie de tiempo, la predicción a un paso es:

$$\hat{y}_{t+1|t} = \alpha y_t + \alpha(1-\alpha) y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \dots$$

donde alpha es el parámetro de suavizado (0<alpha<1). Hace falta definir un valor para el momento inicial, es decir t=1 al que llamamos n_0:

$$\hat{y}_{t+1|t} = \alpha y_t + \alpha(1-\alpha)\hat{y}_{t|t-1}$$

$$\hat{y}_{t+1|t} = \sum_{j=0}^{t-1} \alpha(1-\alpha)^j y_{t-j} + (1-\alpha)^t n_0$$

Suavizado exponencial simple

Si usamos la notación de componentes, este modelo simple se puede describir a partir de una componente de *Nivel:*

$$Pronóstico \quad \hat{y}_{t+h|t} = n_t$$

$$Nivel \quad n_t = \alpha y_t + (1-\alpha) n_{t-1}$$

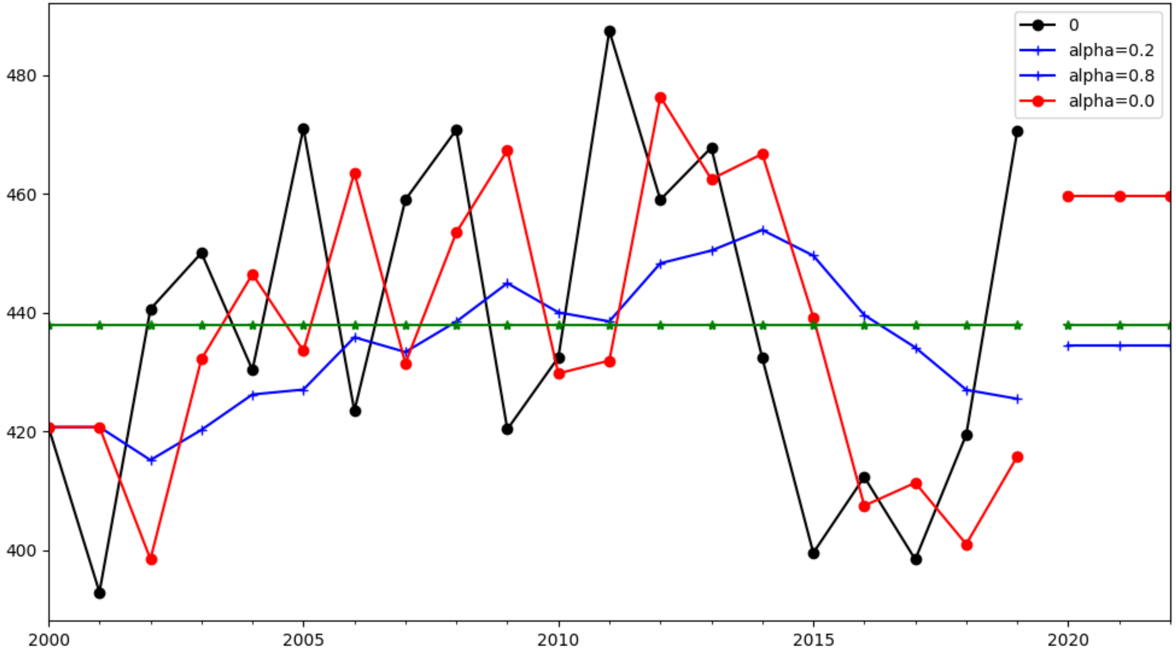
En python estos modelos están incluídos en statsmodels.

```
from statsmodels.tsa.api import SimpleExpSmoothing
```

```
ins1 = SimpleExpSmoothing(data).fit(smoothing_level=0.2,optimized=False)
```

Ref.: *https://www.statsmodels.org/dev/examples/notebooks/generated/exponential_smoothing.html*

Suavizado exponencial simple



Suavizado exponencial lineal de Holt

Esta extensión permite trabajar con series que tengan tendencia lineal. Con b_t la pendiente de la tendencia y Beta el parámetro de suavizado para la tendencia hay una expresión para el pronóstico y dos ecuaciones de componentes:

$$Pronóstico \quad \hat{y}_{t+h|t} = n_t + hb_t$$

$$Nivel \quad n_t = \alpha y_t + (1-\alpha)(n_{t-1} + b_{t-1})$$

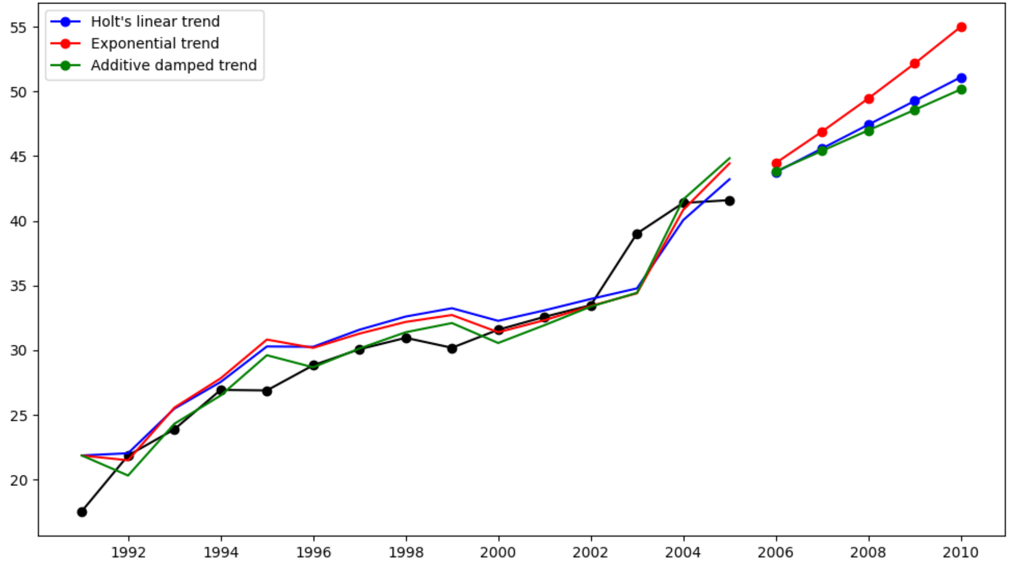
$$Tendencia \quad b_t = \beta(n_t - n_{t-1}) + (1-\beta)b_{t-1}$$

En este modelo la tendencia es constante indefinidamente. Para otros casos se define un factor de damping o amortiguación.

Suavizado exponencial lineal de Holt

```
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
```

```
fit1 = Holt(air).fit(smoothing_level=0.8, smoothing_slope=0.2, optimized=False)
fcast1 = fit1.forecast(5).rename("Holt's linear trend")
fit2 = Holt(air, exponential=True).fit(smoothing_level=0.8, smoothing_slope=0.4,
optimized=False)
fcast2 = fit2.forecast(5).rename("Exponential trend")
```



Suavizado exponencial de tendencia amortiguada

delta es el parámetro de amortiguación, definido entre (0,1).

$$Pronóstico \quad \hat{y}_{t+h|t} = n_t + (\delta + \delta^2 + ... + \delta^h) b_t$$

$$Nivel \quad n_t = \alpha y_t + (1 - \alpha)(n_{t-1} + \delta b_{t-1})$$

$$Tendencia \quad b_t = \beta(n_t - n_{t-1}) + (1 - \beta)\delta b_{t-1}$$

Holt - Winters

Holt-Winters resulta una generalización de estos métodos para poder trabajar con series estacionales, agregando una ecuación de suavizado para la componente estacional e_t. Se presenta el método aditivo y el multiplicativo:

Método aditivo:

$$Pronóstico \quad \hat{y}_{t+h|t} = n_t + hb_t + e_{t+h-m(k+1)}$$

$$Nivel \quad n_t = \alpha(y_t - e_{t-m} + (1-\alpha)(n_{t-1} + b_{t-1})$$

$$Tendencia \quad b_t = \beta(n_t - n_{t-1}) + (1-\beta)b_{t-1}$$

$$Estacionalidad \quad e_t = \gamma(y_t - n_{t-1} - b_{t-1}) + (1-\gamma)e_{t-m}$$

Holt - Winters

gamma es el parámetro de estacionalidad (0,1), m es el período estacional, por ejemplo 12 meses y $k = \lfloor (h-1)/k \rfloor$ es un índice.

Método multiplicativo:

$$Pronóstico \quad \hat{y}_{t+h|t} = (n_t + hb_t)e_{t+h-m(k+1)}$$

$$Nivel \quad n_t = \alpha \frac{y_t}{e_{t-m}} + (1-\alpha)(n_{t-1} + b_{t-1})$$

$$Tendencia \quad b_t = \beta(n_t - n_{t-1}) + (1-\beta)b_{t-1}$$

$$Estacionalidad \quad e_t = \gamma \frac{y_t}{n_{t-1} + b_{t-1}} + (1-\gamma)e_{t-m}$$

Holt - Winters

Agregando el factor de damping a este modelo se tiene que:

$$Pronóstico \quad \hat{y}_{t+h|t} = [n_t + (\delta + \delta^2 + \ldots + \delta^h) b_t] e_{t+h-m(k+1)}$$

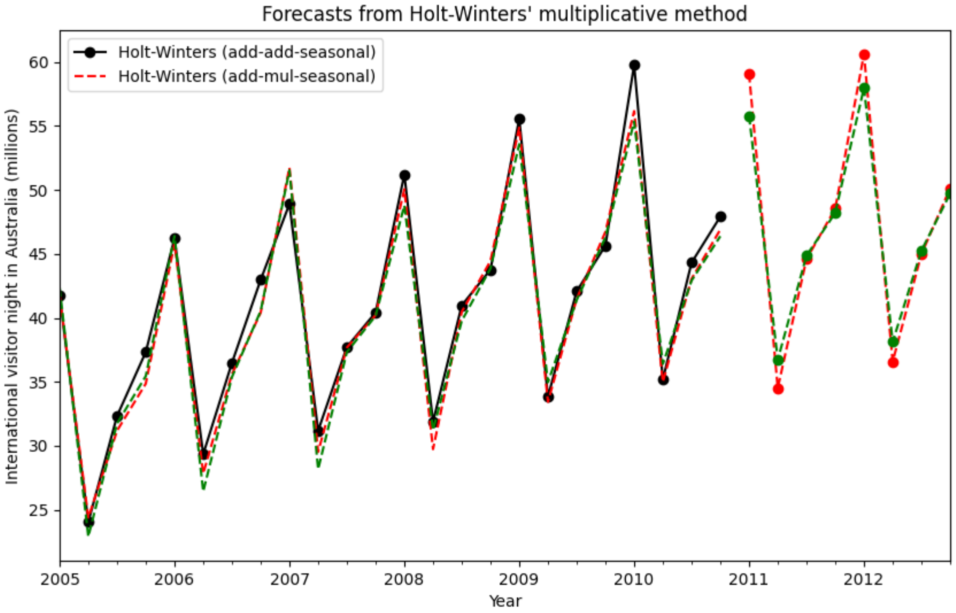
$$Nivel \quad n_t = \alpha \frac{y_t}{e_{t-m}} + (1-\alpha)(n_{t-1} + \delta b_{t-1})$$

$$Tendencia \quad b_t = \beta(n_t - n_{t-1}) + (1-\beta)\delta b_{t-1}$$

$$Estacionalidad \quad e_t = \gamma \frac{y_t}{n_{t-1} + \delta b_{t-1}} + (1-\gamma) e_{t-m}$$

Holt - Winters

```
fit1 = ExponentialSmoothing(
    aust,
    seasonal='mul',
    seasonal_periods=12
).fit()
fit2 = ExponentialSmoothing(
    aust,
    seasonal='add',
    seasonal_periods=12
).fit()
```



ver repo: exponentialSmoothing.py

Ref.: https://otexts.com/fpp2/expsmooth.html

Predicciones

Predicciones

Uno de los principales motivos de entrenar un modelo es para poder hacer predicciones acerca de los valores que va a tomar la serie de tiempo.

Es sumamente importante también conocer la precisión de estas estimaciones.

⚠ Para la mayor parte de lo que veamos esta clase vamos a suponer que conocemos perfectamente el modelo verdadero. Si bien es una suposición que en la práctica no se cumple, el uso de parámetros estimados cuando se cuenta con una cantidad grande de muestras no modifica significativamente los resultados presentados.

Objetivo

Dadas la historia disponible en un instante t, $Y_1, \ldots, Y_t$ queremos predecir el valor deY_{t+k} que va a ocurrir dentro de k instantes de tiempo.

De la materia Probabilidad y Estadística, sabemos que el mejor predictor de Y_{t+k} basado en las muestras $Y_1, \ldots, Y_t$ es la esperanza condicional

$$\hat{Y}_t(\ell) = \mathbb{E}[Y_{t+\ell}|Y_1, \ldots, Y_t]$$

Predicción con tendencias determinísticas

Consideremos el caso visto en la clase 3 donde Yt podía modelarse como

$$Y_t = X_t + \mu_t$$

donde X_t es una serie de tiempo de media nula y μ_t es la tendencia determinística.

Si además X_t tiene las propiedades de ruido blanco de varianza C_0^2, ocurre que X_t, X_l son independientes $\forall t \neq l$ y

$$\hat{Y}_{t+\ell} = \mathbb{E}[Y_{t+\ell}|Y_1, \ldots, Y_t] = \mathbb{E}[\mu_{t+\ell} + X_{t+\ell}|Y_1, \ldots, Y_t] = \mu_{t+\ell}$$

El error de predicción en este caso resulta

$$e_t(\ell) = Y_{t+\ell} - \hat{Y}_t(\ell) = X_{t+\ell}$$

con

$$\mathbb{E}[e_t(\ell)] = 0 \text{ y } var(e_t(\ell)) = C_0$$

Predicción de ARIMA - ejemplo AR(1)

Consideremos un AR(1) con media distinta de cero:

$$Y_t - \mu = a_1 (Y_{t-1} - \mu) + e_t$$

Luego, la predicción a un paso resulta

$$\hat{Y}_t(1) - \mu = \mathbb{E}[Y_{t+1} | Y_1, \ldots, Y_t] = \mathbb{E}[a_1 (Y_t - \mu) + e_t | Y_1, \ldots, Y_t] = a_1 (Y_t - \mu)$$

Podemos generalizar la predicción a ℓ pasos como

$$\hat{Y}_t(\ell) = \mu + a_1^{\ell} (Y_t - \mu)$$

Observar que si $|a_1| < 1$ y ℓ es grande $\mathbb{E}[\hat{Y}_t(\ell)] \approx \mu$.

Predicción de ARIMA - ejemplo AR(1)

Siguiendo la misma lógica de antes,

$$\boxed{e_t(1)} = Y_{t+1} - \hat{Y}_t(1) = Y_{t+1} - (a_1(Y_t - \mu) + \mu)$$
$$= a_1(Y_t - \mu) + \mu + e_{t+1} - (a_1(Y_1 - \mu) + \mu) = \boxed{e_{t+1}}$$

De donde obtenemos que

$$\mathbb{E}[e_t(1)] = 0 \text{ y } var(e_t(1)) = C_0$$

Para el caso general:

$$e_t(\ell) = Y_{t+\ell} - \hat{Y}_t(\ell) = Y_{t+\ell} - (a_1^{\ell}(Y_t - \mu) + \mu)$$
$$= e_{t+\ell} + a_1 e_{t+\ell-1} + \ldots + a_1^{\ell-1} e_t$$
$$= e_{t+\ell} + \psi_1 e_{t+\ell-1} + \ldots + \psi_{\ell-1} e_{t+1}$$

$$\mathbb{E}[e_t(\ell)] = 0 \text{ y } var(e_t(\ell)) = (1 + \psi_1^{2} + \ldots + \psi_{\ell-1})\sigma_e^2$$
$$= \sigma_e^2 \left(\frac{1 - a_1^{2\ell}}{1 - a_1^2} \right)$$

Es insesgado y la var del error crece con la cantidad de pasos .

Predicción de ARIMA - ejemplo MA(1)

Consideremos ahora un MA con media no nula: $Y_t = \mu + e_t - b_1 e_{t-1}$

En este caso resulta que la predicción a un paso se puede aproximar como

$$\hat{Y}_t(1) = \mathbb{E}[\mu + e_{t+1} - b_1 e_t | Y_1, \ldots, Y_t] = \mu - b_1 \mathbb{E}[e_t | Y_1, \ldots, Y_t] \approx \mu - b_1 e_t$$

Nuevamente, podemos obtener una expresión más general

$$\hat{Y}_t(\ell) = \mathbb{E}[\mu + e_{t+\ell} - b_1 e_{t+\ell-1} | Y_1, \ldots, Y_t] = \mu \quad \ell > 1$$

Para el caso de $\ell = 1$, tenemos que el error resulta

$$e_t(1) = e_{t+1}$$

Predicción de ARIMA - ejemplo Random Walk

Consideremos el caso del caminante aleatorio con deriva (*drift*)

$$Y_t = Y_{t-1} + \theta - e_t$$

donde la predicción a un paso resulta

$$\hat{Y}_t(1) = \mathbb{E}[Y_t + \theta - e_{t+1} | Y_1, \ldots, Y_t] = Y_t + \theta$$

Similar al caso AR(1) la predicción a ℓ pasos se puede obtener a partir de una expresión recursiva como

$$\hat{Y}_t(\ell) = Y_t + \theta\ell, \quad \ell \geq 1$$

Observar que si $\theta \neq 0$ la predicción no converge a un valor

Predicción de ARIMA - ejemplo Random Walk

El error para este ejemplo del caminante aleatorio resulta

$$
\begin{aligned}
e_t(\ell) = Y_{t+1} - \hat{Y}_t(\ell) &= (Y_t + e_{t+1} + \ldots + e_{t+\ell} + \ell\theta) - (Y_t + \theta\ell) \\
&= e_{t+1} + \ldots + e_{t+\ell}
\end{aligned}
$$

Se obtiene entonces que

$$\mathbb{E}[e_t(\ell)] = 0 \text{ y } var(e_t(\ell)) = \ell\sigma_e^2$$

El error de predicción crece sin límites a medida que aumentamos el paso de la predicción

Predicción de ARIMA - ejemplo ARMA(p,q)

Consideremos ahora un modelo ARMA(p,q)

$$Y_t = a_1 Y_{t-1} + \ldots + a_p Y_{t-p} + e_t - b_1 e_{t-1} + \ldots + b_q e_{t-q}$$

En este caso, se puede ver que

$$\begin{aligned}\hat{Y}_t(\ell) =& \mathbb{E}[Y_{t+\ell}|Y_1,\ldots Y_t] = a_1 \hat{Y}_t(\ell-1) + \ldots + a_p \hat{Y}_t(p) + \theta_0 - \\ & b_1 \mathbb{E}[e_{t-1}|Y_1,\ldots,Y_t] + \ldots + b_q \mathbb{E}[e_{t-q}|Y_1,\ldots,Y_t]\end{aligned}$$

donde $\theta_0 = \mu(1 - a_1 - \ldots - a_p)$.

Se observa que si $\ell > q$ los términos asociados al MA desaparecen y el comportamiento queda regido por la componente AR del modelo. Con lo cual tenemos que

$$\hat{Y}_t(\ell) = a_1(\hat{Y}_t(\ell-1) - \mu) + \ldots + a_p(\hat{Y}_t(p) - \mu) + \mu \quad \ell > q$$

Predicción de ARIMA - ejemplo ARMA(p,q)

Observar que la expresión hallada de $\hat{Y}_t(\ell)$ sigue la misma recursión de Yule-Walker que la función de autocorrelación del proceso, y por lo tanto las raíces de la ec. característica determinarán el comportamiento para l grande. En particular, se puede escribir a$\hat{Y}_t(\ell)$ como sumas de términos que decaen exponencialmente con l, y senoidales amortiguadas.

> Los términos de $\hat{Y}_t(\ell)$ decaen a cero cuando l cerece y la predicción se reduce a la media μ.

Modelo ARMA como proceso lineal truncado

Se puede demostrar (ver. apéndice G de Time Series Analysis) que todo modelo ARMA(p,q) se puede representar usando un proceso lineal truncado:

$$Y_{y+\ell} = C_t(\ell) + I_t(\ell), \quad \ell > 1$$

donde Ct(l) es una función de Y_t, Y_{t-1},... y

$$I_t(\ell) = e_{t+\ell} + \psi_1 e_{t+\ell-1} + \ldots + \psi_{\ell-1} e_{t+1}$$

Además, si l es lo suficientemente grande y el sistema es invertible $C_t(l)$ depende sólo de Y_t, Y_{t-1},... ,Y_1, y

$$\hat{Y}_t(\ell) = C_t(\ell) \text{ y } e_t(\ell) = Y_{t+\ell} - \hat{Y}_t(\ell) = I_t(\ell)$$

$$\mathbb{E}[e_t(\ell)] = 0$$

$$var(e_t(\ell) = (1 + \psi_1 + \ldots + \psi_{\ell-1})\sigma_e^2$$

$$var(e_t(\ell)) \approx C_0 \text{ para } \ell \text{ grande}$$

Predicción de ARIMA - ARIMA(p,d,q)

Se puede ver que un modelo ARIMA(p,d,q) se puede reescribir como un modelo ARMA(p+d,q) no estacionario:

$$Y_t = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \ldots + \varphi_{t-p-d} + e_t - b_1(e_t - 1) - \ldots b_q e_{t-q}$$

y las predicciones pueden cacularse igual que para el caso ARMA(p+d,q) y

$$e_t(\ell) = e_{t+\ell} + \psi_1 e_{t-1} + \ldots + \psi_{\ell-1} e_{t+\ell-1}$$

Nuevamente es válido que

$$\mathbb{E}[e_t(\ell)] = 0 \text{ y } var(e_t(\ell)) = \sigma_e^2 \sum_{i=1}^{\ell-1} \psi_i^2$$

La diferencia es que al ser un proceso no estacionario los coeficientes no convergen a cero a medida que aumenta i, y la varianza aumenta sin cota a medida que l aumenta.

Límites de la predicción

Como siempre, queremos saber la bondad de nuestra estimación. Hasta ahora analizamos la media y la varianza del error para cada instante predicho.

El objetivo es poder brindar un intervalo de confianza alrededor del valor predicho para cada valor de l.

Si las innovaciones (*et*) siguen una distribución gaussiana, luego el error *et*(l) también va a seguir una distribución gaussiana, de media y varianza ya calculadas. Luego,

$$\mathbb{P}\left(z_{\alpha/2} \leq \frac{Y_{y+\ell} - \hat{Y}_t(\ell)}{\sqrt{var(e_t(\ell))}} \leq z_{1-\alpha_2}\right) = 1 - \alpha$$

resultando que

$$Y_{t+\ell} \in [\hat{Y}_t(\ell) - z_{1-\alpha/2}\sqrt{var(e_t(\ell))}, \hat{Y}_t(\ell) + z_{1-\alpha/2}\sqrt{var(e_t(\ell))}$$

Actualizando las predicciones ARIMA

¿Cómo actualizamos las predicciones realizadas una vez que nos llega una nueva muestra de la serie?

Supongamos que tengo la predicción $\hat{Y}_t(\ell+1)$, que es la predicción del instante *t*+l+1, basados en las observaciones hasta tiempo *t*. Supongamos que de pronto me llega información acerca del instante t+1. Luego, quiero actualizar mi estimación de Y_{t+l+1}: $\hat{Y}_{t+1}(\ell)$

$$\hat{Y}_{t+1}(\ell) = \hat{Y}_t(\ell+1) + \psi_\ell[Y_{t+1} - \hat{Y}_t(1)]$$

Observar que se parece mucho a la predicción de Kalman!

Predicción de series transformadas

Pongamos el ejemplo de la transformación logarítmica. En este caso tenemo Y_t la serie original y $W_t = \log(Y_t)$. Se puede demostrar que

$$\mathbb{E}[Y_{t+\ell}|Y_1, \ldots, Y_t] \geq e^{\mathbb{E}[W_{t+\ell}|Y_1,\ldots,Y_t]}$$

con lo cual el estimador naive $\hat{Y}_t(\ell) = e^{\hat{W}_t(\ell)}$ no es el de ECM. Si Wt tiene distribución normal, vale que predictor de ECM es

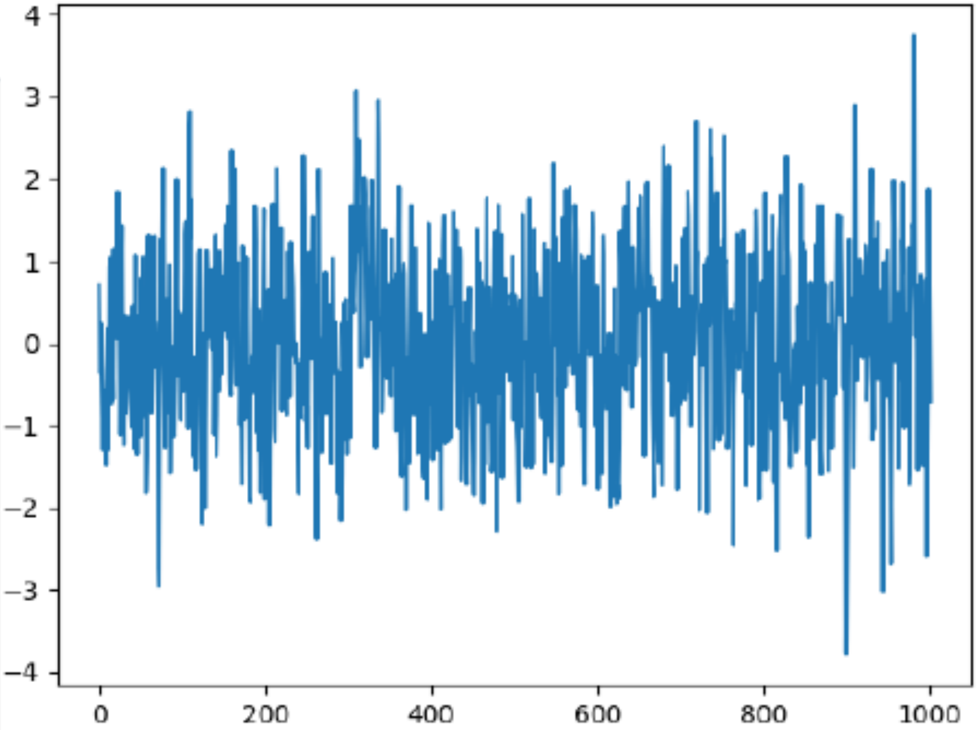
$$\hat{Y}_t(\ell) = e^{\hat{W}_t(\ell)+0.5var(e_t(\ell))}$$

Ejemplos

Modelo de Promedio Móvil MA(1)

$$y_t = b_1\epsilon_{t-1} + \epsilon_t$$

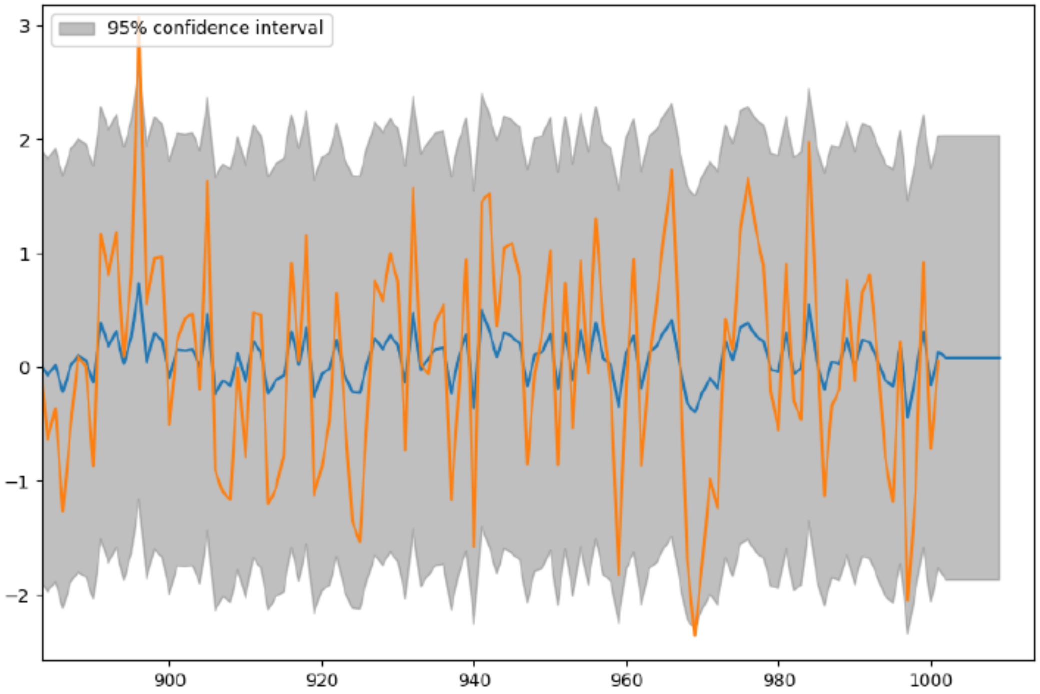
```python
# MA(1)
N =1000
b1 =0.25
e_t0 = np.random.normal(0,1)
e_t1 = np.random.normal(0,1)
y = np.append(e_t0,b1*e_t0+e_t1)  #

for i in range(N):
    e_t0 = e_t1
    e_t1 = np.random.normal(0,1)
    y = np.append(y,b1*e_t0+e_t1)
```



Modelo de Promedio Móvil MA(1)

$$y_t = b_1\epsilon_{t-1} + \epsilon_t$$

```
# Predict
ma1 = ARIMA(y, order=(0, 0, 1))
ma1_res = ma1.fit()
print(ma1_res.summary())

fig, ax = plt.subplots(figsize=(10, 8))
fig = plot_predict(ma1_res, start=1,end=
plt.plot(y)
legend = ax.legend(loc="upper left")
plt.show()
```

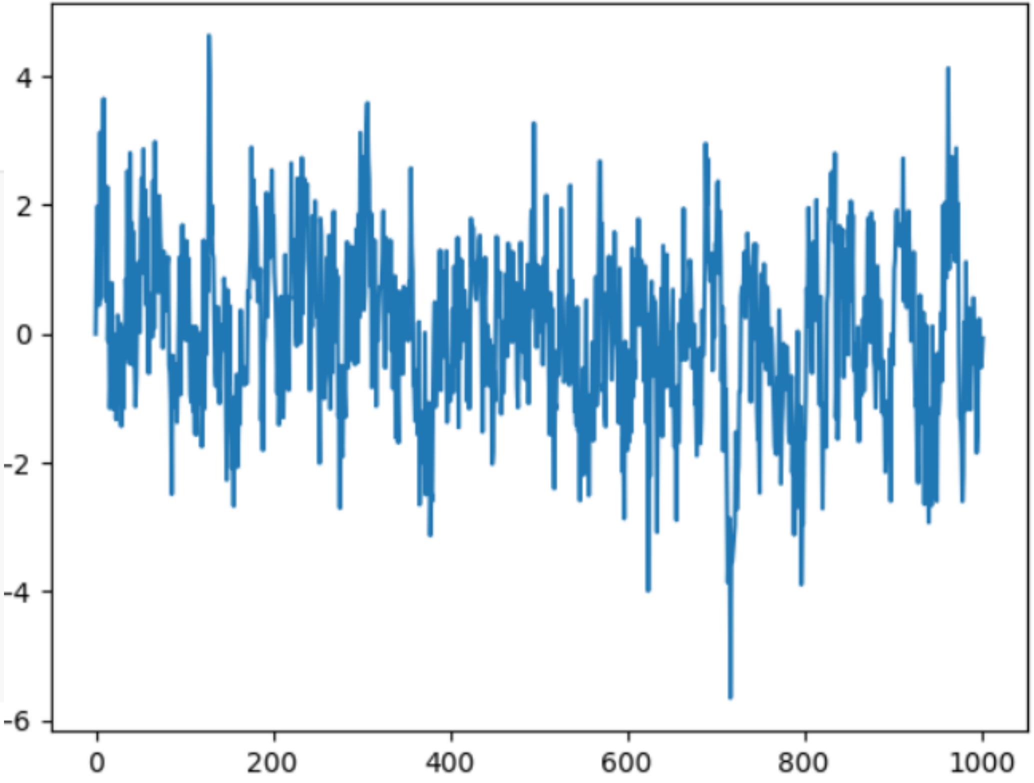


$$\gamma_0 = (1 + \theta_1^2 + \theta_2^2 + \ldots + \theta_q^2)\sigma_e^2$$

Modelo Autoregresivo AR(2)

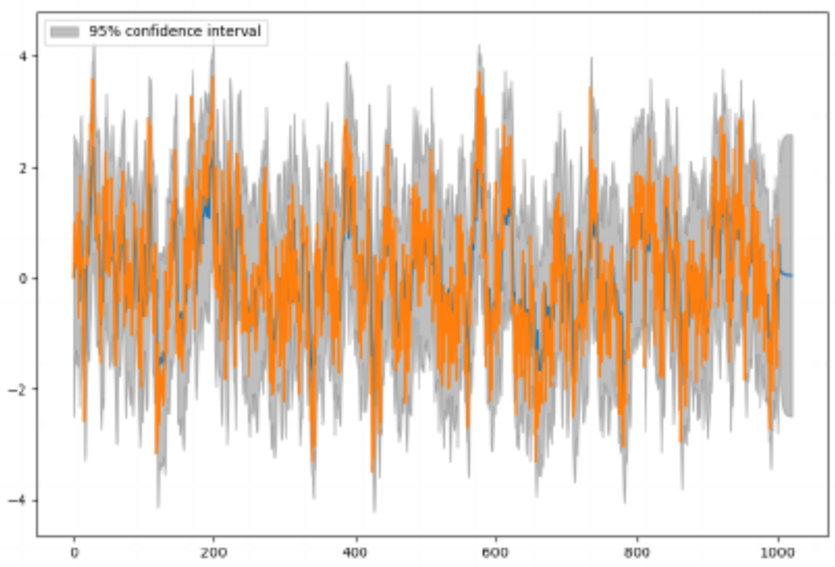
$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \epsilon_t$$

```
# AR(2)
N=1000
a1=0.4
a2=0.35
x=np.arange(2)
e_t=np.random.normal(0,1)
y=np.append(x,a1*x[1]+a2*x[0]+e_t)

for i in range(N):
    e_t=np.random.normal(0,1)
    y=np.append(y,a1*y[-1]+a2*y[-2]+e_t)
```



Modelo Autoregresivo AR(2)

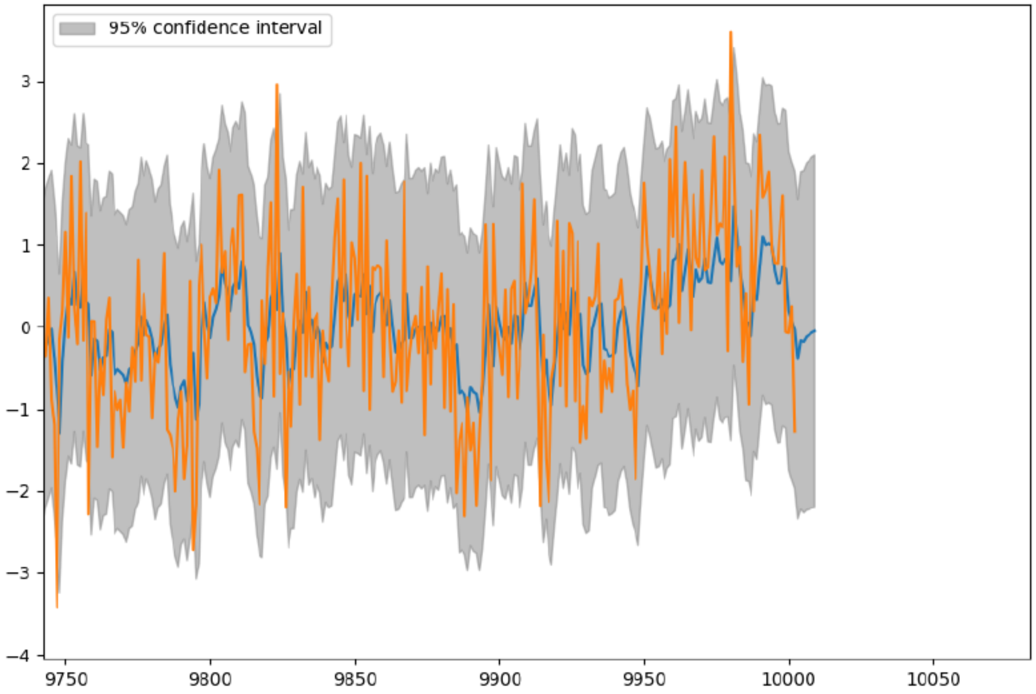
```
# Predict
l=20
ar2 = ARIMA(y, order=(2, 0, 0))
ar2_res = ar2.fit()
print(ar2_res.summary())

fig, ax = plt.subplots(figsize=(10, 8))
fig = plot_predict(ar2_res, start=0,end=N+l, ax=ax)
plt.plot(y)
legend = ax.legend(loc="upper left")
plt.show()
```



→ ver resultados del modelo

Modelo ARMA(2,1)

```
# Predict
y_arma = ARIMA(y, order=(2, 0, 1))
y_arma_res = y_arma.fit()
print(y_arma_res.summary())

fig, ax = plt.subplots(figsize=(10, 8))
fig = plot_predict(y_arma_res, start=1,end=N
plt.plot(y)
```



Modelo ARIMA(2,1,1)

```
y=np.append(y,a1*y[-1]+a2*y[-2]+ b1*e_0 + e_t)
```

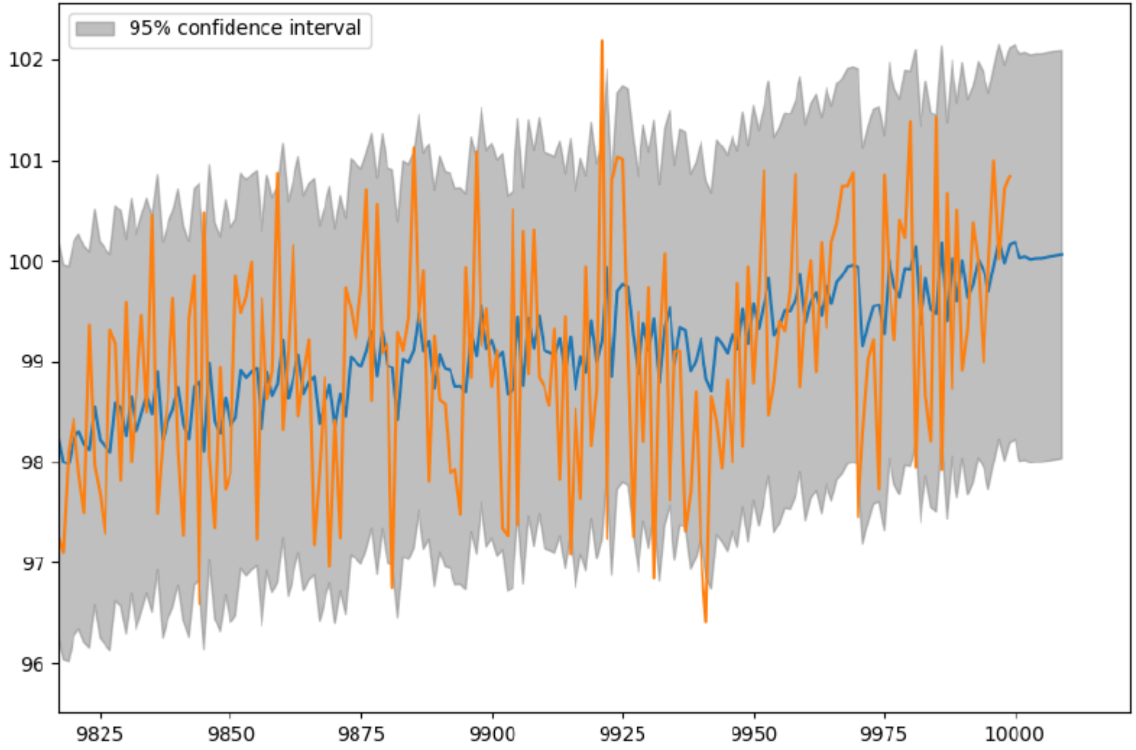
```
t = np.arange(int(N*0.01), step=0.01)
y=y[2:N+2]

y = y+t
```

```
# Predict
y_arma = ARIMA(y, order=(2, 1, 1), trend='t')
y_arma_res = y_arma.fit()
print(y_arma_res.summary())

fig, ax = plt.subplots(figsize=(10, 8))
fig = plot_predict(y_arma_res, start=1,end=N+10, ax=ax)
```

Modelo ARIMA(2,1,1)

```
                                SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                10000
Model:                 ARIMA(2, 1, 1)   Log Likelihood              -14245.820
Date:                jue, 18 nov 2021   AIC                          28501.641
Time:                        19:40:41   BIC                          28537.692
Sample:                             0   HQIC                         28513.844
                              - 10000
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.0100   9.94e-06   1006.340      0.000       0.010       0.010
ar.L1          0.0335      0.010      3.470      0.001       0.015       0.052
ar.L2          0.2474      0.010     25.610      0.000       0.228       0.266
ma.L1         -0.9996      0.001  -1875.605      0.000      -1.001      -0.999
sigma2         1.0100      0.014     70.300      0.000       0.982       1.038
===================================================================================
Ljung-Box (L1) (Q):                   4.66   Jarque-Bera (JB):                 0.24
Prob(Q):                              0.03   Prob(JB):                         0.89
Heteroskedasticity (H):               0.98   Skew:                             0.00
Prob(H) (two-sided):                  0.61   Kurtosis:                         2.98
===================================================================================
```

Modelo ARIMA(2,1,1)




```
#ARIMA

N=10000
a1=0.35
a2=0.25
b1 =-0.35
```

```
t = np.arange(int(N*0.01), step=0.01)
```

```
==========================
                      coef
--------------------------
x1                  0.0100
ar.L1               0.0335
ar.L2               0.2474
ma.L1              -0.9996
sigma2              1.0100
==========================
```

Prediction of time series

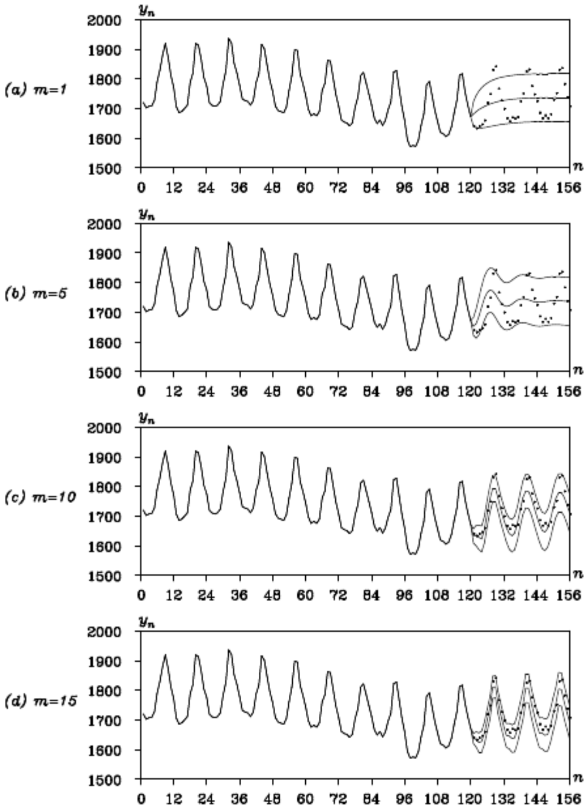


Figure 9.2 *Increasing horizon predictive distributions (bold line: mean, thin line: ± (standard deviation) and ○: observed value). Orders of the AR models are 1, 5, 10 and 15, respectively.*

Interpolation of missing data

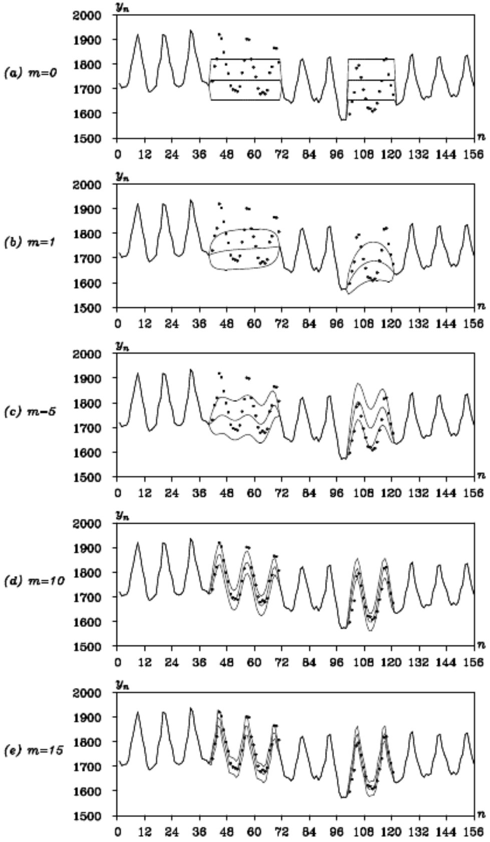


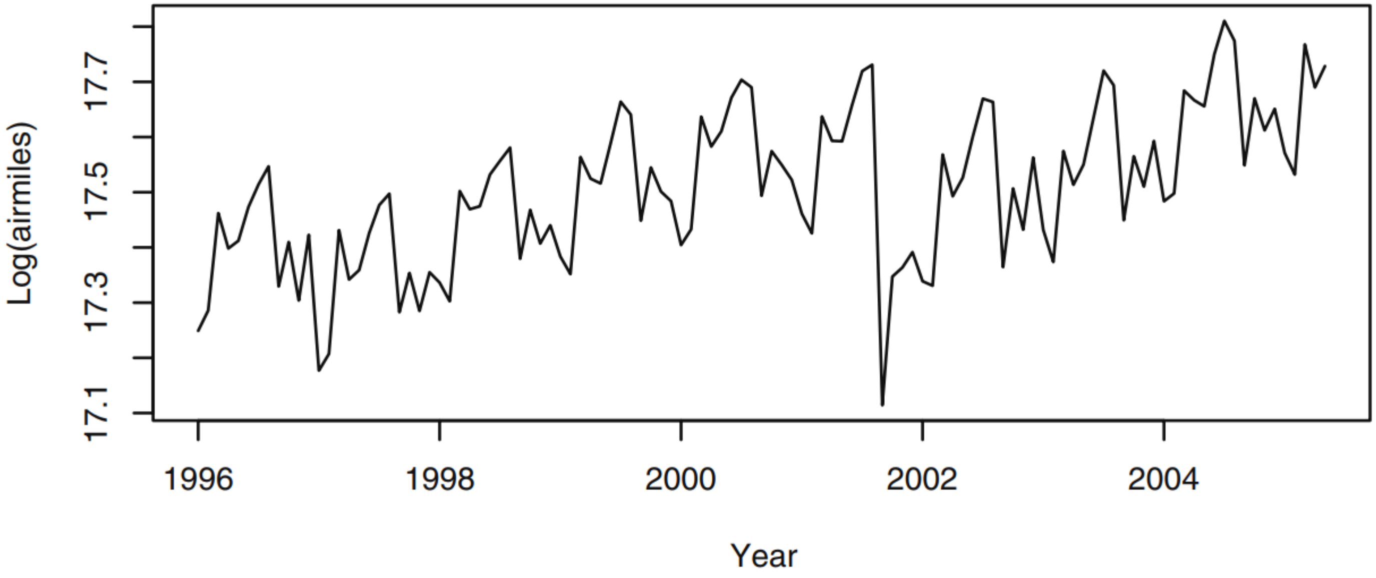
Figure 9.3 *Interpolation of missing values (bold line: mean, thin line: ± (standard deviation) and ○: observed value). Orders of the AR models are 0, 1, 5, 10 and 15.*

Ref.:*"Introduction to Time Series Modeling with applications in R", Genshiro Kitagawa, p.143-150*

Análisis de intervenciones

Análisis de intervenciones

Exhibit 11.1 Monthly U.S. Airline Miles: January 1996 through May 2005



Análisis de intervenciones

Consideremos el caso de una única intervención:

$$Y_t = \boxed{m_t} + \boxed{N_t}$$

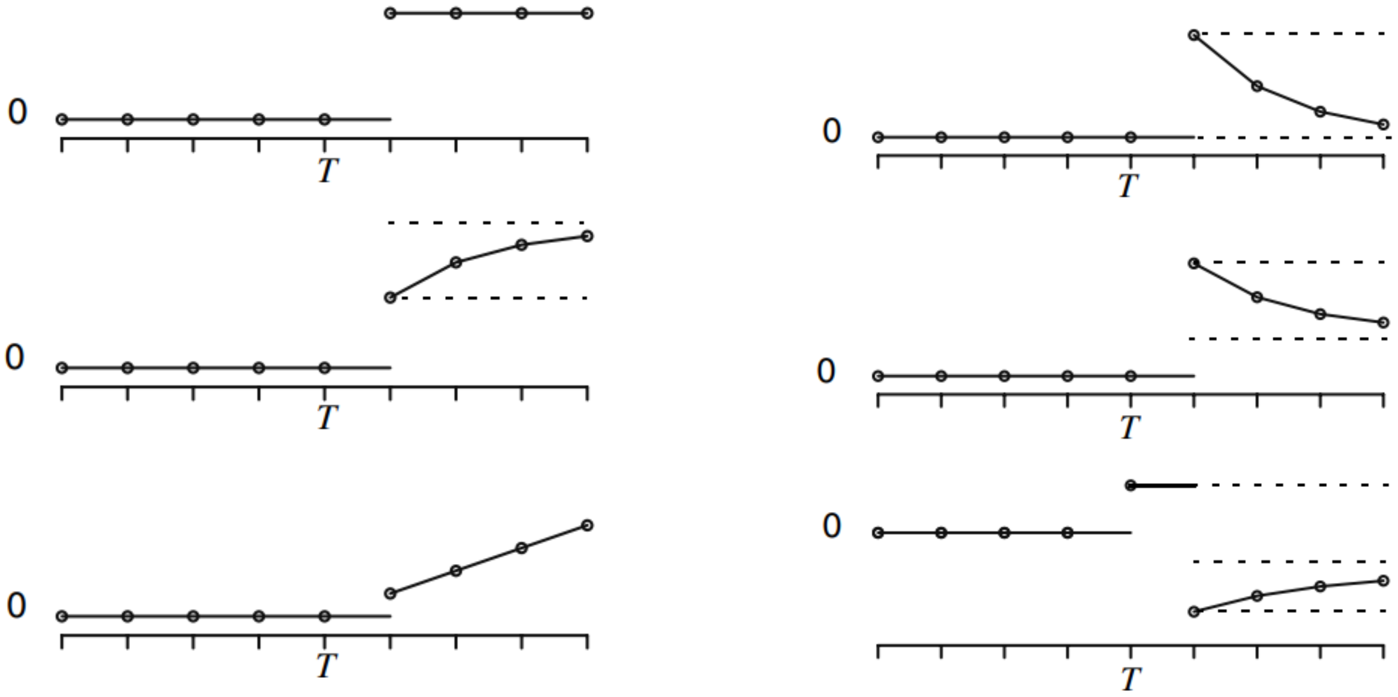
Intervención:
cambio en la media

Proceso natural:
serie de tiempo que puede ser modelada como un (S)ARIMA

Si la intervención ocurre en un instante T, se asume que $m_t = 0 \text{ para } t < T$ y a $\{Y_t\}_{t<T}$ se la conoce como *datos preintervención* y puede ser usada para modelar N_t.

La estimación de los parámetros de m_t se puede realizar por MV o el enfoque de modelo de estados ya presentados.

Análisis de intervenciones



Ejemplo

Ver notebook: intervention_analysis.ipynb

Análisis de outliers

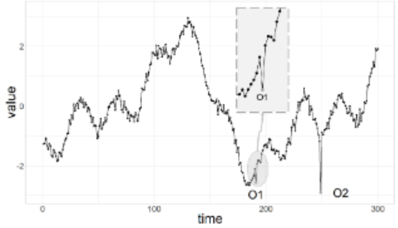
Tipos de outliers

Significado del outlier

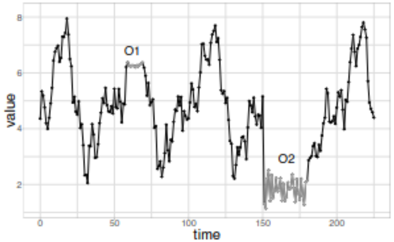
- Dato no deseado — objetivo — Limpieza del dato
- Evento de interés — objetivo — Análisis del outlier

Tipo de outlier

- Puntual (Point)

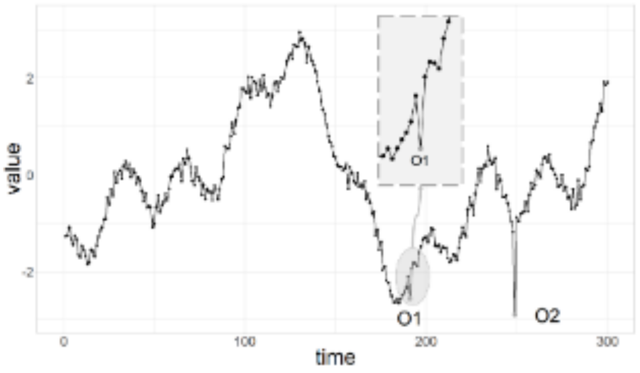


- Subsecuencia *(Subsequence)*

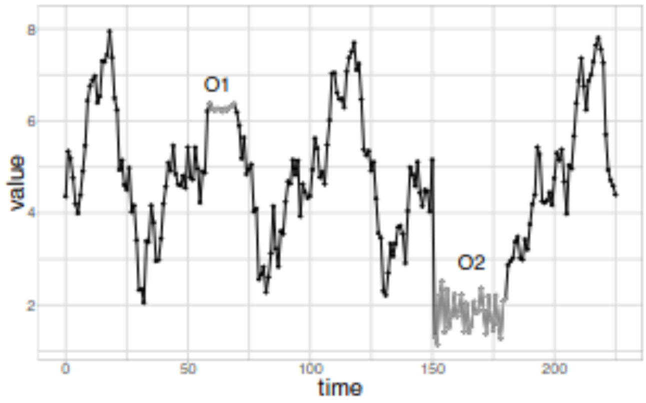


Tipos puntuales

Outliers puntuales: Dato que se comporta de manera inusual en un instante de tiempo específico, comparado con la serie total, o sus puntos vecinos. Son los más comunes.

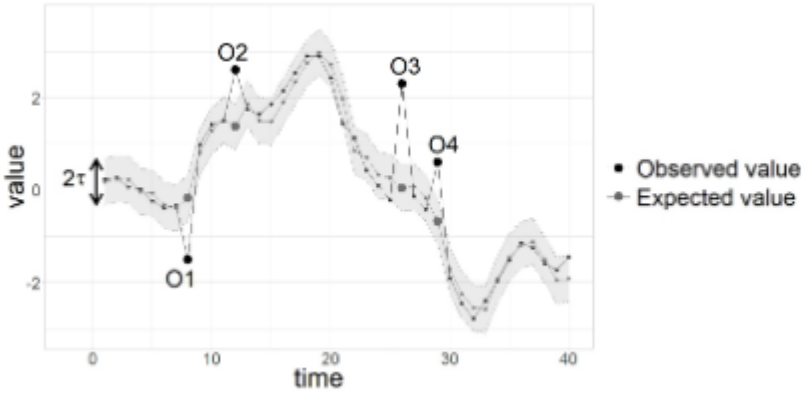


Outliers de subsecuencia: corresponde a puntos que en conjunto se comportan de forma peculiar respecto del resto de la serie, aunque cada observación de forma individual no resulta necesariamente un outlier.



Análisis de Outliers puntuales:

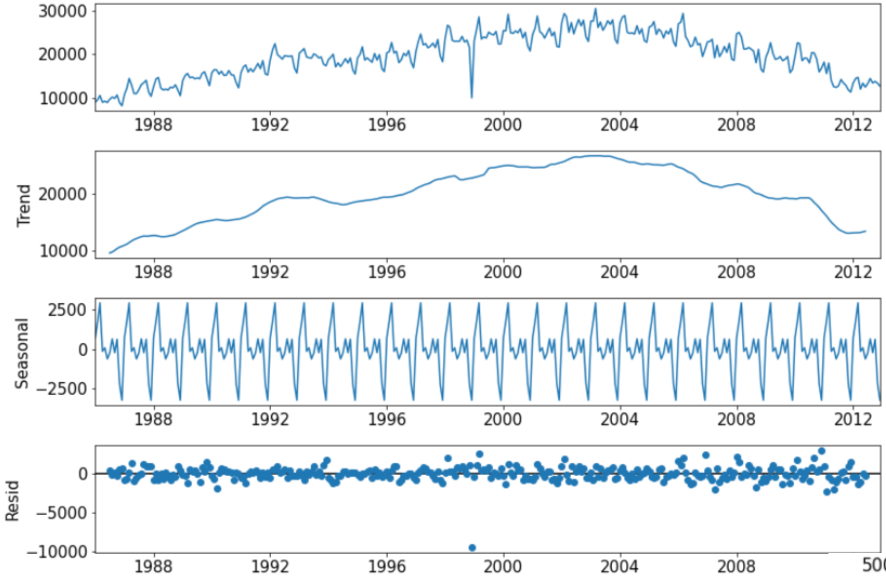
Podemos definir un outlier como un punto que se desvía significativamente de su valor esperado, es decir $|x_t - \hat{x}_t| > \tau$



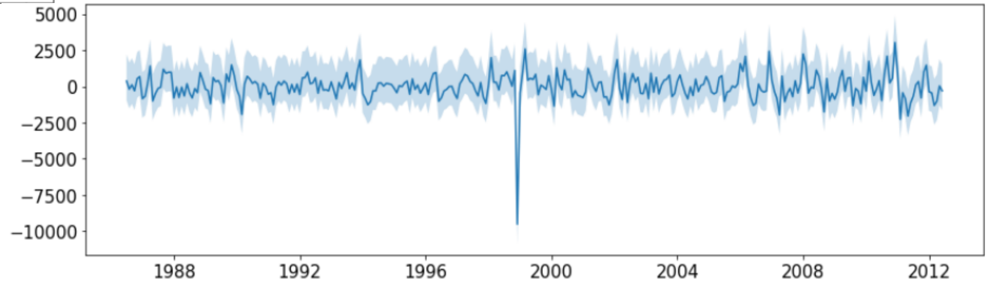
Hay distintas formas de determinar ese valor esperado y el umbral.

- Modelos constantes a tramos, donde se toman estadísticas como mediana o MAD (mean absolute deviation)
- Identificación de puntos poco probables bajo un cierto modelo o distribución
- Analizando residuos (ej. usando la descomposición STL, o directamente analizando residuos del ARIMA)
- Etc.

Detección de outliers usando la descomposición STL




```python
fig = plt.figure(figsize=(14,4))
q975 = residuals.quantile(.975)
q025 = residuals.quantile(.025)
plt.plot(residuals)
plt.fill_between(x=residuals.index,
                 y1=residuals + q025,
                 y2=residuals + q975,
                 alpha=0.25
                 )
```



Cómo tratar los outliers

- Usar la media en una ventana para suavizar el outlier
- Modelar el dato basándome en el problema de estudio
- Quitar el punto (mucho cuidado!)

Conclusiones

- Los modelos **AR, MA, ARMA** se basan en procesos estacionarios y están bien estudiados.
- Si un proceso no estacionario pueda ser **diferenciado** ‘i’ veces y volverse estacionario, entonces podemos usar la extensión del modelo **ARIMA**
- Para procesos con **tendencia constante o lineal**, el framework funciona normalmente bien con **ARIMA.fit()**
- Con la misma idea, procesos a los que se les puede extraer una tendencia determinística conviene tratarlos con modelos conocidos.
- Vimos que **Cuadrados Mínimos** puede ser útil para ajustar tendencias determinísticas y obtener los coeficientes del modelo
- La extensión **SARIMA** usa estos métodos con la misma idea de extraer componentes que se pueden modelar por descomposición
- Los **coeficientes** son **todo** a la hora de modelar para predecir. Una buena predicción se da cuando el modelo es el adecuado y los coeficientes están bien ajustados.