Series de tiempo univariadas - Presentación 18

Mauricio Alejandro Mazo Lopera

Universidad Nacional de Colombia
Facultad de Ciencias
Escuela de Estadística
Medellín

UNIVERSIDAD NACIONAL DE COLOMBIA

Transformaciones estabilizadoras de varianza:

La mayoría de los modelos de predicción asumen la la variación de la serie permanece constante a lo largo del tiempo. Sin embargo, este supuesto suele no cumplirse y por tanto, es necesario aplicar una transformación que busque estabilizar la varinza. Dos transformaciones usualmente utilizadas son:

- **Logaritmo natural:** En este caso, aplicamos a la serie $\{X_t\}$ el logaritmo natural, es decir, la serie a modelar sería $X_t^* = \ln(X_t)$.
- **Transformación Box-Cox:** Está basada en la siguiente fórmula

$$X_t^* = \begin{cases} \frac{X_t^\lambda - 1}{\lambda}, & \text{si } \lambda \neq 0 \\ \ln(X_t), & \text{si } \lambda = 0 \end{cases}$$

Veamos un ejemplo típico relacionado con este tipo de situaciones:

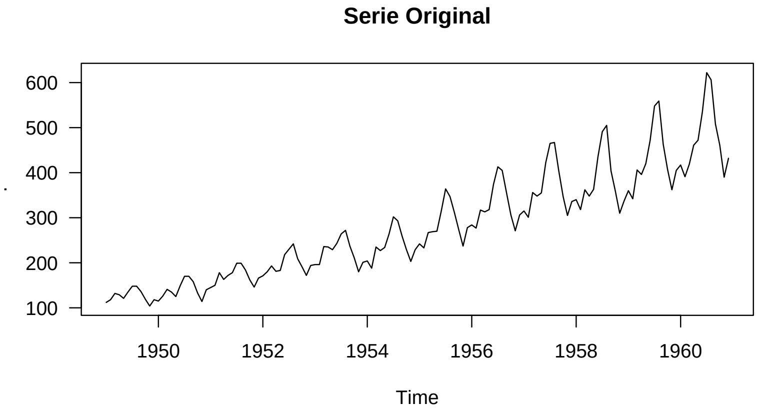
```
require(forecast)
require(magrittr)
require(tidyverse)
pas_lambda <- BoxCox.lambda(AirPassengers)
pas_lambda
```

```
## [1] -0.2947156
```

```
pasa_transform <- BoxCox(AirPassengers,
                         lambda=pas_lambda)
```

Con esta transformación pasamos de:

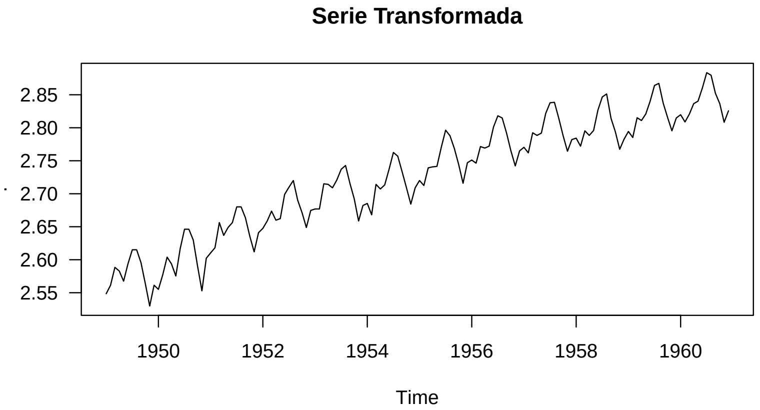
```
AirPassengers %>% plot(main="Serie Original",
                       las=1)
```



Transformaciones estabilizadoras de varianza: Ejemplo 1

A la serie transformada:

```
pasa_transform %>% plot(main="Serie Transformada",
                        las=1)
```



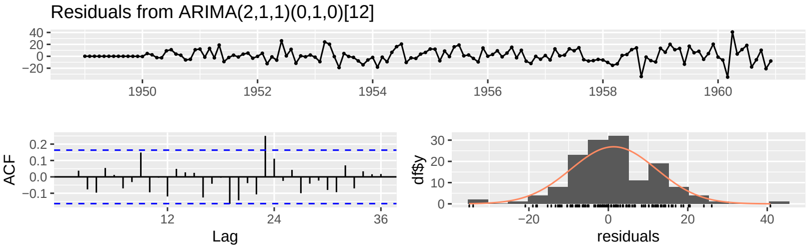
Transformaciones estabilizadoras de varianza: Ejemplo 1

Aplicamos la función **auto.arima** del paquete **forecast**:

```r
modelo1_sin <- auto.arima(AirPassengers)
modelo1_sin
```

```
## Series: AirPassengers
## ARIMA(2,1,1)(0,1,0)[12]
##
## Coefficients:
##          ar1     ar2      ma1
##       0.5960  0.2143  -0.9819
## s.e.  0.0888  0.0880   0.0292
##
## sigma^2 = 132.3:  log likelihood = -504.92
## AIC=1017.85   AICc=1018.17   BIC=1029.35
```

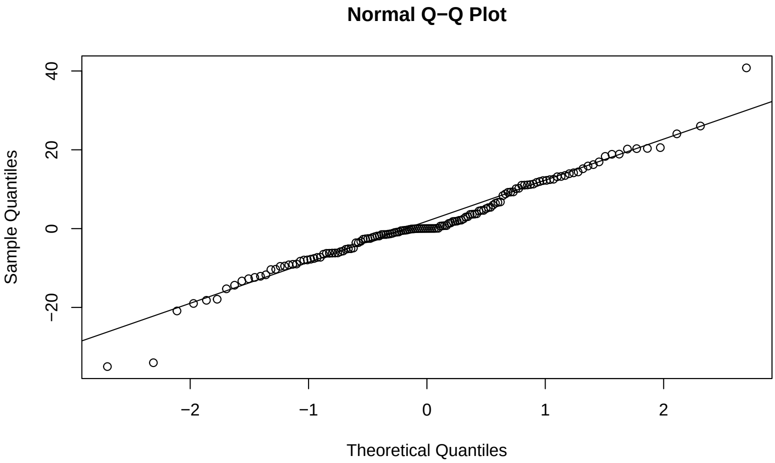
```
modelo1_sin %>% checkresiduals()
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)(0,1,0)[12]
## Q* = 37.784, df = 21, p-value = 0.01366
##
## Model df: 3.   Total lags used: 24
```

Transformaciones estabilizadoras de varianza: Ejemplo 1

```
modelo1_sin$residuals %>% qqnorm()
modelo1_sin$residuals %>% qqline()
```



Transformaciones estabilizadoras de varianza: Ejemplo 1

```
require(tseries)
modelo1_sin$residuals %>% shapiro.test()
```

```
##
##  Shapiro-Wilk normality test
##
## data:  .
## W = 0.97235, p-value = 0.005165
```

```
modelo1_sin$residuals %>% jarque.bera.test()
```

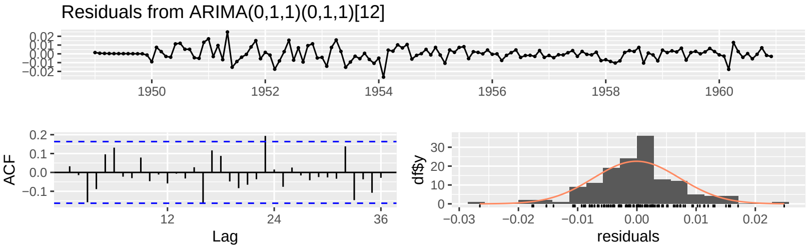
```
##
##  Jarque Bera Test
##
## data:  .
## X-squared = 15.131, df = 2, p-value = 0.0005181
```

Aplicamos la función **auto.arima** del paquete **forecast**:

```
modelo1_con <- auto.arima(pasa_transform)
modelo1_con
```

```
## Series: pasa_transform
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.4355  -0.5847
## s.e.   0.0908   0.0725
##
## sigma^2 = 5.855e-05:  log likelihood = 451.6
## AIC=-897.19   AICc=-897.01   BIC=-888.57
```

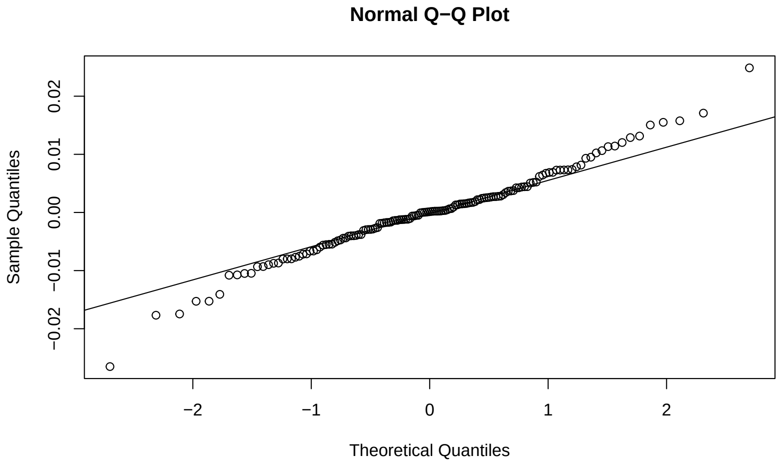
```
modelo1_con %>% checkresiduals()
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(0,1,1)[12]
## Q* = 28.635, df = 22, p-value = 0.1556
##
## Model df: 2.   Total lags used: 24
```

Transformaciones estabilizadoras de varianza: Ejemplo 1

```
modelo1_con$residuals %>% qqnorm()
modelo1_con$residuals %>% qqline()
```



Transformaciones estabilizadoras de varianza: Ejemplo 1

```
modelo1_con$residuals %>% shapiro.test()
```

```
##
##  Shapiro-Wilk normality test
##
## data:  .
## W = 0.98016, p-value = 0.03484
```

```
modelo1_con$residuals %>% jarque.bera.test()
```

```
##
##  Jarque Bera Test
##
## data:  .
## X-squared = 13.604, df = 2, p-value = 0.001111
```

Regresión lineal con errores ARIMA

Recuerden que un modelo de regresión lineal puede ser representado por:

$$X_t = \beta_0 + \beta_1 Z_{1t} + \beta_2 Z_{2t} + \cdots + \beta_p Z_{kt} + e_t$$

donde $Z_{1t}, \ldots, Z_{kt}$ son covariables medidas en el tiempo t utilizadas para explicar X_t, y además, se asume que e_t es un ruido blanco (comúnmente Gaussiano o Normal).

Regresión lineal con errores ARIMA

Recuerden que un modelo de regresión lineal puede ser representado por:

$$X_t = \beta_0 + \beta_1 Z_{1t} + \beta_2 Z_{2t} + \cdots + \beta_p Z_{kt} + e_t$$

donde $Z_{1t}, \ldots, Z_{kt}$ son covariables medidas en el tiempo t utilizadas para explicar X_t, y además, se asume que e_t es un ruido blanco (comúnmente Gaussiano o Normal). Sin embargo, este último supuesto suele NO cumplirse.

Una solución consiste en modelar los residuales por medio de un modelo $ARIMA(p, d, q)$ (o $SARIMA$), es decir:

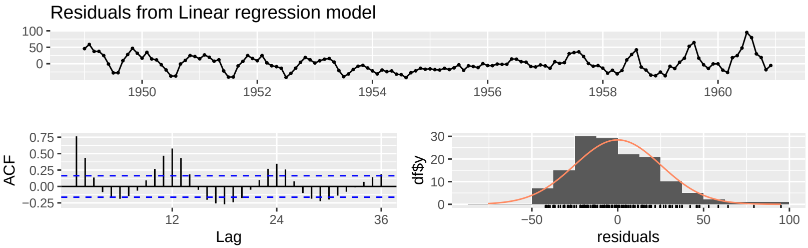
$$X_t = \beta_0 + \beta_1 Z_{1t} + \beta_2 Z_{2t} + \cdots + \beta_p Z_{kt} + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j w_{t-j} + w_t$$

donde w_t se asume como un ruido blanco (comúnmente Gaussiano o Normal).

Regresión lineal con errores ARIMA: Ejemplo 1

Dentro del paquete **forecast** existe una función llamada **tslm** que permite ajustar un modelo de regresión a una serie de tiempo (un objeto **ts**):

```
modelo21 <- tslm(AirPassengers~trend+season)
checkresiduals(modelo21)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up
##
```

Note que en el modelo anterior, **AirPassengers** es un objeto **ts** y por tanto, automáticamente la función **tslm** reconoce que este tiene una tendencia (trend) y un compoenente estacional (season).

En los gráficos del diagnóstico de los residuales se observa que no se cumple el supuesto de no autocorrelación. El resumen de dicho modelo se obtiene con el código:

```
modelo21 %>% summary()
```

```
##
## Call:
## tslm(formula = AirPassengers ~ trend + season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.121 -18.564  -3.268  15.189  95.085
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  63.50794    8.38856   7.571 5.88e-12 ***
## trend         2.66033    0.05297  50.225  < 2e-16 ***
## season2      -9.41033   10.74941  -0.875 0.382944
## season3      23.09601   10.74980   2.149 0.033513 *
## season4      17.35235   10.75046   1.614 0.108911
## season5      19.44202   10.75137   1.808 0.072849 .
## season6      56.61502   10.75254   5.265 5.58e-07 ***
## season7      93.62136   10.75398   8.706 1.17e-14 ***
## season8      90.71103   10.75567   8.434 5.32e-14 ***
## season9      39.38403   10.75763   3.661 0.000363 ***
## season10      0.89037   10.75985   0.083 0.934177
## season11    -35.51996   10.76232  -3.300 0.001244 **
## season12     -9.18029   10.76506  -0.853 0.395335
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.33 on 131 degrees of freedom
## Multiple R-squared:  0.9559, Adjusted R-squared:  0.9518
## F-statistic: 236.5 on 12 and 131 DF,  p-value: < 2.2e-16
```

Regresión lineal con errores ARIMA: Ejemplo 1

El intercepto, la tendencia y varios valores relacionados con algunos meses son significativos. Además, el R^2-Ajustado muestra que el modelo explica un 95.18 % de la variabilidad de la serie de tiempo. Sin embargo, el supuesto de no autocorrelación de los residuales definitivamente no se cumple.

Debido a lo anterior, definimos lo siguiente:

```r
require(TSstudio)
pasajeros <- ts_to_prophet(AirPassengers)
names(pasajeros) <- c("fecha", "valores")
pasajeros %>% head(3)
```

```
##        fecha valores
## 1 1949-01-01     112
## 2 1949-02-01     118
## 3 1949-03-01     132
```

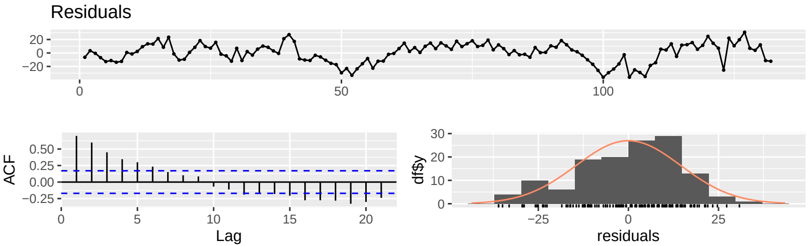
Regresión lineal con errores ARIMA: Ejemplo 1

Luego, creamos algunas variables que nos servirán para explicar la variabilidad de la serie:

```
require(tidyverse)
require(lubridate)

# Creamos una variable que representa la estacionalidad anual:
pasajeros$lag12 <- dplyr::lag(pasajeros$valores, n=12)

# Creamos la variable mes que representa el componente
# estacional:
pasajeros$mes <- pasajeros$fecha %>% month(label=TRUE) %>%
                 factor(ordered=FALSE)

# Cresmos una variable que denote el paso a lo largo del
# tiempo:
pasajeros$tendencia <- 1:nrow(pasajeros)
```

Ajustamos entonces un nuevo modelo considerando la función **lm** del R:

```
modelo22 <- lm(valores~tendencia+mes+lag12,data=pasajeros)
checkresiduals(modelo22)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up
##
## data:  Residuals
```

Regresión lineal con errores ARIMA: Ejemplo 1

Los residuales aún no se ajustan bien y presentan autocorrelación. Por este motivo, ajustamos un modelo $ARIMA(p, d, q)$ (o también $SARIMA(p, d, q) \times (P, D, Q)_s$ a los residuales:

```
matriz_disenio <- model.matrix(~mes+tendencia+lag12,
                                data=pasajeros)[,-1]
x_t <- ts(pasajeros$valores[13:nrow(pasajeros)],
          frequency = 12)
modelo23 <- auto.arima(x_t, xreg=matriz_disenio,
                       seasonal = TRUE,
                       stepwise = FALSE,
                       approximation = FALSE)
```

Note que en la matriz de diseño, dejamos afuera la primera columna (correspondiente a una variable dummy de unos) para que el modelo no se "confunda" con el intercepto.

Vemos el modelo ajustado:

```
modelo23
```

```
## Series: x_t
## Regression with ARIMA(2,0,0)(2,0,0)[12] errors
##
## Coefficients:
##          ar1     ar2     sar1     sar2  intercept   mesfeb   mesmar  mesabr
##       0.4676  0.3058  -0.4505  -0.1982    15.3395  -3.2534  -0.7725  1.6408
## s.e.  0.0842  0.0870   0.1045   0.1085     5.5906   2.0778   2.4096  2.5872
##       mesmay  mesjun   mesjul   mesago  messept  mesoct   mesnov   mesdic
##       3.6479  8.2835  14.7739  14.0767   4.9765  1.5624  -3.4306  -1.1302
## s.e.  2.7170  3.4527   4.4101   4.3006   3.0654  2.5210   2.5572   2.1281
##       tendencia   lag12
##          0.1873  0.9926
## s.e.     0.1187  0.0401
##
## sigma^2 = 102.3:  log likelihood = -484.79
## AIC=1007.58   AICc=1014.37   BIC=1062.35
```

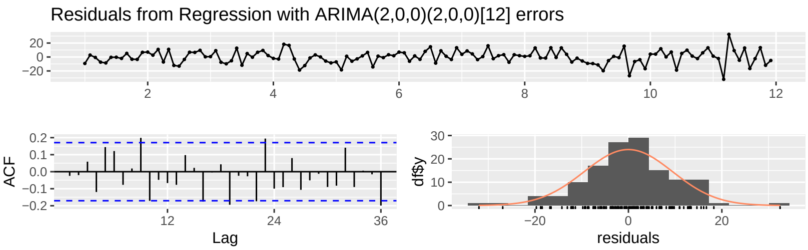
Vemos el modelo ajustado:

```
require(lmtest)
modelo23 %>% coeftest()
```

```
##
## z test of coefficients:
##
##            Estimate Std. Error z value  Pr(>|z|)
## ar1        0.467609   0.084233  5.5514 2.834e-08 ***
## ar2        0.305752   0.087033  3.5131  0.000443 ***
## sar1      -0.450527   0.104482 -4.3120 1.618e-05 ***
## sar2      -0.198169   0.108523 -1.8261  0.067842 .
## intercept 15.339514   5.590631  2.7438  0.006073 **
## mesfeb    -3.253368   2.077764 -1.5658  0.117395
## mesmar    -0.772511   2.409560 -0.3206  0.748512
## mesabr     1.640813   2.587166  0.6342  0.525942
## mesmay     3.647868   2.716979  1.3426  0.179395
## mesjun     8.283532   3.452668  2.3992  0.016432 *
## mesjul    14.773947   4.410100  3.3500  0.000808 ***
## mesago    14.076703   4.300579  3.2732  0.001063 **
## messept    4.976471   3.065380  1.6234  0.104495
## mesoct     1.562373   2.521001  0.6197  0.535427
## mesnov    -3.430616   2.557188 -1.3416  0.179739
## mesdic    -1.130204   2.128087 -0.5311  0.595357
## tendencia  0.187297   0.118728  1.5775  0.114673
## lag12      0.992596   0.040051 24.7835 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Vemos el diagnóstico del moelo:

```
checkresiduals(modelo23)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,0,0)(2,0,0
## Q* = 45.447, df = 6, p-value = 3.814e-08
##
```

Actividad de clase:

Considere las bases de datos del petróleo brent (con nombre **petroleo_brent_historico.csv**) y de la tasa representativa del mercado (**trm_historico.csv**) que se encuentran en la carpeta DATOS del Google Drive.

1. Lea ambas bases de datos.
2. Una las dos bases de datos en una sola BD.
3. Realice un gráfico donde aparezcan ambas series de tiempo.
4. Ajuste un modelo de regresión con los errores descritos por medio de un proceso ARIMA.
5. Realice un diagnóstico del modelo y saque conclusiones.