Series de tiempo univariadas - Presentación 20

Mauricio Alejandro Mazo Lopera

Universidad Nacional de Colombia
Facultad de Ciencias
Escuela de Estadística
Medellín

Ejemplo: TRM modelada a través de Brent y ARIMA:

Recuerden que en la presentación pasada tratamos de realizar un proceso de Backtesting para el modelo que buscaba explicar la TRM a través del precio internacional del petróleo Brent y un ARIMA:

```r
library(tidyverse)
library(magrittr)
library(forecast)
library(janitor)
library(lubridate)
library(lmtest)

petroleo<-read.csv("../../DATOS/petroleo_brent_historico.csv",
                   encoding = "utf-8", dec = ",")
trm <- read.csv("../../DATOS/trm_historico.csv")
```

Ejemplo: TRM modelada a través de Brent y ARIMA:

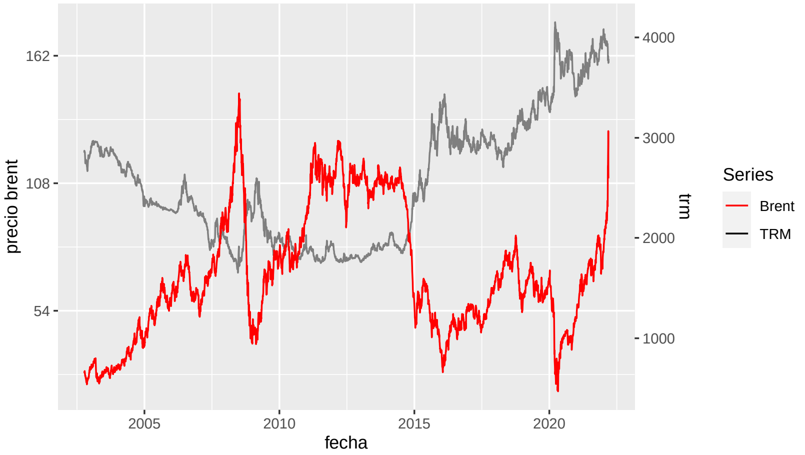
Organizamos las BD:

```
petroleo %<>% clean_names()
petroleo$i_fecha %<>% str_replace_all(.,"\\.","-") %>%
  as.Date(format = "%d-%m-%Y")
trm$VIGENCIADESDE %<>% as.Date(format = "%d/%m/%Y")
petroleo %<>% rename("fecha" = "i_fecha")
trm %<>% rename("fecha" = "VIGENCIADESDE")

bd_juntas <- merge(petroleo, trm, by = "fecha")

bd_juntas %<>% rename("brent" = "apertura",
                      "trm" = "VALOR")

bd_juntas %<>% arrange(-desc(fecha))
```

Graficamos:

```
bd2 = round(c(2,4,6)*sd(bd_juntas$brent), 0)

bd_juntas %>% ggplot(aes(x = fecha, y = trm/sd(trm))) +
  geom_line(aes(color="trm/sd(trm)")) +
  geom_line(aes(x = fecha, y = brent/sd(brent)),col="red") +
  scale_y_continuous(name = "precio brent", labels = bd2,
                     breaks = c(2,4,6),
                     sec.axis = sec_axis(~.*sd(bd_juntas$trm),
                                         name = "trm"))+
  scale_color_manual(name = "Series",
      values = c("Brent" = "red", "TRM" = "black"))
```

Ejemplo: TRM modelada a través de Brent y ARIMA:

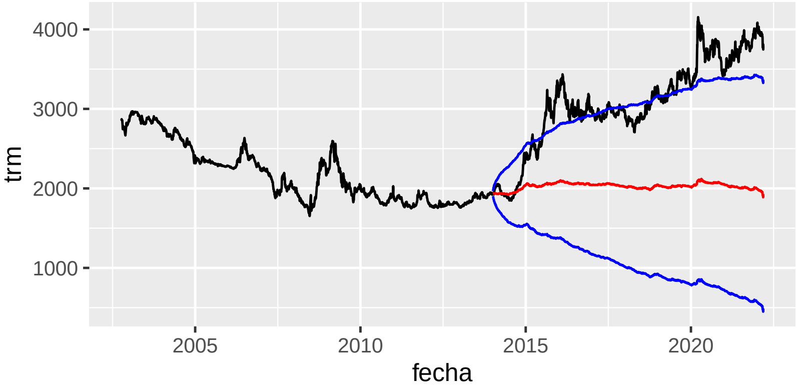


Ejemplo: TRM modelada a través de Brent y ARIMA:

```
bd_juntas$anio <- year(bd_juntas$fecha)
train <- bd_juntas %>% filter(anio < 2014)
test <- bd_juntas %>% filter(anio >= 2014)
disenio3 <- model.matrix(~-1+train$brent)
colnames(disenio3) <- "brent"
modelo3 <- auto.arima(train$trm, xreg = disenio3,
                stepwise = F, approximation = F)
require(forecast)
disenio3_test <- model.matrix(~-1+test$brent)
colnames(disenio3_test) <- "brent"
fore3 <- forecast(modelo3, xreg = disenio3_test,
                  h = nrow(test))
test$pred <- fore3$mean
test$li95 <- fore3$lower[,2]
test$ls95 <- fore3$upper[,2]
```

Ejemplo: TRM modelada a través de Brent y ARIMA:

```
ggplot(bd_juntas, aes(x=fecha, y=trm))+ geom_line()+
  geom_line(data=test, aes(x=fecha, y=pred), col="red")+
  geom_line(data=test, aes(x=fecha, y=li95), col="blue")+
  geom_line(data=test, aes(x=fecha, y=ls95), col="blue")
```



TAREA: Repetir el proceso con otros datos **train** y **test**.

Anteriormente vimos que es posible suavizar una serie de tiempo $X_1, X_2, \ldots, X_n$ por medio de promedios móviles bilaterales de "tamaño" m (para m impar):

$$X_t^* = \frac{X_{t-(m-1)/2} + \cdots + X_t + \cdots + X_{t+(m-1)/2}}{m}$$

El problema de este método es que para hacer predicciones a futuro se hace necesario tener la mitad de las observaciones por encima de t, lo cual técnicamente no es posible cuando tenemos, por ejemplo, el último valor de la serie (X_n) y queremos conocer cuál es el valor del pronóstico siguiente (X_{n+1}).

Pronósticos con modelos de suavizamiento exponencial

Una solución a este problema consiste en tomar los promedios unilaterales a izquierda de tamaño m, es decir,

$$X_t^* = \frac{X_{t-(m-1)} + \cdots + X_{t-1} + X_t}{m}$$

En este caso, el pronóstico de la serie 1 valor a futuro, se podría definir como:

$$\widehat{X}_{n+1} = \frac{X_{n-(m-1)} + \cdots + X_{n-1} + X_n}{m}$$

Siguiendo este mismo proceso, el pronóstico 2 valores a futuro sería:

$$\widehat{X}_{n+2} = \frac{X_{n-(m-2)} + \cdots + X_{n-1} + X_n + \widehat{X}_{n+1}}{m}$$

Así mismo, el pronóstico 3 valores a futuro sería:

$$\widehat{X}_{n+3} = \frac{X_{n-(m-3)} + \cdots + X_{n-1} + X_n + \widehat{X}_{n+1} + \widehat{X}_{n+2}}{m}$$

Así mismo, el pronóstico 3 valores a futuro sería:

$$\widehat{X}_{n+3} = \frac{X_{n-(m-3)} + \cdots + X_{n-1} + X_n + \widehat{X}_{n+1} + \widehat{X}_{n+2}}{m}$$

Se deduce entonces que el pronóstico h valores a futuro estaría dado por:

$$\widehat{X}_{n+h} = \frac{X_{n-(m-h)} + \cdots + X_{n-1} + X_n + \widehat{X}_{n+(h-2)} + \widehat{X}_{n+(h-1)}}{m}$$

con $n-(m-h) > 0$.

Así mismo, el pronóstico 3 valores a futuro sería:

$$\widehat{X}_{n+3} = \frac{X_{n-(m-3)} + \cdots + X_{n-1} + X_n + \widehat{X}_{n+1} + \widehat{X}_{n+2}}{m}$$

Se deduce entonces que el pronóstico h valores a futuro estaría dado por:

$$\widehat{X}_{n+h} = \frac{X_{n-(m-h)} + \cdots + X_{n-1} + X_n + \widehat{X}_{n+(h-2)} + \widehat{X}_{n+(h-1)}}{m}$$

con $n - (m - h) > 0$.

NOTE QUE: Cuando h es muy grande, los pronósticos más lejanos se ven muy influenciados por los pronósticos anteriores. Además, el tamaño de la ventana, m, puede ser seleccionado de tal forma que los pronósticos funcionen de la "mejor" forma utilizando, por ejemplo, backtesting.

La función ya está programada y podemos cargarla deste un archivo que tenemos debajo de esta presentación en Moodle con el nombre de **sma_funcion.r** y que contiene la función **sma_forecast**:

```
source("sma_funcion.r")
```

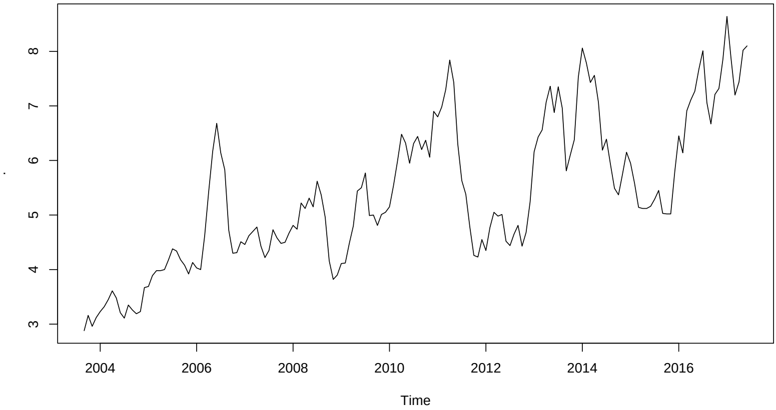
Esta función fue tomada del texto de **Rami Krispin - Hands-On Time Series Analysis With R_ Perform Time Series Analysis And Forecasting Using R-Packt Publishing (2019)**, páginas 289-290.

Veamos cómo funciona con una serie de tiempo denominada **salmon** y que se relaciona con los precios de exportación por kilogramo de salmon.

```
require(astsa) # Paquete que contiene la BD
require(TSstudio)
require(tidyverse)
require(magrittr)
salmon %>% ts_info()
```

```
##  The . series is a ts object with 1 variable and 166 observations
##  Frequency: 12
##  Start time: 2003 9
##  End time: 2017 6
```

```
salmon %>% plot()
```



Realizamos el pronóstico de los siguientes 24 meses ($h = 24$) utilizando un tamaño de ventana $m = 4$:

```
salmon_aux <- salmon %>% ts_to_prophet()
salmon_fore <- sma_forecast(salmon_aux, h=24, 4)
salmon_fore %>% head(3)
```

```
##         date test train yhat    y
## 1 2003-09-01   NA  2.88   NA 2.88
## 2 2003-10-01   NA  3.16   NA 3.16
## 3 2003-11-01   NA  2.96   NA 2.96
```

```
salmon_fore %>% tail(3)
```

```
##           date test train     yhat    y
## 164 2017-04-01 7.44    NA 5.138001 7.44
## 165 2017-05-01 8.02    NA 5.138000 8.02
## 166 2017-06-01 8.10    NA 5.138000 8.10
```

Pronósticos con modelos de suavizamiento exponencial

Evaluamos la precisión de las predicciones:

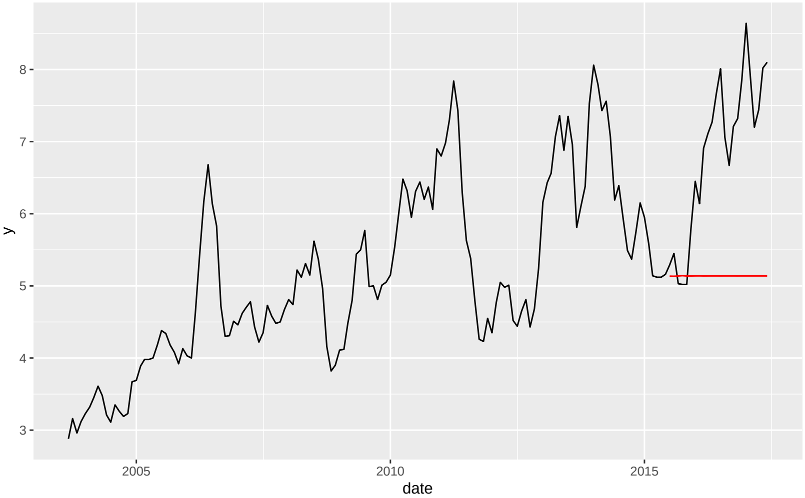
```
require(forecast)
accuracy(salmon_fore$test, salmon_fore$yhat)
```

```
##                 ME     RMSE      MAE       MPE     MAPE
## Test set -1.718917 2.025525 1.747711 -33.45533 34.01567
```

Graficamos:

```
salmon_fore  %>% ggplot(aes(x=date, y=y))+
  geom_line()+
  geom_line(aes(x=date, y=yhat), col="red")
```

Pronósticos con modelos de suavizamiento exponencial



A pesar de que el suavizamiento por medias móviles simples (Simple Moving Average, **SMA**) es bastante simple de aplicar y computacionalmente poco demandantes, tiene las siguientes limitaciones:

- Los pronósticos no son buenos para un horizonte muy amplio (h muy grande).
- Cuando se presenta estacionalidad, este método no es recomendable debido a que tiende a suavizar en exceso como vimos en el ejemplo anterior.

Este procedimiento consiste en realizar pronósticos con base en promedios ponderados no necesariamente iguales:

$$\widehat{X}_{n+h} = w_1 X_{n-(m-h)} + \cdots + w_m X_{n+(h-1)}$$

donde $w_1, w_2, \ldots, w_m$ son los pesos correcpondientes a cada una de las observaciones.

Este procedimiento consiste en realizar pronósticos con base en promedios ponderados no necesariamente iguales:

$$\widehat{X}_{n+h} = w_1 X_{n-(m-h)} + \cdots + w_m X_{n+(h-1)}$$

donde $w_1, w_2, \ldots, w_m$ son los pesos correcpondientes a cada una de las observaciones. En el ejemplo de los datos de **salmon** vemos que existe una estacionalidad anual y podemos entonces darle pesos a los pronósticos de acuerdo a los valores de un año atrás (enero de el año actual con enero del año pasado, etc).

Pronósticos con suavizamiento exponencial ponderado:

Aplicamos la función **sma_forecast** utilizando ponderaciones:

```
salmon_fore_w1 <- sma_forecast(salmon_aux, h=12,
                  m=48, w= c( rep(0,11),0.45,
                             rep(0,11), 0.35,
                             rep(0,11), 0.15,
                             rep(0,11), 0.05))
```

En este caso se da un peso de 0.45 (45 %) al dato de un año antes, 0.35 (35 %) al dato dos años antes, etc.

Pronósticos con suavizamiento exponencial ponderado:

Aplicamos la función **sma_forecast** utilizando ponderaciones:

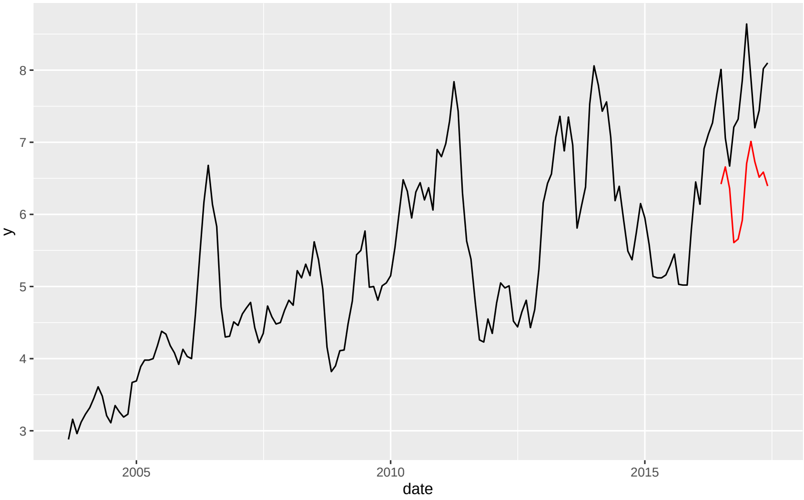
```
salmon_fore_w1 <- sma_forecast(salmon_aux, h=12,
                  m=48, w= c( rep(0,11),0.45,
                              rep(0,11), 0.35,
                              rep(0,11), 0.15,
                              rep(0,11), 0.05))
```

En este caso se da un peso de 0.45 (45 %) al dato de un año antes, 0.35 (35 %) al dato dos años antes, etc.

Graficamos:

```
salmon_fore_w1  %>% ggplot(aes(x=date, y=y))+
  geom_line()+
  geom_line(aes(x=date, y=yhat), col="red")
```

Pronósticos con modelos de suavizamiento exponencial



NOTA: No se captura la tendencia creciente.

Pronósticos con el método de Holt

Este método se basa en estimar el nivel más reciente y la tendencia mediante el uso de dos parámetros de suavizamiento α y β. Si denotamos por L_n al valor del nivel más reciente y T_n a la tendencia más reciente, usamos el pronóstico de la serie en el instante $n+h$ como:

$$\widehat{X}_{n+h} = L_n + hT_n$$

Los cálculos para los valores más recientes de los niveles y la tendencia se obtienen con las ecuaciones:

$$L_n = \alpha X_n + (1-\alpha)(L_{n-1} + T_{n-1})$$

$$T_n = \beta(L_n - L_{n-1}) + (1-\beta)T_{n-1}$$

Dado un conjunto de **entrenamiento** el método de Holt consiste en encontrar el α y el β que minimicen la suma cuadrática del error.

Pronósticos con el método de Holt

```r
salmon_par <- ts_split(salmon, sample.out = 12)
train <- salmon_par$train
test <- salmon_par$test
fore_holt <- holt(train, h = 12, initial = "optimal")
fore_holt$model
```

```
## Holt's method
##
## Call:
##  holt(y = train, h = 12, initial = "optimal")
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 1e-04
##
##   Initial states:
##     l = 2.6089
##     b = 0.0339
##
##   sigma:  0.3949
##
##      AIC     AICc      BIC
## 495.4985 495.9039 510.6832
```

Pronósticos con el método de Holt

Obtenemos la precisión del modelo:

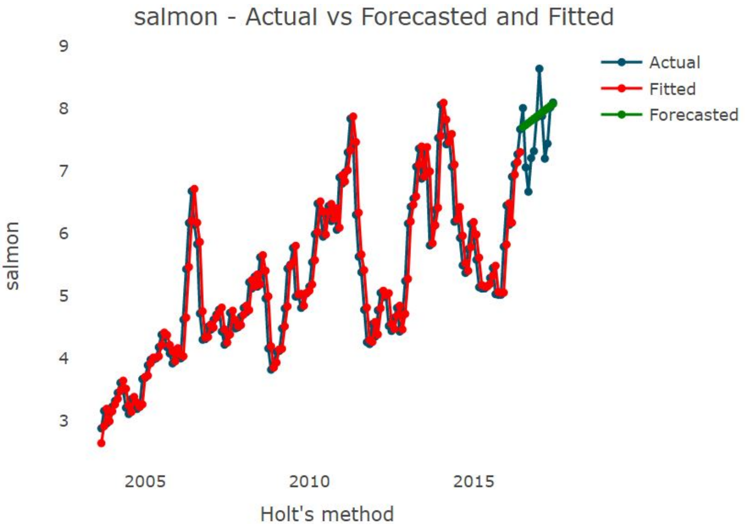
```
accuracy(fore_holt, test)
```

```
##                         ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.001025402 0.3897762 0.2976661 -0.2428901 5.598641 0.2765364
## Test set     -0.272667037 0.5669235 0.4497604 -4.0420436 6.141209 0.4178343
##                   ACF1 Theil's U
## Training set 0.2890116        NA
## Test set     0.3038011  1.060374
```

Graficamos con la función

```
test_forecast(salmon, forecast.obj = fore_holt,
                   test = test)
```

Pronósticos con el método de Holt



Como vimos antes, el problema del método de Holt es que no tiene en cuenta la estacionalidad de la serie. El método de Holt-Winters es una extensión del método de Holt y se plantea como:

$$\widehat{X}_{n+h} = L_n + hT_n + S_{n+h-k}$$

donde:

$$L_n = \alpha(X_n - S_{n-k}) + (1-\alpha)(L_{n-1} + T_{n-1})$$

$$T_n = \beta(L_n - L_{n-1}) + (1-\beta)T_{n-1}$$

$$S_n = \gamma(X_n - L_n) + (1-\gamma)S_{n-k}$$

Dado un conjunto de **entrenamiento** el método de Holt-Winters consiste en encontrar el α, β y γ que minimicen la suma cuadrática del error.

Pronósticos con el método de Holt-Winters

En el paquete **stats** se encuentra la función **HoltWinters**:

```r
salmon_par <- ts_split(salmon, sample.out = 12)
train <- salmon_par$train
test <- salmon_par$test
modelo_hw <- HoltWinters(train)
modelo_hw
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = train)
##
## Smoothing parameters:
##  alpha: 0.9297855
##  beta : 0
##  gamma: 1
##
## Coefficients:
##            [,1]
## a    7.69325851
## b    0.04523456
## s1  -0.09571717
## s2  -0.19934946
## s3  -0.40825507
## s4  -0.41252986
## s5  -0.32527174
```

Pronósticos con el método de Holt-Winters

Realizamos los pronósticos y vemos la precisión del modelo ajustado:

```
fore_hw <- forecast(modelo_hw, h = 12)
fore_hw
```

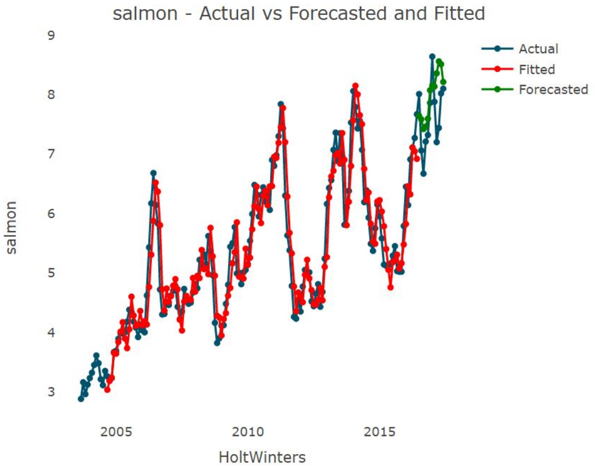
```
##          Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
## Jul 2016       7.642776 7.161272  8.124280 6.906379  8.379172
## Aug 2016       7.584378 6.926900  8.241856 6.578853  8.589904
## Sep 2016       7.420707 6.625278  8.216137 6.204203  8.637211
## Oct 2016       7.461667 6.548902  8.374431 6.065714  8.857620
## Nov 2016       7.594160 6.577513  8.610806 6.039333  9.148986
## Dec 2016       8.073866 6.963010  9.184722 6.374959  9.772774
## Jan 2017       8.150661 6.952983  9.348339 6.318971  9.982351
## Feb 2017       8.135370 6.856753  9.413988 6.179893 10.090847
## Mar 2017       8.358468 7.003737  9.713198 6.286587 10.430349
## Apr 2017       8.558764 7.131976  9.985553 6.376680 10.740849
## May 2017       8.511305 7.015926 10.006683 6.224321 10.798288
## Jun 2017       8.212815 6.651858  9.773772 5.825537 10.600093
```

```
accuracy(fore_hw, test)
```

```
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.01428828 0.3746667 0.2933025 -0.5638717 5.389225 0.2724825
## Test set     -0.35791140 0.5993108 0.5006719 -5.0231274 6.731163 0.4651319
##                   ACF1 Theil's U
## Training set 0.2986668        NA
```

Graficamos con la función

```
test_forecast(salmon, forecast.obj = fore_hw,
                      test = test)
```



Pronósticos: Ejemplo de USgas

Como vimos antes, los datos de la BD **salmon** no tienen un comportamiento estacional anual muy evidente o regular, por este motivo el método de Holter-Winters no generó pronósticos totalmente claros.

Pronósticos: Ejemplo de USgas

Como vimos antes, los datos de la BD **salmon** no tienen un comportamiento estacional anual muy evidente o regular, por este motivo el método de Holter-Winters no generó pronósticos totalmente claros.

A continuación aplicarémos tres métodos vistos en esta clase para tratar de pronosticar la demanda de gas:

```
require(TSstudio)
USgas %>% ts_info()
```

```
##  The . series is a ts object with 1 variable and 238 observat
##  Frequency: 12
##  Start time: 2000 1
##  End time: 2019 10
```

Pronósticos con el método de Holt-Winters

```r
USgas_par <- ts_split(USgas, 12)
train_gas <- USgas_par$train
test_gas <- USgas_par$test

# Medias móviles simples:
USgas_aux <- USgas %>% ts_to_prophet()
modelo_sma <- sma_forecast(USgas_aux, h=12, m=4)

# Modelo Holt:
modelo_holt <- holt(train_gas, h = 12,
                    initial = "optimal")
# Modelo Holt-Winters:
modelo_hw <- HoltWinters(train_gas)
fore_hw <- forecast(modelo_hw, h = 12)
```

Vemos la precisión de los tres modelos:

```
accuracy(modelo_sma$yhat, modelo_sma$test)
```

```
##                 ME     RMSE      MAE      MPE     MAPE
## Test set 327.1106 516.2866 386.7022 10.67589 13.46154
```

```
accuracy(modelo_holt, test_gas)
```

```
##                       ME     RMSE      MAE        MPE     MAPE     MASE
## Training set   0.8646902 285.2349 228.1735 -0.8280563 10.94771 1.981613
## Test set     310.1067256 502.4591 380.6257 10.0214029 13.31273 3.305612
##                   ACF1 Theil's U
## Training set 0.3747401        NA
## Test set     0.6969289  1.508713
```

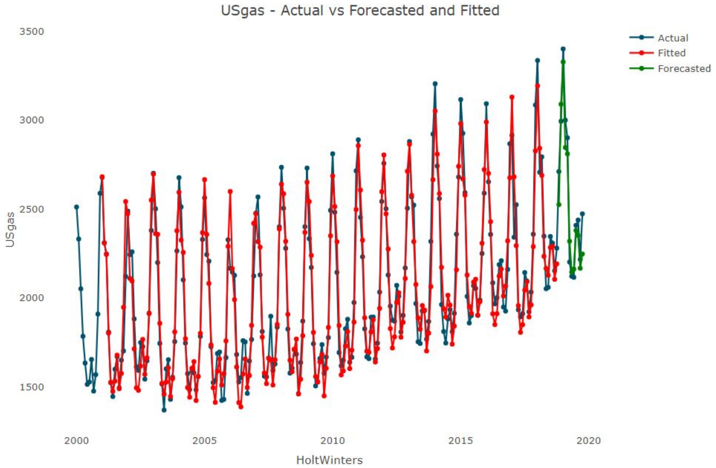
```
accuracy(fore_hw, test_gas)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set  7.828361 115.2062 87.67420 0.2991952 4.248131 0.7614222
## Test set     51.013877 115.1555 98.06531 1.7994297 3.766099 0.8516656
##                     ACF1 Theil's U
## Training set  0.21911103        NA
## Test set     -0.01991923 0.3652142
```

Pronósticos con el método de Holt-Winters

Graficamos con la función

```
test_forecast(actual = USgas,
              forecast.obj = fore_hw, test = test_gas)
```



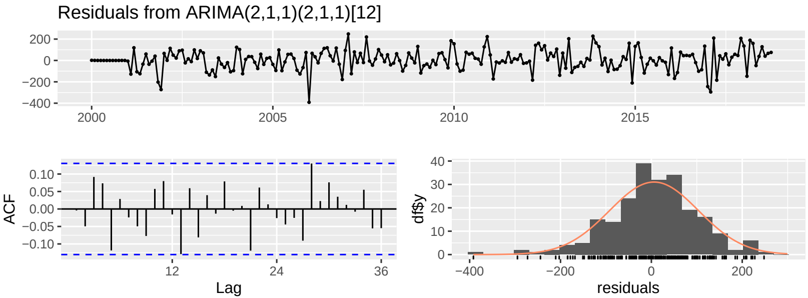
Si aplicamos un **auto.arima** a la serie vemos que:

```
modelo_auto1 <- auto.arima(train_gas)
modelo_auto1
```

```
## Series: train_gas
## ARIMA(2,1,1)(2,1,1)[12]
##
## Coefficients:
##          ar1      ar2      ma1    sar1     sar2     sma1
##       0.4301  -0.0372  -0.9098  0.0117  -0.2673  -0.7431
## s.e.  0.0794   0.0741   0.0452  0.0887   0.0830   0.0751
##
## sigma^2 = 10446:  log likelihood = -1292.83
## AIC=2599.67   AICc=2600.22   BIC=2623.2
```

Pronósticos: Ejemplo de USgas

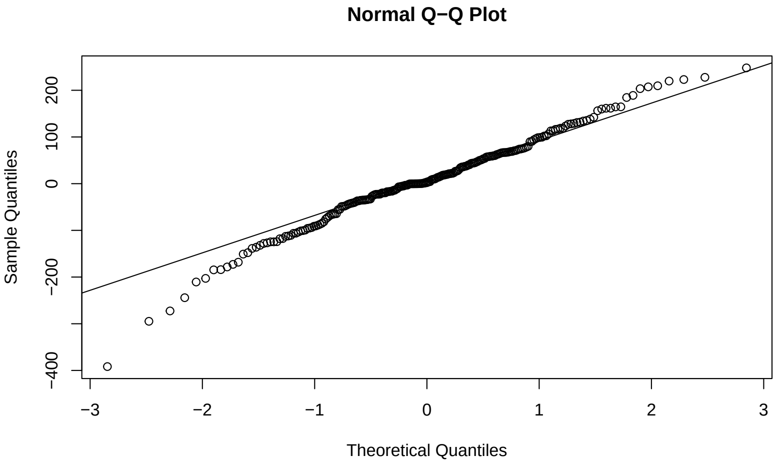
```
modelo_auto1 %>% checkresiduals(lag=25)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)(2,1,1)[12]
## Q* = 25.451, df = 19, p-value = 0.1462
##
## Model df: 6.   Total lags used: 25
```

Pronósticos: Ejemplo de USgas

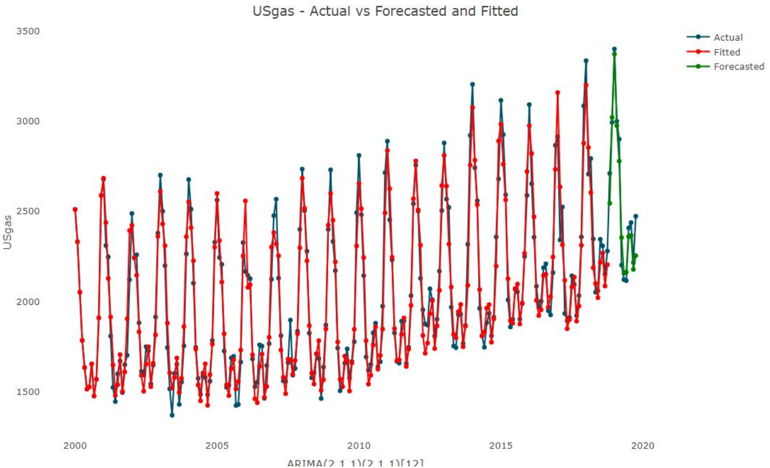
```
modelo_auto1$residuals %>% qqnorm()
modelo_auto1$residuals %>% qqline()
```



```
require(forecast)
USgas_fc_auto1 <- forecast(modelo_auto1, h = 12)
USgas_fc_auto1
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Nov 2018       2543.848 2412.863 2674.832 2343.524 2744.171
## Dec 2018       3020.624 2872.969 3168.278 2794.805 3246.442
## Jan 2019       3372.018 3219.976 3524.059 3139.491 3604.545
## Feb 2019       2975.209 2821.146 3129.272 2739.590 3210.828
## Mar 2019       2777.974 2622.463 2933.485 2540.141 3015.807
## Apr 2019       2353.685 2196.894 2510.476 2113.894 2593.476
## May 2019       2155.955 1997.939 2313.970 1914.291 2397.618
## Jun 2019       2161.846 2002.630 2321.062 1918.346 2405.346
## Jul 2019       2359.808 2199.405 2520.212 2114.492 2605.125
## Aug 2019       2365.725 2204.144 2527.306 2118.608 2612.842
## Sep 2019       2177.364 2014.615 2340.114 1928.460 2426.268
## Oct 2019       2253.671 2089.761 2417.581 2002.993 2504.350
```

```
test_forecast(actual = train_USgas,
              forecast.obj = USgas_fc_auto1,
              test = test_gas)
```



Vemos la precisión de los dos modelos:

```
accuracy(fore_hw, test_gas)
```

```
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set  7.828361 115.2062 87.67420 0.2991952 4.248131 0.7614222
## Test set     51.013877 115.1555 98.06531 1.7994297 3.766099 0.8516656
##                     ACF1 Theil's U
## Training set  0.21911103        NA
## Test set     -0.01991923 0.3652142
```

```
accuracy(USgas_fc_auto1$mean, test_gas)
```

```
##                ME     RMSE      MAE      MPE     MAPE        ACF1 Theil's U
## Test set 37.84788 103.2285 81.46603 1.310799 3.261643 -0.04670893 0.3404092
```

Como vemos el modelo obtenido con **auto.arima** arroja mejores resultados, pero no cumple con el supuesto de normalidad. Con Holter-Winters dicho supuesto no fue necesario.