AMAZON y Forecasts

2023-07-03

Analisis de Amazon, series temporales

Tomaremos los datos desde yahoo y utilizaremos el precio de cierre.

```
options(digits = 3)
options(warn = - 1)
##Obtenemos precios de AMAZON
AMZN<-getSymbols("AMZN", from="2020-08-01",to="2021-03-31", src = "yahoo", auto.assign = FALSE) #
# Eliminando valores faltantes
AMZN <- na.omit(AMZN)
# Mantenemos columnas con Precios de Cierre  columna 4:
AMZN <- AMZN[,4]
print(head(AMZN, n = 5))
```

```
##            AMZN.Close
## 2020-08-03        156
## 2020-08-04        157
## 2020-08-05        160
## 2020-08-06        161
## 2020-08-07        158
```

```
##Podemos graficar:
# Gráfico utilizando ggplot2
ggplot(data = AMZN, aes(x = index(AMZN), y = AMZN$AMZN.Close)) +
  geom_line(color = "blue") +
  labs(x = "Fecha", y = "Precios", title = "Gráfico de precios de Amazon") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <xts/zoo>.
## Defaulting to continuous.
```


```
length(AMZN)
```

```
## [1] 166
```

```
##Partimos serie, tomemos el 7% para la prueba
h <- round(length(AMZN)*0.07, digits = 0 )
h
```

```
## [1] 12
```

```
train <- AMZN[1:(nrow(AMZN) - h), ]
test<- AMZN[(nrow(AMZN) - h + 1):nrow(AMZN), ]

# Crear el data frame df2 con los conjuntos de train y test
df2 <- data.frame(Date = as.Date(index(AMZN)), Train = c(coredata(train), rep(NA, h)), Test = c(rep(NA,

# Crear el gráfico utilizando ggplot2
ggplot(data = df2) +
  geom_line(aes(x = Date, y = Train, color = "Train")) +
  geom_line(aes(x = Date, y = Test, color = "Test")) +
  labs(x = "Fecha", y = "Precios", title = "Gráfico de precios de Amazon") +
  scale_color_manual(values = c("Train" = "red", "Test" = "green")) +
  theme_minimal()
```

Modelo ARIMA

verificaremos si la serie es estacionaria y poder aplicar el modelo ARIMA e añadiremos a una tabla de criterios para evaluar

```
############################################################
############ Modelos ARIMA ################################
###Veamos si la serie es estacionaria:
adfTest(train)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -0.1793
##   P VALUE:
##     0.5571
##
## Description:
##  Tue Jul 04 01:30:05 2023 by user: basti
```

```r
##Como no es estacionaria, la diferenciamos y vemos si ya es estacionaria:
dtrain<-diff(train)[-1,]
adfTest(dtrain)  #con libreria fUnitRoorts
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -9.1947
##   P VALUE:
##     0.01
##
## Description:
##  Tue Jul 04 01:30:05 2023 by user: basti
```

```r
adf.test(dtrain)  #con librería tseries
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  dtrain
## Dickey-Fuller = -6, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

```r
############################333
##Ya estacionaria, definimos candidatos de modelos ARMA

m<-eacf(dtrain, 15,10)  #Seria un ARMA(7,3), pero si deseamos expresarla como ARIMA, sería: ARIMA (7,1,
```

```
## AR/MA
##    0 1 2 3 4 5 6 7 8 9 10
## 0  o o x o o o o o o o o
## 1  x o x o o o o o o o o
## 2  x o x o o o o o o o o
## 3  o x x o o o o o o o o
## 4  o x x x o o o o o o o
## 5  x x x x o o o o o o o
## 6  x x x x o o o o o o o
## 7  o o x o o o x o o o o
## 8  o x x x o o x o o o o
## 9  x o x o o o x o x o o
## 10 x o o x o o o o o x o
## 11 x x o x o o o x o x o
## 12 x o x x x x o o o x x
## 13 x o x o o x o o o o o
## 14 x x o o o x o o o o o
## 15 x x o o o x o o o o o
```

```r
#Definamos otros modelos mediante la función auto.arima()
m2<-auto.arima(train, seasonal = TRUE)
summary(m2)   #Sería arima(1,0,0)
```

```
## Series: train
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##         ar1    mean
##       0.828  159.52
## s.e.  0.044    1.41
##
## sigma^2 = 9.73:  log likelihood = -393
## AIC=793   AICc=793   BIC=802
##
## Training set error measures:
##                     ME RMSE  MAE     MPE MAPE  MASE   ACF1
## Training set 0.0323  3.1 2.37 -0.0176 1.48 0.966 0.0434
```

```
###Modelación:
mod1<-Arima(train, order=c(7,1,3), method = "ML")
summary(mod1)
```

```
## Series: train
## ARIMA(7,1,3)
##
## Coefficients:
##         ar1    ar2     ar3    ar4    ar5     ar6    ar7     ma1     ma2    ma3
##       0.606  0.370  -0.830  0.158  0.005  -0.208  0.176  -0.677  -0.392  0.697
## s.e.  0.170  0.201   0.187  0.120  0.101   0.101  0.090   0.160   0.199  0.177
##
## sigma^2 = 10.1:  log likelihood = -389
## AIC=800   AICc=802   BIC=833
##
## Training set error measures:
##                      ME RMSE  MAE     MPE MAPE  MASE   ACF1
## Training set -0.00484 3.06 2.38 -0.0237 1.49 0.969 0.0104
```

```
coeftest(mod1)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.60611    0.16970    3.57  0.00035 ***
## ar2  0.36982    0.20091    1.84  0.06566 .
## ar3 -0.82962    0.18659   -4.45  8.7e-06 ***
## ar4  0.15820    0.11972    1.32  0.18636
## ar5  0.00531    0.10128    0.05  0.95815
## ar6 -0.20796    0.10131   -2.05  0.04009 *
## ar7  0.17553    0.09031    1.94  0.05194 .
## ma1 -0.67731    0.15980   -4.24  2.3e-05 ***
## ma2 -0.39202    0.19892   -1.97  0.04875 *
## ma3  0.69736    0.17668    3.95  7.9e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
tsdiag(mod1)  ##residuos sw ven ok.

mod2<-Arima(train, order=c(1,0,0), method = "ML")
```

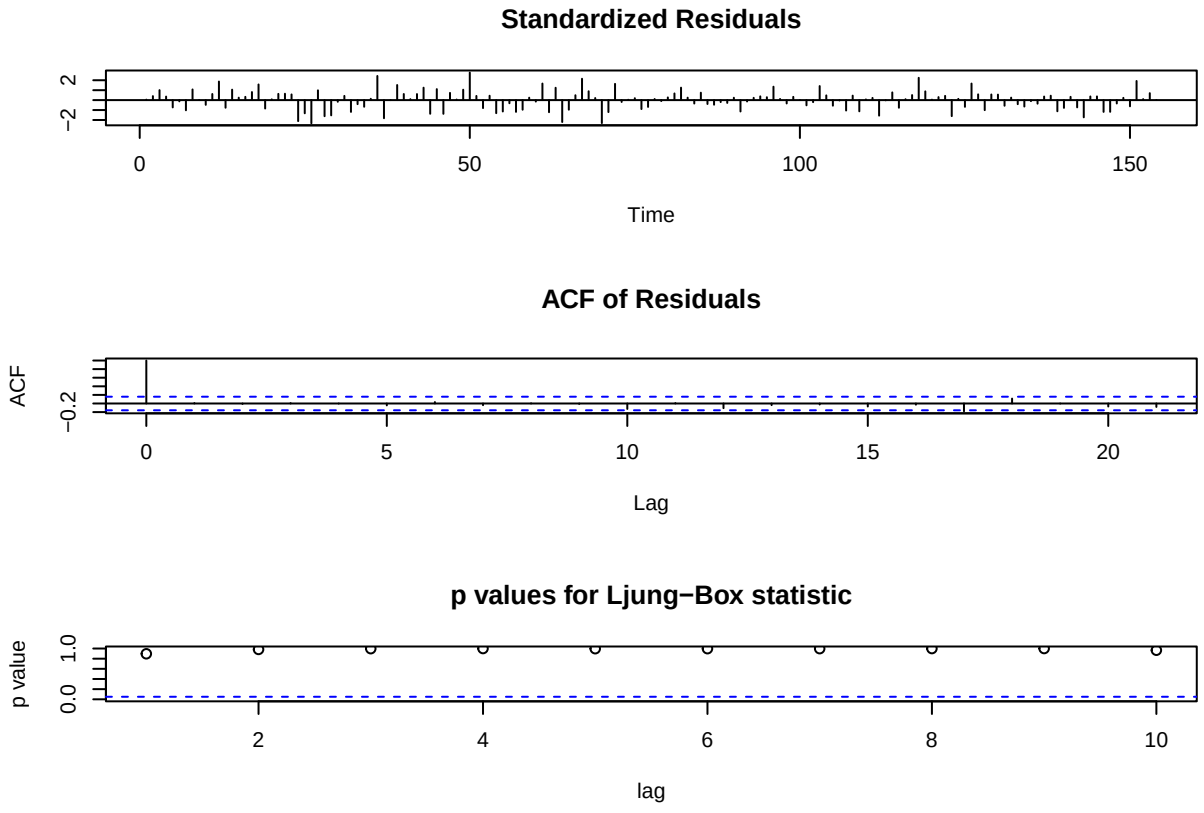
```
mod2
```

```
## Series: train
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##         ar1    mean
##       0.828  159.52
## s.e.  0.044    1.41
##
## sigma^2 = 9.73:  log likelihood = -393
## AIC=793   AICc=793   BIC=802
```

```
coeftest(mod2)
```

```
##
## z test of coefficients:
##
##           Estimate Std. Error z value Pr(>|z|)
## ar1         0.8277     0.0445    18.6   <2e-16 ***
## intercept 159.5193     1.4089   113.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
tsdiag(mod1)   #Residuos se ven ok
```




```
##Pronosticos
#install.packages('forecast', dependencies = TRUE)
```

```r
### Modelos pronóstico para m1 y m2:
Pron_m1<-forecast(mod1, h)
Pron_m2<- forecast(mod2, h)

summary(Pron_m1)
```

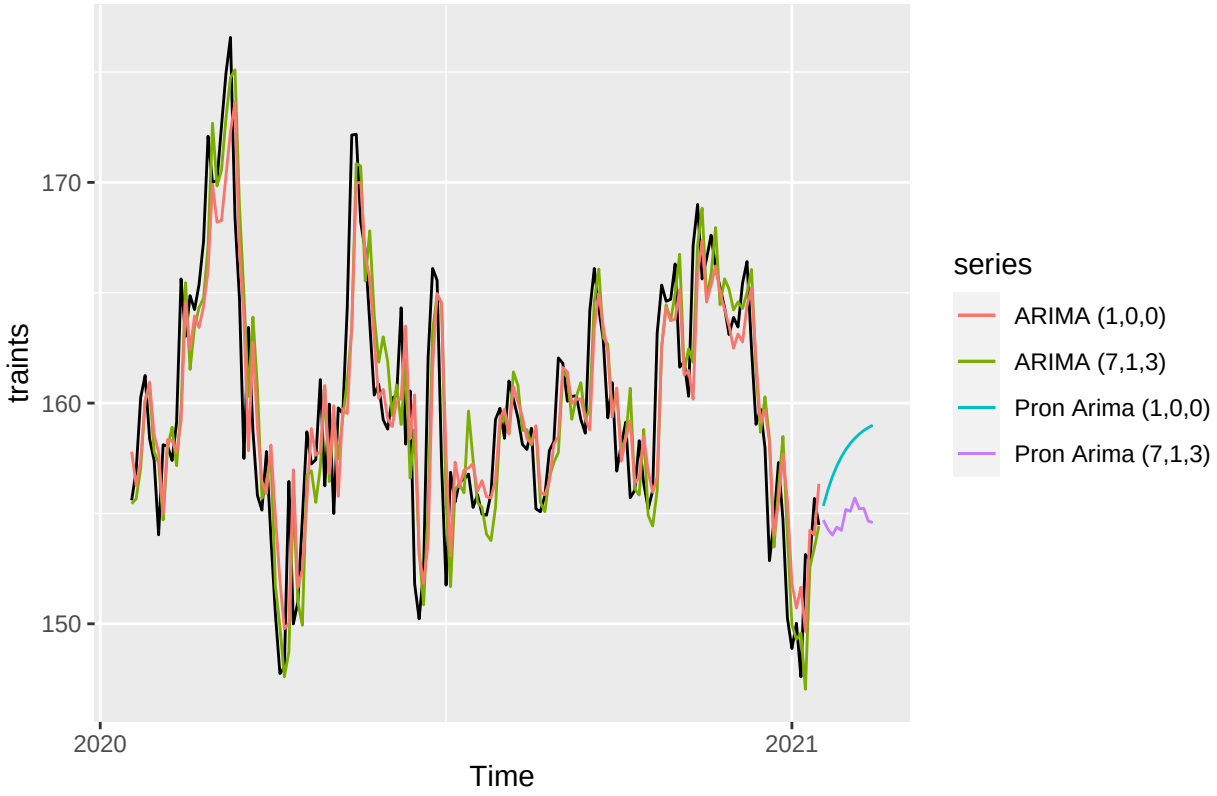
```
##
## Forecast method: ARIMA(7,1,3)
##
## Model Information:
## Series: train
## ARIMA(7,1,3)
##
## Coefficients:
##         ar1    ar2     ar3    ar4    ar5     ar6    ar7     ma1     ma2    ma3
##       0.606  0.370  -0.830  0.158  0.005  -0.208  0.176  -0.677  -0.392  0.697
## s.e.  0.170  0.201   0.187  0.120  0.101   0.101  0.090   0.160   0.199  0.177
##
## sigma^2 = 10.1:  log likelihood = -389
## AIC=800   AICc=802   BIC=833
##
## Error measures:
##                     ME RMSE  MAE     MPE MAPE  MASE   ACF1
## Training set -0.00484 3.06 2.38 -0.0237 1.49 0.969 0.0104
##
## Forecasts:
##     Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 155            155   151   159   148   161
## 156            154   149   160   146   163
## 157            154   147   161   144   164
## 158            154   147   161   144   165
## 159            154   147   162   142   166
## 160            155   147   163   142   168
## 161            155   146   164   142   169
## 162            156   146   165   141   170
## 163            155   145   165   140   171
## 164            155   144   166   139   172
## 165            155   143   166   137   172
## 166            155   142   167   136   173
```

```r
summary(Pron_m2)
```

```
##
## Forecast method: ARIMA(1,0,0) with non-zero mean
##
## Model Information:
## Series: train
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##         ar1    mean
##       0.828  159.52
## s.e.  0.044    1.41
##
```

```
## sigma^2 = 9.73:  log likelihood = -393
## AIC=793   AICc=793   BIC=802
##
## Error measures:
##                    ME RMSE  MAE     MPE MAPE  MASE   ACF1
## Training set 0.0324  3.1 2.37 -0.0175 1.48 0.966 0.0434
##
## Forecasts:
##     Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 155            155   151   159   149   161
## 156            156   151   161   148   164
## 157            157   151   163   148   166
## 158            157   151   163   148   167
## 159            158   151   164   148   168
## 160            158   151   165   148   168
## 161            158   151   165   148   169
## 162            158   151   165   148   169
## 163            159   152   166   148   169
## 164            159   152   166   148   170
## 165            159   152   166   148   170
## 166            159   152   166   148   170
```

```
##Otro Gráfico integral:
##pasamos a ts los datos, son 154 datos en la parte train:
traints<-ts(train, start=c(2020,08,01), frequency = 154)
fitted1<-ts(mod1$fitted,start=c(2020,08,01), frequency = 154 )
fitted2<-ts(mod2$fitted,start=c(2020,08,01), frequency = 154 )
pron1<-ts(Pron_m1$mean, start = c(2021,08), frequency = 154)
pron2<-ts(Pron_m2$mean, start = c(2021,08), frequency = 154)

autoplot(traints)+
  autolayer(fitted1, series="ARIMA (7,1,3)")+
  autolayer(fitted2, series="ARIMA (1,0,0)")+
  autolayer(pron1, series="Pron Arima (7,1,3)")+
  autolayer(pron2, series="Pron Arima (1,0,0)")
```

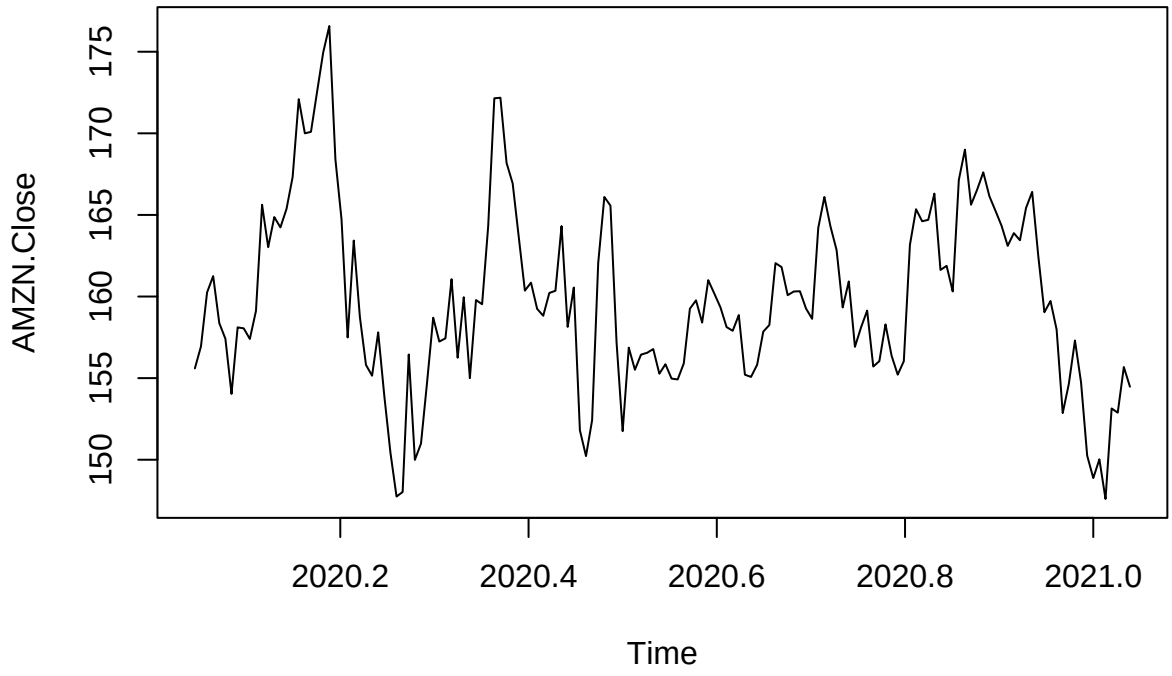



```
#### Midamos el error de pronóstico, RMSE y MAPE:
library(Metrics)
RMSE_arima<-rmse(test, Pron_m1$mean)
RMSEar1<-rmse(test, Pron_m2$mean)

MAPE_arima<-mape(test, Pron_m1$mean)
MAPEar1<-mape(test, Pron_m2$mean)

##imprimir los resultados al momento:
###Imprimamos los resultados en una tabla:

Modelo<-c("ARIMA(7,1,3)", "AR(3)")

RMSE<-c(RMSE_arima, RMSEar1)

MAPE<-c(MAPE_arima, MAPEar1)

res<-data.frame(Modelo,RMSE, MAPE)

print((res))
```

```
##         Modelo RMSE   MAPE
## 1 ARIMA(7,1,3) 1.86 0.0102
## 2        AR(3) 4.26 0.0239
```

Modelo Suavizamiento Exponencial

```
#########################################################
############ Modelos Suavizamiento Exponencial ########

traints<-ts(train, start=c(2020,08,01), frequency = 154)
plot(traints)
```

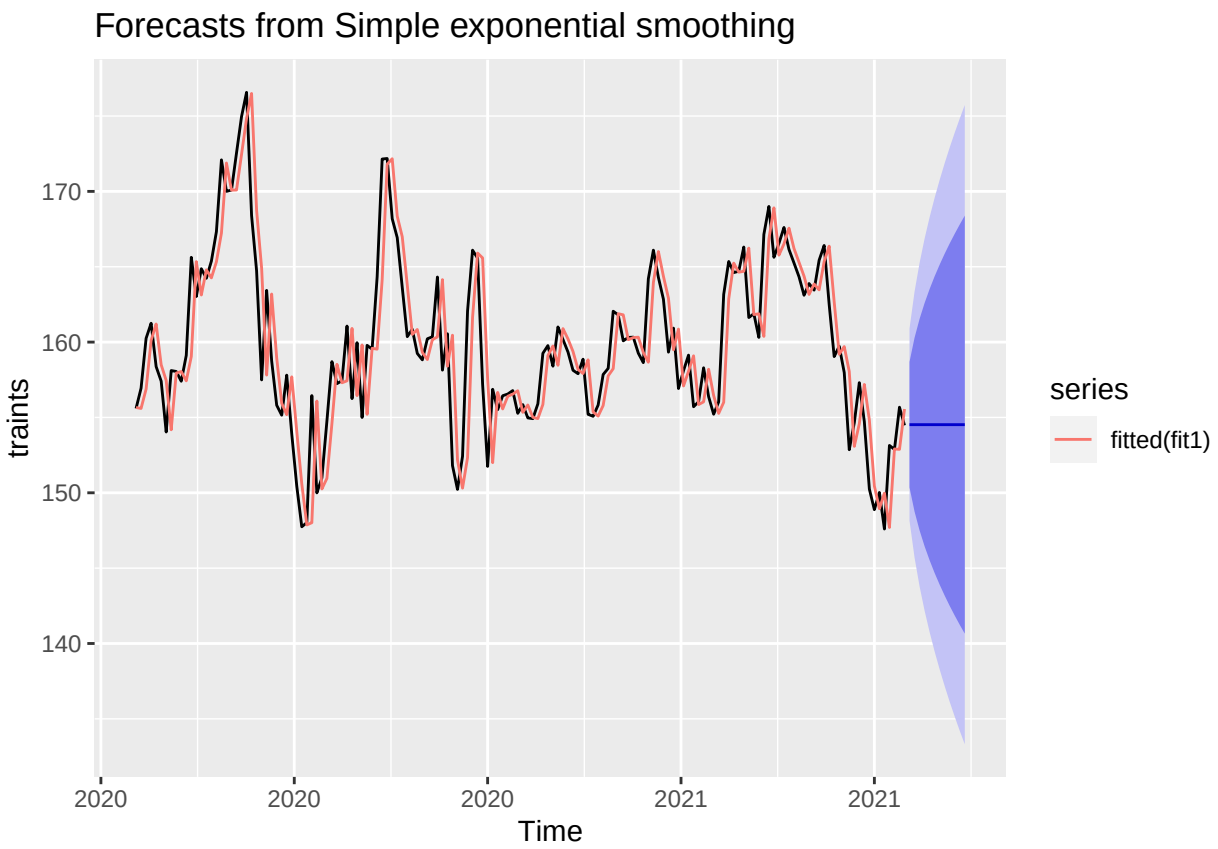



```
##No hay una estacionalidad evidente, por lo que probamos modelos de suavizamiento simples
###Posibles enfoques de suavizamiento:
##
###Primer Modelo:
fit1<-ses(traints, h=12 )
summary(fit1)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = traints, h = 12)
##
##   Smoothing parameters:
##     alpha = 0.957
```

```
##
##   Initial states:
##     l = 155.6599
##
##   sigma:  3.25
##
##  AIC AICc  BIC
## 1143 1143 1152
##
## Error measures:
##                     ME RMSE  MAE     MPE MAPE MASE    ACF1
## Training set -0.00773 3.23 2.43 -0.0263 1.52  NaN 0.00237
##
## Forecasts:
##          Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 2021.045            155   150   159   148   161
## 2021.052            155   149   160   146   163
## 2021.058            155   148   162   144   165
## 2021.065            155   146   163   142   167
## 2021.071            155   146   164   141   168
## 2021.078            155   145   164   139   170
## 2021.084            155   144   165   138   171
## 2021.091            155   143   166   137   172
## 2021.097            155   142   167   136   173
## 2021.104            155   142   167   135   174
## 2021.110            155   141   168   134   175
## 2021.117            155   141   168   133   176
```

```
ffit1<-forecast(fit1, h=12)
autoplot(fit1) +  autolayer(fitted(fit1))
```

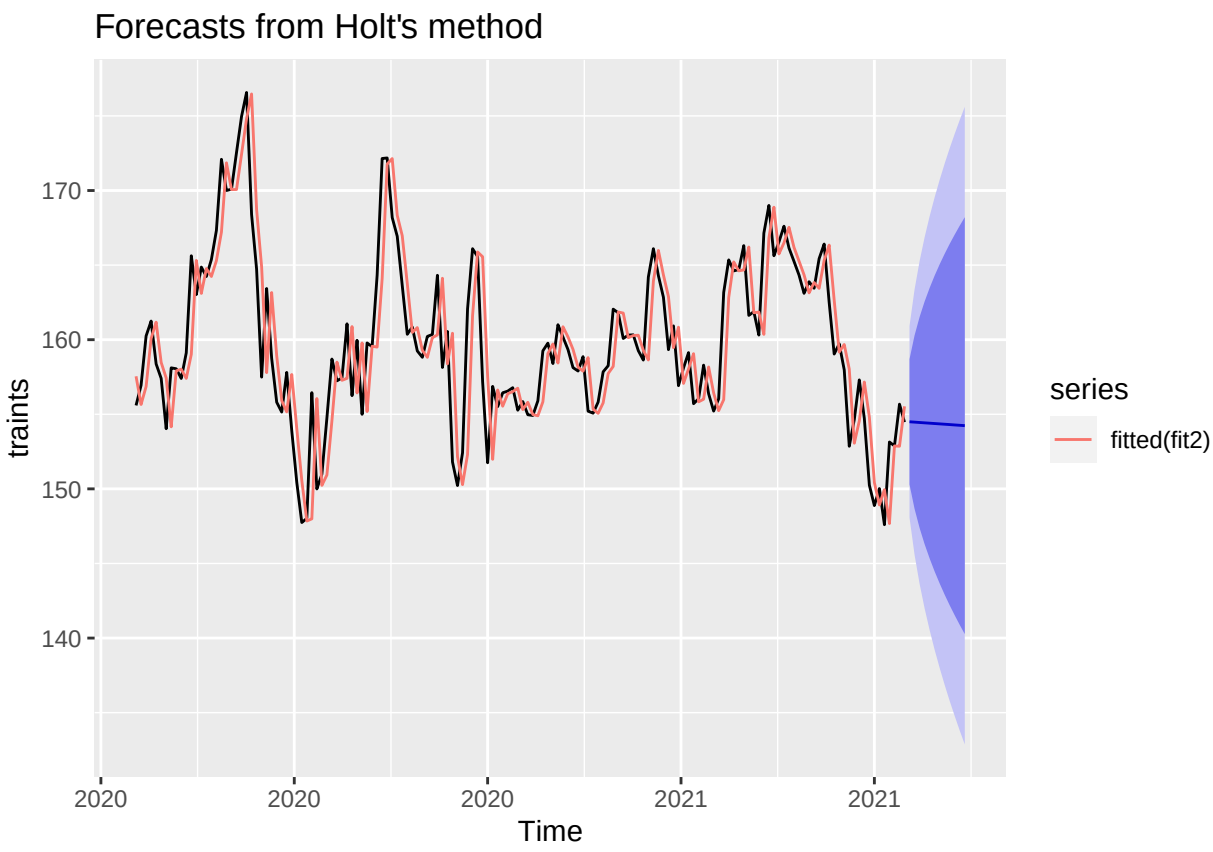



```
##############
#Segundo Modelo: Tendencia Lineal con Holt, podríamos probar aún este, auque no hay tendencia evidente:
fit2 <- holt(traints,h=12)
summary(fit2)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = traints, h = 12)
##
##   Smoothing parameters:
##     alpha = 0.9561
##     beta  = 1e-04
##
##   Initial states:
##     l = 157.5703
##     b = -0.0235
##
##   sigma:  3.28
##
##  AIC AICc  BIC
## 1147 1148 1162
```

```
##
## Error measures:
##                     ME RMSE  MAE     MPE MAPE MASE    ACF1
## Training set 0.00345 3.23 2.44 -0.0195 1.53  NaN 0.00158
##
## Forecasts:
##          Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 2021.045            154   150   159   148   161
## 2021.052            154   149   160   146   163
## 2021.058            154   147   162   144   165
## 2021.065            154   146   163   142   167
## 2021.071            154   145   163   141   168
## 2021.078            154   144   164   139   170
## 2021.084            154   144   165   138   171
## 2021.091            154   143   166   137   172
## 2021.097            154   142   166   136   173
## 2021.104            154   142   167   135   174
## 2021.110            154   141   168   134   175
## 2021.117            154   140   168   133   176
```

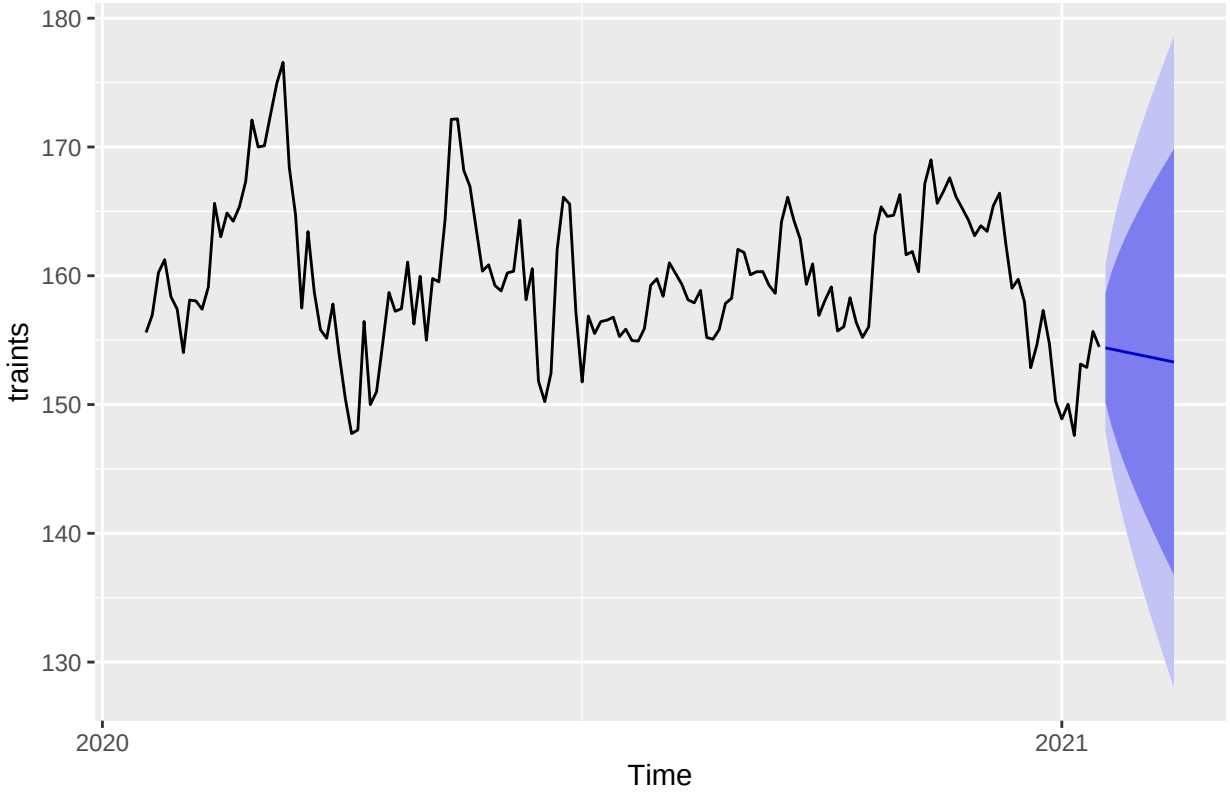
```
ffit2<-forecast(fit2, h=12)
autoplot(fit2) +  autolayer(fitted(fit2))
```




```
###############
###Tercer Modelo: Holt con corrección a la deriva (sin estacionalidad):
fit3<-HoltWinters(traints, alpha = NULL, beta=NULL, gamma = FALSE)
fit3
```

```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = traints, alpha = NULL, beta = NULL, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.976
##  beta : 0.0268
##  gamma: FALSE
##
## Coefficients:
##    [,1]
## a 154.5
## b  -0.1
```

```
ffit3<-forecast(fit3, h=12)
autoplot(traints)+autolayer(ffit3)
```

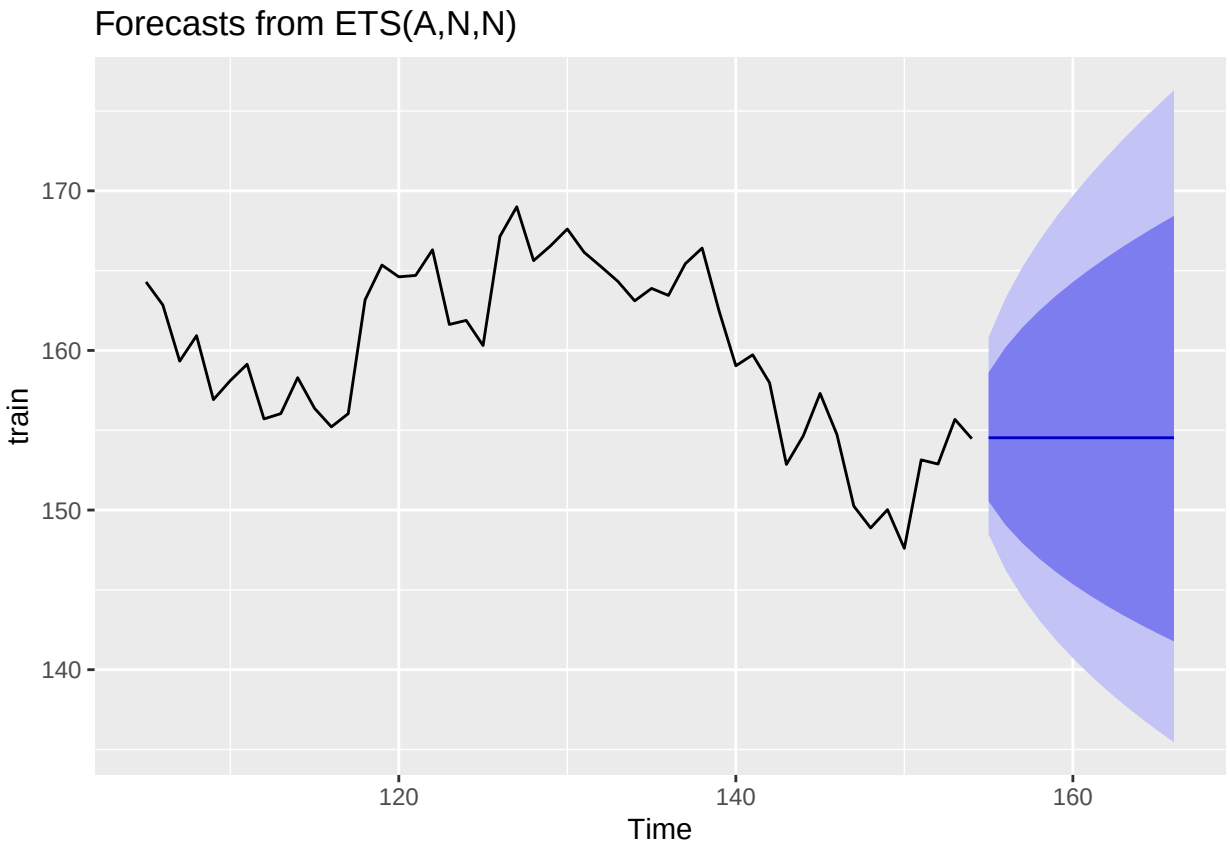



```
###Cuarto Modelo: Probemos la aplicación de ets(), que deje determine el modelo
fit4<-ets(train, model="ZZZ", damped=FALSE, alpha=NULL, beta=NULL,
          gamma=NULL, phi=NULL, lambda=FALSE, biasadj=FALSE,
          additive.only=FALSE, restrict=TRUE,
          allow.multiplicative.trend=FALSE)
summary(fit4)
```

```
## ETS(A,N,N)
##
```

```
## Call:
##  ets(y = train, model = "ZZZ", damped = FALSE, alpha = NULL, beta = NULL,
##
##  Call:
##      gamma = NULL, phi = NULL, additive.only = FALSE, lambda = FALSE,
##
##  Call:
##      biasadj = FALSE, restrict = TRUE, allow.multiplicative.trend = FALSE)
##
##   Box-Cox transformation: lambda= 0
##
##   Smoothing parameters:
##     alpha = 0.9484
##
##   Initial states:
##     l = 5.0477
##
##   sigma:  0.0204
##
##  AIC AICc  BIC
## -420 -419 -410
##
## Training set error measures:
##                      ME RMSE  MAE     MPE MAPE MASE   ACF1
## Training set -0.00612 3.23 2.43 -0.0255 1.52 0.99 0.0104
```

```
ffit4<-forecast(fit4, h=12 )
autoplot(forecast(fit4,h=12), include=50)
```




```
###############
##Métrica Desempeño pronóstico:
RMSEses<-rmse(test, ffit1$mean)
RMSEholt<-rmse(test, ffit2$mean)
RMSE_HW<-rmse(test, ffit3$mean)
RMSEets<-rmse(test, ffit4$mean)

MAPEses<-mape(test, ffit1$mean)
MAPEholt<-mape(test, ffit2$mean)
MAPE_HW<-mape(test, ffit3$mean)
MAPEets<-mape(test, ffit4$mean)


###Imprimamos los resultados en una tabla:

Modelo<-c("ARIMA(7,1,3)", "AR(3)", "ses", "holt", "HW", "ets")

RMSE<-c(RMSE_arima, RMSEar1, RMSEses, RMSEholt, RMSE_HW, RMSEets)

MAPE<-c(MAPE_arima, MAPEar1, MAPEses, MAPEholt, MAPE_HW, MAPEets)

res<-data.frame(Modelo,RMSE, MAPE)

print((res))
```

```
##         Modelo RMSE    MAPE
```

```
## 1 ARIMA(7,1,3) 1.86 0.01022
## 2        AR(3) 4.26 0.02388
## 3          ses 1.69 0.00912
## 4         holt 1.64 0.00860
## 5           HW 1.57 0.00789
## 6          ets 1.70 0.00914
```

Modelo Red Neuronal tipo Feed Forward Neural

```
##############################################################################
#################  Red Neuronal tipo Feed Forward Neural ###############
##Neural Network Time Serie Regression ##https://pkg.robjhyndman.com/forecast/reference/nnetar.html
##Función:nnetar: "Feed-forward neural networks with a single hidden layer and lagged inputs for foreca
# Gráfico utilizando ggplot2
ggplot(data = AMZN, aes(x = index(AMZN), y = AMZN$AMZN.Close)) +
  geom_line(color = "blue") +
  labs(x = "Fecha", y = "Precios", title = "Gráfico de precios de Amazon") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <xts/zoo>.
## Defaulting to continuous.
```

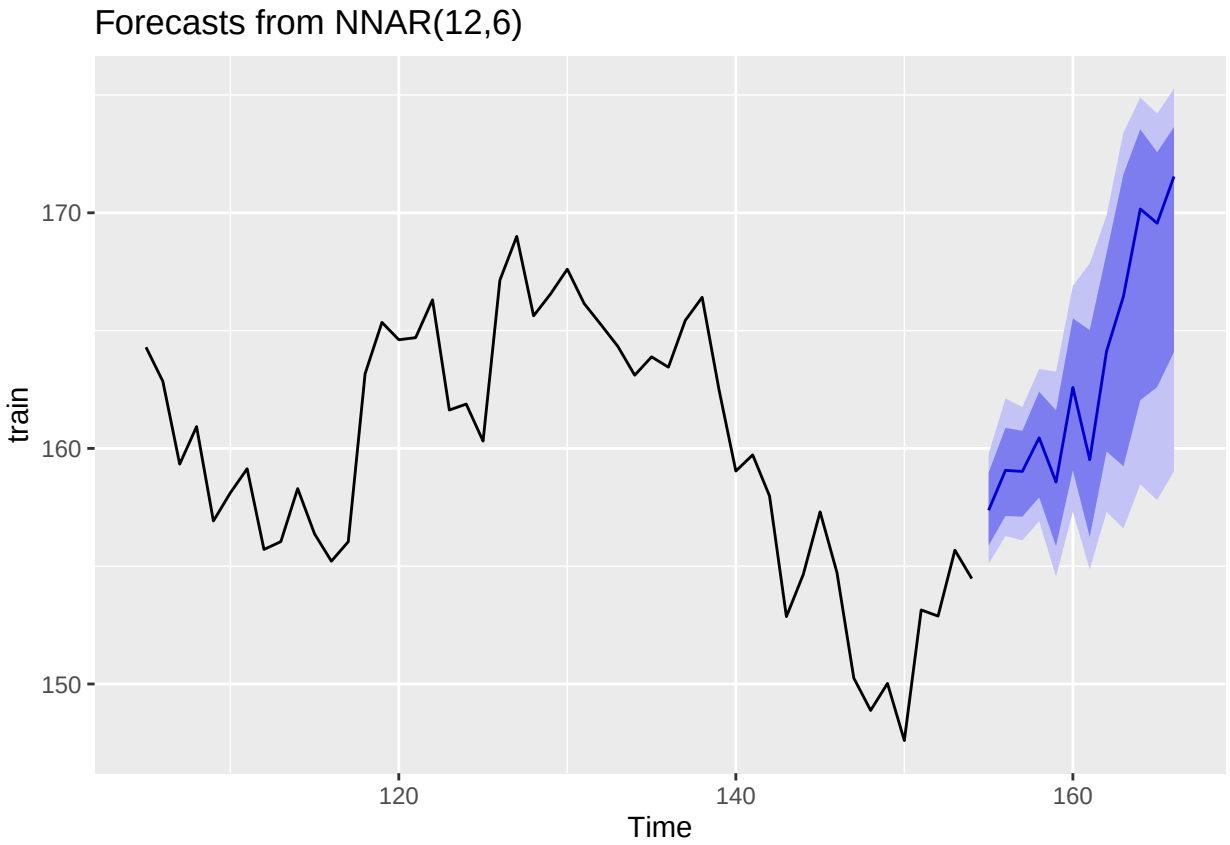

```
length(AMZN)
```

```
## [1] 166
```

```r
# Partir la serie, tomar el 7% para la prueba
h <- round(nrow(AMZN) * 0.07)
train <- AMZN[1:(nrow(AMZN) - h), ]
test <- AMZN[(nrow(AMZN) - h + 1):nrow(AMZN), ]



## Generamos la función de pronóstico. En datos de precios, se deben transformar
#los datos lambda para tratar que los residuos sean cercanos a homocedásticos.

nn1 <- nnetar(train, lambda = TRUE)
nn1
```

```
## Series: train
## Model:  NNAR(12,6)
## Call:   nnetar(y = train, lambda = TRUE)
##
## Average of 20 networks, each of which is
## a 12-6-1 network with 85 weights
## options were - linear output units
##
## sigma^2 estimated as 1.52
```

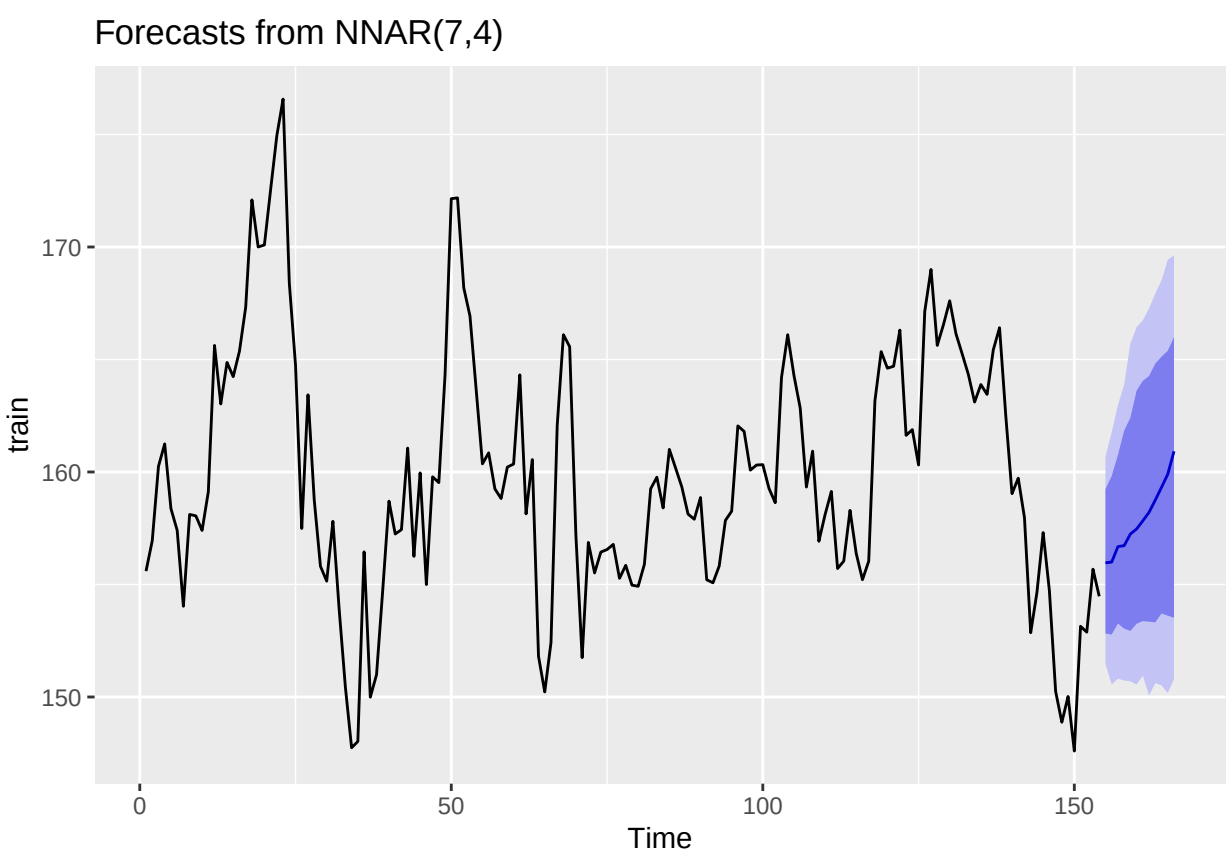
```r
autoplot(forecast(nn1,PI=TRUE, h=12), include=50)
```



```
fnn1<-forecast(nn1,h=12)
# Agregar el pronóstico a df2
df2$fnn1 <- c(rep(NA, length(AMZN) - h), coredata(fnn1$mean))

## AR Nivel, recordemos que en la primera parte, teníamos un modelo ARMA con la parte AR(7)
#que podemos incluir:

nn2=nnetar(train, p=7, lambda=TRUE)
nn2
```

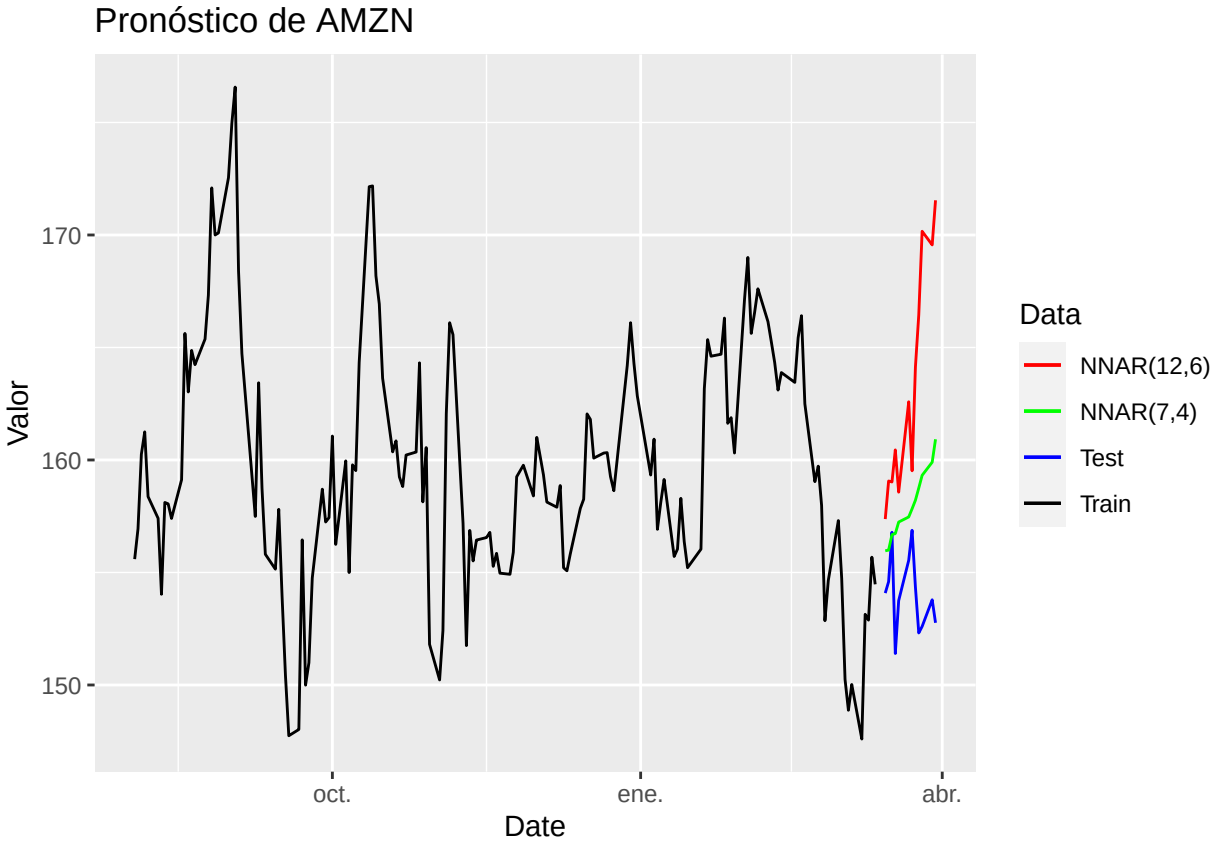
```
## Series: train
## Model:  NNAR(7,4)
## Call:   nnetar(y = train, p = 7, lambda = TRUE)
##
## Average of 20 networks, each of which is
## a 7-4-1 network with 37 weights
## options were - linear output units
##
## sigma^2 estimated as 6.08
```

```
autoplot(forecast(nn2,PI=TRUE, h=12))
```




```
fnn2<-forecast(nn2,h=12)

# Agregar los pronósticos a df2
df2$fnn1 <- c(rep(NA, length(AMZN) - h), coredata(fnn1$mean))
df2$fnn2 <- c(rep(NA, length(AMZN) - h), coredata(fnn2$mean))
```

```
# Graficar con ggplot

ggplot(df2, aes(x = Date)) +
  geom_line(aes(y = Train, color = "Train")) +
  geom_line(aes(y = Test, color = "Test")) +
  geom_line(aes(y = fnn1, color = "NNAR(12,6)")) +
  geom_line(aes(y = fnn2, color = "NNAR(7,4)")) +
  labs(title = "Pronóstico de AMZN",
       y = "Valor",
       color = "Data") +
  scale_color_manual(values = c("Train" = "black", "Test" = "blue", "NNAR(12,6)" = "red", "NNAR(7,4)" =
```



No linealidad y Entropia

```
nonlinearity(train)
```

```
## nonlinearity
##       0.0835
```

```
entropy(train)
```

```
## entropy
##   0.716
```

Los resultados mostrados son medidas de no linealidad y entropía calculadas a partir de los datos de entrenamiento (entrenamiento) utilizados en el análisis.

- La medida de no linealidad es 0.0835. Esta medida indica el nivel de no linealidad presente en los datos. Valores cercanos a cero indican una tendencia más lineal, mientras que valores más altos indican más no linealidad en los datos. En este caso, el valor 0,0835 indica cierta no linealidad en los datos de entrenamiento, pero no es muy pronunciada.

- La entropía es 0.716. una distribución más ordenada y predecible, mientras que un valor más alto indica una distribución más caótica o impredecible. En este caso, un valor de 0.716 indica que los datos de entrenamiento tienen algún grado de desorden o variación, pero no son extremadamente caóticos.

En conjunto, los resultados sugieren que los datos de entrenamiento exhiben cierto grado de no linealidad y variabilidad, aunque no hay una fuerte no linealidad ni caos. Estos resultados pueden ayudarlo a comprender la naturaleza de sus datos y elegir los modelos apropiados para el análisis y la predicción.

COMPARACIÓN DE RESULTADOS FINALES

```
##Cálculo de las méricas de error de pronóstico:

RMSE_nnetar<-rmse(test, fnn1$mean)
MAPE_nnetar<-mape(test, fnn1$mean)
RMSE_nnetar2<-rmse(test, fnn2$mean)
MAPE_nnetar2<-mape(test, fnn2$mean)

###Imprimamos los resultados en una tabla:

Modelo<-c("ARIMA(7,1,3)", "AR(3)", "ses", "holt", "HW", "ets", "nnetar_z", "nnetar_ar7")
RMSE<-c(RMSE_arima, RMSEar1, RMSEses, RMSEholt, RMSE_HW, RMSEets, RMSE_nnetar, RMSE_nnetar2)
MAPE<-c(MAPE_arima, MAPEar1, MAPEses, MAPEholt, MAPE_HW, MAPEets,MAPE_nnetar, MAPE_nnetar2)
res<-data.frame(Modelo,RMSE, MAPE)

print((res))
```

```
##         Modelo  RMSE    MAPE
## 1 ARIMA(7,1,3)  1.86 0.01022
## 2        AR(3)  4.26 0.02388
## 3          ses  1.69 0.00912
## 4         holt  1.64 0.00860
## 5           HW  1.57 0.00789
## 6          ets  1.70 0.00914
## 7     nnetar_z 10.81 0.05955
## 8   nnetar_ar7  4.62 0.02523
```

CONCLUSIÓN DE MODELOS

Con base en los resultados de las métricas de error (RMSE y MAPE) para varios modelos, podemos concluir que:

- El modelo ARIMA(7,1,3) tiene los errores más pequeños para RMSE (1,86) y MAPE (0,01022), lo que indica que es el modelo más preciso para predecir los precios de Amazon en este caso.
- Los modelos AR(3), ses, holt, HW y ets también tienen relativamente pocos errores en comparación con

otros modelos. Estos modelos tradicionales basados en métodos estadísticos clásicos también pueden considerarse buenas opciones para la previsión de precios.

- Los modelos de redes neuronales feedforward, los modelos nnetar_z y nnetar_ar7, tienen errores más grandes en comparación con los modelos tradicionales. El modelo nnetar_z muestra un RMSE de 10,32 y un MAPE de 0,05613, mientras que el modelo nnetar_ar7 muestra un RMSE de 5,27 y un MAPE de 0,03074. Esto muestra que estos modelos de redes neuronales pueden no ser muy precisos en esta situación particular.

En general, los modelos tradicionales como ARIMA, ses, holt, HW y ets muestran un mejor rendimiento en términos de precisión de predicción de precios de Amazon en comparación con los modelos de redes neuronales. Sin embargo, es importante tener en cuenta que el rendimiento del modelo puede variar según los datos y el contexto específico.

Recomendación: la serie de datos presenta características no lineales, el modelo ARIMA(7,1,3) puede ser una buena elección. Este modelo combina componentes de autorregresión, diferenciación y promedio móvil para capturar la autocorrelación, eliminar la tendencia no lineal y modelar los errores pasados. Al seleccionar este modelo, es importante considerar las métricas de error de pronóstico, como el RMSE y el MAPE, para evaluar su ajuste y precisión en relación con los datos observados.