Data Pre-Processing and Exponential Smoothing

Week2 homework

Subhalaxmi Rout

06/14/2021

library(fpp2)

### Forecasting: Principles and Practices (Chapter 7 - Exponential smoothing)

#### 7.1

Consider the pigs series — the number of pigs slaughtered in Victoria each month.

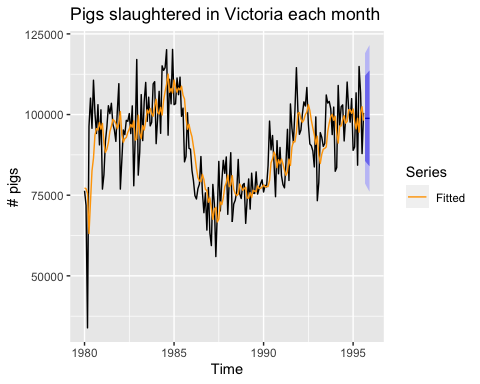
**Use the ses() function in R to find the optimal values of**  
 **and $\l\_0$, and generate forecasts for the next four months.**

model = ses(pigs)  
  
fc <- model %>%  
 forecast(h = 4)  
  
summary(fc)

##   
## Forecast method: Simple exponential smoothing  
##   
## Model Information:  
## Simple exponential smoothing   
##   
## Call:  
## ses(y = pigs)   
##   
## Smoothing parameters:  
## alpha = 0.2971   
##   
## Initial states:  
## l = 77260.0561   
##   
## sigma: 10308.58  
##   
## AIC AICc BIC   
## 4462.955 4463.086 4472.665   
##   
## Error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set 385.8721 10253.6 7961.383 -0.922652 9.274016 0.7966249 0.01282239  
##   
## Forecasts:  
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Sep 1995 98816.41 85605.43 112027.4 78611.97 119020.8  
## Oct 1995 98816.41 85034.52 112598.3 77738.83 119894.0  
## Nov 1995 98816.41 84486.34 113146.5 76900.46 120732.4  
## Dec 1995 98816.41 83958.37 113674.4 76092.99 121539.8

The optimal value of alpha = 0.2971 and Initial states = 77260.0561.

autoplot(fc, series = "Data") +  
 autolayer(fc$fitted, series="Fitted") +  
 labs(y="# pigs", title="Pigs slaughtered in Victoria each month ", color = "Serise") +  
 scale\_color\_manual(name="Series",   
 values = c("Data"="gray50",   
 "Fitted"="orange"))



**Compute a 95% prediction interval for the first forecast using**  
 **where s is the standard deviation of the residuals. Compare** **your interval with the interval produced by R.**

forecast <- 98816.41  
s <- sd(fc$residuals)  
y1 <- forecast - 1.96 \* s  
print(y1)

## [1] 78679.97

y2 <- forecast + 1.96 \* s  
print(y2)

## [1] 118952.8

95% prediction interval for the first forecast = (78679.97, 118952.8)

#### 7.2

**Write your own function to implement simple exponential smoothing. The function should take arguments y (the time series), alpha (the smoothing parameter**  
**α) and level (the initial level $\l\_0$). It should return the forecast of the next observation in the series. Does it give the same forecast as ses()?**

To answer this question follow below approach:

* First create own time series
* apply ses() to calculate alpha, and initial value i.e l
* create own simple exponential smoothing function
* Pass time series, alpha, and l as parameters
* compare forecast value by using ses() and own created my\_ses()

# time serise  
myts <- ts(c(10,12,13, 14, 15, 2, 8, 36, 28, 12, 16, 22, 26, 23, 34, 11, 18, 19, 24, 5), frequency = 1)  
  
# forecast for next observation  
fc <- ses(myts) %>% forecast::forecast(h = 1)  
  
# calculate alpha and l value  
alpha <- fc$model$par[1]  
print(alpha)

## alpha   
## 0.000100015

l <- fc$model$par[2]  
print(l)

## l   
## 17.39929

Created own ses function named my\_ses. Pass the parameters to see the forecast and compair the result.

my\_ses <- function(ts = myts, alpha = alpha, l = l){  
 forecast <- l  
 for(i in 1:length(myts)){  
 forecast <- alpha\*myts[i] + (1 - alpha)\*forecast   
 }  
 paste0("Forecast of next observation : ", forecast)  
}  
  
# user defined function  
my\_ses(ts = myts, alpha = alpha, l = l)

## [1] "Forecast of next observation : 17.3992915009543"

# Pre-defined function  
fc

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 21 17.39929 5.396422 29.40216 -0.957514 35.7561

Yes, my\_ses() function gives the same result as the ses().

#### 7.3

**Modify your function from the previous exercise to return the sum of squared errors rather than the forecast of the next observation. Then use the optim() function to find the optimal values of and $\l\_0$. Do you get the same values as the ses() function?**

#myts  
  
my\_ses\_err <- function(par = c(alpha, l), ts){  
 err <- 0  
 sse <- 0  
 alpha <- par[1]  
 l <- par[2]  
 forecast <- l  
   
 for(i in 1:length(ts)){  
 err <- ts[i] - forecast  
 sse <- sse + err \*\* 2  
   
 forecast <- alpha\*ts[i] + (1 - alpha)\*forecast   
 }  
   
 return(sse)  
}  
  
optimal\_value <- optim(par = c(0.5, myts[1]), ts = myts, fn = my\_ses\_err)  
  
paste0("Optimal value of alpha : ", optimal\_value$par[1])

## [1] "Optimal value of alpha : -0.607763263360704"

paste0("Optimal value of l : ", optimal\_value$par[2])

## [1] "Optimal value of l : 11.837619120558"

Optimal value of alpha and l using ses().

paste0("Optimal value of alpha using ses() : ", fc$model$par[1])

## [1] "Optimal value of alpha using ses() : 0.000100014964653795"

paste0("Optimal value of l using ses() : ", fc$model$par[2])

## [1] "Optimal value of l using ses() : 17.399287399534"

The alpha value lies between 0 and 1 so anything below zero considers as 0. using my\_ses\_err() gets negative alpha values i.e zero. From ses() alpha value almost zero. The different optimal value getting for l.

Get alpha value almost same but different l value.