DATA624 - Homework 2

Esteban Aramayo

2022-06-10

# Week 2: Data Pre-processing & Exponential Smoothing [ Jun 6 - Jun 12]

## Exercise 7.8.1 (aka 7.1)

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

1. Use the () function in R to find the optimal values of and , and generate forecasts for the next four months.

### Solution

Below is the forecast for the next four months using simple exponential smoothing (ses) forecasts applied to pigs time series.

library(fpp2)  
library(forecast)  
  
fc <- ses(y = pigs, h = 4)  
  
fc

#> 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

Extract the optimal values of and from the model component of the previously calculated forecast.

optimal\_alpha <- fc$model$par[1]  
optimal\_l0 <- fc$model$par[2]  
  
print(optimal\_alpha)

#> alpha   
#> 0.2971488

print(optimal\_l0)

#> l   
#> 77260.06

The optimal values are: = **0.2971488** and = **77260.06**.

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

### Solution

Below we can see that our first forecast corresponds to September 1995.

print(fc)

#> 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

Calculate the standard deviation of the residuals

s <- sd( fc$residuals )  
  
print(s)

#> [1] 10273.69

Calculate the Lower and Upper Confidence values for a 95% prediction interval using the residuals of the forecast and using the results from the model.

# residuals results  
Lo95 <- fc$mean[1] - 1.96 \* s  
Hi95 <- fc$mean[1] + 1.96 \* s  
  
# model results  
R.Lo95 <- fc$lower[1,2]   
R.Hi95 <- fc$upper[1,2]

When we compare our calculated values vs the values produced by R we see that the values obtained by both methods are close but not the same. The lower value is higher for the residuals method, while the upper value is smaller for the residuals method.

| Calculation Method | Lower 95 | Upper 95 |
| --- | --- | --- |
| Residuals based () | 78679.97 | 118952.84 |
| R model based | 78611.97 | 119020.84 |

## Exercise 7.8.2 (aka 7.2)

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

### Solution

Let’s create our own version of the () function using the formula for weighted average form

myses <- function(y, alpha, level) {  
   
 # set initial estimated y with level  
 y\_hat <- level  
   
 # traverse elements of series  
 for(i in 1:length(y)) {  
   
 # calculate the next estimated y  
 y\_hat <- alpha \* y[i] + (1 - alpha) \* y\_hat  
   
 }  
   
 return(y\_hat)  
   
}

Now let’s see if the forecast of the next observation in the series returned by our () function matches the value returned by the () function.

fc\_myses <- myses(y = pigs, alpha = optimal\_alpha, level = optimal\_l0)  
  
fc\_ses <- ses(y = pigs, h = 4)  
  
print(fc\_myses)

#> alpha   
#> 98816.41

print(fc\_ses$mean[1])

#> [1] 98816.41

When comparing the forecast results for the next observation we can see that the two calculation methods yield very close results.

| Calculation Method | Forecast value |
| --- | --- |
| myses() | 98816.41 |
| ses() | 98816.41 |

## Exercise 7.8.3 (aka 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 () function to find the optimal values of and . Do you get the same values as the () function?

### Solution

Based on the () function, create a function to return the Sum of Squared Errors (SSE).

mySSE <- function( pars = c(alpha, level), y ) {  
   
 # unpack pars array to get alpha and level values  
 alpha <- pars[1]  
 level <- pars[2]  
   
 # set initial estimated y with level  
 y\_hat <- level  
   
 err <- 0  
   
 SSE <- 0  
  
 # traverse elements of series  
 for(i in 1:length(y)) {  
   
 # calculate error by subtracting estimated y from actual y  
 err <- y[i] - y\_hat  
   
 # sum up and accumulate squared errors  
 SSE <- SSE + err ^ 2  
   
 # calculate the next estimated y  
 y\_hat <- alpha \* y[i] + (1 - alpha) \* y\_hat  
  
 }  
   
 return(SSE)  
   
}

Let’s use our () function to calculate the optimum values and compare them to the values from R’s () function.

result\_mySSE <- optim( par = c(0.5, pigs[1]), y = pigs, fn = mySSE )  
  
  
mySSE\_optimal\_alpha <- result\_mySSE$par[1]  
mySSE\_optimal\_l0 <- result\_mySSE$par[2]  
  
ses\_optimal\_alpha <- fc\_ses$model$par[1]  
ses\_optimal\_l0 <- fc\_ses$model$par[2]

When comparing the results of both methods, we see that the values are very close. The Optimal for the () function is slightly bigger than that of the () function. While the Optimal for the () function is slightly smaller than that of the () function.

| Calculation Method | Optimal | Optimal |
| --- | --- | --- |
| () function | 0.2990081 | 76379.27 |
| R () function | 0.2971488 | 77260.06 |