Lab2_ExponentialSmoothingwithR_a24kimwu

2025-09-08

Loading packages & data

```
#Load necessary libraries
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tsutils)
```

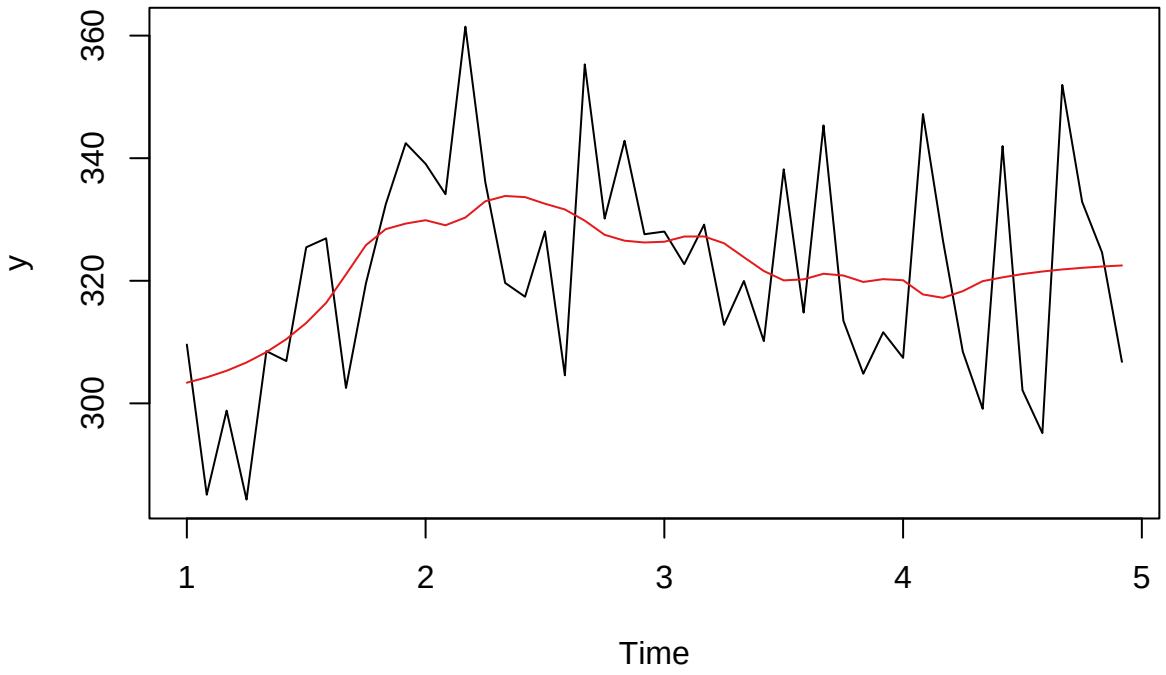
```
#Load data from csv file
Y<-read.csv("./workshop1R.csv")

#Pick the first time series for modelling
y<-Y[,1]

#Transform it into a time series
y<-ts(y,frequency=12)
```

Constructing estimation & hold-out sets

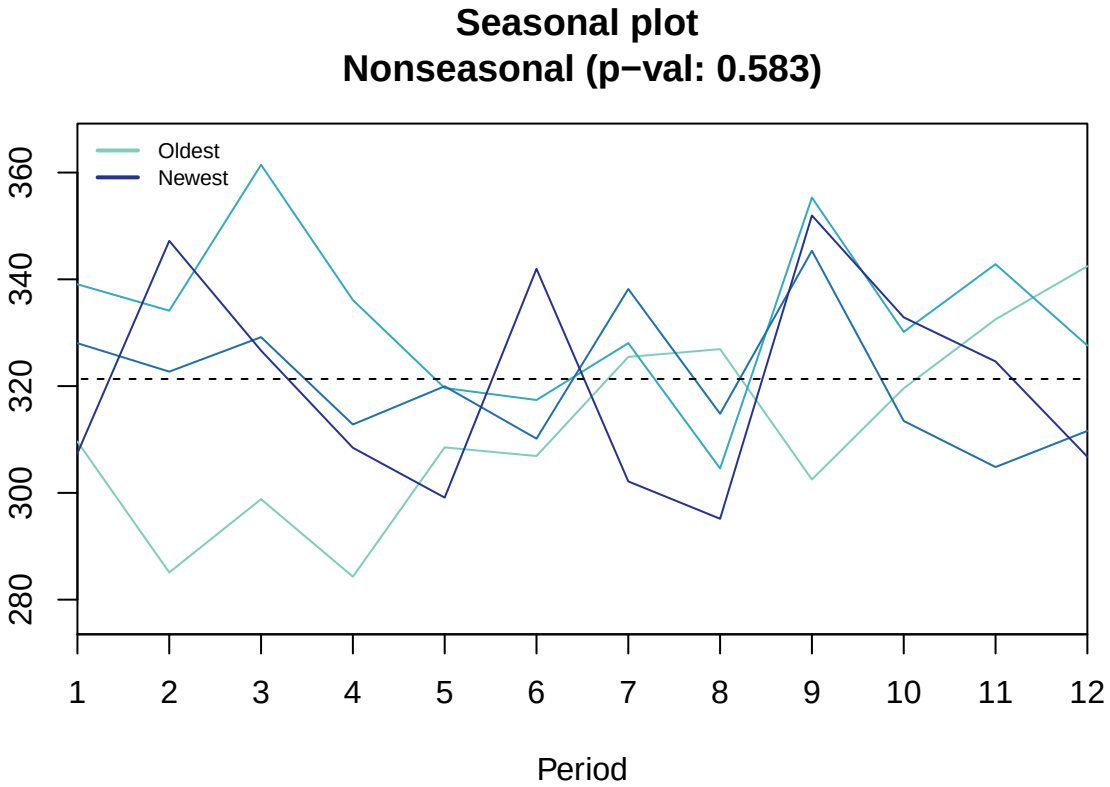
```
y.tst <- tail(y,12)
y.trn <- head(y,48)
```

Exploring a time series

```
cma <- cmav(y.trn,outplot=1) # The argument outplot produces a plot
```




```
seasplot(y.trn)
```




```
## Results of statistical testing
## Evidence of trend: FALSE  (pval: 0.154)
## Evidence of seasonality: FALSE  (pval: 0.583)
```

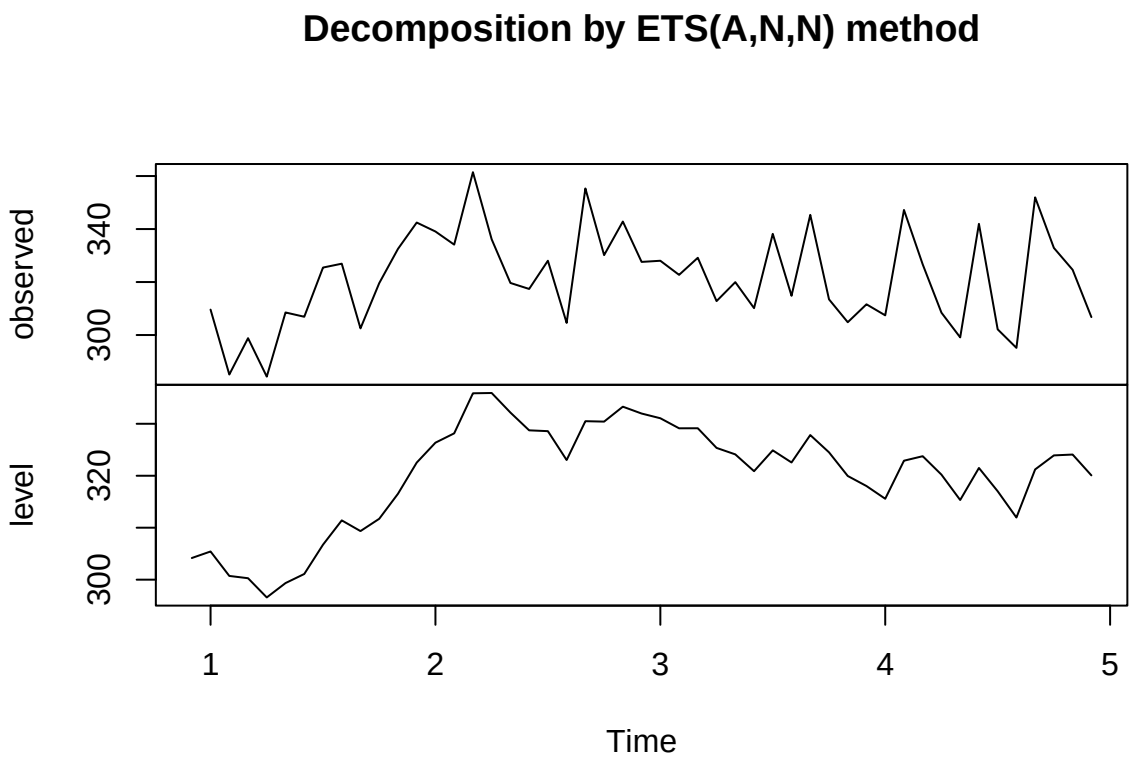
Forecasting

1. Model fitting

```
fit1 <- ets(y.trn,model="ANN")
print(fit1)
```

```
## ETS(A,N,N)
##
## Call:
## ets(y = y.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.2315
##
##   Initial states:
##     l = 304.1632
##
##   sigma:  17.7462
##
##      AIC     AICc      BIC
## 465.8872 466.4326 471.5008
```

```
plot(fit1)
```




```
# names gives you the variables in the list fit1, an object ets (the result of class) is a list.
names(fit1)
```

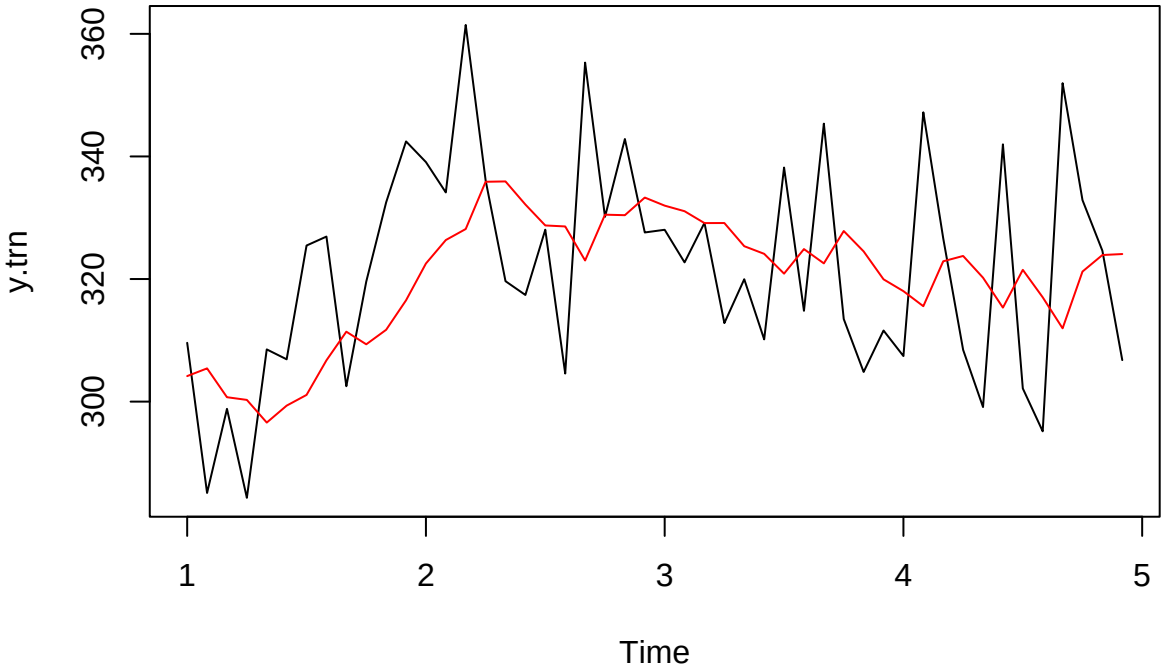
```
##  [1] "loglik"     "aic"        "bic"        "aicc"       "mse"
##  [6] "amse"       "fit"        "residuals"  "fitted"     "states"
## [11] "par"        "m"          "method"     "series"     "components"
```

```
## [16] "call"       "initstate"  "sigma2"     "x"
```

```
 fit1$fitted
```

```
##        Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1 304.1632 305.4201 300.7152 300.2764 296.5785 299.3423 301.0918 306.7338
## 2 322.5299 326.3602 328.1574 335.8658 335.9240 332.1556 328.7385 328.5782
## 3 331.9730 331.0612 329.1279 329.1359 325.3560 324.1078 320.8773 324.8847
## 4 318.0192 315.5674 322.8906 323.7610 320.2151 315.3298 321.4977 317.0179
##        Sep      Oct      Nov      Dec
## 1 311.4076 309.3478 311.7161 316.5288
## 2 323.0216 330.4963 330.4161 333.2917
## 3 322.5529 327.8304 324.5061 319.9523
## 4 311.9554 321.2157 323.9146 324.0768
```

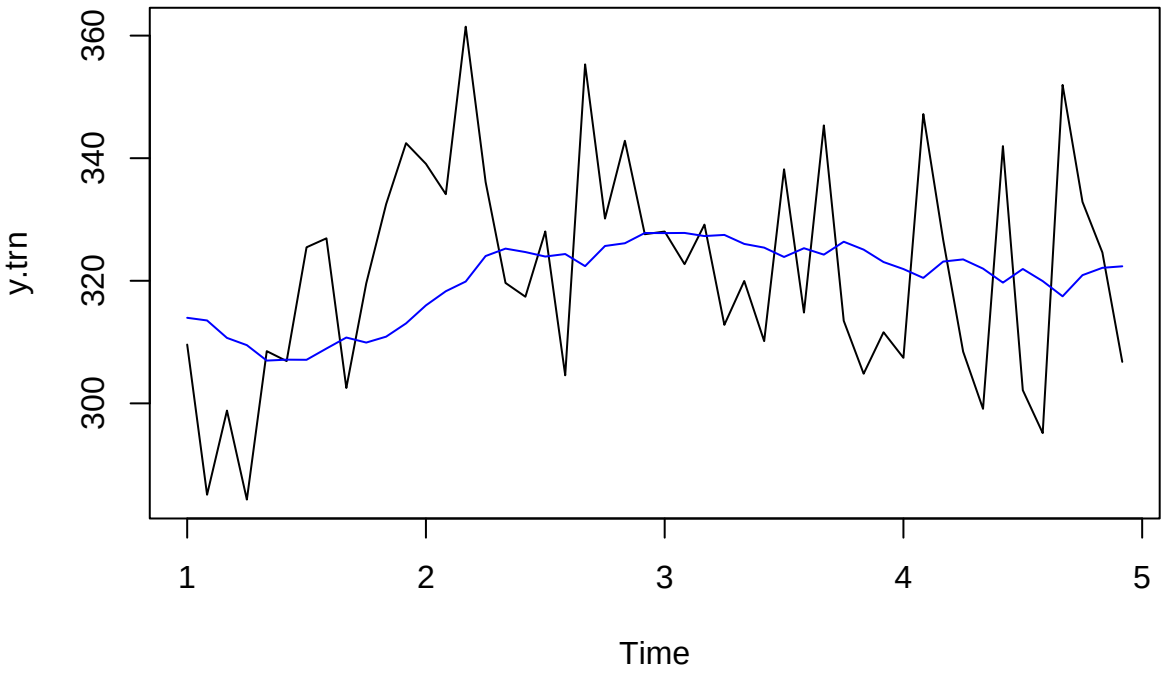
```
plot(y.trn)
lines(fit1$fitted, col="red") # col=... specifies the colour of the line
```



Question: Given your understanding of single exponential smoothing, is this a good model fit? Does the forecast look "smooth"? What should we do with the alpha parameter?

Answer: The model is a good fit but it could be better. The forecast still has a lot of noise and does not represent a smooth curve. In order to achieve a smoother curve we should use a smaller alpha parameter.

```
fit2 <- ets(y.trn, model = "ANN", alpha = 0.1)
plot(y.trn)
lines(fit2$fitted,col="blue")
```




```
fit1$mse
```

```
## [1] 301.8053
```

```
fit2$mse
```

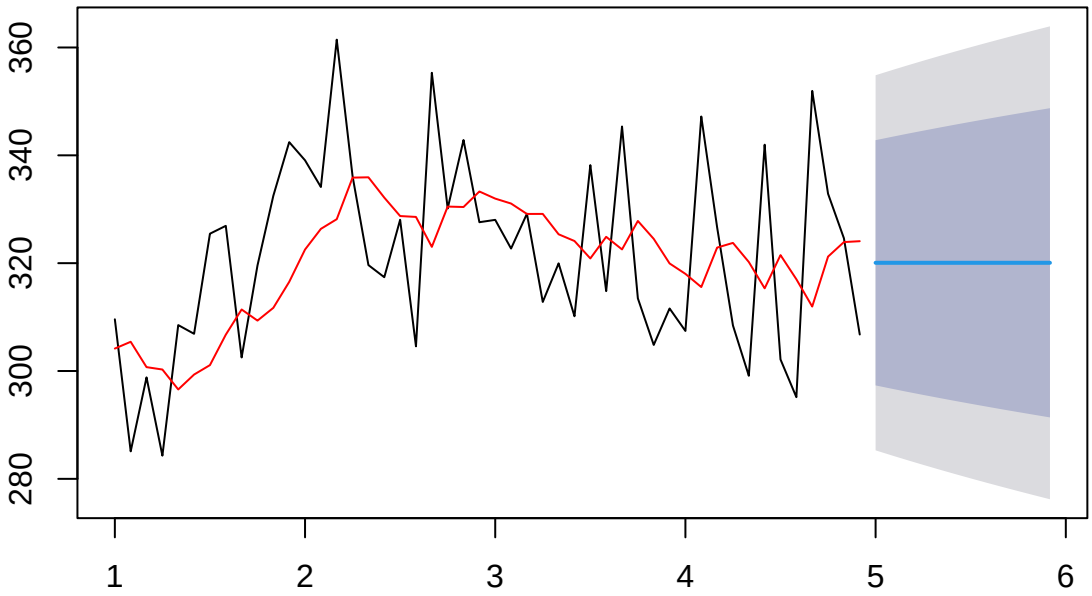
```
## [1] 313.4249
```

2. Forecasting

```
frc1 <- forecast(fit1, h=12)
print(frc1)
```

```
##       Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 5       320.0694 297.3267 342.8121 285.2875 354.8513
## Feb 5       320.0694 296.7253 343.4135 284.3676 355.7712
## Mar 5       320.0694 296.1389 343.9999 283.4708 356.6679
## Apr 5       320.0694 295.5666 344.5722 282.5955 357.5433
## May 5       320.0694 295.0073 345.1315 281.7402 358.3986
## Jun 5       320.0694 294.4602 345.6786 280.9035 359.2353
## Jul 5       320.0694 293.9246 346.2142 280.0844 360.0544
## Aug 5       320.0694 293.3997 346.7391 279.2817 360.8571
## Sep 5       320.0694 292.8850 347.2538 278.4945 361.6443
## Oct 5       320.0694 292.3798 347.7590 277.7219 362.4169
## Nov 5       320.0694 291.8837 348.2551 276.9631 363.1757
## Dec 5       320.0694 291.3962 348.7426 276.2175 363.9213
```

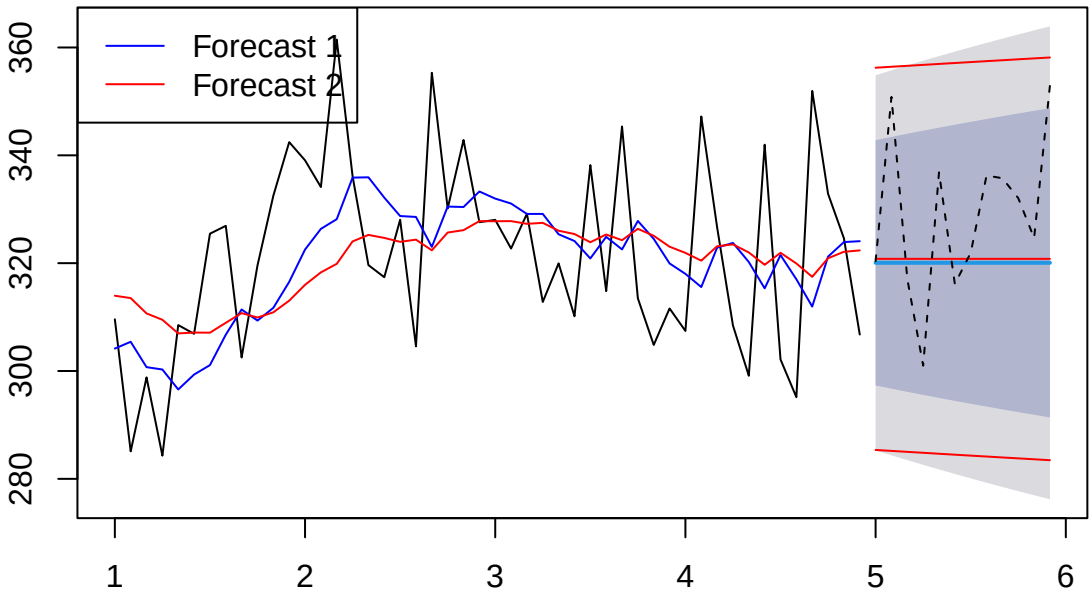
```
plot(frc1)
lines(fit1$fitted,col="red")
```




```
names(frc1)
```

```
##  [1] "model"     "mean"      "level"     "x"         "upper"     "lower"
##  [7] "fitted"    "method"    "series"    "residuals"
```

```
frc2 <- forecast(fit2,h=12) # Store the forecasts
plot(frc1)
lines(fit1$fitted,col="blue")
lines(frc2$mean,col="red")
lines(fit2$fitted,col="red")
lines(frc2$lower[,2],col="red") # 95% lower
lines(frc2$upper[,2],col="red") # 95% upper
lines(y.tst,lty=2) # lty=2 gives us a dashed line.

# Add legend to the plot
legend("topleft",c("Forecast 1","Forecast 2"),col=c("blue","red"),lty=1)
```



Question: Which is the best forecast?

Answer: Forecast number 2 is the better forecast as it captures the noise better than forecast 1.

```
MAE1 <- mean(abs(y.tst- frc1$mean))
MAE2 <- mean(abs(y.tst- frc2$mean))
MAE <- c(MAE1, MAE2) # Collect them in a single vector
names(MAE) <- paste0("Forecast ",1:2) # Name the errors
round(MAE,3) # round to 3rd decimal point
```

```
## Forecast 1 Forecast 2
##     13.087     12.794
```

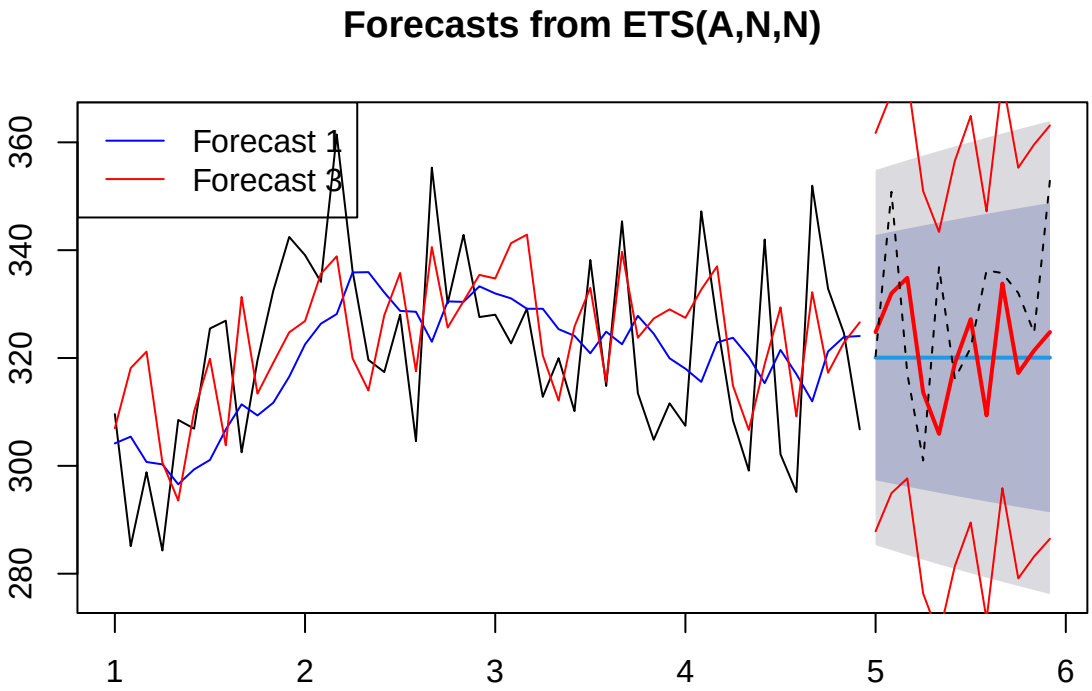
3. Model selection

```
fit3 <- ets(y.trn, model= "AAA", damped=TRUE)
frc3 <- forecast(fit3,h=12)
print(fit3)
```

```
## ETS(A,Ad,A)
##
## Call:
## ets(y = y.trn, model = "AAA", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.0833
##     beta  = 1e-04
##     gamma = 0.0082
##     phi   = 0.8651
##
##   Initial states:
##     l = 300.0511
##     b = 4.7163
```

```
##     s = 2.9611 -0.7194 -4.9892 11.6525 -12.6768 5.3307
##           -3.0213 -16.3023 -8.3262 13.0008 10.2468 2.8432
##
##   sigma:  18.8453
##
##      AIC     AICc      BIC
## 482.7128 506.2990 516.3944
```

```
plot(frc1)
lines(fit1$fitted,col="blue")
lines(frc3$mean,col="red",lwd=2) # lwd=2 makes the line thicker
lines(fit3$fitted,col="red")
lines(frc3$upper[,2],col="red")
lines(frc3$lower[,2],col="red")
lines(y.tst,lty=2)
legend("topleft",c("Forecast 1","Forecast 3"),col=c("blue","red"),lty=1)
```



Question: What do you expect to be the performance of this forecast? Better or worse than the other two options? Remember the time series does not contain either trend or seasonality.

Answer: I expect this model to perform worse than the ANN model since this is a Level time series that does not contain trend or seasonality.

```
MAE3 <- mean(abs(y.tst- frc3$mean))
round(MAE3,3)
```

```
## [1] 13.99
```

```
# The erros from before were:
round(MAE,3)
```

```
## Forecast 1 Forecast 2
##     13.087     12.794
```

```
MSE1<-mean((y.tst-frc1$mean)^2)
```

```
MSE2<-mean((y.tst-frc2$mean)^2)
MSE3<-mean((y.tst-frc3$mean)^2)
MSE<-c(MSE1,MSE2,MSE3) #Collect them in asingle vector
RMSE<-sqrt(MSE) #We calculate the Root Mean Squared Error
names(RMSE)<-paste0("Forecast",1:3) #Name the errors
round(RMSE,3) #round to 3rd decimal point
```

```
## Forecast1 Forecast2 Forecast3
##    16.770    16.399    17.306
```

```
# Good practice is to name your array dimensions. We can do that using the dimnames,which accepts a lis

crit<-array(NA,c(3,3),dimnames=list(c("Forecast1","Forecast2","Forecast3"),
c("AIC","AICc","BIC"))) #Icansplitlines!
print(crit)
```

```
##           AIC AICc BIC
## Forecast1  NA   NA  NA
## Forecast2  NA   NA  NA
## Forecast3  NA   NA  NA
```

```
#Allvaluesfromfit1-1strow
 crit[1,1]<-fit1$aic #1stcolumn
 crit[1,2]<-fit1$aicc #2ndcolumn
 crit[1,3]<-fit1$bic #3rdcolumn

#Allvaluesforfit2-2ndrow
 crit[2,1]<-fit2$aic #1stcolumn
 crit[2,2]<-fit2$aicc #2ndcolumn
 crit[2,3]<-fit2$bic #3rdcolumn

#Allvaluesforfit3-3rdrow
 crit[3,1]<-fit3$aic #1stcolumn
 crit[3,2]<-fit3$aicc #2ndcolumn
 crit[3,3]<-fit3$bic #3rdcolumn

#Andtheresultis...
 print(crit)
```

```
##                AIC     AICc      BIC
## Forecast1 465.8872 466.4326 471.5008
## Forecast2 465.7005 465.9672 469.4429
## Forecast3 482.7128 506.2990 516.3944
```

```
 fit4 <- ets(y.trn)
 print(fit4)
```

```
## ETS(A,N,N)
##
## Call:
## ets(y = y.trn)
##
##   Smoothing parameters:
##     alpha = 0.2315
##
##   Initial states:
```

```
##     l = 304.1632
##
##   sigma:  17.7462
##
##      AIC     AICc      BIC
## 465.8872 466.4326 471.5008
```

One important thing to remember: the smoothing parameters are in the state-space formulation. This means that: • The alpha value is as you know it • The beta value is alpha*beta', where beta' is the beta as you know it • The gamma value is (1-alpha)*gamma', where gamma' is the gamma as you know it

In practice, the beta you put in the model is much larger than the typical beta we are used to (as the typical beta is multiplied by alpha <1, so it gets smaller). Therefore, do not be surprised by very small beta values. The gamma is again scaled by (1-alpha), so it will be smaller than the gamma in the format we have in the slides.

Exercises

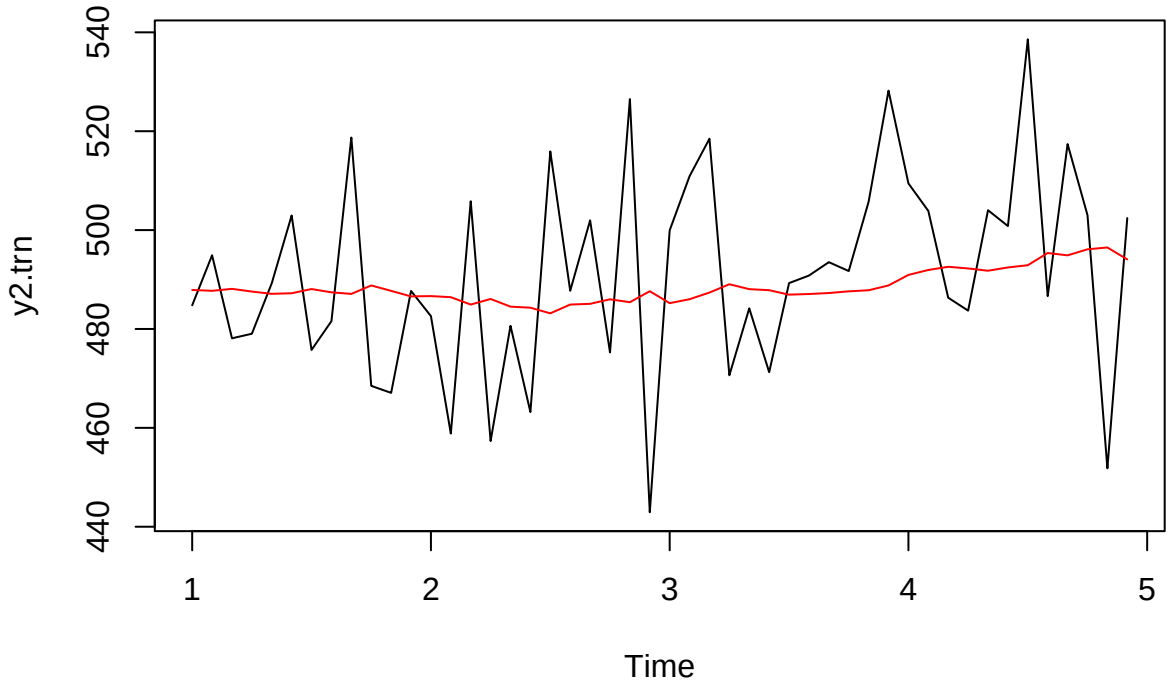
1. Level B

```r
#Transform it into a time series
y2<-ts(Y[,2],frequency=12)

y2.tst <- tail(y2,12)
y2.trn <- head(y2,48)
```

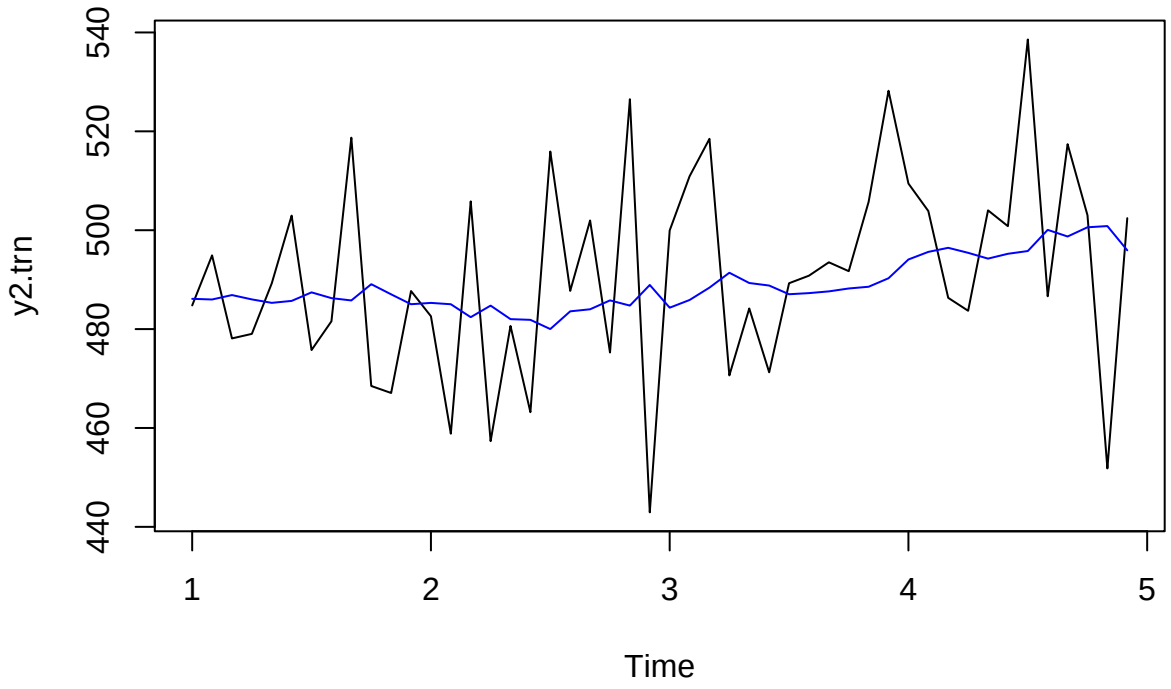
```r
fit1_2 <- ets(y2.trn,model="ANN")
print(fit1_2)
```

```
## ETS(A,N,N)
##
## Call:
## ets(y = y2.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.0541
##
##   Initial states:
##     l = 487.8891
##
##   sigma:  20.9366
##
##      AIC     AICc      BIC
## 481.7588 482.3043 487.3724
```

```r
plot(y2.trn)
lines(fit1_2$fitted, col="red") # col=... specifies the colour of the line
```




```
fit2_2 <- ets(y2.trn, model = "ANN", alpha = 0.1)
plot(y2.trn)
lines(fit2_2$fitted,col="blue")
```



Question: Automatic vs. Manual parameters - Which one is best using your judgement?

Answer: I would say that the automatic parameter performs best since it appears to be smoother and on one level, which makes sense for a leveled time series.

```
fit1_2$mse
```

```
## [1] 420.0784
```

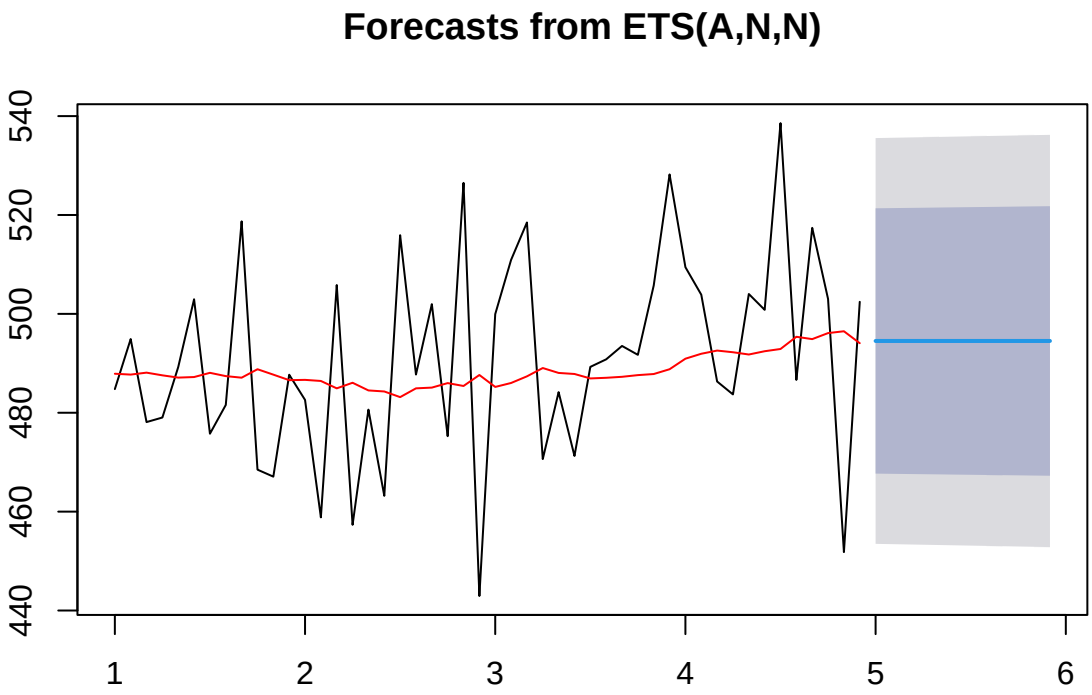
```
fit2_2$mse
```

```
## [1] 424.1008
```

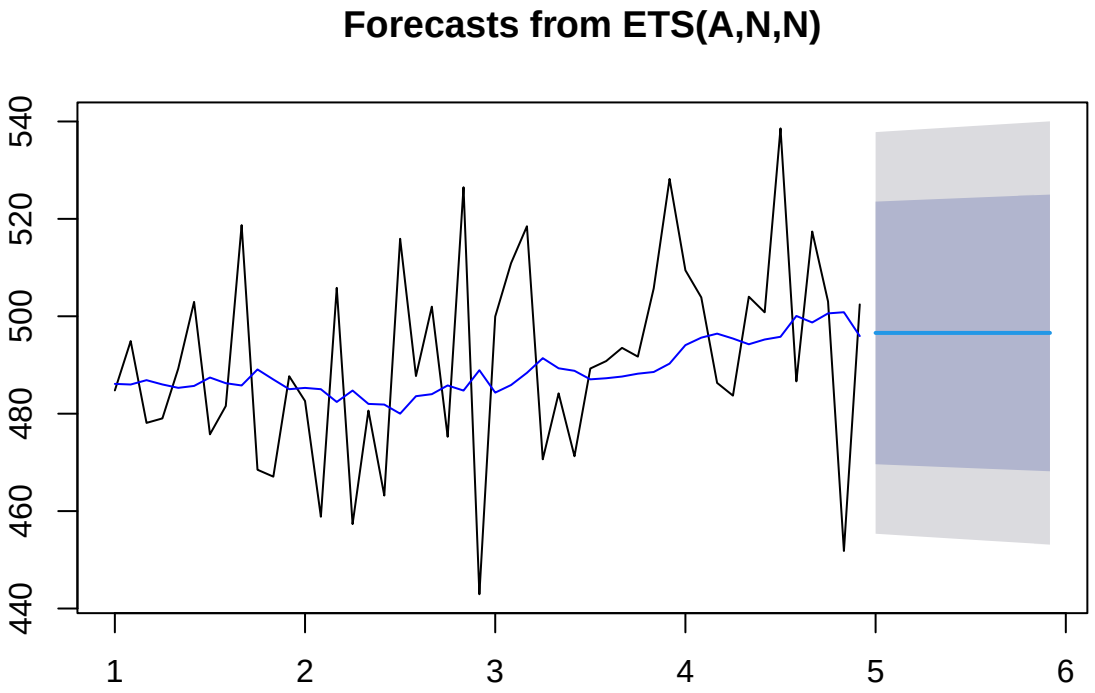
Question: Automatic vs. Manual parameters - Which one is best using errors?

Answer: The MSE suggests that the automatic parameters perform better as well.

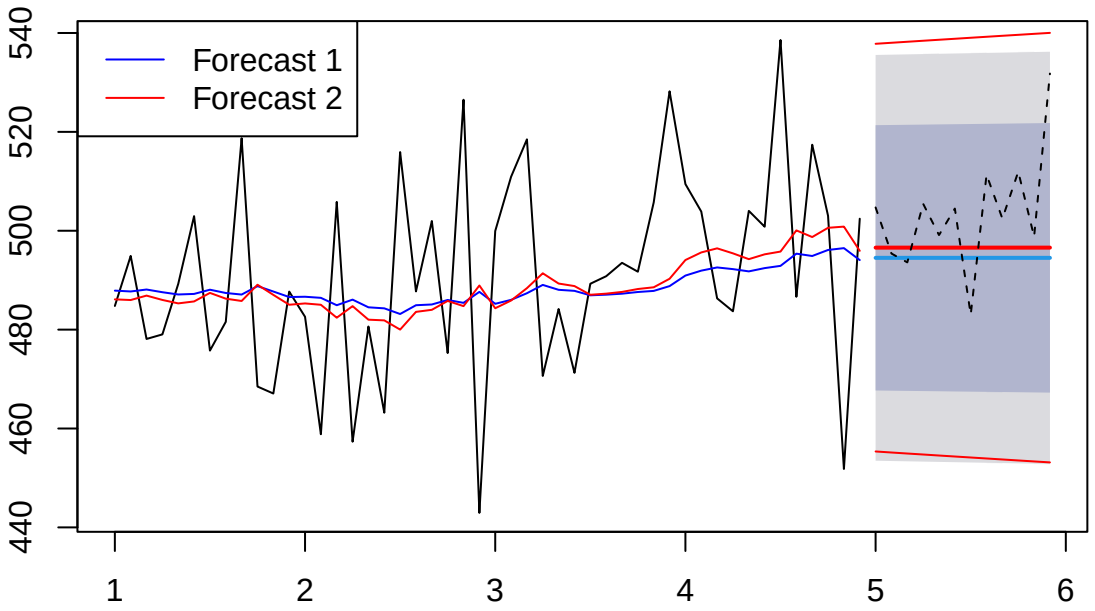
```
frc1_2 <- forecast(fit1_2, h=12)
plot(frc1_2)
lines(fit1_2$fitted,col="red")
```




```
frc2_2 <- forecast(fit2_2, h=12)
plot(frc2_2)
lines(fit2_2$fitted,col="blue")
```




```
plot(frc1_2)
 lines(fit1_2$fitted,col="blue")
 lines(frc2_2$mean,col="red",lwd=2) # lwd=2 makes the line thicker
 lines(fit2_2$fitted,col="red")
 lines(frc2_2$upper[,2],col="red")
 lines(frc2_2$lower[,2],col="red")
 lines(y2.tst,lty=2)
 legend("topleft",c("Forecast 1","Forecast 2"),col=c("blue","red"),lty=1)
```




```
MAE1_2 <- mean(abs(y2.tst- frc1_2$mean))
```

```
MAE2_2 <- mean(abs(y2.tst- frc2_2$mean))
MAE_2 <- c(MAE1_2, MAE2_2) # Collect them in a single vector
names(MAE_2) <- paste0("Forecast ",1:2) # Name the errors
round(MAE_2,3) # round to 3rd decimal point
```

```
## Forecast 1 Forecast 2
##     11.058      9.881
```

```
MSE1_2<-mean((y2.tst-frc1_2$mean)^2)
MSE2_2<-mean((y2.tst-frc2_2$mean)^2)
MSE_2<-c(MSE1_2,MSE2_2) #Collect them in asingle vector
RMSE_2<-sqrt(MSE_2) #We calculate the Root Mean Squared Error
names(RMSE_2)<-paste0("Forecast",1:2) #Name the errors
round(RMSE_2,3) #round to 3rd decimal point
```

```
## Forecast1 Forecast2
##    14.534    13.357
```

Question: Does the selected model perform best in the out-of-sample data?

Answer: The MAE & MSE on the other hand suggest that the second model which was trained with manual parameters works better for forecasting.

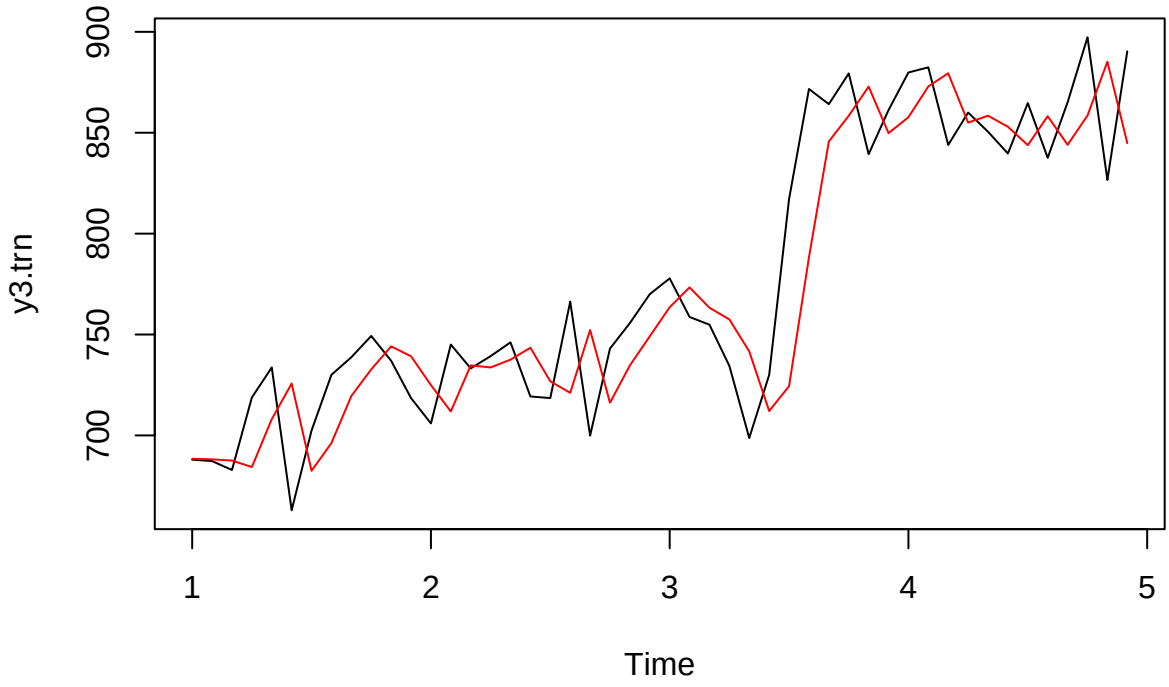
2. Level Shift

```
#Transform it into a time series
y3<-ts(Y[,3],frequency=12)

y3.tst <- tail(y3,12)
y3.trn <- head(y3,48)
```

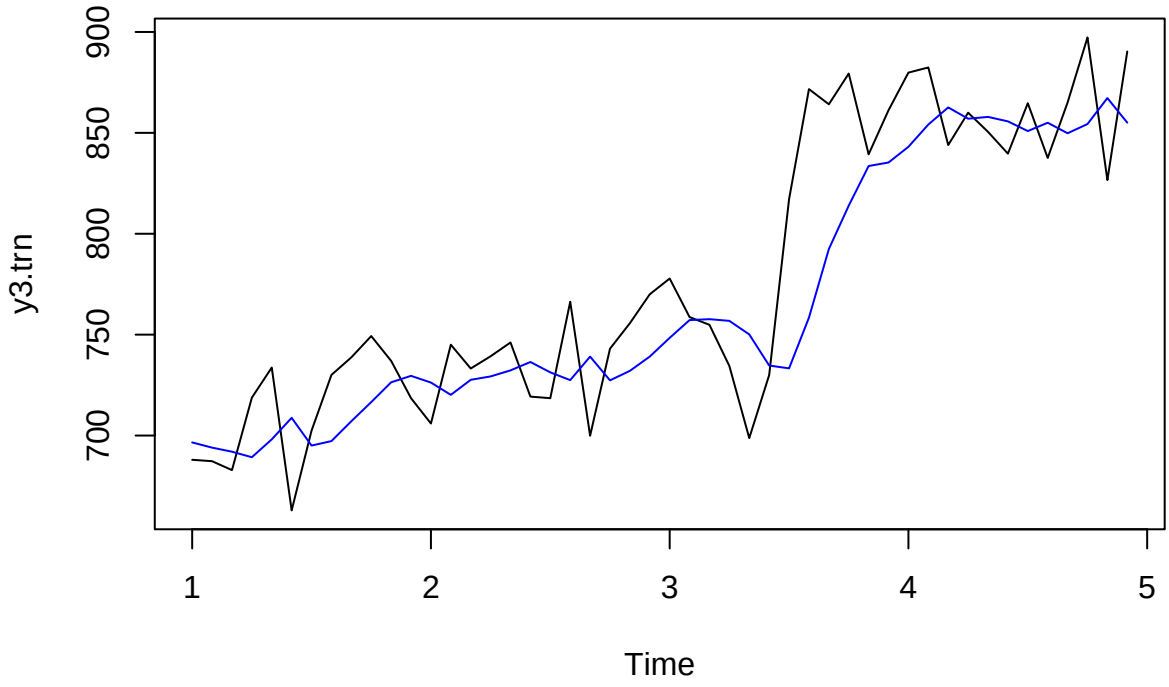
```
fit1_3 <- ets(y3.trn,model="ANN")
print(fit1_3)
```

```
## ETS(A,N,N)
##
## Call:
## ets(y = y3.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.6886
##
##   Initial states:
##     l = 688.4287
##
##   sigma:  32.3896
##
##      AIC     AICc      BIC
## 523.6472 524.1926 529.2608
```

```
plot(y3.trn)
lines(fit1_3$fitted, col="red") # col=... specifies the colour of the line
```




```
fit2_3 <- ets(y3.trn, model = "ANN", alpha = 0.3)
plot(y3.trn)
lines(fit2_3$fitted,col="blue")
```



Question: Automatic vs. Manual parameters - Which one is best using your judgement?

Answer: The model using automatic parameters seems to follow the level shift better than the one trained with manual parameters.

```
fit1_3$mse
```

```
## [1] 1005.374
```

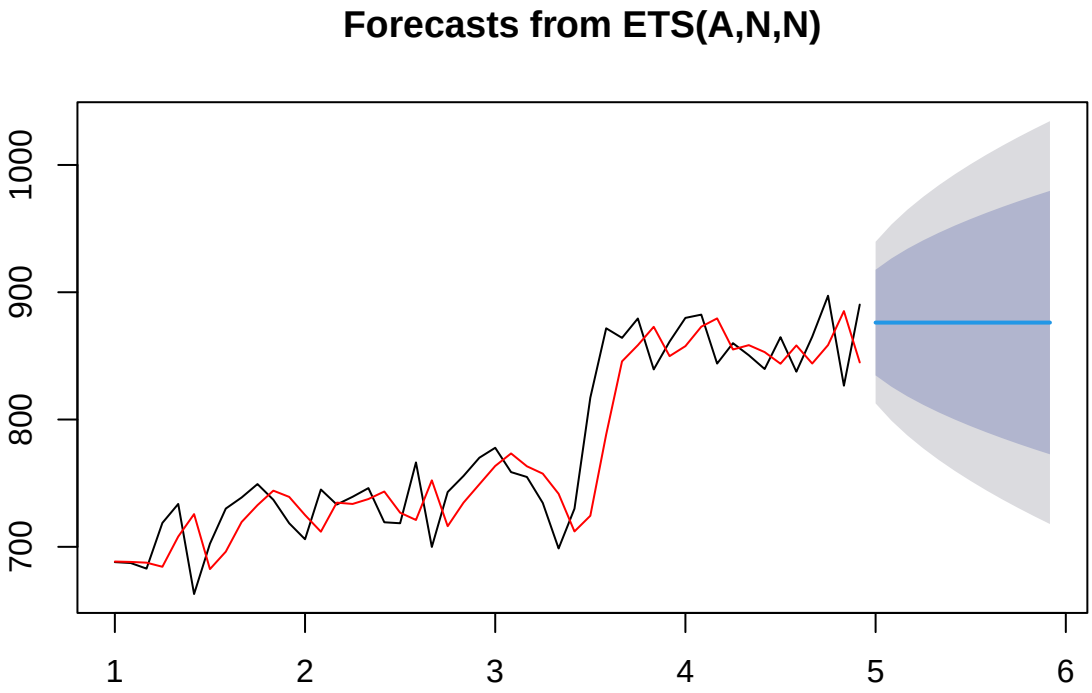
```
fit2_3$mse
```

```
## [1] 1183.48
```

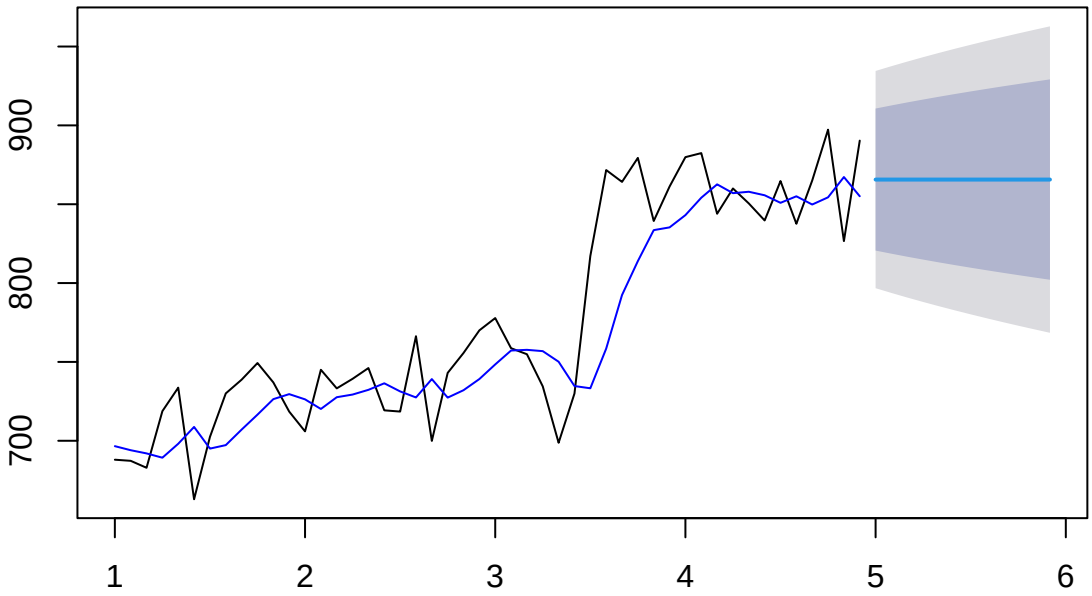
Question: Automatic vs. Manual parameters - Which one is best using errors?

Answer: The MSE suggests that the automatic parameters perform better.

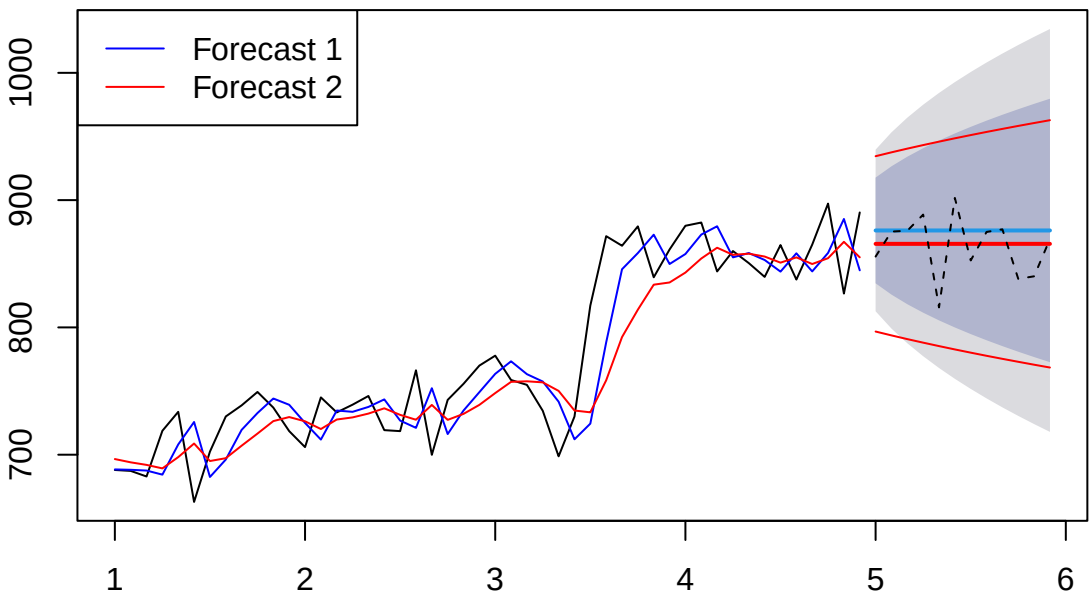
```
frc1_3 <- forecast(fit1_3, h=12)
plot(frc1_3)
lines(fit1_3$fitted,col="red")
```




```
frc2_3 <- forecast(fit2_3, h=12)
plot(frc2_3)
lines(fit2_3$fitted,col="blue")
```




```
plot(frc1_3)
 lines(fit1_3$fitted,col="blue")
 lines(frc2_3$mean,col="red",lwd=2) # lwd=2 makes the line thicker
 lines(fit2_3$fitted,col="red")
 lines(frc2_3$upper[,2],col="red")
 lines(frc2_3$lower[,2],col="red")
 lines(y3.tst,lty=2)
 legend("topleft",c("Forecast 1","Forecast 2"),col=c("blue","red"),lty=1)
```




```
MAE1_3 <- mean(abs(y3.tst- frc1_3$mean))
```

```
MAE2_3 <- mean(abs(y3.tst- frc2_3$mean))
MAE_3 <- c(MAE1_3, MAE2_3) # Collect them in a single vector
names(MAE_3) <- paste0("Forecast ",1:2) # Name the errors
round(MAE_3,3) # round to 3rd decimal point
```

```
## Forecast 1 Forecast 2
##     18.878     19.130
```

```
MSE1_3<-mean((y3.tst-frc1_3$mean)^2)
MSE2_3<-mean((y3.tst-frc2_3$mean)^2)
MSE_3<-c(MSE1_3,MSE2_3) #Collect them in asingle vector
RMSE_3<-sqrt(MSE_3) #We calculate the Root Mean Squared Error
names(RMSE_3)<-paste0("Forecast",1:2) #Name the errors
round(RMSE_3,3) #round to 3rd decimal point
```

```
## Forecast1 Forecast2
##    26.193    23.150
```

Question: Does the selected model perform best in the out-of-sample data?

Answer: The MAE suggests that the model with automatic parameters is better but the MSE on the other hand suggests that the second model which was trained with manual parameters works better for forecasting.

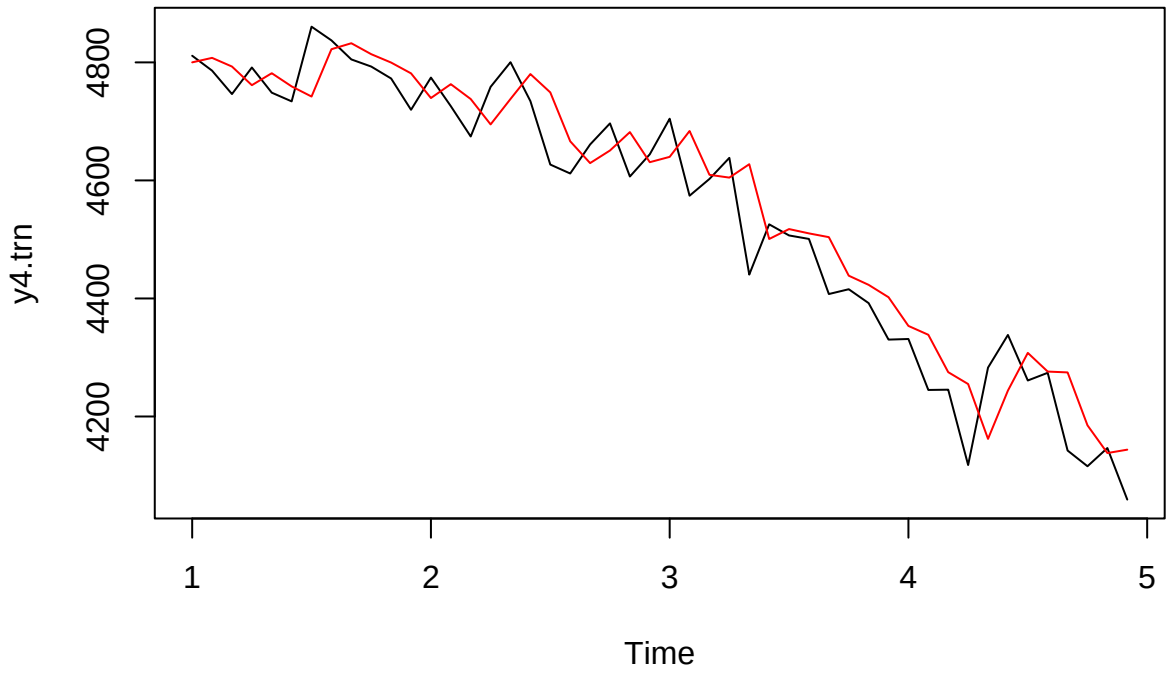
3. Trend A

```
#Transform it into a time series
y4<-ts(Y[,4],frequency=12)

y4.tst <- tail(y4,12)
y4.trn <- head(y4,48)
```

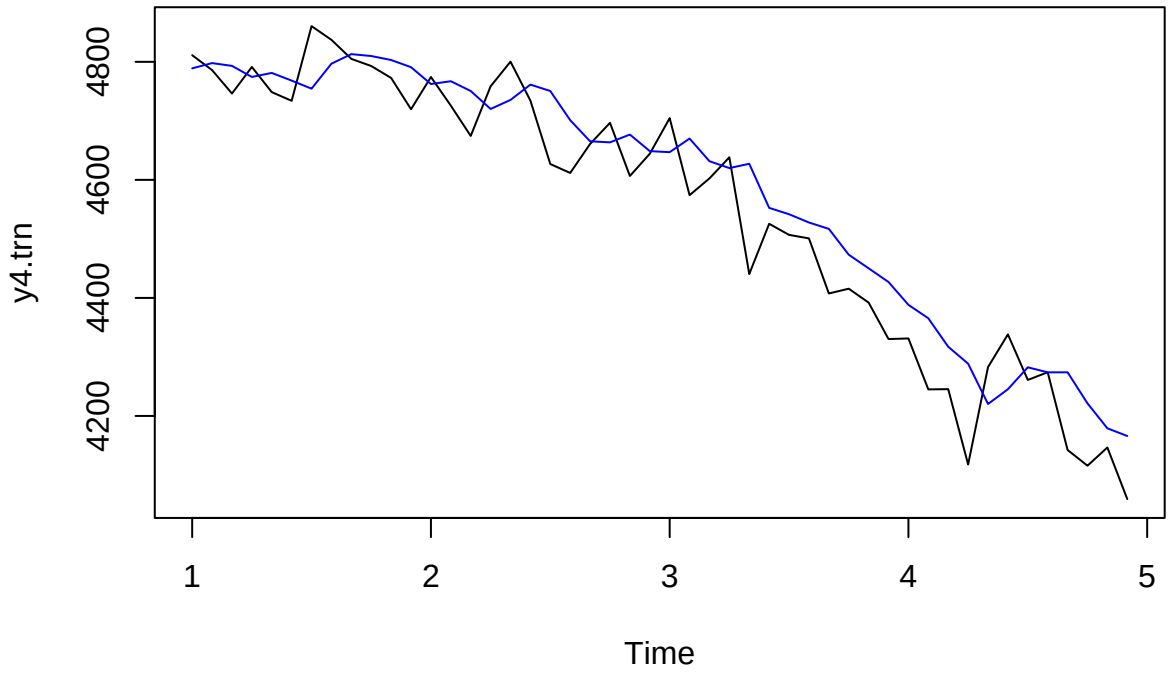
```
fit1_4 <- ets(y4.trn,model="ANN")
print(fit1_4)
```

```
## ETS(A,N,N)
##
## Call:
## ets(y = y4.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.6774
##
##   Initial states:
##     l = 4800.0578
##
##   sigma:  69.4928
##
##      AIC     AICc      BIC
## 596.9323 597.4777 602.5459
```

```
plot(y4.trn)
lines(fit1_4$fitted, col="red") # col=... specifies the colour of the line
```




```
fit2_4 <- ets(y4.trn, model = "ANN", alpha = 0.4)
plot(y4.trn)
lines(fit2_4$fitted,col="blue")
```



Question: Automatic vs. Manual parameters - Which one is best using your judgement?

Answer: The model using automatic parameters seems to follow the downwards trend better than the one trained with manual parameters.

```
fit1_4$mse
```

```
## [1] 4628.035
```

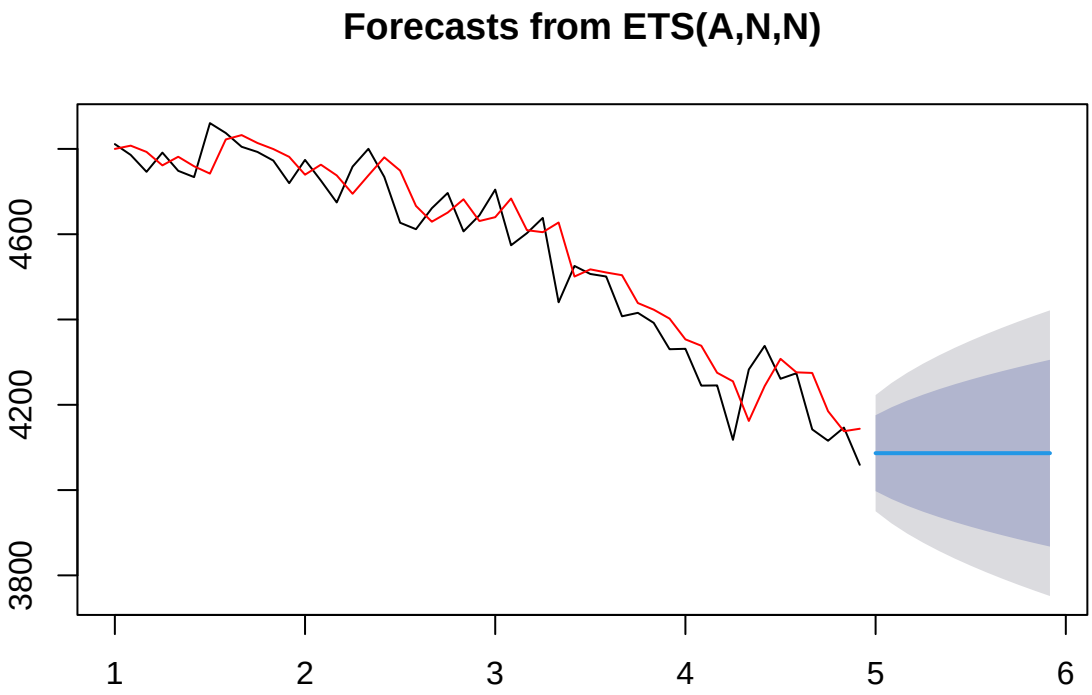
```
fit2_4$mse
```

```
## [1] 5277.633
```

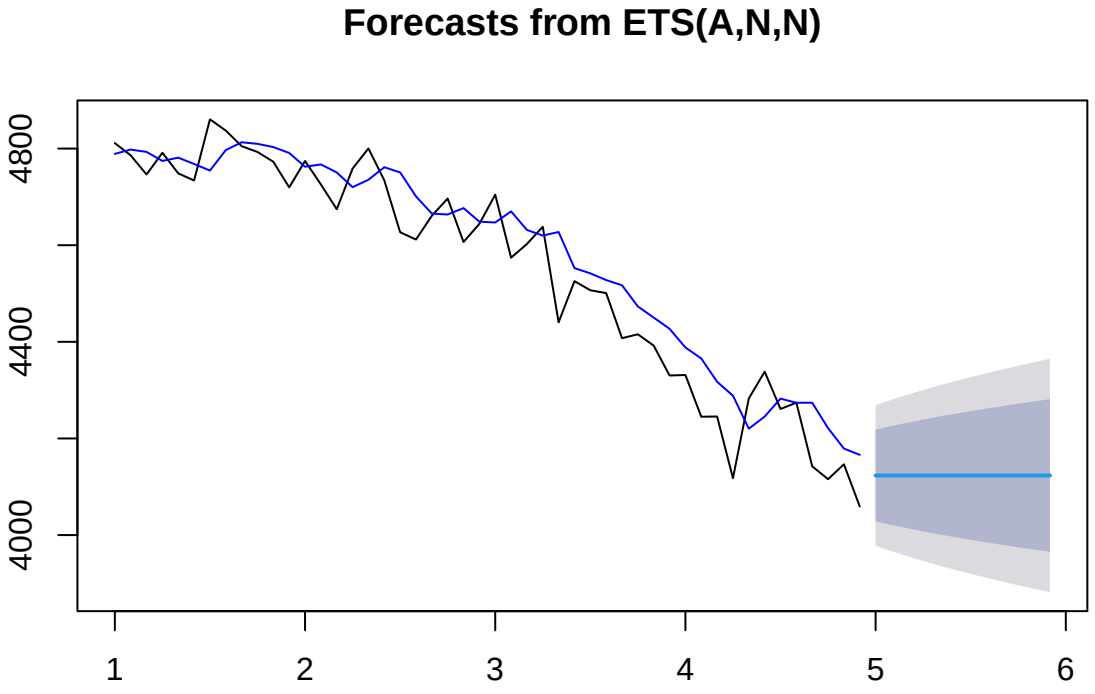
Question: Automatic vs. Manual parameters - Which one is best using errors?

Answer: The MSE suggests that the automatic parameters perform better

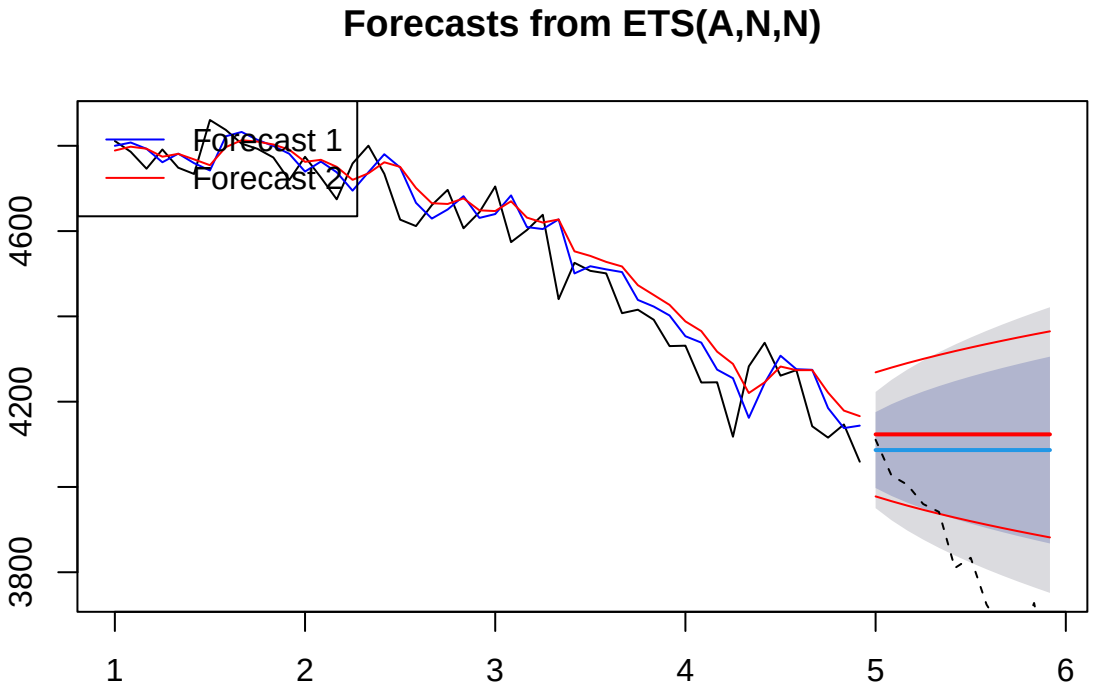
```
frc1_4 <- forecast(fit1_4, h=12)
plot(frc1_4)
lines(fit1_4$fitted,col="red")
```




```
frc2_4 <- forecast(fit2_4, h=12)
plot(frc2_4)
lines(fit2_4$fitted,col="blue")
```




```
plot(frc1_4)
 lines(fit1_4$fitted,col="blue")
 lines(frc2_4$mean,col="red",lwd=2) # lwd=2 makes the line thicker
 lines(fit2_4$fitted,col="red")
 lines(frc2_4$upper[,2],col="red")
 lines(frc2_4$lower[,2],col="red")
 lines(y4.tst,lty=2)
 legend("topleft",c("Forecast 1","Forecast 2"),col=c("blue","red"),lty=1)
```




```
MAE1_4 <- mean(abs(y4.tst- frc1_4$mean))
```

```
MAE2_4 <- mean(abs(y4.tst- frc2_4$mean))
MAE_4 <- c(MAE1_4, MAE2_4) # Collect them in a single vector
names(MAE_4) <- paste0("Forecast ",1:2) # Name the errors
round(MAE_4,3) # round to 3rd decimal point
```

```
## Forecast 1 Forecast 2
##    259.361    292.128
```

```
MSE1_4<-mean((y4.tst-frc1_4$mean)^2)
MSE2_4<-mean((y4.tst-frc2_4$mean)^2)
MSE_4<-c(MSE1_4,MSE2_4) #Collect them in asingle vector
RMSE_4<-sqrt(MSE_4) #We calculate the Root Mean Squared Error
names(RMSE_4)<-paste0("Forecast",1:2) #Name the errors
round(RMSE_4,3) #round to 3rd decimal point
```

```
## Forecast1 Forecast2
##   307.361   338.553
```

Question: Does the selected model perform best in the out-of-sample data?

Answer: The MAE & MSE also suggest that the first model which was trained with automatic parameters works better for forecasting.

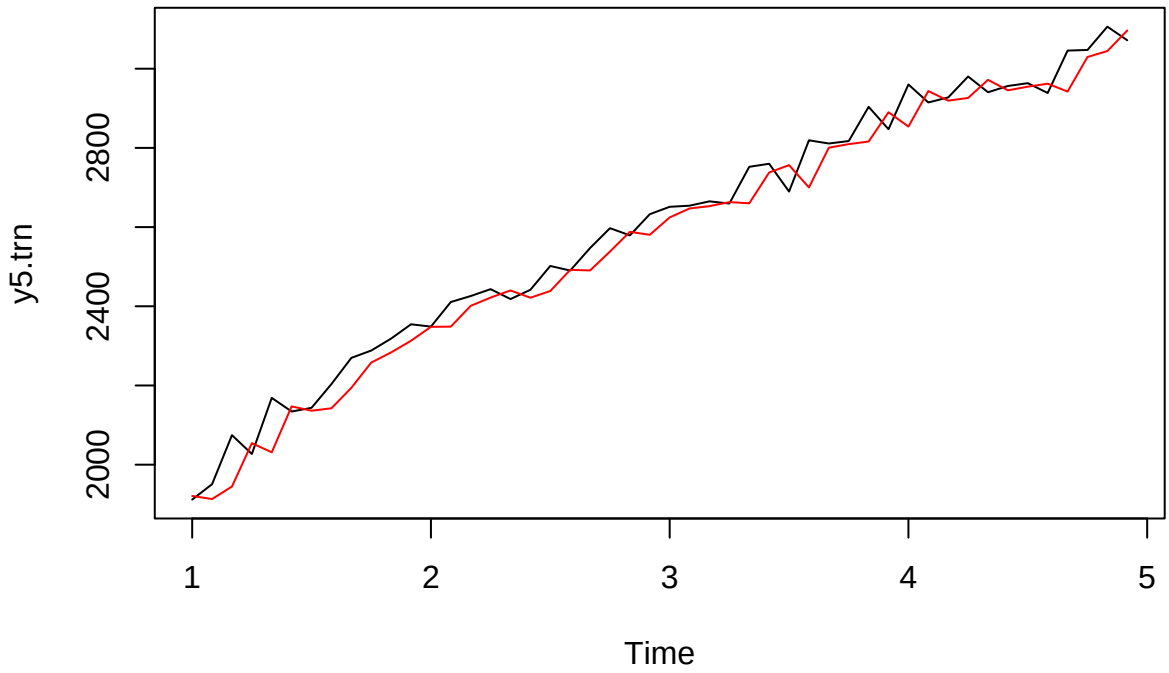
4. Trend B

```
#Transform it into a time series
y5<-ts(Y[,5],frequency=12)

y5.tst <- tail(y5,12)
y5.trn <- head(y5,48)
```

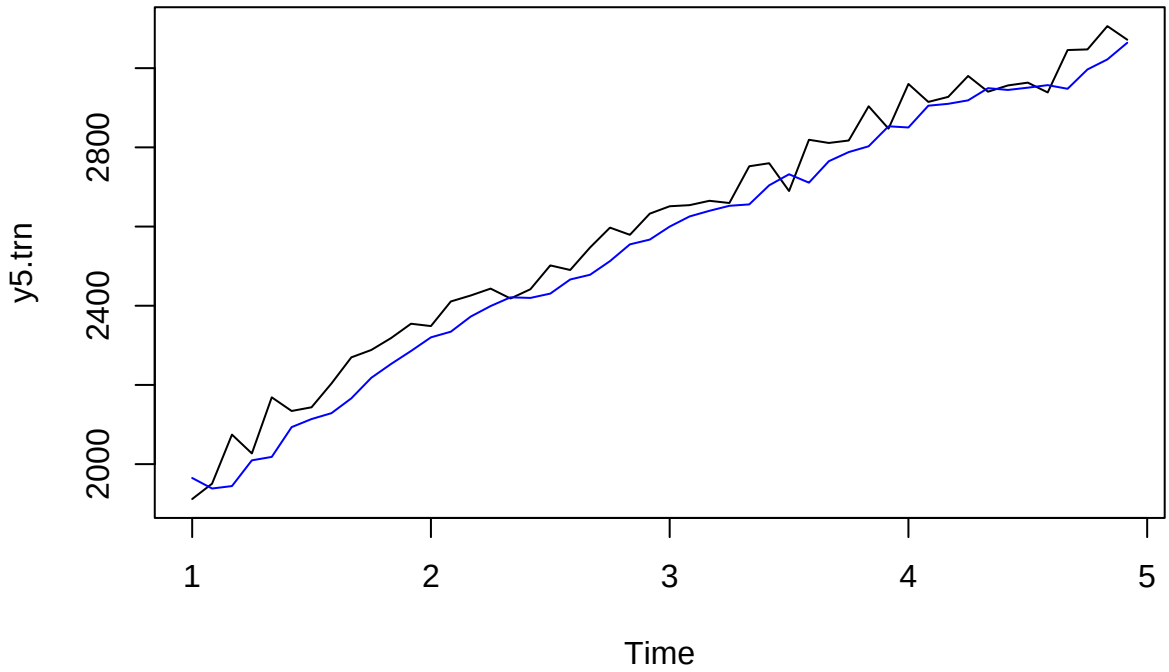
```
fit1_5 <- ets(y5.trn,model="ANN")
print(fit1_5)
```

```
## ETS(A,N,N)
##
## Call:
## ets(y = y5.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.8439
##
##   Initial states:
##     l = 1920.9275
##
##   sigma:  55.423
##
##      AIC     AICc      BIC
## 575.2143 575.7598 580.8279
```

```
plot(y5.trn)
lines(fit1_5$fitted, col="red") # col=... specifies the colour of the line
```




```
fit2_5 <- ets(y5.trn, model = "ANN", alpha = 0.5)
plot(y5.trn)
lines(fit2_5$fitted,col="blue")
```



Question: Automatic vs. Manual parameters - Which one is best using your judgement?

Answer: The model using automatic parameters seems to follow the upward trend better and closer to the actual data than the one trained with manual parameters.

```
fit1_5$mse
```

```
## [1] 2943.725
```

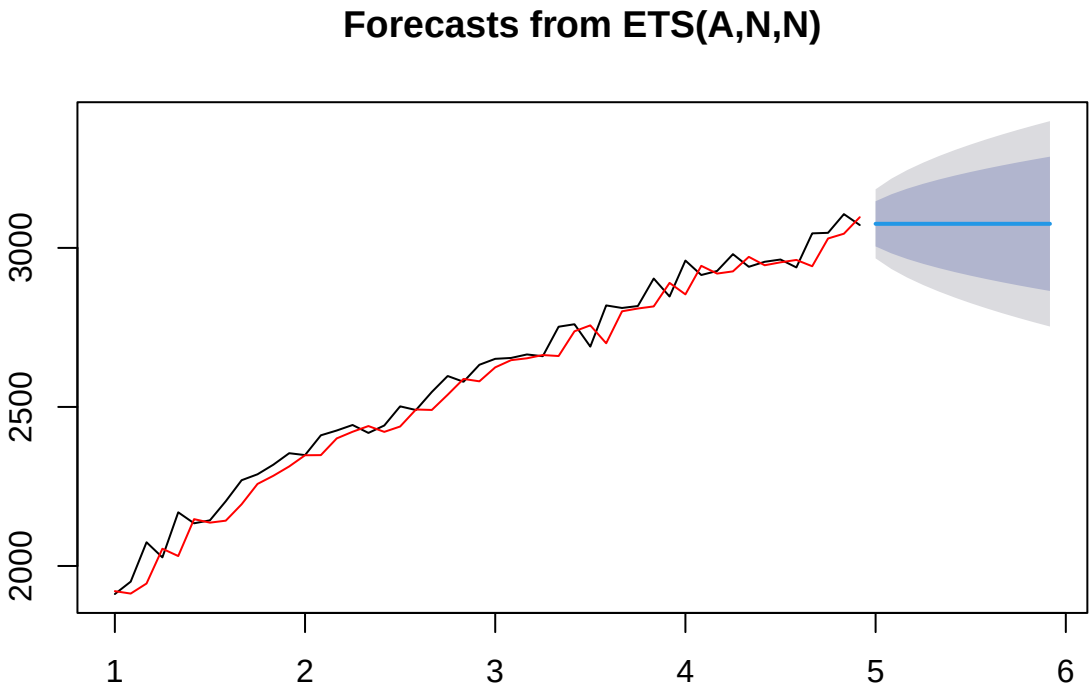
```
fit2_5$mse
```

```
## [1] 3947.611
```

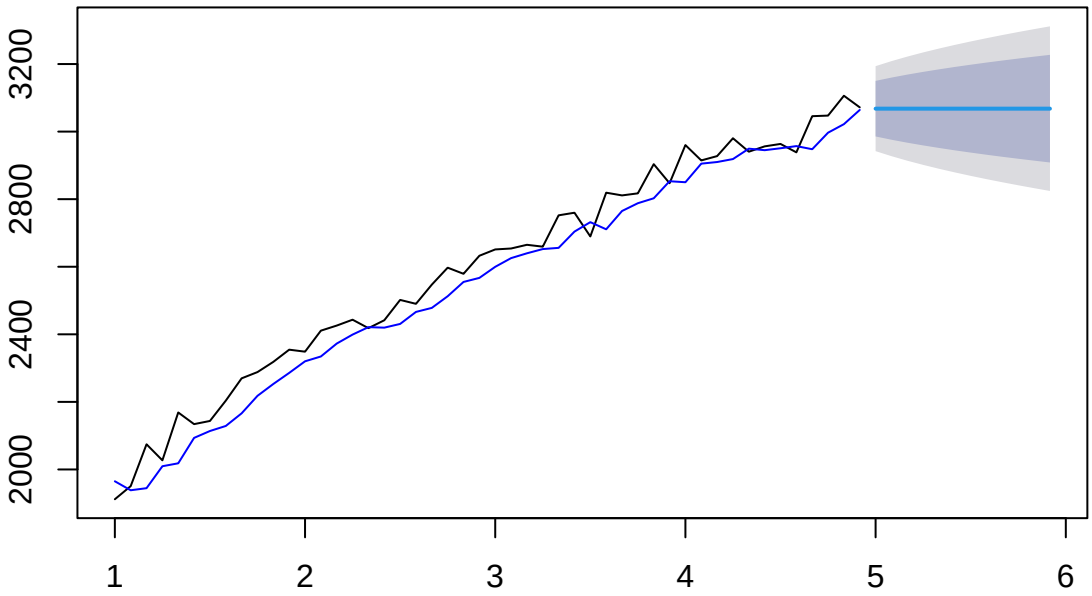
Question: Automatic vs. Manual parameters - Which one is best using errors?

Answer: The MSE suggests that the automatic parameters perform better.

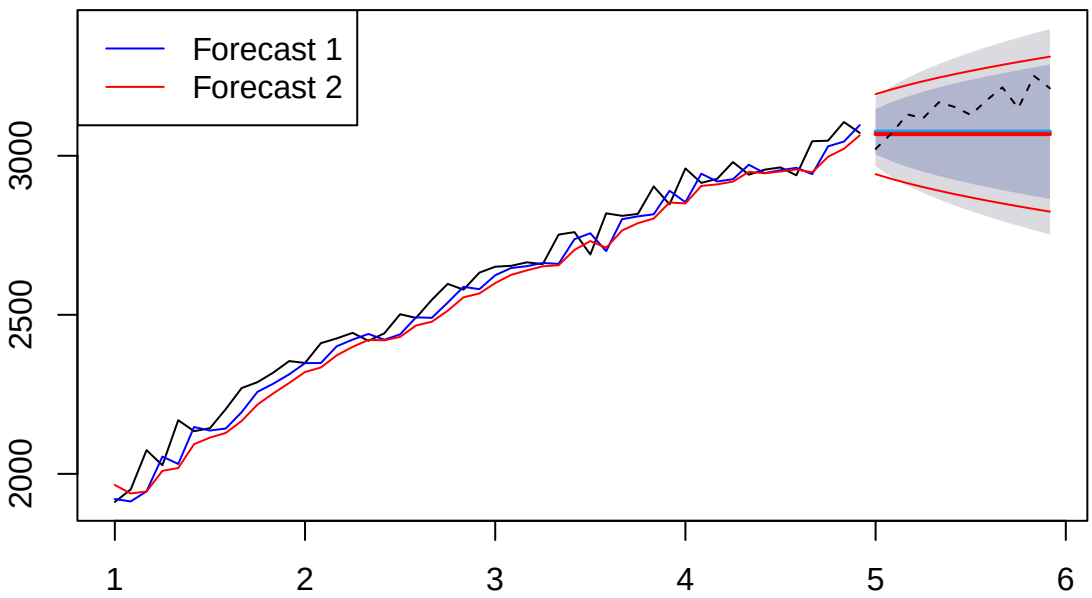
```
frc1_5 <- forecast(fit1_5, h=12)
plot(frc1_5)
lines(fit1_5$fitted,col="red")
```




```
frc2_5 <- forecast(fit2_5, h=12)
plot(frc2_5)
lines(fit2_5$fitted,col="blue")
```




```
plot(frc1_5)
 lines(fit1_5$fitted,col="blue")
 lines(frc2_5$mean,col="red",lwd=2) # lwd=2 makes the line thicker
 lines(fit2_5$fitted,col="red")
 lines(frc2_5$upper[,2],col="red")
 lines(frc2_5$lower[,2],col="red")
 lines(y5.tst,lty=2)
 legend("topleft",c("Forecast 1","Forecast 2"),col=c("blue","red"),lty=1)
```




```
MAE1_5 <- mean(abs(y5.tst- frc1_5$mean))
```

```r
MAE2_5 <- mean(abs(y5.tst- frc2_5$mean))
MAE_5 <- c(MAE1_5, MAE2_5) # Collect them in a single vector
names(MAE_5) <- paste0("Forecast ",1:2) # Name the errors
round(MAE_5,3) # round to 3rd decimal point
```

```
## Forecast 1 Forecast 2
##     83.610     89.158
```

```r
MSE1_5<-mean((y5.tst-frc1_5$mean)^2)
MSE2_5<-mean((y5.tst-frc2_5$mean)^2)
MSE_5<-c(MSE1_5,MSE2_5) #Collect them in asingle vector
RMSE_5<-sqrt(MSE_5) #We calculate the Root Mean Squared Error
names(RMSE_5)<-paste0("Forecast",1:2) #Name the errors
round(RMSE_5,3) #round to 3rd decimal point
```

```
## Forecast1 Forecast2
##    95.352   101.431
```

Question: Does the selected model perform best in the out-of-sample data?

Answer: The MAE & MSE also suggest that the first model which was trained with automatic parameters works better for forecasting.

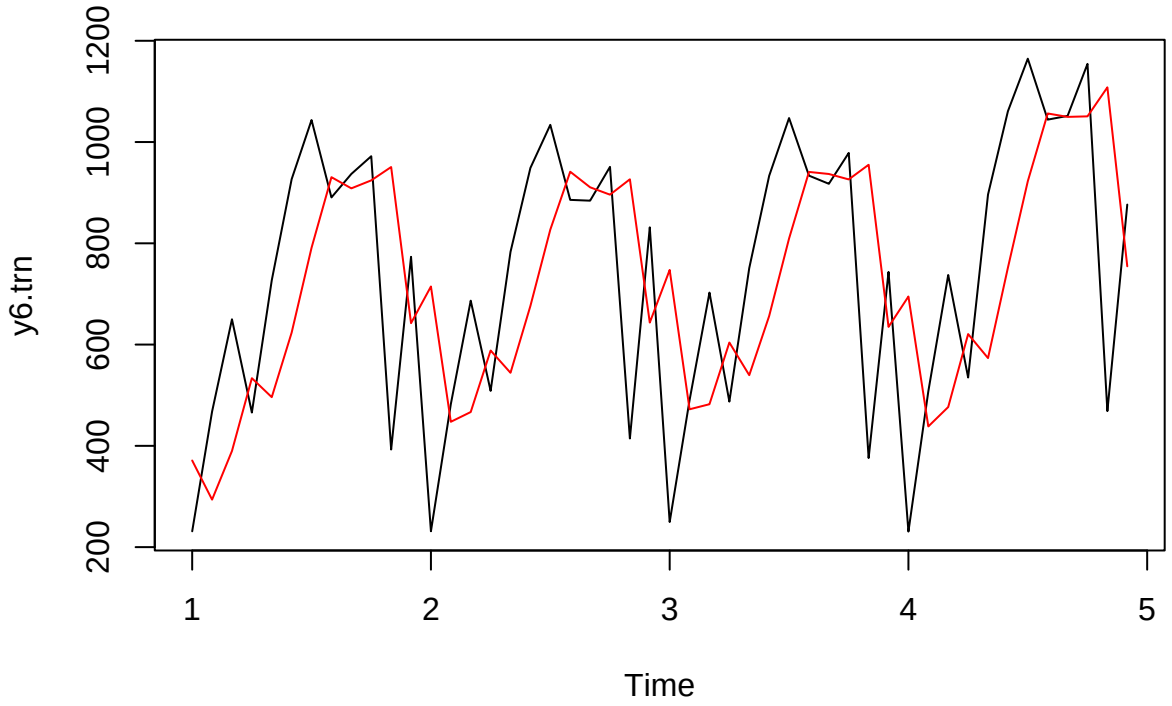
5. Season A

```r
#Transform it into a time series
y6<-ts(Y[,6],frequency=12)

y6.tst <- tail(y6,12)
y6.trn <- head(y6,48)
```

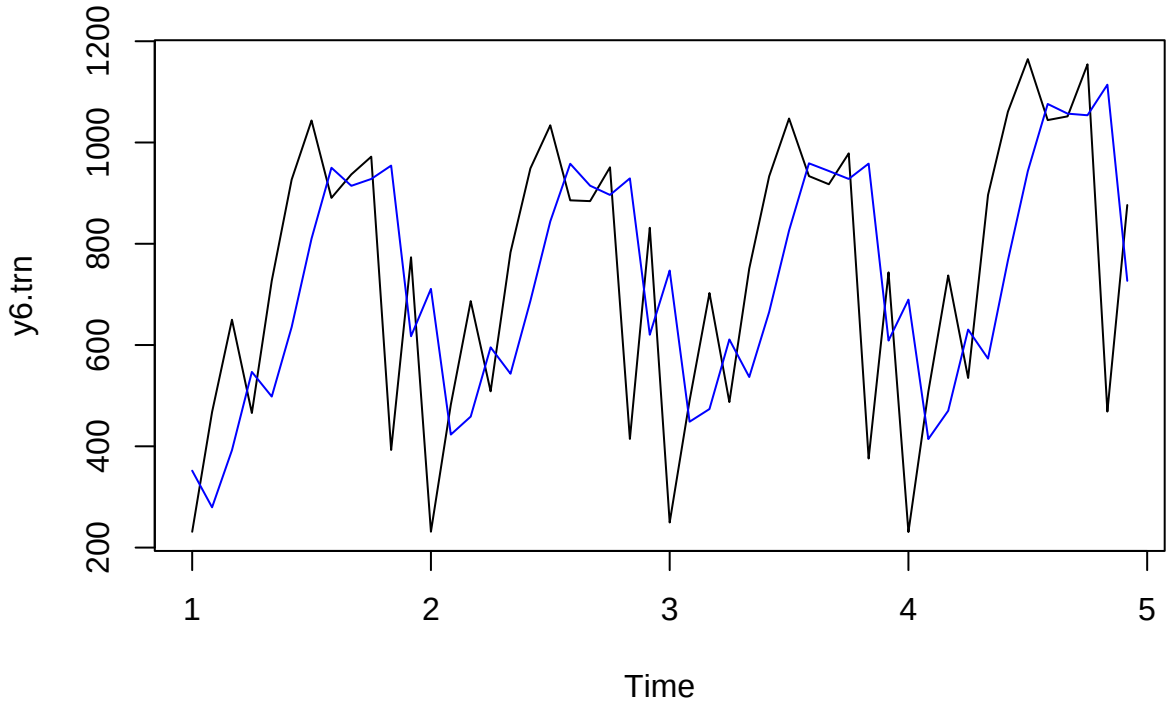
```r
fit1_6 <- ets(y6.trn,model="ANN")
print(fit1_6)
```

```
## ETS(A,N,N)
##
## Call:
## ets(y = y6.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.5527
##
##   Initial states:
##     l = 371.1084
##
##   sigma:  266.1989
##
##      AIC     AICc      BIC
## 725.8622 726.4076 731.4758
```

```r
plot(y6.trn)
lines(fit1_6$fitted, col="red") # col=... specifies the colour of the line
```




```
fit2_6 <- ets(y6.trn, model = "ANN", alpha = 0.6)
plot(y6.trn)
lines(fit2_6$fitted,col="blue")
```



Question: Automatic vs. Manual parameters - Which one is best using your judgement?

Answer: The model using automatic parameters seems to follow the season better than the one trained with manual parameters since it has a similar shape to the actual time series but both of them are quite bad since ANN does not work for seasonal data.

```
fit1_6$mse
```

```
## [1] 67909.28
```

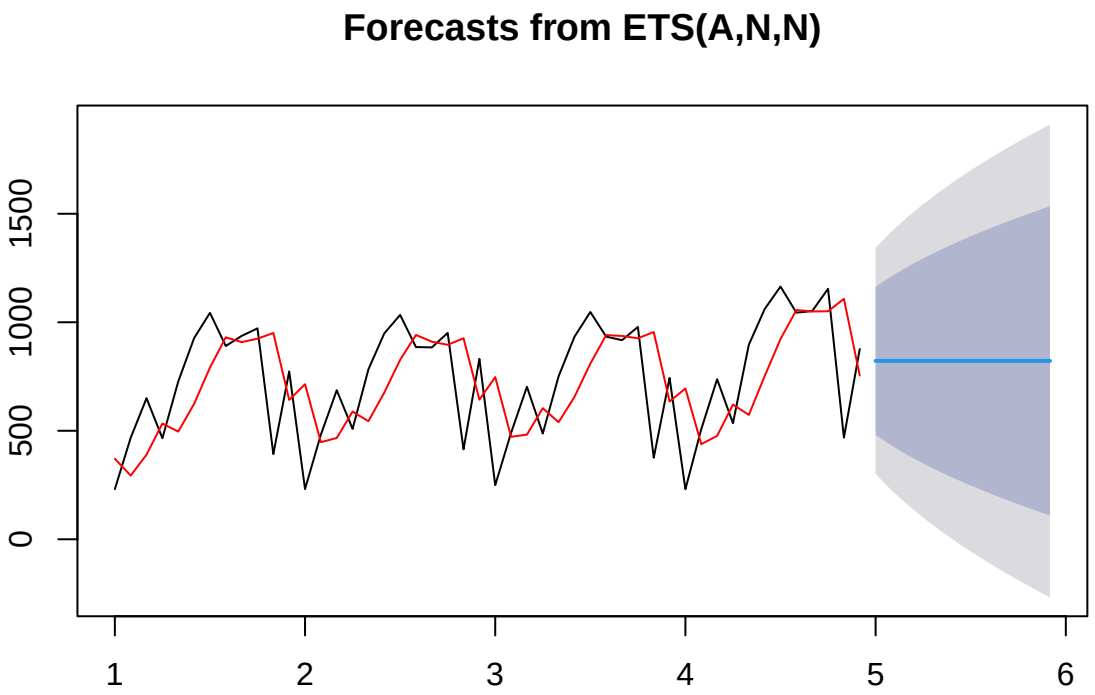
```
fit2_6$mse
```

```
## [1] 68099.11
```

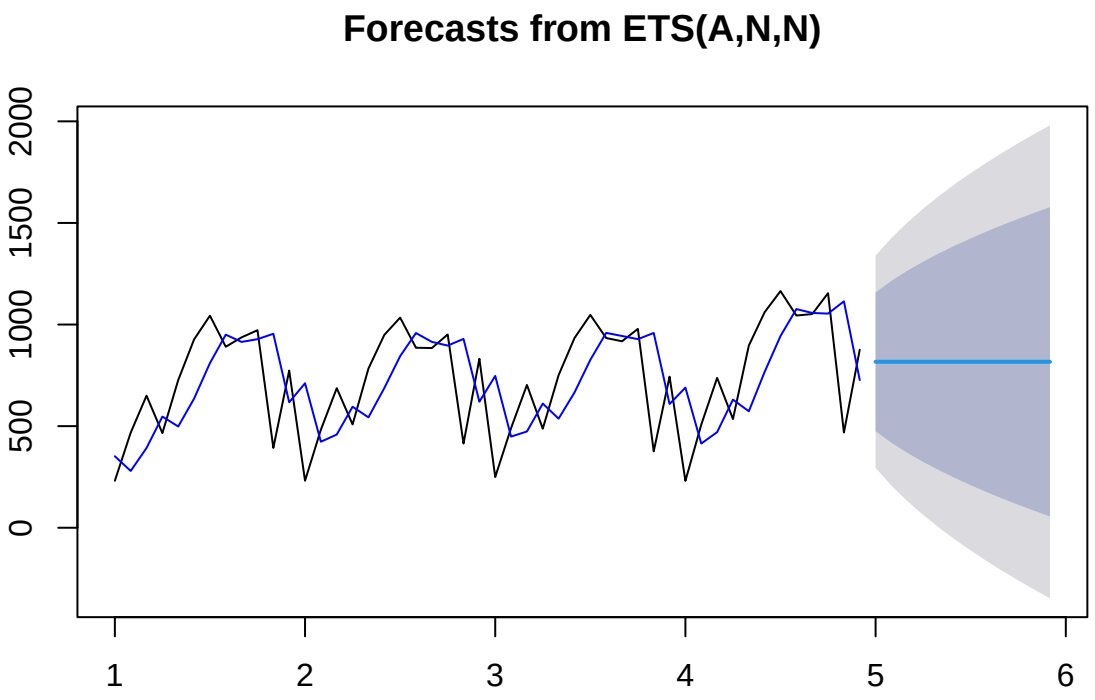
Question: Automatic vs. Manual parameters - Which one is best using errors?

Answer: The MSE suggests that the automatic parameters perform better.

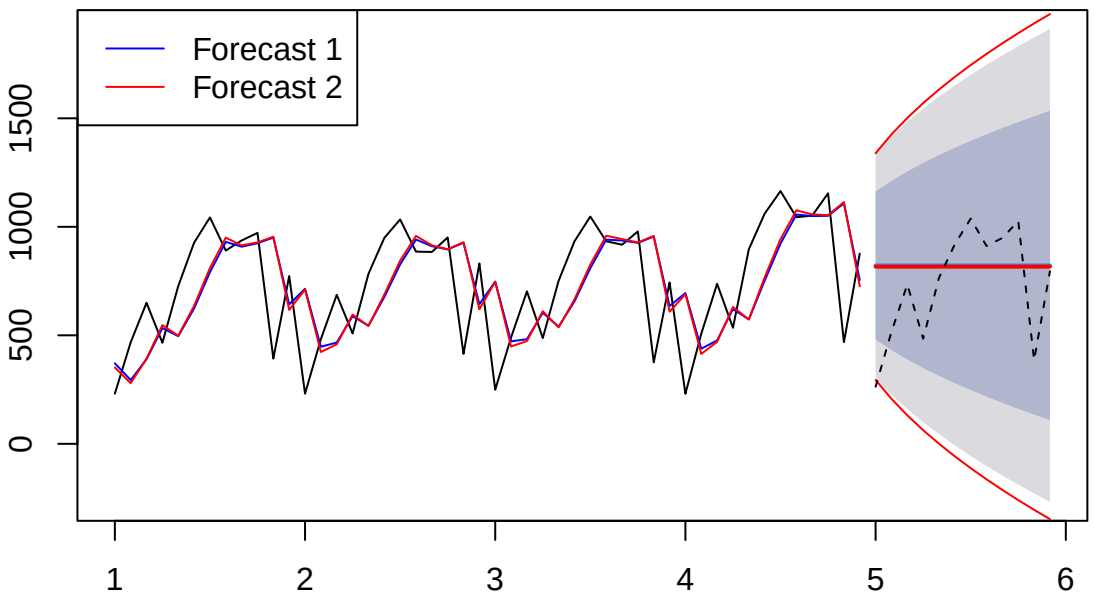
```
frc1_6 <- forecast(fit1_6, h=12)
plot(frc1_6)
lines(fit1_6$fitted,col="red")
```




```
frc2_6 <- forecast(fit2_6, h=12)
plot(frc2_6)
lines(fit2_6$fitted,col="blue")
```




```
plot(frc1_6)
 lines(fit1_6$fitted,col="blue")
 lines(frc2_6$mean,col="red",lwd=2) # lwd=2 makes the line thicker
 lines(fit2_6$fitted,col="red")
 lines(frc2_6$upper[,2],col="red")
 lines(frc2_6$lower[,2],col="red")
 lines(y6.tst,lty=2)
 legend("topleft",c("Forecast 1","Forecast 2"),col=c("blue","red"),lty=1)
```




```
MAE1_6 <- mean(abs(y6.tst- frc1_6$mean))
```

```
MAE2_6 <- mean(abs(y6.tst- frc2_6$mean))
MAE_6 <- c(MAE1_6, MAE2_6) # Collect them in a single vector
names(MAE_6) <- paste0("Forecast ",1:2) # Name the errors
round(MAE_6,3) # round to 3rd decimal point
```

```
## Forecast 1 Forecast 2
##    212.019    211.128
```

```
MSE1_6<-mean((y6.tst-frc1_6$mean)^2)
MSE2_6<-mean((y6.tst-frc2_6$mean)^2)
MSE_6<-c(MSE1_6,MSE2_6) #Collect them in asingle vector
RMSE_6<-sqrt(MSE_6) #We calculate the Root Mean Squared Error
names(RMSE_6)<-paste0("Forecast",1:2) #Name the errors
round(RMSE_6,3) #round to 3rd decimal point
```

```
## Forecast1 Forecast2
##   265.139   263.381
```

Question: Does the selected model perform best in the out-of-sample data?

Answer: The MAE & MSE on the other hand suggest that the second model which was trained with manual parameters works better for forecasting.

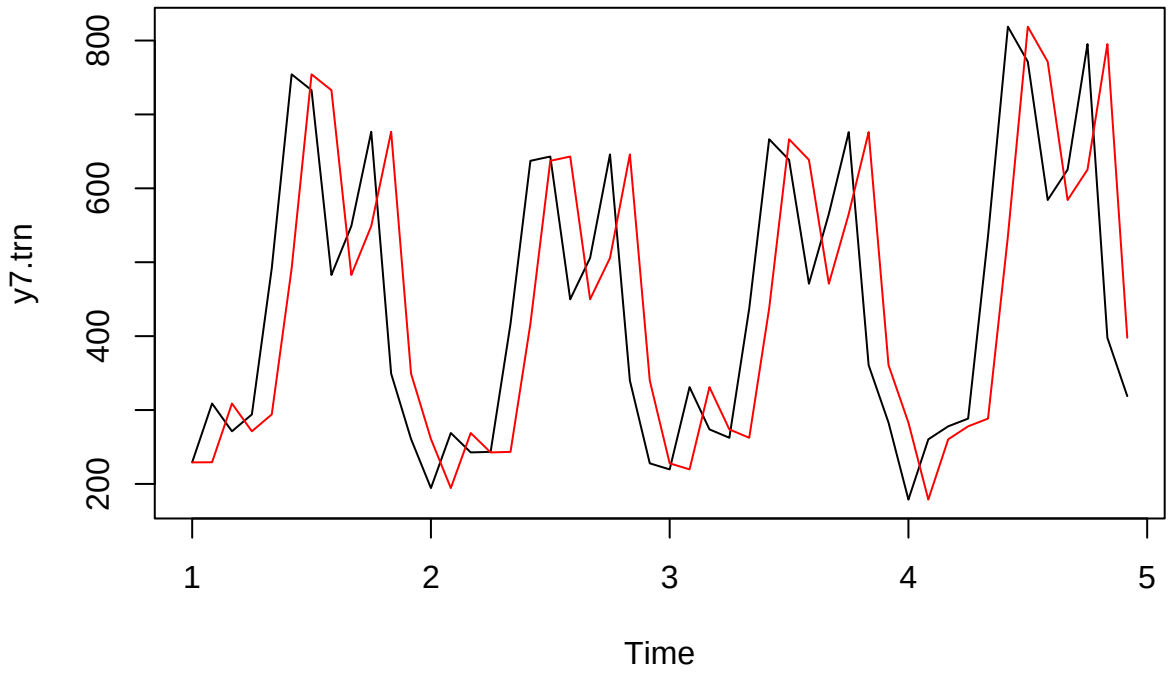
6. Season B

```
#Transform it into a time series
y7<-ts(Y[,7],frequency=12)

y7.tst <- tail(y7,12)
y7.trn <- head(y7,48)
```

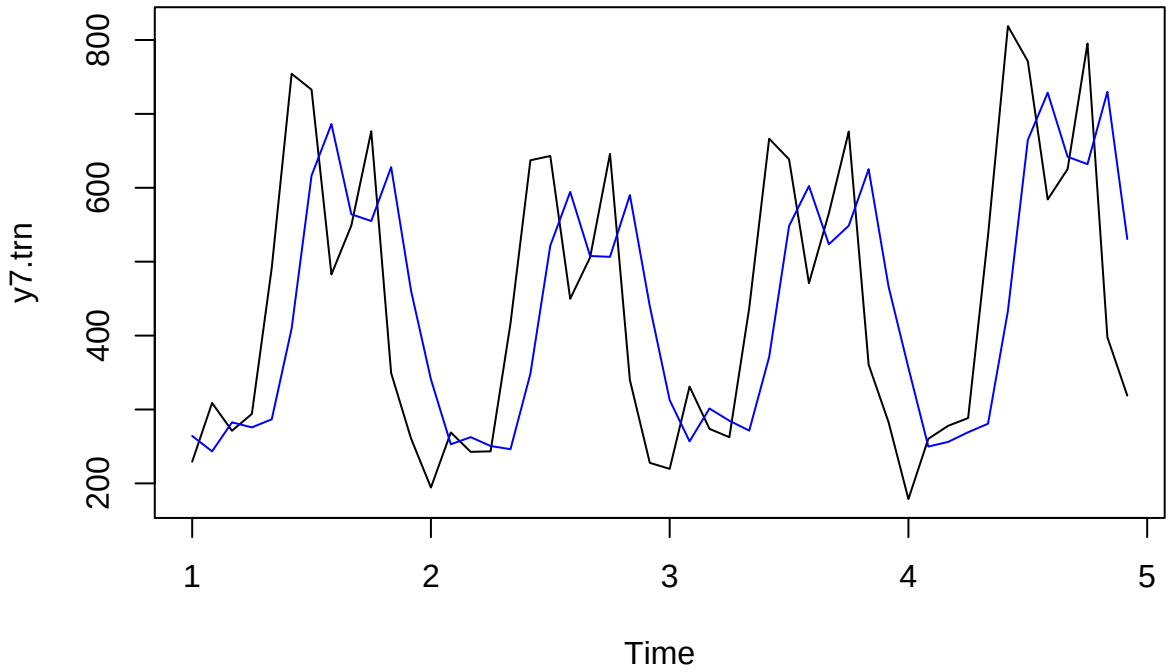
```
fit1_7 <- ets(y7.trn,model="ANN")
print(fit1_7)
```

```
## ETS(A,N,N)
##
## Call:
## ets(y = y7.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 229.1407
##
##   sigma:  162.5972
##
##      AIC     AICc      BIC
## 678.5373 679.0827 684.1509
```

```
plot(y7.trn)
lines(fit1_7$fitted, col="red") # col=... specifies the colour of the line
```




```
fit2_7 <- ets(y7.trn, model = "ANN", alpha = 0.6)
plot(y7.trn)
lines(fit2_7$fitted,col="blue")
```



Question: Automatic vs. Manual parameters - Which one is best using your judgement?

Answer: The model using automatic parameters seems to follow the season better than the one trained with manual parameters since it has a similar shape to the actual time series but both of them are quite bad since ANN does not work for seasonal data.

```
fit1_7$mse
```

```
## [1] 25336.28
```

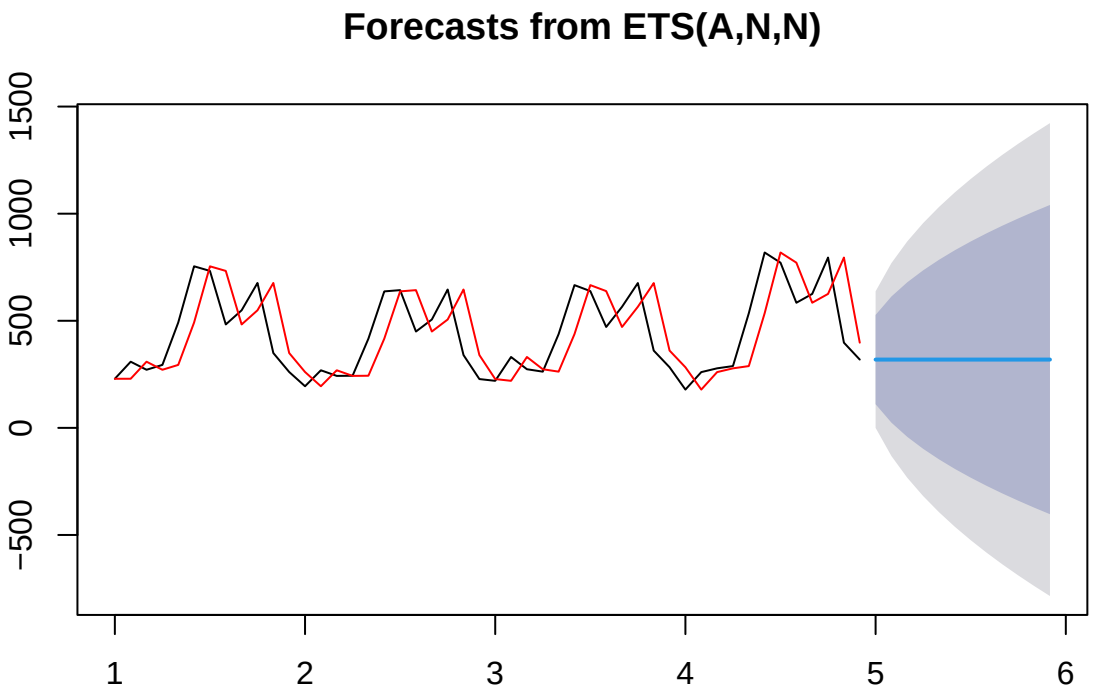
```
fit2_7$mse
```

```
## [1] 28933.35
```

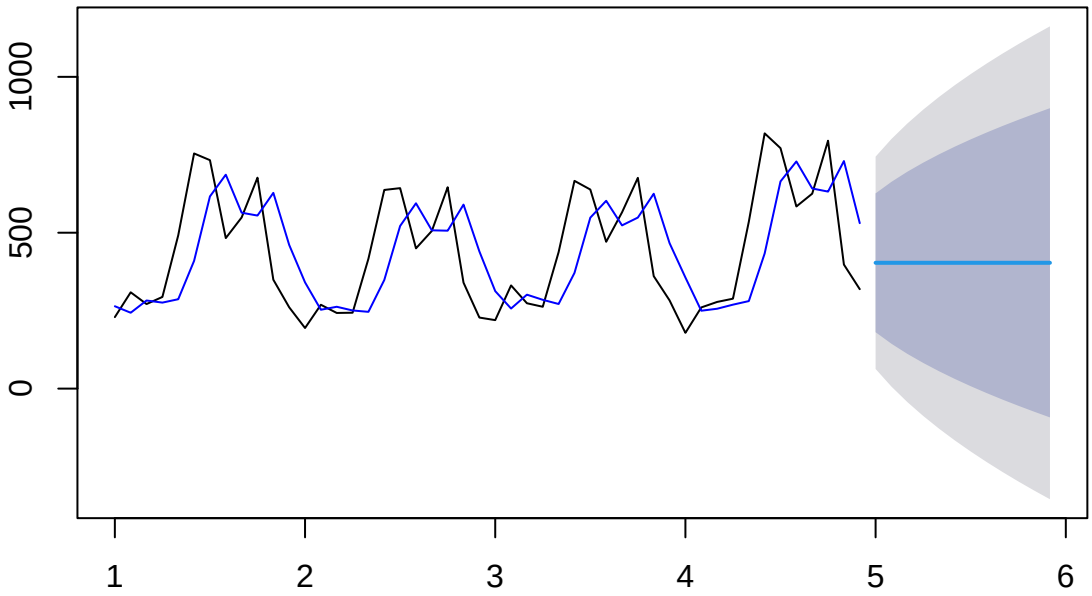
Question: Automatic vs. Manual parameters - Which one is best using errors?

Answer: The MSE suggests that the automatic parameters perform slightly better.

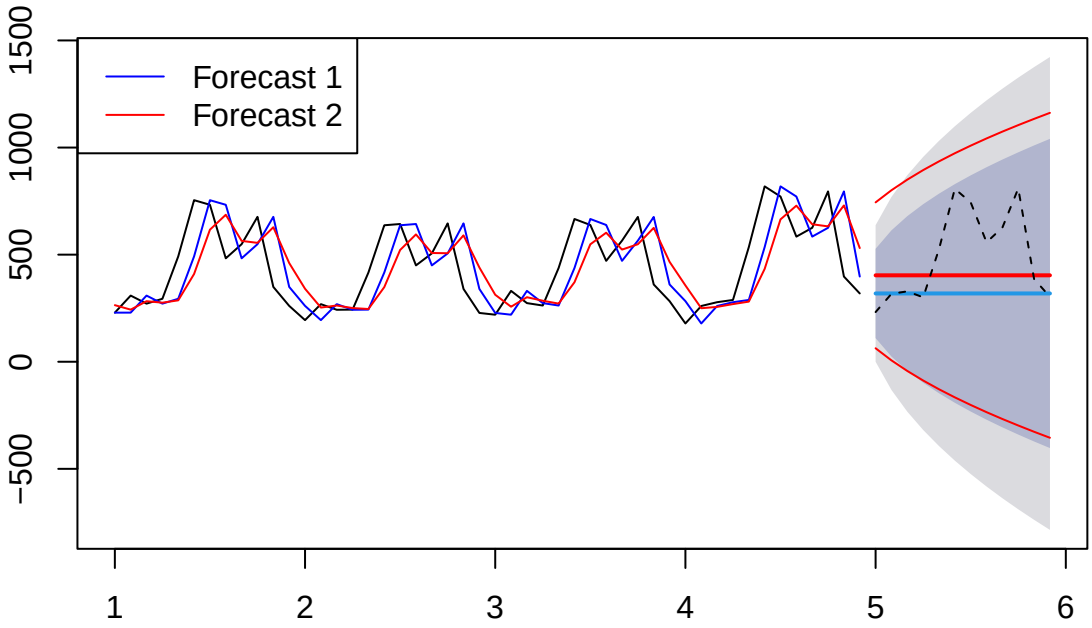
```
frc1_7 <- forecast(fit1_7, h=12)
plot(frc1_7)
lines(fit1_7$fitted,col="red")
```




```
frc2_7 <- forecast(fit2_7, h=12)
plot(frc2_7)
lines(fit2_7$fitted,col="blue")
```




```
plot(frc1_7)
 lines(fit1_7$fitted,col="blue")
 lines(frc2_7$mean,col="red",lwd=2) # lwd=2 makes the line thicker
 lines(fit2_7$fitted,col="red")
 lines(frc2_7$upper[,2],col="red")
 lines(frc2_7$lower[,2],col="red")
 lines(y7.tst,lty=2)
 legend("topleft",c("Forecast 1","Forecast 2"),col=c("blue","red"),lty=1)
```




```
MAE1_7 <- mean(abs(y7.tst- frc1_7$mean))
```

```
MAE2_7 <- mean(abs(y7.tst- frc2_7$mean))
MAE_7 <- c(MAE1_7, MAE2_7) # Collect them in a single vector
names(MAE_7) <- paste0("Forecast ",1:2) # Name the errors
round(MAE_7,3) # round to 3rd decimal point
```

```
## Forecast 1 Forecast 2
##    196.406    183.555
```

```
MSE1_7<-mean((y7.tst-frc1_7$mean)^2)
MSE2_7<-mean((y7.tst-frc2_7$mean)^2)
MSE_7<-c(MSE1_7,MSE2_7) #Collect them in a single vector
RMSE_7<-sqrt(MSE_7) #We calculate the Root Mean Squared Error
names(RMSE_7)<-paste0("Forecast",1:2) #Name the errors
round(RMSE_7,3) #round to 3rd decimal point
```

```
## Forecast1 Forecast2
##   268.774   222.716
```

Question: Does the selected model perform best in the out-of-sample data?

Answer: The MAE & MSE on the other hand suggest that the second model which was trained with manual parameters works better for forecasting.

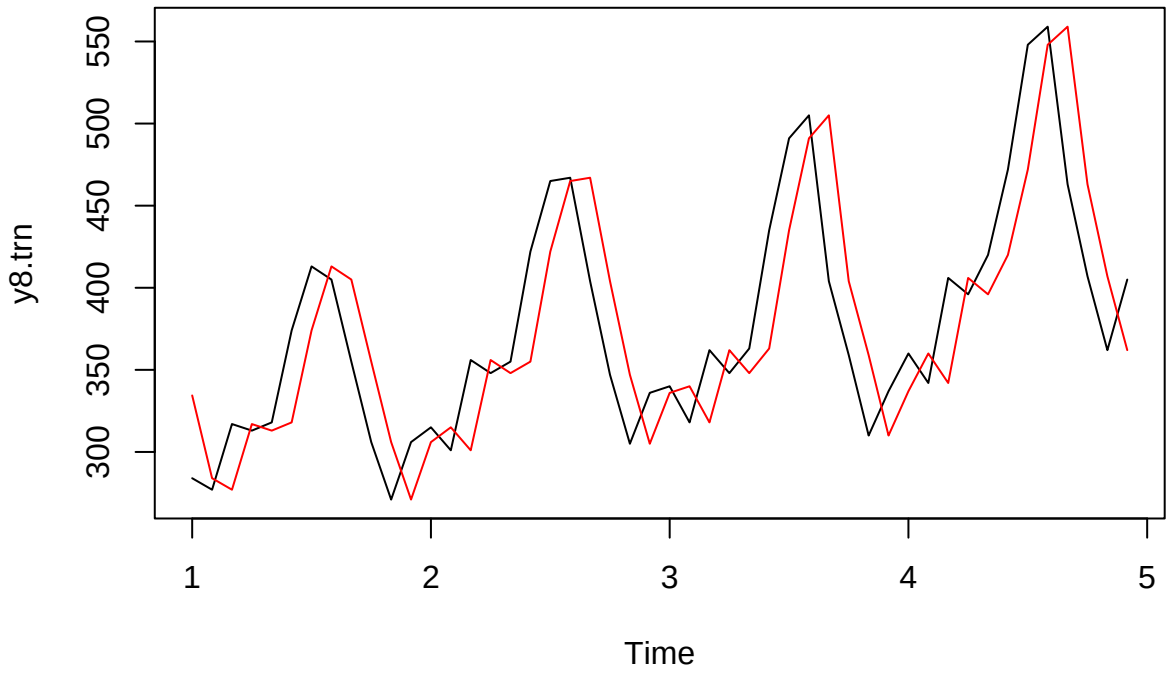
7. Trend Season

```
#Transform it into a time series
y8<-ts(Y[,8],frequency=12)

y8.tst <- tail(y8,12)
y8.trn <- head(y8,48)
```

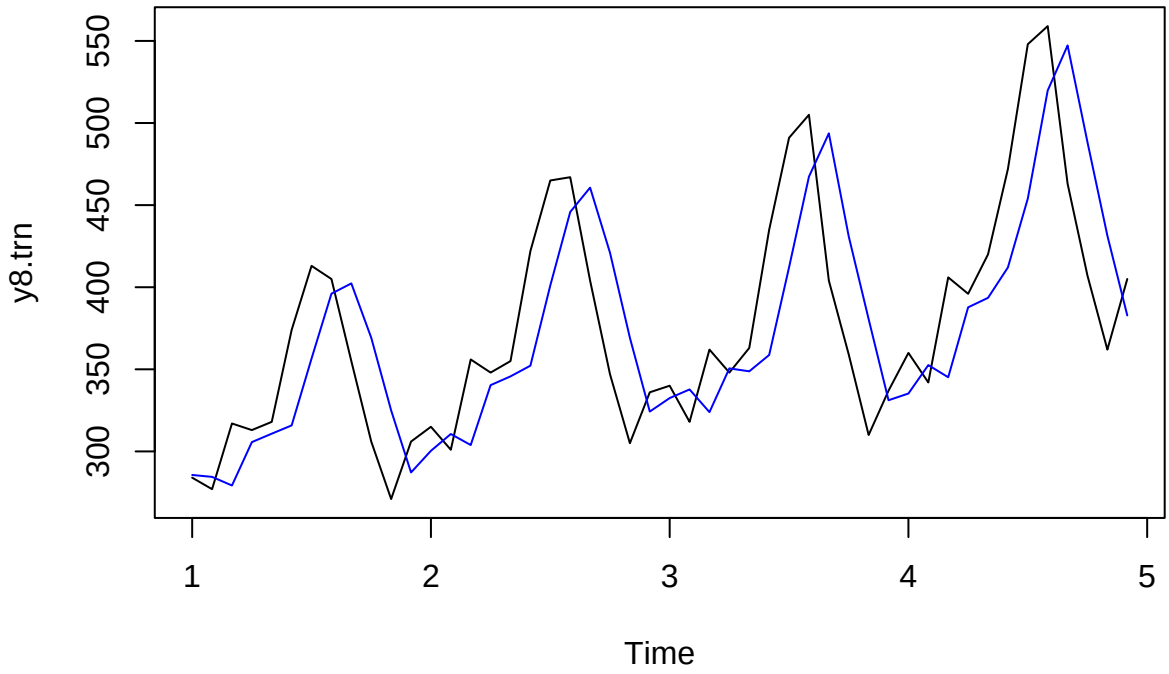
```
fit1_8 <- ets(y8.trn,model="ANN")
print(fit1_8)
```

```
## ETS(A,N,N)
##
## Call:
## ets(y = y8.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 334.407
##
##   sigma:  45.0096
##
##      AIC     AICc      BIC
## 555.2349 555.7803 560.8485
```

```
plot(y8.trn)
lines(fit1_8$fitted, col="red") # col=... specifies the colour of the line
```




```
fit2_8 <- ets(y8.trn, model = "ANN", alpha = 0.7)
plot(y8.trn)
lines(fit2_8$fitted,col="blue")
```



Question: Automatic vs. Manual parameters - Which one is best using your judgement?

Answer: The model using automatic parameters seems to follow the season better than the one trained with manual parameters since it has a similar shape to the actual time series but both of them are quite bad since ANN does not work for seasonal/trended data.

```
fit1_8$mse
```

```
## [1] 1941.454
```

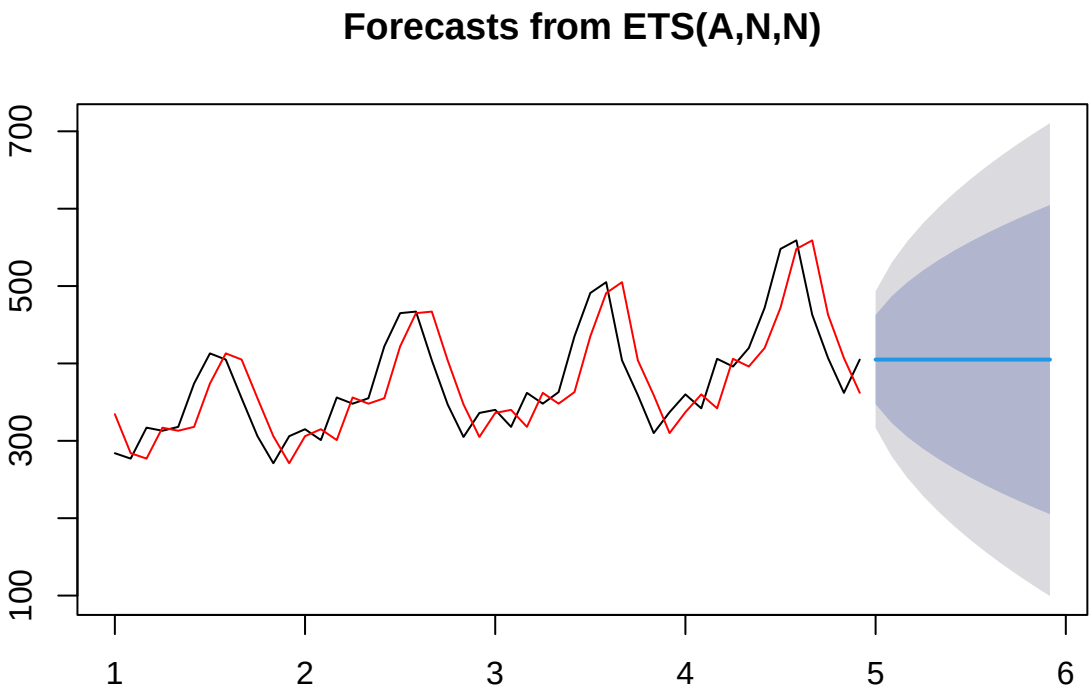
```
fit2_8$mse
```

```
## [1] 2400.245
```

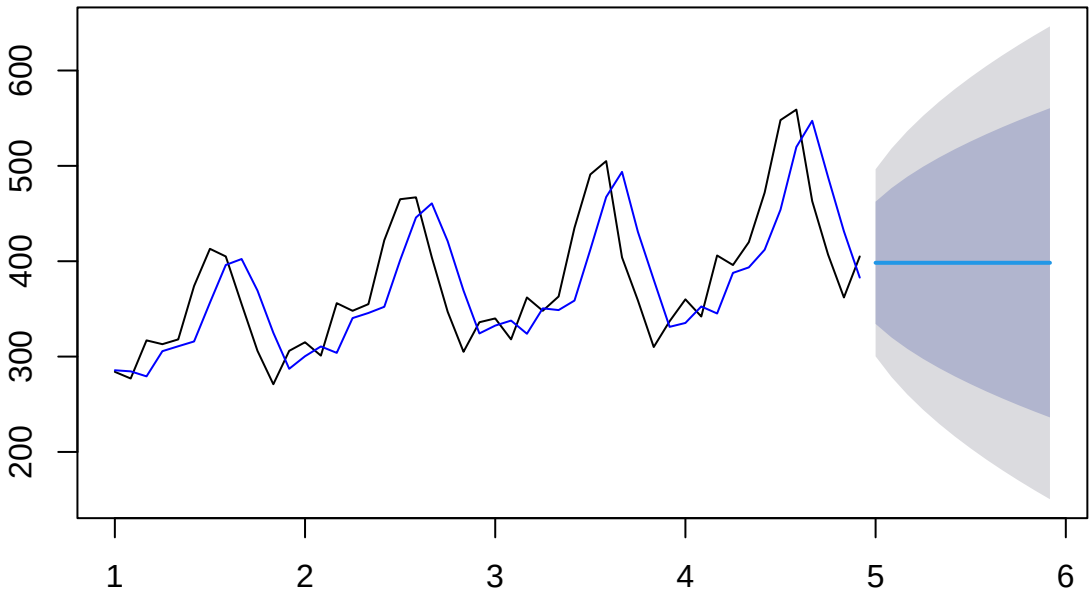
Question: Automatic vs. Manual parameters - Which one is best using errors?

Answer: The MSE suggests that the automatic parameters perform better.

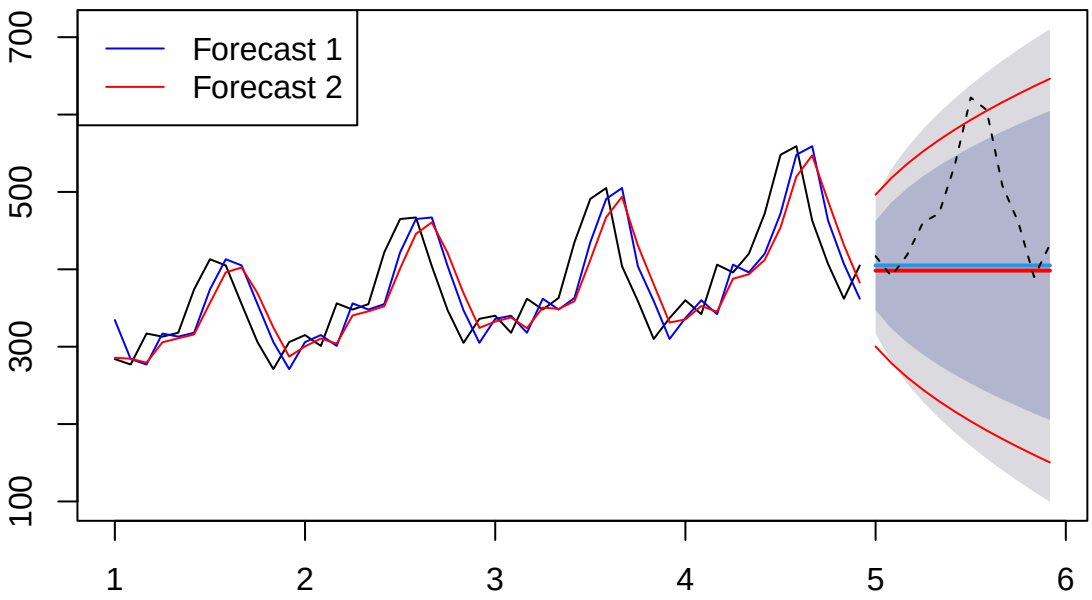
```
frc1_8 <- forecast(fit1_8, h=12)
plot(frc1_8)
lines(fit1_8$fitted,col="red")
```




```
frc2_8 <- forecast(fit2_8, h=12)
plot(frc2_8)
lines(fit2_8$fitted,col="blue")
```




```
plot(frc1_8)
 lines(fit1_8$fitted,col="blue")
 lines(frc2_8$mean,col="red",lwd=2) # lwd=2 makes the line thicker
 lines(fit2_8$fitted,col="red")
 lines(frc2_8$upper[,2],col="red")
 lines(frc2_8$lower[,2],col="red")
 lines(y8.tst,lty=2)
 legend("topleft",c("Forecast 1","Forecast 2"),col=c("blue","red"),lty=1)
```




```
MAE1_8 <- mean(abs(y8.tst- frc1_8$mean))
```

```
MAE2_8 <- mean(abs(y8.tst- frc2_8$mean))
MAE_8 <- c(MAE1_8, MAE2_8) # Collect them in a single vector
names(MAE_8) <- paste0("Forecast ",1:2) # Name the errors
round(MAE_8,3) # round to 3rd decimal point
```

```
## Forecast 1 Forecast 2
##     76.003     80.437
```

```
MSE1_8<-mean((y8.tst-frc1_8$mean)^2)
MSE2_8<-mean((y8.tst-frc2_8$mean)^2)
MSE_8<-c(MSE1_8,MSE2_8) #Collect them in asingle vector
RMSE_8<-sqrt(MSE_8) #We calculate the Root Mean Squared Error
names(RMSE_8)<-paste0("Forecast",1:2) #Name the errors
round(RMSE_8,3) #round to 3rd decimal point
```

```
## Forecast1 Forecast2
##   102.980   107.684
```

Question: Does the selected model perform best in the out-of-sample data?

Answer: The MAE & MSE also suggest that the first model which was trained with automatic parameters works better for forecasting.

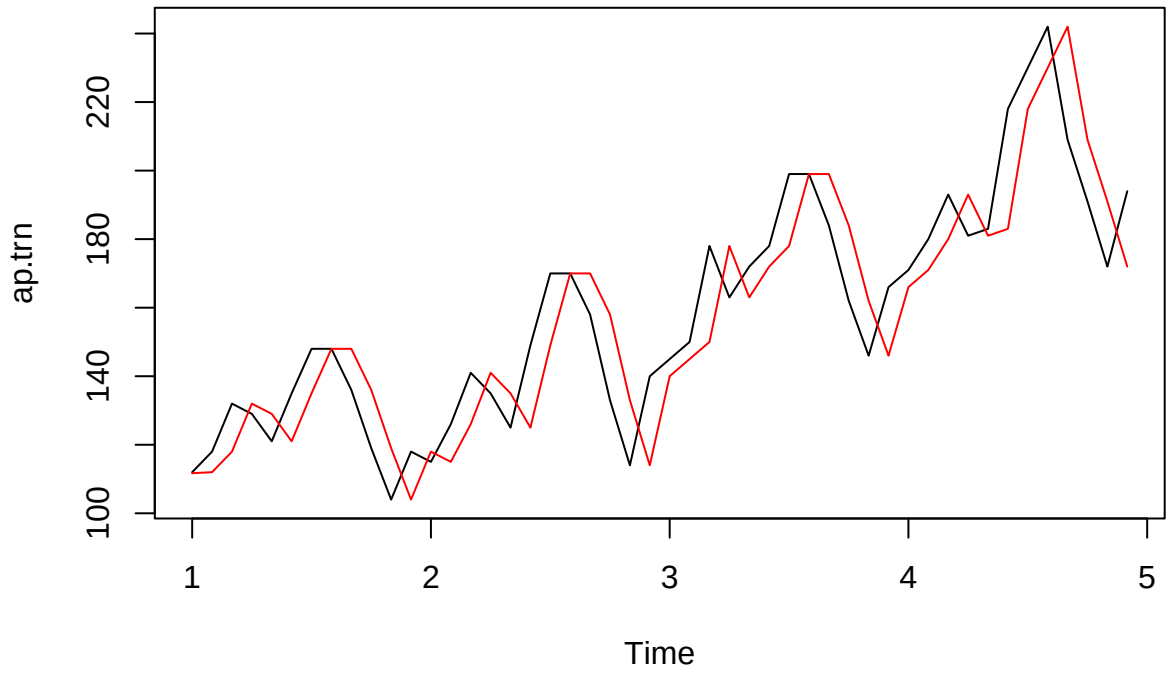
8. Air Passengers

```
#Transform it into a time series
ap <- AirPassengers
ap <-ts(ap,frequency=12)

ap.tst <- tail(ap,12)
ap.trn <- head(ap,48)
```

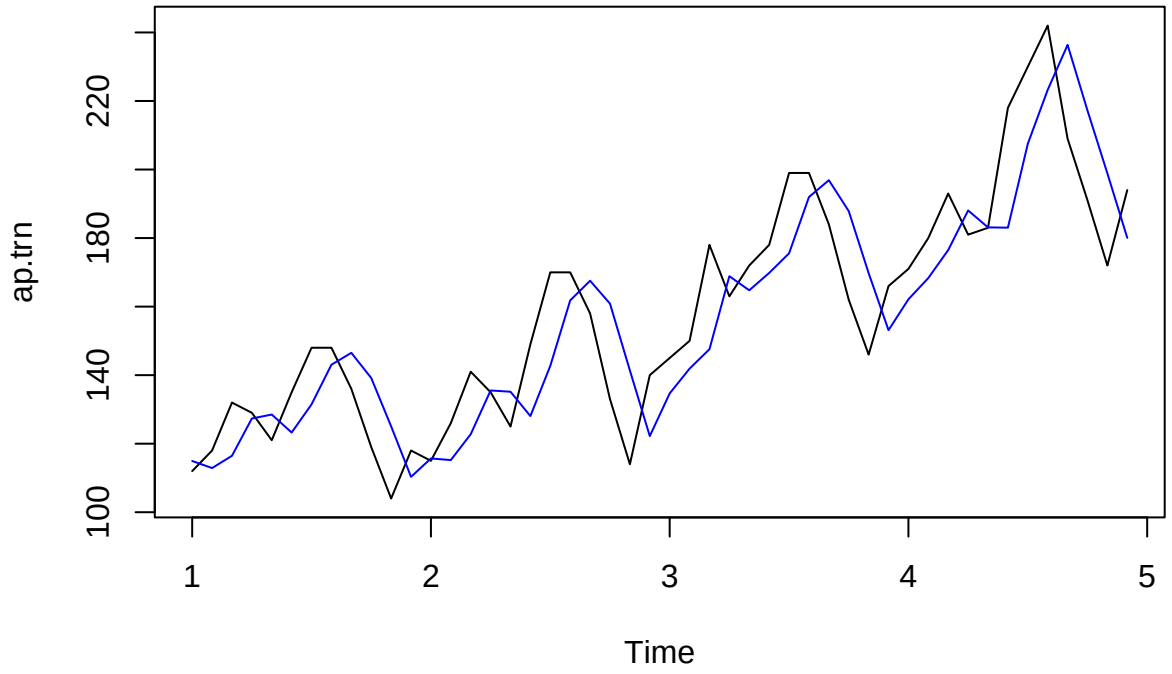
```
fit1_ap <- ets(ap.trn,model="ANN")
print(fit1_ap)
```

```
## ETS(A,N,N)
##
## Call:
## ets(y = ap.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 111.6874
##
##   sigma:  16.1612
##
##      AIC     AICc      BIC
## 456.9058 457.4513 462.5194
```

```
plot(ap.trn)
lines(fit1_ap$fitted, col="red") # col=... specifies the colour of the line
```




```
fit2_ap <- ets(ap.trn, model = "ANN", alpha = 0.7)
plot(ap.trn)
lines(fit2_ap$fitted,col="blue")
```



Question: Automatic vs. Manual parameters - Which one is best using your judgement?

Answer: The model using automatic parameters seems to follow the season better than the one trained with manual parameters since it has a similar shape to the actual time series but both of them are quite bad since ANN does not work for seasonal/trended data.

```
fit1_ap$mse
```

```
## [1] 250.3025
```

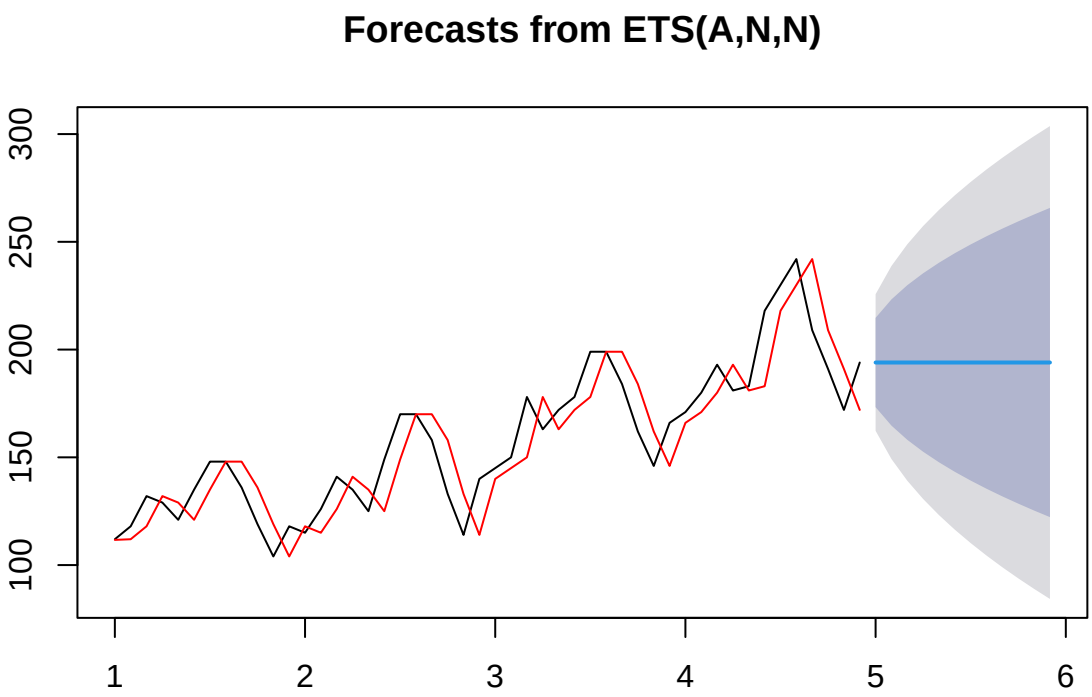
```
fit2_ap$mse
```

```
## [1] 289.8231
```

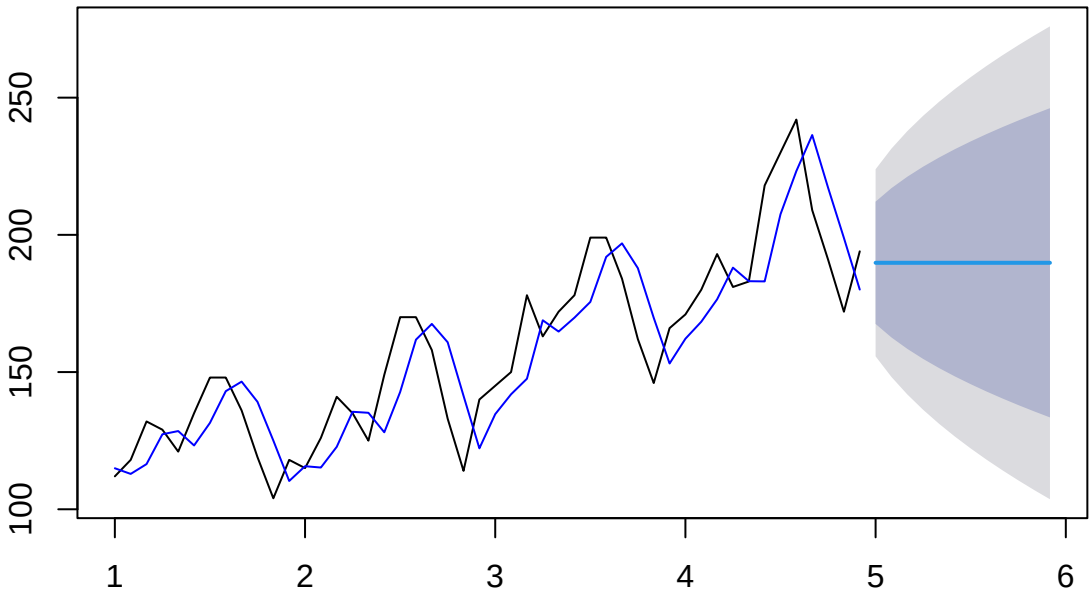
Question: Automatic vs. Manual parameters - Which one is best using errors?

Answer: The MSE suggests that the automatic parameters perform better.

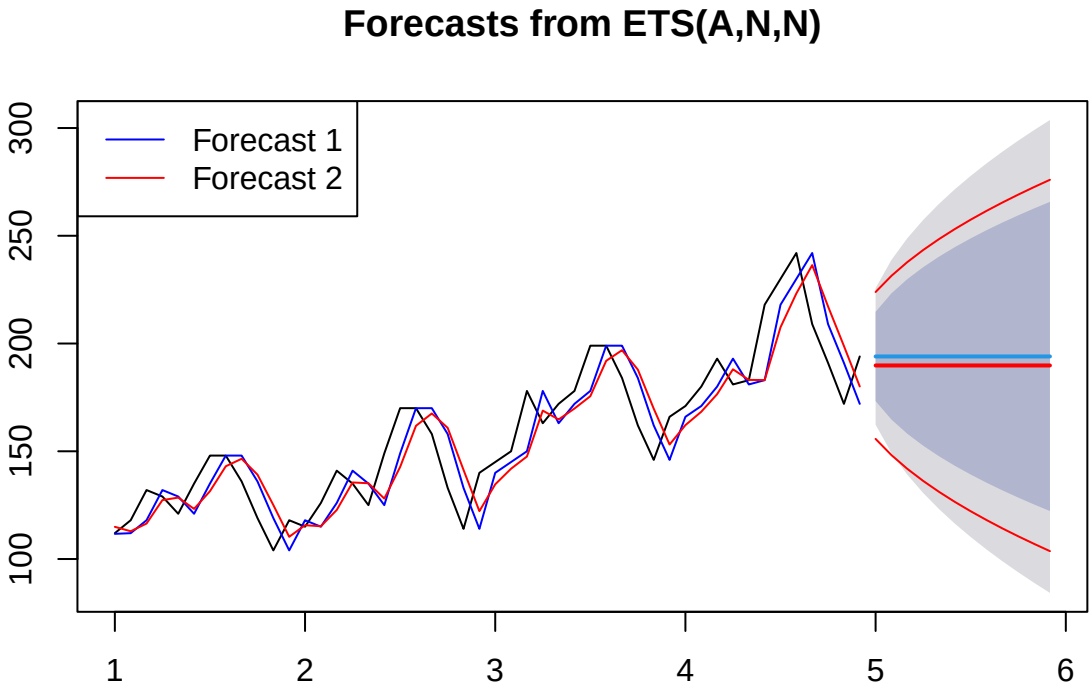
```
frc1_ap <- forecast(fit1_ap, h=12)
plot(frc1_ap)
lines(fit1_ap$fitted,col="red")
```




```
frc2_ap <- forecast(fit2_ap, h=12)
plot(frc2_ap)
lines(fit2_ap$fitted,col="blue")
```




```
plot(frc1_ap)
 lines(fit1_ap$fitted,col="blue")
 lines(frc2_ap$mean,col="red",lwd=2) # lwd=2 makes the line thicker
 lines(fit2_ap$fitted,col="red")
 lines(frc2_ap$upper[,2],col="red")
 lines(frc2_ap$lower[,2],col="red")
 lines(ap.tst,lty=2)
 legend("topleft",c("Forecast 1","Forecast 2"),col=c("blue","red"),lty=1)
```




```
MAE1_ap <- mean(abs(as.numeric(ap.tst) - as.numeric(frc1_ap$mean)))
```

```r
MAE2_ap <- mean(abs(as.numeric(ap.tst) - as.numeric(frc2_ap$mean)))
MAE_ap <- c(MAE1_ap, MAE2_ap) # Collect them in a single vector
names(MAE_ap) <- paste0("Forecast ",1:2) # Name the errors
round(MAE_ap,3) # round to 3rd decimal point
```

```
## Forecast 1 Forecast 2
##    282.169    286.349
```

```r
MSE1_ap<-mean((as.numeric(ap.tst) - as.numeric(frc1_ap$mean))^2)
MSE2_ap<-mean((as.numeric(ap.tst) - as.numeric(frc2_ap$mean))^2)
MSE_ap<-c(MSE1_ap,MSE2_ap) #Collect them in a single vector
RMSE_ap<-sqrt(MSE_ap) #We calculate the Root Mean Squared Error
names(RMSE_ap)<-paste0("Forecast",1:2) #Name the errors
round(RMSE_ap,3) #round to 3rd decimal point
```

```
## Forecast1 Forecast2
##   291.820   295.863
```

Question: Does the selected model perform best in the out-of-sample data?

Answer: The MAE & MSE also suggest that the first model which was trained with automatic parameters works better for forecasting.

Comments:

Trend, Trend Season - The model with automatic parameters seems to work better.

Season & Level- The model with manual parameters seems to work better.