Assignment 2

Rizny Mubarak

2023-09-05

Contents

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

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

```r
library(tsutils)
```

```
## Warning: package 'tsutils' was built under R version 4.2.3
```

```r
library(ggplot2)
```

0.1 Level A

0.1.1 1. Loading Data

```r
# 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)
```

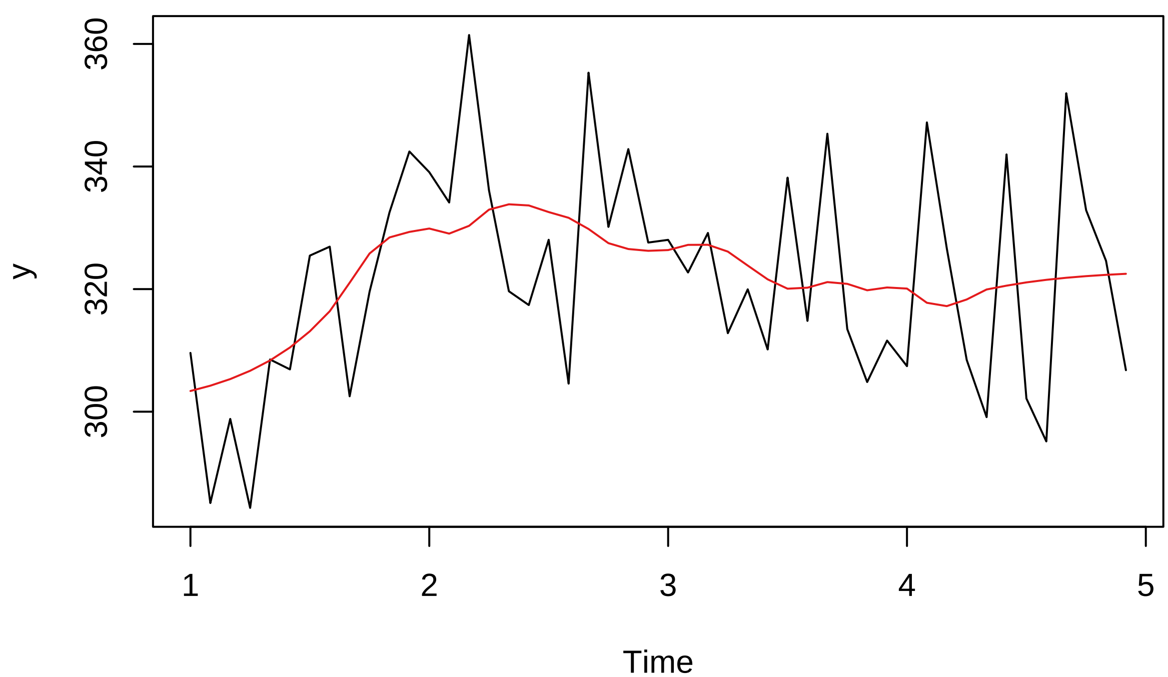
0.1.2 2. Constructing estimation and hold-out sets

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

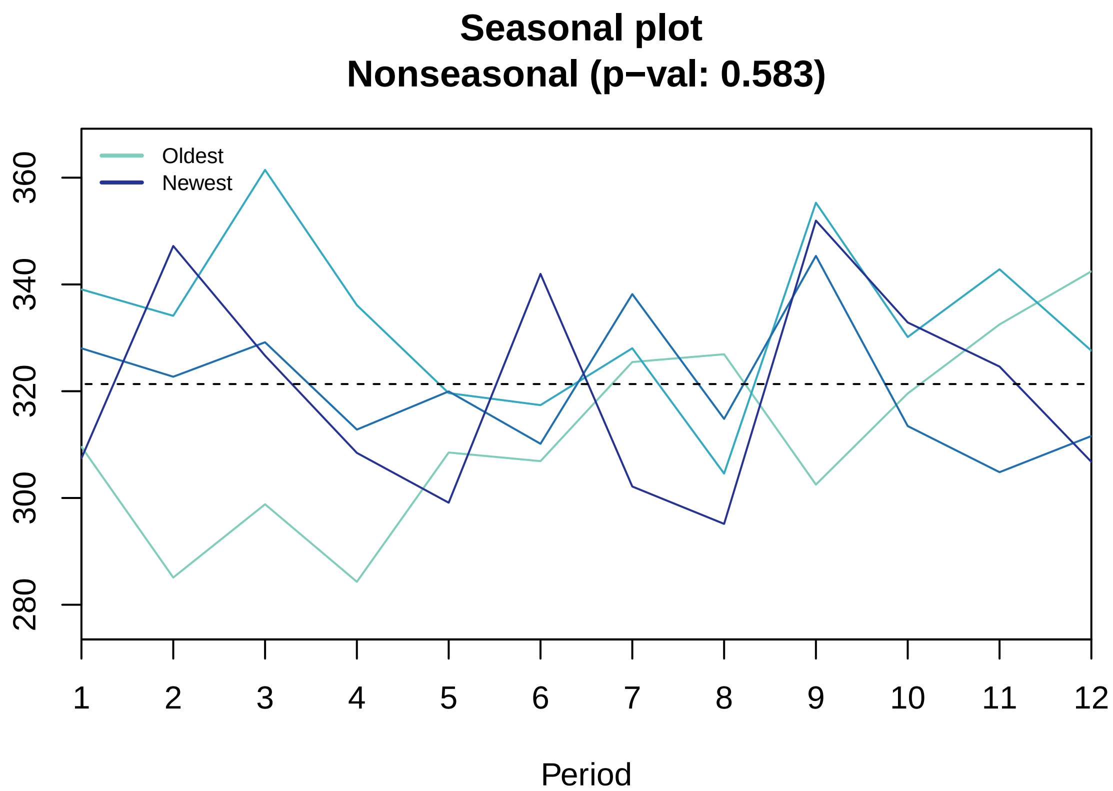
```
##        Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1 309.5927 285.0966 298.8200 284.3028 308.5171 306.8993 325.4628 326.9226
## 2 339.0753 334.1232 361.4546 336.1173 319.6460 317.3951 328.0461 304.5760
## 3 328.0343 322.7103 329.1623 312.8083 319.9644 310.1535 338.1874 314.8126
## 4 307.4285 347.2009 326.6501 308.4443 299.1128 341.9722 302.1470 295.1502
##        Sep      Oct      Nov      Dec
## 1 302.5102 319.5777 332.5051 342.4510
## 2 355.3091 330.1500 342.8376 327.5952
## 3 345.3493 313.4708 304.8354 311.6021
## 4 351.9557 332.8738 324.6155 306.7664
```

0.1.3 3. Exploration

```r
cma <- cmav(y.trn,outplot=1)
```



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




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

0.1.4 4. Forecasting

```r
ets(y.trn,model="ANN")
```

0.1.4.1 4.1 Model fitting

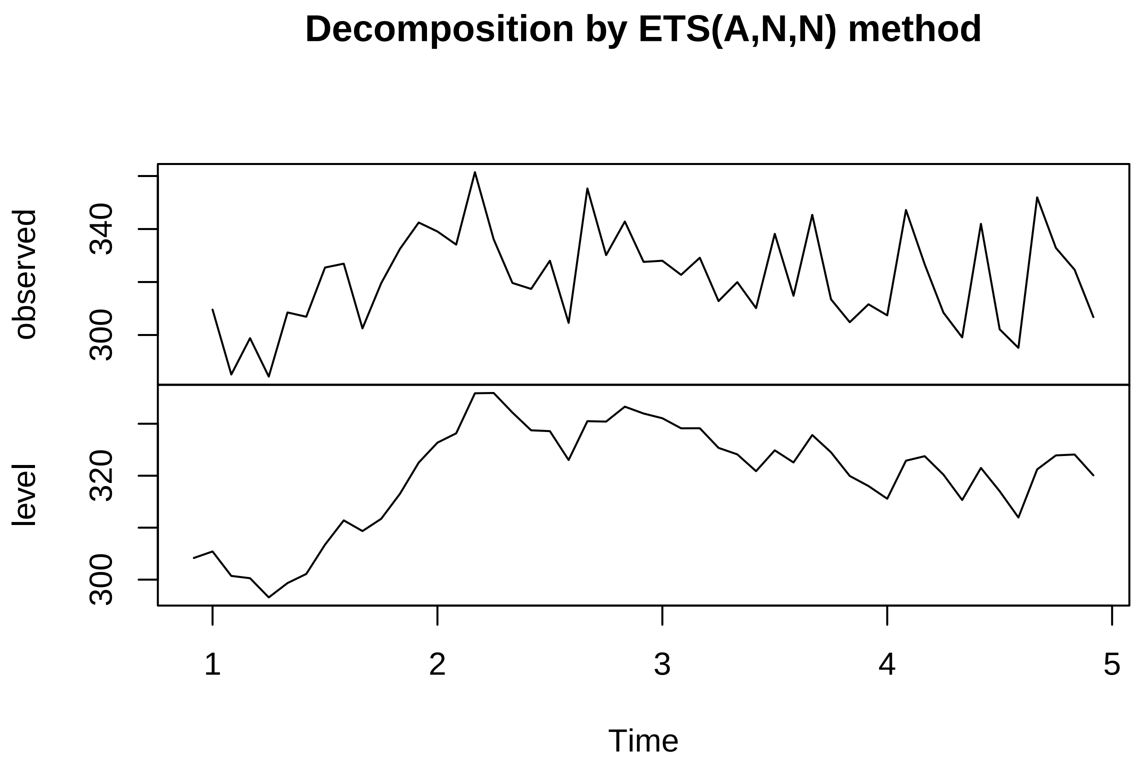
```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.2315
##
##   Initial states:
```

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

```r
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
```

```r
plot(fit1)
```




```r
class(fit1)
```

```
## [1] "ets"
```

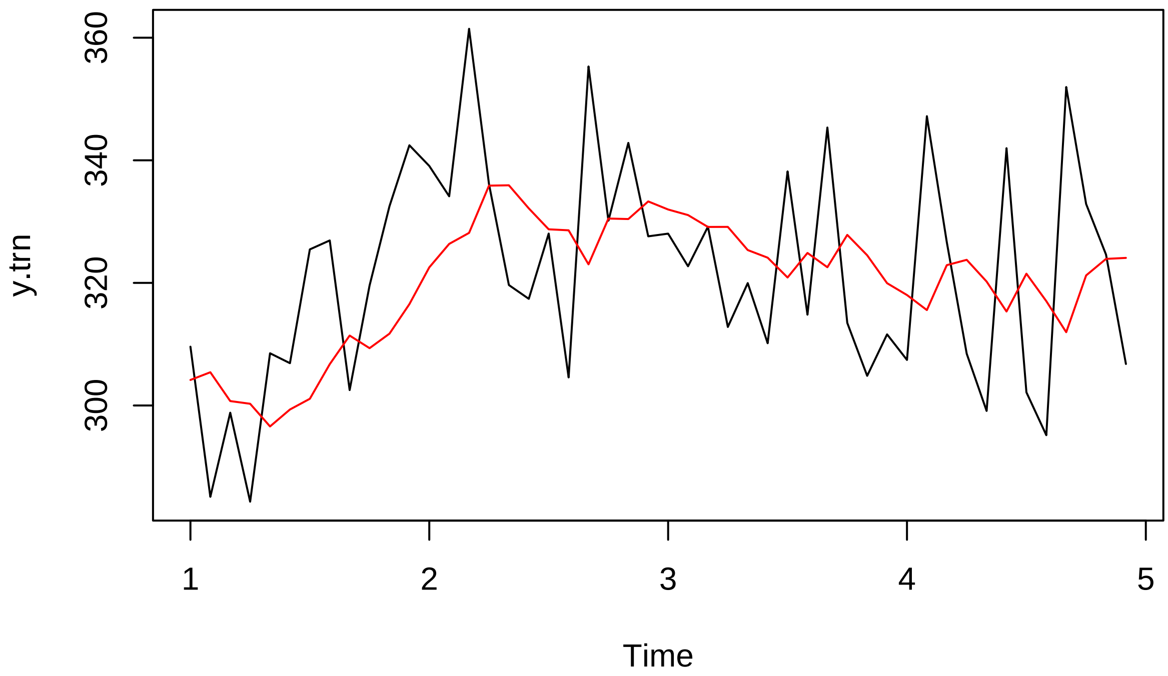
```r
names(fit1)
```

```
##  [1] "loglik"     "aic"        "bic"        "aicc"       "mse"
##  [6] "amse"       "fit"        "residuals"  "fitted"     "states"
## [11] "par"        "m"          "method"     "series"     "components"
## [16] "call"       "initstate"  "sigma2"     "x"
```

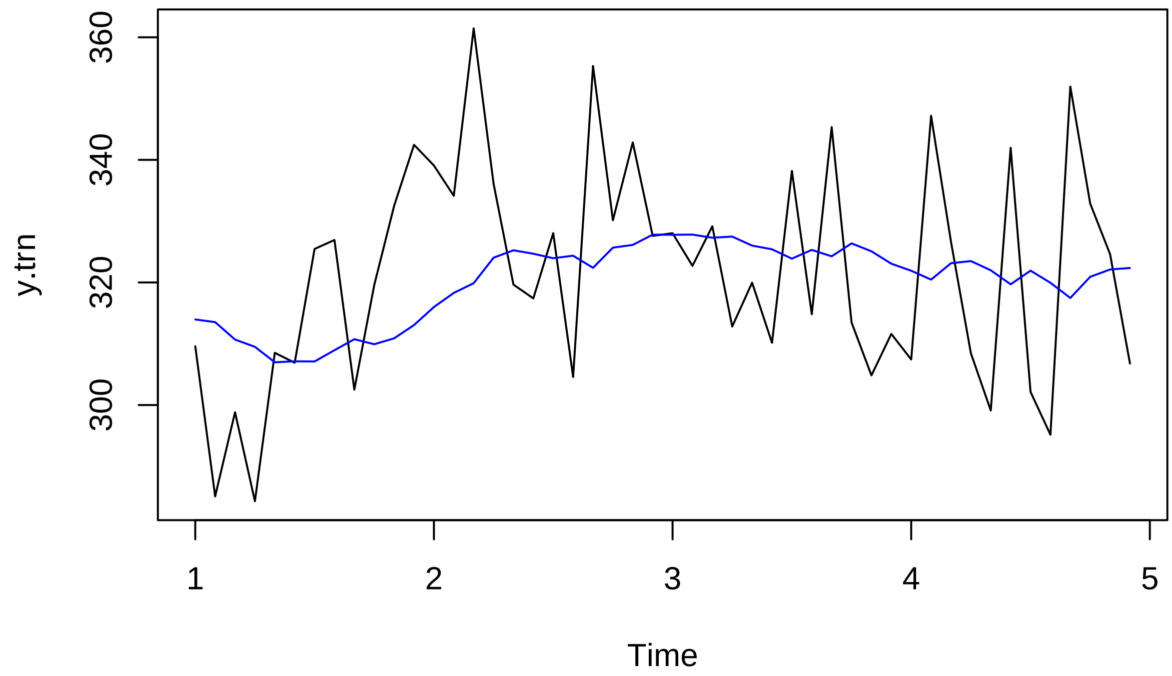
```r
fit1$fitted
```

```
##        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
```

```r
plot(y.trn)
lines(fit1$fitted, col="red")
```




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




```r
fit1$mse
```

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

```r
fit2$mse
```

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

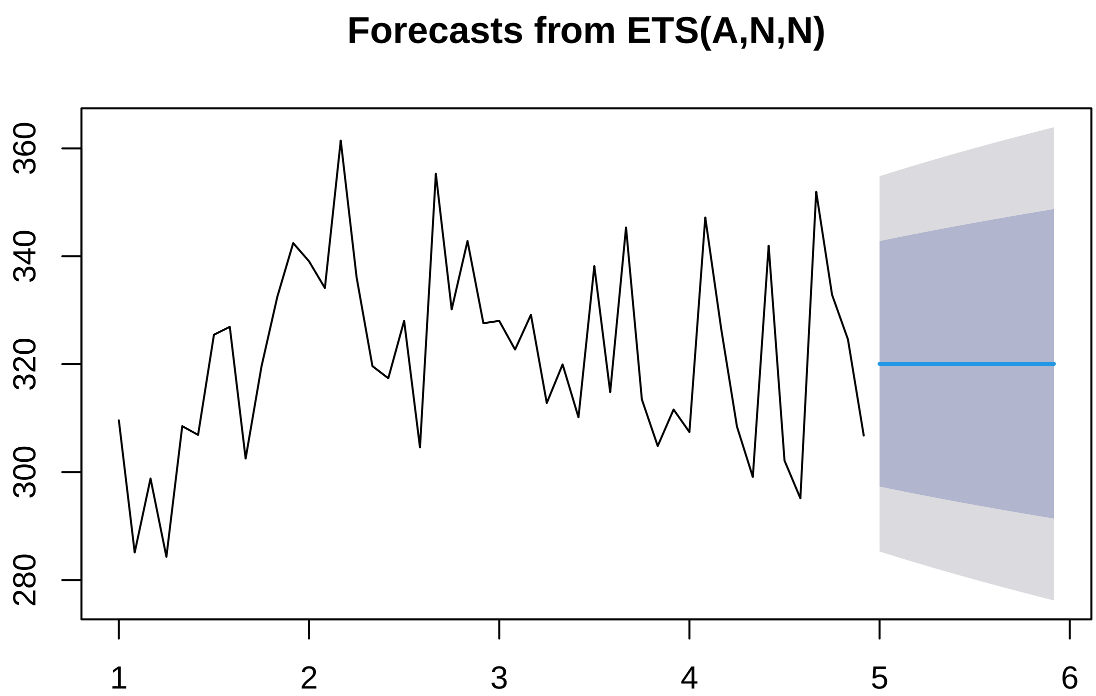
```r
frc1 <- forecast(fit1, h=12)
print(frc1)
```

0.1.4.2 4.2 Forecasting

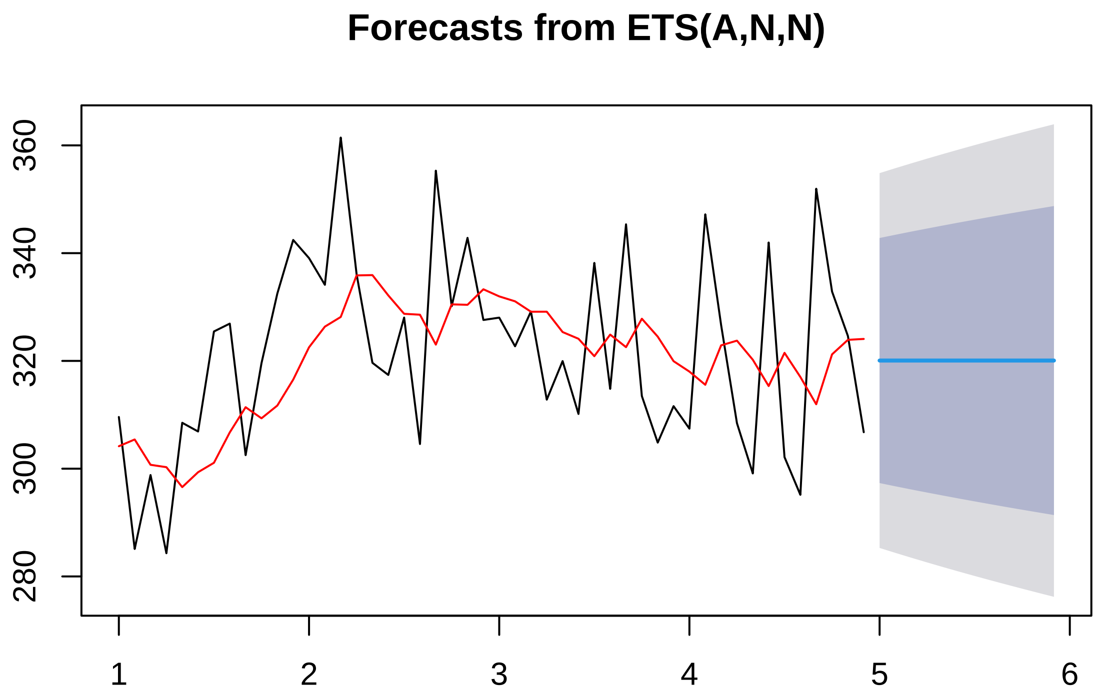
```
##        Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 5        320.0694 297.3267 342.8121 285.2875 354.8513
## Feb 5        320.0694 296.7253 343.4135 284.3676 355.7712
## Mar 5        320.0694 296.1389 343.9999 283.4708 356.6679
## Apr 5        320.0694 295.5666 344.5722 282.5955 357.5433
## 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
```

```r
plot(frc1)
```



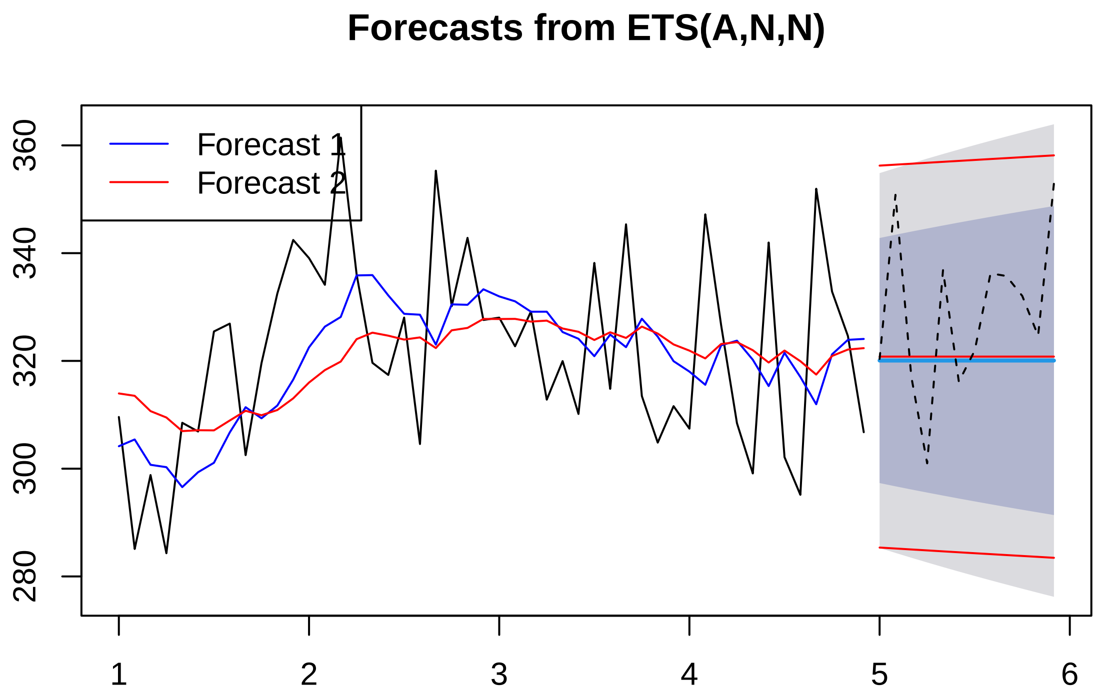

```r
plot(frc1)
lines(fit1$fitted,col="red")
```




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




```r
MAE1 <- mean(abs(y.tst - frc1$mean))
MAE2 <- mean(abs(y.tst - frc2$mean))
MAE <- c(MAE1, MAE2)
names(MAE) <- paste0("Forecast ",1:2)
round(MAE,3)
```

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

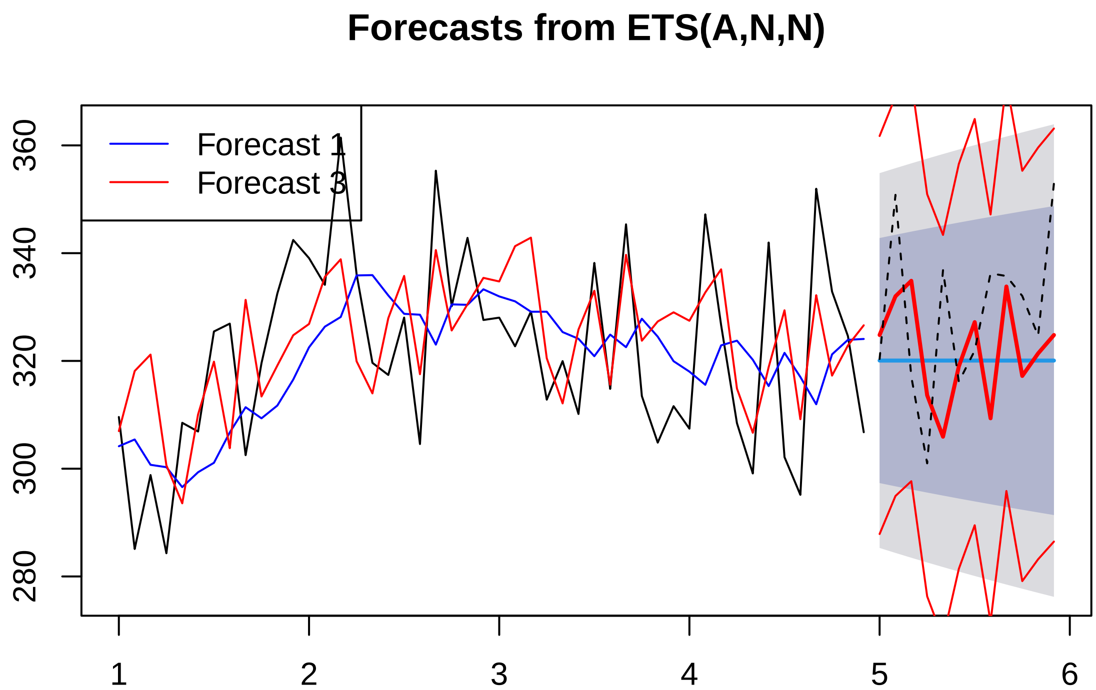
```r
fit3 <- ets(y.trn, model= "AAA", damped=TRUE)
frc3 <- forecast(fit3,h=12)
print(fit3)
```

0.1.4.3 4.3 Model selection

```
## 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
```

```r
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)
```



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

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

```r
round(MAE,3)
```

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

```r
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)
RMSE <- sqrt(MSE)
names(RMSE) <- paste0("Forecast ",1:3)
round(RMSE,3)
```

```
## Forecast 1 Forecast 2 Forecast 3
##     16.770     16.399     17.306
```

```r
crit <- array(NA,c(3,3))
print(crit)
```

```
##      [,1] [,2] [,3]
## [1,]   NA   NA   NA
## [2,]   NA   NA   NA
## [3,]   NA   NA   NA
```

```r
crit <- array(NA,c(3,3),dimnames=list(c("Forecast 1", "Forecast 2", "Forecast 3"),
c("AIC","AICc","BIC"))) # I can split lines!
print(crit)
```

```
##            AIC AICc BIC
## Forecast 1  NA   NA  NA
## Forecast 2  NA   NA  NA
## Forecast 3  NA   NA  NA
```

```r
models <- list(fit1, fit2, fit3)

for (i in 1:3) {
  crit[i, "AIC"] <- models[[i]]$aic
  crit[i, "AICc"] <- models[[i]]$aicc
  crit[i, "BIC"] <- models[[i]]$bic
}

print(crit)
```

```
##                 AIC     AICc      BIC
## Forecast 1 465.8872 466.4326 471.5008
## Forecast 2 465.7005 465.9672 469.4429
## Forecast 3 482.7128 506.2990 516.3944
```

```r
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
```

1 Exercise

"To enhance both conciseness and clarity within this report, the subsequent sections will adopt the following terminology conventions in the forthcoming code:

1. "fit" will be employed when specifying an ANN model with automatic alpha selection.

2. "fit_m1", "fit_m2" will be utilized when denoting an ANN model with an explicitly specified alpha value.

Moreover, these identical nomenclature conventions will be consistently applied to the forecasting variable, which will uniformly be denoted as "frc" in all relevant code sections."

1.1 Level B

1.1.1 1. Loading Data

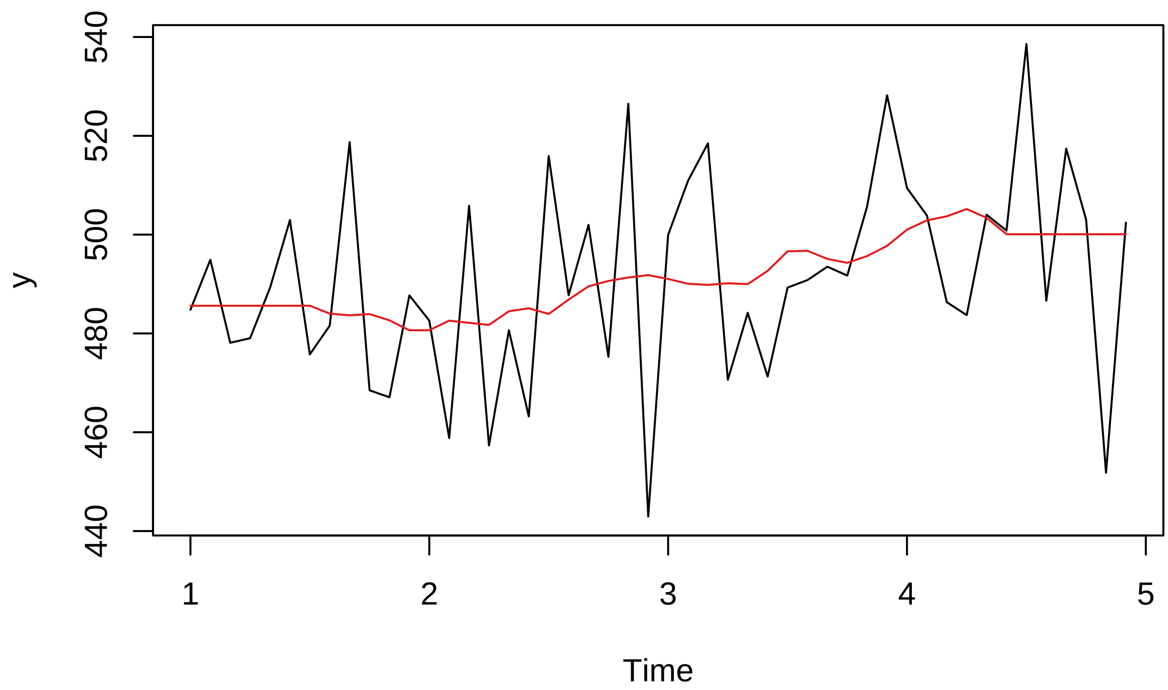
```r
y <- Y[,2]
# Transform it into a time series
y <- ts(y,frequency=12)
```

1.1.2 2. Constructing estimation and hold-out sets

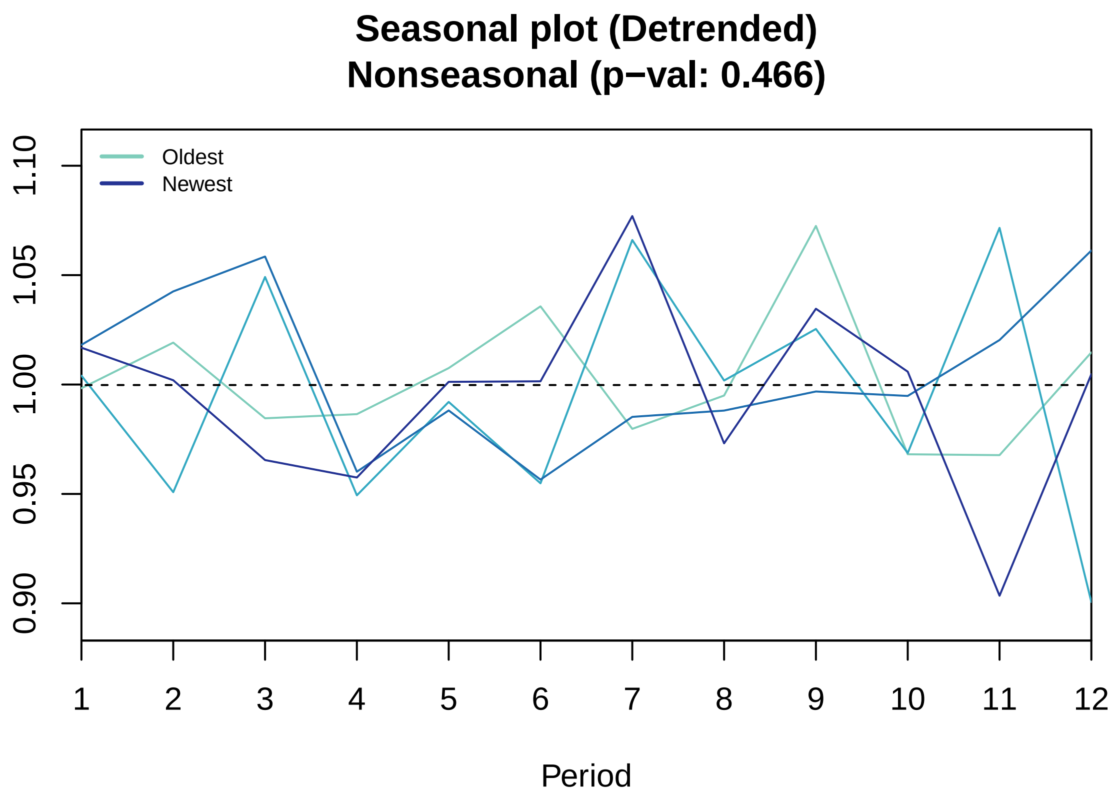
```r
y.tst <- tail(y,12)
y.trn <- head(y,48)
```

1.1.3 3. Exploration

```r
cma <- cmav(y.trn,outplot=1)
```




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




```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0)
## Evidence of seasonality: FALSE  (pval: 0.466)
```

1.1.4 4. Forecasting

```r
# Automatic Alpha ANN
fit <- ets(y.trn,model="ANN")
print(fit)
```

1.1.4.1 4.1 Model fitting

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.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
# Alpha ANN
fit_m1 <- ets(y.trn,model="ANN", alpha = 0.08 )
print(fit)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.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
# Alpha M2 ANN
fit_m2 <- ets(y.trn,model="ANN", alpha = 0.02 )
print(fit_m2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.02)
##
##   Smoothing parameters:
##     alpha = 0.02
##
##   Initial states:
##     l = 490.3465
##
##   sigma:  20.9358
##
##      AIC     AICc      BIC
## 479.7548 480.0215 483.4972
```

```r
cirt <- array(NA, c(3, 4), dimnames = list(c("Automatic", "M1", "M2"),
                                          c("MSE", "AIC", "AICc","BIC")))

models <- list(fit, fit_m1, fit_m2)

for (i in 1:3) {
```

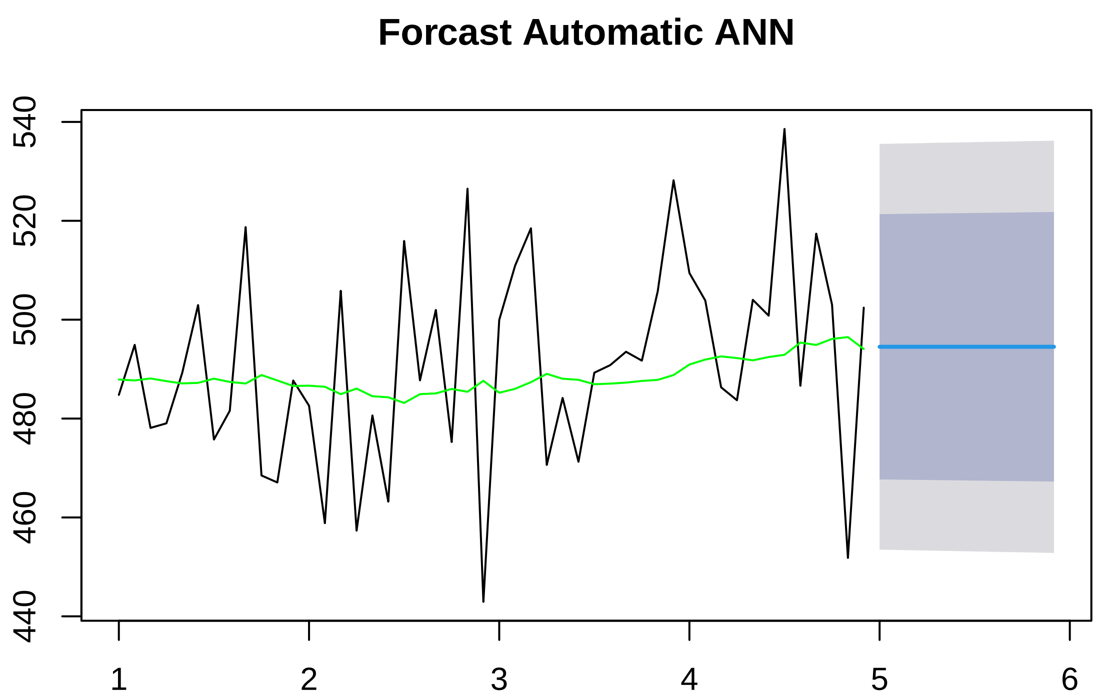
```r
  cirt[i, "MSE"] <- models[[i]]$mse
  cirt[i, "AIC"] <- models[[i]]$aic
  cirt[i, "AICc"] <- models[[i]]$aicc
  cirt[i, "BIC"] <- models[[i]]$bic
}

print(cirt)
```

```
##                MSE      AIC     AICc      BIC
## Automatic 420.0784 481.7588 482.3043 487.3724
## M1        421.3674 479.9059 480.1726 483.6483
## M2        420.0430 479.7548 480.0215 483.4972
```

1.1.4.2 4.2 Forecasting

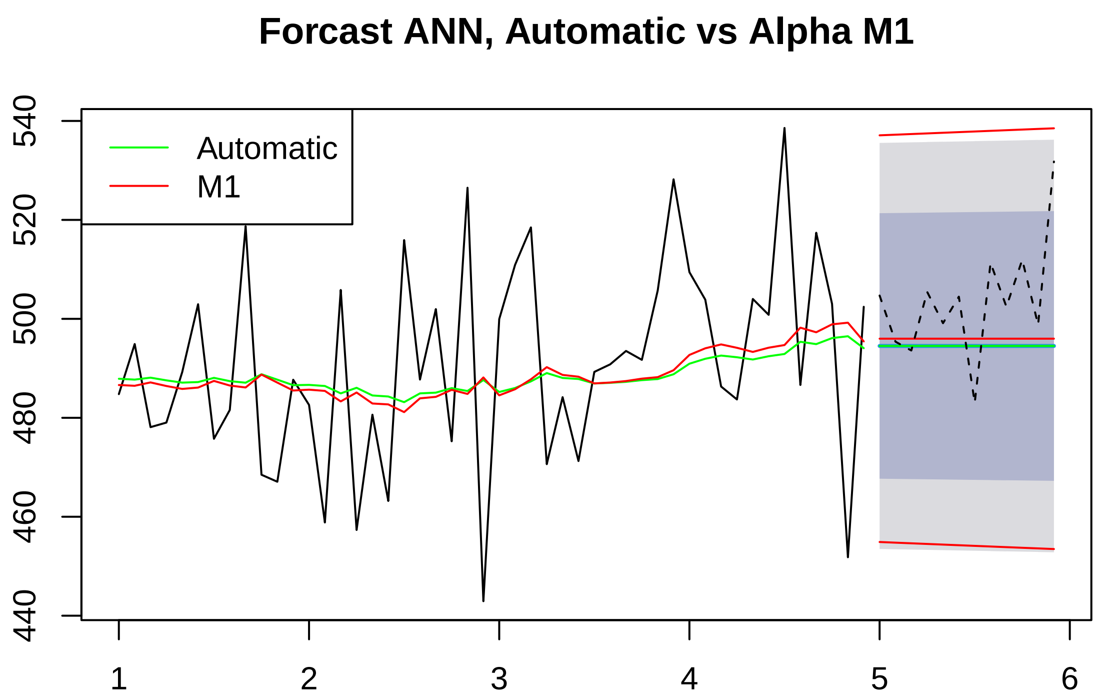
- Forecast ANN

```r
# ploting Automatic ANN
frc <- forecast(fit, h=12)
plot(frc, main = "Forcast Automatic ANN")
lines(fit$fitted,col="green")
```



- Forecast ANN and Alpha M1

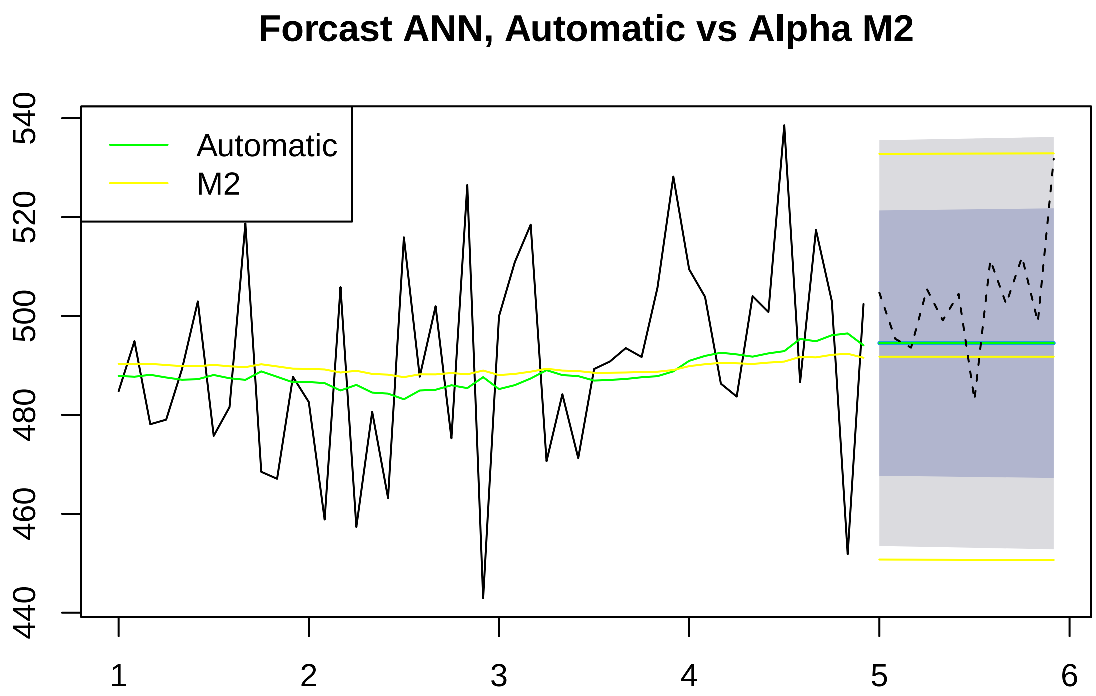
```r
# ploting M1 vs ANN
frc_m1 <- forecast(fit_m1,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M1")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red") # 95% upper
lines(y.tst,lty=2)

# legends
legend("topleft",c("Automatic","M1"),col=c("green","red"),lty=1)
```



- Forecast ANN and Alpha M2

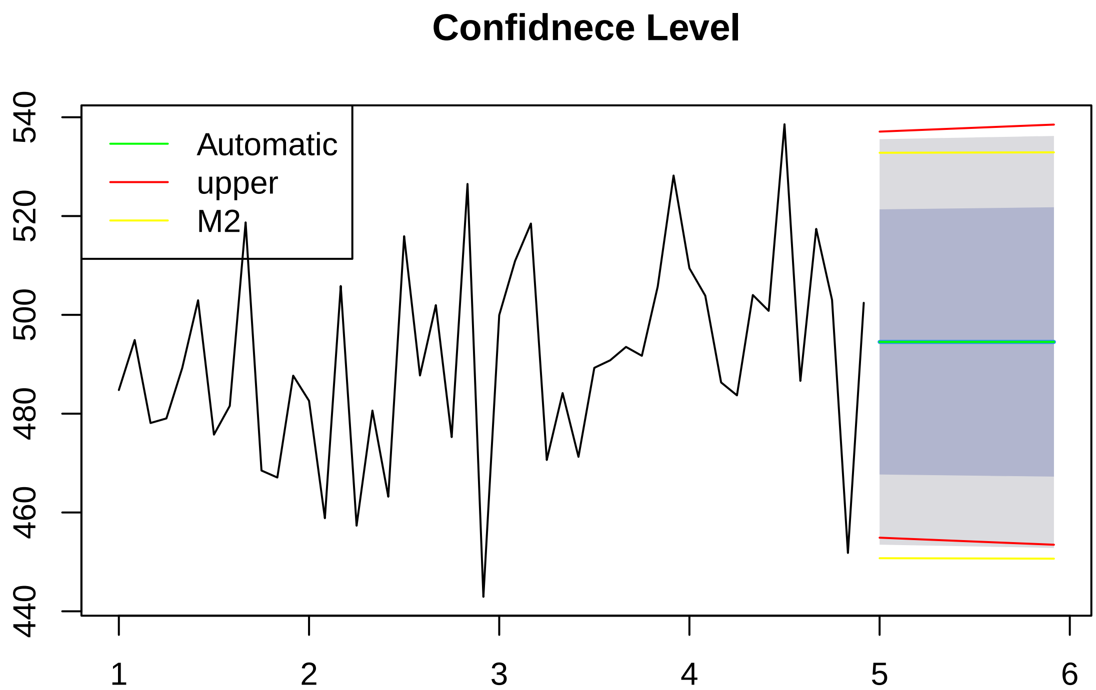
```r
# Ploting M2 vs ANN
frc_m2 <- forecast(fit_m2,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M2")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m2$fitted,col="yellow")

lines(frc_m2$mean,col="yellow")
```

```
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper
lines(y.tst,lty=2)

# Add legend to the plot
legend("topleft",c("Automatic","M2"),col=c("green","yellow"),lty=1)
```

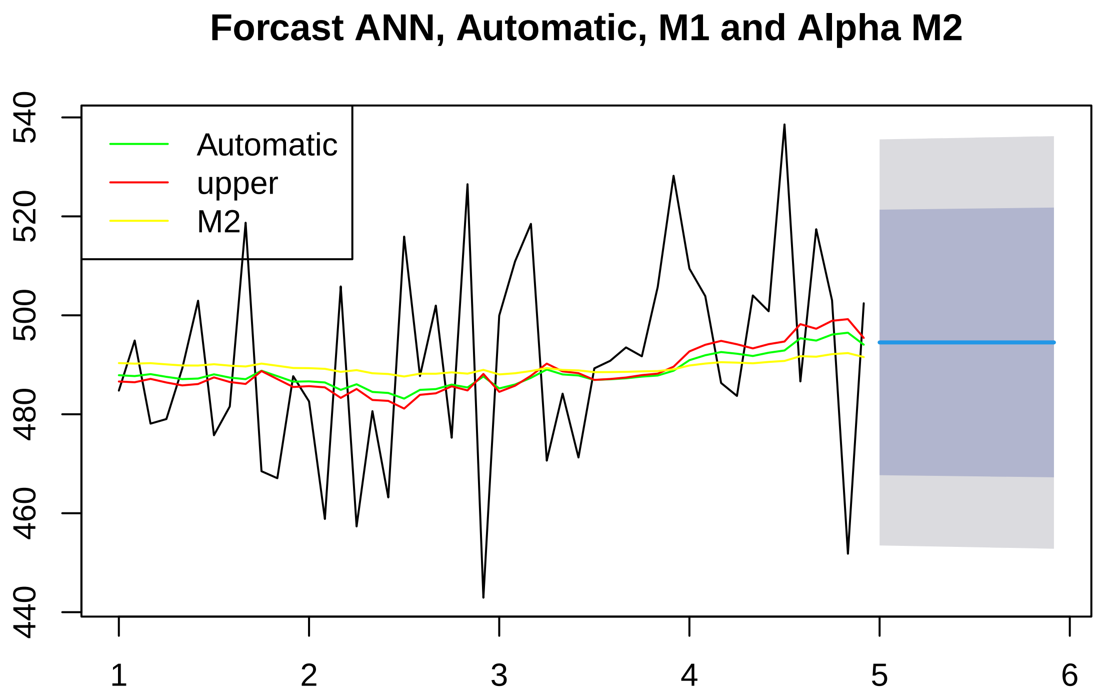


- Confidence level ANN, Alpha M1 and Alpha M2

```
# Ploting confidence level of all 3
plot(frc, main = "Confidnece Level")
lines(frc$mean,col="green")

lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red")
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper

# Add legend to the plot
legend("topleft",c("Automatic","upper","M2"),col=c("green","red","yellow"),lty=1)
```




```
plot(frc, main = "Forcast ANN, Automatic, M1 and Alpha M2")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")
lines(fit_m2$fitted,col="yellow")
legend("topleft",c("Automatic","upper","M2"),col=c("green","red","yellow"),lty=1)
```



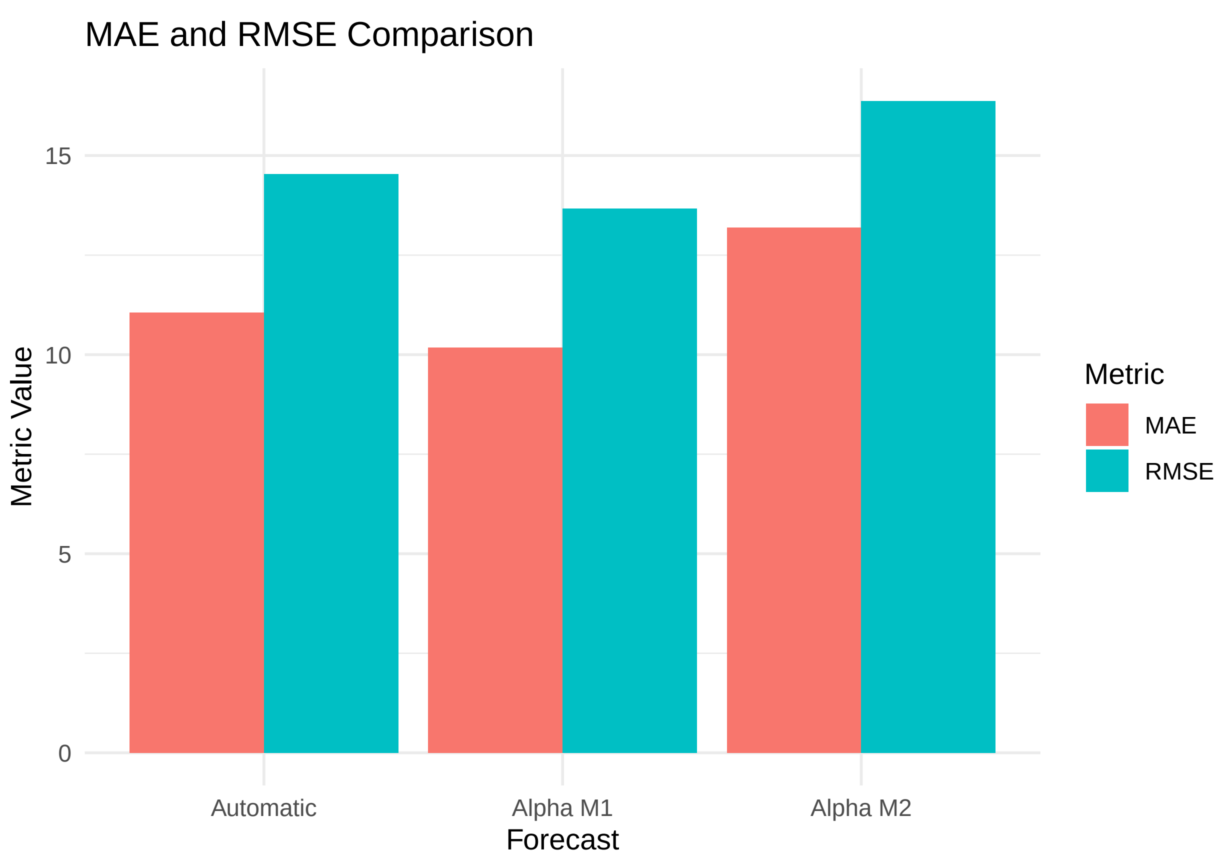

```r
# Function to calculate metrics
calculate_metrics <- function(y, frc) {
  MAE <- mean(abs(y - frc$mean))
  MSE <- mean((y - frc$mean)^2)
  RMSE <- sqrt(MSE)
  return(list(MAE = MAE, MSE = MSE, RMSE = RMSE))
}

# Calculate metrics for different forecasts
metrics_a <- calculate_metrics(y.tst, frc)
metrics_m1 <- calculate_metrics(y.tst, frc_m1)
metrics_m2 <- calculate_metrics(y.tst, frc_m2)

# Create a matrix for the metrics
metrics_matrix <- matrix(c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
                          metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE,
                         metrics_a$MSE, metrics_m1$MSE, metrics_m2$MSE),
                         ncol = 3, byrow = TRUE)

# Add row and column names
rownames(metrics_matrix) <- c("MAE", "MSE", "RMSE")
colnames(metrics_matrix) <- c("Automatic", "Alpha M1", "Alpha M2")
```

```r
# Display the metrics matrix
metrics_matrix
```

1.1.4.3 4.3 Model selection

```
##      Automatic  Alpha M1  Alpha M2
## MAE   11.05761  10.17653  13.19261
## MSE   14.53394  13.67233  16.37449
## RMSE 211.23547 186.93268 268.12393
```

```r
# Create a data frame from the metrics matrix, excluding MSE
metrics_df <- data.frame(
  Metric = rep(c("MAE", "RMSE"), each = 3),
  Value = c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
            metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE),
  Forecast = rep(c("Automatic", "Alpha M1", "Alpha M2"), times = 2)
)
metrics_df$Forecast <- factor(metrics_df$Forecast, levels = c("Automatic", "Alpha M1", "Alpha M2"))

# Create a bar plot
ggplot(metrics_df, aes(x = Forecast, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "MAE and RMSE Comparison",
       y = "Metric Value") +
  theme_minimal()
```



1.1.5 5. Answers

- Which one is best using your judgement?

  Based on an evaluation of the in-sample data, it is observed that the **Automatic** model has performed commendably. This model effectively filters out noise and outliers, resulting in a smoother representation of the underlying level. Conversely, the M2 model, while smoothing out a significant amount of noise and outliers, may not fully capture the nuances of the level. The upper model, although not consistently following the level, provides an alternative perspective

- Which one is best using errors?

  In terms of error metrics such as AIC, MSE, RMSE, and others, it is notable that the **M2** model yields the lowest errors. Despite its occasional deviations from the level, its overall predictive accuracy, as indicated by the error metrics, is superior to the other models.

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

  Contrary to the in-sample results, the out-of-sample data analysis reveals that the selected model may not be the optimal choice. In this context, the **M1**model demonstrates superior performance, as it yields the lowest RMSE and MAE values. This suggests that the upper model might generalize better to unseen data points, capturing the underlying patterns more effectively.

1.2 LevelShift

1.2.1 1. Loading Data

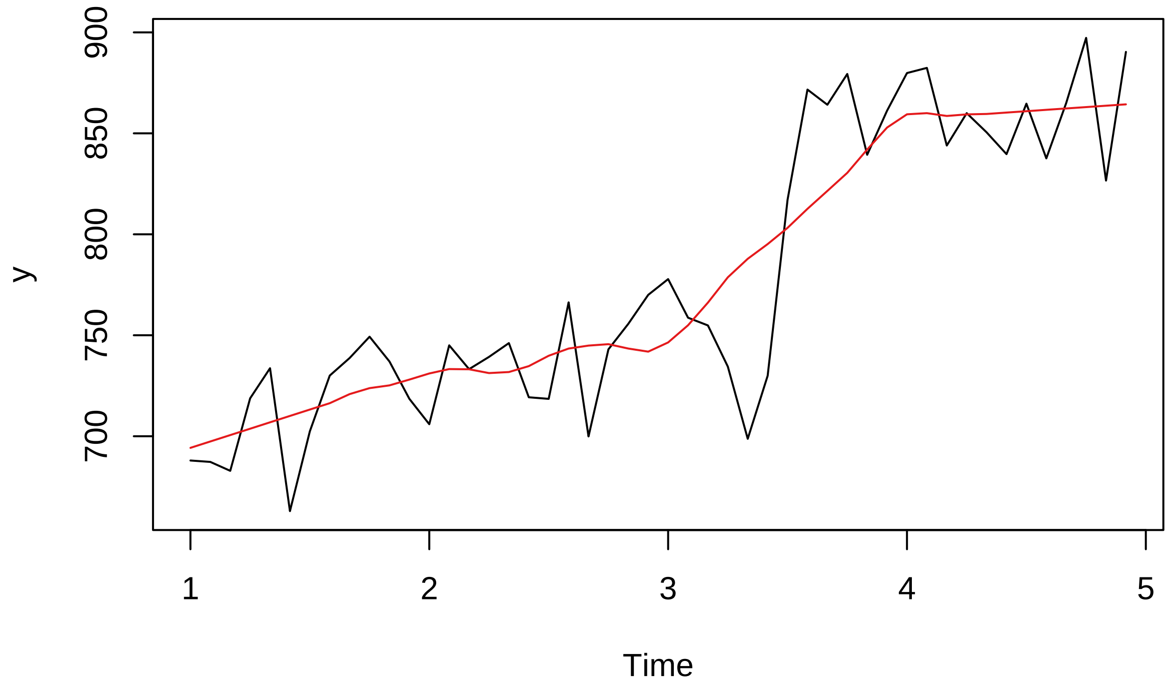
```
y <- Y[,3]
# Transform it into a time series
y <- ts(y,frequency=12)
```

1.2.2 2. Constructing estimation and hold-out sets

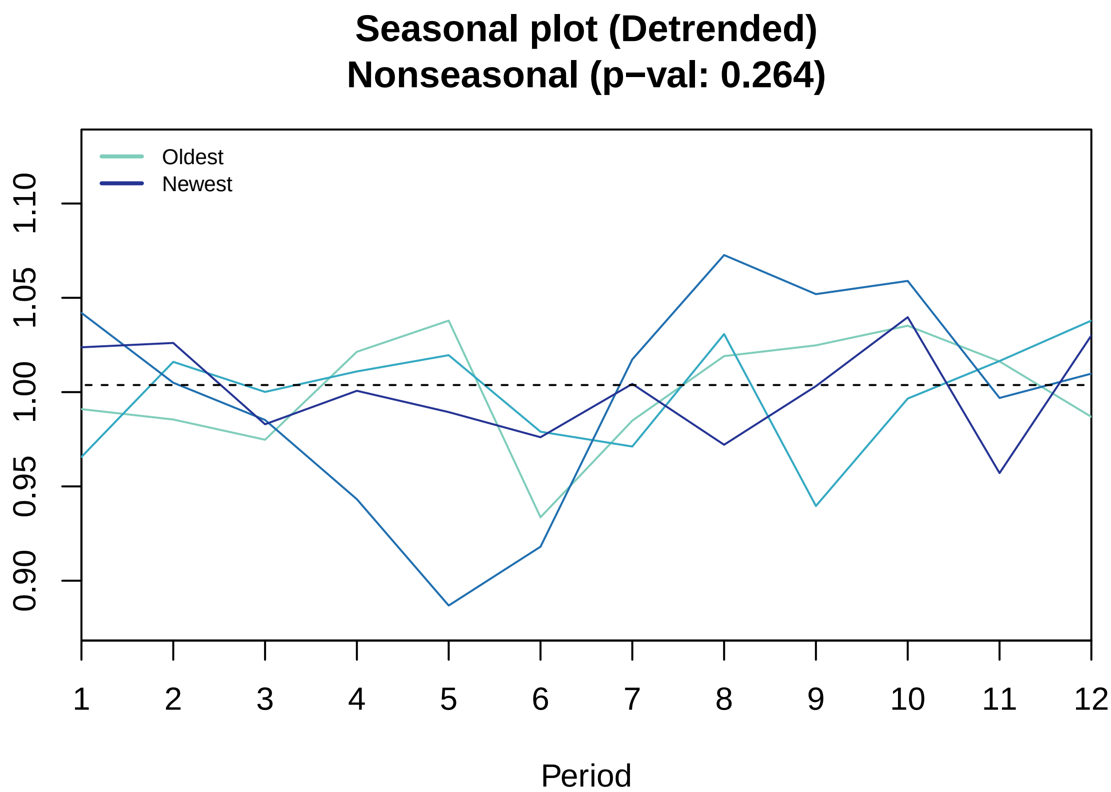
```
y.tst <- tail(y,12)
y.trn <- head(y,48)
```

1.2.3 3. Exploration

```
cma <- cmav(y.trn,outplot=1)
```




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




```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0)
## Evidence of seasonality: FALSE  (pval: 0.264)
```

1.2.4 4. Forecasting

```
# Automatic Alpha ANN
fit <- ets(y.trn,model="ANN")
print(fit)
```

1.2.4.1 4.1 Model fitting

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.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
```

```r
# Alpha M1 ANN
fit_m1 <- ets(y.trn,model="ANN", alpha = 0.8 )
print(fit_m1)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.8)
##
##   Smoothing parameters:
##     alpha = 0.8
##
##   Initial states:
##     l = 687.9913
##
##   sigma:  32.5357
##
##      AIC     AICc      BIC
## 522.0791 522.3458 525.8215
```

```r
# Alpha M2 ANN
fit_m2 <- ets(y.trn,model="ANN", alpha = 0.4 )
print(fit_m2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.4)
##
##   Smoothing parameters:
##     alpha = 0.4
##
##   Initial states:
##     l = 692.7714
##
##   sigma:  33.6499
##
##      AIC     AICc      BIC
## 525.3118 525.5784 529.0542
```

```r
cirt <- array(NA, c(3, 4), dimnames = list(c("Automatic", "M1", "M2"),
                                         c("MSE", "AIC", "AICc","BIC")))

models <- list(fit, fit_m1, fit_m2)

for (i in 1:3) {
```

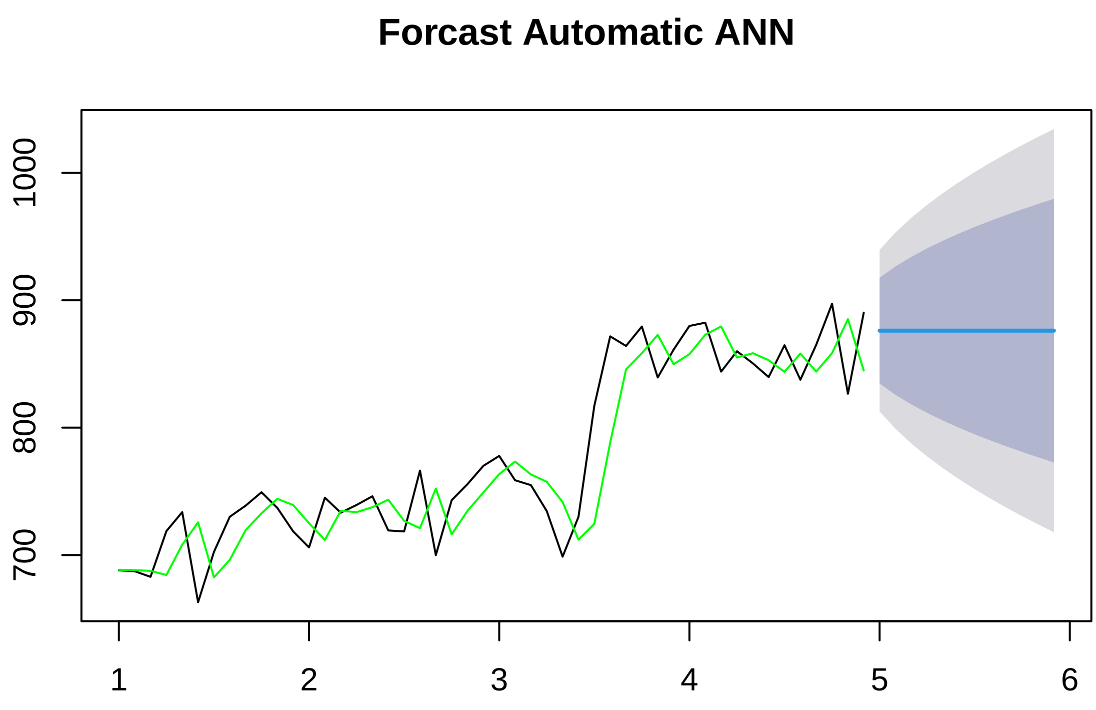
```
  cirt[i, "MSE"] <- models[[i]]$mse
  cirt[i, "AIC"] <- models[[i]]$aic
  cirt[i, "AICc"] <- models[[i]]$aicc
  cirt[i, "BIC"] <- models[[i]]$bic
}

print(cirt)
```

```
##                MSE      AIC     AICc      BIC
## Automatic 1005.374 523.6472 524.1926 529.2608
## M1        1014.462 522.0791 522.3458 525.8215
## M2        1085.136 525.3118 525.5784 529.0542
```

1.2.4.2 4.2 Forecasting

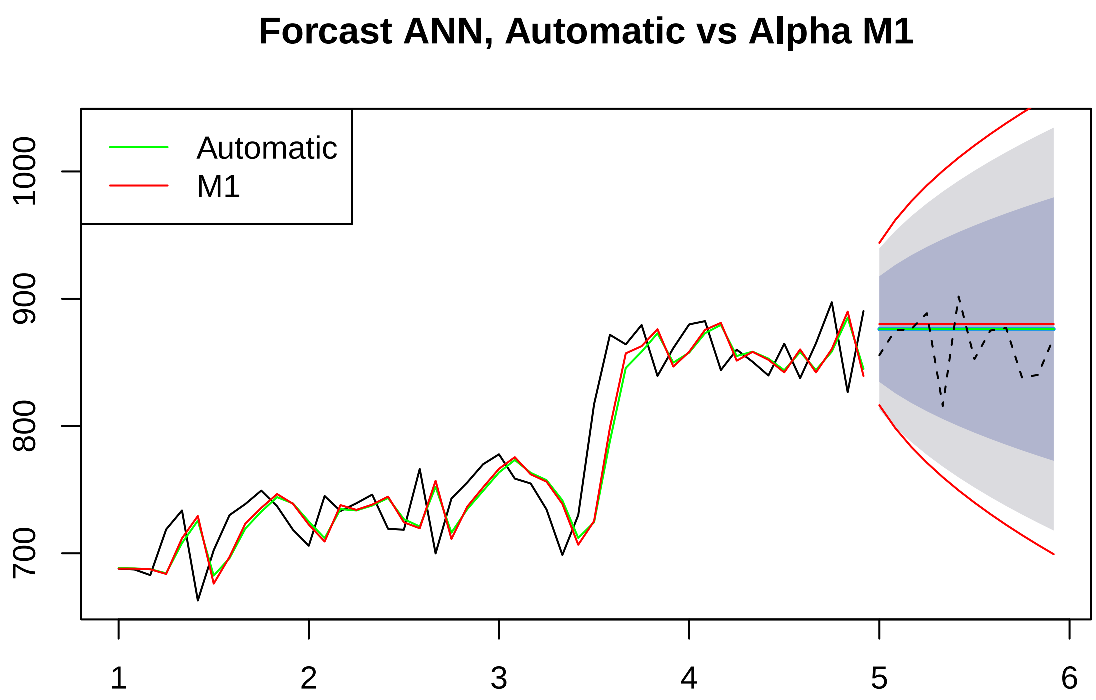
- Forecast ANN

```
# ploting Automatic ANN
frc <- forecast(fit, h=12)
plot(frc, main = "Forcast Automatic ANN")
lines(fit$fitted,col="green")
```



- Forecast ANN and Alpha M1

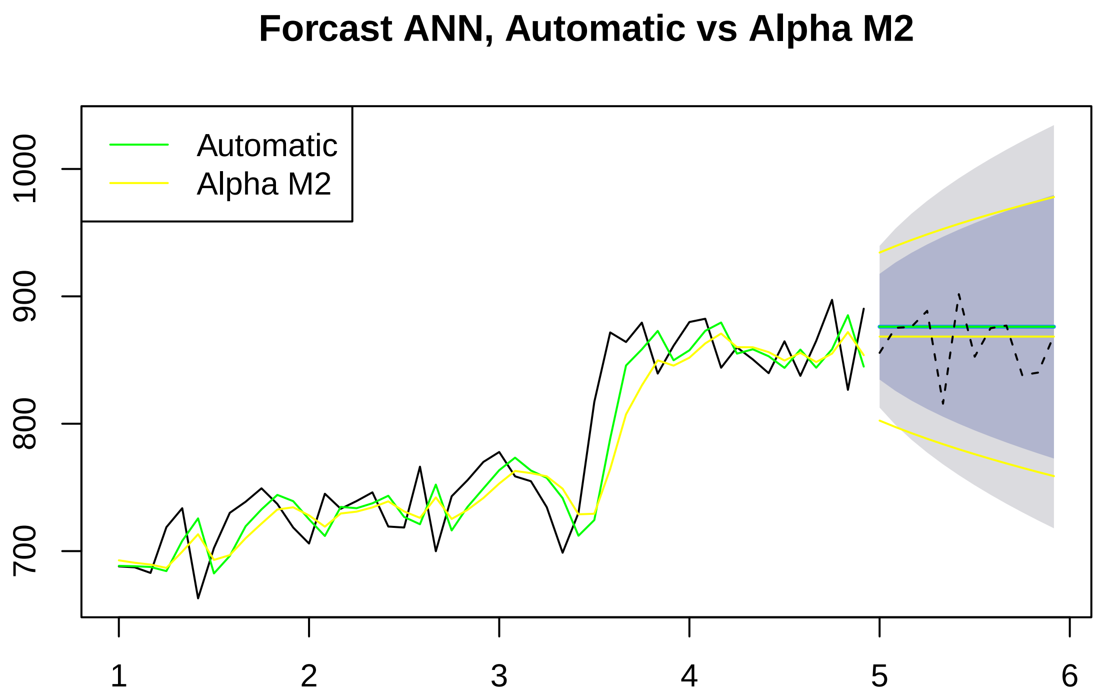
```r
# ploting M1 vs ANN
frc_m1 <- forecast(fit_m1,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M1")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red") # 95% upper
lines(y.tst,lty=2)

# legends
legend("topleft",c("Automatic","M1"),col=c("green","red"),lty=1)
```



- Forecast ANN and Alpha M2

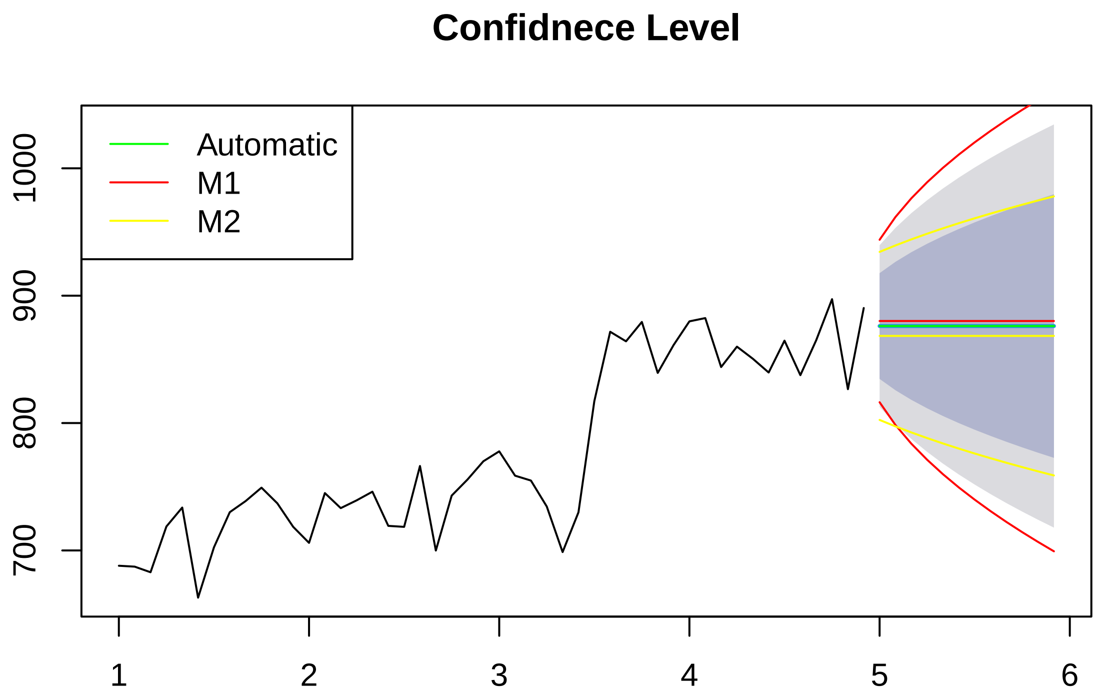
```r
# Ploting M2 vs ANN
frc_m2 <- forecast(fit_m2,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M2")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m2$fitted,col="yellow")

lines(frc_m2$mean,col="yellow")
```

```r
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper
lines(y.tst,lty=2)

# Add legend to the plot
legend("topleft",c("Automatic","Alpha M2"),col=c("green","yellow"),lty=1)
```

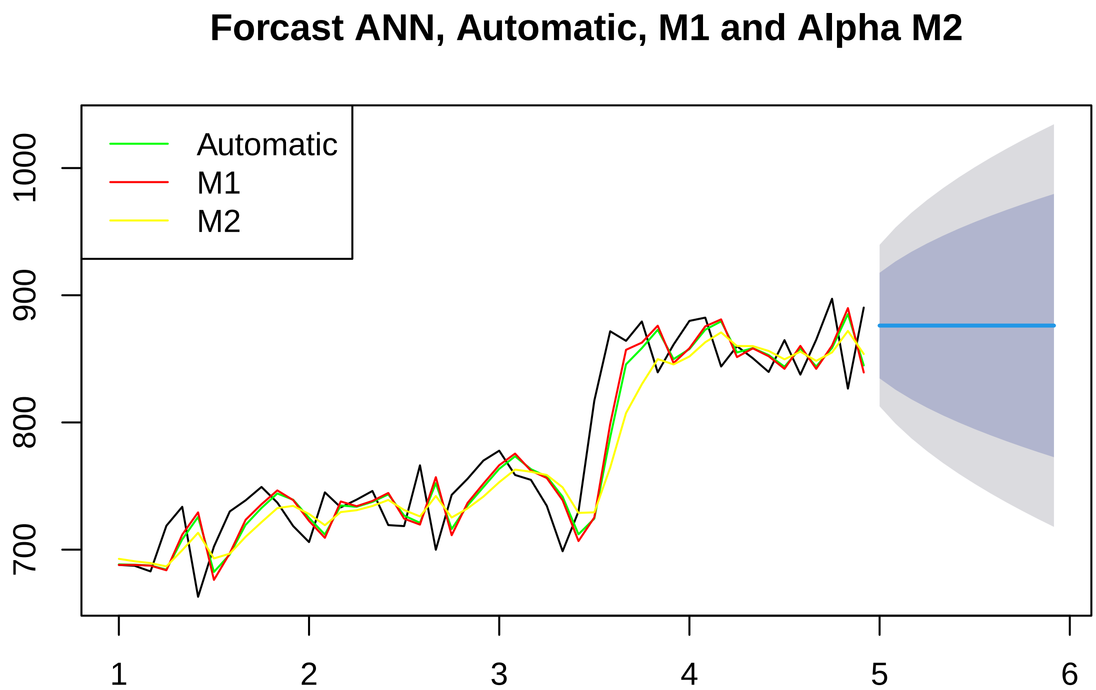


- Confidence level ANN, Alpha M1 and Alpha M2

```r
# Ploting confidence level of all 3
plot(frc, main = "Confidnece Level")
lines(frc$mean,col="green")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red")

lines(frc_m2$mean,col="yellow")
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper

# Add legend to the plot
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```




```r
plot(frc, main = "Forcast ANN, Automatic, M1 and Alpha M2")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")
lines(fit_m2$fitted,col="yellow")
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```



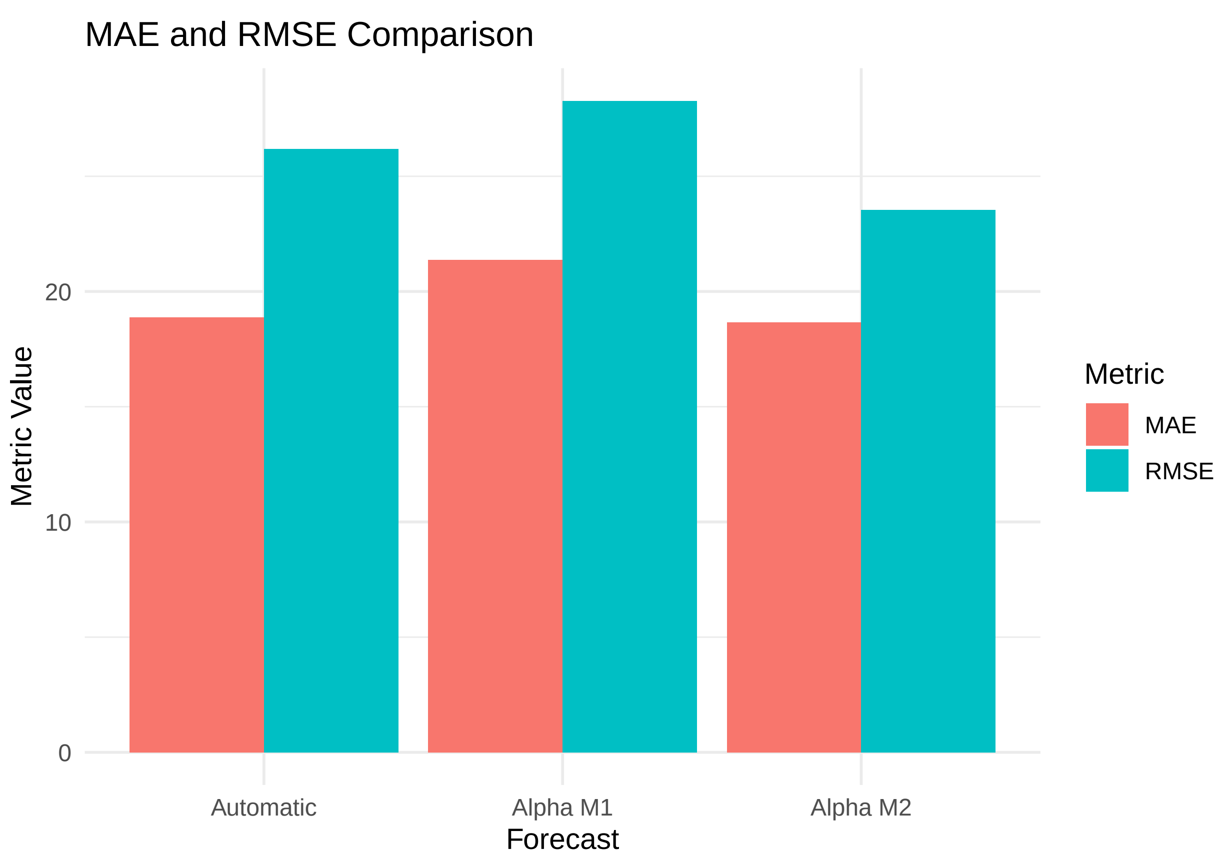

```r
# Function to calculate metrics
calculate_metrics <- function(y, frc) {
  MAE <- mean(abs(y - frc$mean))
  MSE <- mean((y - frc$mean)^2)
  RMSE <- sqrt(MSE)
  return(list(MAE = MAE, MSE = MSE, RMSE = RMSE))
}

# Calculate metrics for different forecasts
metrics_a <- calculate_metrics(y.tst, frc)
metrics_m1 <- calculate_metrics(y.tst, frc_m1)
metrics_m2 <- calculate_metrics(y.tst, frc_m2)

# Create a matrix for the metrics
metrics_matrix <- matrix(c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
                           metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE,
                          metrics_a$MSE, metrics_m1$MSE, metrics_m2$MSE),
                          ncol = 3, byrow = TRUE)

# Add row and column names
rownames(metrics_matrix) <- c("MAE", "MSE", "RMSE")
colnames(metrics_matrix) <- c("Automatic", "Alpha M1", "Alpha M2")
```

```r
# Display the metrics matrix
metrics_matrix
```

1.2.4.3 4.3 Model selection

```
##      Automatic  Alpha M1  Alpha M2
## MAE   18.87846  21.36188  18.66568
## MSE   26.19250  28.27705  23.53792
## RMSE 686.04709 799.59154 554.03354
```

```r
# Create a data frame from the metrics matrix, excluding MSE
metrics_df <- data.frame(
  Metric = rep(c("MAE", "RMSE"), each = 3),
  Value = c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
            metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE),
  Forecast = rep(c("Automatic", "Alpha M1", "Alpha M2"), times = 2)
)
metrics_df$Forecast <- factor(metrics_df$Forecast, levels = c("Automatic", "Alpha M1", "Alpha M2"))

# Create a bar plot
ggplot(metrics_df, aes(x = Forecast, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "MAE and RMSE Comparison",
       y = "Metric Value") +
  theme_minimal()
```



1.2.5 5. Answers

- Which one is best using your judgement?

  In assessing the performance of different models for the "LevelShift" component within the in-sample data, it is evident that the M2 model is the preferred choice. Despite exhibiting the ability to effectively follow the level and filter out noise and outliers, it is acknowledged that the M2 model does not yield optimal error metrics. However, prioritizing the ability to capture the level and mitigate noise appears to be paramount in this context. So **M2** model

- Which one is best using errors?

  Analyzing error metrics, it is evident that the upper model consistently outperforms the other models in terms of AIC and other error measures, except for MSE, where it lags behind the M2 model. Despite the MSE difference, the **M1** model excels in most error aspects.

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

  Contrary to the in-sample findings, the out-of-sample analysis reveals that the selected model may not be the optimal choice for forecasting the "LevelShift" component. In this case, the automatic model outperforms others by yielding the best RMSE and MAE values, which are also in proximity to the mean. This suggests that, when considering out-of-sample performance, the **Automatic** model demonstrates better predictive accuracy.

1.3 Trend A

1.3.1 1. Loading Data

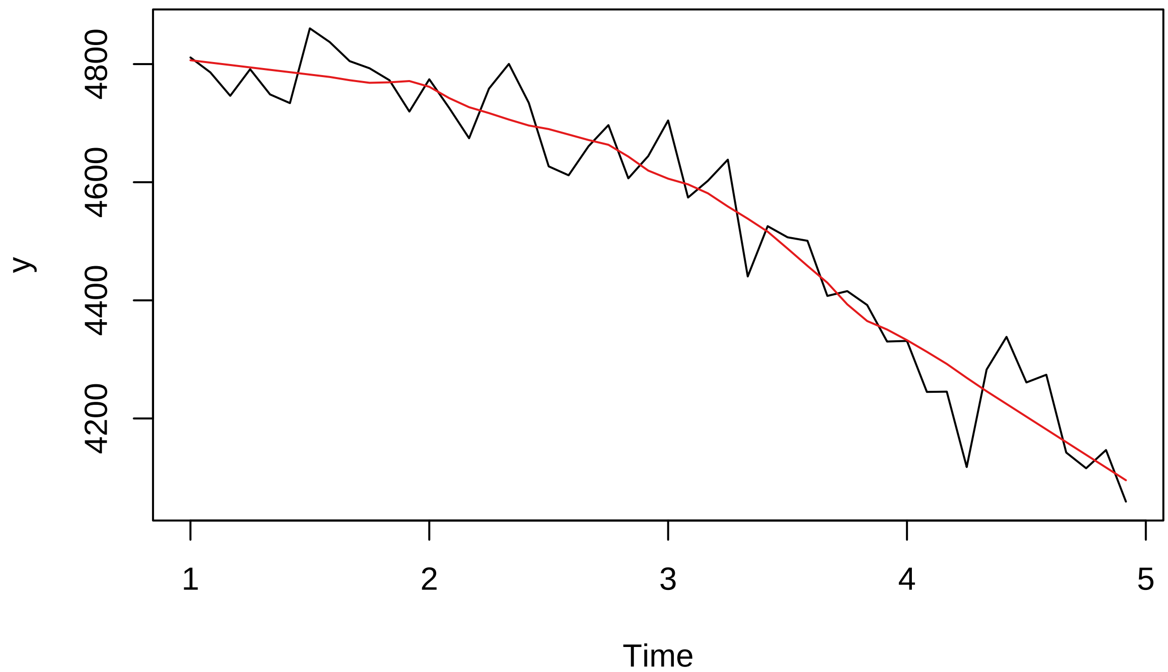
```
y <- Y[,4]
# Transform it into a time series
y <- ts(y,frequency=12)
```

1.3.2 2. Constructing estimation and hold-out sets

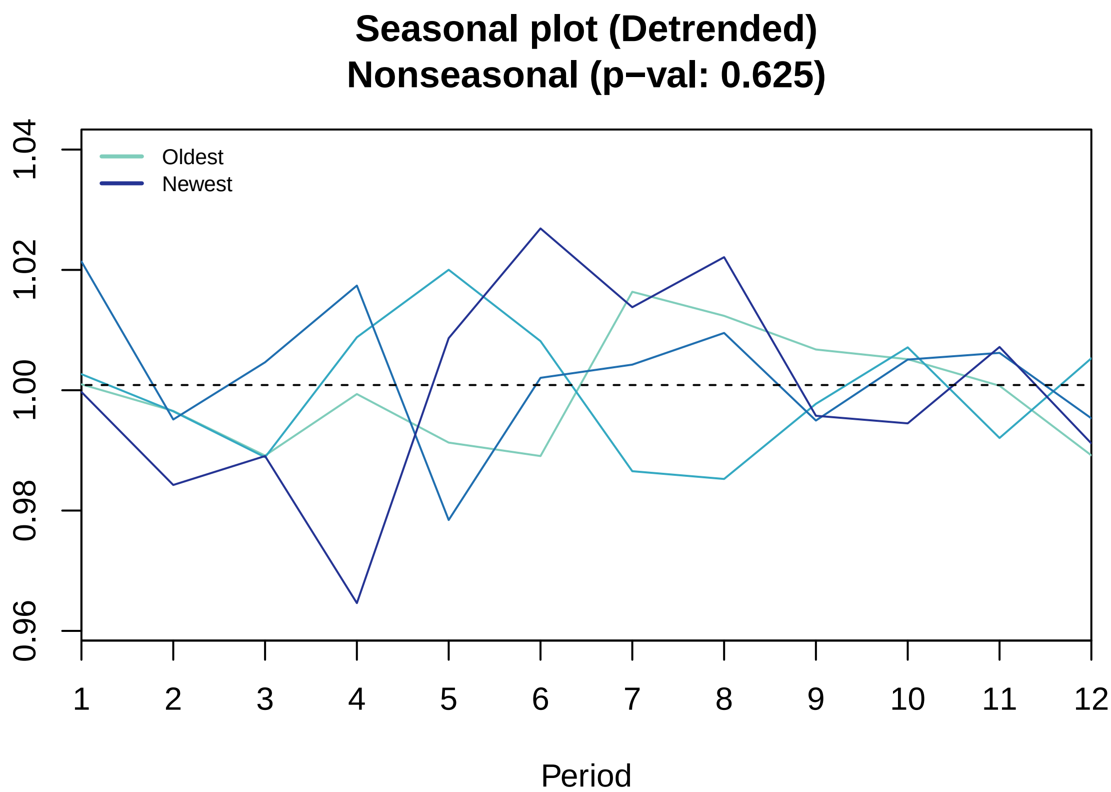
```
y.tst <- tail(y,12)
y.trn <- head(y,48)
```

1.3.3 3. Exploration

```
cma <- cmav(y.trn,outplot=1)
```




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




```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0)
## Evidence of seasonality: FALSE  (pval: 0.625)
```

1.3.4 4. Forecasting

```r
# Automatic Alpha ANN
fit <- ets(y.trn,model="ANN")
print(fit)
```

1.3.4.1 4.1 Model fitting

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.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
```

```r
# Alpha M1 ANN
fit_m1 <- ets(y.trn,model="ANN", alpha = 0.7 )
print(fit_m1)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.7)
##
##   Smoothing parameters:
##     alpha = 0.7
##
##   Initial states:
##     l = 4800.9086
##
##   sigma:  69.5128
##
##      AIC     AICc      BIC
## 594.9598 595.2265 598.7022
```

```r
# Alpha M2 ANN
fit_m2 <- ets(y.trn,model="ANN", alpha = 0.4 )
print(fit_m2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.4)
##
##   Smoothing parameters:
##     alpha = 0.4
##
##   Initial states:
##     l = 4788.9771
##
##   sigma:  74.2098
##
##      AIC     AICc      BIC
## 601.2368 601.5035 604.9792
```

```r
cirt <- array(NA, c(3, 4), dimnames = list(c("Automatic", "M1", "M2"),
                                          c("MSE", "AIC", "AICc","BIC")))

models <- list(fit, fit_m1, fit_m2)

for (i in 1:3) {
```

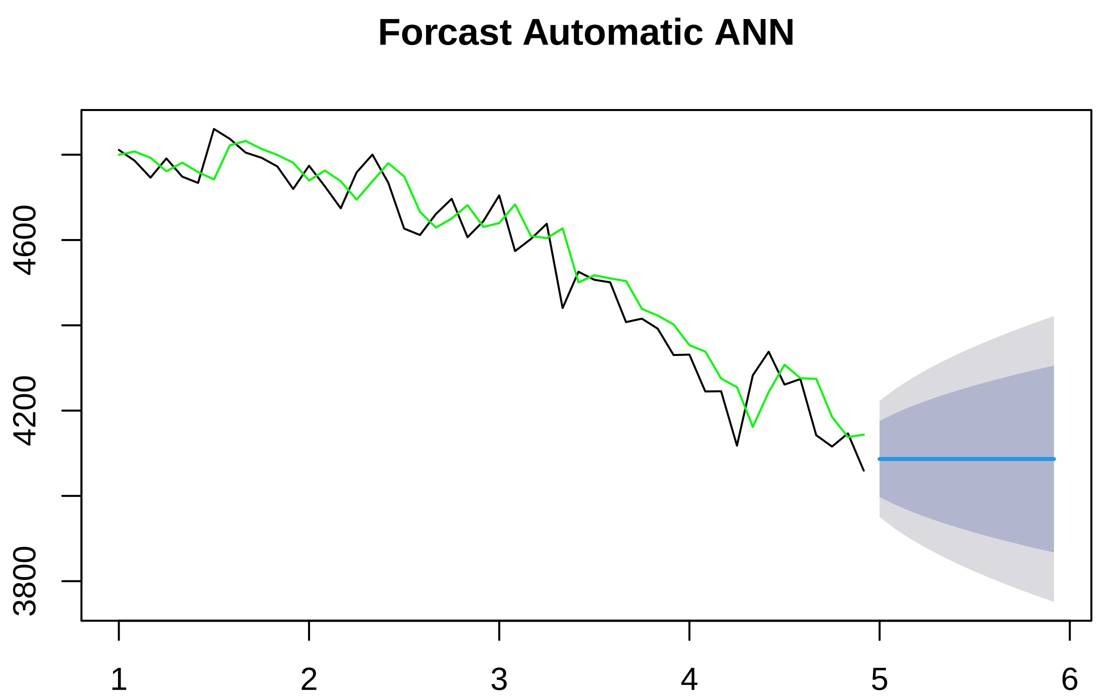
```r
  cirt[i, "MSE"] <- models[[i]]$mse
  cirt[i, "AIC"] <- models[[i]]$aic
  cirt[i, "AICc"] <- models[[i]]$aicc
  cirt[i, "BIC"] <- models[[i]]$bic
}

print(cirt)
```

```
##                MSE      AIC     AICc      BIC
## Automatic 4628.035 596.9323 597.4777 602.5459
## M1        4630.694 594.9598 595.2265 598.7022
## M2        5277.633 601.2368 601.5035 604.9792
```

1.3.4.2 4.2 Forecasting

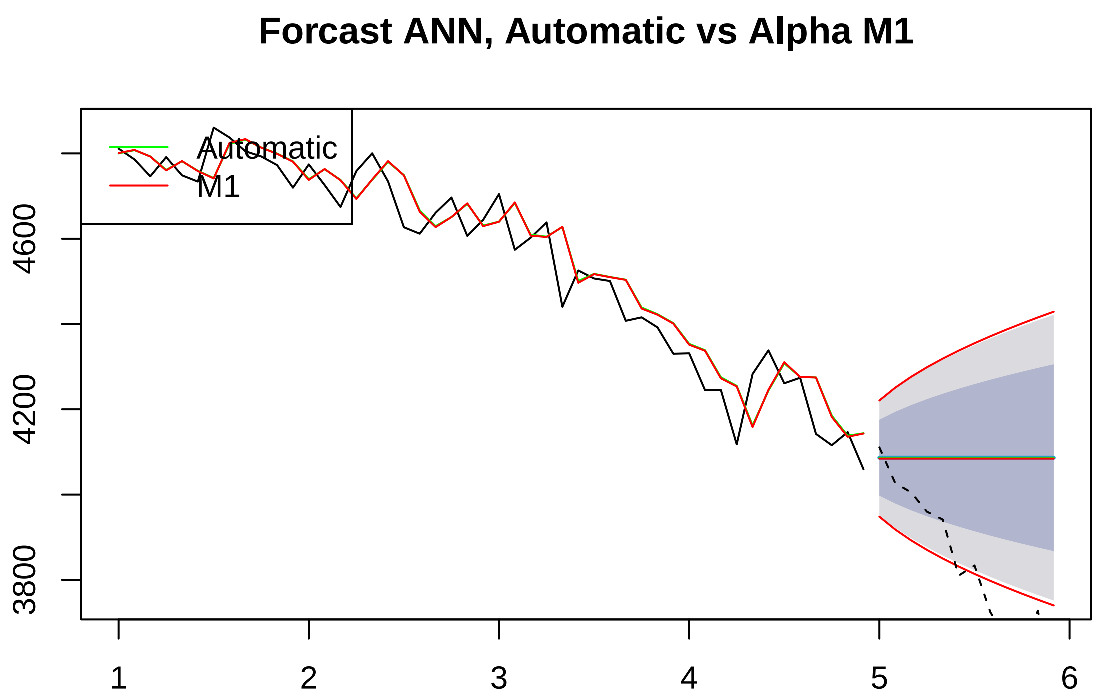
- Forecast ANN

```r
# ploting Automatic ANN
frc <- forecast(fit, h=12)
plot(frc, main = "Forcast Automatic ANN")
lines(fit$fitted,col="green")
```



- Forecast ANN and Alpha M1

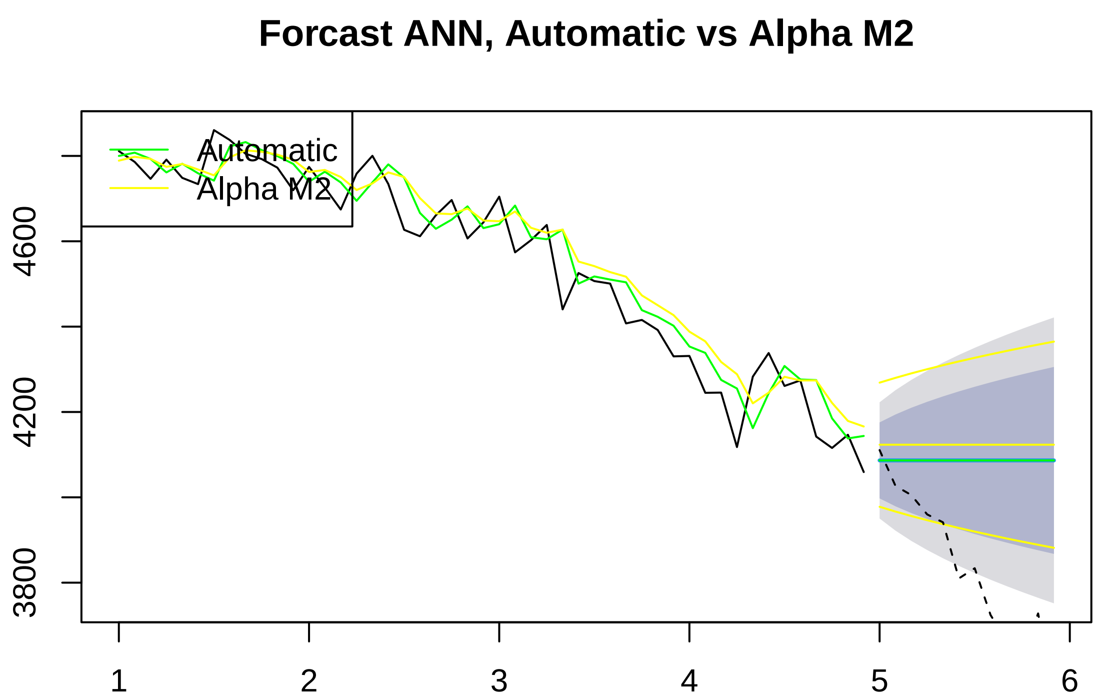
```r
# ploting M1 vs ANN
frc_m1 <- forecast(fit_m1,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M1")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red") # 95% upper
lines(y.tst,lty=2)

# legends
legend("topleft",c("Automatic","M1"),col=c("green","red"),lty=1)
```



- Forecast ANN and Alpha M2

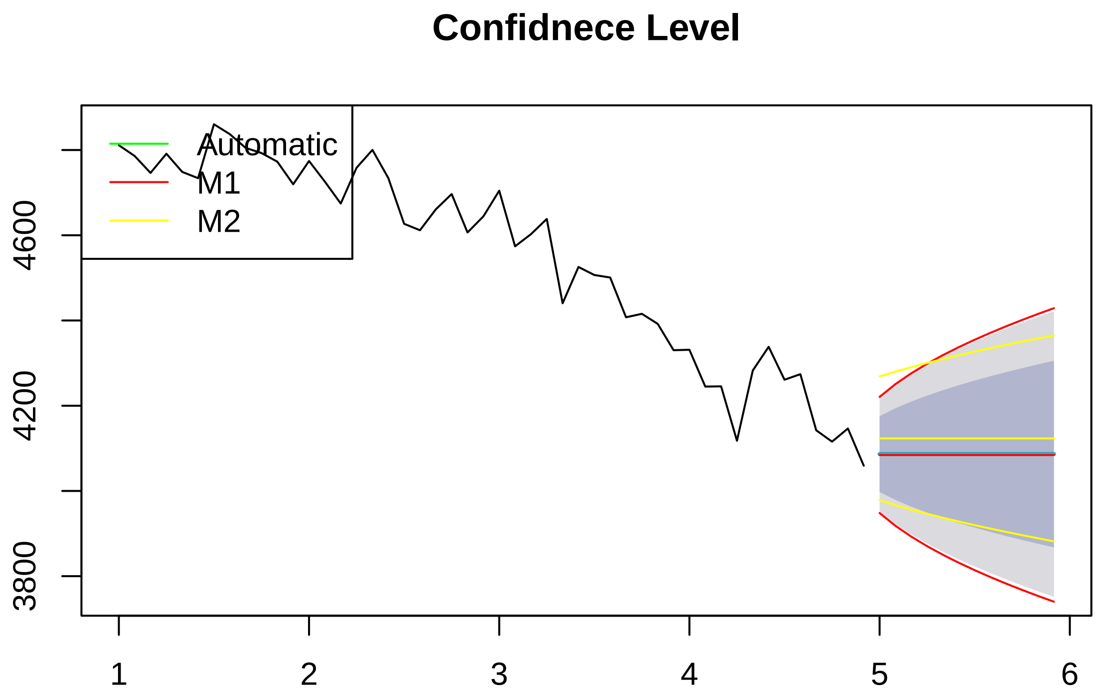
```r
# Ploting M2 vs ANN
frc_m2 <- forecast(fit_m2,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M2")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m2$fitted,col="yellow")

lines(frc_m2$mean,col="yellow")
```

```
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper
lines(y.tst,lty=2)

# Add legend to the plot
legend("topleft",c("Automatic","Alpha M2"),col=c("green","yellow"),lty=1)
```

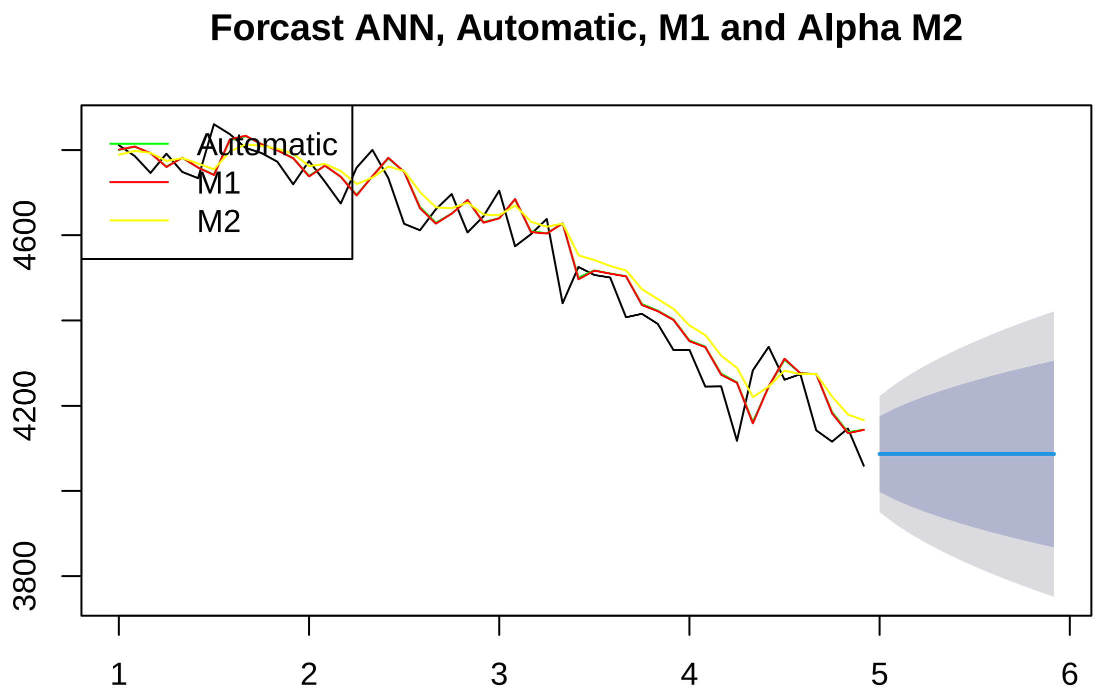


- Confidence level ANN, Alpha M1 and Alpha M2

```
# Ploting confidence level of all 3
plot(frc, main = "Confidnece Level")
lines(frc$mean,col="green")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red")

lines(frc_m2$mean,col="yellow")
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper

# Add legend to the plot
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```




```r
plot(frc, main = "Forcast ANN, Automatic, M1 and Alpha M2")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")
lines(fit_m2$fitted,col="yellow")
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```



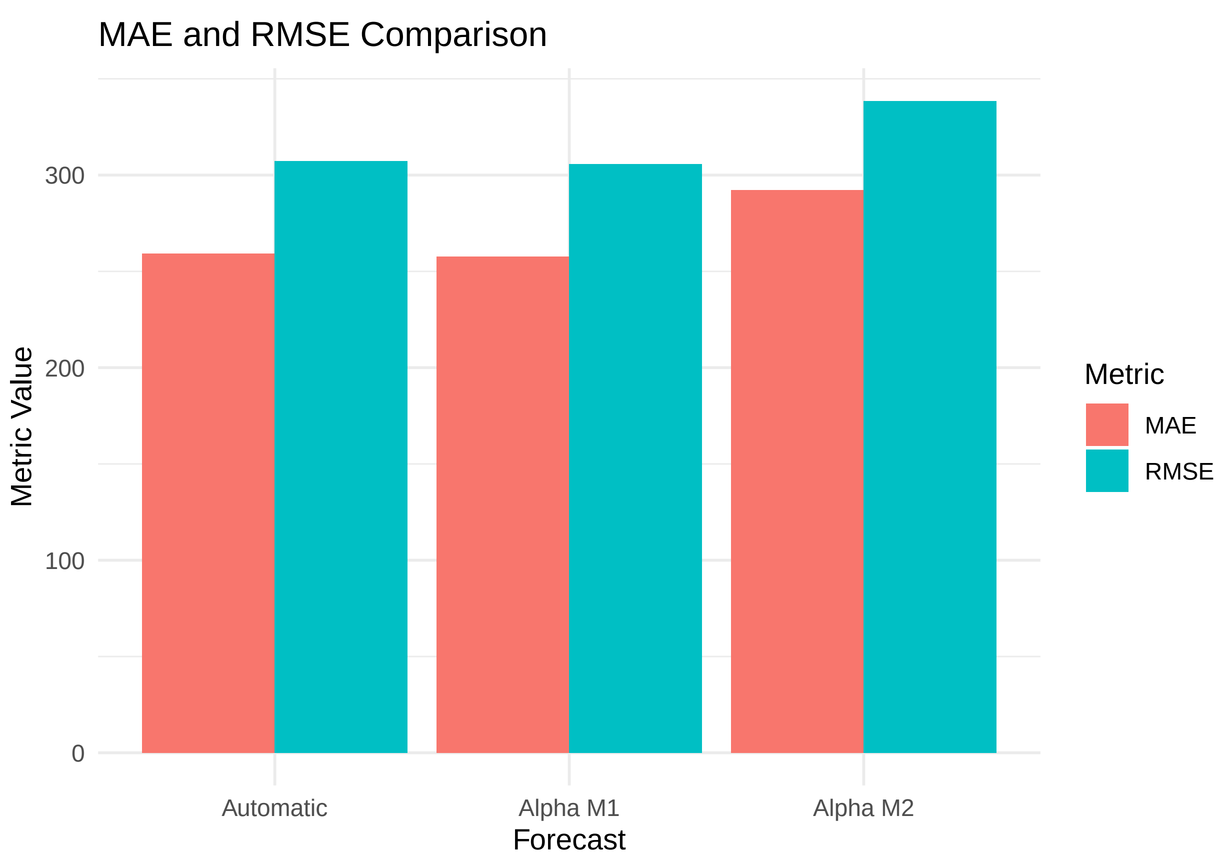

```r
# Function to calculate metrics
calculate_metrics <- function(y, frc) {
  MAE <- mean(abs(y - frc$mean))
  MSE <- mean((y - frc$mean)^2)
  RMSE <- sqrt(MSE)
  return(list(MAE = MAE, MSE = MSE, RMSE = RMSE))
}

# Calculate metrics for different forecasts
metrics_a <- calculate_metrics(y.tst, frc)
metrics_m1 <- calculate_metrics(y.tst, frc_m1)
metrics_m2 <- calculate_metrics(y.tst, frc_m2)

# Create a matrix for the metrics
metrics_matrix <- matrix(c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
                            metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE,
                           metrics_a$MSE, metrics_m1$MSE, metrics_m2$MSE),
                           ncol = 3, byrow = TRUE)

# Add row and column names
rownames(metrics_matrix) <- c("MAE", "MSE", "RMSE")
colnames(metrics_matrix) <- c("Automatic", "Alpha M1", "Alpha M2")
```

```r
# Display the metrics matrix
metrics_matrix
```

1.3.4.3 4.3 Model selection

```
##        Automatic   Alpha M1    Alpha M2
## MAE     259.3613   257.6299    292.1281
## MSE     307.3608   305.6371    338.5529
## RMSE 94470.6834 93414.0422 114618.0610
```

```r
# Create a data frame from the metrics matrix, excluding MSE
metrics_df <- data.frame(
  Metric = rep(c("MAE", "RMSE"), each = 3),
  Value = c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
            metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE),
  Forecast = rep(c("Automatic", "Alpha M1", "Alpha M2"), times = 2)
)
metrics_df$Forecast <- factor(metrics_df$Forecast, levels = c("Automatic", "Alpha M1", "Alpha M2"))

# Create a bar plot
ggplot(metrics_df, aes(x = Forecast, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "MAE and RMSE Comparison",
       y = "Metric Value") +
  theme_minimal()
```



1.3.5 5. Answers

- Which one is best using your judgement?

  In evaluating different models for the "Trend_A" component within the in-sample data, it is noted that **Model M2** offers a good fit to the data. However, it falls short in terms of effectively filtering out noise and outliers, which could affect its suitability for certain applications. In contrast, the automatic model performs well in filtering out noise and outliers, emphasizing data robustness, although it might not achieve the best fit. Model M3, on the other hand, does not provide a satisfactory fitting to the data.

- Which one is best using errors?

  The assessment of error metrics indicates that the **Automatic** model excels in minimizing MSE, suggesting that it provides the most accurate predictions for the "Trend_A" component. Conversely, **Model M1** stands out in terms of AIC error minimization. These observations underline the importance of considering multiple error metrics, as they may highlight different aspects of model performance.

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

  In the context of out-of-sample data, **Model M2** emerges as the preferred choice, as it exhibits the best RMSE and MAE values among the models considered. This suggests that Model M2 generalizes effectively to unseen data points for forecasting "Trend_A".

1.4 Trend B

1.4.1 1. Loading Data

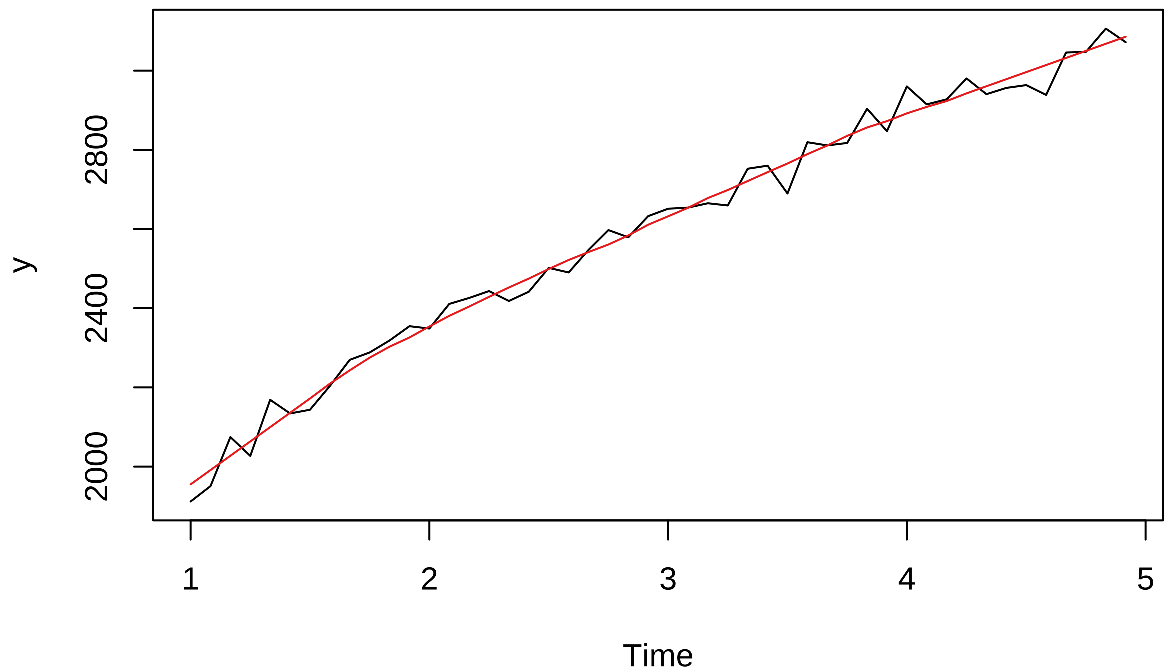
```
y <- Y[,5]
# Transform it into a time series
y <- ts(y,frequency=12)
```

1.4.2 2. Constructing estimation and hold-out sets

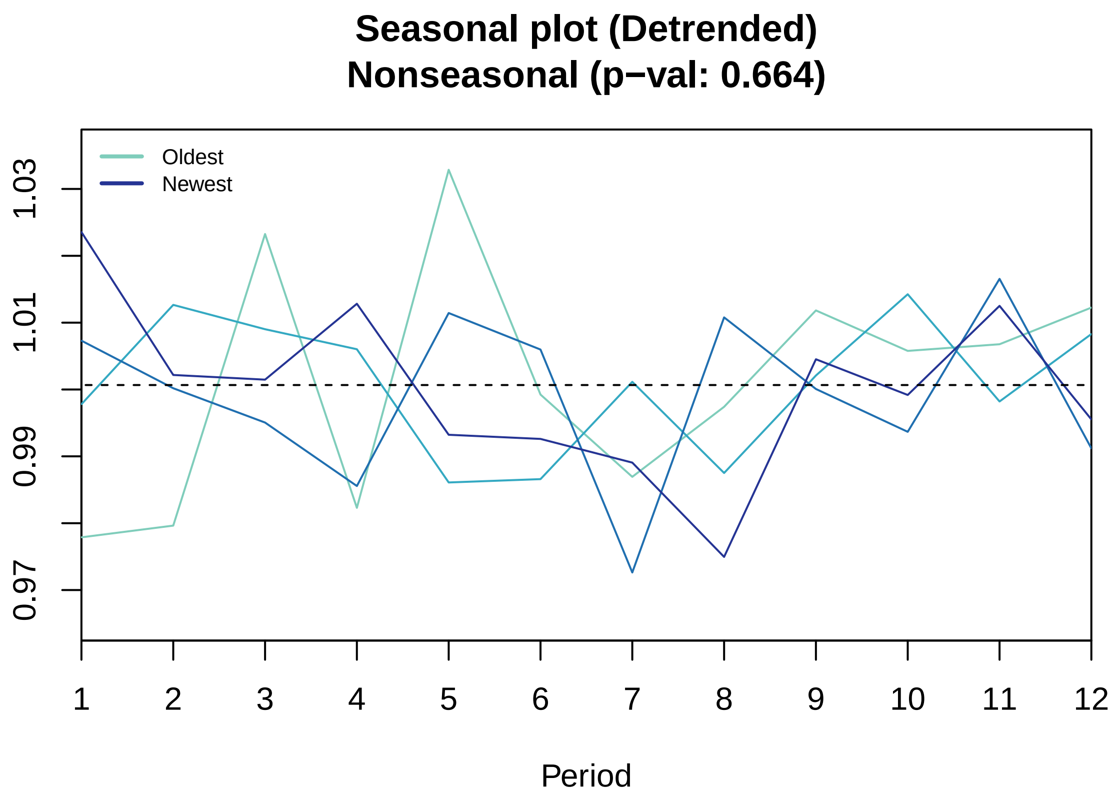
```
y.tst <- tail(y,12)
y.trn <- head(y,48)
```

1.4.3 3. Exploration

```
cma <- cmav(y.trn,outplot=1)
```




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




```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0)
## Evidence of seasonality: FALSE  (pval: 0.664)
```

1.4.4 4. Forecasting

```
# Automatic Alpha ANN
fit <- ets(y.trn,model="ANN")
print(fit)
```

1.4.4.1 4.1 Model fitting

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.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
```

```
# Alpha M1 ANN
fit_m1 <- ets(y.trn,model="ANN", alpha = 0.6 )
print(fit_m1)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.6)
##
##   Smoothing parameters:
##     alpha = 0.6
##
##   Initial states:
##     l = 1947.6703
##
##   sigma:  59.2916
##
##      AIC     AICc      BIC
## 579.6917 579.9584 583.4341
```

```
# Alpha M2 ANN
fit_m2 <- ets(y.trn,model="ANN", alpha = 0.3 )
print(fit_m2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.3)
##
##   Smoothing parameters:
##     alpha = 0.3
##
##   Initial states:
##     l = 2024.5847
##
##   sigma:  87.7993
##
##      AIC     AICc      BIC
## 617.3800 617.6466 621.1224
```

```
cirt <- array(NA, c(3, 4), dimnames = list(c("Automatic", "M1", "M2"),
                                           c("MSE", "AIC", "AICc","BIC")))

models <- list(fit, fit_m1, fit_m2)

for (i in 1:3) {
```

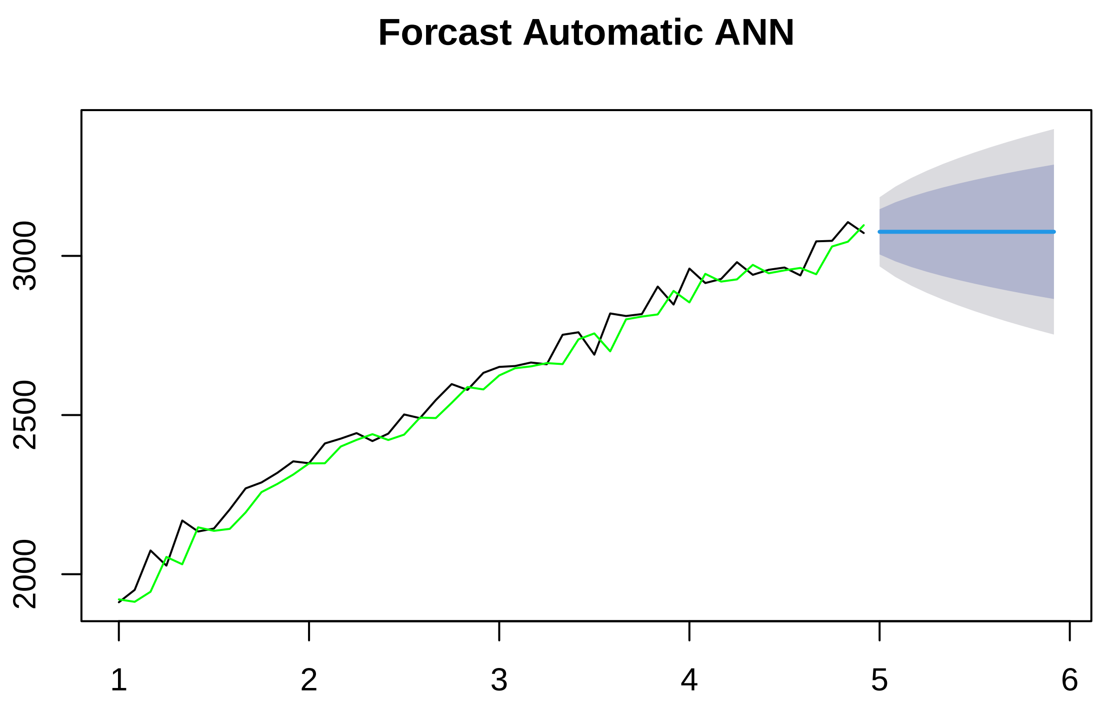
```
  cirt[i, "MSE"] <- models[[i]]$mse
  cirt[i, "AIC"] <- models[[i]]$aic
  cirt[i, "AICc"] <- models[[i]]$aicc
  cirt[i, "BIC"] <- models[[i]]$bic
}

print(cirt)
```

```
##                MSE      AIC     AICc      BIC
## Automatic 2943.725 575.2143 575.7598 580.8279
## M1        3369.016 579.6917 579.9584 583.4341
## M2        7387.527 617.3800 617.6466 621.1224
```

1.4.4.2 4.2 Forecasting

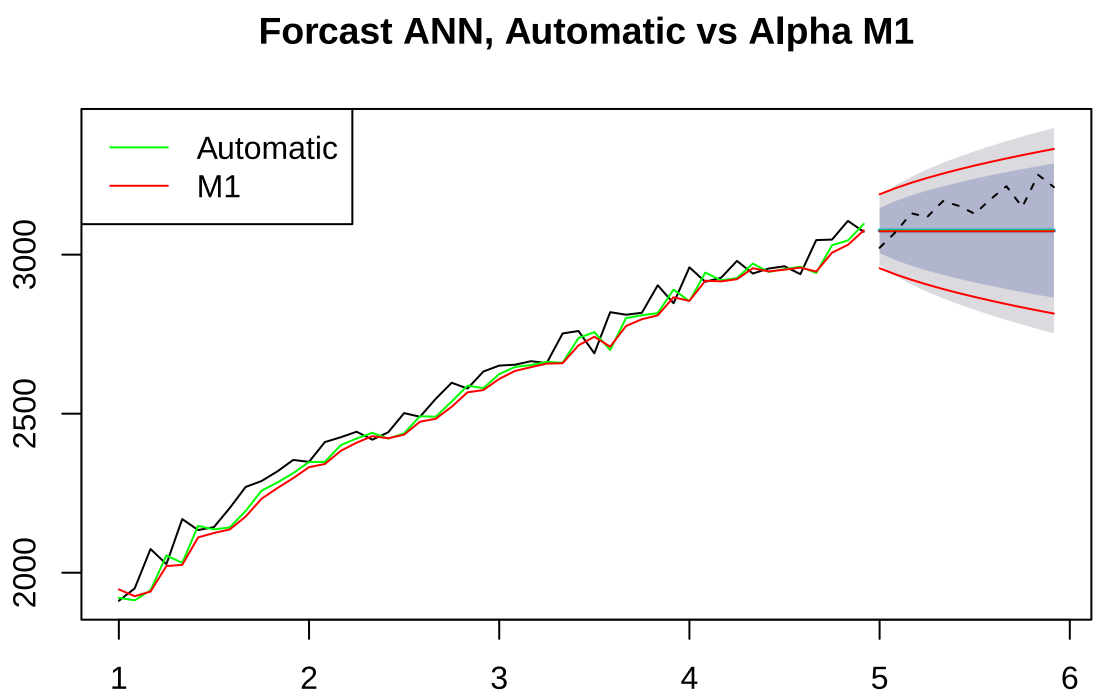
- Forecast ANN

```
# ploting Automatic ANN
frc <- forecast(fit, h=12)
plot(frc, main = "Forcast Automatic ANN")
lines(fit$fitted,col="green")
```



- Forecast ANN and Alpha M1

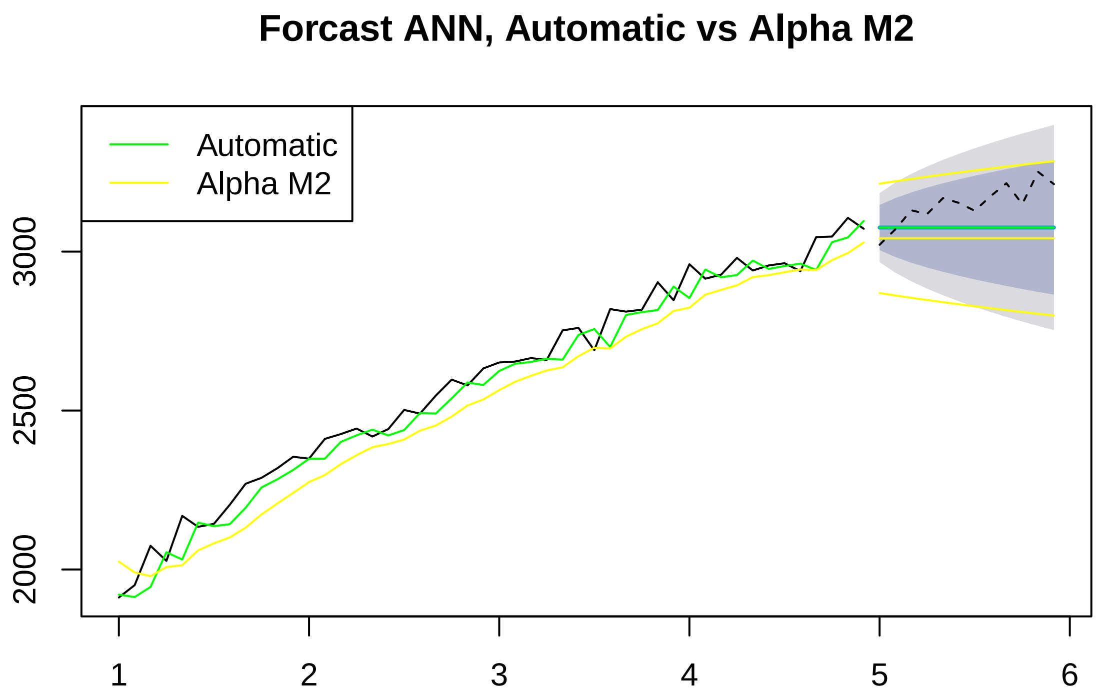
```r
# ploting M1 vs ANN
frc_m1 <- forecast(fit_m1,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M1")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red") # 95% upper
lines(y.tst,lty=2)

# legends
legend("topleft",c("Automatic","M1"),col=c("green","red"),lty=1)
```



- Forecast ANN and Alpha M2

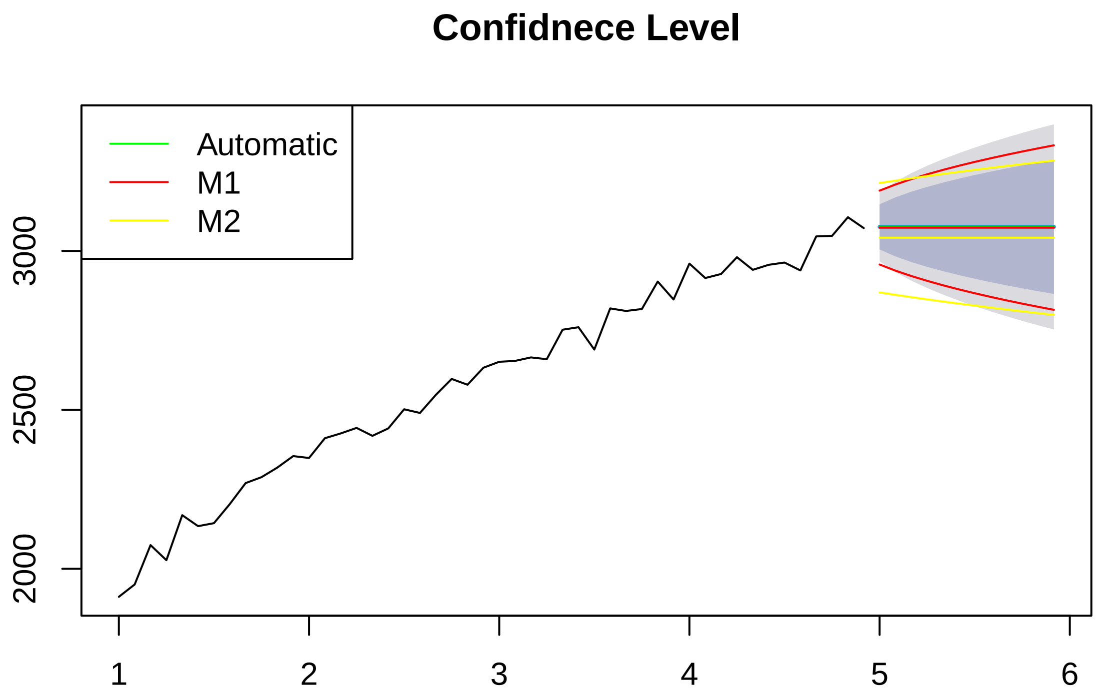
```r
# Ploting M2 vs ANN
frc_m2 <- forecast(fit_m2,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M2")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m2$fitted,col="yellow")

lines(frc_m2$mean,col="yellow")
```

```
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper
lines(y.tst,lty=2)

# Add legend to the plot
legend("topleft",c("Automatic","Alpha M2"),col=c("green","yellow"),lty=1)
```

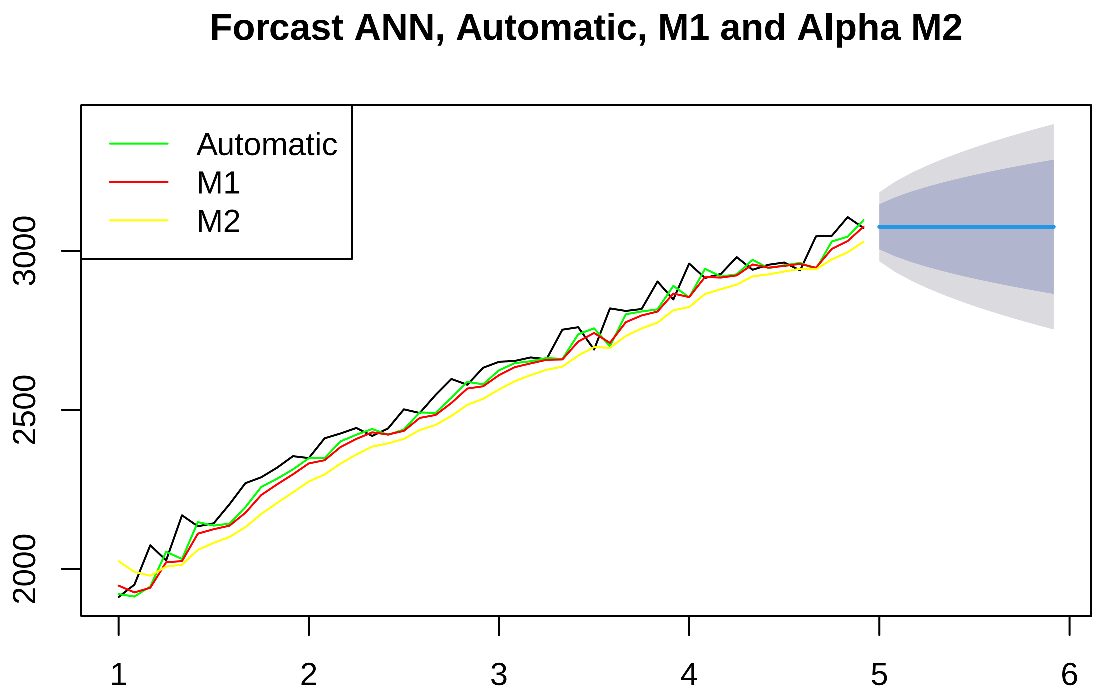


- Confidence level ANN, Alpha M1 and Alpha M2

```
# Ploting confidence level of all 3
plot(frc, main = "Confidnece Level")
lines(frc$mean,col="green")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red")

lines(frc_m2$mean,col="yellow")
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper

# Add legend to the plot
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```




```r
plot(frc, main = "Forcast ANN, Automatic, M1 and Alpha M2")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")
lines(fit_m2$fitted,col="yellow")
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```



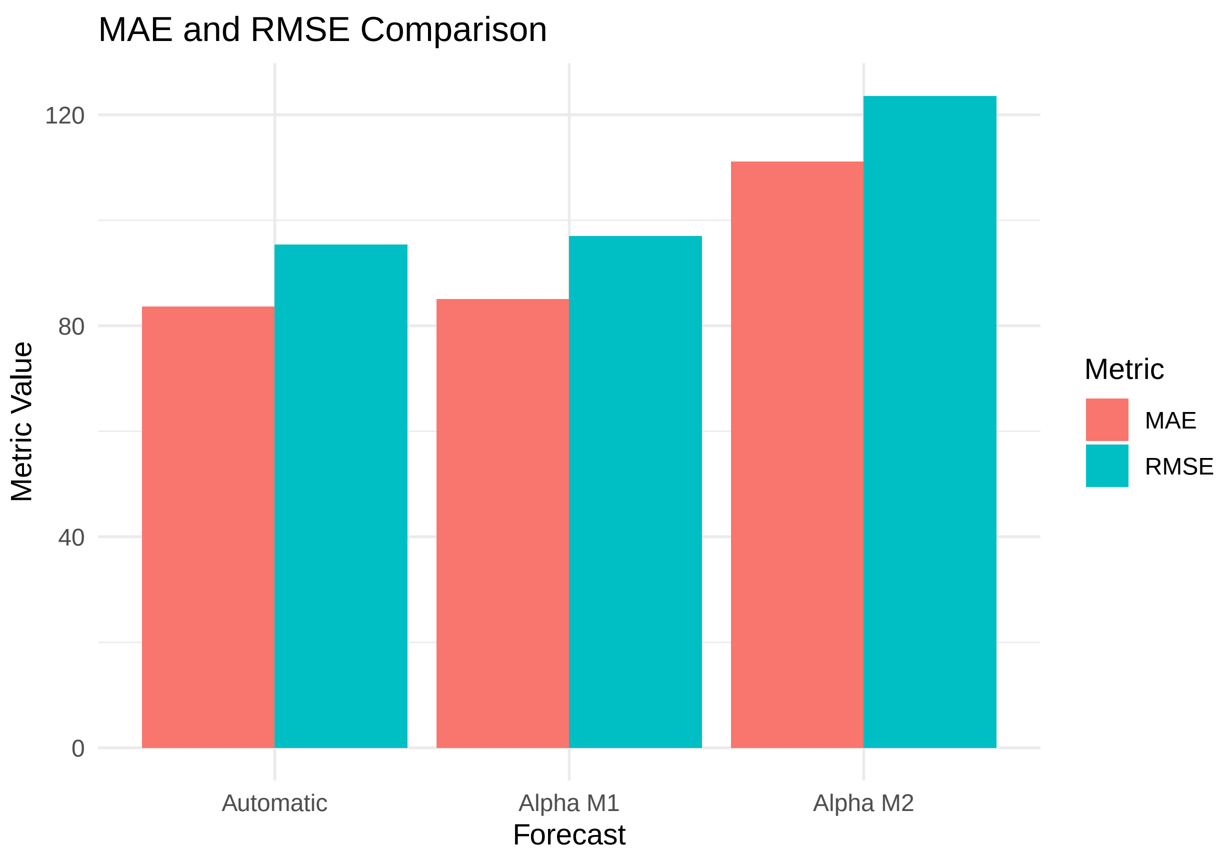

```
# Function to calculate metrics
calculate_metrics <- function(y, frc) {
  MAE <- mean(abs(y - frc$mean))
  MSE <- mean((y - frc$mean)^2)
  RMSE <- sqrt(MSE)
  return(list(MAE = MAE, MSE = MSE, RMSE = RMSE))
}

# Calculate metrics for different forecasts
metrics_a <- calculate_metrics(y.tst, frc)
metrics_m1 <- calculate_metrics(y.tst, frc_m1)
metrics_m2 <- calculate_metrics(y.tst, frc_m2)

# Create a matrix for the metrics
metrics_matrix <- matrix(c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
                           metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE,
                          metrics_a$MSE, metrics_m1$MSE, metrics_m2$MSE),
                          ncol = 3, byrow = TRUE)

# Add row and column names
rownames(metrics_matrix) <- c("MAE", "MSE", "RMSE")
colnames(metrics_matrix) <- c("Automatic", "Alpha M1", "Alpha M2")
```

```r
# Display the metrics matrix
metrics_matrix
```

1.4.4.3 4.3 Model selection

```
##        Automatic   Alpha M1   Alpha M2
## MAE     83.61050   85.05749   111.1186
## MSE     95.35191   97.03912   123.5854
## RMSE 9091.98703 9416.58993 15273.3534
```

```r
# Create a data frame from the metrics matrix, excluding MSE
metrics_df <- data.frame(
  Metric = rep(c("MAE", "RMSE"), each = 3),
  Value = c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
            metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE),
  Forecast = rep(c("Automatic", "Alpha M1", "Alpha M2"), times = 2)
)
metrics_df$Forecast <- factor(metrics_df$Forecast, levels = c("Automatic", "Alpha M1", "Alpha M2"))

# Create a bar plot
ggplot(metrics_df, aes(x = Forecast, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "MAE and RMSE Comparison",
       y = "Metric Value") +
  theme_minimal()
```



1.4.5 5. Answers

- Which one is best using your judgement?

  In the context of modeling the "Trend_B" component within the in-sample data, it is observed that the **Automatic** model offers a good fit to the data. However, it does not effectively filter out noise and outliers. Considering this, the judgment leans toward the automatic model as the preferable choice.

- Which one is best using errors?

  Both error metrics, MSE and AIC, indicate that the **Automatic** model performs the best among the considered models. This suggests that the automatic model provides the most accurate predictions for the "Trend_B" component within the in-sample data.

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

  In the out-of-sample analysis, the **automatic** model continues to demonstrate its superiority by yielding the best RMSE and MAE values. However, it is important to note that single exponential smoothing, such as the automatic model, may have limitations in modeling time series data with trends.

  It is crucial to recognize that while the automatic model performs well in the current analysis, it may not be the final solution for modeling time series data with trends. More sophisticated models, such as those incorporating trend components explicitly, might be necessary for a more comprehensive modeling approach in certain cases.

1.5 Season A

1.5.1 1. Loading Data

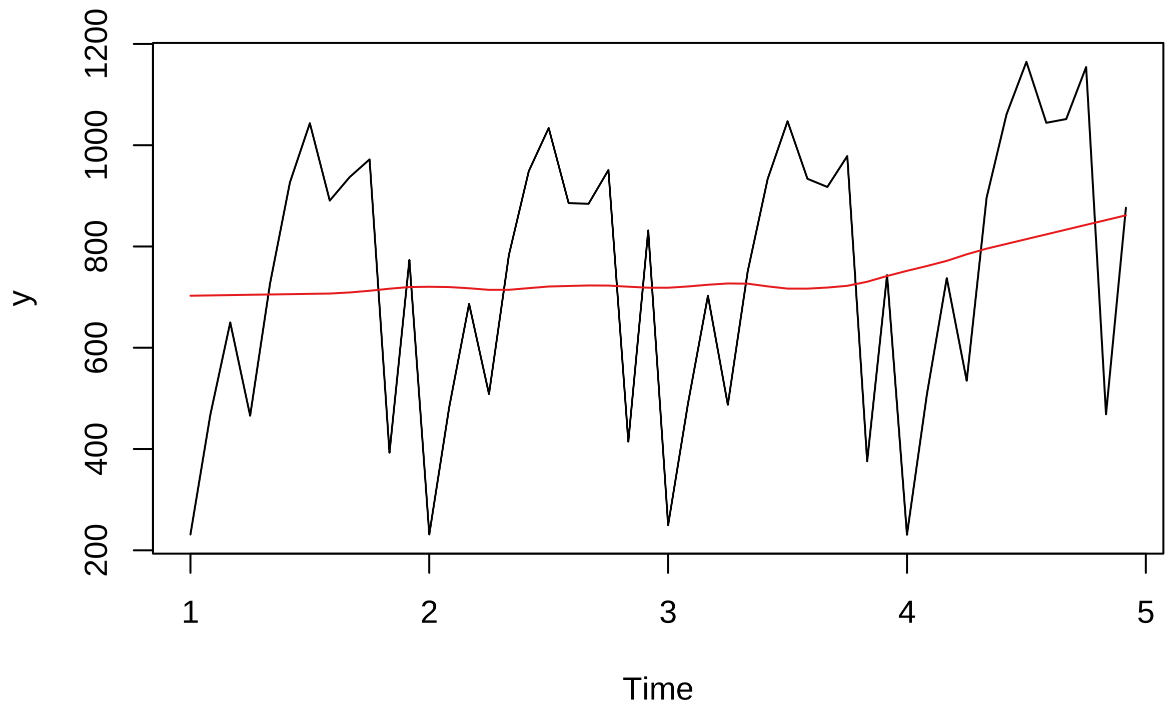
```r
y <- Y[,6]
# Transform it into a time series
y <- ts(y,frequency=12)
```

1.5.2 2. Constructing estimation and hold-out sets

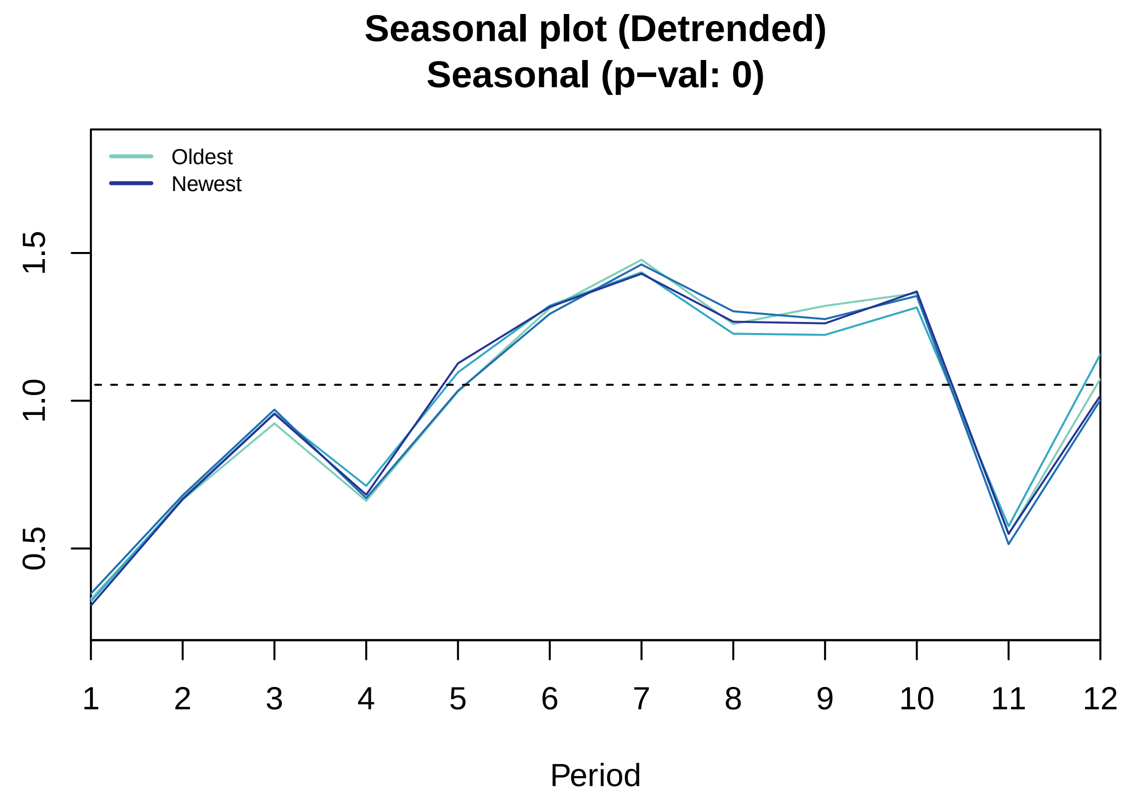
```r
y.tst <- tail(y,12)
y.trn <- head(y,48)
```

1.5.3 3. Exploration

```r
cma <- cmav(y.trn,outplot=1)
```




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




```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0)
## Evidence of seasonality: TRUE  (pval: 0)
```

1.5.4 4. Forecasting

```
# Automatic Alpha ANN
fit <- ets(y.trn,model="ANN")
print(fit)
```

1.5.4.1 4.1 Model fitting

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.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
# Alpha M1 ANN
fit_m1 <- ets(y.trn,model="ANN", alpha = 0.7 )
print(fit_m1)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.7)
##
##   Smoothing parameters:
##     alpha = 0.7
##
##   Initial states:
##     l = 316.4399
##
##   sigma:  269.9186
##
##      AIC     AICc      BIC
## 725.1943 725.4610 728.9367
```

```r
# Alpha M2 ANN
fit_m2 <- ets(y.trn,model="ANN", alpha = 0.4 )
print(fit_m2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.4)
##
##   Smoothing parameters:
##     alpha = 0.4
##
##   Initial states:
##     l = 438.6748
##
##   sigma:  269.3941
##
##      AIC     AICc      BIC
## 725.0076 725.2743 728.7500
```

```r
cirt <- array(NA, c(3, 4), dimnames = list(c("Automatic", "M1", "M2"),
                                          c("MSE", "AIC", "AICc","BIC")))

models <- list(fit, fit_m1, fit_m2)

for (i in 1:3) {
```

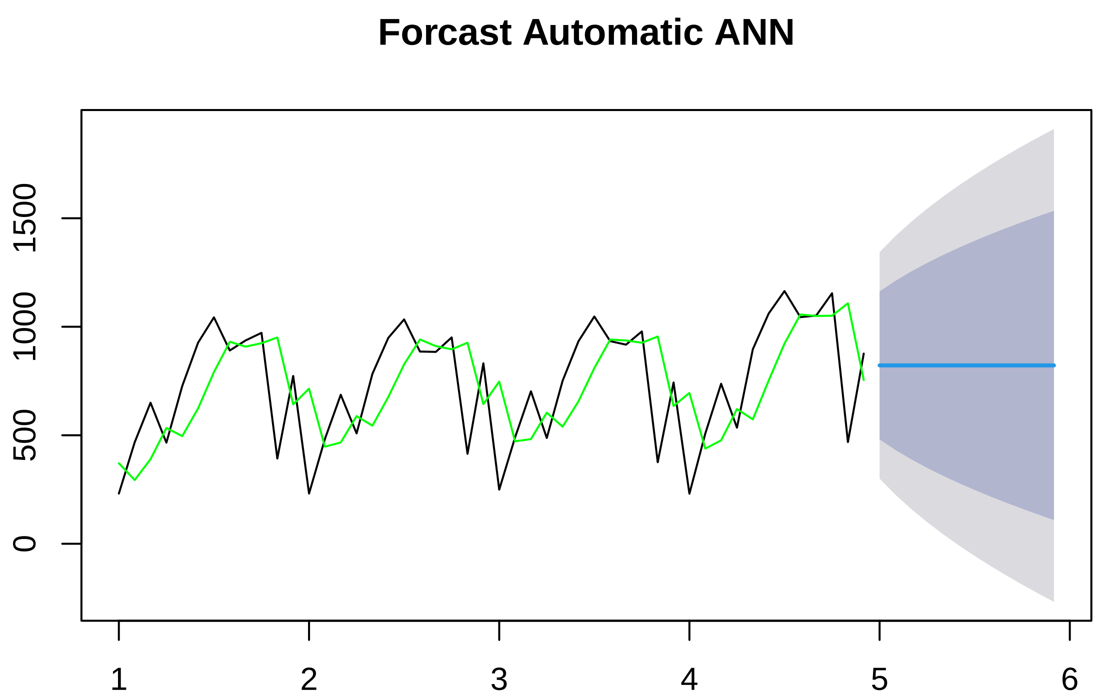
```
  cirt[i, "MSE"] <- models[[i]]$mse
  cirt[i, "AIC"] <- models[[i]]$aic
  cirt[i, "AICc"] <- models[[i]]$aicc
  cirt[i, "BIC"] <- models[[i]]$bic
}

print(cirt)
```

```
##                MSE      AIC     AICc      BIC
## Automatic 67909.28 725.8622 726.4076 731.4758
## M1        69820.37 725.1943 725.4610 728.9367
## M2        69549.29 725.0076 725.2743 728.7500
```

1.5.4.2 4.2 Forecasting

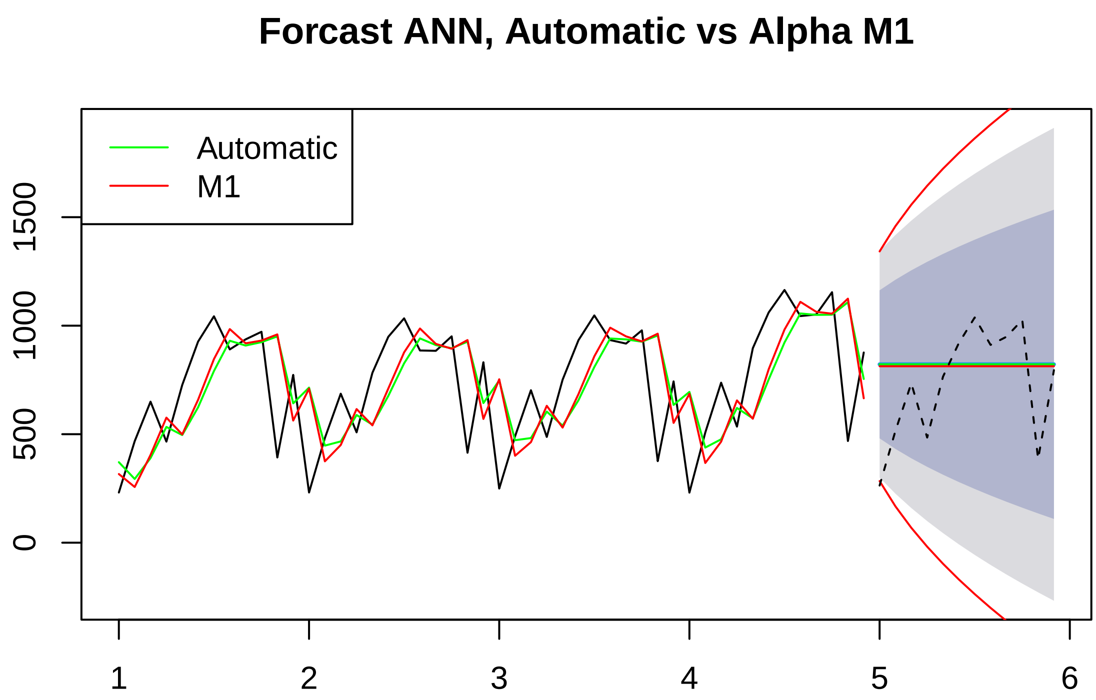
- Forecast ANN

```
# ploting Automatic ANN
frc <- forecast(fit, h=12)
plot(frc, main = "Forcast Automatic ANN")
lines(fit$fitted,col="green")
```



- Forecast ANN and Alpha M1

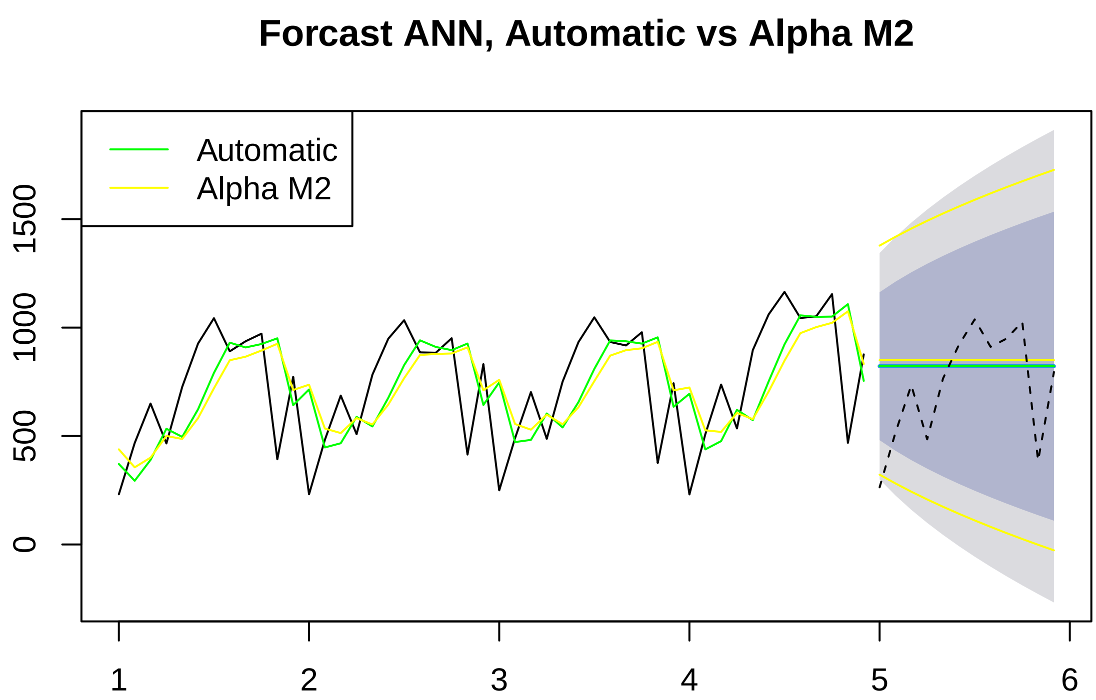
```r
# ploting M1 vs ANN
frc_m1 <- forecast(fit_m1,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M1")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red") # 95% upper
lines(y.tst,lty=2)

# legends
legend("topleft",c("Automatic","M1"),col=c("green","red"),lty=1)
```



- Forecast ANN and Alpha M2

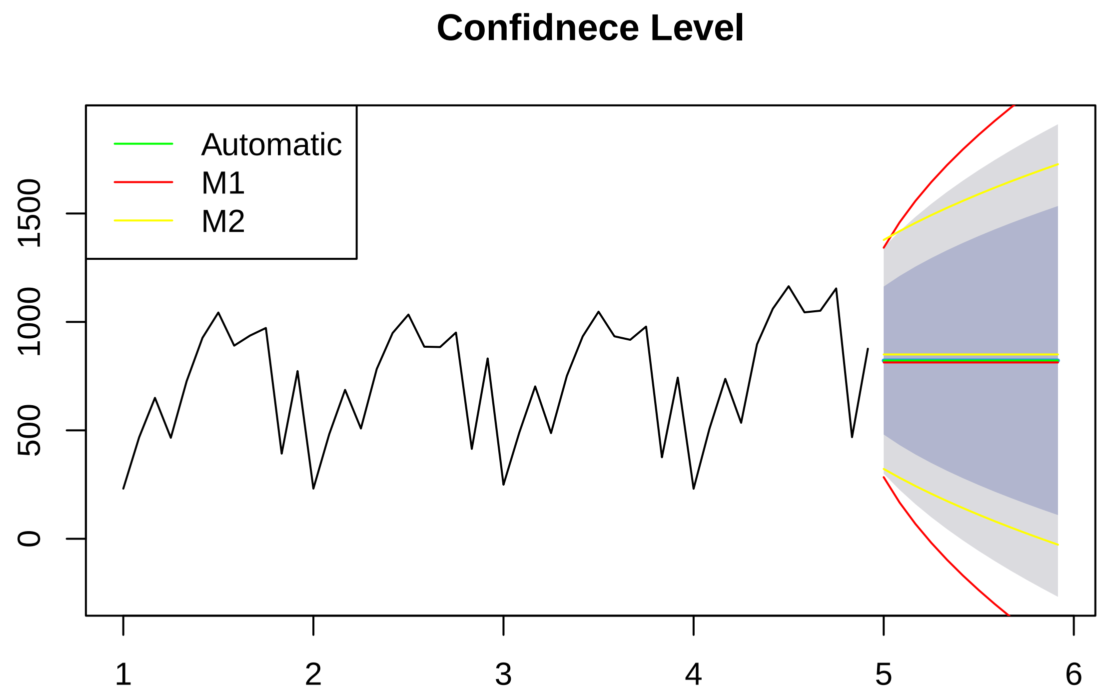
```r
# Ploting M2 vs ANN
frc_m2 <- forecast(fit_m2,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M2")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m2$fitted,col="yellow")

lines(frc_m2$mean,col="yellow")
```

```r
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper
lines(y.tst,lty=2)

# Add legend to the plot
legend("topleft",c("Automatic","Alpha M2"),col=c("green","yellow"),lty=1)
```

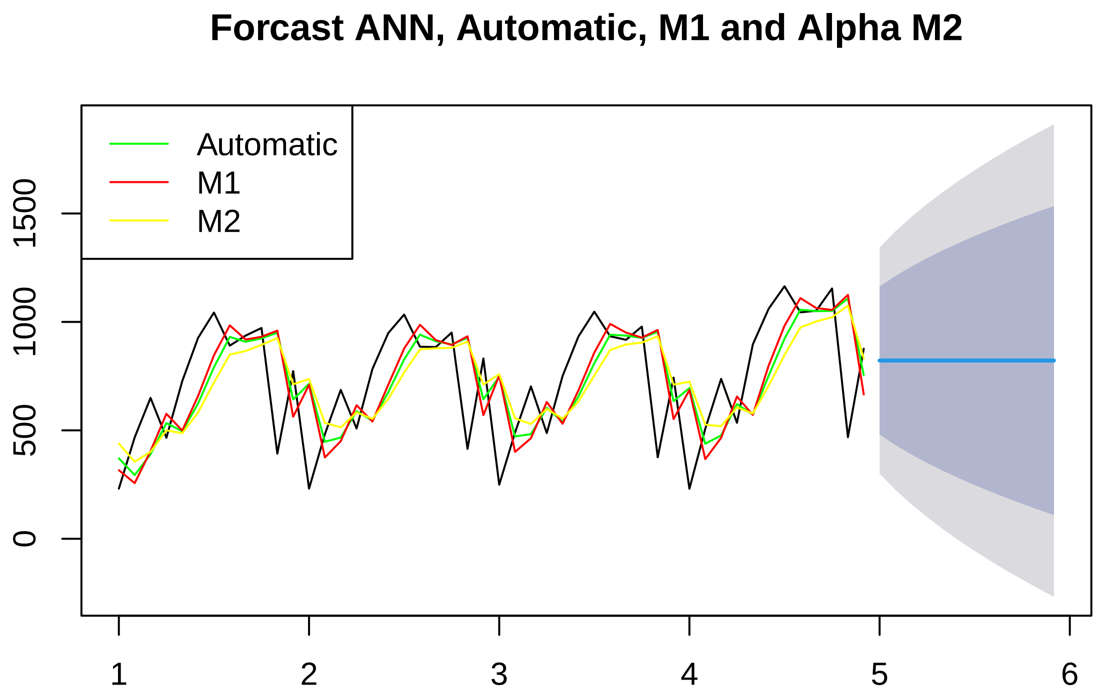


- Confidence level ANN, Alpha M1 and Alpha M2

```r
# Ploting confidence level of all 3
plot(frc, main = "Confidnece Level")
lines(frc$mean,col="green")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red")

lines(frc_m2$mean,col="yellow")
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper

# Add legend to the plot
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```




```
plot(frc, main = "Forcast ANN, Automatic, M1 and Alpha M2")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")
lines(fit_m2$fitted,col="yellow")
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```



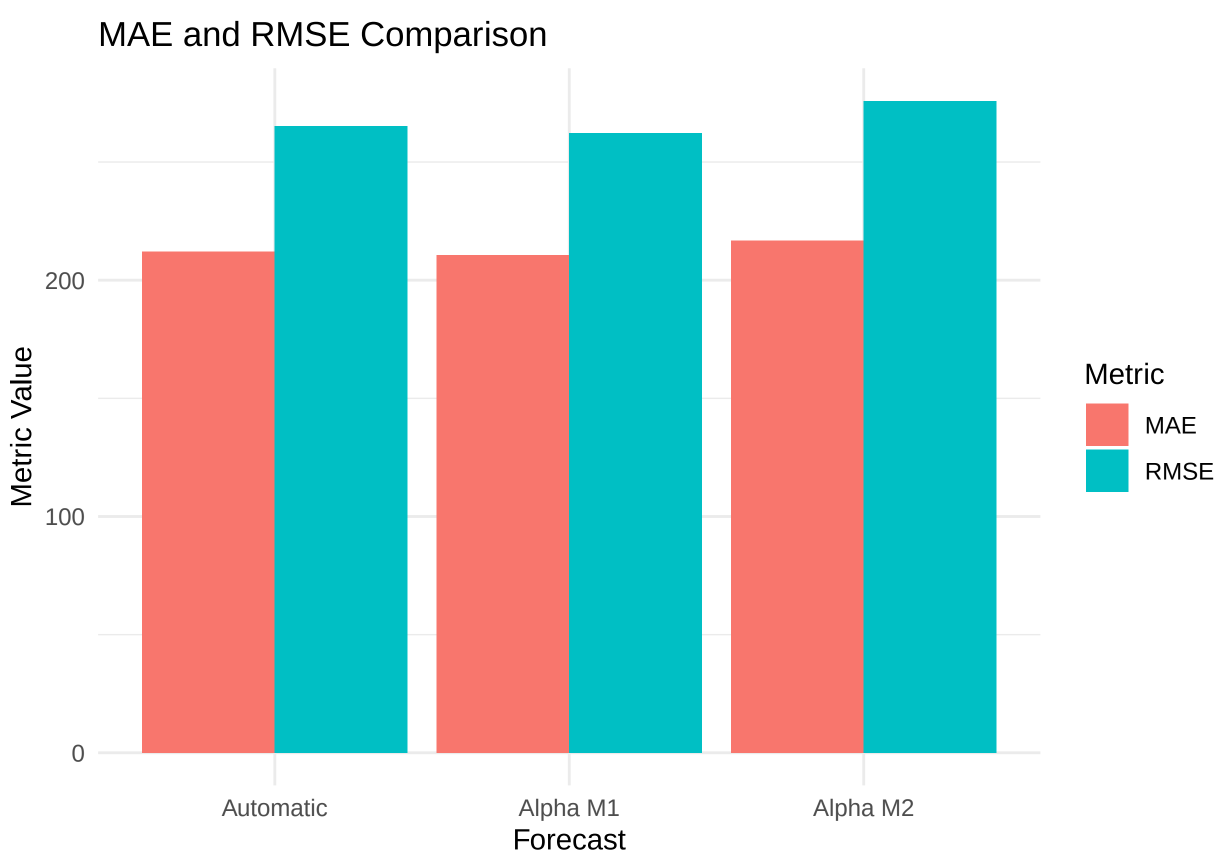

```
# Function to calculate metrics
calculate_metrics <- function(y, frc) {
  MAE <- mean(abs(y - frc$mean))
  MSE <- mean((y - frc$mean)^2)
  RMSE <- sqrt(MSE)
  return(list(MAE = MAE, MSE = MSE, RMSE = RMSE))
}

# Calculate metrics for different forecasts
metrics_a <- calculate_metrics(y.tst, frc)
metrics_m1 <- calculate_metrics(y.tst, frc_m1)
metrics_m2 <- calculate_metrics(y.tst, frc_m2)

# Create a matrix for the metrics
metrics_matrix <- matrix(c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
                           metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE,
                          metrics_a$MSE, metrics_m1$MSE, metrics_m2$MSE),
                          ncol = 3, byrow = TRUE)

# Add row and column names
rownames(metrics_matrix) <- c("MAE", "MSE", "RMSE")
colnames(metrics_matrix) <- c("Automatic", "Alpha M1", "Alpha M2")
```

```r
# Display the metrics matrix
metrics_matrix
```

1.5.4.3 4.3 Model selection

```
##         Automatic   Alpha M1   Alpha M2
## MAE      212.0187   210.5535   216.6974
## MSE      265.1394   262.3003   275.8902
## RMSE 70298.8889 68801.4236 76115.4113
```

```r
# Create a data frame from the metrics matrix, excluding MSE
metrics_df <- data.frame(
  Metric = rep(c("MAE", "RMSE"), each = 3),
  Value = c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
            metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE),
  Forecast = rep(c("Automatic", "Alpha M1", "Alpha M2"), times = 2)
)
metrics_df$Forecast <- factor(metrics_df$Forecast, levels = c("Automatic", "Alpha M1", "Alpha M2"))

# Create a bar plot
ggplot(metrics_df, aes(x = Forecast, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "MAE and RMSE Comparison",
       y = "Metric Value") +
  theme_minimal()
```



1.5.5 5. Answers

- Which one is best using your judgement?

  In assessing the models for the "Season_A" component within the in-sample data, it is evident that the **Automatic** model captures seasonality effectively, which is a crucial aspect of the time series. However, it falls short in filtering out noise and outliers, indicating a potential drawback in terms of robustness. Model M1 fits the data well but with more noise and outliers, while Model M2 focuses on noise and outlier reduction but doesn't provide an ideal fit.

- Which one is best using errors?

  Both error metrics, MSE and AIC, suggest that the **Automatic** model outperforms the other models in terms of error minimization. This indicates that the automatic model provides the most accurate predictions for the "Season_A" component within the in-sample data..

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

  The out-of-sample analysis reveals that the automatic model may not be the optimal choice, as it does not yield the best RMSE and MAE values. **Model M2** demonstrates superior performance in terms of these metrics.

  It is crucial to acknowledge that while the automatic model captures seasonality effectively in the in-sample data, its performance may vary when applied to out-of-sample data. Model M2, despite its limited fitting capability, appears to generalize better to unseen data points for forecasting "Season_A."

1.6 Season B

1.6.1 1. Loading Data

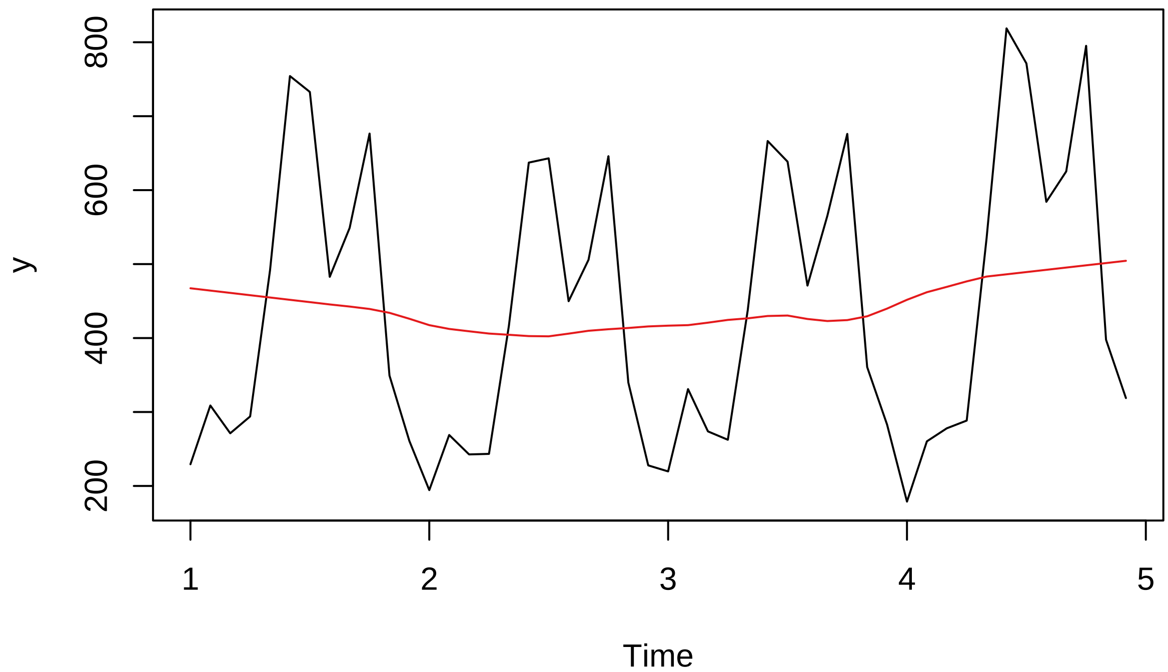
```r
y <- Y[,7]
# Transform it into a time series
y <- ts(y,frequency=12)
```

1.6.2 2. Constructing estimation and hold-out sets

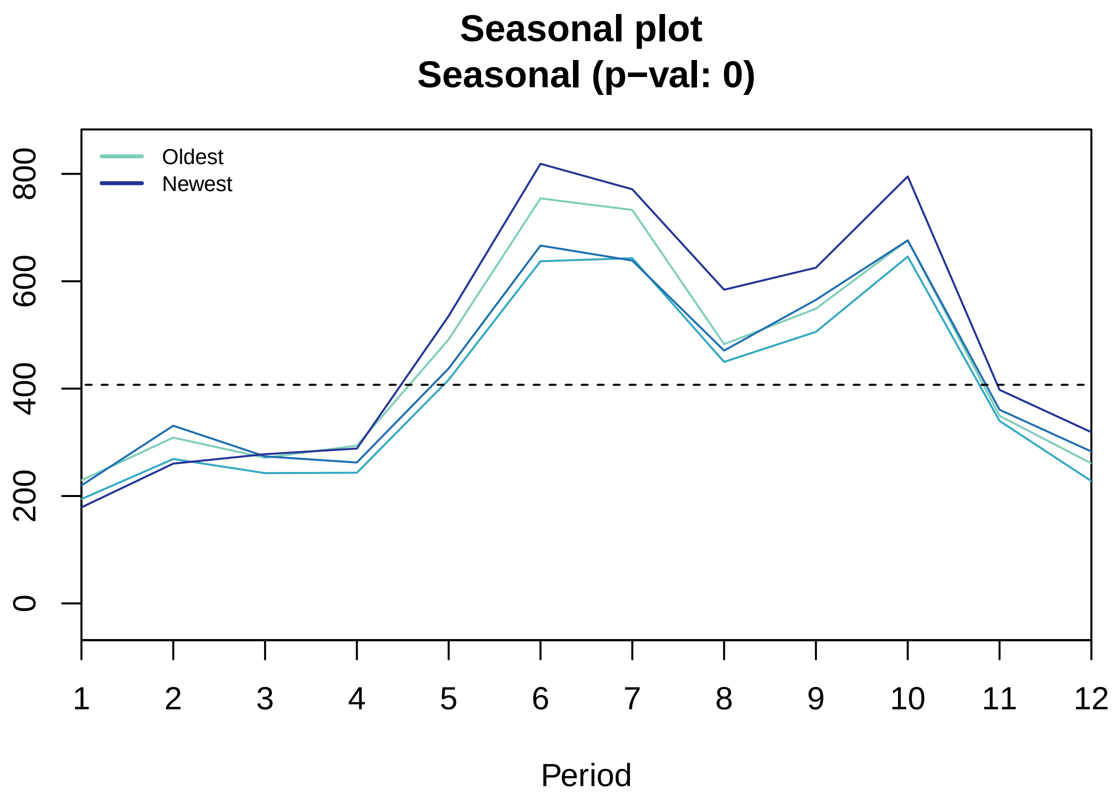
```r
y.tst <- tail(y,12)
y.trn <- head(y,48)
```

1.6.3 3. Exploration

```r
cma <- cmav(y.trn,outplot=1)
```




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




```
## Results of statistical testing
## Evidence of trend: FALSE  (pval: 0.419)
## Evidence of seasonality: TRUE  (pval: 0)
```

1.6.4 4. Forecasting

```r
# Automatic Alpha ANN
fit <- ets(y.trn,model="ANN")
print(fit)
```

1.6.4.1 4.1 Model fitting

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.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
```

```r
# Alpha M1 ANN
fit_m1 <- ets(y.trn,model="ANN", alpha = 0.7 )
print(fit_m1)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.7)
##
##   Smoothing parameters:
##     alpha = 0.7
##
##   Initial states:
##     l = 252.2465
##
##   sigma:  169.5319
##
##      AIC     AICc      BIC
## 680.5467 680.8134 684.2891
```

```r
# Alpha M2 ANN
fit_m2 <- ets(y.trn,model="ANN", alpha = 0.4 )
print(fit_m2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.4)
##
##   Smoothing parameters:
##     alpha = 0.4
##
##   Initial states:
##     l = 306.8874
##
##   sigma:  185.4211
##
##      AIC     AICc      BIC
## 689.1472 689.4139 692.8896
```

```r
cirt <- array(NA, c(3, 4), dimnames = list(c("Automatic", "M1", "M2"),
                                          c("MSE", "AIC", "AICc","BIC")))

models <- list(fit, fit_m1, fit_m2)

for (i in 1:3) {
```

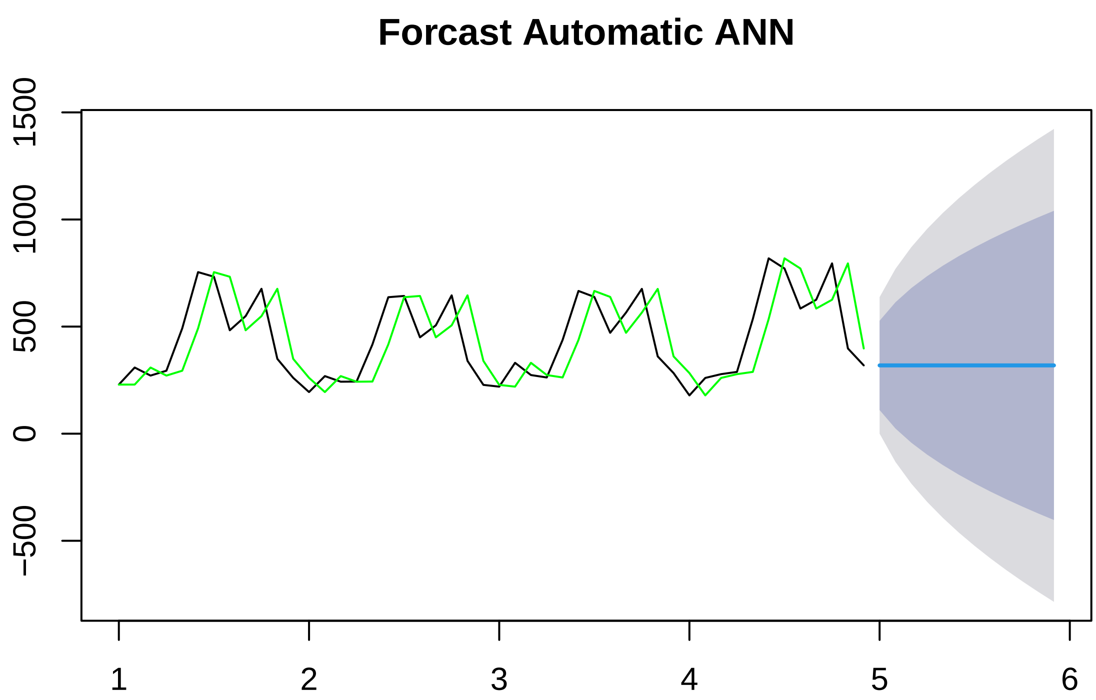
```
  cirt[i, "MSE"] <- models[[i]]$mse
  cirt[i, "AIC"] <- models[[i]]$aic
  cirt[i, "AICc"] <- models[[i]]$aicc
  cirt[i, "BIC"] <- models[[i]]$bic
}

print(cirt)
```

```
##                MSE      AIC     AICc      BIC
## Automatic 25336.28 678.5373 679.0827 684.1509
## M1        27543.51 680.5467 680.8134 684.2891
## M2        32948.45 689.1472 689.4139 692.8896
```

1.6.4.2 4.2 Forecasting

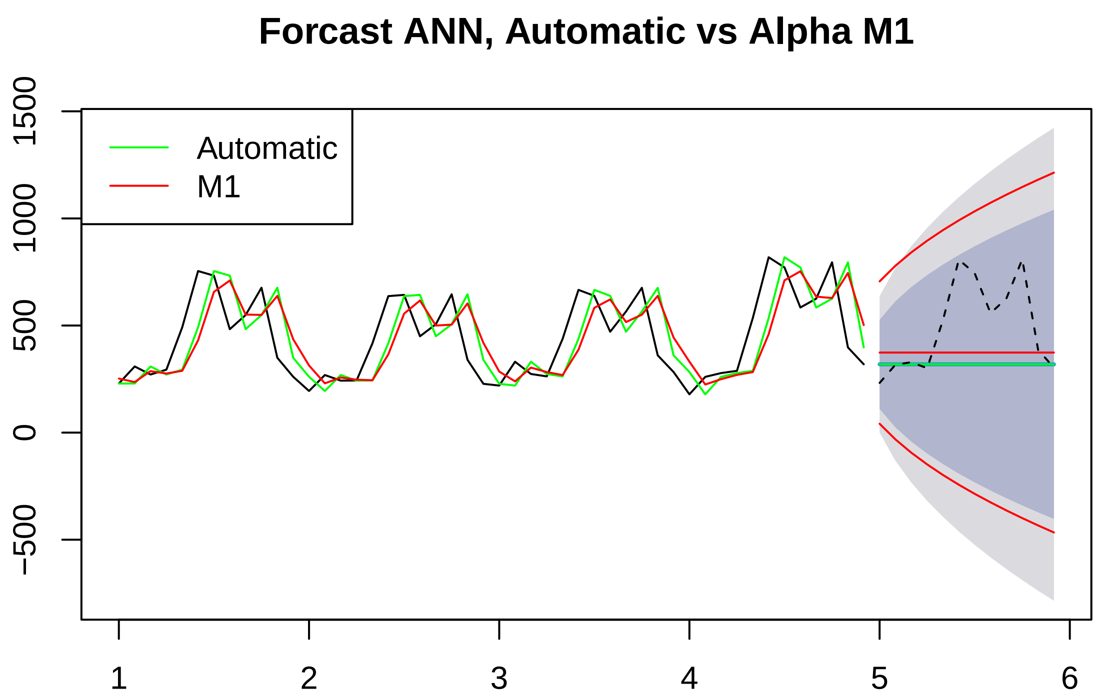
- Forecast ANN

```
# ploting Automatic ANN
frc <- forecast(fit, h=12)
plot(frc, main = "Forcast Automatic ANN")
lines(fit$fitted,col="green")
```



- Forecast ANN and Alpha M1

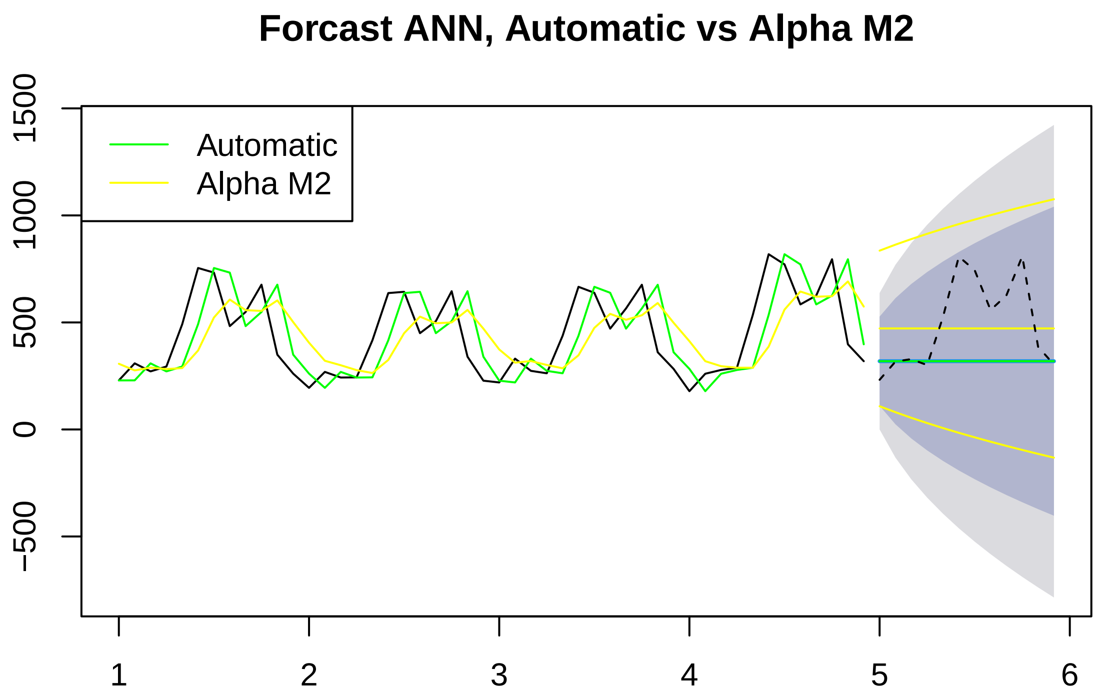
```r
# ploting M1 vs ANN
frc_m1 <- forecast(fit_m1,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M1")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red") # 95% upper
lines(y.tst,lty=2)

# legends
legend("topleft",c("Automatic","M1"),col=c("green","red"),lty=1)
```



- Forecast ANN and Alpha M2

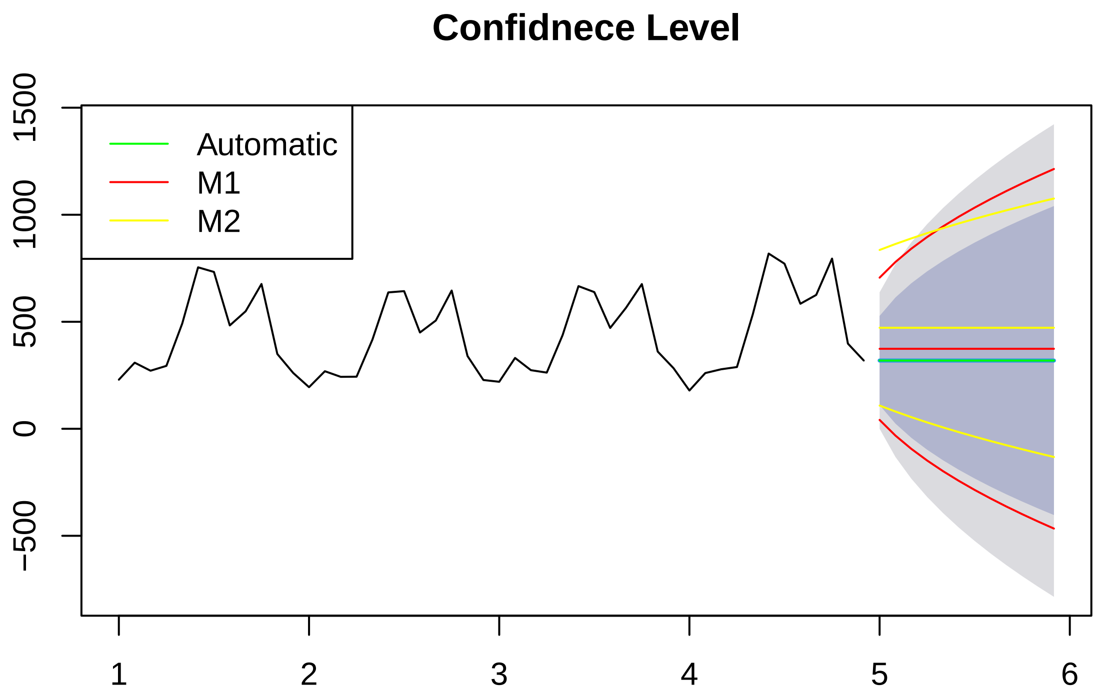
```r
# Ploting M2 vs ANN
frc_m2 <- forecast(fit_m2,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M2")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m2$fitted,col="yellow")

lines(frc_m2$mean,col="yellow")
```

```r
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper
lines(y.tst,lty=2)

# Add legend to the plot
legend("topleft",c("Automatic","Alpha M2"),col=c("green","yellow"),lty=1)
```

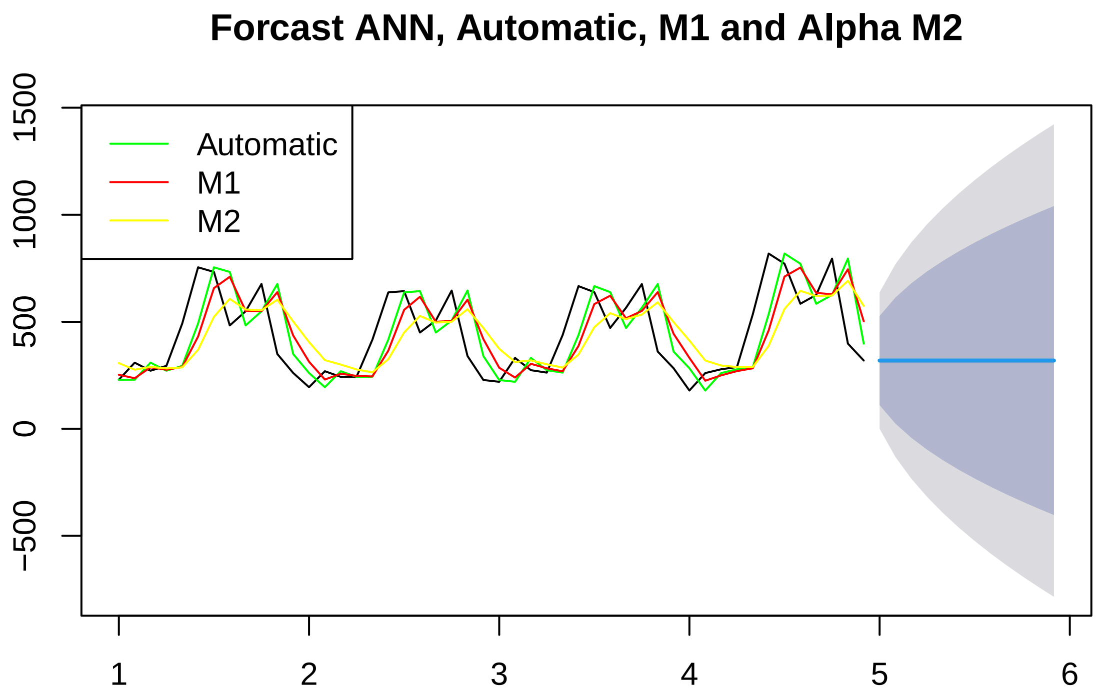


- Confidence level ANN, Alpha M1 and Alpha M2

```r
# Ploting confidence level of all 3
plot(frc, main = "Confidnece Level")
lines(frc$mean,col="green")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red")

lines(frc_m2$mean,col="yellow")
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper

# Add legend to the plot
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```




```r
plot(frc, main = "Forcast ANN, Automatic, M1 and Alpha M2")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")
lines(fit_m2$fitted,col="yellow")
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```



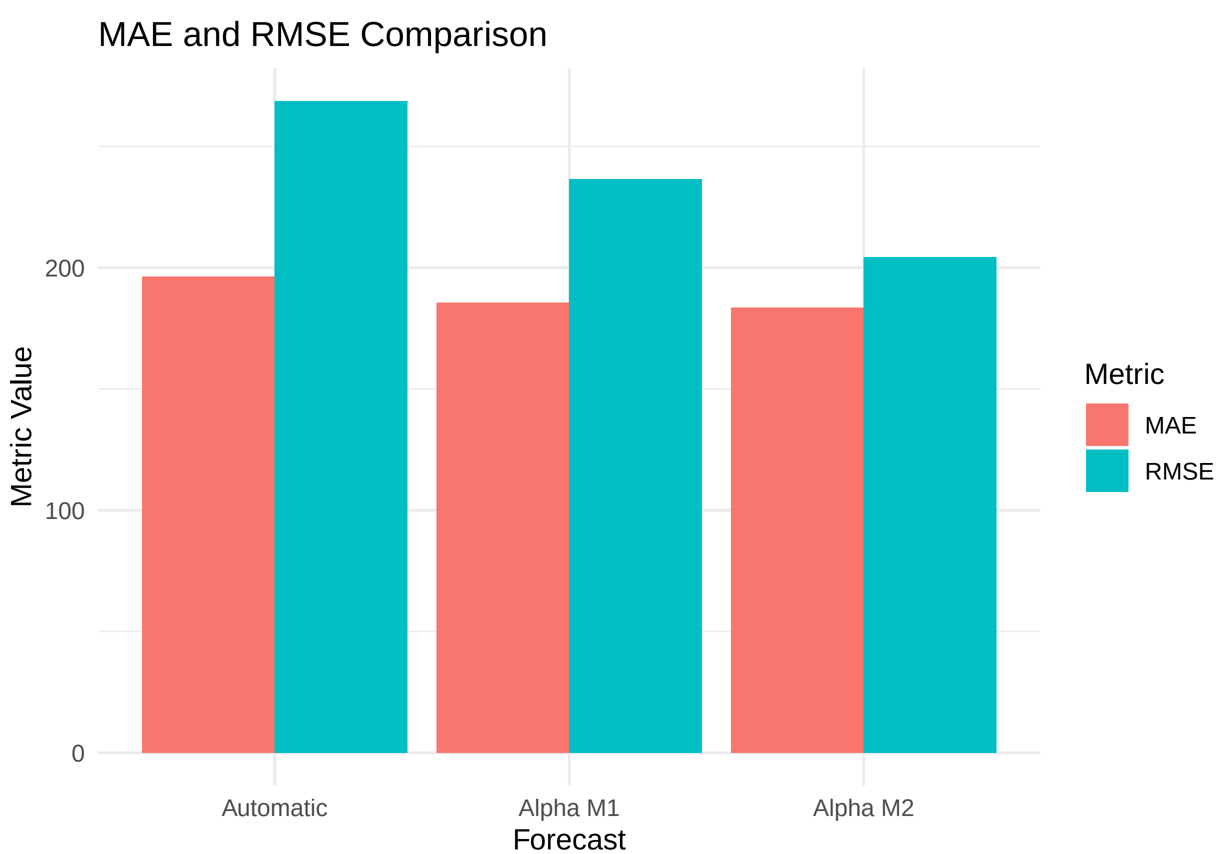

```
# Function to calculate metrics
calculate_metrics <- function(y, frc) {
  MAE <- mean(abs(y - frc$mean))
  MSE <- mean((y - frc$mean)^2)
  RMSE <- sqrt(MSE)
  return(list(MAE = MAE, MSE = MSE, RMSE = RMSE))
}

# Calculate metrics for different forecasts
metrics_a <- calculate_metrics(y.tst, frc)
metrics_m1 <- calculate_metrics(y.tst, frc_m1)
metrics_m2 <- calculate_metrics(y.tst, frc_m2)

# Create a matrix for the metrics
metrics_matrix <- matrix(c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
                           metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE,
                          metrics_a$MSE, metrics_m1$MSE, metrics_m2$MSE),
                          ncol = 3, byrow = TRUE)

# Add row and column names
rownames(metrics_matrix) <- c("MAE", "MSE", "RMSE")
colnames(metrics_matrix) <- c("Automatic", "Alpha M1", "Alpha M2")
```

```r
# Display the metrics matrix
metrics_matrix
```

1.6.4.3 4.3 Model selection

```
##       Automatic   Alpha M1   Alpha M2
## MAE    196.4063   185.6204   183.5548
## MSE    268.7739   236.4825   204.4416
## RMSE 72239.4193 55923.9927 41796.3854
```

```r
# Create a data frame from the metrics matrix, excluding MSE
metrics_df <- data.frame(
  Metric = rep(c("MAE", "RMSE"), each = 3),
  Value = c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
            metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE),
  Forecast = rep(c("Automatic", "Alpha M1", "Alpha M2"), times = 2)
)
metrics_df$Forecast <- factor(metrics_df$Forecast, levels = c("Automatic", "Alpha M1", "Alpha M2"))

# Create a bar plot
ggplot(metrics_df, aes(x = Forecast, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "MAE and RMSE Comparison",
       y = "Metric Value") +
  theme_minimal()
```



1.6.5 5. Answers

- Which one is best using your judgement?

  In the context of modeling the "Season_B" component within the in-sample data, it is observed that Model M1 excels in filtering out noise and outliers while providing a good fit to the data. Comparatively, Model M2, although successful in noise and outlier reduction, falls short in capturing the seasonality effectively. The automatic model, while fitting the data well in terms of seasonality, struggles to filter out noise and outliers.

  Given these considerations, the judgment leans toward **Model M1** as it achieves a balance between noise reduction and fitting performance.

- Which one is best using errors?

  The error metrics, MSE and AIC, suggest that the **Automatic** model performs better in terms of error minimization. This implies that the automatic model provides the most accurate predictions for the "Season_B" component within the in-sample data.

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

  In the out-of-sample analysis, Model M1 does not emerge as the best choice, as it does not yield the best RMSE and MAE values. **Model M2** exhibits superior performance in terms of these metrics.

  It is essential to acknowledge that while Model M1 excels in filtering out noise and outliers in the in-sample data while fitting well, its out-of-sample performance may differ. Model M2, despite its limitations in fitting, appears to generalize better to unseen data points for forecasting "Season_B."

1.7 Trend Season

1.7.1 1. Loading Data

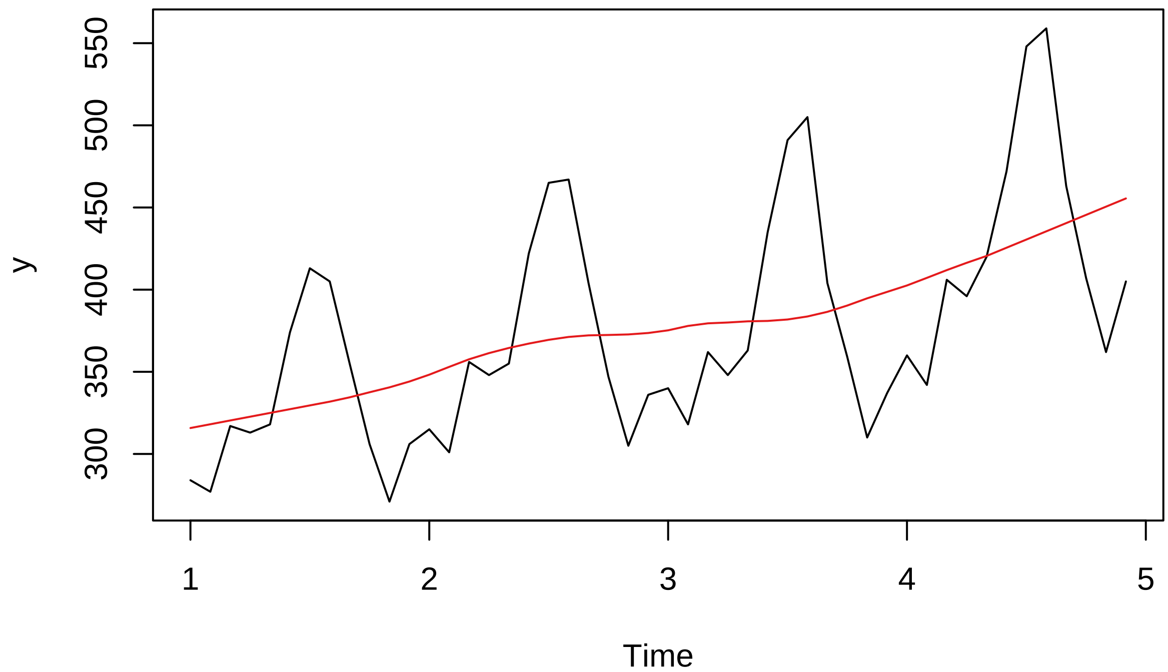
```r
y <- Y[,8]
# Transform it into a time series
y <- ts(y,frequency=12)
```

1.7.2 2. Constructing estimation and hold-out sets

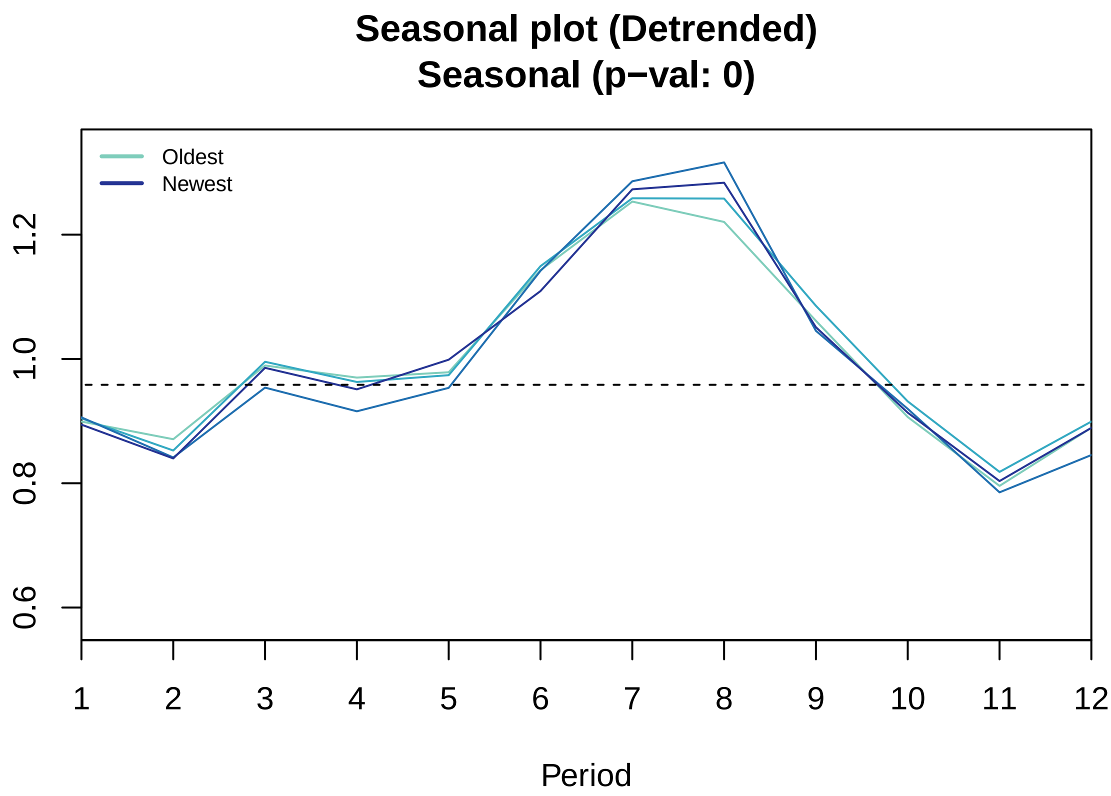
```r
y.tst <- tail(y,12)
y.trn <- head(y,48)
```

1.7.3 3. Exploration

```r
cma <- cmav(y.trn,outplot=1)
```




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




```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0)
## Evidence of seasonality: TRUE  (pval: 0)
```

1.7.4 4. Forecasting

```r
# Automatic Alpha ANN
fit <- ets(y.trn,model="ANN")
print(fit)
```

1.7.4.1 4.1 Model fitting

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.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
```

```
# Alpha M1 ANN
fit_m1 <- ets(y.trn,model="ANN", alpha = 0.7 )
print(fit_m1)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.7)
##
##   Smoothing parameters:
##     alpha = 0.7
##
##   Initial states:
##     l = 285.6387
##
##   sigma:  50.046
##
##      AIC     AICc      BIC
## 563.4173 563.6840 567.1597
```

```
# Alpha M2 ANN
fit_m2 <- ets(y.trn,model="ANN", alpha = 0.4 )
print(fit_m2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.4)
##
##   Smoothing parameters:
##     alpha = 0.4
##
##   Initial states:
##     l = 298.5088
##
##   sigma:  57.9453
##
##      AIC     AICc      BIC
## 577.4867 577.7534 581.2291
```

```
cirt <- array(NA, c(3, 4), dimnames = list(c("Automatic", "M1", "M2"),
                                          c("MSE", "AIC", "AICc","BIC")))

models <- list(fit, fit_m1, fit_m2)

for (i in 1:3) {
```

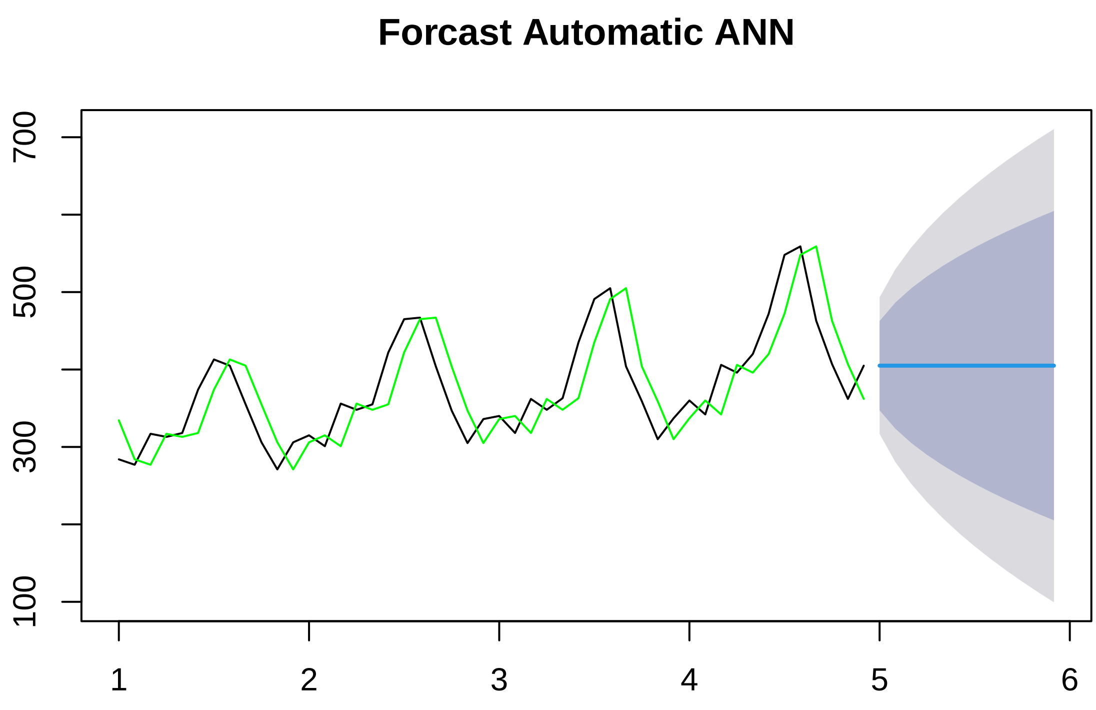
```
  cirt[i, "MSE"] <- models[[i]]$mse
  cirt[i, "AIC"] <- models[[i]]$aic
  cirt[i, "AICc"] <- models[[i]]$aicc
  cirt[i, "BIC"] <- models[[i]]$bic
}

print(cirt)
```

```
##                MSE      AIC     AICc      BIC
## Automatic 1941.454 555.2349 555.7803 560.8485
## M1        2400.245 563.4173 563.6840 567.1597
## M2        3217.755 577.4867 577.7534 581.2291
```

1.7.4.2 4.2 Forecasting

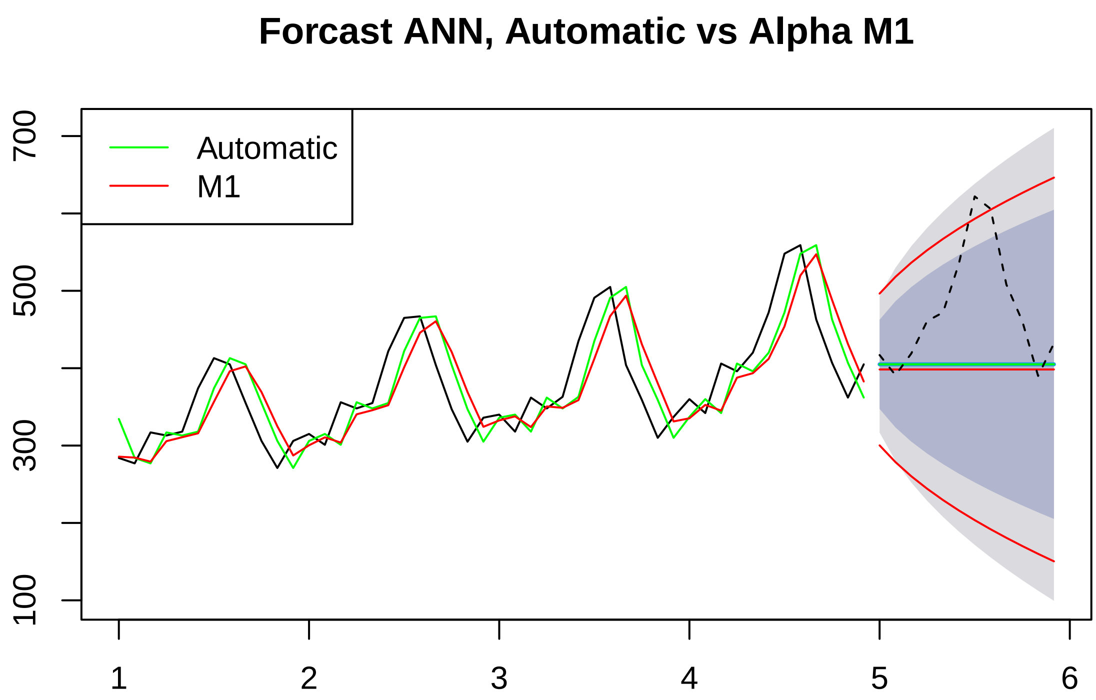
- Forecast ANN

```
# ploting Automatic ANN
frc <- forecast(fit, h=12)
plot(frc, main = "Forcast Automatic ANN")
lines(fit$fitted,col="green")
```



- Forecast ANN and Alpha M1

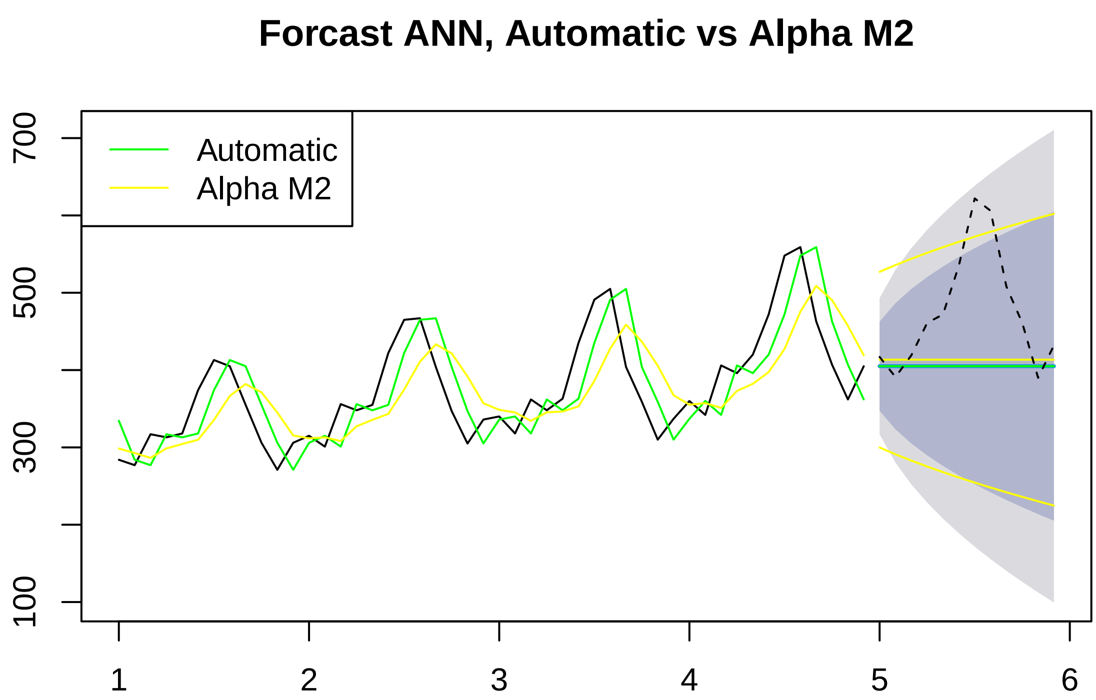
```r
# ploting M1 vs ANN
frc_m1 <- forecast(fit_m1,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M1")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red") # 95% upper
lines(y.tst,lty=2)

# legends
legend("topleft",c("Automatic","M1"),col=c("green","red"),lty=1)
```



- Forecast ANN and Alpha M2

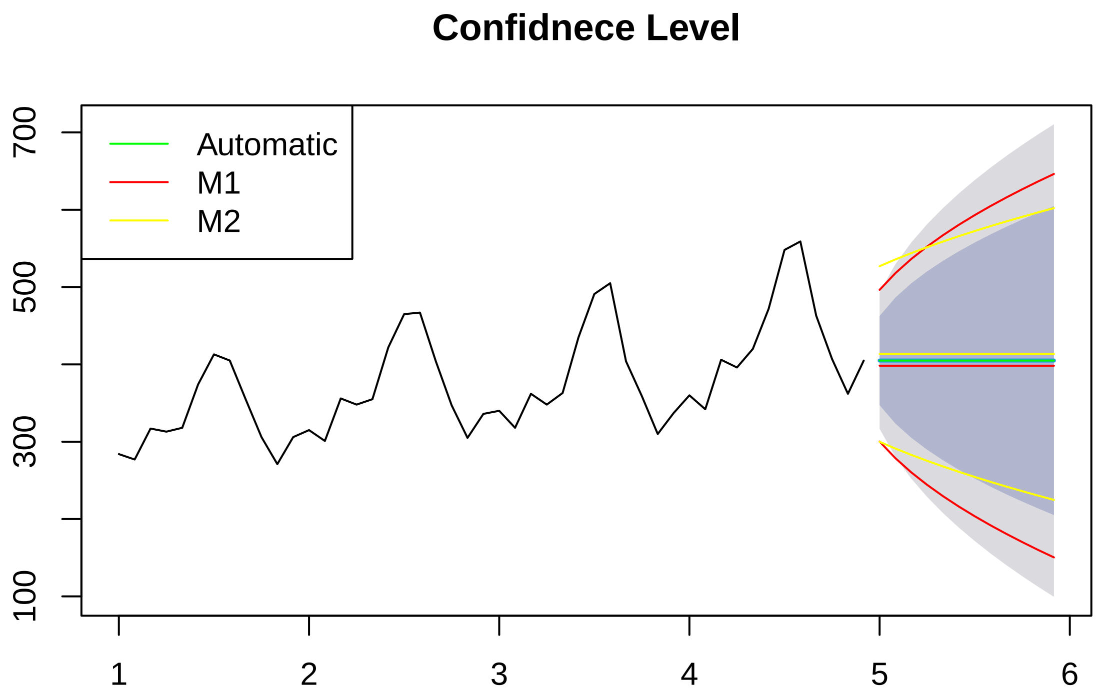
```r
# Ploting M2 vs ANN
frc_m2 <- forecast(fit_m2,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M2")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m2$fitted,col="yellow")

lines(frc_m2$mean,col="yellow")
```

```r
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper
lines(y.tst,lty=2)

# Add legend to the plot
legend("topleft",c("Automatic","Alpha M2"),col=c("green","yellow"),lty=1)
```

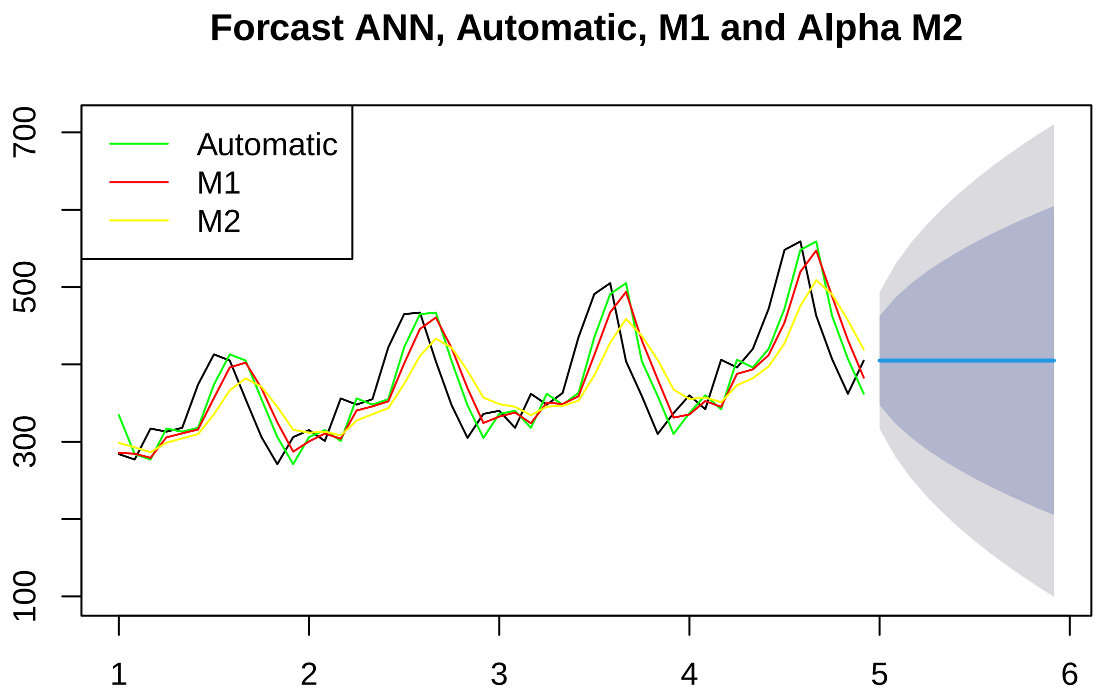


- Confidence level ANN, Alpha M1 and Alpha M2

```r
# Ploting confidence level of all 3
plot(frc, main = "Confidnece Level")
lines(frc$mean,col="green")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red")

lines(frc_m2$mean,col="yellow")
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper

# Add legend to the plot
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```




```r
plot(frc, main = "Forcast ANN, Automatic, M1 and Alpha M2")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")
lines(fit_m2$fitted,col="yellow")
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```



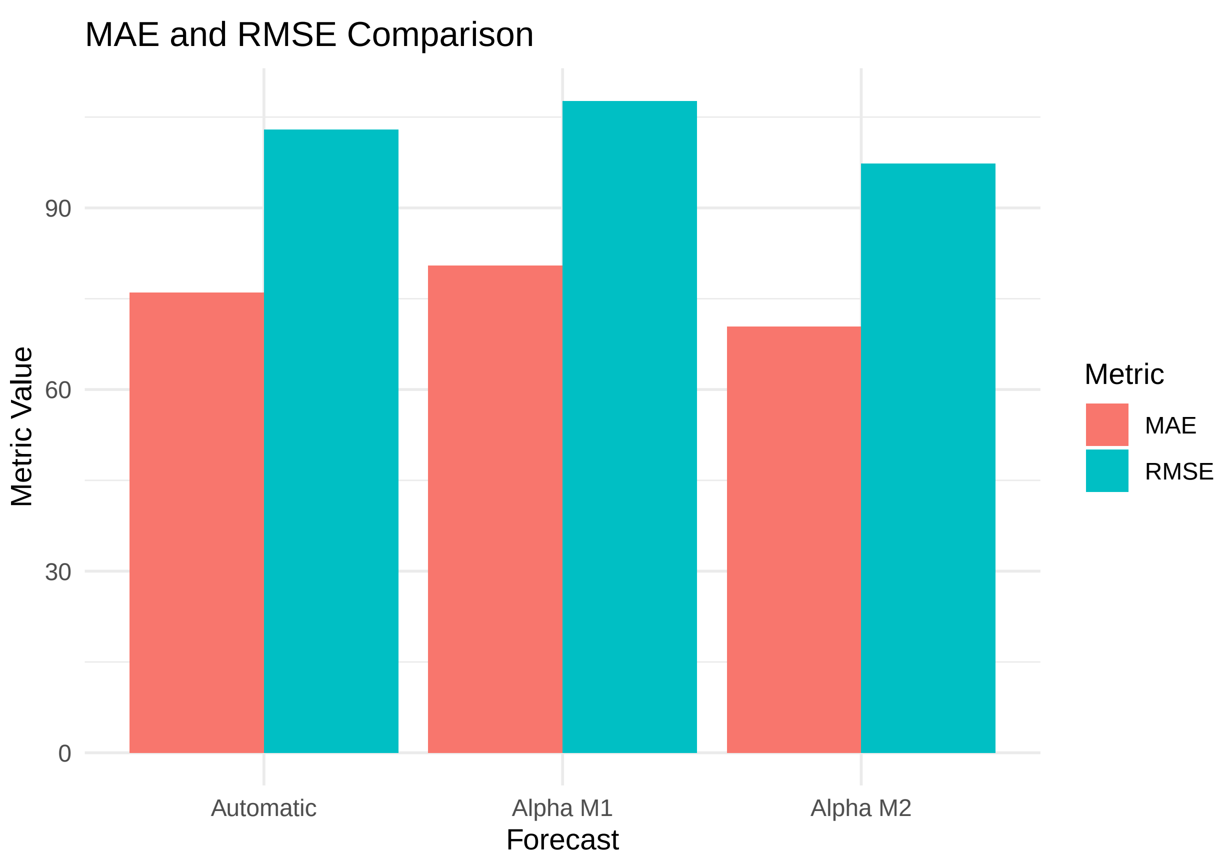

```r
# Function to calculate metrics
calculate_metrics <- function(y, frc) {
  MAE <- mean(abs(y - frc$mean))
  MSE <- mean((y - frc$mean)^2)
  RMSE <- sqrt(MSE)
  return(list(MAE = MAE, MSE = MSE, RMSE = RMSE))
}

# Calculate metrics for different forecasts
metrics_a <- calculate_metrics(y.tst, frc)
metrics_m1 <- calculate_metrics(y.tst, frc_m1)
metrics_m2 <- calculate_metrics(y.tst, frc_m2)

# Create a matrix for the metrics
metrics_matrix <- matrix(c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
                           metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE,
                          metrics_a$MSE, metrics_m1$MSE, metrics_m2$MSE),
                          ncol = 3, byrow = TRUE)

# Add row and column names
rownames(metrics_matrix) <- c("MAE", "MSE", "RMSE")
colnames(metrics_matrix) <- c("Automatic", "Alpha M1", "Alpha M2")
```

```r
# Display the metrics matrix
metrics_matrix
```

1.7.4.3 4.3 Model selection

```
##         Automatic   Alpha M1   Alpha M2
## MAE      76.00287    80.4371   70.36908
## MSE     102.97951   107.6837   97.33092
## RMSE 10604.77875 11595.7842 9473.30712
```

```r
# Create a data frame from the metrics matrix, excluding MSE
metrics_df <- data.frame(
  Metric = rep(c("MAE", "RMSE"), each = 3),
  Value = c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
            metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE),
  Forecast = rep(c("Automatic", "Alpha M1", "Alpha M2"), times = 2)
)
metrics_df$Forecast <- factor(metrics_df$Forecast, levels = c("Automatic", "Alpha M1", "Alpha M2"))

# Create a bar plot
ggplot(metrics_df, aes(x = Forecast, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "MAE and RMSE Comparison",
       y = "Metric Value") +
  theme_minimal()
```



1.7.5 5. Answers

- Which one is best using your judgement?

  When analyzing the in-sample data for the "TrendSeason" component, it becomes evident that **Model M1** effectively filters out noise and outliers while achieving a satisfactory fit compared to the Automatic model. Conversely, Model M2, while excelling in noise and outlier reduction, falls short in capturing the seasonality pattern effectively. The Automatic model, though capable of capturing seasonality, does not effectively filter out noise and outliers.

- Which one is best using errors?

  Examining error metrics such as MSE and AIC, it becomes apparent that the **Automatic** model performs better in terms of error minimization. This indicates that the Automatic model provides the most accurate predictions for the "TrendSeason" component within the in-sample data.

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

  In the out-of-sample analysis, Model M1 does not emerge as the optimal choice. It does not yield the best RMSE and MAE values. On the contrary, Model M2 demonstrates superior performance in terms of these metrics.

  It is important to note that while Model M1 excels in filtering out noise and outliers and provides a good fit within the in-sample data, its out-of-sample performance may vary. **Model M2**, despite its limitations in capturing seasonality during fitting, appears to generalize better to unseen data points when forecasting "TrendSeason.

1.8 AirPassenger

1.8.1 1. Loading Data

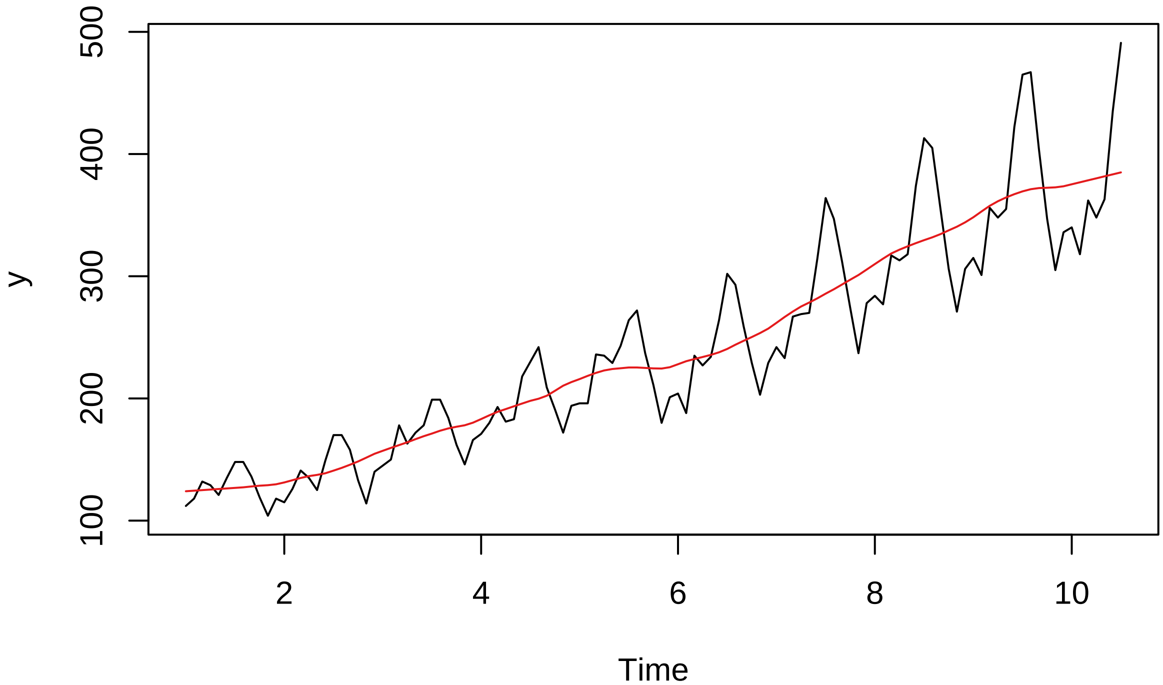
```r
y <- AirPassengers
# Transform it into a time series
y <- ts(y,frequency=12)
```

1.8.2 2. Constructing estimation and hold-out sets

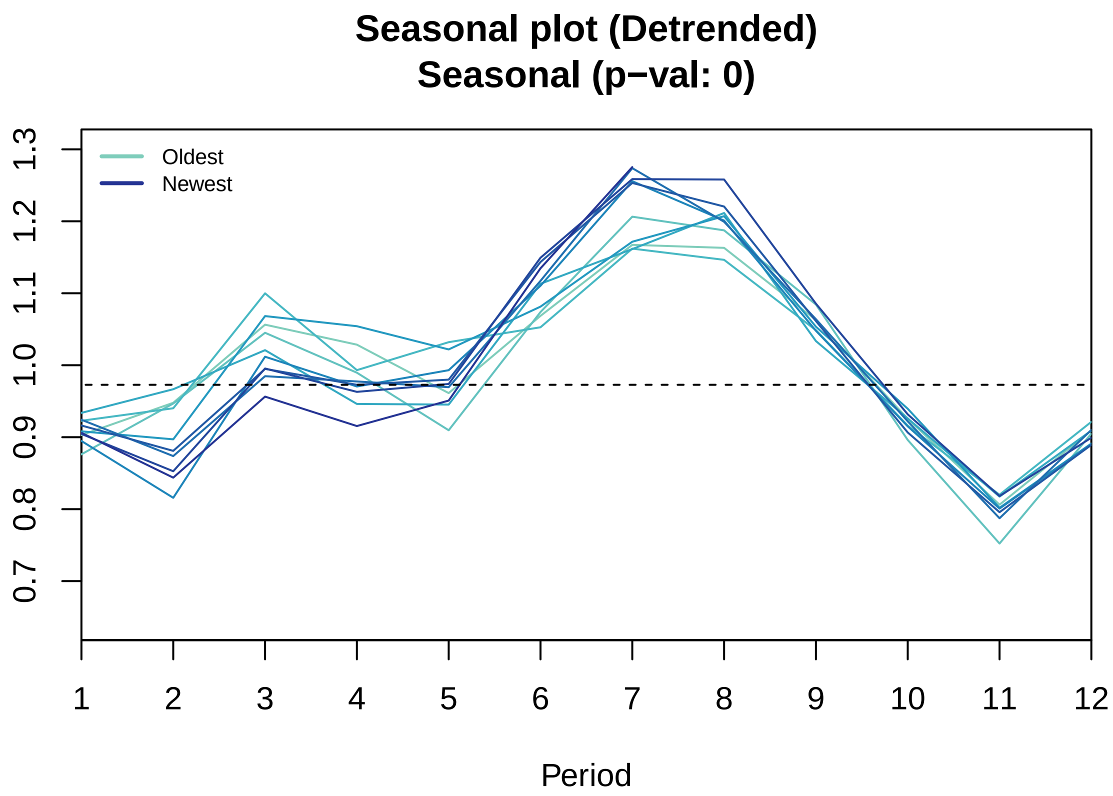
```r
y.tst <- tail(y,29)
y.trn <- head(y,115)
```

1.8.3 3. Exploration

```r
cma <- cmav(y.trn,outplot=1)
```




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




```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0)
## Evidence of seasonality: TRUE  (pval: 0)
```

1.8.4 4. Forecasting

```
# Automatic Alpha ANN
fit <- ets(y.trn,model="ANN")
print(fit)
```

1.8.4.1 4.1 Model fitting

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 112.0508
```

```
##
##   sigma:  27.0371
##
##      AIC     AICc      BIC
## 1308.008 1308.224 1316.243
```

```r
# Alpha M1 ANN
fit_m1 <- ets(y.trn,model="ANN", alpha = 0.7 )
print(fit_m1)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.7)
##
##   Smoothing parameters:
##     alpha = 0.7
##
##   Initial states:
##     l = 114.9947
##
##   sigma:  30.1427
##
##      AIC     AICc      BIC
## 1331.016 1331.124 1336.506
```

```r
# Alpha M2 ANN
fit_m2 <- ets(y.trn,model="ANN", alpha = 0.4 )
print(fit_m2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = y.trn, model = "ANN", alpha = 0.4)
##
##   Smoothing parameters:
##     alpha = 0.4
##
##   Initial states:
##     l = 120.065
##
##   sigma:  34.8518
##
##      AIC     AICc      BIC
## 1364.404 1364.511 1369.894
```

```r
cirt <- array(NA, c(3, 4), dimnames = list(c("Automatic", "M1", "M2"),
                                          c("MSE", "AIC", "AICc","BIC")))

models <- list(fit, fit_m1, fit_m2)

for (i in 1:3) {
```

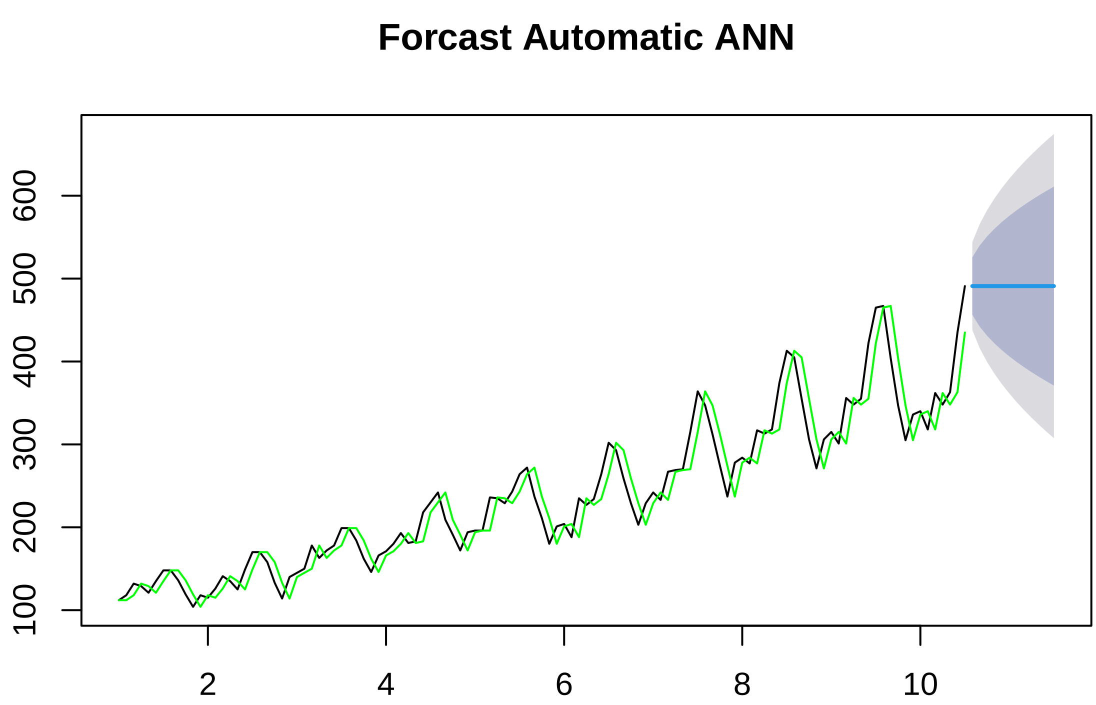
```
  cirt[i, "MSE"] <- models[[i]]$mse
  cirt[i, "AIC"] <- models[[i]]$aic
  cirt[i, "AICc"] <- models[[i]]$aicc
  cirt[i, "BIC"] <- models[[i]]$bic
}

print(cirt)
```

```
##                  MSE      AIC     AICc      BIC
## Automatic  718.2927 1308.008 1308.224 1316.243
## M1         892.7806 1331.016 1331.124 1336.506
## M2        1193.5265 1364.404 1364.511 1369.894
```

1.8.4.2 4.2 Forecasting

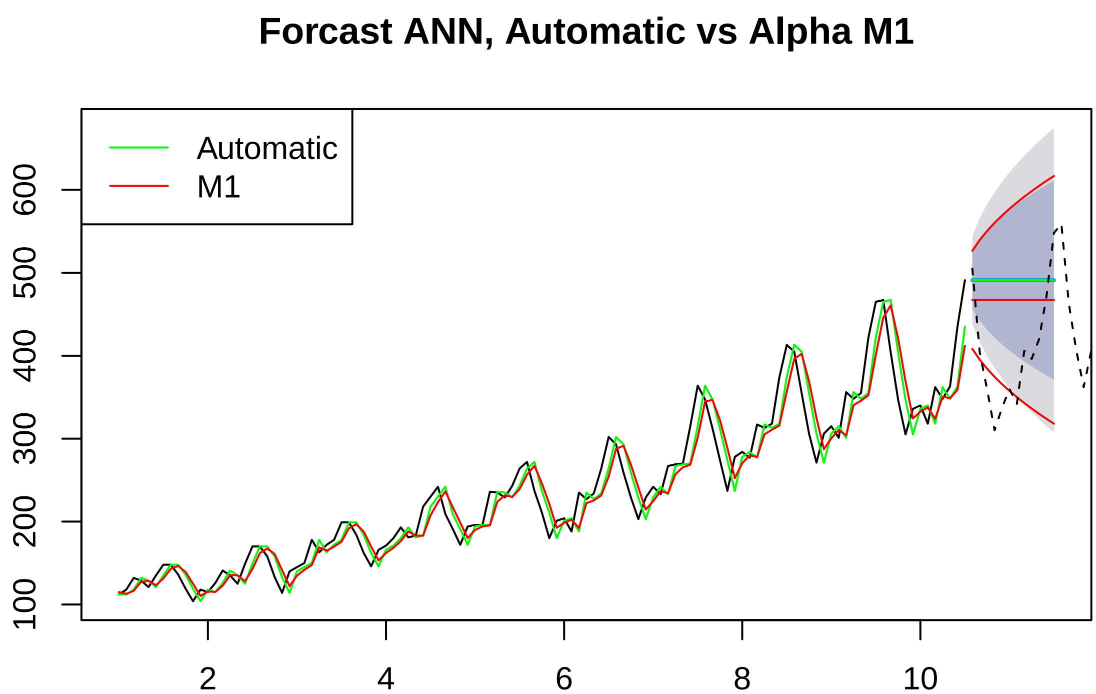
- Forecast ANN

```
# ploting Automatic ANN
frc <- forecast(fit, h=12)
plot(frc, main = "Forcast Automatic ANN")
lines(fit$fitted,col="green")
```



- Forecast ANN and Alpha M1

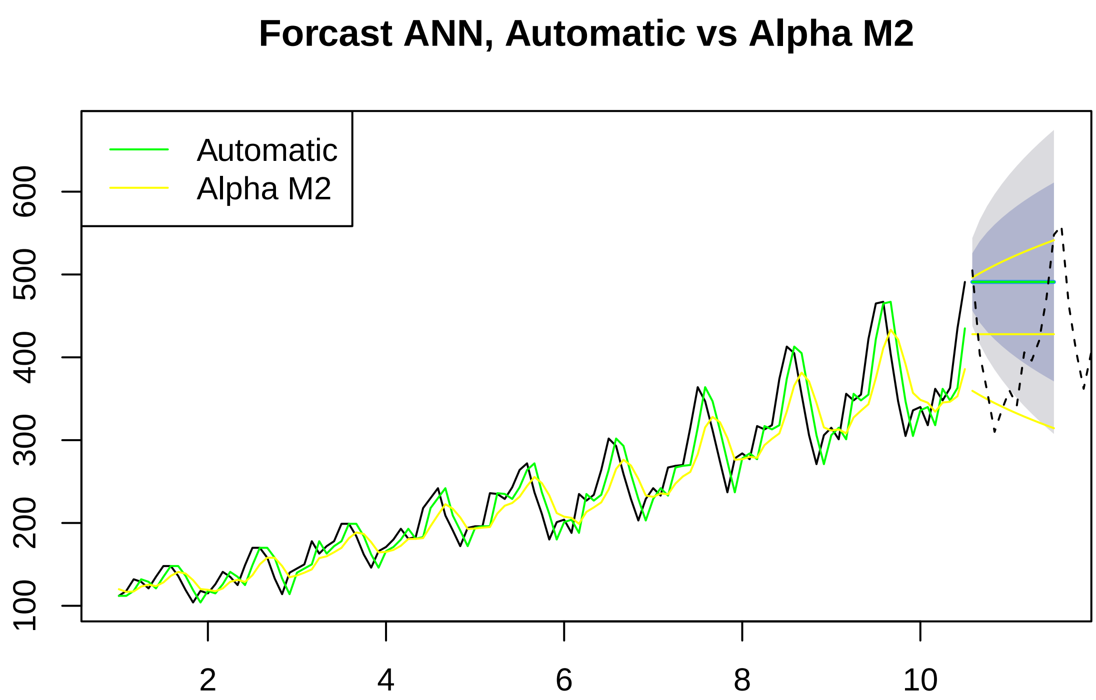
```r
# ploting M1 vs ANN
frc_m1 <- forecast(fit_m1,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M1")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red") # 95% upper
lines(y.tst,lty=2)

# legends
legend("topleft",c("Automatic","M1"),col=c("green","red"),lty=1)
```



- Forecast ANN and Alpha M2

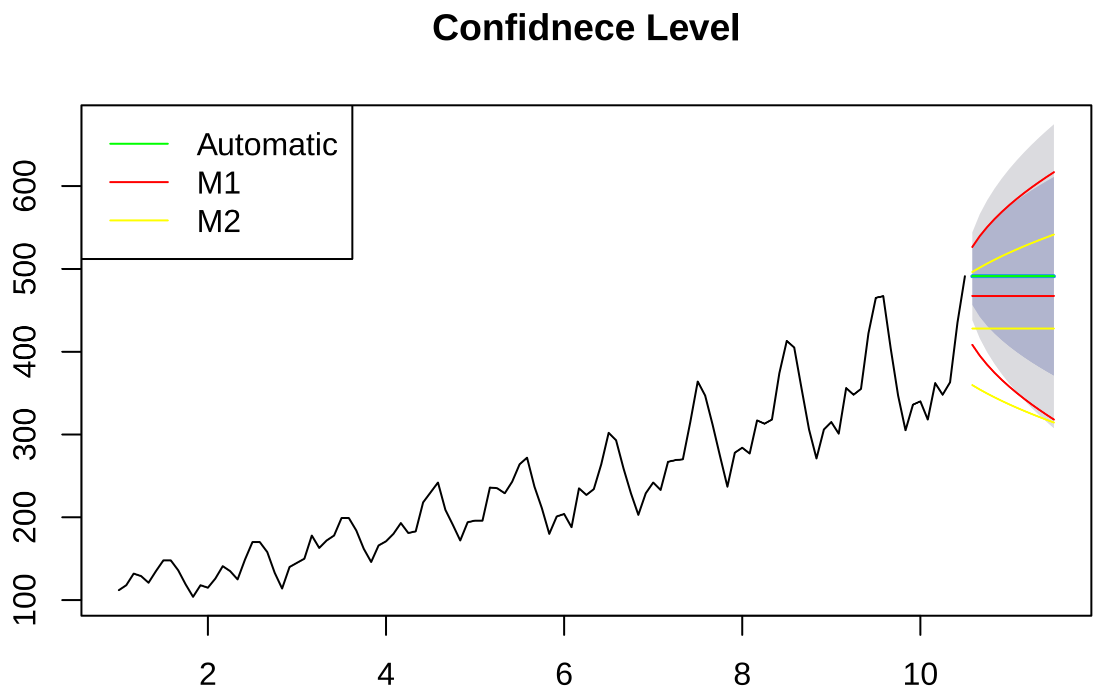
```r
# Ploting M2 vs ANN
frc_m2 <- forecast(fit_m2,h=12)
plot(frc, main = "Forcast ANN, Automatic vs Alpha M2")
lines(frc$mean,col="green")
lines(fit$fitted,col="green")
lines(fit_m2$fitted,col="yellow")

lines(frc_m2$mean,col="yellow")
```

```
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper
lines(y.tst,lty=2)

# Add legend to the plot
legend("topleft",c("Automatic","Alpha M2"),col=c("green","yellow"),lty=1)
```

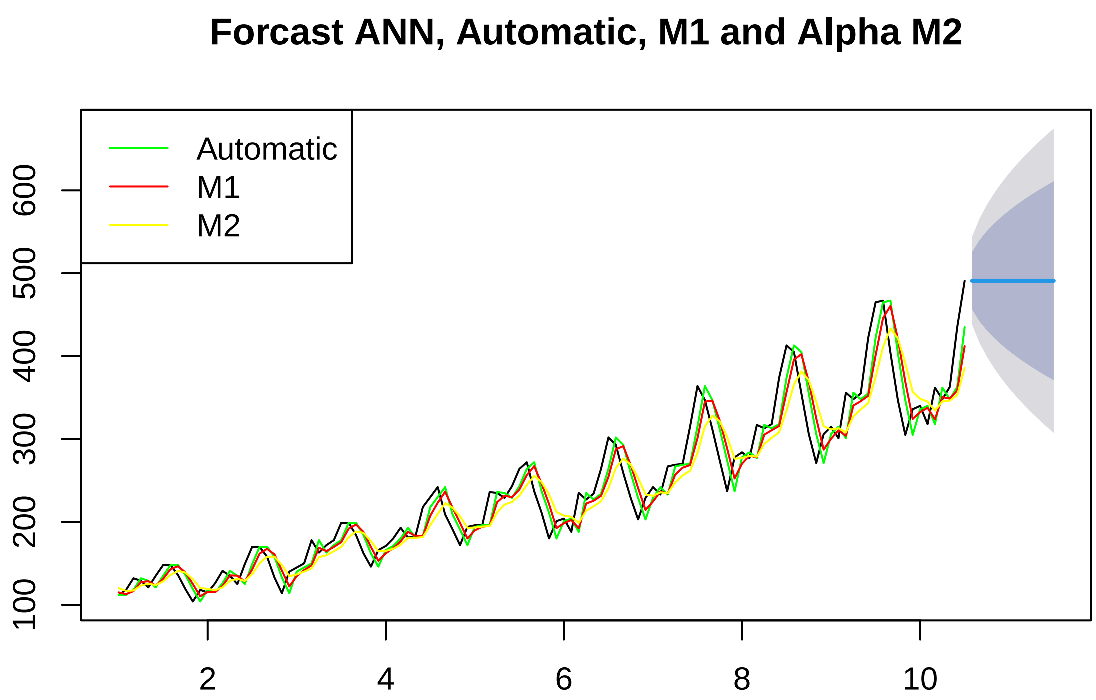


- Confidence level ANN, Alpha M1 and Alpha M2

```
# Ploting confidence level of all 3
plot(frc, main = "Confidnece Level")
lines(frc$mean,col="green")

lines(frc_m1$mean,col="red")
lines(frc_m1$lower[,2],col="red") # 95% lower
lines(frc_m1$upper[,2],col="red")

lines(frc_m2$mean,col="yellow")
lines(frc_m2$lower[,2],col="yellow") # 95% lower
lines(frc_m2$upper[,2],col="yellow") # 95% upper

# Add legend to the plot
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```




```
plot(frc, main = "Forcast ANN, Automatic, M1 and Alpha M2")
lines(fit$fitted,col="green")
lines(fit_m1$fitted,col="red")
lines(fit_m2$fitted,col="yellow")
legend("topleft",c("Automatic","M1","M2"),col=c("green","red","yellow"),lty=1)
```



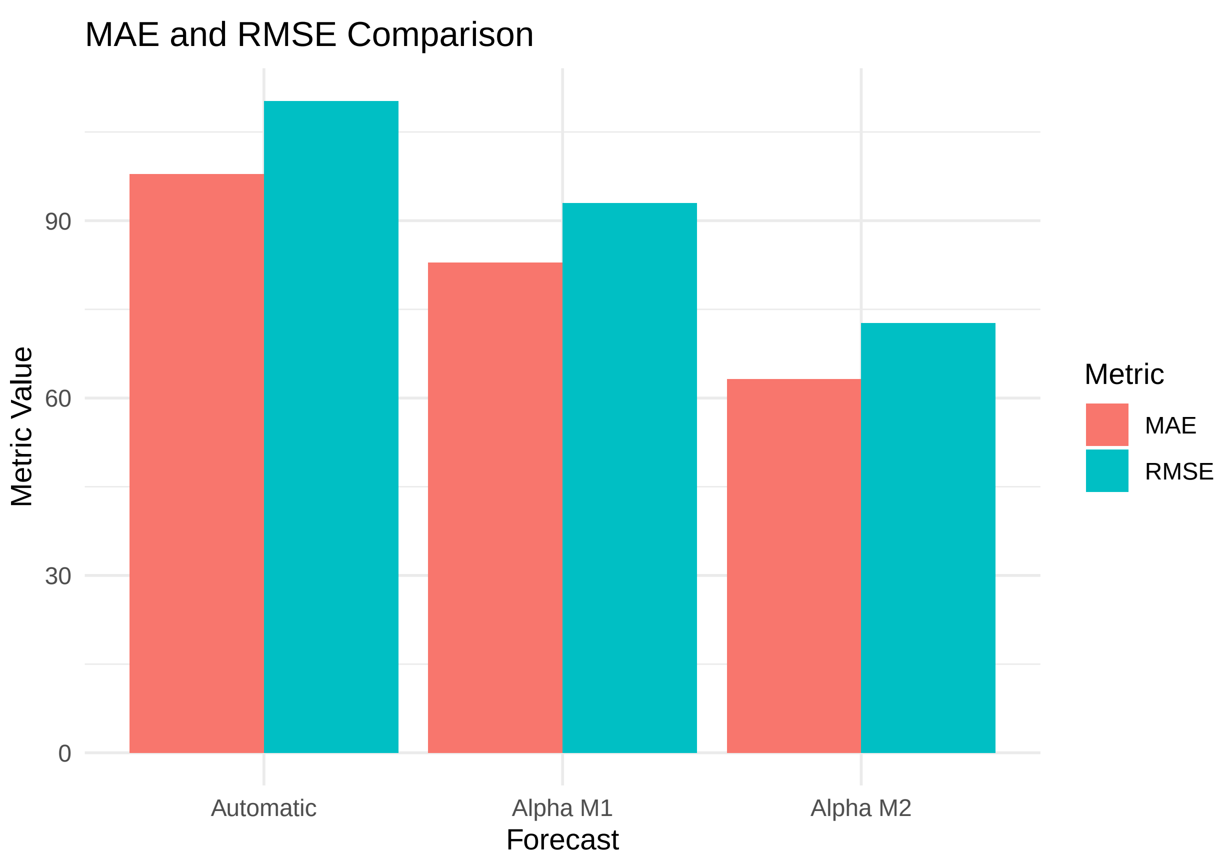

```r
# Function to calculate metrics
calculate_metrics <- function(y, frc) {
  MAE <- mean(abs(y - frc$mean))
  MSE <- mean((y - frc$mean)^2)
  RMSE <- sqrt(MSE)
  return(list(MAE = MAE, MSE = MSE, RMSE = RMSE))
}

# Calculate metrics for different forecasts
metrics_a <- calculate_metrics(y.tst, frc)
metrics_m1 <- calculate_metrics(y.tst, frc_m1)
metrics_m2 <- calculate_metrics(y.tst, frc_m2)

# Create a matrix for the metrics
metrics_matrix <- matrix(c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
                           metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE,
                          metrics_a$MSE, metrics_m1$MSE, metrics_m2$MSE),
                          ncol = 3, byrow = TRUE)

# Add row and column names
rownames(metrics_matrix) <- c("MAE", "MSE", "RMSE")
colnames(metrics_matrix) <- c("Automatic", "Alpha M1", "Alpha M2")
```

```r
# Display the metrics matrix
metrics_matrix
```

1.8.4.3 4.3 Model selection

```
##         Automatic   Alpha M1   Alpha M2
## MAE      97.91293   82.91794   63.20054
## MSE     110.27141   92.98692   72.65461
## RMSE 12159.78442 8646.56703 5278.69264
```

```r
# Create a data frame from the metrics matrix, excluding MSE
metrics_df <- data.frame(
  Metric = rep(c("MAE", "RMSE"), each = 3),
  Value = c(metrics_a$MAE, metrics_m1$MAE, metrics_m2$MAE,
            metrics_a$RMSE, metrics_m1$RMSE, metrics_m2$RMSE),
  Forecast = rep(c("Automatic", "Alpha M1", "Alpha M2"), times = 2)
)
metrics_df$Forecast <- factor(metrics_df$Forecast, levels = c("Automatic", "Alpha M1", "Alpha M2"))

# Create a bar plot
ggplot(metrics_df, aes(x = Forecast, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "MAE and RMSE Comparison",
       y = "Metric Value") +
  theme_minimal()
```



1.8.5 5. Answers

- Which one is best using your judgement?

  Upon examining the in-sample data for the "AirPassengers" component, it becomes evident that **Model M1** effectively mitigates noise and outliers while achieving a robust fit compared to the Automatic model. Conversely, Model M2, while less successful in capturing seasonality during fitting, excels in noise and outlier reduction. The Automatic model, regrettably, does not effectively filter out noise and outliers.

  Given these insights, Model M1 emerges as the preferred choice due to its balanced approach, prioritizing noise reduction and fitting performance

- Which one is best using errors?

  When considering error metrics such as MSE and AIC, it becomes clear that the **Automatic** model performs better in terms of error minimization. This suggests that the Automatic model provides the most accurate predictions for the "AirPassengers" component within the in-sample data.

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

  in the out-of-sample analysis, Model M1 does not prove to be the optimal choice. It does not yield the best RMSE and MAE values. Conversely, Model M2 exhibits superior performance in terms of these metrics.

  It is essential to acknowledge that while Model M1 excels in noise and outlier reduction and provides a strong fit within the in-sample data, its out-of-sample performance may differ. **Model M2**, despite its limitations in fitting seasonality, appears to generalize better to unseen data points when forecasting "AirPassengers."