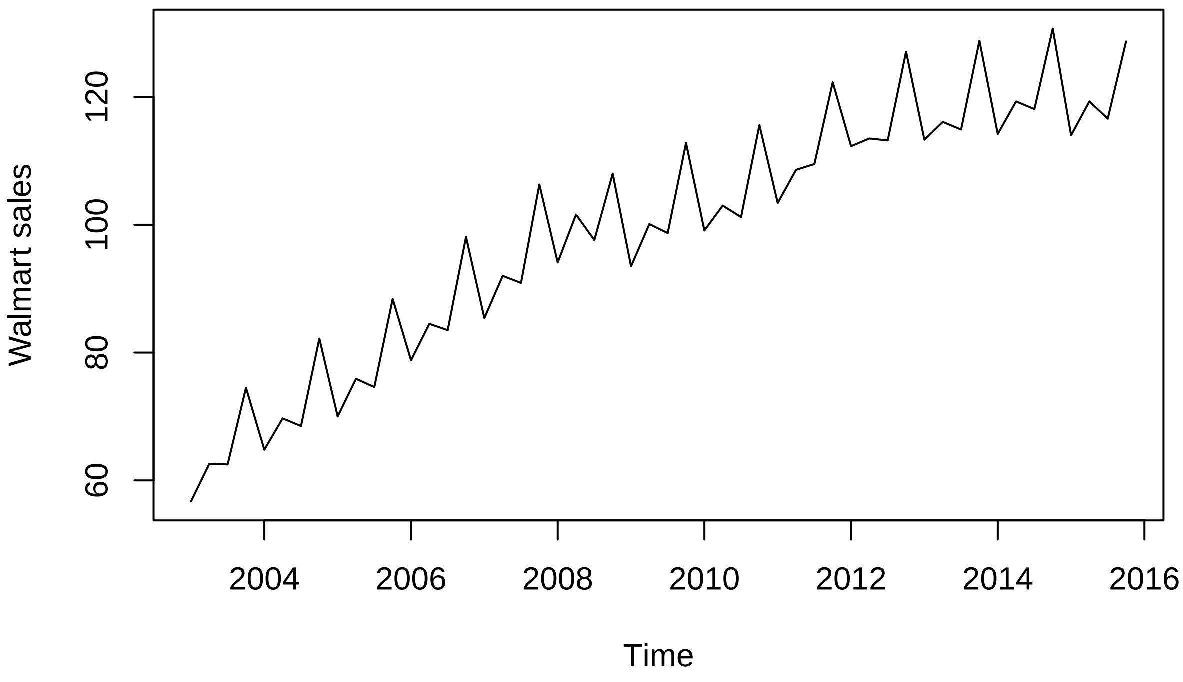
Lab5_Lasso_a24kimwu

2025-10-01

Load dataset

```
x <- ts(read.csv("./walmart.csv"),frequency=4,start=c(2003,1))
plot(x[,1],ylab="Walmart sales")
```



Build a benchmark regression

```
y.trn <- window(x[,1],end=c(2013,4))
y.tst <- window(x[,1],start=c(2014,1))
```

```
n <- length(y.trn)
X <- array(NA,c(n,6))
# Loop to create lags
for (i in 1:6){
 X[i:n,i] <- y.trn[1:(n-i+1)]
}
# Name the columns
colnames(X) <- c("y",paste0("lag",1:5))
X <- as.data.frame(X)
head(X)
```

```
##      y lag1 lag2 lag3 lag4 lag5
## 1 56.7   NA   NA   NA   NA   NA
## 2 62.6 56.7   NA   NA   NA   NA
## 3 62.5 62.6 56.7   NA   NA   NA
## 4 74.5 62.5 62.6 56.7   NA   NA
## 5 64.8 74.5 62.5 62.6 56.7   NA
## 6 69.7 64.8 74.5 62.5 62.6 56.7
```

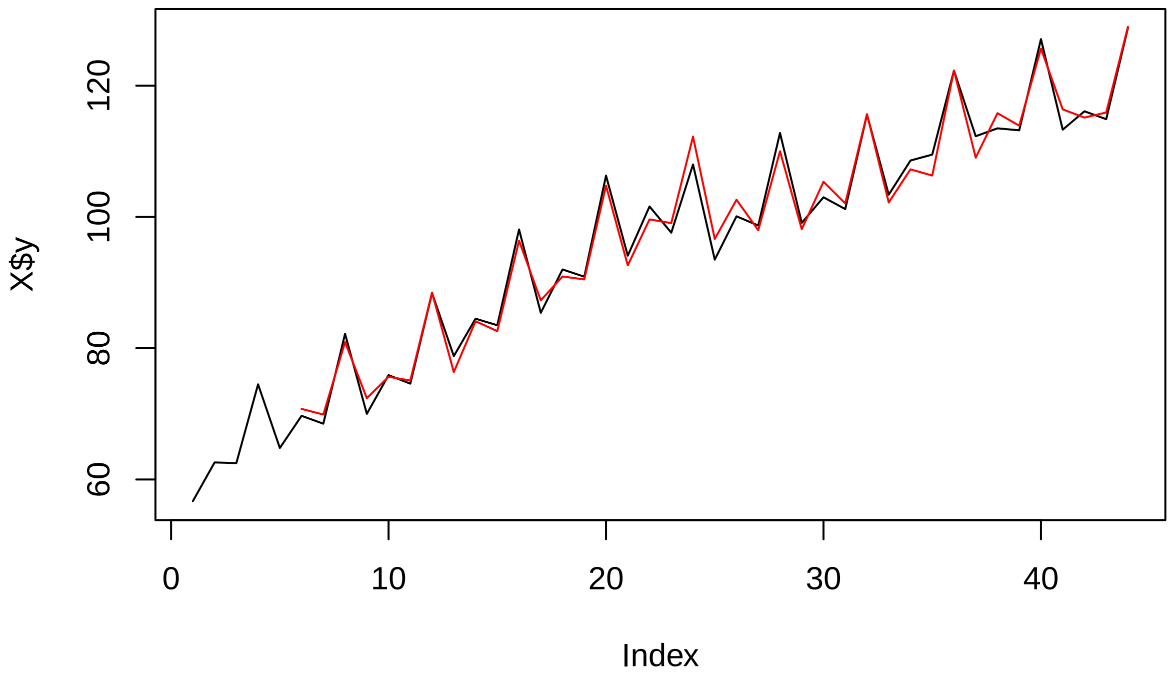
```
# The complete model
fit1 <- lm(y~.,data=X)
# The stepwise model
fit2 <- step(fit1)
```

```
## Start:  AIC=58.18
## y ~ lag1 + lag2 + lag3 + lag4 + lag5
##
##        Df Sum of Sq     RSS     AIC
## - lag2  1      0.35  127.79  56.286
## - lag3  1      1.28  128.71  56.567
## <none>               127.44  58.178
## - lag5  1    105.04  232.48  79.624
## - lag1  1    110.17  237.61  80.475
## - lag4  1   1590.97 1718.40 157.638
##
## Step:  AIC=56.29
## y ~ lag1 + lag3 + lag4 + lag5
##
##        Df Sum of Sq     RSS     AIC
## - lag3  1      1.51  129.29  54.743
## <none>               127.79  56.286
## - lag5  1    104.99  232.78  77.674
## - lag1  1    111.14  238.92  78.691
## - lag4  1   2717.34 2845.12 175.302
##
## Step:  AIC=54.74
## y ~ lag1 + lag4 + lag5
##
##        Df Sum of Sq     RSS     AIC
## <none>               129.29  54.743
## - lag1  1    110.09  239.39  76.766
## - lag5  1    116.20  245.50  77.749
## - lag4  1   2910.88 3040.17 175.888
```

```
summary(fit2)
```

```
##
## Call:
## lm(formula = y ~ lag1 + lag4 + lag5, data = X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2420 -1.2261  0.2523  1.3036  3.2640
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.5873     2.4497   1.873   0.0695 .
## lag1          0.6783     0.1242   5.459 3.99e-06 ***
## lag4          0.9824     0.0350  28.071  < 2e-16 ***
## lag5         -0.6927     0.1235  -5.609 2.53e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.922 on 35 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.9868, Adjusted R-squared:  0.9856
## F-statistic: 870.6 on 3 and 35 DF,  p-value: < 2.2e-16
```

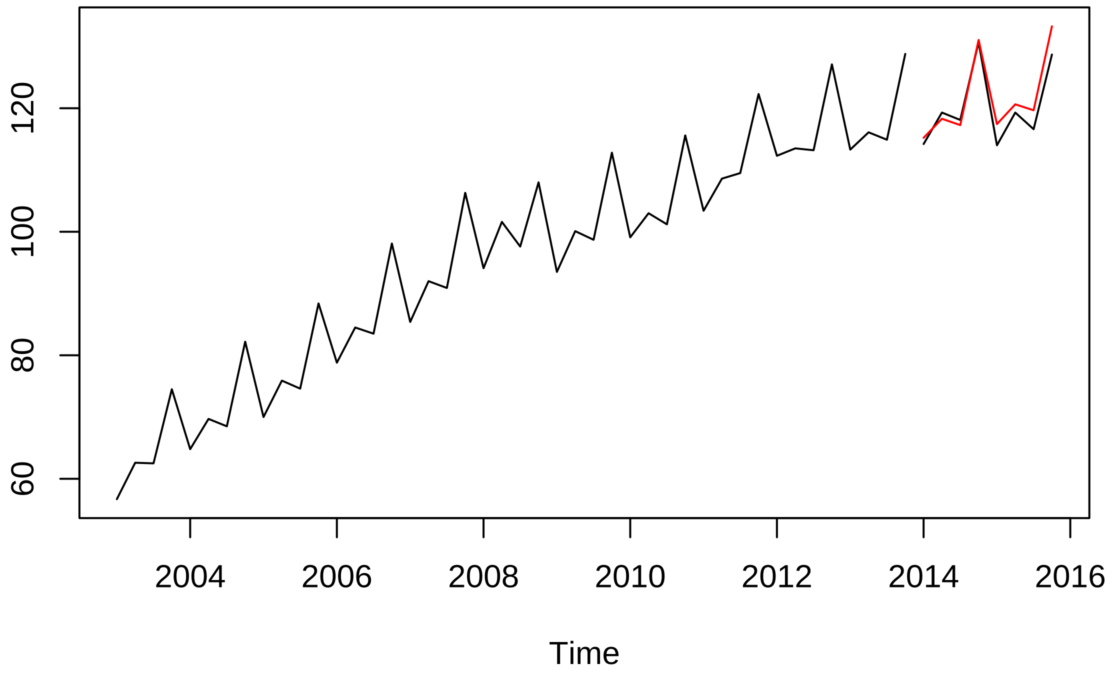
```r
# In-sample fit:
plot(X$y,type="l")
frc <- predict(fit2,X)
lines(frc,col="red")
```



```r
#Initialise an array to save the forecasts
frc1 <- array(NA,c(8,1))
for(i in 1:8){
 #For the X new we use the last five observations as before
 Xnew <- tail(y.trn,5)
 #Add to that the forecasted values
 Xnew <- c(Xnew,frc1)
 #Take the relevant 5 values. The index i helps us to get the right ones
 Xnew <- Xnew[i:(4+i)]
 #If i=1 then this becomes Xnew[1:5].
 #If i=2 then this becomes Xnew[2:6] - just as the example above.
 #Reverse the order
 Xnew <- Xnew[5:1]
 #Make X new an array and name the inputs
 Xnew <- array(Xnew, c(1,5))#c(1,5) are the dimensions of the array
 colnames(Xnew) <- paste0("lag",1:5) #I have already reversed the order
 #Convert to data.frame
 Xnew <- as.data.frame(Xnew)
 #Forecast
 frc1[i]<-predict(fit2,Xnew)
}
frc1
```

```
##          [,1]
## [1,] 115.2038
## [2,] 118.2922
## [3,] 117.2685
## [4,] 131.0602
## [5,] 117.4294
## [6,] 120.6364
## [7,] 119.6665
## [8,] 133.2663
```

```
frc1 <- ts(frc1,frequency=frequency(y.tst),start=start(y.tst))
ts.plot(y.trn,y.tst,frc1,col=c("black","black","red"))
```



Lasso Regression

```
if (!require("glmnet")){install.packages("glmnet")}; library(glmnet)
```

```
## Loading required package: glmnet
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-10
```

```
# I remove the first 5 rows by-(1:5) that contain NAs
# For the explanatories I remove the first column
xx <- as.matrix(X[-(1:5),-1])
# For the target I retain only the first column
```

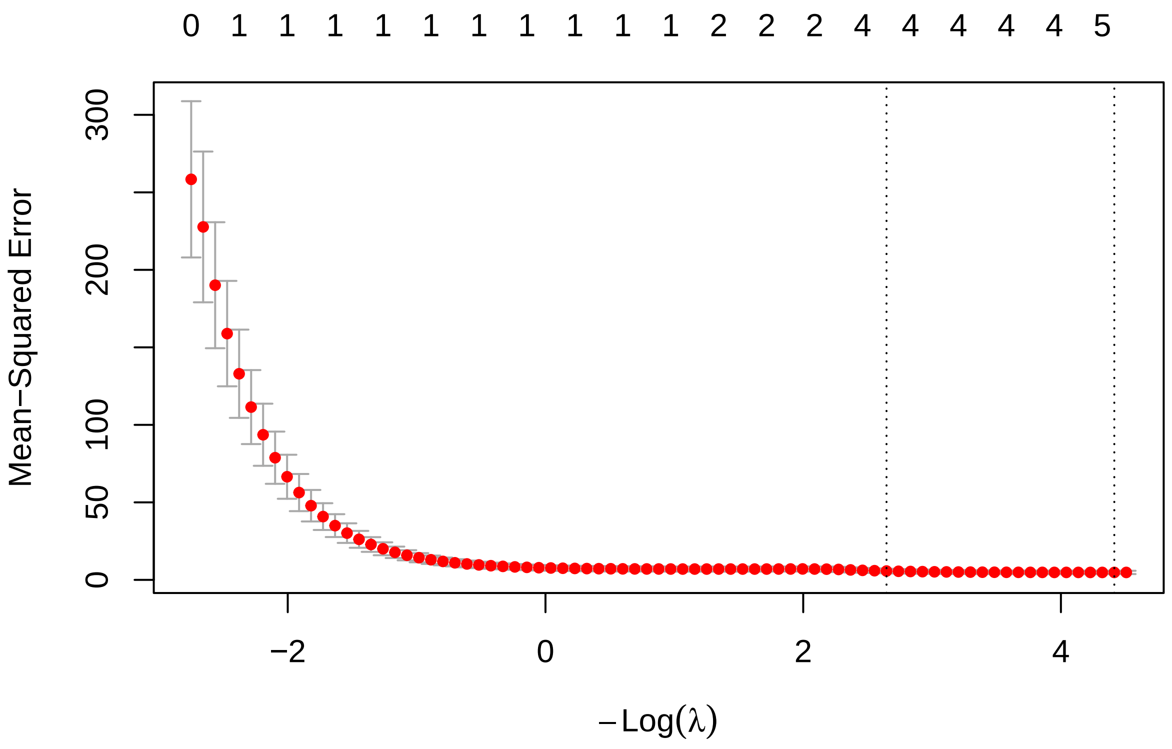
```r
yy <- as.matrix(X[-(1:5),1])
```

```r
lasso <- cv.glmnet(x=xx,y=yy)
```

```r
coef(lasso)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##              lambda.1se
## (Intercept) 10.15473072
## lag1         0.28616491
## lag2         .
## lag3        -0.01074002
## lag4         0.93888175
## lag5        -0.28230139
```

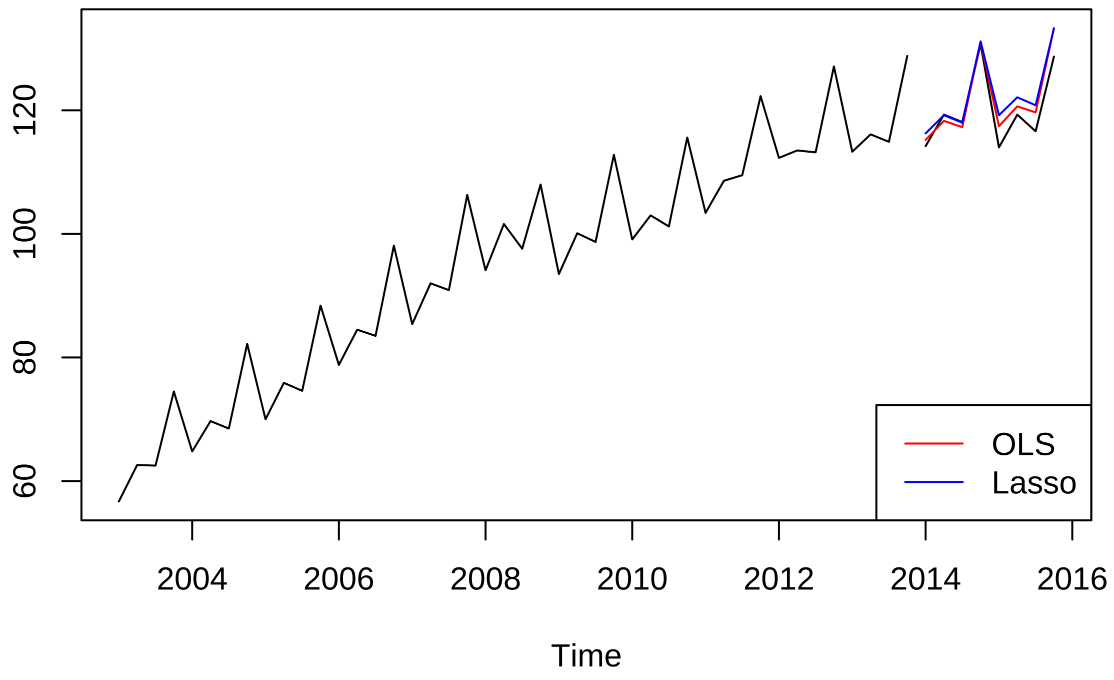
```r
 plot(lasso)
```



```r
frc2<-array(NA,c(8,1))
for(i in 1:8){
#Create inputs - note for lasso we do not transform these in to data.frame
 Xnew<-c(tail(y.trn,5),frc2)
 Xnew<-(Xnew[i:(4+i)])[5:1]
 Xnew<-array(Xnew,c(1,5))
 colnames(Xnew)<-paste0("lag",1:5)
 #Forecast
 frc2[i]<-predict(lasso,Xnew)
}
```

```r
#Transform to time series
frc2 <- ts(frc2,frequency=frequency(y.tst),start=start(y.tst))
#Plot together with fit2
ts.plot(y.trn,y.tst,frc1,frc2,col=c("black","black","red","blue"))
```

```
legend("bottomright",c("OLS","Lasso"),col=c("red","blue"),lty=1)
```



```
ridge<-cv.glmnet(x=xx,y=yy,alpha=0)
coef(ridge)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##               lambda.1se
## (Intercept) 13.69617584
## lag1         0.08707777
## lag2         0.18955063
## lag3         0.03488126
## lag4         0.57627115
## lag5         0.01392702
```

```
cc <- as.matrix(cbind(coef(lasso),coef(ridge)))
colnames(cc) <- c("lasso","ridge")
round(cc,3)
```

```
##               lasso  ridge
## (Intercept) 10.155 13.696
## lag1         0.286  0.087
## lag2         0.000  0.190
## lag3        -0.011  0.035
## lag4         0.939  0.576
## lag5        -0.282  0.014
```

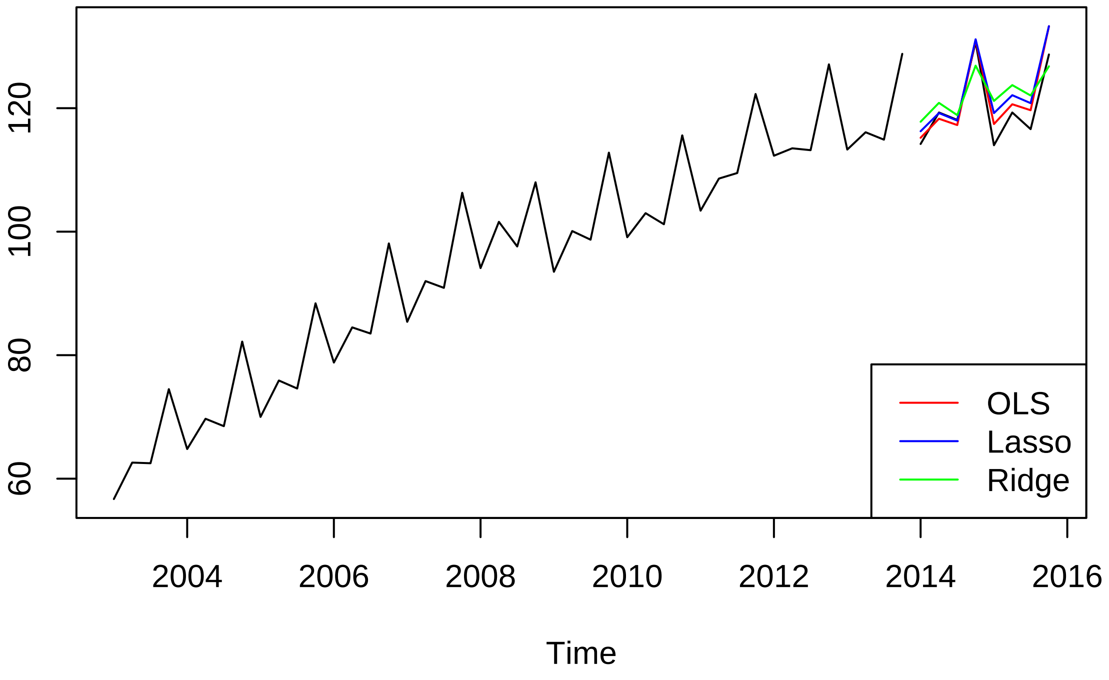
Exercises

Evaluate the performance of the ols, lasso, and ridge forecasts. Which performs best here? How do they compare with an exponential smoothing benchmark?

```
#Prediction ridge regression
frc3<-array(NA,c(8,1))
```

```r
for(i in 1:8){
#Create inputs
 Xnew<-c(tail(y.trn,5),frc3)
 Xnew<-(Xnew[i:(4+i)])[5:1]
 Xnew<-array(Xnew,c(1,5))
 colnames(Xnew)<-paste0("lag",1:5)
 #Forecast
 Xnew <- as.data.frame(Xnew)
 frc3[i]<-predict(ridge,Xnew)
}
```

```r
#Transform to time series
frc3 <- ts(frc3,frequency=frequency(y.tst),start=start(y.tst))
#Plot together with fit2
ts.plot(y.trn,y.tst,frc1,frc2,frc3, col=c("black","black","red","blue", "green"))
legend("bottomright",c("OLS","Lasso", "Ridge"),col=c("red","blue", "green"),lty=1)
```



Based on the line plot we can see that the OLS model seems to perform best since it is represented by the red line which is closest to the actual data shown by the black line. Lasso and Ridge regression do not seem to work as well since the green line (ridge) and the blue line (lasso) are further away from the actual data.

```r
 actual <- matrix(rep(y.tst,3),ncol=3)
 actual
```

```
##       [,1]  [,2]  [,3]
## [1,] 114.2 114.2 114.2
## [2,] 119.3 119.3 119.3
## [3,] 118.1 118.1 118.1
## [4,] 130.7 130.7 130.7
## [5,] 114.0 114.0 114.0
## [6,] 119.3 119.3 119.3
## [7,] 116.6 116.6 116.6
## [8,] 128.7 128.7 128.7
```

```
error <- abs(actual- cbind(frc1,frc2,frc3))
MAE <- colMeans(error)
MAE
```

```
##     frc1     frc2     frc3
## 1.950239 2.440542 3.590244
```

The results with the MAE align with the line plot from before. The OLS model performed best with a mean absolute error of 1.95. The Lasso regression model (MAE = 2.56) performed better than the ridge regression model (MAE = 3.73). This indicates that variable selection has a slight advantage but the difference is not great.

```
#exponential smoothing benchmark
library(forecast)
```

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

```
fit4 <- ets(y.trn, model="AAA")
frc4 <- forecast(fit4, h=8)
preds <- as.numeric(frc4$mean)
preds <- data.frame(preds)
preds
```

```
##      preds
## 1 116.5815
## 2 121.5932
## 3 120.4262
## 4 133.9972
## 5 121.8057
## 6 126.8175
## 7 125.6505
## 8 139.2215
```

```
actual <- matrix(rep(y.tst,4),ncol=4)
actual
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,] 114.2 114.2 114.2 114.2
## [2,] 119.3 119.3 119.3 119.3
## [3,] 118.1 118.1 118.1 118.1
## [4,] 130.7 130.7 130.7 130.7
## [5,] 114.0 114.0 114.0 114.0
## [6,] 119.3 119.3 119.3 119.3
## [7,] 116.6 116.6 116.6 116.6
## [8,] 128.7 128.7 128.7 128.7
```

```
error <- abs(actual- cbind(frc1, frc2, frc3, preds))
MAE <- colMeans(error)
MAE
```

```
##     frc1     frc2     frc3    preds
## 1.950239 2.440542 3.590244 5.649165
```

Compared to the exponential smoothing benchmark, the regression models provided better forecasts. This suggests that the predictors added useful information. Overall OLS provided the most accurate forecast in this case. The regression-based models outperformed the exponential smoothing indicating that including predictors improved the forecast accuracy.