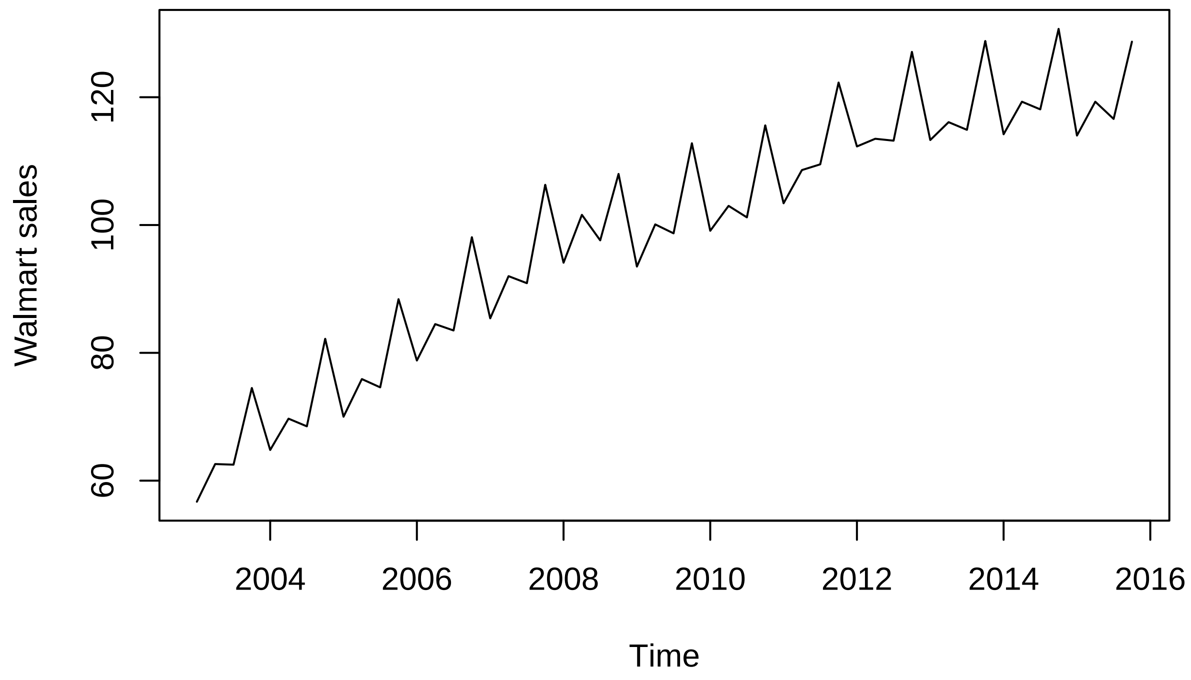
Lasso regression modelling

Rizny Mubarak

Contents

1. Load Data

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



2. Build a benchmark regression

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

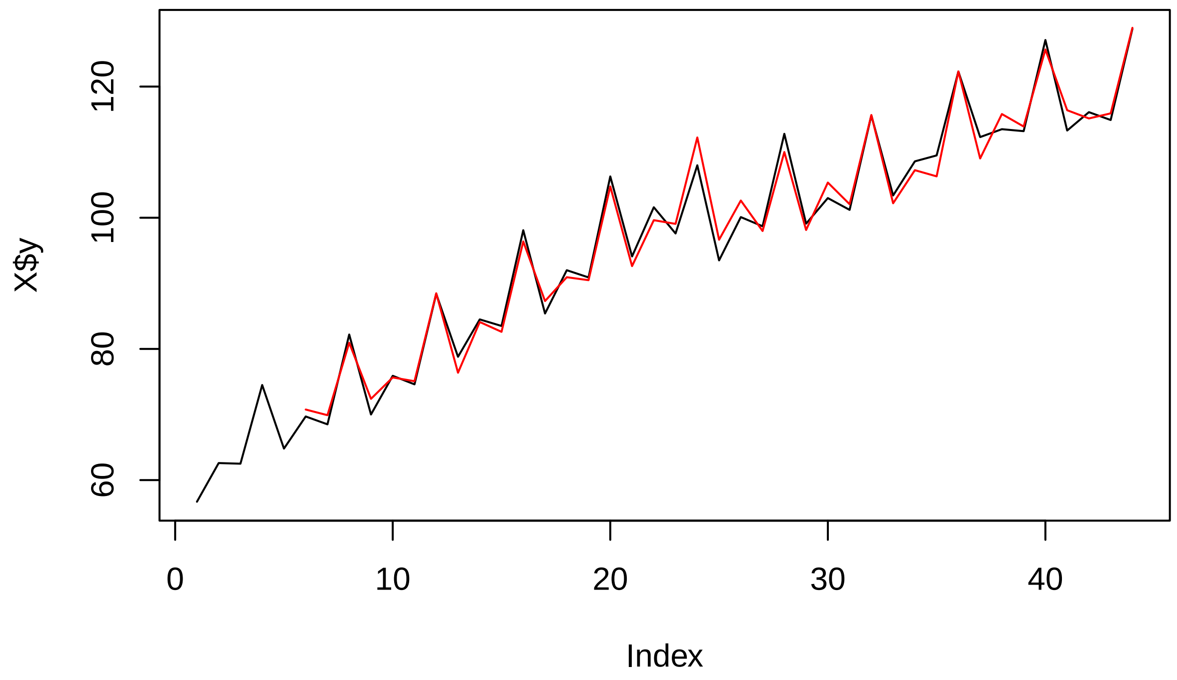
```
n <- length(y.trn)
X <- array(NA,c(n,6))
# Loop to create lags
for (i in 1:6){
X[i:n,i] <- y.trn[1:(n-i+1)]
}
#Name the columns
colnames(X) <- c("y",paste0("lag",1:5))
X <- as.data.frame(X)
head(X)
```

```
##      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
```

```r
# The complete model
fit1 <- lm(y~.,data=X)
# The stepwise model
fit2 <- step(fit1,trace=0)
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
```

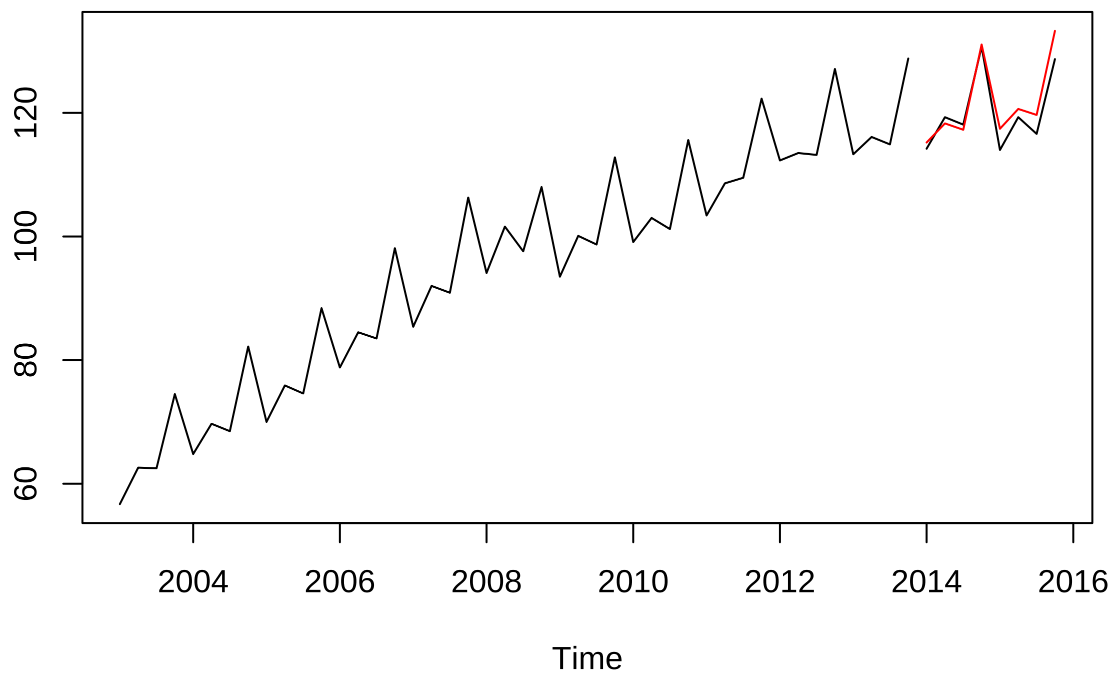
```r
# In-sample fit:
plot(X$y,type="l")
frc <- predict(fit2,X)
lines(frc,col="red")
```



```r
# Initialise an array to save the forecasts
frc1 <- array(NA,c(8,1))
for (i in 1:8){
# For the Xnew 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 Xnew 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"))
```



3. Lasso regression

```
library(glmnet)
```

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

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

```
## Loaded glmnet 4.1-8
```

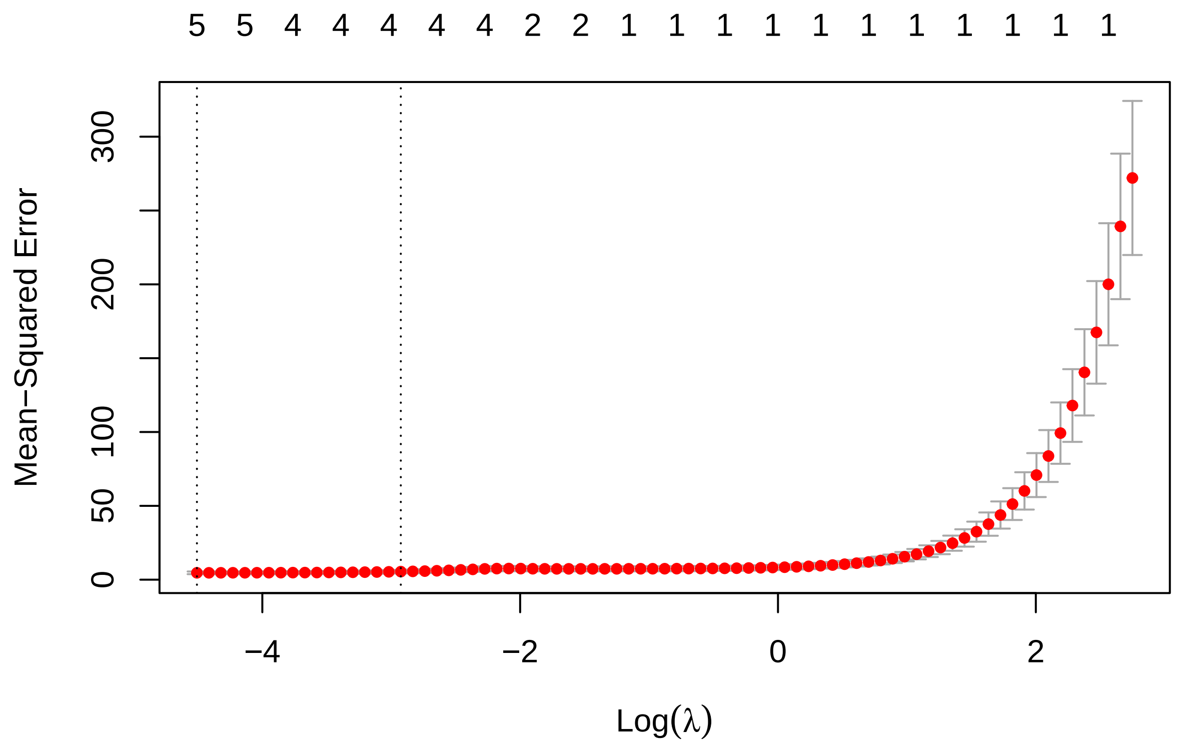
```r
xx <- as.matrix(X[-(1:5),-1])
# For the target I retain only the first column
yy <- as.matrix(X[-(1:5),1])
```

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

```r
coef(lasso)
```

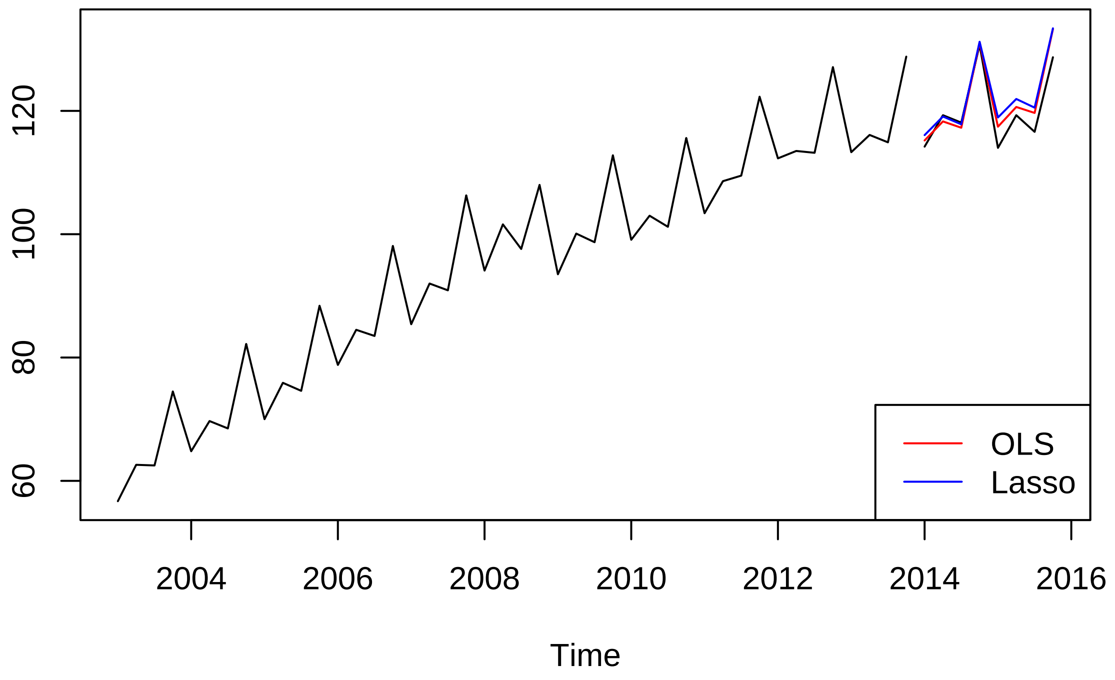
```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  8.90385409
## lag1         0.37999900
## lag2         .
## lag3        -0.01596498
## lag4         0.95096183
## lag5        -0.37505802
```

```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 into 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)
}
```

```
# 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"
##                      s1
## (Intercept) 14.12539157
## lag1         0.08585042
## lag2         0.19785460
## lag3         0.04386667
## lag4         0.54218640
## lag5         0.02730617
```

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

```
##              lasso  ridge
## (Intercept)  8.904 14.125
## lag1         0.380  0.086
## lag2         0.000  0.198
## lag3        -0.016  0.044
## lag4         0.951  0.542
## lag5        -0.375  0.027
```

4. Exercises on lasso regression

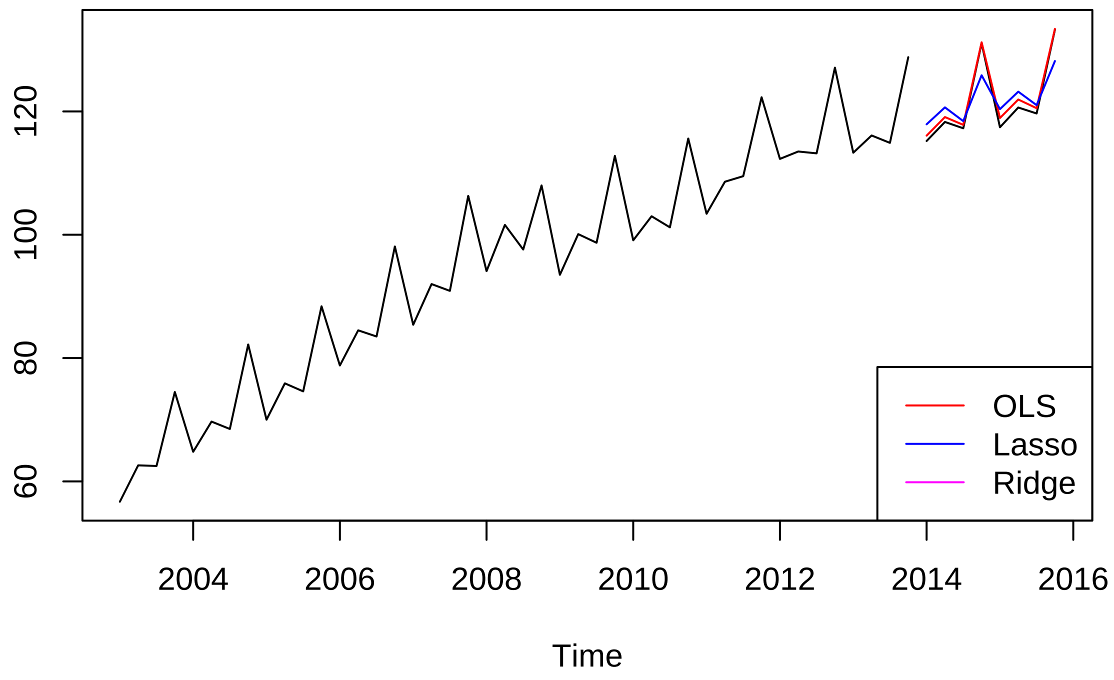
4.1 Ridge regression

- Forecast

```r
frc3 <- array(NA,c(8,1))
for (i in 1:8){
# Create inputs - note for lasso we do not transform these into 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
frc3[i] <- predict(ridge,Xnew)
}
```

- Plot Ridge with other forecasts

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



```r
library(forecast)
```

4.2 Exponential smoothing

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

- Fit the ETS Model

```r
fit4 <- ets(y.trn)
summary(fit4)
```

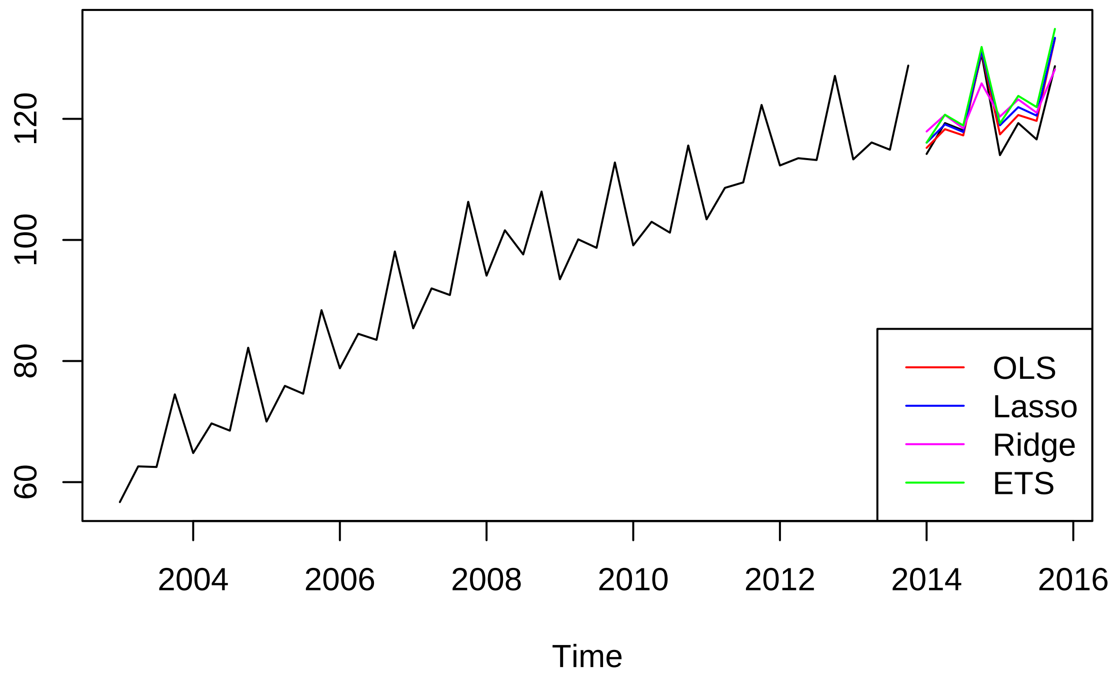
```
## ETS(M,Ad,A)
##
## Call:
##  ets(y = y.trn)
##
##   Smoothing parameters:
##     alpha = 0.8208
##     beta  = 0.0028
##     gamma = 1e-04
##     phi   = 0.98
##
##   Initial states:
##     l = 59.9851
##     b = 2.0685
##     s = 8.8003 -3.3914 -0.8089 -4.6
##
##   sigma:  0.0163
##
##      AIC     AICc      BIC
## 212.8437 219.5104 230.6856
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 0.004101084 1.421315 1.085659 -0.01030883 1.133684 0.1964994
##                     ACF1
## Training set 0.02581449
```

- Forecast using ETS

```r
frc4 <- forecast(fit4, h=8)
frc4 <- frc4$mean
```

- Plot ETS wit others

```r
# Plot together with fit2
ts.plot(y.trn,y.tst,frc1,frc2,frc3,frc4,col=c("black","black","red","blue","magenta","green"))
legend("bottomright",c("OLS","Lasso","Ridge","ETS"),col=c("red","blue","magenta","green"),lty=1)
```



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

4.3 Calculate Error

```
##       [,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
```

```r
error <- abs(actual - cbind(frc1,frc2,frc3,frc4))
colnames(error) <- c("OLS", "Lasson", "Ridge", "ETS")
MAE <- colMeans(error)
MAE
```

```
##      OLS   Lasson    Ridge      ETS
## 1.950239 2.381372 3.180951 3.309276
```

5. Answer

Among the models tested, the OLS model with autoregressive components emerged as the most predictive, exhibiting the lowest MAE and superior visual performance. In contrast, the Lasso model showed promise in feature selection and predictive precision but was less efficient than OLS. The ETS benchmark, renowned for accurate predictions, ranked third in performance, serving as a reliable reference despite not surpassing OLS and Lasso. However, despite its regularization advantages, the Ridge model demonstrated the lowest projected accuracy and poorest overall performance. In conclusion, this scientific study underscores the practical and analytical advantages of utilizing the OLS model with autoregressive components for precise sales value predictions, showcasing its effectiveness in capturing trends and correlations within sales data.