Elastic Net Test

LIBRARIES

```
# libraries
library(MASS)  # Package needed to generate correlated precictors
library(glmnet)  # Package to fit ridge/lasso/elastic net models
library(ggplot2)
source("elastic.net.R")
```

GENERATE DATA

```
# seed
set.seed(19875)  # Set seed for reproducibility

# globals
n <- 1000  # Number of observations
p <- 5000  # Number of predictors included in model
real_p <- 15  # Number of true predictors
x <- matrix(rnorm(n*p), nrow=n, ncol=p) # this are fake cytokines
y <- apply(x[,1:real_p], 1, sum) + rnorm(n) # fake gene expression
```

SPLIT DATA INTO TEST AND TRAIN

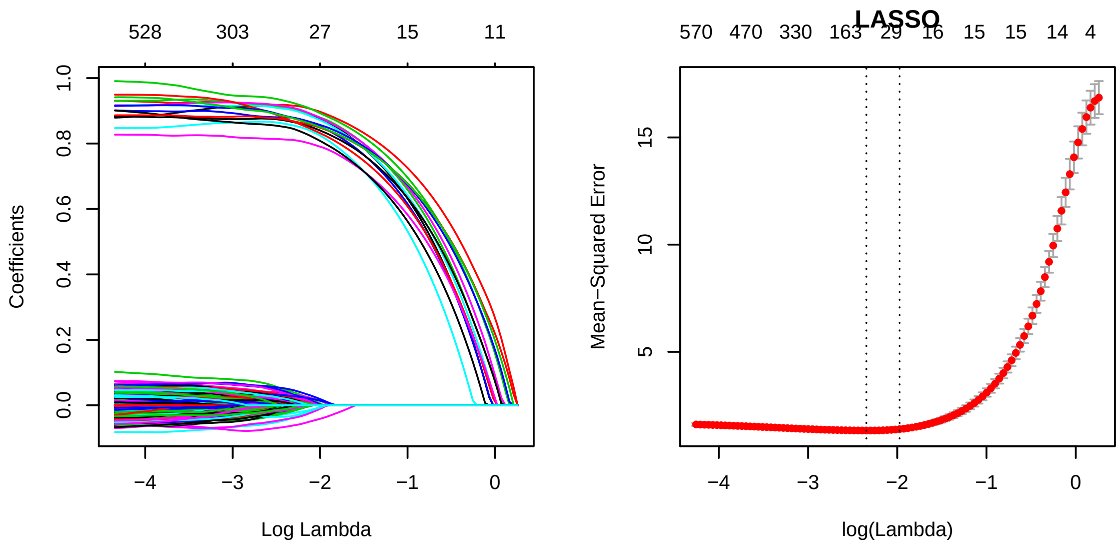
```
# split data
train_rows <- sample(1:n, .66*n) # random data set
x.train <- x[train_rows, ] # train (2/3)
x.test <- x[-train_rows, ] # test (1/3)

y.train <- y[train_rows]
y.test <- y[-train_rows]
```

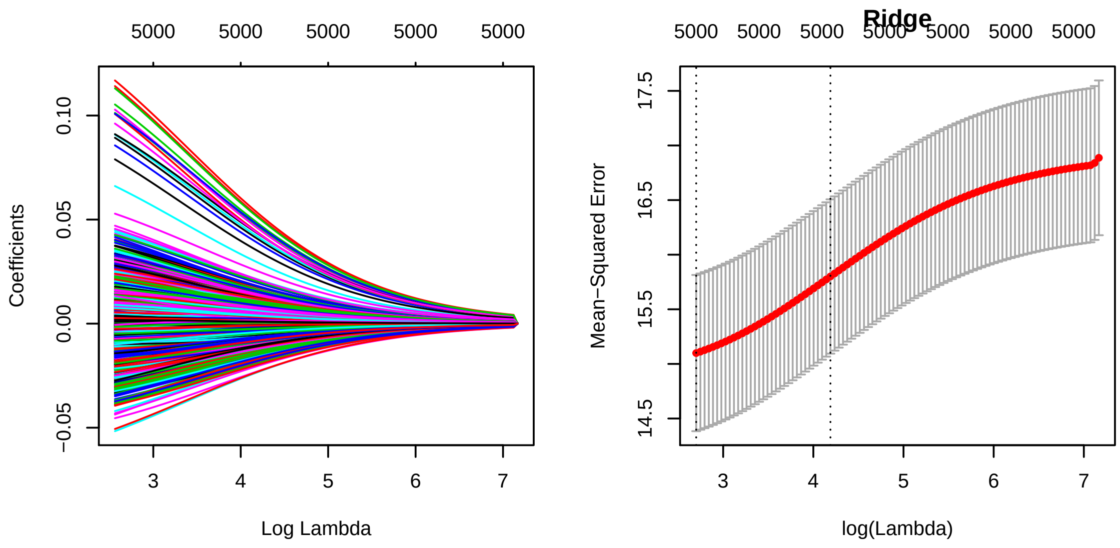
FIT MODELS

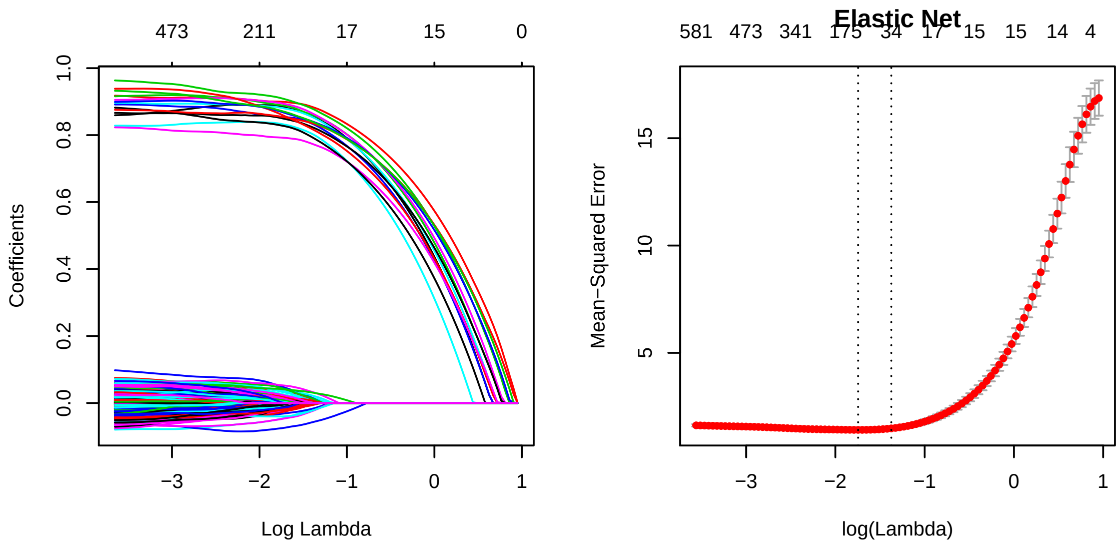
```
# Fit models
fit.lasso <- glmnet(x.train, y.train, family="gaussian", alpha=1)
fit.ridge <- glmnet(x.train, y.train, family="gaussian", alpha=0)
fit.elnet <- glmnet(x.train, y.train, family="gaussian", alpha=.5)


# 10-fold Cross validation for each alpha = 0, 0.1, ... , 0.9, 1.0
# (For plots on Right)
for (i in 0:10) {
  assign(paste("fit", i, sep=""), cv.glmnet(x.train, y.train, type.measure="mse",
                                             alpha=i/10,family="gaussian"))

}
```

PLOT SOLUTION PATH AND CROSS-VALIDATED MSE AS FUNCTION OF λ

```r
# Plot solution paths:
par(mfrow=c(3,2))
# For plotting options, type '?plot.glmnet' in R console
plot(fit.lasso, xvar="lambda")
plot(fit10, main="LASSO")

plot(fit.ridge, xvar="lambda")
plot(fit0, main="Ridge")

plot(fit.elnet, xvar="lambda")
plot(fit5, main="Elastic Net")
```

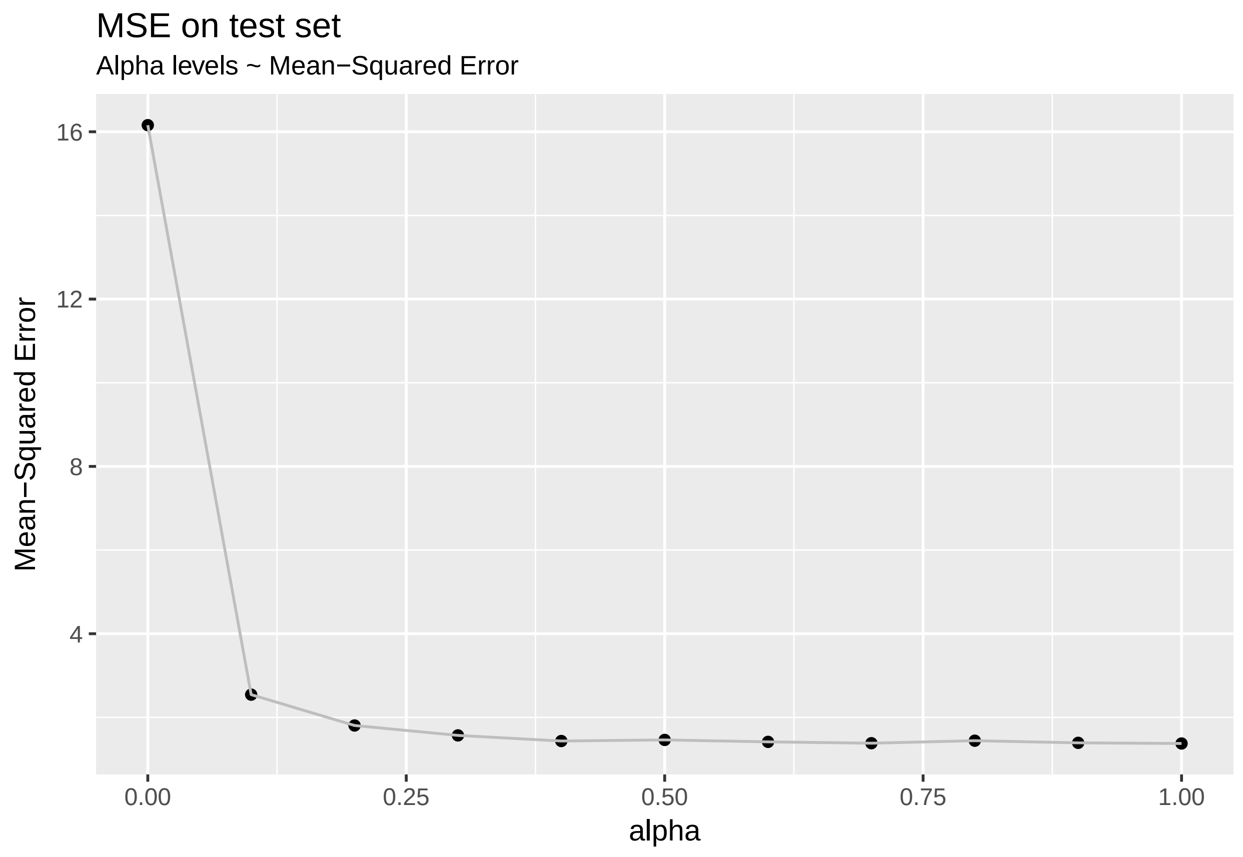

LASSO
528 303 27 15 11
Coefficients
Log Lambda
570 470 330 163 29 16 15 15 14 4
Mean−Squared Error
log(Lambda)


Ridge
5000 5000 5000 5000 5000
Coefficients
Log Lambda
5000 5000 5000 5000 5000 5000 5000
Mean−Squared Error
log(Lambda)


Elastic Net
473 211 17 15 0
Coefficients
Log Lambda
581 473 341 175 34 17 15 15 14 4
Mean−Squared Error
log(Lambda)

MSE ON TEST SET

```r
# Predict fit on test data
for (i in 0:10) {
  assign(paste("yhat", i, sep=""),
         predict(get(paste0("fit", i)),
                 s=get(paste0("fit", i))$lambda.1se,
                 newx=x.test))

}

# calculate Mean-Squared error of model
a <- seq(0, 1, .1)
mse <- c()
for (i in 0:10) {
  m <- mean((y.test - get(paste0("yhat", i)))^2)
  mse <- c(mse, m)
}

ggplot(data = NULL, aes(a, mse) ) +
  geom_point() +
  geom_line(col="grey") +
  labs(subtitle="Alpha levels ~ Mean-Squared Error",
       x="alpha",
       y="Mean-Squared Error",
       title="MSE on test set")
```



FOLDS FUNCTION

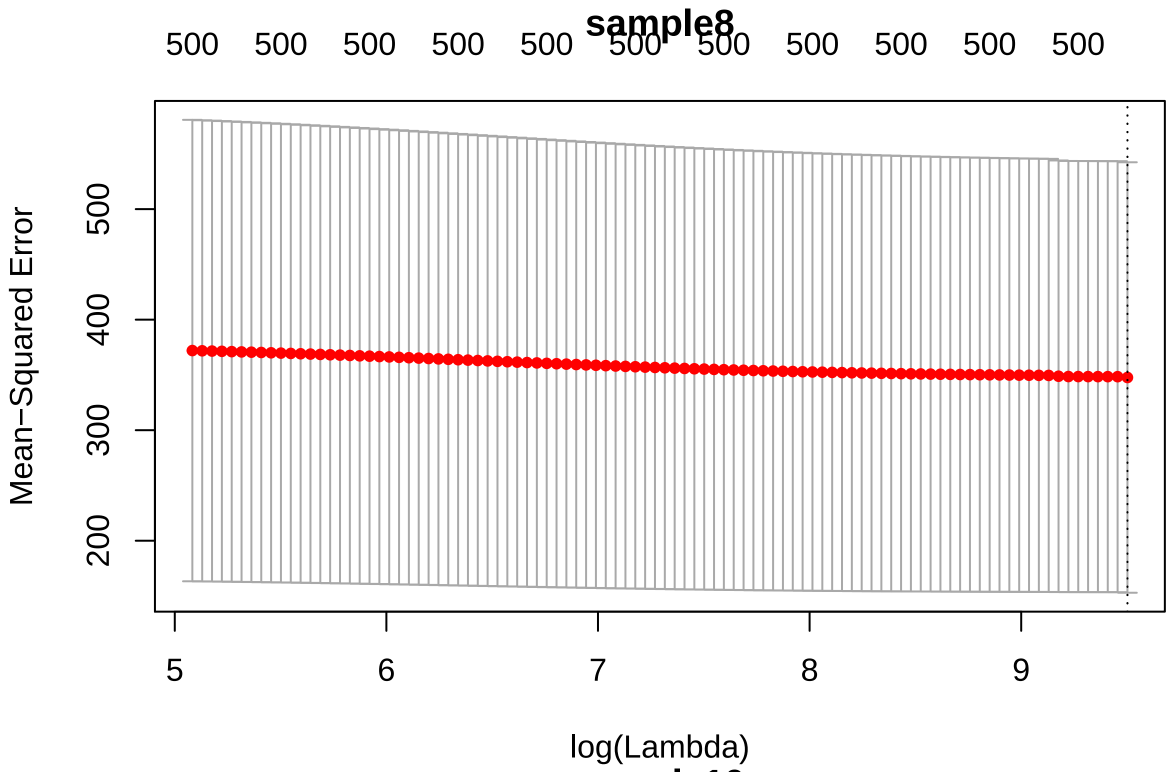
```r
Folds <- function(y, k=10) {
  n <- length(y)
  if (n == 0)
    stop('response length is zero')

  uniqY <- unique(y)
  if (!is.factor(y) && length(y) / length(uniqY) >= k) {
# Intepret the integer-valued y as class labels. Stratify if the number of class labels is <= 5.
    y <- factor(y)
  } else if (is.numeric(y)) {
# 5-stratum Stratified sampling
    if (n >= 5 * k) {
      breaks <- unique(quantile(y, probs=seq(0, 1, length.out=5)))
      y <- as.integer(cut(y, breaks, include.lowest=TRUE))
    } else
      y <- rep(1, length(y))
  }

  sampList <- tapply(seq_along(y), y, sFolds, k=k, simplify=FALSE)
  list0 <- list()
  length(list0) <- k
  samp <- Reduce(function(list1, list2) {
                 mapply(c, list1, list2, SIMPLIFY=FALSE)
  }, sampList, list0)

  return(samp)
}

sFolds <- function(yy, k=10) {
  if (length(yy) > 1)
    allSamp <- sample(yy)
  else
    allSamp <- yy

  n <- length(yy)
  nEach <- n %/% k
  samp <- list()
  length(samp) <- k
  for (i in seq_along(samp)) {
    if (nEach > 0)
      samp[[i]] <- allSamp[1:nEach + (i - 1) * nEach]
    else
      samp[[i]] <- numeric(0)
  }
  restSamp <- allSamp[seq(nEach * k + 1, length(allSamp), length.out=length(allSamp) - nEach * k)]
  restInd <- sample(k, length(restSamp))
  for (i in seq_along(restInd)) {
    sampInd <- restInd[i]
    samp[[sampInd]] <- c(samp[[sampInd]], restSamp[i])
  }
  return(samp)
}
```

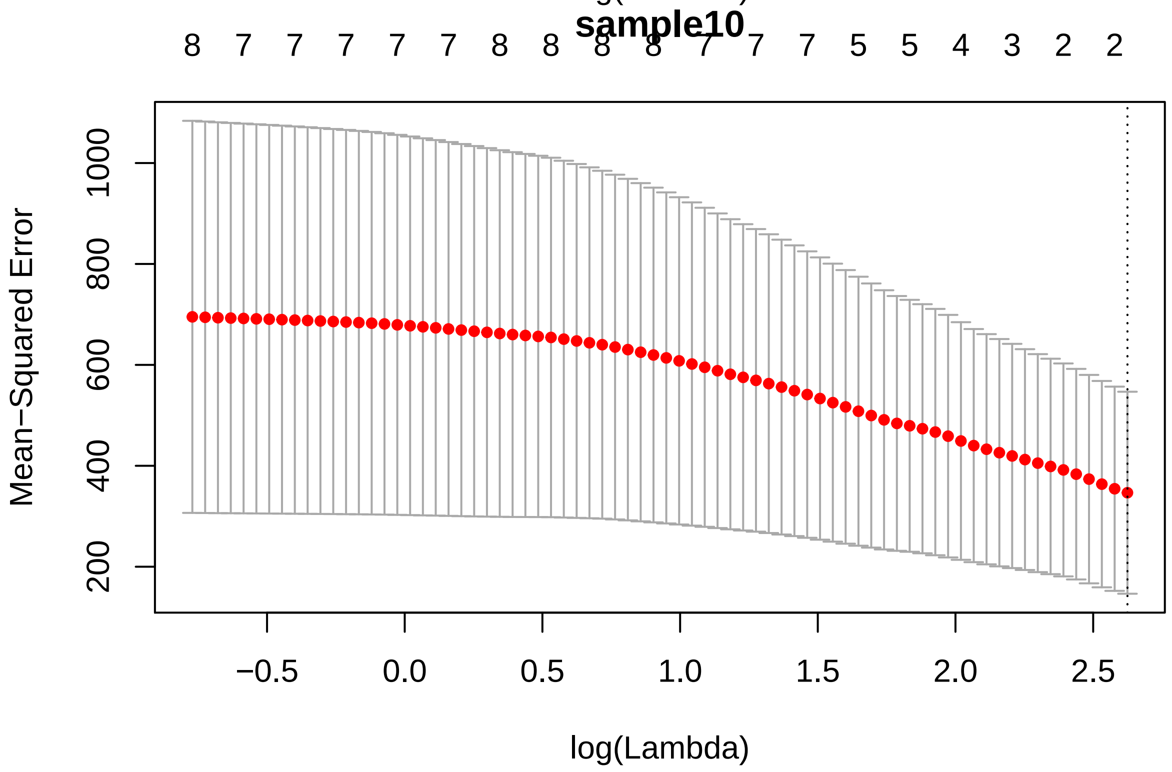
GLMNET.WRAPPER

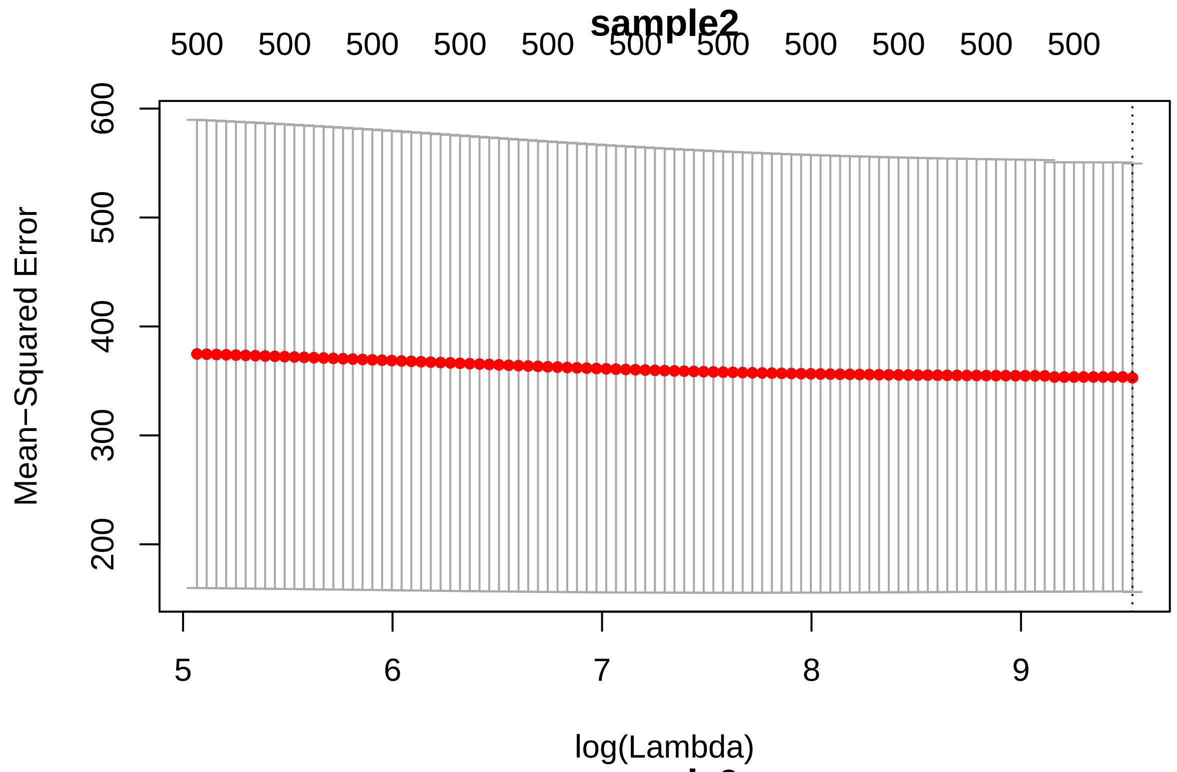
```r
# libraries
library(MASS)  # Package needed to generate correlated precictors
library(glmnet)  # Package to fit ridge/lasso/elastic net models
library(ggplot2)
source("elastic.net.R")
# Generate data
set.seed(19874)
n <- 10    # Number of observations
p <- 500     # Number of predictors included in model
real_p <- 150  # Number of true predictors
x <- matrix(rnorm(n*p), nrow=n, ncol=p)
y <- apply(x[,1:real_p], 1, sum) + rnorm(n)
nrow(x)
```

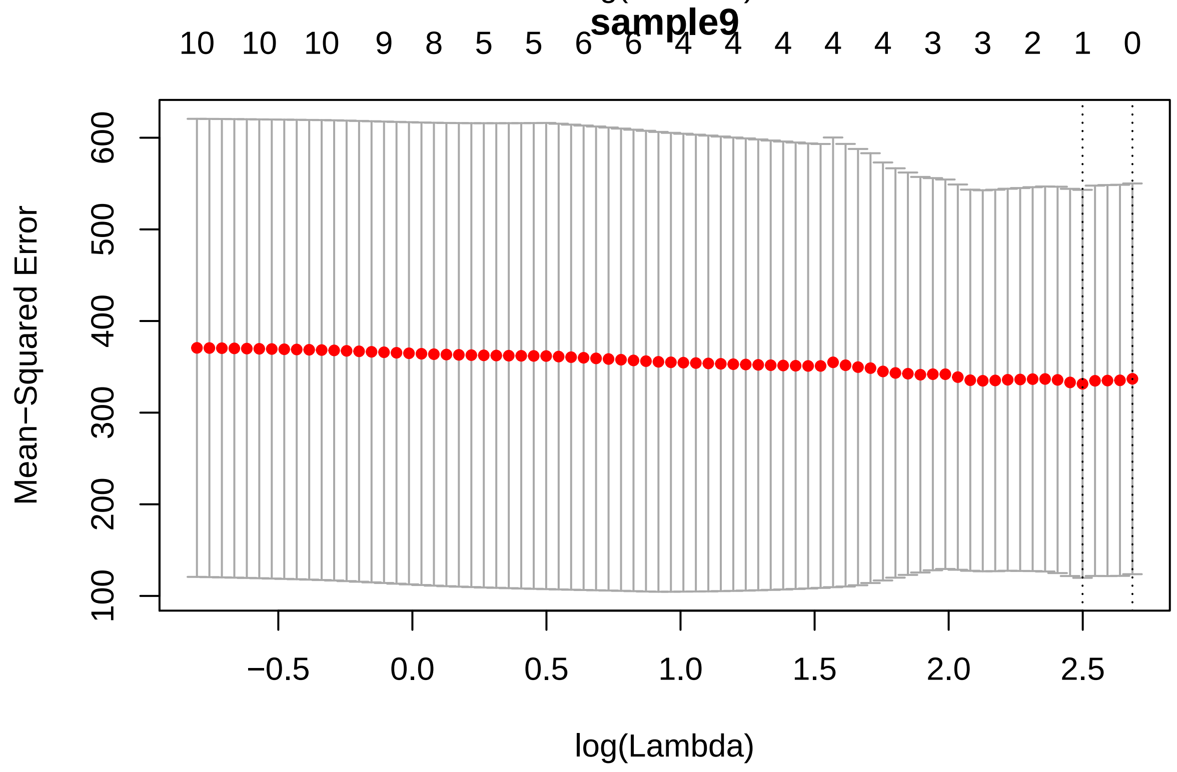
```
## [1] 10
```

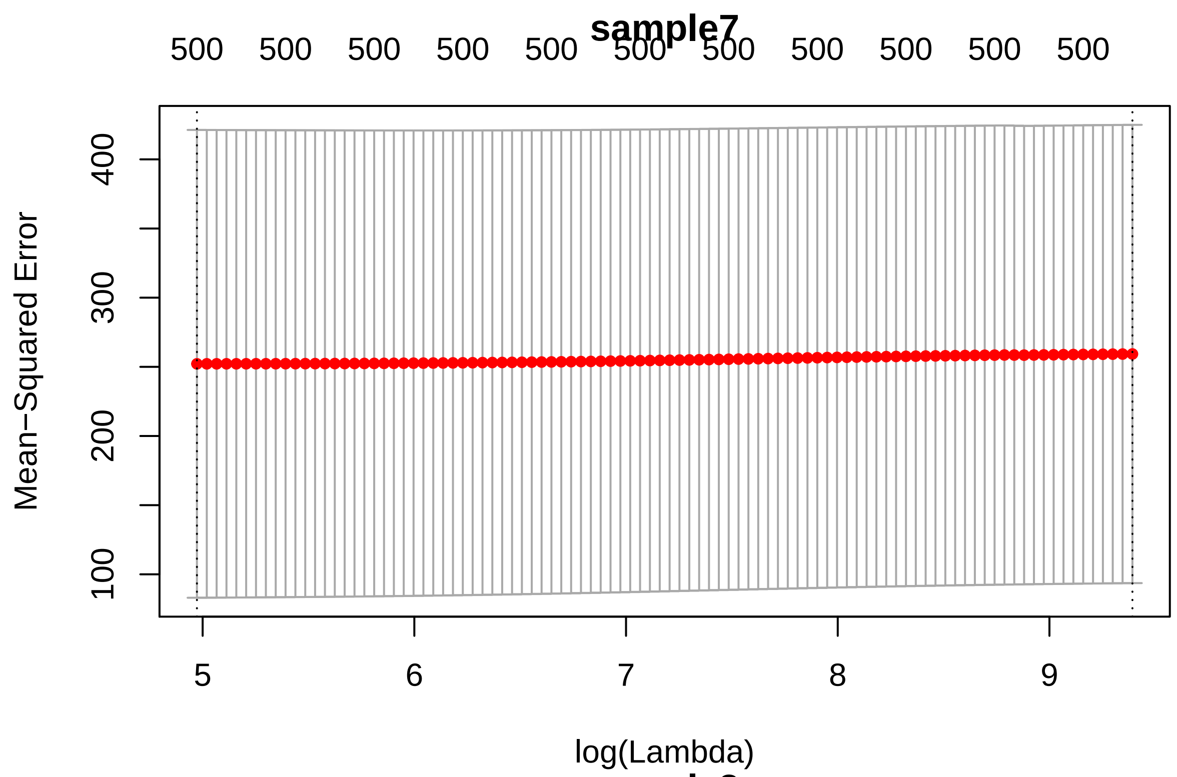
```r
# add colnames
rownames(x) <- sprintf("sample%s", seq(1:nrow(x)))
# vector names
y <- setNames(y, sprintf("sample%s", seq(1:length(y))))
# --------------------------------------------

# Remove NA values
y   <- y[!is.na(y)]
x   <- na.omit(x)
ol  <- intersect(rownames(x), names(y))
y   <- y[ol]
x   <- x[ol, , drop=F]

# create a list of length k, containing the test-set indices
folds <- Folds(y, k=10)

# glmt.net
results <- lapply(folds, function(fold, y, x, parallel=F) {
    return(glmnet.wrapper(y=y[-fold],
                          x=x[-fold, , drop=F],
                          newx=x[fold,  , drop=F],
                          newy=y[fold]))
  }, y=y, x=x)


# plot
for (i in 1:length(folds)) {
  plot(results[[i]]$best.model, main=names(results[[i]]$pred))

}
```

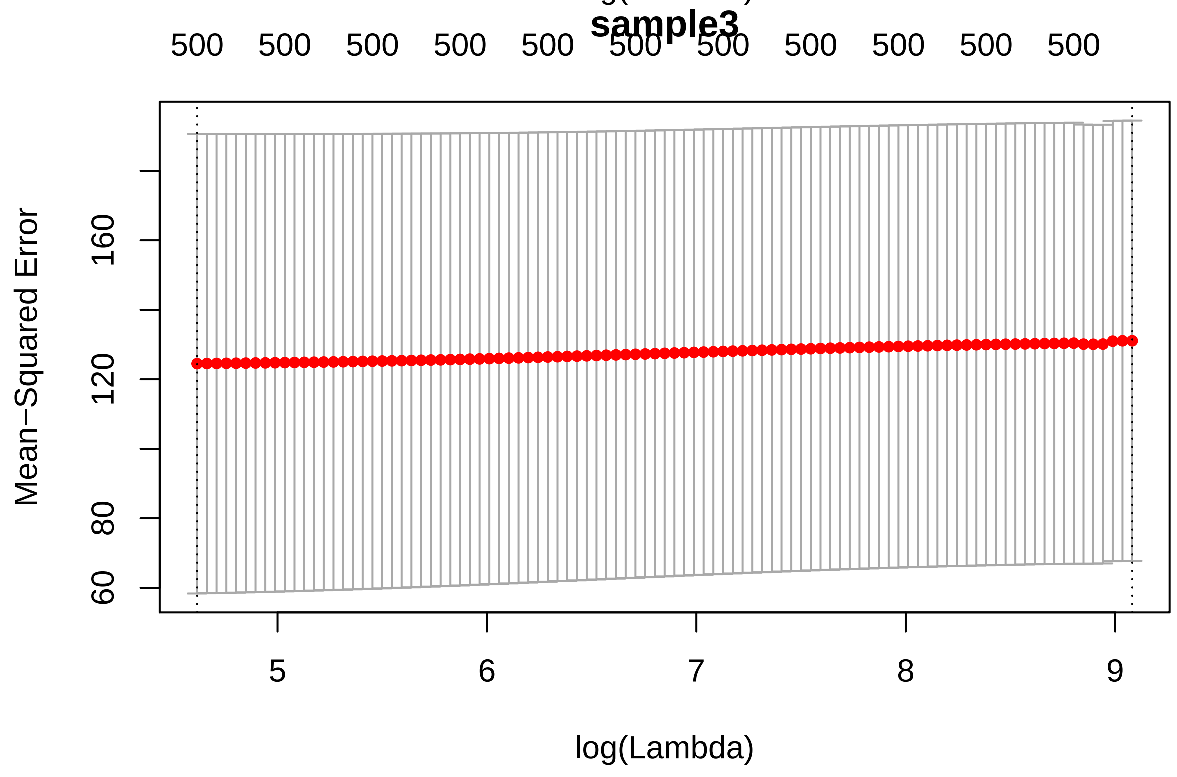

sample8
500 500 500 500 500 500 500 500 500 500 500
Mean−Squared Error
200 300 400 500
5 6 7 8 9
log(Lambda)

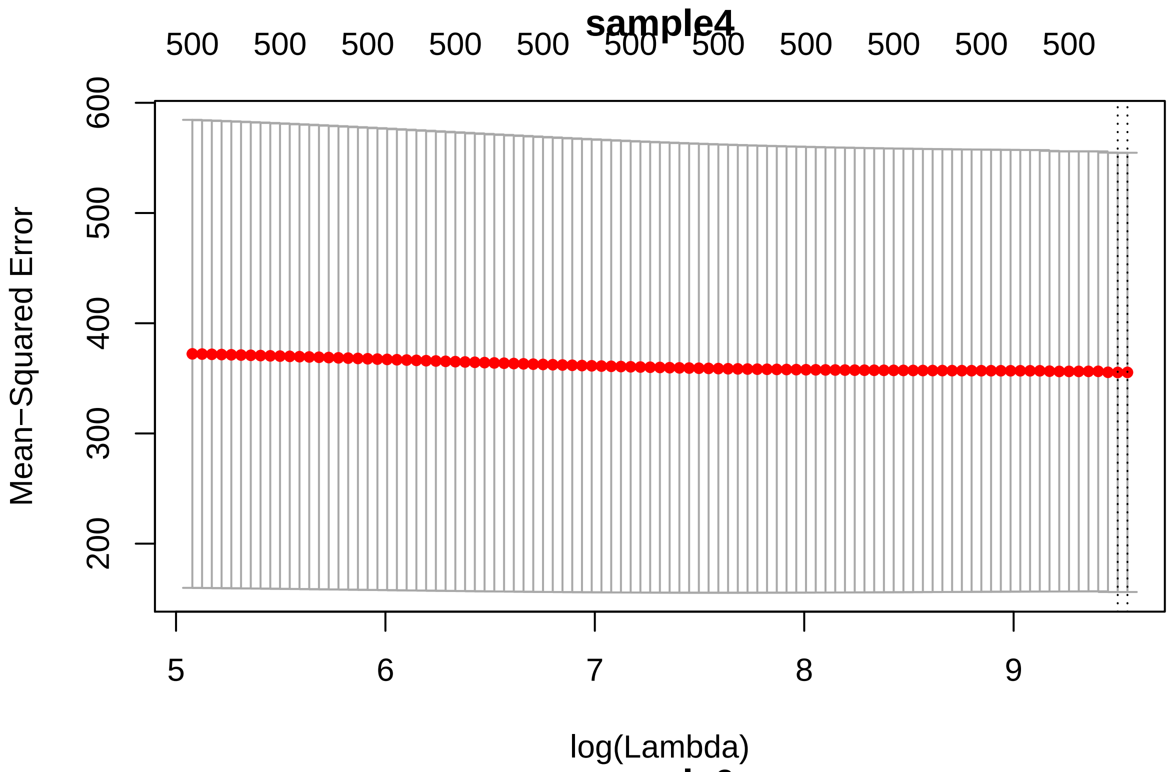

sample10
8 7 7 7 7 7 8 8 8 8 7 7 7 5 5 4 3 2 2
Mean−Squared Error
200 400 600 800 1000
−0.5 0.0 0.5 1.0 1.5 2.0 2.5
log(Lambda)

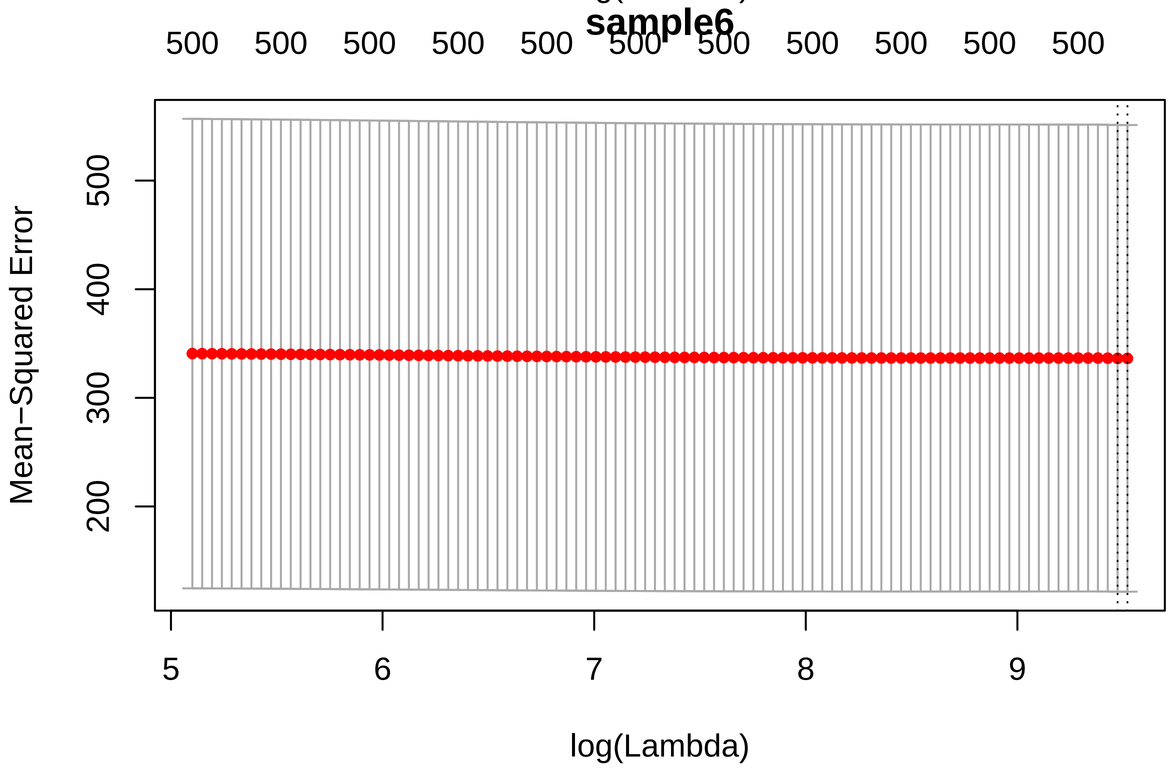

sample2
500 500 500 500 500 500 500 500 500 500 500
Mean−Squared Error
200 300 400 500 600
5 6 7 8 9
log(Lambda)

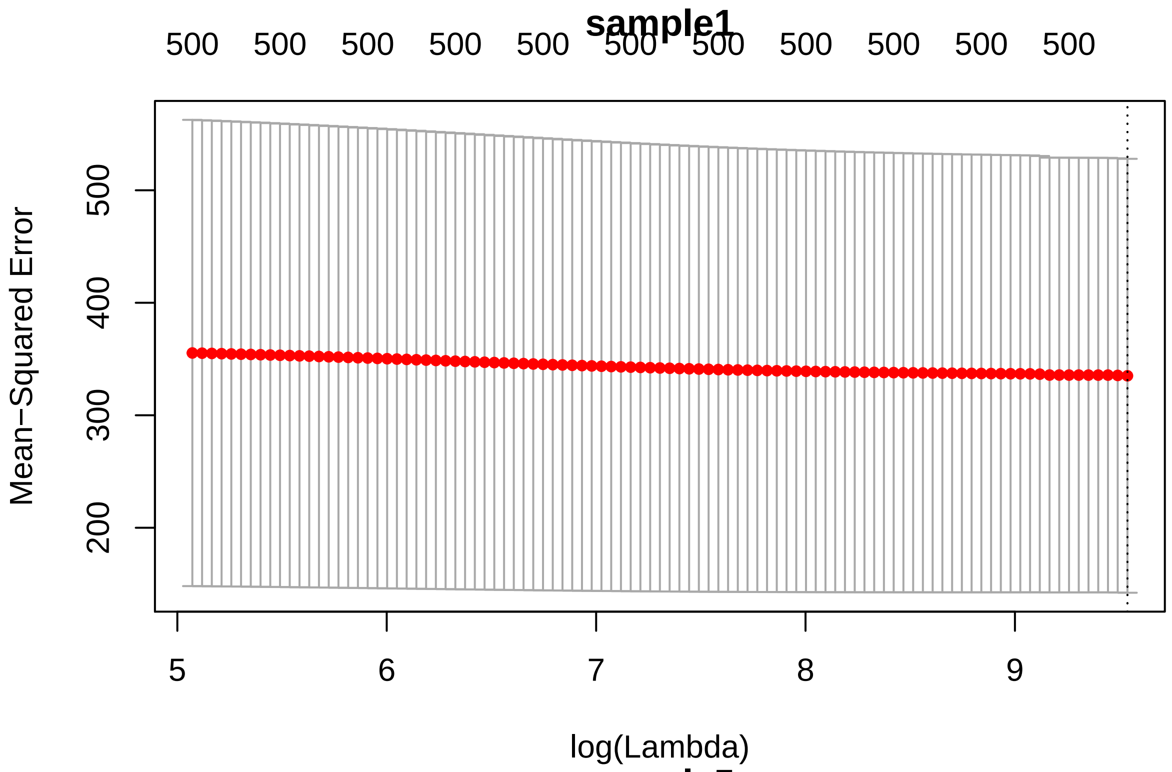

sample9
10 10 10 9 8 5 5 6 6 4 4 4 4 4 3 3 2 1 0
Mean−Squared Error
100 200 300 400 500 600
−0.5 0.0 0.5 1.0 1.5 2.0 2.5
log(Lambda)

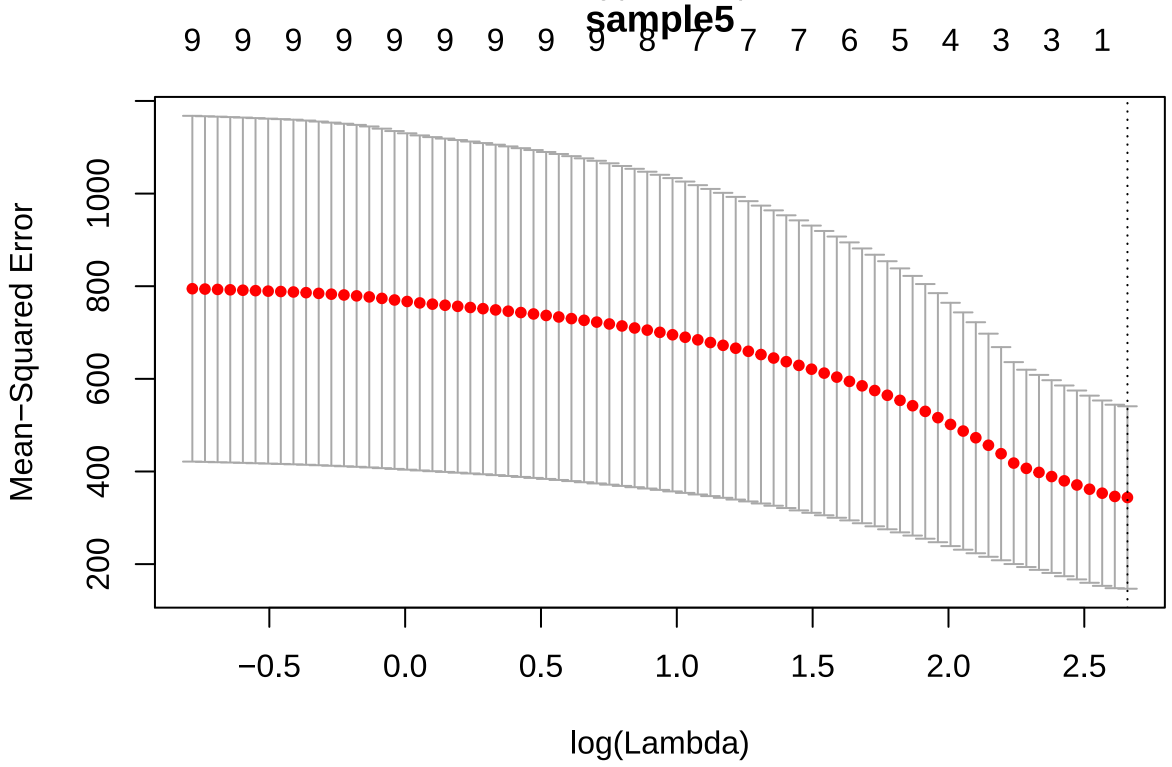

sample7
500 500 500 500 500 500 500 500 500 500 500
Mean−Squared Error
100
200
300
400
5
6
7
8
9
log(Lambda)


sample3
500 500 500 500 500 500 500 500 500 500 500
Mean−Squared Error
60
80
120
160
5
6
7
8
9
log(Lambda)


sample4
500 500 500 500 500 500 500 500 500 500 500
Mean−Squared Error
200
300
400
500
600
5
6
7
8
9
log(Lambda)


sample6
500 500 500 500 500 500 500 500 500 500 500
Mean−Squared Error
200
300
400
500
5
6
7
8
9
log(Lambda)





```
results[[1]]$mse
```

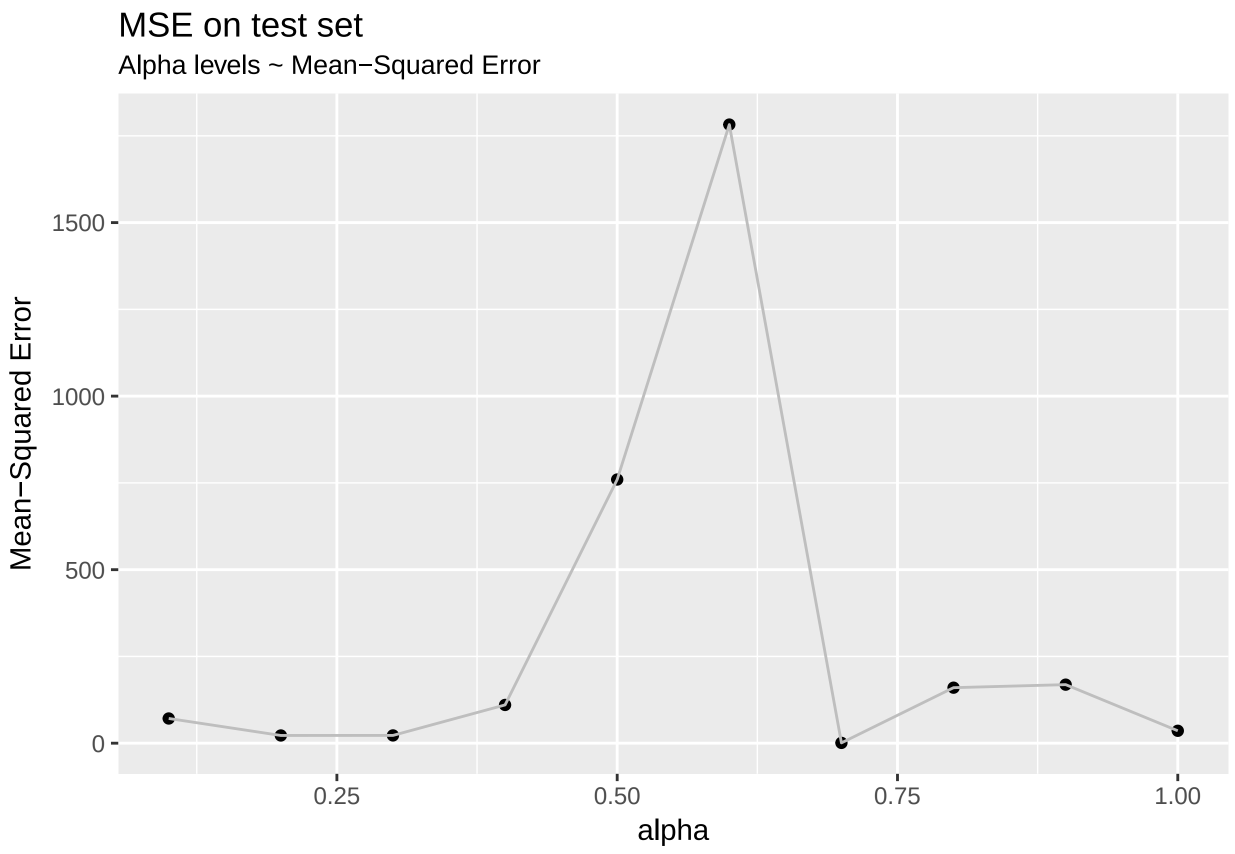
```
## [1] 71.17961
```

```
# calculate Mean-Squared error of model
a <- seq(0.1, 1, .1)
```

```
mse <- c()
for (i in 1:10) {
  m <- results[[i]]$mse
  mse <- c(mse, m)
}
length(a)
```

```
## [1] 10
```

```
ggplot(data = NULL, aes(a, mse) ) +
  geom_point() +
  geom_line(col="grey") +
  labs(subtitle="Alpha levels ~ Mean-Squared Error",
       x="alpha",
       y="Mean-Squared Error",
       title="MSE on test set")
```


MSE on test set
Alpha levels ~ Mean−Squared Error
Mean−Squared Error
alpha