Packages required for Regularized regression

Code

Hide

```
set.seed(123)    # seef for reproducibility
library(glmnet)  # for ridge regression
library(dplyr)   # for data cleaning
library(psych)   # for function tr() to compute trace of a matrix
library(caret)
```

Import Dataset from package 'MASS'

Hide

```
data("Boston", package = "MASS")
head(Boston)
```

	crim <dbl>	zn <dbl>	indus <dbl>	chas <int>	nox <dbl>	rm <dbl>	age <dbl>	dis <dbl>	rad <int>
1	0.00632	18	2.31	0	0.538	6.575	65.2	4.0900	1
2	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2
3	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2
4	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3
5	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3
6	0.02985	0	2.18	0	0.458	6.430	58.7	6.0622	3

6 rows | 1-10 of 14 columns

Preparing the data and We randomly split the data into training set (80% for building a predictive model) and test set (20% for evaluating the model).

Hide

```
set.seed(123)
training.samples <- Boston$medv %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- Boston[training.samples, ]
test.data <- Boston[-training.samples, ]
```

We need to create two objects X for holding predictor variables

Hide

```
X = model.matrix(medv ~ ., train.data)[, -1]
```

y for storing the outcome variable

Hide

```
y = train.data$medv
```

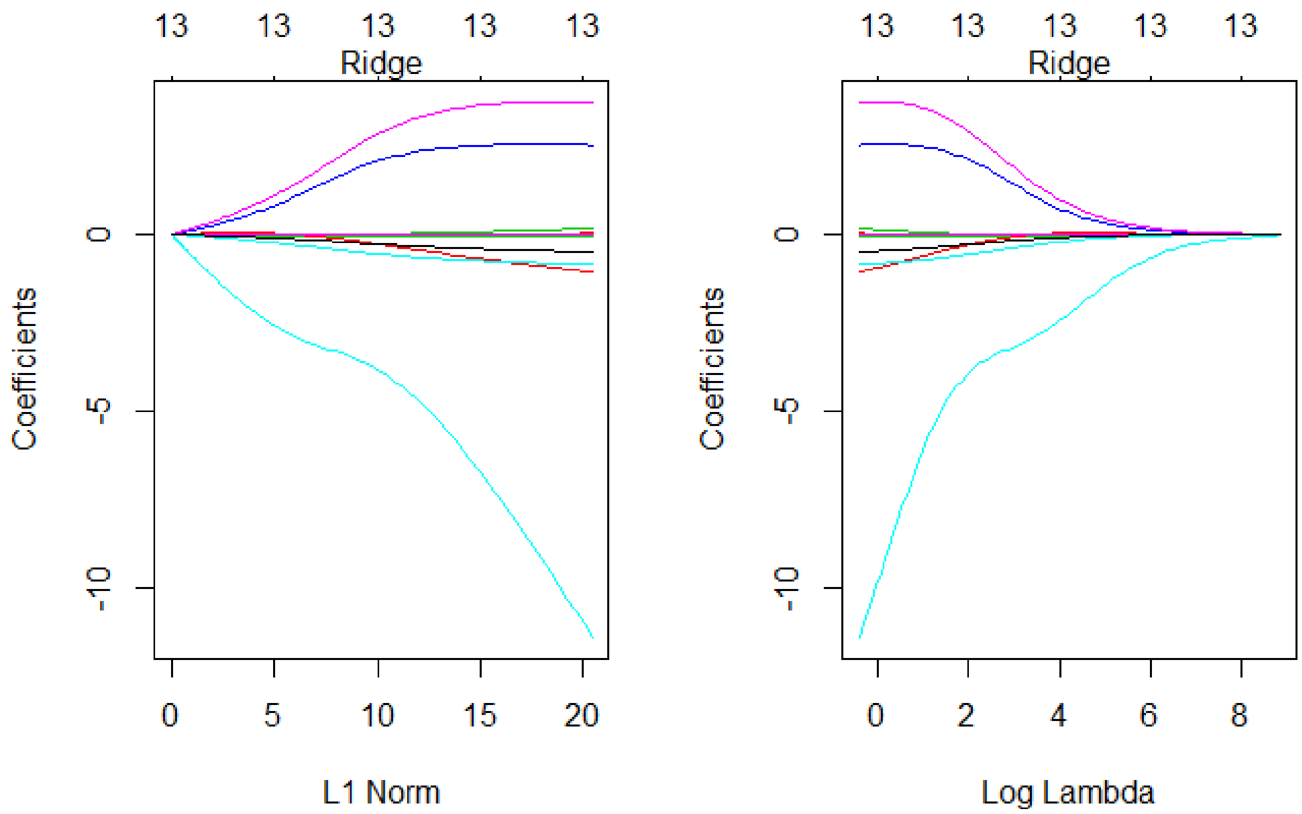
Computing Ridge regression model

Hide

```
par(mfrow = c(1, 2))
fit_ridge = glmnet(X, y, alpha = 0)
plot(fit_ridge)
mtext("Ridge")
```

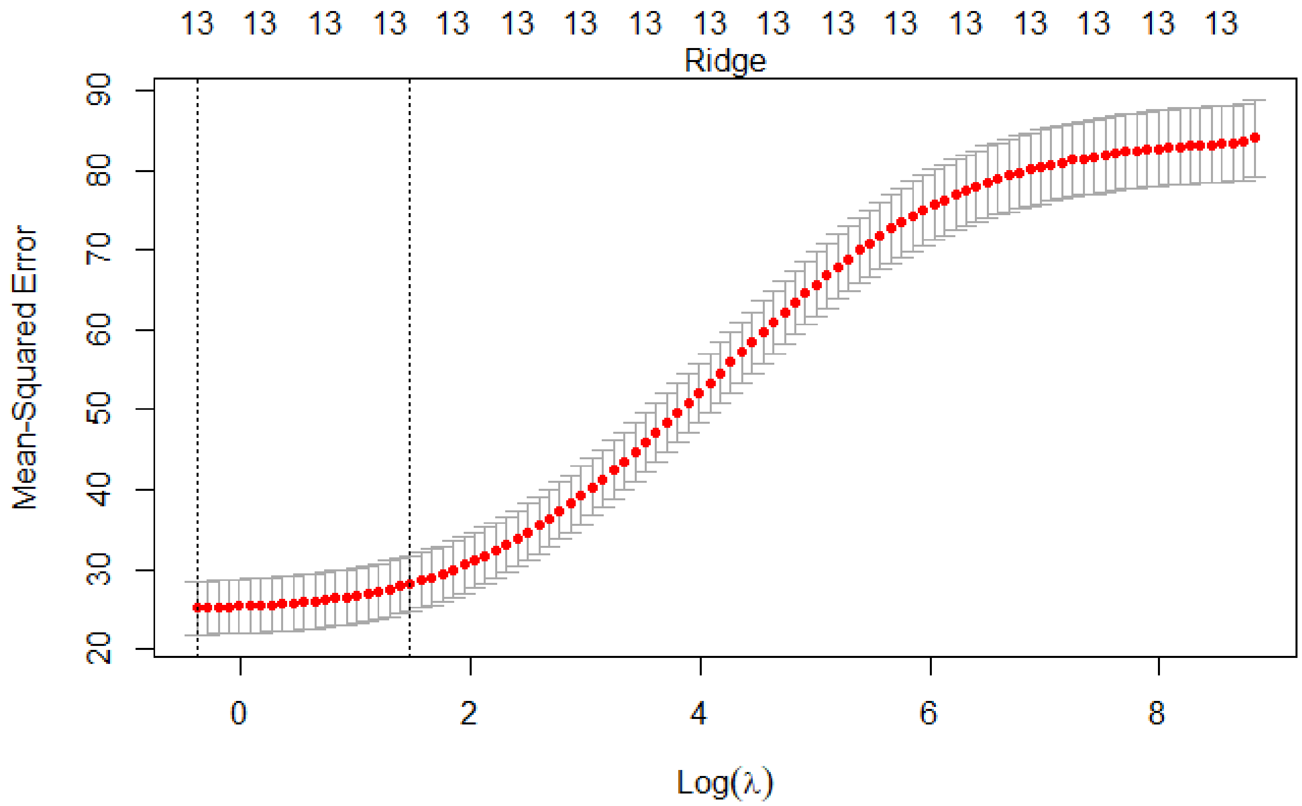
Hide

```
plot(fit_ridge, xvar = "lambda", label = TRUE)
mtext("Ridge")
```



Hide

```
cv_ridge = cv.glmnet(X, y, alpha = 0)
plot(cv_ridge)
mtext("Ridge")
```



Hide

```
cv_ridge$lambda.min
```

```
[1] 0.6803611
```

Hide

```
# Fit the final model on the training data
model_ridge <- train(
  medv ~., data = train.data, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneGrid = expand.grid(alpha = 0, lambda = cv_ridge$lambda.min)
  )
# Model coefficients
coef(model_ridge$finalModel, model_ridge$bestTune$lambda)
```

```
14 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  29.199522433
crim         -0.076456377
zn            0.026794700
indus        -0.062277495
chas          2.525893584
nox         -11.388819192
rm            3.723870172
age           0.002204446
dis          -1.058455239
rad           0.165876918
tax          -0.005354632
ptratio      -0.862205203
black         0.009610093
lstat        -0.498825572
```

Hide

```
# Make predictions on the test data
x.test <- model.matrix(medv ~., test.data)[,-1]
predictions <- model_ridge %>% predict(x.test) %>% as.vector()
# Model performance metrics
data.frame(
  RMSE = RMSE(predictions, test.data$medv),
  Rsquare = R2(predictions, test.data$medv)
)
```

RMSE <dbl>	Rsquare <dbl>
4.646655	0.7655889

1 row

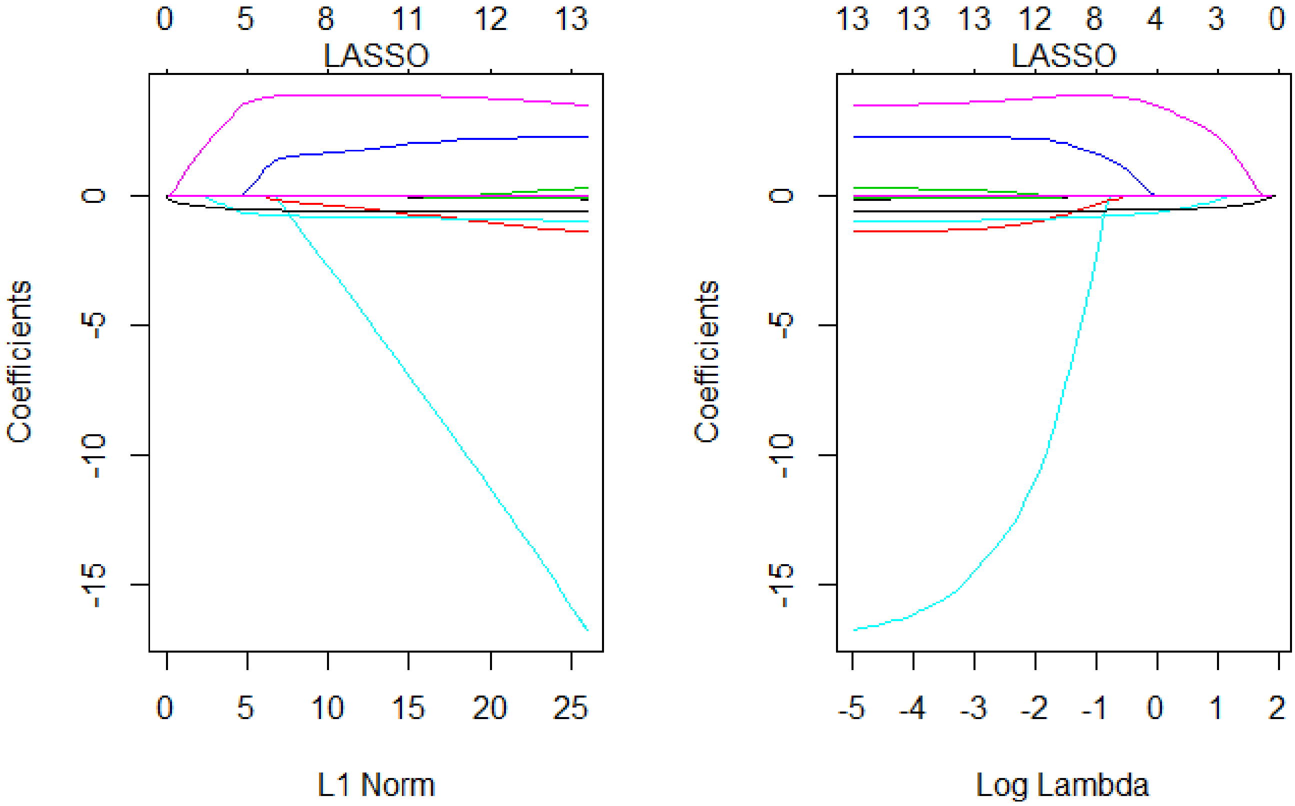
Computing Lasso regression model

Hide

```
set.seed(123)
par(mfrow = c(1, 2))
fit_lasso = glmnet(X, y, alpha = 1)
plot(fit_lasso)
mtext("LASSO")
```

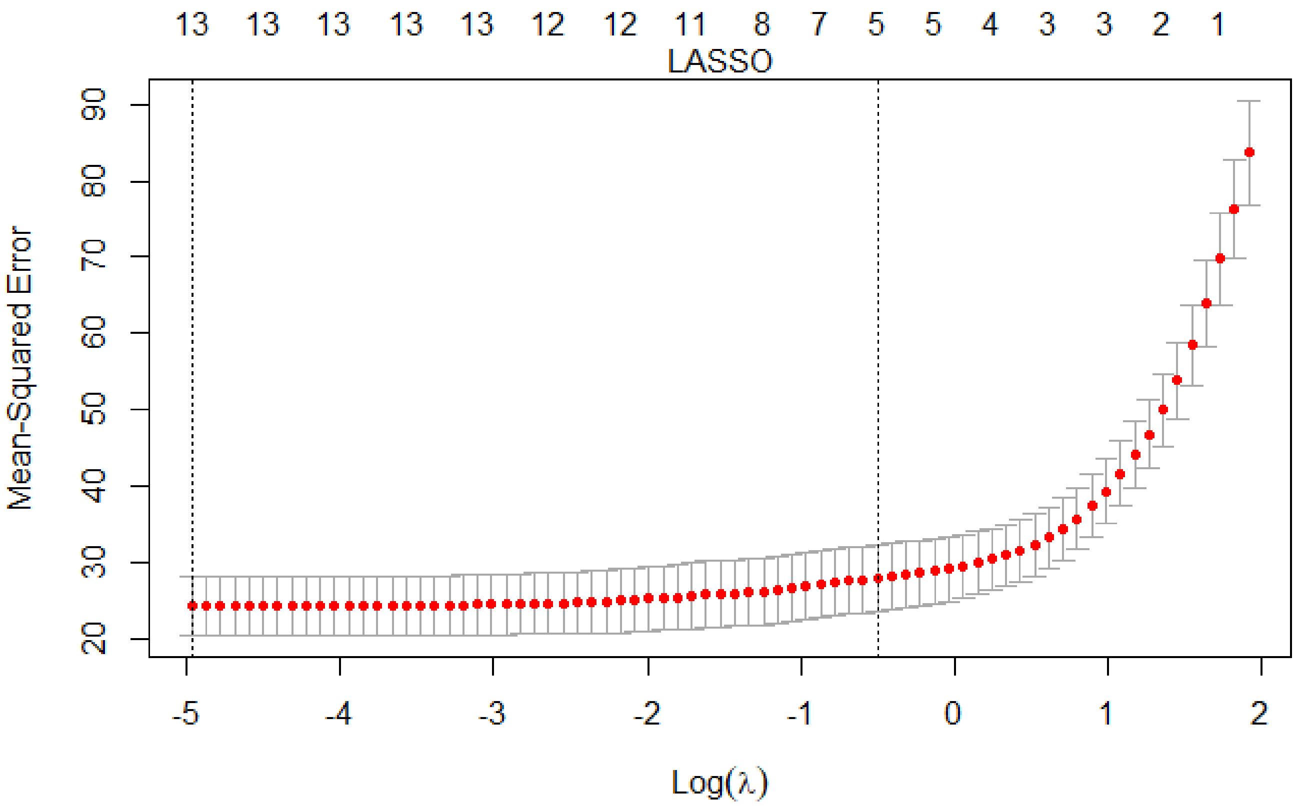
Hide

```
plot(fit_lasso, xvar = "lambda", label = TRUE)
mtext("LASSO")
```



Hide

```
cv_lasso = cv.glmnet(X, y, alpha = 1)
plot(cv_lasso)
mtext("LASSO")
```



Hide

```
cv_lasso$lambda.min
```

```
[1] 0.006963707
```

Hide

```
# Fit the final model on the training data
model_lasso <- train(
  medv ~., data = train.data, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneGrid = expand.grid(alpha = 1, lambda = cv_lasso$lambda.min)
  )
# Model coefficients
coef(model_lasso$finalModel, model_lasso$bestTune$lambda)
```

```
14 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  37.199465655
crim         -0.091088918
zn            0.038010919
indus        -0.014083319
chas          2.291379000
nox         -16.745915299
rm            3.520653532
age           0.008671758
dis          -1.370592996
rad           0.316149936
tax          -0.011743014
ptratio      -0.955953859
black         0.009790457
lstat        -0.560615758
```

Hide

```
x.test <- model.matrix(medv ~., test.data)[,-1]
predictions <- model_lasso %>% predict(x.test) %>% as.vector()
# Model performance metrics
data.frame(
  RMSE = RMSE(predictions, test.data$medv),
  Rsquare = R2(predictions, test.data$medv)
)
```

RMSE <dbl>	Rsquare <dbl>
4.587305	0.761773

1 row

Computing Elastic net regression model

Hide

```
set.seed(123)
model_elastic <- train(
  medv ~., data = train.data, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneLength = 10
)
# Best tuning parameter
model_elastic$bestTune
```

	alpha <dbl>	lambda <dbl>
4	0.1	0.03875385

1 row

Hide

```
coef(model_elastic$finalModel, model_elastic$bestTune$lambda)
```

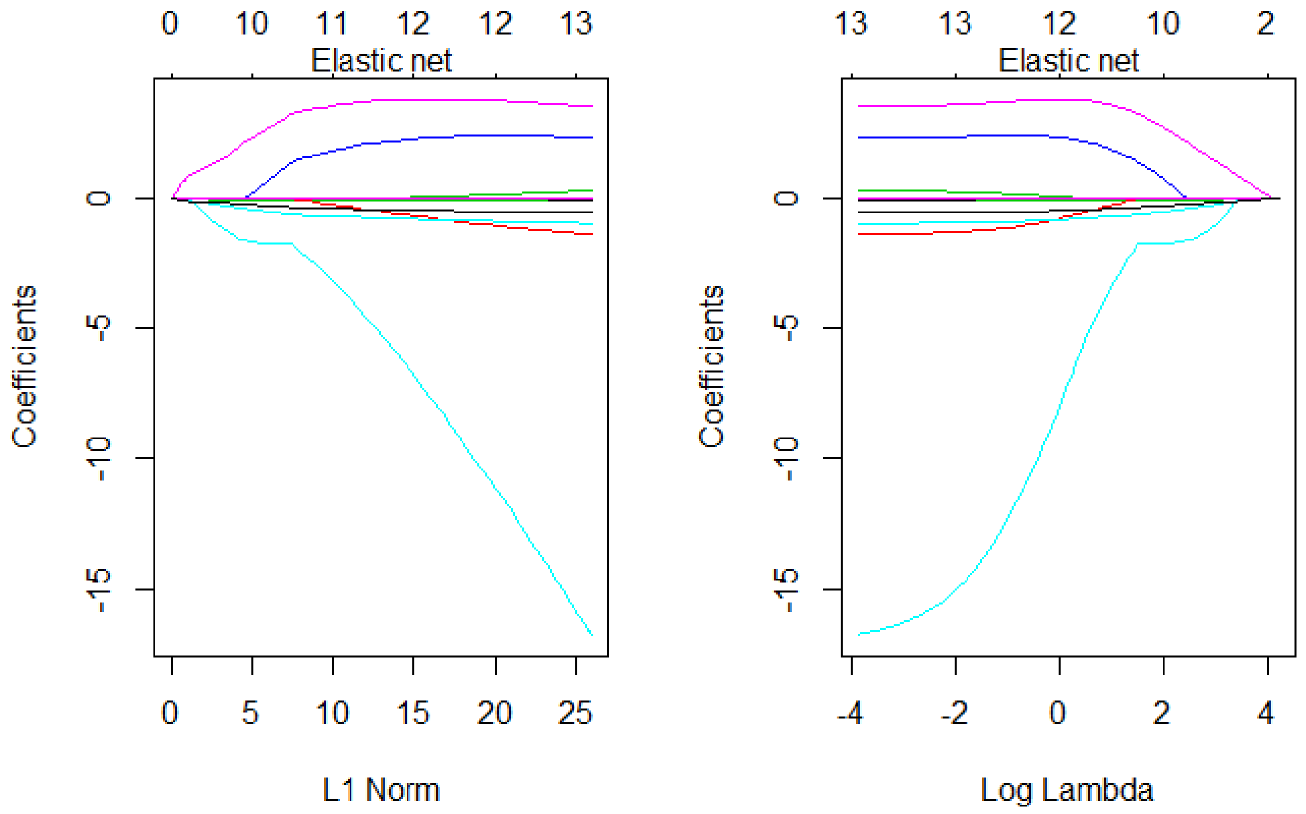
```
14 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  36.727488405
crim         -0.090703096
zn            0.037446370
indus        -0.020415545
chas          2.320231739
nox         -16.457742172
rm            3.533267286
age           0.008434395
dis          -1.358834790
rad           0.305260173
tax          -0.011175268
ptratio      -0.950290918
black         0.009799131
lstat        -0.557178910
```

Hide

```
par(mfrow = c(1, 2))
fit_elastic = glmnet(X, y, alpha = 0.1)
plot(fit_elastic)
mtext("Elastic net")
```

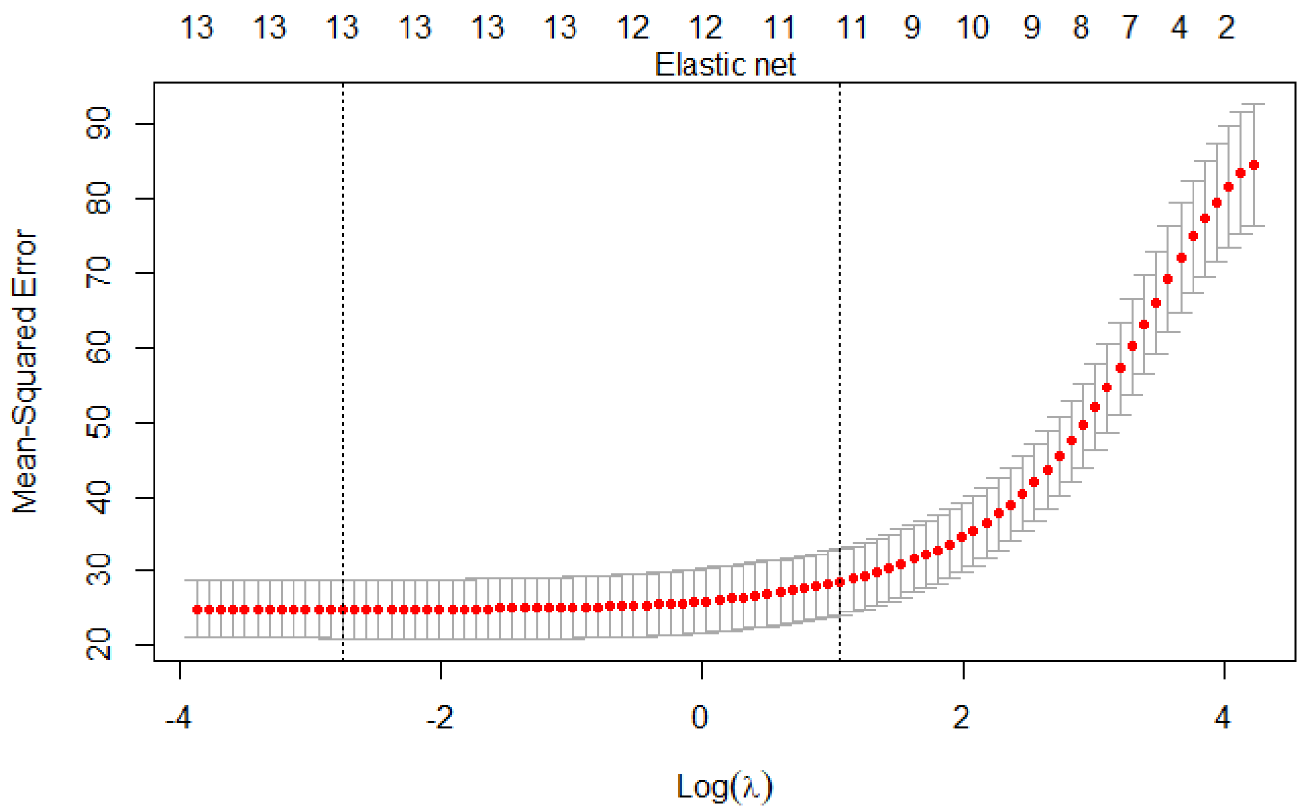
Hide

```
plot(fit_elastic, xvar = "lambda", label = TRUE)
mtext("Elastic net")
```



Hide

```
cv_elastic = cv.glmnet(X, y, alpha = 0.1)
plot(cv_elastic)
mtext("Elastic net")
```



Hide

```
x.test <- model.matrix(medv ~., test.data)[,-1]
predictions <- model_elastic %>% predict(x.test)
# Model performance metrics
data.frame(
  RMSE = RMSE(predictions, test.data$medv),
  Rsquare = R2(predictions, test.data$medv)
)
```

RMSE <dbl>	Rsquare <dbl>
4.587382	0.7621341

1 row

Comparing model performance

Hide

```
models <- list(ridge = model_ridge, lasso = model_lasso, elastic = model_elastic)
resamples(models) %>% summary( metric = "RMSE")
```

```


Call:
summary.resamples(object = ., metric = "RMSE")

Models: ridge, lasso, elastic
Number of resamples: 10

RMSE
            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
ridge   3.306977 3.929839 4.599626 4.847751 5.624674 7.203218    0
lasso   3.618948 4.271887 4.612680 4.793215 5.078720 7.499981    0
elastic 3.630082 4.148174 4.692444 4.816828 5.404555 6.163305    0
```