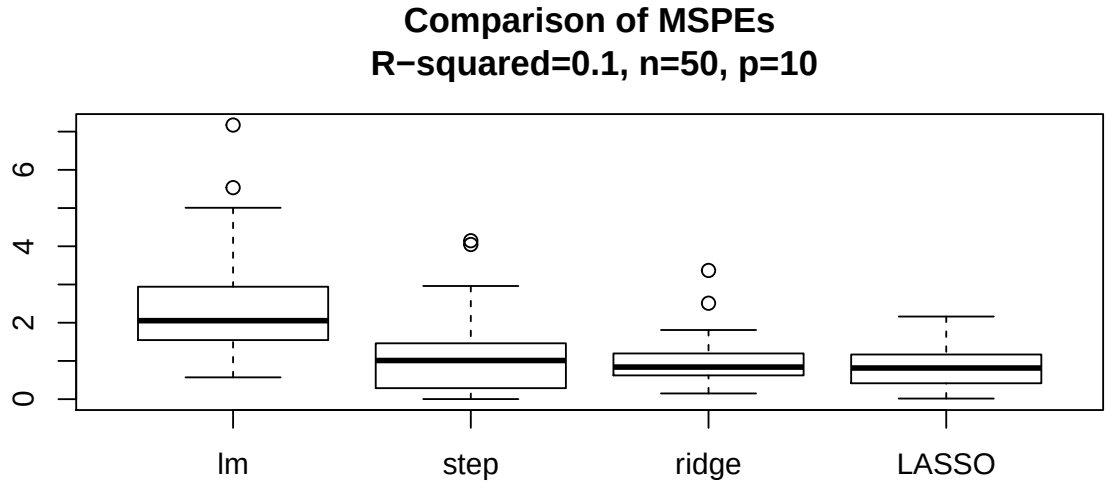
STAT452/652 Solution to HW05 - Lecture 6

1 Concepts

1.1 Question 1

```
n = 50 #  PLAY WITH THIS NUMBER
p = 10 #  PLAY WITH THIS NUMBER
sigma = 3 # PLAY WITH THIS NUMBER

source("L6 Selection Bias and shrinkage (HWK Version) - Modified.R")
```

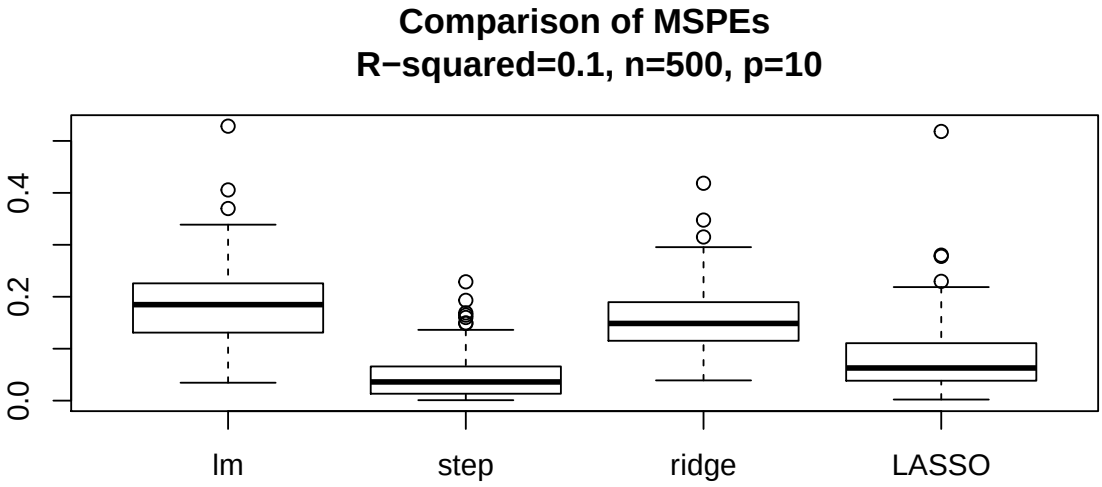


For this parameter combination, LASSO looks like the best model.

1.2 Question 2

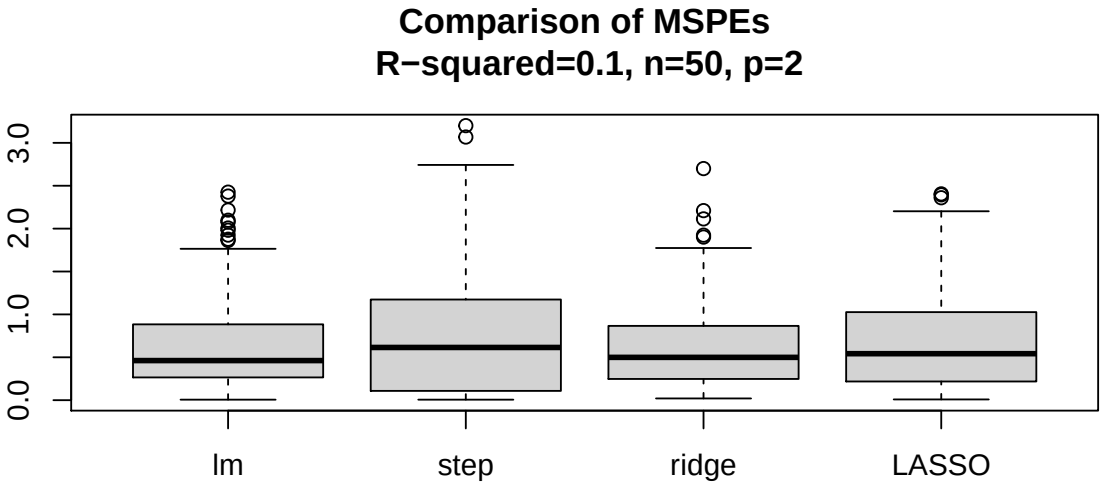
```
n = 500 #  PLAY WITH THIS NUMBER
p = 10 #  PLAY WITH THIS NUMBER
sigma = 3 # PLAY WITH THIS NUMBER

source("L6 Selection Bias and shrinkage (HWK Version) - Modified.R")
```



For this parameter combination, stepwise looks like the best model.

1.3 Question 3

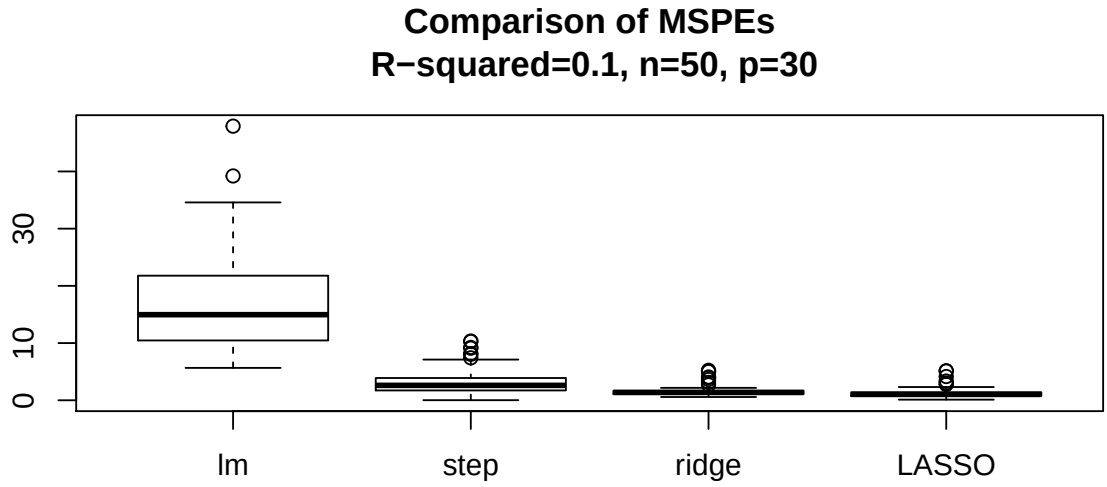


For this parameter combination, LS and ridge look like the best models.

1.4 Question 4

```
n = 50 #  PLAY WITH THIS NUMBER
p = 30 #  PLAY WITH THIS NUMBER
sigma = 3 # PLAY WITH THIS NUMBER
```

```
source("L6 Selection Bias and shrinkage (HWK Version) - Modified.R")
```

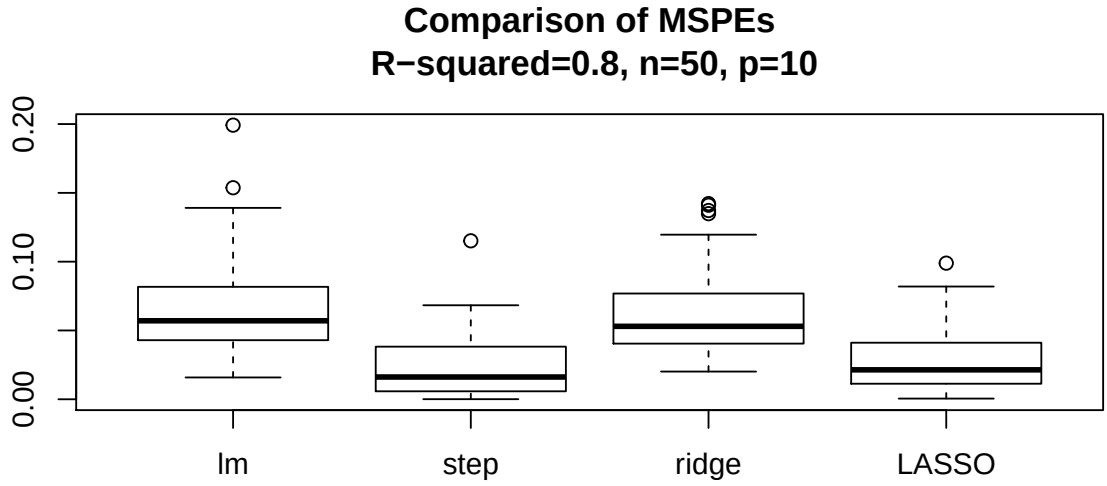


For this parameter combination, ridge and LASSO look like the best models.

1.5 Question 5

```
n = 50 #  PLAY WITH THIS NUMBER
p = 10 #  PLAY WITH THIS NUMBER
sigma = 0.5 # PLAY WITH THIS NUMBER

source("L6 Selection Bias and shrinkage (HWK Version) - Modified.R")
```

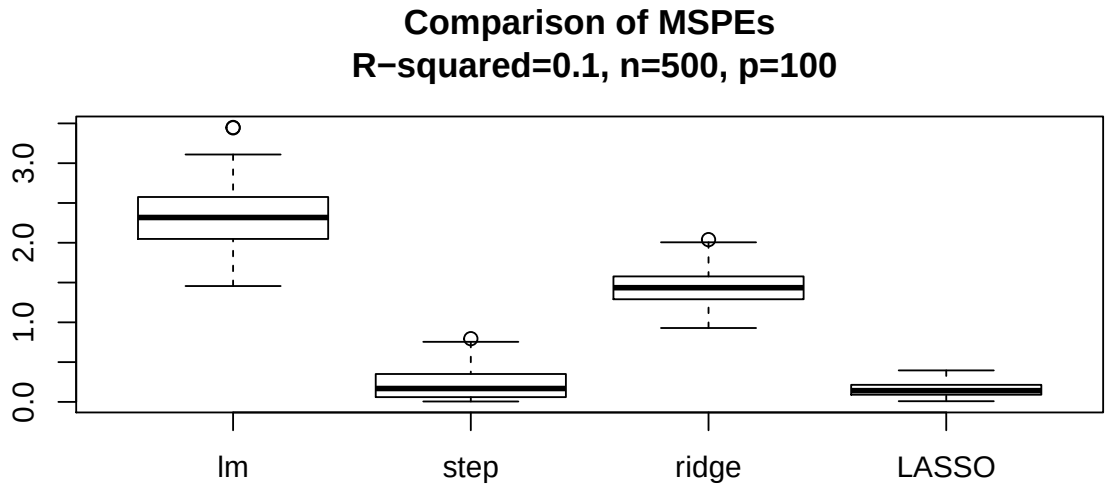


For this parameter combination, stepwise looks like the best model.

1.6 Question 6

```
n = 500 #  PLAY WITH THIS NUMBER
p = 100 #  PLAY WITH THIS NUMBER
sigma = 3 # PLAY WITH THIS NUMBER

source("L6 Selection Bias and shrinkage (HWK Version) - Modified.R")
```

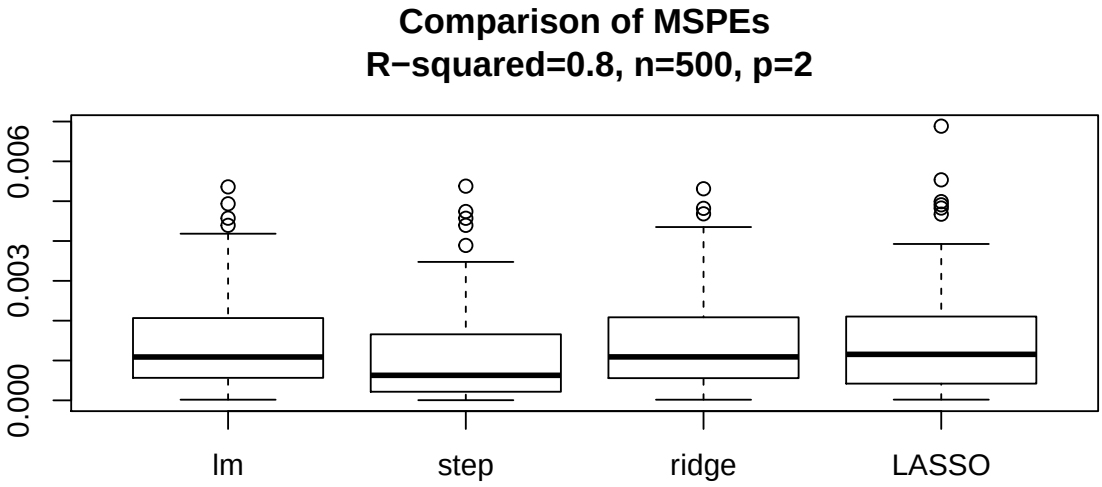


For this parameter combination, LASSO looks like the best model.

1.7 Question 7

```
n = 500 #  PLAY WITH THIS NUMBER
p = 2 #  PLAY WITH THIS NUMBER
sigma = 0.5 # PLAY WITH THIS NUMBER

source("L6 Selection Bias and shrinkage (HWK Version) - Modified.R")
```



For this parameter combination, stepwise looks like the best model.

2 Applications

```
### Import and clean the air quality data
data("airquality")
AQ.raw = na.omit(airquality[,1:4])

### Construct new variables
AQ = AQ.raw
AQ$TWcp = with(AQ.raw, Temp * Wind)
AQ$TWrat = with(AQ.raw, Temp / Wind)
```

2.1 Question 1

```
### Construct candidate lambda values
lambda.vals = seq(from = 0, to = 100, by = 0.05)

### Fit ridge regression
fit.ridge = lm.ridge(Ozone ~ ., lambda = lambda.vals,
  data = AQ)

### Get optimal lambda value
ind.min.GCV = which.min(fit.ridge$GCV)
lambda.min = lambda.vals[ind.min.GCV]
```

The optimal λ value chosen by GCV is 0.2. The fitted coefficient values are as follows:

```r
### Get coefficients for optimal lambda
all.coefs.ridge = coef(fit.ridge)
coef.min.ridge = all.coefs.ridge[ind.min.GCV,]

### Fit lm and extract coefficients
fit.lm = lm(Ozone ~ ., data = AQ)
coef.lm = fit.lm$coef

### Create and print a matrix comparing coefficents across lm and ridge
data.coef = rbind(coef.lm, coef.min.ridge)
rownames(data.coef) = c("lm", "Ridge")
print(signif(data.coef, 3))
```

```
##       (Intercept) Solar.R Wind Temp   TWcp TWrat
## lm           -191  0.0638 9.56 2.89 -0.148  1.37
## Ridge        -162  0.0628 6.83 2.46 -0.109  1.65
```

Not all ridge regression coefficients are smaller. Specifically, the coefficient on the temperature / wind speed ratio is larger for the optimal ridge regression model.

2.2 Question 2

```r
seed = 4099183
set.seed(seed)

### Create design matrix. Drop intercept so glmnet can candle it correctly later
data.mat.raw = model.matrix(Ozone ~ ., data = AQ)
data.mat = data.mat.raw[,-1]

### Get response vector
Y = AQ$Ozone

### Fit CV LASSO
fit.lasso = cv.glmnet(data.mat, Y)

### Extract optimal lambda values
lambda.min = fit.lasso$lambda.min
lambda.1se = fit.lasso$lambda.1se
```

Note that no seed is explicitly specified in this question. I used 4099183 for consistency with the Lecture 3 assignment.

The chosen values of λ are: $\lambda_{min} = 1.35$, and $\lambda_{1se} = 10.42$.

Fitted coefficient values are:

```r
### Extract fitted coefficients
coef.min = predict(fit.lasso, s = lambda.min, type = "coef")
coef.1se = predict(fit.lasso, s = lambda.1se, type = "coef")

### Combine and print coefficient vectors
coef.lasso = cbind(coef.min, coef.1se)
```

```r
coef.lasso = t(coef.lasso)
rownames(coef.lasso) = c("min", "1se")
print(signif(coef.lasso, 3))
```

```
## 2 x 6 sparse Matrix of class "dgCMatrix"
##     (Intercept) Solar.R Wind  Temp    TWcp TWrat
## min       -83.0  0.0484    . 1.280 -0.0075  2.39
## 1se       -31.5  .         . 0.742  .       1.68
```

Under the 1SE rule, more coefficients are set to zero, and the non-zero coefficients are mostly smaller.

In the Lecture 5 assignment, hybrid stepwise selected temperature, solar radiation, and the ratio of temperature over wind speed (and also the intercept, but this shouldn't have been included in the selection process). Under the Min rule, all these variables remain active, and the product of temperature and wind speed is also included. Under the 1SE rule, solar radiation is excluded, but otherwise the active variables are the same as hybrid stepwise.

2.3 Question 3

```r
### Create function to compute MSPEs
get.MSPE = function(Y, Y.hat){
  return(mean((Y - Y.hat)^2))
}

### Create function which constructs folds for CV
### n is the number of observations, K is the number of folds
get.folds = function(n, K) {
  ### Get the appropriate number of fold labels
  n.fold = ceiling(n / K) # Number of observations per fold (rounded up)
  fold.ids.raw = rep(1:K, times = n.fold)
  fold.ids = fold.ids.raw[1:n]

  ### Shuffle the fold labels
  folds.rand = fold.ids[sample.int(n)]

  return(folds.rand)
}
```

```r
K = 10 #Number of folds

set.seed(2928893)

### Container for CV MSPEs
all.models = c("LS", "Step", "Ridge", "LASSO-Min", "LASSO-1se")
CV.MSPEs = array(0, dim = c(length(all.models), K))
rownames(CV.MSPEs) = all.models
colnames(CV.MSPEs) = 1:K

### Get CV fold labels
n = nrow(AQ)
folds = get.folds(n, K)
```

```r
### Perform cross-validation
for (i in 1:K) {
  ### Get training and validation sets
  # data.train = na.omit(AQ[folds != i, ])
  # data.valid = na.omit(AQ[folds == i, ])
  data.train = AQ[folds != i, ]
  data.valid = AQ[folds == i, ]
  Y.train = data.train$Ozone
  Y.valid = data.valid$Ozone

  ### We need the data matrix to have an intercept for ridge, and to not have an intercept for LASSO. B
  mat.train.int = model.matrix(Ozone ~ ., data = data.train)
  mat.train = mat.train.int[,-1]
  mat.valid.int = model.matrix(Ozone ~ ., data = data.valid)
  mat.valid = mat.valid.int[,-1]

  ##########
  ### LS ###
  ##########

  fit.ls = lm(Ozone ~ ., data = data.train)
  pred.ls = predict(fit.ls, data.valid)
  MSPE.ls = get.MSPE(Y.valid, pred.ls)
  CV.MSPEs["LS", i] = MSPE.ls


  ############
  ### Step ###
  ############

  fit.start = lm(Ozone ~ 1, data = data.train)
  fit.step = step(fit.start, list(upper = fit.ls), trace = 0)
  pred.step = predict(fit.step, data.valid)
  MSPE.step = get.MSPE(Y.valid, pred.step)
  CV.MSPEs["Step", i] = MSPE.step

  ##############
  ### Ridge ###
  ##############

  ### Fit ridge regression
  ### We already definted lambda.vals. No need to re-invent the wheel
  fit.ridge = lm.ridge(Ozone ~ ., lambda = lambda.vals,
    data = data.train)

  ### Get optimal lambda value
  ind.min.GCV = which.min(fit.ridge$GCV)
  lambda.min = lambda.vals[ind.min.GCV]

  ### Get coefficients for optimal model
  all.coefs.ridge = coef(fit.ridge)
  coef.min.ridge = all.coefs.ridge[ind.min.GCV,]
```

```
  ### Get predictions and MSPE on validation set
  pred.ridge = mat.valid.int %*% coef.min.ridge
  pred.ridge = as.numeric(pred.ridge)
  MSPE.ridge = get.MSPE(Y.valid, pred.ridge)

  CV.MSPEs["Ridge", i] = MSPE.ridge


  ##############
  ### LASSO ###
  ##############

  ### Fit model
  fit.LASSO = cv.glmnet(mat.train, Y.train)

  ### Get optimal lambda values
  lambda.min = fit.LASSO$lambda.min
  lambda.1se = fit.LASSO$lambda.1se

  ### Get predictions
  pred.min = predict(fit.LASSO, mat.valid, lambda.min)
  pred.1se = predict(fit.LASSO, mat.valid, lambda.1se)

  ### Get and store MSPEs
  MSPE.min = get.MSPE(Y.valid, pred.min)
  MSPE.1se = get.MSPE(Y.valid, pred.1se)
  CV.MSPEs["LASSO-Min", i] = MSPE.min
  CV.MSPEs["LASSO-1se", i] = MSPE.1se
}

### Get full-data MSPEs
full.MSPEs = apply(CV.MSPEs, 1, mean)
```

The MSPEs of ridge regression and both LASSO versions on each fold, as well as on the full dataset, are as follows.

```
### Short-list of models for printing
short.models = c("Ridge", "LASSO-Min", "LASSO-1se")

### Get relevant fold-wise and overall MSPEs
short.CV.MSPEs = CV.MSPEs[short.models,]
short.full.MSPEs = full.MSPEs[short.models]

### Combine and print foldwise/full MSPEs
short.MSPEs = cbind(short.CV.MSPEs, short.full.MSPEs)
colnames(short.MSPEs) = c(1:K, "Full")
print(signif(short.MSPEs, 3))
```

```
##             1   2   3   4   5   6   7   8    9  10 Full
## Ridge     281 364 539 108 185 372 220 588 1040 725  442
## LASSO-Min 314 317 586 115 190 373 217 586 1050 651  439
## LASSO-1se 416 237 717 157 324 669 367 643 1130 378  503
```

Boxplots of MSPEs and RMSPEs are given in Figures 1 and 2 respectively.

```
### MSPE Boxplot
plot.MSPEs = t(CV.MSPEs)
boxplot(plot.MSPEs)
```

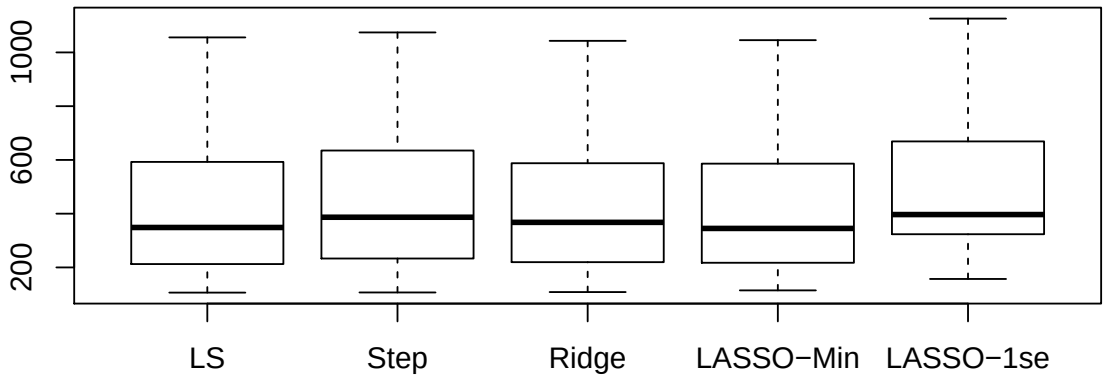


Figure 1: MSPE Boxplots

```
### Compute RMSPEs
plot.RMSPEs = apply(CV.MSPEs, 2, function(W){
  best = min(W)
  return(W/best)
})
plot.RMSPEs = t(plot.RMSPEs)

### RMSPE Boxplot
boxplot(plot.RMSPEs)
```

Based on the MSPEs in Figure 1, LS, ridge and LASSO-min look like the best models, with stepwise a bit worse, and LASSO-1se a bit worse still. That being said, all the models' boxplots have considerable overlap, and no model looks substantially better or worse.

Based on the RMSPEs in Figure 2, LS and ridge seem to be performing best, with LS looking a bit better. Stepwise and LASSO-min are the next-best pair, with LASSO-min looking a bit better than stepwise. LASSO-1se is clearly the worst performing model.

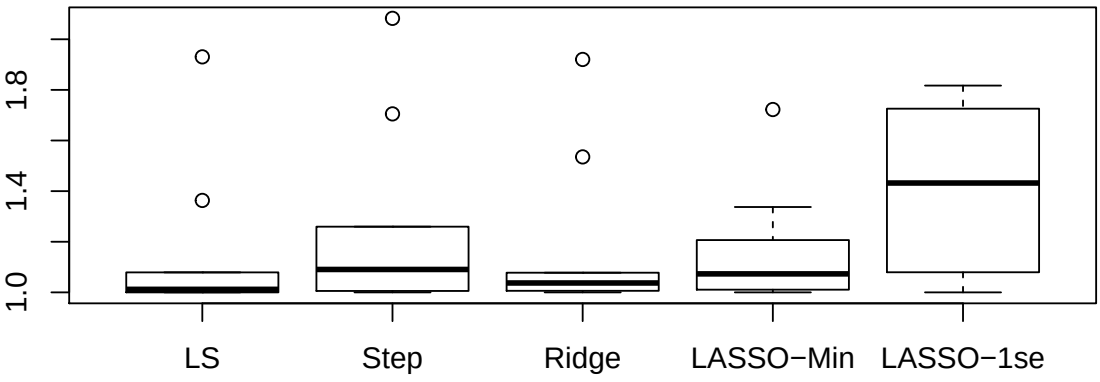


Figure 2: RMSPE Boxplots