STAT452/652 Solution to Lecture 7

```
knitr::opts_chunk$set(echo = T, message = F,
  fig.width = 7, fig.height=3.5)
options(scipen=999)
```

```
library(MASS)
library(glmnet)
library(caret)
library(pls)
```

1 Applications - Insurance

1.1 Question 1

```
### Read-in data
ins = read.csv("Insurance.csv")

### Convert categorical predictors to factors
ins$zone = as.factor(ins$zone)
ins$make = as.factor(ins$make)

### Remove observations with no claims
ins = dplyr::filter(ins, claims > 0)

### Convert factors to redundant indicators
ins.dv = data.frame(predict(dummyVars(~ ., data=ins), newdata=ins))
```

After constructing indicator variables, our data frame has 1797 rows and 21 columns.

1.2 Question 2

```
### Carry out a PCA without the response (which is stored in column 1)
fit.PCA = prcomp(ins.dv[,-1], scale. = T)

### Compute proportion of variance explained
vars = fit.PCA$sdev^2

### Get number of PCs with variance explained >= 1
n.keep = sum(vars >= 1)
n.comps = length(vars) # Total number of PCs, for reference
```

Using the rule of only keeping PCs which explain an above average proportion of variance, we would keep 15 of the 20 components.

See Figures 1 and 2 for the scree and cumulative variance plots respectively. Note that we add the point (0,0) to the cumulative variance plot to illustrate the effect of the first component (if 0 components are included, the proportion of variability explained is 0).

```
plot(1:length(vars), vars,
  xlab = "Principal Component", ylab = "Variance Explained")
abline(h = 1)
```

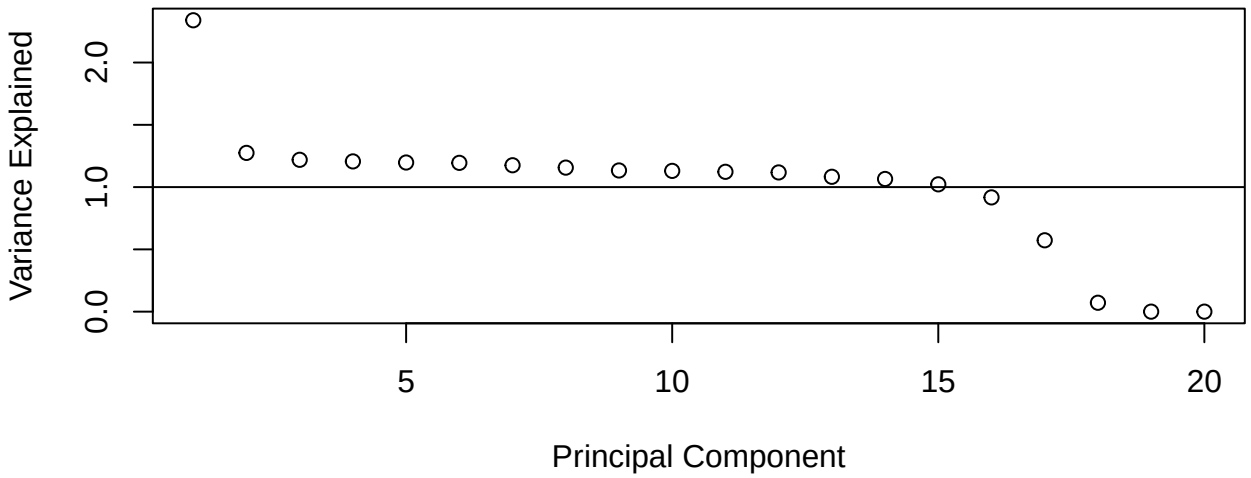

Figure 1: Scree plot for insurance data.

```
c.vars = cumsum(vars)   ### Cumulative variance explained
rel.c.vars = c.vars / max(c.vars)   ### Cumulative proportion of
                                    ### variance explained
### Create cumulative variance plot. Add 0 components to illustrate dramatic
### drop in improvement after first component.
plot(0:length(rel.c.vars), c(0,rel.c.vars), ylim = c(0,1),
  xlab = "Number of Components", ylab = "Proportion of Variance Explained")
```

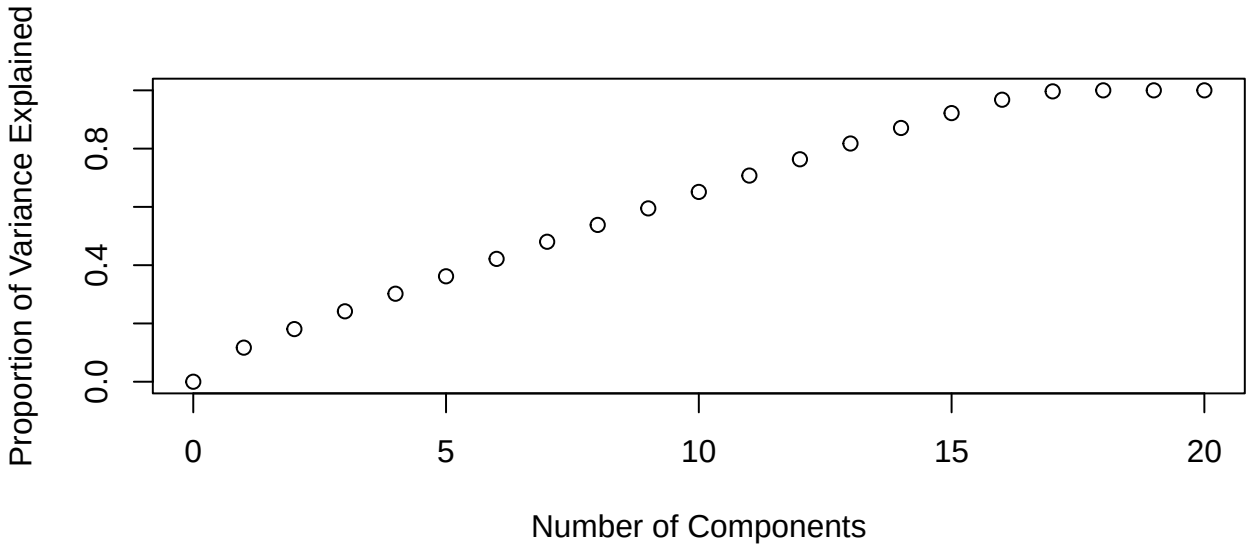

Figure 2: Cumulative variance plot for insurance data.

Based on Figures 1 and 2, I think that a single component best balances variance explained with dimension

reduction. Based on the scree plot (Figure 1), we see that variance explained by the first component is nearly double that of the second component, and that none of the other dropoffs are this dramatic.

One might also prefer to choose 15 or 16 components, because most components up to this point are about equally valuable (other than the first component), and after this point there is little variance left to be explained.

2 Applications - Air Quality

```
##############################
### Import and process data ###
##############################

### 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)

########################
### Helper Functions ###
########################

### 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)
}
```

```
K = 10 #Number of folds

set.seed(2928893)

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

```r
colnames(CV.MSPEs) = 1:K

### Container for number of components chosen by PLS for each fold
CV.n.comps = rep(0, times = K)

### Construct candidate lambda values (outside loop to save time)
lambda.vals = seq(from = 0, to = 100, by = 0.05)

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

### 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
```

```r
### 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["LAS-Min", i] = MSPE.min
CV.MSPEs["LAS-1se", i] = MSPE.1se


##############################
### Partial Least Squares ###
##############################

### Fit PLS
fit.pls = plsr(Ozone ~ ., data = data.train, validation = "CV",
  segments = 10)

### Get optimal number of folds
CV.pls = fit.pls$validation
PRESS.pls = CV.pls$PRESS
```

```
  n.comps = which.min(PRESS.pls)
  CV.n.comps[i] = n.comps

  ### Get predictions and MSPE
  pred.pls = predict(fit.pls, data.valid, ncomp = n.comps)
  MSPE.pls = get.MSPE(Y.valid, pred.pls)
  CV.MSPEs["PLS", i] = MSPE.pls
}

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

The chosen number of components and MSPE on each fold is as follows.

```
info.pls = rbind(round(CV.n.comps, 0),
  round(CV.MSPEs["PLS",], 0))
rownames(info.pls) = c("Components", "MSPE")
print(info.pls)
```

```
##              1   2   3   4   5   6   7   8    9  10
## Components   5   5   3   3   3   3   3   3    5   3
## MSPE       260 323 514 138 146 417 266 656 1056 699
```

The full-data MSPE for PLS is 447.

Boxplots of MSPEs and RMSPEs are given in Figures 3 and 4 respectively.

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

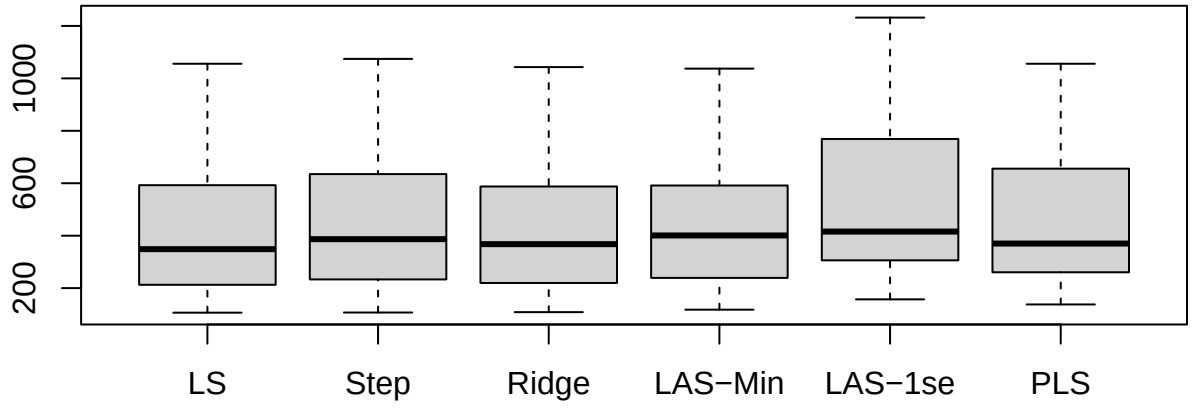

Figure 3: MSPE Boxplots

```
### 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)
```

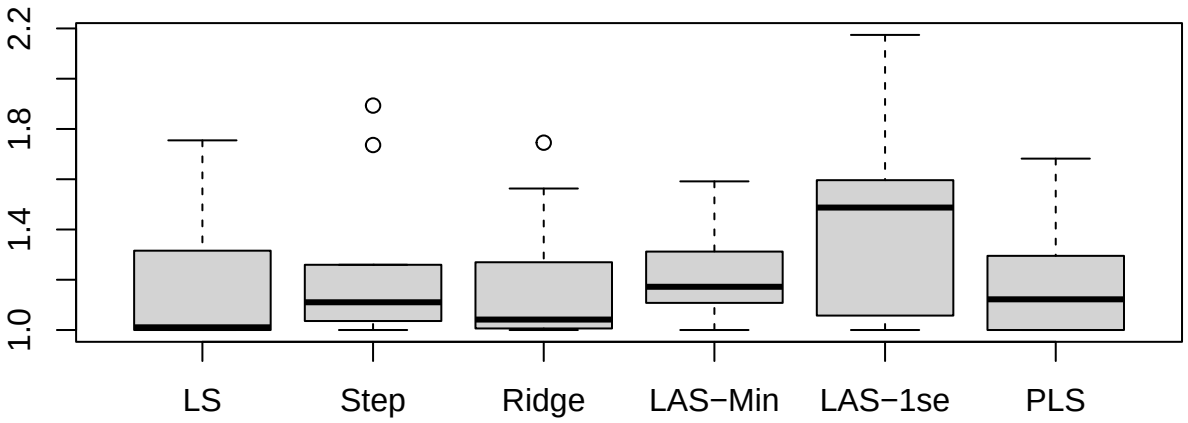

Figure 4: RMSPE Boxplots

Based on the MSPE boxplots in Figure 3, PLS appears to be competitive with stepwise selection as a second-tier model (i.e. better than LASSO-1SE, but worse than the rest).

Based on the RMSPE boxplots in Figure 4, PLS appears to be almost competitive with LS and Ridge, which have the lowest median RMSPEs. While PLS wins the competition at least 3 times (the bottom of its box, i.e. the first quartile, is at 1), this model's median RMSPE is worse than either LS or Ridge.