STAT 452/652: Statistical Learning and Prediction

Owen G. Ward
owen_ward@sfu.ca

Fall 2023

These notes are largely based on previous iterations of this course, taught by Prof. Tom Loughin and Prof. Haolun Shi.

6 Dimension Reduction

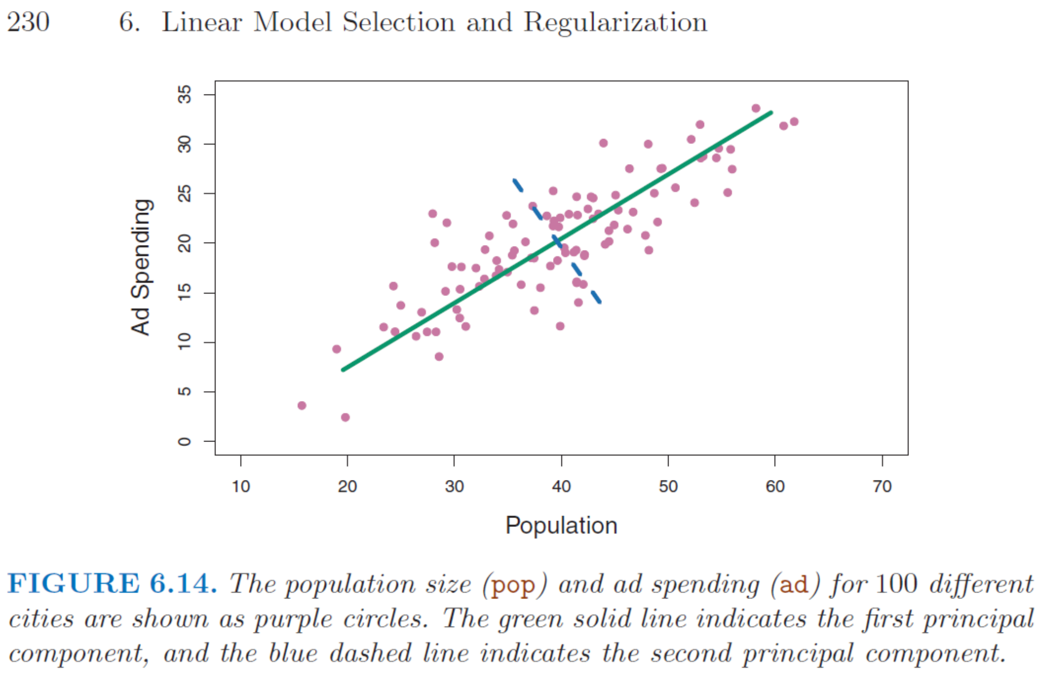
(Reading: ISLR 6.3)

Goals of this section

- In order to reduce variance in prediction it helps to reduce the number of parameters to be estimated
- Sometimes explanatory variables are highly correlated
  - It is difficult to know which variable(s) to select from a group
  - Keeping them all adds variability needlessly when not all of them are needed
- It would help to have a way to "boil down" correlated variables to a smaller number of measures that contain the main information within them.
  - E.G., maybe there are 10 variables all measuring some aspect of "size" of a person, and some notion of size is needed in a model, but not in 10 different versions.
- Can we use the data to engineer these new features for us automatically?

6.1 Principal Components

- Sometimes p is too large to work with (e.g. $p > n$)
- Even when it's not, it might be much larger than the dimensionality of the actual information contained with in the X-space.
  - For example, when X_1 and X_2 are strongly correlated, the information in (X_1, X_2) lies "almost" in a 1-dimensional space
  - See Fig 1.



230 6. Linear Model Selection and Regularization

FIGURE 6.14. *The population size (*`pop`*) and ad spending (*`ad`*) for* 100 *different cities are shown as purple circles. The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.*

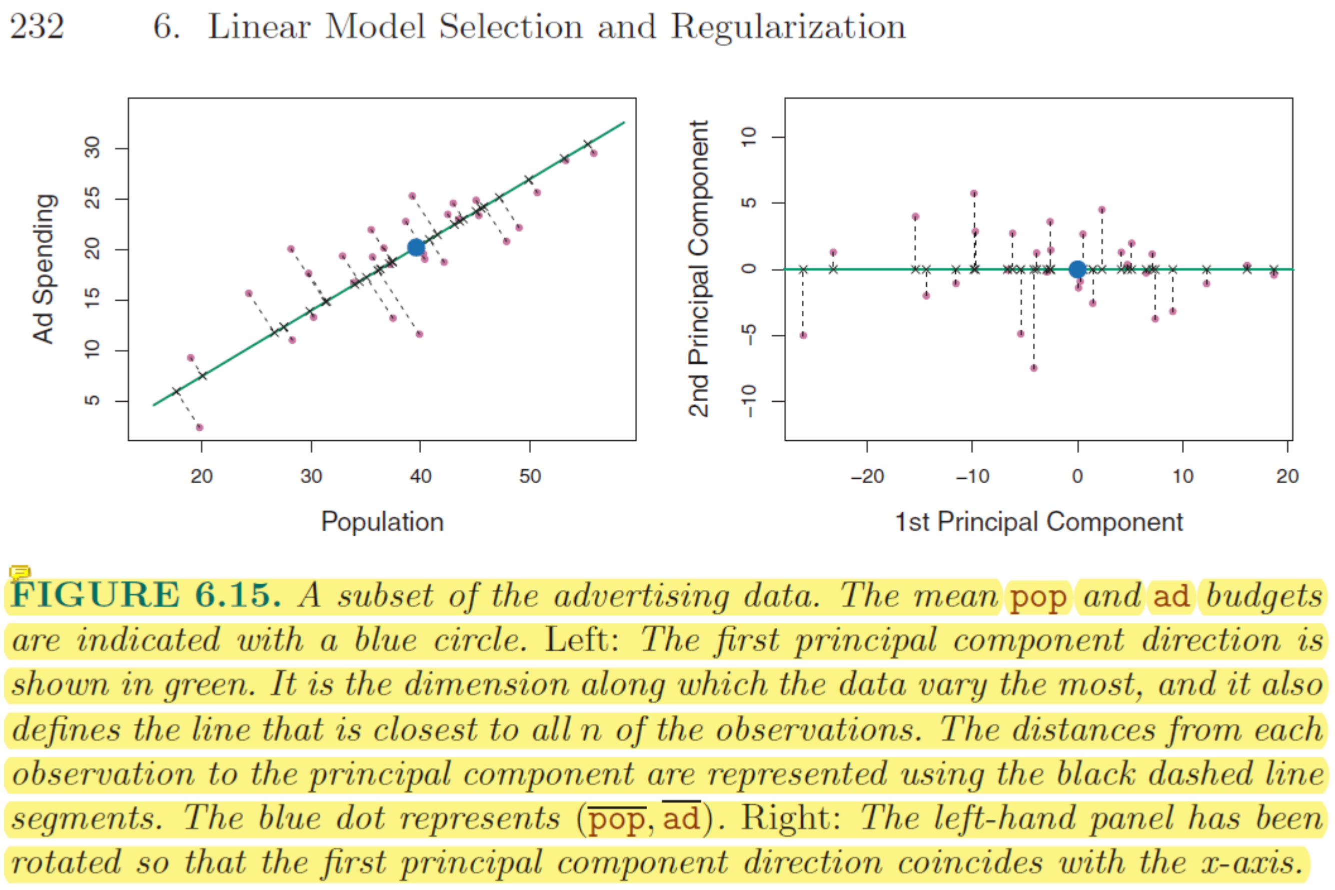
Figure 1: Correlated variables in $p = 2$ dimensions can almost be represented by a single variable along the straight green line. (From ISLR)

- More complex relations may cause X to be nearly p_1-dimensional, for $p_1 < p$.
- It would be nice to be able to reduce the dimension of X before using it for other things. Might help us when fitting models also.

Principal Components Analysis (PCA)

1. Usually start by standardizing each X_j to have mean 0 and SD 1, but not strictly necessary
2. Find the direction within X that explains the most variability in X.
   (a) "Project" points onto different lines through the middle of the data
   (b) Compute variance of points on each line
   (c) Choose the line where the variance is highest
      i. This is the FIRST PRINCIPAL COMPONENT (PC1)
   (d) Project all data onto this line (See Fig 2).
   (e) The formula for this line is $Z_1 = \sum_{j=1}^{p} X_j \phi_{1_j}$, where the values $\phi_{11}, \ldots, \phi_{1p}$ are found from the data[1]
   (f) Each observation has a value of Z_1 that represents its position on a new coordinate axis.
      i. Projecting p-dimensional points orthogonally onto this axis
   (g) "Total variance" of data is sum on variances for each X_j

[1] Principal components are found using eigen decomposition, in case you were curious.



232 6. Linear Model Selection and Regularization

FIGURE 6.15. *A subset of the advertising data. The mean* `pop` *and* `ad` *budgets are indicated with a blue circle.* Left: *The first principal component direction is shown in green. It is the dimension along which the data vary the most, and it also defines the line that is closest to all n of the observations. The distances from each observation to the principal component are represented using the black dashed line segments. The blue dot represents* $(\overline{\texttt{pop}}, \overline{\texttt{ad}})$. Right: *The left-hand panel has been rotated so that the first principal component direction coincides with the x-axis.*

Figure 2: Principal components in $p = 2$ dimensions. (From ISLR)

   i. Variance of Z_1 points is larger than variance of any single X_j, but less than the total variance

3. Next look for the direction *orthogonal (perpendicular) to the first* that explains the next-most variability in X

   (a) Direction is $Z_2 = \sum_{j=1}^{p} X_j \phi_{2j}$ for a new set of values $\phi_{21}, \ldots, \phi_{2p}$

   (b) Z_1 and Z_2 are uncorrelated

   (c) A second coordinate axis

      i. Projecting points onto this plane

   (d) Variance of points around Z_2 is less than variance around Z_1

   (e) Sum of these is smaller than or equal to total variance.

4. Keep repeating this until there is no more variability (or until you reach p components if there are p variables and they are not perfectly correlated.)

   (a) Have $Z_1, Z_2, \ldots, Z_p$ new variables engineered from X

   (b) These are called the PRINCIPAL COMPONENTS (PC's) of X.

5. Sum of variances of Z's equals the total variance

   (a) It just splits it into new linear variables that each explain as much as they can in sequence

6. The hope is that just a few ($M < p$) PCs explain most of the variability in X, so you just use those PCs for further use.

   (a) In Fig. 2, I *might* choose just to keep the first component and drop the second.

   (b) Projecting the points onto the "reduced-dimensional subspace" loses information, but loss is minimized given M.

7. What we are really doing is rotating the axes to more directly align with the contents of the data. then dropping some axes (Z_j's)

Choosing M

You can do this before the regression or after. See later for the "after" approach.

- Suppose we let T be the "total variance", the sum of the variances from each X_j , $j = 1, \ldots, p$
- Suppose we let v_j be the variance of the PC Z_j, $j = 1, \ldots, p$
- Then we know that $\sum_{j=1}^{p} v_j = T$
- The whole point of a PCA is hoping that you can find an $M < p$ so that $\sum_{j=1}^{M} v_j \approx \sum_{j=1}^{p} v_j$
- So plot $\sum_{j=1}^{m} v_j$ vs. m for different values of m (cumulative variance plot)
  - Increases rapidly, then levels off
  - Choose M to be a value of m where the increase is no longer rapid and the leveling off has begun.
  - Will see this in an example
- Or can plot v_j vs. j (SCREE PLOT) to see if there is a clear point at which v_j's level off
- If you choose to standardize data so that each X_j has SD=1, interpretation is a little easier
  - Total variance $T = \sum_{j=1}^{p} v_j = p$, so average v_j is 1
  - Another possible way to select M is to choose all j with $v_j > 1$

Example: Finding principal components on Prostate Data (`Sec6_Principal_Comp_Partial_LS.R`) We will use the full prostate data with 8 explanatory variables to perform PCA using the `prcomp()` function from the main `stats` package. We produce 8 PC's, $Z_1, \ldots, Z_8$ and see (1) how much of the total variance each one explains, and (2) what the coefficients look like. Adding `scale.=TRUE` does the standardizing. The output shows the `Standard deviation` ($\sqrt{v_j}$), the `Proportion of Variance` (v_j/T), and the `Cumulative Proportion` ($\sum_{l=1}^{j} v_l/T$) for PC Z_j, $j = 1, \ldots, p$.

The output is shown below.

```
> pc <-  prcomp(x=prostate[,2:9], scale.=TRUE)
> summary(pc)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5
Standard deviation     1.8234 1.2846 1.0219 0.79990 0.69177
Proportion of Variance 0.4156 0.2063 0.1305 0.07998 0.05982
Cumulative Proportion  0.4156 0.6219 0.7524 0.83241 0.89223
                           PC6     PC7     PC8
Standard deviation     0.60959 0.51723 0.47227
Proportion of Variance 0.04645 0.03344 0.02788
Cumulative Proportion  0.93868 0.97212 1.00000
```

We can also make a scree plot or could make a cumulative variance plot. The scree plot is available from the `plot()` function directly on the `prcomp` object, but we can create a slightly nicer one with a tiny bit of programming. See Figure 3. It is usually hard to draw a line between “keep” and “drop”, and this is no exception. The curve in the plots changes slope kind of slowly instead of suddenly. The simple $v_j > 1$ rule says to take $M = 3$. It seems like the PCs from 5–8 seem not to add much variance, so you might choose 4 or 3 PCs. However, there is no “right” answer.

Finally the program does print out the actual ϕ weights, but they don't really tell you anything interesting for this example, so we don't show them here. Sometimes, you can see interesting relationships in the data from them.

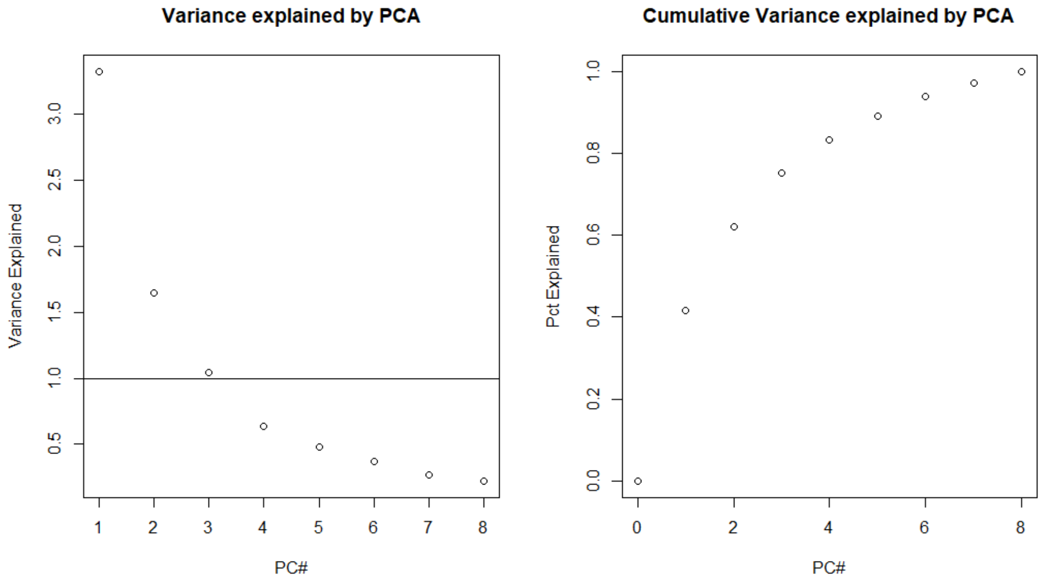


Figure 3: Scree plots for Prostate data. Left: the default produced by `plot.prcomp`; Right: A modifed version of the same information.

6.2 Principal Components Regression

- Principal components regression (PCR) is just ... regression on the principal components!
- Instead of fitting $f_X(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$ with $p+1$ parameters, fit

  $$f_{ZM}(Z) = \theta_0 + \theta_1 Z_1 + \ldots + \theta_M Z_M$$

  which has just $M + 1$ regression parameters.

Details

- Once the PCs $Z_1, \dots, Z_p$ are created, and the choice of $M < p$ is made, there is nothing new.
  - It is just linear regression on the engineered features instead of the original variables.
- Because each Z is made up of different amounts of *all* X_j*'s*, this is not a variable selection process
  - Every original X_j contributes to the predicted values, but in different ways
  - Can function sort of like shrinkage
- Factoid: If you let $M = p$ (i.e., use all of the PCs as long as $n > p$) you get back exactly the same predicted values with $Z_1, \dots, Z_p$ as you would from a linear regression on $X_1, \dots, X_p$

Treating M as a tuning parameter

- Previous selection of M in Section 6.1 used only information about variance within X
  - Process does not use errors in Y to balance bias and variance
  - It is merely trying to restructure information in X to retain "as much of X as possible" while reducing the number of parameters we will later ask regression to estimate.
  - It is totally based on judgment and guesswork.
- Alternatively, you could defer choice of M and let Y help you decide later
- Specifically, you can think of PCR is a *family* of regression models
  - Kind of like polynomials, you can try fitting regressions with $M = 1, 2, 3, \dots, p$
  - Then choose the optimal value of M using CV or some other measure of validation error.
- This method uses Y, and is subject to usual concerns about selecting one model from many

6.3 Partial Least Squares

- PCR suffers from the fact that the newly engineered features are made without knowledge of Y
  - Just because a PC explains the largest amount of variance *within* X doesn't mean that that particular combination explains much variability *in* Y
  - Even if you treat M as a tuning parameter, you still don't use Y to influence the ϕ weights in the PC's
    - ∗ They are entirely determined by the correlation structure among the X_j's
- Partial Least Squares (PLS) similarly creates weighted combinations of the X_j's, but it lets the responses choose the ϕ weights
- The algorithm is more complicated than I want to explain, but the essence is this:
  1. Start by computing the correlation between Y and each X_j

2. Use rescaled versions of the correlations as the ϕ weights for the first component:

$$Z_1^{PLS} = \phi_{11}^{PLS} X_1 + \phi_{12}^{PLS} X_2 + \ldots + \phi_{1p}^{PLS} X_p$$

   (a) The stronger the correlation is between X_j and Y compared to other variables, the more it counts toward Z_1^{PLS}

3. Repeat the process on residuals from regressions of Y vs. Z_1^{PLS} and each X_j vs. Z_1^{PLS} to get a new set of correlations and new weights

   (a) These create Z_2^{PLS}

4. Repeat for a total of M times for some chosen M

5. Do multiple linear regression of $Y = \theta_0 + \theta_1 Z_1^{PLS} + \ldots + \theta_M Z_M^{PLS}$

- Can treat M as a tuning parameter and choose it using CV
  - In both PLS and PCR, must perform *entire process* (creating components, performing regressions for different M) on training set
  - Produce a set of predictions on validation set for each regression model
  - DO NOT use full data to create components, and then run only the regression part on training set.
- PLS has similar properties to PCR, but may need smaller M because components do use information about Y
  - For this reason, PLS is probably the better tool to learn
  - Performance reportedly tends to be similar
  - In both cases, you can rearrange the coefficients algebraically to rewrite the prediction function as a linear regression in terms of all 8 variables, but with coefficients that are different from LS or Ridge.

Example: Partial Least Squares on Prostate Data (`Sec6_Principal_Comp_Partial_LS.R`) This is really not the right data set for showing off what PLS can do. PLS is really meant for cases where p is larger than we are willing to make a model out of, so we need to reduce the dimension somehow. In this case, $p = 8$, so it's kind of a dull demonstration. This is just to show how it works

The `pls` package does both PLS and PCR regression. We will use the `plsr()` function for PLS regression. The code is pretty similar to lm() except that we can specify:

1. The maximum number of components we want it to consider, which is useful when p is very large, and

2. The option to use validation to choose number of components adding the `validation=` argument

   (a) Can use `"none"`, `"CV"`, or `"LOO"` (leave one out CV).

   (b) If `"CV"` is used, V=10 unless changed with `segments=`

3. LOTS more options and helper functions to better understand the model results, which we are not specifically interested in at the moment. [2]

In the output below, the validation error is the root-MSPE, and there are lines for the CV error as we learned it and for a "bias-adjusted" version that we haven't read about. The regular version is fine for our purposes. Under `TRAINING`, the `lpsa` rows show the model R^2 fitting 1,2,. . . 8 PLS components.

```
> library(pls)
>
> mod.pls <- plsr(lpsa~., data=prostate, ncomp=8, validation="CV")
> summary(mod.pls)
Data:   X dimension: 97 8
  Y dimension: 97 1
Fit method: kernelpls
Number of components considered: 8

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps
CV            1.16     1.05    1.002   0.8206   0.7805
adjCV         1.16     1.05    1.038   0.8171   0.7767
       5 comps  6 comps  7 comps  8 comps
CV      0.7599   0.7510   0.7471   0.7462
adjCV   0.7573   0.7465   0.7428   0.7420

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X       93.48    98.52    99.67    99.76    99.94    99.97
lpsa    18.19    26.68    56.31    62.15    63.65    66.06
      7 comps  8 comps
X       99.99   100.00
lpsa    66.31    66.34
```

Strangely, CV has selected a model with all 8 components, although the root-MSPE's are not changing much after about 4. This is clear in the R^2 values as well—there is little gain after about 4 components. 4 might be a reasonable choice.

What to take away from this

- Dimension reduction uses feature engineering on ALL X_j to create new variables that are linear combinations of the originals

- Run regression on small number of these components

  - Considers information from each X_j

  - Uses fewer parameters than running p-dimensional regression

- Drawback is that there is no guarantee that the components capture all relevant information about complicated relationships between Y and the X_j's

- Also no reduction in the number of variables that must be measured to create the predictions

[2] Feel free to explore the package vignette at https://cran.r-project.org/web/packages/pls/vignettes/pls-manual.pdf

Problem Set 8: Dimension Reduction

Application

Motor Insurance Data:

Refer to the motor insurance data that we studied in the exercises in Lecture 4. Recall that there are two factors in the data.

1. Convert the two factors, `make` and `zone` into full sets of indicator variables, add them to the data frame, and drop the factors. An easy way to do this is using the `dummy.vars()` function from the package `caret`. For example, my code looks like this:

```
> library(caret)
> ins.dv = data.frame(predict(dummyVars("~.", data=ins), newdata=ins))
> head(ins.dv)
        per km zone.1 zone.2 zone.3 zone.4 zone.5 zone.6
1 11.649460  2      0      0      1      0      0      0
2  9.615939  3      0      0      1      0      0      0
4  8.028129  2      0      0      0      0      0      0
5 11.222640  1      0      0      0      1      0      0
7 11.290310  3      0      0      0      0      1      0
8 10.163730  2      0      1      0      0      0      0
  zone.7 bonus make.1 make.2 make.3 make.4 make.5 make.6
1      0     2      0      0      0      0      0      1
2      0     1      0      1      0      0      0      0
4      1     7      0      0      0      0      1      0
5      0     5      0      0      0      0      1      0
7      0     2      0      0      0      0      0      1
8      0     5      0      0      0      0      0      0
  make.7 make.8 make.9 insured claims
1      0      0      0  669.34     42
2      0      0      0   55.70      6
4      0      0      0   81.78      3
5      0      0      0  200.68     14
7      0      0      0   64.73      6
8      1      0      0  176.70     10
```

   (a) To make sure that you have done this correctly, **report the dimensions of the new data frame.**

2. Run a principal components analysis on these data, excluding the response variable `per`, using `scale.=TRUE` to standardize the data.

   (a) **How many PC's get chosen** if you use the guideline of choosing all PC's with higher than average contributions to the explained variance?

   (b) Make a scree plot and a cumulative variance plot. **Show these plots.**

   (c) Based on these plots, **indicate how many PC's** you think explain "enough" variability while achieving dimension reduction? **Explain very briefly why you chose that number.**

OZONE DATA Refer Section 5, Problem 3, where you used 10-fold CV for comparing LASSO models

1. Using *the same 10 folds* estimate the MSPE for PLS.

(a) On each training set, run the PLS function, choosing the optimum number of components using 10-fold CV. You can figure out how many components is optimal according to CV using code like this, assuming that I have named my model, `mod.pls`

```
mp.cv = mod.pls$validation
Opt.Comps = which.min(sqrt(mp.cv$PRESS/nrow(ins.dv)))
```

Report the optimal number of components for each of the 10 folds.

(b) Use the `predict()` function to predict responses on the test set. You need to specify `ncomps` for this function.

(c) **Report the separate MSPEs from each fold,** $MSPE_v$, $v = 1, \ldots, 10$ **and the MSPE for the full data.**

(d) **ADD a boxplot for PLS to the boxplots you made for other models. Comment on how well PLS compares to the other models.**

(e) **Remake the plot using relative MSPE and comment.**