Application of Select Regression Methods to Yearly London Housing Data

Tyler Sagendorf

May 11, 2020

Overview

I used the "housing_in_london_yearly_variables.csv" file from Justinas Cirtuatas' "Housing in London" data set on Kaggle. After manipulating the data types of certain variables and standardizing the predictors to remove any influence caused by differences in scales, I split the data into training and testing sets using a 70/30 split. I chose this split because there were only 267 rows of data, and I wanted my models to be accurate, but I did not want them to overfit to the training data.

The purpose of this report was to predict life satisfaction of areas in London based on measurements such as mean salary, the number of jobs, and the size of the area. Since many of the predictors were correlated with one another (see Figure 1), I chose methods that were best suited to dealing with correlation, rather than performing variable selection and fitting reduced models. The regression methods I used were K-Nearest-Neighbors, Elastic Net, Principal Component Regression, and a Random Forest. If applicable, optimal tuning parameters were selected by applying 5-fold cross-validation to the training data. For each of the five methods, I used the final models and the testing data to calculate their associated test mean squared errors (MSEs) to determine which one performed best. The Random Forest with 2 variables randomly sampled as candidates at each split produced the lowest test MSE.

Link to data source:

https://www.kaggle.com/justinas/housing-in-london?select=housing_in_london_yearly_variables.csv

Exploratory Data Analysis

In order to determine if any pairs of predictors are correlated, we can look at a plot of the correlation matrix.

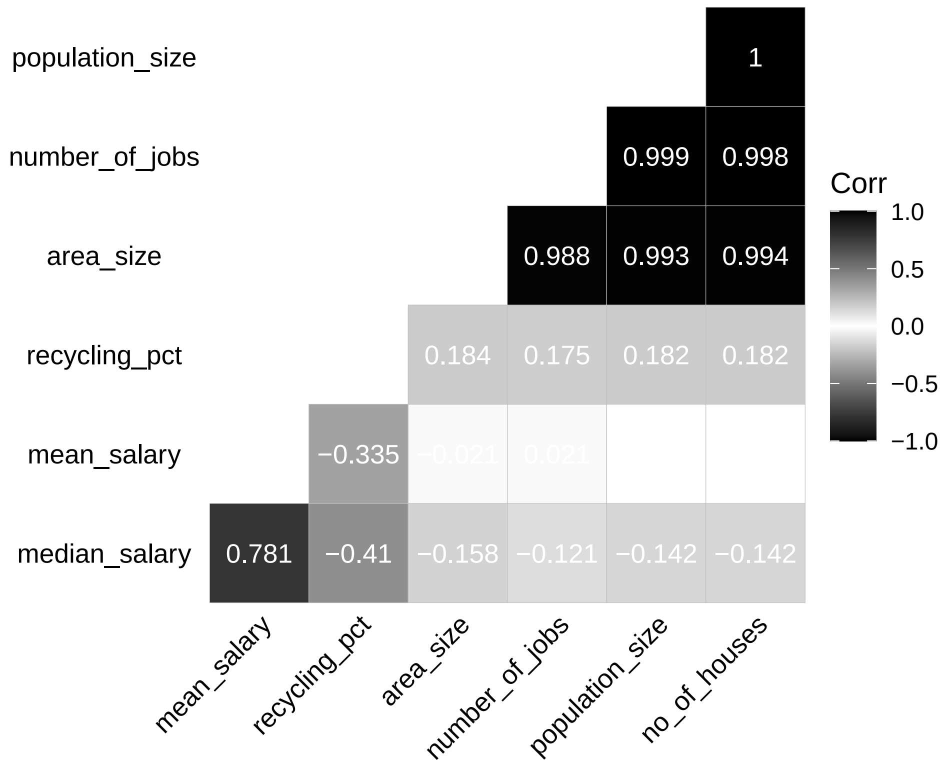

Figure 1: Correlogram for Predictors of Life Satisfaction. A value close to -1 or 1 indicates that there is a strong linear relationship between the two variables. Weaker correlations are less visible, as they are not as important.

Based on Figure 1, the number of houses is very strongly correlated with population size (1.000), the number of jobs (0.998), and area size (0.994); population size is very strongly correlated with the number of jobs (0.999) and area size (0.993); area size is very strongly correlated with the number of jobs (0.988); and mean salary is strongly correlated with median salary (0.781). We have several examples of near-perfect multicollinearity. As mentioned previously, we will mainly focus on methods that can deal with this issue.

K-Nearest-Neighbors Regression

The first method that we will look at is k-Nearest Neighbors Regression, which is a simple, nonparametric method that "first identifies the K training observations that are closest" to the prediction point x_0. Then, it averages the responses for these k-nearest training observations to estimate the response at x_0. It does this for all points in the (training) data set (James et al. 2017, 104-105).

I used 5-fold cross-validation to find the optimal value of k. I only considered values from 1 to 30 because the training data only had 187 rows. The optimal value was determined to be $k = 27$, so I fit a 27-Nearest-Neighbors Regression model to the training data. I used this model with the testing data to predict `life_satisfaction` and then computed the test MSE (0.04328) by taking the average of the squared differences between the predictions and the true values.

It would have been best to apply a dimensionality reduction technique prior to performing KNN regression, since there is multicollinearity in the feature space. I did not do this because I wanted to compare KNN regression on its own to the other methods.

Elastic Net Regression

Elastic Net Regression combines the penalties of both Lasso and Ridge Regression to perform both variable selection and variable shrinkage. I chose Elastic Net over Ridge and Lasso because many of the predictors were highly correlated, and I was unsure how many predictors truly affected the response. Elastic Net Regression groups and shrinks the parameters associated with the correlated variables and leaves them in the equation or removes them all at once, so this seemed like the best choice (Starmer 2018).

Again, I used 5-fold cross-validation to find the optimal values of the parameters. I considered all combinations where `lambda` (the Lasso tuning parameter) and `fraction` (the Ridge tuning parameter) ranged from 0 to 1 in increments of 0.1. I determined that `lambda` = 0 and `fraction` = 0.2 were the values which produced the lowest CV training RMSE. This is the same as a Lasso Regression model. I fit this regression model to the training data, and computed the test MSE (0.04455). The model coefficients for the standardized predictors are shown below in Table 1.

Table 1: Elastic Net Regression Model Coefficients.

	Coeff.
median_salary	0.0351058
mean_salary	-0.0082920
recycling_pct	0.0499640
population_size	0.0000000
number_of_jobs	0.6812295
area_size	0.2769342
no_of_houses	-0.9352829

We can see that one standard unit increase in the number of houses actually decreases life satisfaction by 0.9353 units (holding all other variables constant). Also, one standard unit increase in the number of jobs increases life satisfaction by 0.6812 units (holding all other variables constant). Through standardization, we can see that these are the two variables with the largest effect on life satisfaction. Also note that the coefficient for `population_size` is 0&emdash;indicating that it is not included in the final model.

Principal Component Regression

Principal Component Regression is an extension of Principal Component Analysis, which is a dimensionality reduction technique. Instead of using the original predictors in a regression model, we use the principal components; these are uncorrelated linear combinations of the predictors (useful when there is multicollinearity in the data). The principal components are selected such that the first principal component (PC1) explains most of the variance in the original data, the second principal component (PC2) explains the second most variance, and so on. The total number of possible principal components is the same as the number of original predictors. PCA is an unsupervised learning method, unlike PCR, since we don't include the response variable when determining the principal components (James et al. 2017, 230-238, 375).

The principal components, in this case, were selected to maximize the proportion of variance explained while both reducing the dimensionality of the data and minimizing the 5-fold cross-validation MSE.

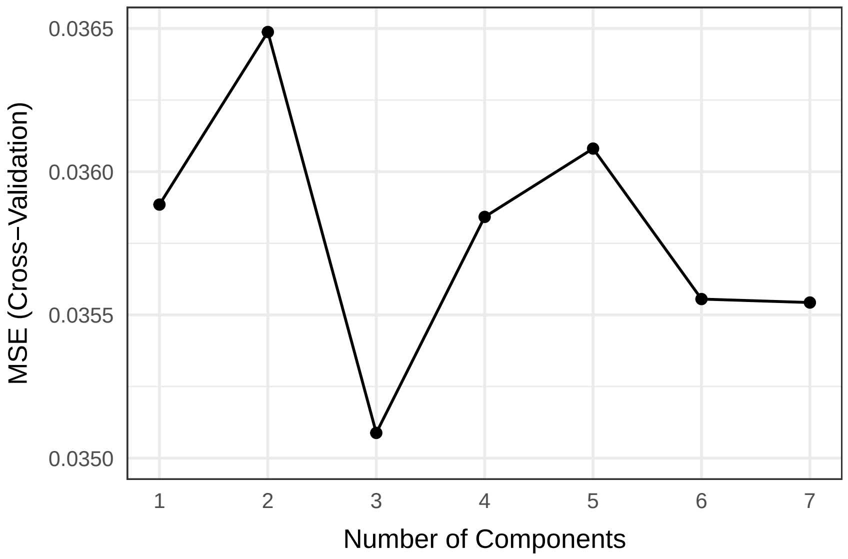

Figure 2: 5-fold cross-validation MSE for each of the 7 principal components in the Yearly Housing in London data set.

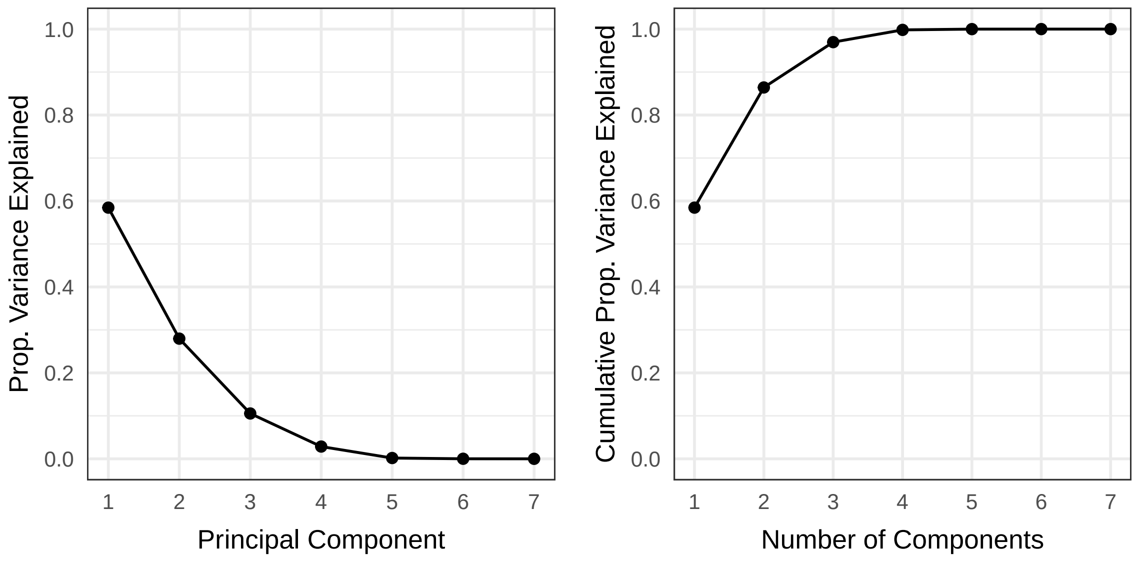

Figure 3: Left: scree plot depicting the proportion of variance explained by each of the 7 principal components in the Yearly Housing in London data set. Right: the cumulative proportion of variance explained by each of the 7 principal components.

From Figure 2, we see that 3 principal components results in the lowest CV MSE. Also, Figure 3 shows that the first 3 principal components explain most of the variance (96.96%; see Table 4 in the Appendix), and the addition of the fourth component provides less than a 3% increase in variance explained (Table 3 in the Appendix). The use of 4 principal components also results in a slightly higher CV MSE than just 3 principal components. Therefore, I used a PCR model with 3 principal components (see Table 5 in the Appendix) and the testing data to predict `life_satisfaction`. Then, I calculated the test MSE (0.04577).

Random Forest Regression

A Random Forest consists of many decision trees created with the stipulation that the variables that can be considered as candidates for each split are randomly selected from the list of all possible predictors. The number of variables that can be selected at each split is denoted as m, and is the value for `mtry` in the `randomForest` function. I decided to build a Random Forest for the same reason that I chose the last two methods (multicollinearity), since we know that "using a small value of m in building a random forest will typically be helpful when we have a large number of correlated predictors" (James et al. 2017, 320).

I created Random Forests using different values of m and computed their associated out-of-bag MSE, which is similar to performing cross-validation (James et al. 2017, 317-319). The optimal value of m was found to be 2. I constructed a Random Forest consisting of 500 regression trees with `mtry`=2 from the training data and predicted values of `life_satisfaction` using the testing data. These predictions are calculated by averaging the estimates from all 500 regression trees (Drakos 2019). From there, I calculated the test MSE (0.03255).

Model Comparison

Table 2: Comparison of Methods

Method	Test MSE
Random Forest	0.0325511
KNN	0.0432837
Elastic Net	0.0445454
PCR	0.0457723

Table 2 displays the name and associated test MSE for each method that was considered. They are sorted in ascending order by MSE. We can see that the best method was Random Forest Regression with `mtry` $= 2$, as it produced the smallest test MSE (0.03255). Surprisingly, the KNN Regression model outperformed both Elastic Net Regression and PCR. I think that the Random Forest performed so well because of how it manages the bias-variance tradeoff. Even though individual Regression Trees have "extremely low bias" and high variance because "they maximally overfit to the training data," creating hundreds of these trees from bootstrapped samples of the training data (while only considering a subset of the total number of predictors at each split) and averaging their estimates dramatically reduces the overall variance while still keeping the bias relatively low (Ramchandani 2018). This means that Random Forests can perform well on new data sets, as they tend to capture the true relationship between the predictors and the response without also capturing the random noise (Drakos 2019).

References

Drakos, Georgios. 2019. "Random Forest Regressor Explained in Depth." GDCoder. https://gdcoder.com/random-forest-regressor-explained-in-depth/.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2017. *An Introduction to Statistical Learning: With Applications in R*. Corrected at 8th Printing 2017. Springer. http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf.

Ramchandani, Prratek. 2018. "Random Forests and the Bias-Variance Tradeoff." Towards Data Science. https://towardsdatascience.com/random-forests-and-the-bias-variance-tradeoff-3b77fee339b4.

Starmer, Joshua. 2018. "Regularization Part 3: Elastic Net Regression." YouTube. Video, 5:18. StatQuest with Josh Starmer. https://www.youtube.com/watch?v=1dKRdX9bfIo.

Appendix

PCR

PCA of the standardized predictors.

Table 3: Prop. of Variance Explained.

	Prop. Variance
Comp 1	0.5845274
Comp 2	0.2796337
Comp 3	0.1054722
Comp 4	0.0284910
Comp 5	0.0018067
Comp 6	0.0000682
Comp 7	0.0000008

Table 4: Cumulative Prop. of Variance Explained.

	Variance
Comp 1	0.5845274
Comp 2	0.8641612
Comp 3	0.9696333
Comp 4	0.9981243
Comp 5	0.9999310
Comp 6	0.9999992
Comp 7	1.0000000

Table 5: Principal Components

	PC1	PC2	PC3
median_salary	0.1156559	-0.6182411	0.2777001
mean_salary	0.0427454	-0.6287939	0.3678935
recycling_pct	-0.1392087	0.4421628	0.8839832
population_size	-0.4915758	-0.0811467	-0.0370646
number_of_jobs	-0.4928006	-0.0960738	-0.0281957
area_size	-0.4890316	-0.0674476	-0.0501436
no_of_houses	-0.4916975	-0.0807716	-0.0377834

Random Forest

Forest Overview.

```
##
## Call:
##  randomForest(formula = life_satisfaction ~ ., data = train_data,      mtry = m)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.02591619
##                     % Var explained: 29.88
```

R Code

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE,
                      fig.pos = 'H', fig.align = 'center', out.extra = '')
# Reproducibility
set.seed(05112020)

# Required packages
library(caret)
library(elasticnet)
library(ggcorrplot)
library(ggfortify)
library(ggplot2)
library(ggthemes)
library(kableExtra)
library(pls)
library(randomForest)
library(tidyverse)

# Read the data
housing <- read.csv("housing_in_london_yearly_variables.csv")

# Data processing
housing <- housing %>%
  mutate(mean_salary = as.numeric(mean_salary),
         recycling_pct = as.numeric(recycling_pct)) %>%
  na.omit(life_satisfaction) %>% # Remove NA's
  dplyr::select(-code, -area, -date, -borough_flag)

# Separate response variable from predictors
Y <- housing %>%
  dplyr::select(life_satisfaction)
```

```r
# Standardize predictors only
X <- housing %>%
  dplyr::select(-life_satisfaction) %>%
  scale()

# Recombine response and predictors into one data set
Z <- cbind(Y,X)

# Separate into training and testing data sets (70/30) ---
# Training set
train_index <- sort(sample(1:nrow(Z),
                           size = round(0.7*nrow(Z))))
train_data <- Z[train_index, ]

# Testing set
test_data <- Z[-train_index, ]

# Are the predictors correlated? ---
# Correlation matrix
cor.matrix <- cor(train_data[ ,seq(2,8)])

# Correlogram
ggcorrplot(cor.matrix, type = "lower", show.diag = FALSE,
           lab = TRUE, lab_col = "white",  digits = 3,
           hc.order = TRUE, lab_size = 3.5, ggtheme = theme_void(),
           tl.cex = 10) +
  scale_fill_gradient2(name = "Corr", low = "black", mid = "white",
                       high = "black", midpoint = 0, limits = c(-1,1)) +
  theme(plot.title.position = "plot")

# --- K-Nearest-Neighbors Regression ---
# Optimize k
knn_optim <- train(life_satisfaction ~ .,
                   data=train_data,
                   method="knn",
                   trControl=trainControl(method="cv", 5),
                   tuneGrid=data.frame(k = seq(1,30)))

# Optimal value of k
k <- knn_optim$bestTune # k=27

# Final Model
knn.mod <- train(life_satisfaction ~ .,
                 data=train_data,
                 method="knn",
                 tuneGrid=data.frame(k = k))

# Predictions
knn_pred <- predict(knn.mod, test_data)

# Test MSE
knn_mse <- mean((test_data$life_satisfaction - knn_pred)^2) # 0.04328368

# --- Elastic Net Regression ---
# Optimize tuning parameters
enet_optim <- train(life_satisfaction ~ .,
                   data=train_data,
                   method="enet",
                   trControl=trainControl(method="cv", 5),
                   tuneGrid=expand.grid(lambda = seq(0,1,0.1),
                                        fraction = seq(0,1,0.1)))

# Optimal values of fraction and lambda
fraction <- enet_optim$bestTune$fraction # fraction = 0.2
lambda <- enet_optim$bestTune$lambda # lambda = 0

# Final Model
enet.mod <- train(life_satisfaction ~ .,
                  data=train_data,
                  method="enet",
                  tuneGrid = data.frame(fraction = fraction,
                                        lambda = lambda))

# Predictions
enet_pred <- predict(enet.mod, test_data)

# Test MSE
enet_mse <- mean((test_data$life_satisfaction - enet_pred)^2) # 0.04454541

# Final model coefficients
enet_coeff <- enet.mod$finalModel$beta.pure[7,]

kable(enet_coeff, format="latex", col.names = "Coeff.",
  caption = "Elastic Net Regression Model Coefficients.") %>%
  kable_styling(latex_options = "HOLD_position")

# --- Principal Component Regression ---
# Optimize number of principal components
```

```r
pcr_optim <- train(life_satisfaction ~ ., data = train_data,
                   method="pcr",
                   trControl = trainControl(method="cv", 5),
                   tuneGrid=data.frame(ncomp=1:7))

# CV MSE vs Number of Components
ggplot(aes(x=ncomp, y=RMSE^2), data=pcr_optim$results) +
  geom_line() +
  geom_point() +
  scale_y_continuous(name="MSE (Cross-Validation)",
                     limits = c(0.0350, 0.0365),
                     breaks = seq(0.0350, 0.0365, 0.0005),
                     labels = scales::unit_format(accuracy = 0.0001,
                                                  suffix = "")) +
  scale_x_continuous(name="Number of Components",
                   limits = c(1,7),
                   breaks = seq(1,7),
                   labels = seq(1,7)) +
  theme_bw() +
  theme(axis.title.x = element_text(size = 10,
                                    margin = ggplot2::margin(t=6)),
        axis.title.y = element_text(size = 10,
                                    margin = ggplot2::margin(r=6)),
        axis.text = element_text(size=8),
        axis.ticks = element_blank(),
        panel.grid.minor.x = element_blank())

# Scree plot ---
pcr.mod2 <- train(life_satisfaction ~ .,
                 data = train_data,
                 method = "pcr",
                 tuneGrid=data.frame(ncomp=7))

# Variance in the Predictors Explained by Each Component
var.comp <- pcr.mod2$finalModel$Xvar/pcr.mod2$finalModel$Xtotvar

# Plot 1
v1 <- ggplot() +
  geom_line(aes(x=1:7, as.numeric(var.comp))) +
  geom_point(aes(x=1:7, as.numeric(var.comp))) +
  scale_x_continuous(name = "Principal Component",
                     limits=c(1,7),
                     breaks=seq(1,7),
                     labels = seq(1,7)) +
  scale_y_continuous(name = "Prop. Variance Explained",
                     limits = c(0,1),
                     breaks = seq(0,1,.2)) +
  theme_bw() +
  theme(axis.title.x = element_text(size = 10,
                                    margin = ggplot2::margin(t=6)),
        axis.title.y = element_text(size = 10,
                                    margin = ggplot2::margin(r=4)),
        axis.text = element_text(size=8),
        axis.ticks = element_blank(),
        panel.grid.minor.x = element_blank())

prop.var <- pcr.mod2$finalModel$Xvar/pcr.mod2$finalModel$Xtotvar

# Cumulative Variance Explained
tot.var.comp <- cumsum(prop.var)

# Plot 2
v2 <- ggplot() +
  geom_line(aes(x=1:7, y=as.numeric(tot.var.comp))) +
  geom_point(aes(x=1:7, y=as.numeric(tot.var.comp))) +
  scale_x_continuous(name = "Number of Components",
                     limits=c(1,7),
                     breaks=seq(1,7),
                     labels = seq(1,7)) +
  scale_y_continuous(name = "Cumulative Prop. Variance Explained",
                     limits = c(0,1),
                     breaks = seq(0,1,0.2)) +
  theme_bw() +
  theme(axis.title.x = element_text(size = 10,
                                    margin = ggplot2::margin(t=6)),
        axis.title.y = element_text(size = 10,
                                    margin = ggplot2::margin(r=4, l=4)),
        axis.text = element_text(size=8),
        axis.ticks = element_blank(),
        panel.grid.minor.x = element_blank())

# Side-by-side variance plots
gridExtra::grid.arrange(v1, v2, nrow=1)


# Final Model
pcr.mod <- train(life_satisfaction ~ .,
                 data = train_data,
                 method = "pcr",
                 tuneGrid=data.frame(ncomp=3))

# Predictions
pcr_pred <- predict(pcr.mod, test_data)

# Test MSE
pcr_mse <- mean((test_data$life_satisfaction - pcr_pred)^2) # 0.04577228

# --- Random Forest Regression ---
# Empty vector for test MSE of each model
rf_mse <- rep(0,7)

# p random forests,
# where p is the number of predictors
for (p in 1:7) {
    mod <- randomForest(life_satisfaction ~ .,
                        data = train_data, mtry = p)
    # Compute Out-of-Bag MSE
    rf_mse[p] <- mean((mod$predicted - train_data$life_satisfaction)^2)
}

# Optimal value of mtry
m <- which.min(rf_mse) # m = 2

# Final Model ---
forest.mod <- randomForest(life_satisfaction ~ .,
                           data = train_data, mtry = m)

# Predictions ---
forest_pred <- predict(forest.mod, test_data)

# Test MSE ---
ft_mse <- mean((test_data$life_satisfaction - forest_pred)^2) # 0.03255107

# Models and Test MSEs
compare.mods <- data.frame(method = c("KNN",
                                      "Elastic Net",
                                      "Random Forest",
                                      "PCR"),
                           test.MSE = c(knn_mse, enet_mse,
                                        ft_mse, pcr_mse))

# Ascending order by test MSE
```

```r
compare.mods <- compare.mods %>%
  arrange(test.MSE)

# Nice table
kable(compare.mods, format="latex",
      col.names = c("Method", "Test MSE"),
      align = c('l', 'c'),
      caption = "Comparison of Methods") %>%
  kable_styling(latex_options = "HOLD_position")

# Variance in predictors explained by each principal component
kable(var.comp, format="latex", col.names = "Prop. Variance",
  caption = "Prop. of Variance Explained.") %>%
  kable_styling(latex_options = "HOLD_position")

# Cumulative variance
kable(tot.var.comp, format="latex", col.names = "Variance",
      caption = "Cumulative Prop. of Variance Explained.") %>%
  kable_styling(latex_options = "HOLD_position")

# Principal Components
pca.comp <- pcr.mod$finalModel$projection
kable(pca.comp, format="latex", col.names = c("PC1", "PC2", "PC3"),
  caption = "Principal Components") %>%
  kable_styling(latex_options = "HOLD_position")

# Forest overview
forest.mod
```