Data Mining and Statistical Learning

Homework 2

Introduction

In multiple linear regression, there are many models/methods that are able to fit a training data well. However, selecting the best model is a decision that is usually made based on several criteria (**Predictive ability**, **Simplicity**, and **Analytic measure**). For this homework, several model and variable selection methods will be explored and applied to the famous **fat** dataset from the **faraway** library in R.

The dataset has **252** observations and **18** variables which include Age, weight, height, and 10 body circumference measurements are recorded for **252 men**. Each man's percentage of body fat was accurately estimated by an underwater weighing technique.

The **18** variables are described below:

- **brozek**: Percent body fat using Brozek's equation, 457/Density - 414.2
- **siri**: Percent body fat using Siri's equation, 495/Density - 450
- **density**: Density (gm/cm^3)
- **age**: Age (yrs)
- **weight**: Weight (lbs)
- **height**: Height (inches)
- **adipos**: Adiposity index = Weight/Height^2 (kg/m^2)
- **free**: Fat Free Weight = (1 - fraction of body fat) * Weight, using Brozek's formula (lbs)
- **neck**: Neck circumference (cm)
- **chest**: Chest circumference (cm)
- **abdom**: Abdomen circumference (cm) at the umbilicus and level with the iliac crest
- **hip**: Hip circumference (cm)

- **thigh**: Thigh circumference (cm)
- **knee**: Knee circumference (cm)
- **ankle**: Ankle circumference (cm)
- **biceps**: Extended biceps circumference (cm)
- **forearm**: Forearm circumference (cm)
- **wrist**: Wrist circumference (cm) distal to the styloid processes

Problem and Data Description

The purpose of this assignment is to get a better understanding of linear regression. We will be working with **fat1train** as the training set and **fat1test** as the testing set. The training set consists of **227** observations, while the testing set is roughly **25** observations ($\approx 10\%$ of the overall data set). The response variable is stored in the first column (**brozek**) and it is described as the percentage of body fat using Brozek's equation $\left(\frac{457}{\rho} - 414.2\right)$ where ρ is the person's average density (i.e. total mass divided by total volume). The other 17 variables as potential predictors.

The objective of this analysis is to predict **brozek** using the other 17 potential predictors. For that purpose, we will use several different statistical methods. Since our data is relatively small, Cross-Validation will be utilized to assess the robustness of each method.

Exploratory Data Analysis

For any real dataset, we first need to conduct empirical data analysis to have a better understanding of the data. A useful command in R is the summary function. It shows the minimum, 25th quantile, median, mean, 75th quantile, and maximum values of each variable in the training dataset. In the following summary table,

Table 1: Summary statistics 1

brozek	siri	density	age	weight	height
Min. : 0.00	Min. : 0.00	Min. :0.995	Min. :22.00	Min. :118.5	Min. :29.50
1st Qu.:12.80	1st Qu.:12.47	1st Qu.:1.041	1st Qu.:35.75	1st Qu.:159.0	1st Qu.:68.25
Median :19.00	Median :19.20	Median :1.055	Median :43.00	Median :176.5	Median :70.00
Mean :18.94	Mean :19.15	Mean :1.056	Mean :44.88	Mean :178.9	Mean :70.15
3rd Qu.:24.60	3rd Qu.:25.30	3rd Qu.:1.070	3rd Qu.:54.00	3rd Qu.:197.0	3rd Qu.:72.25
Max. :45.10	Max. :47.50	Max. :1.109	Max. :81.00	Max. :363.1	Max. :77.75

Table 2: Summary statistics 2

adipos	free	neck	chest	abdom	hip
Min. :18.10	Min. :105.9	Min. :31.10	Min. : 79.30	Min. : 69.40	Min. : 85.0
1st Qu.:23.10	1st Qu.:131.3	1st Qu.:36.40	1st Qu.: 94.35	1st Qu.: 84.58	1st Qu.: 95.5
Median :25.05	Median :141.6	Median :38.00	Median : 99.65	Median : 90.95	Median : 99.3
Mean :25.44	Mean :143.7	Mean :37.99	Mean :100.82	Mean : 92.56	Mean : 99.9
3rd Qu.:27.32	3rd Qu.:153.9	3rd Qu.:39.42	3rd Qu.:105.38	3rd Qu.: 99.33	3rd Qu.:103.5
Max. :48.90	Max. :240.5	Max. :51.20	Max. :136.20	Max. :148.10	Max. :147.7

Table 3: Summary statistics 3

thigh	knee	ankle	biceps	forearm	wrist
Min. :47.20	Min. :33.00	Min. :19.1	Min. :24.80	Min. :21.00	Min. :15.80
1st Qu.:56.00	1st Qu.:36.98	1st Qu.:22.0	1st Qu.:30.20	1st Qu.:27.30	1st Qu.:17.60
Median :59.00	Median :38.50	Median :22.8	Median :32.05	Median :28.70	Median :18.30
Mean :59.41	Mean :38.59	Mean :23.1	Mean :32.27	Mean :28.66	Mean :18.23
3rd Qu.:62.35	3rd Qu.:39.92	3rd Qu.:24.0	3rd Qu.:34.33	3rd Qu.:30.00	3rd Qu.:18.80
Max. :87.30	Max. :49.10	Max. :33.9	Max. :45.00	Max. :34.90	Max. :21.40

Another integral part of EDA is the histogram plot. Histograms are good for showing general distributional features of the variables.

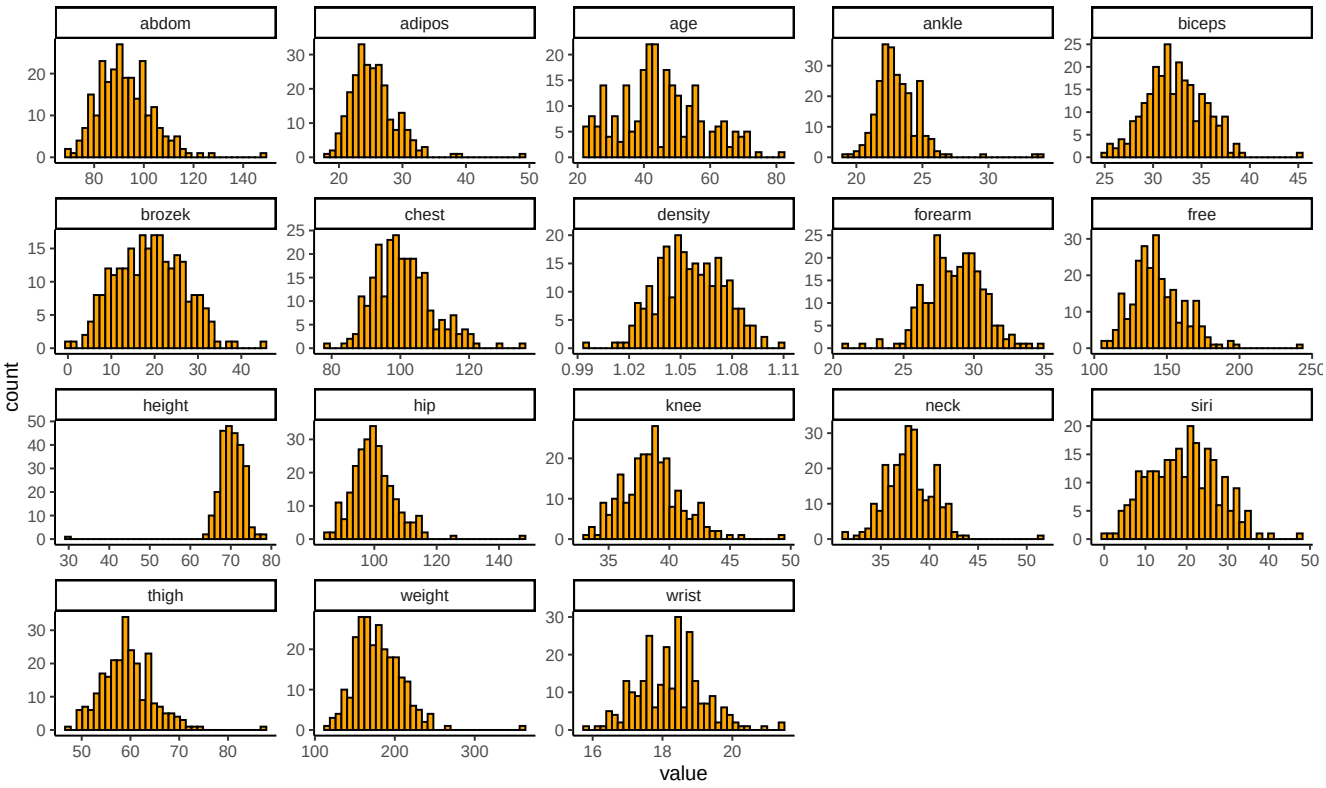


Figure 1: Histogram of each column in the fat dataset

We can see roughly where the peaks of the distribution are, whether the distribution is skewed or symmetric, and if there are any outliers.

To have a better understanding of which pairs of predictors are positively or negatively correlated locally, we will look into a local cross correlation. The local cross correlation is part of the Ranked Cross Correlation package. The chart below represents the local cross correlation of the maximum values per category.

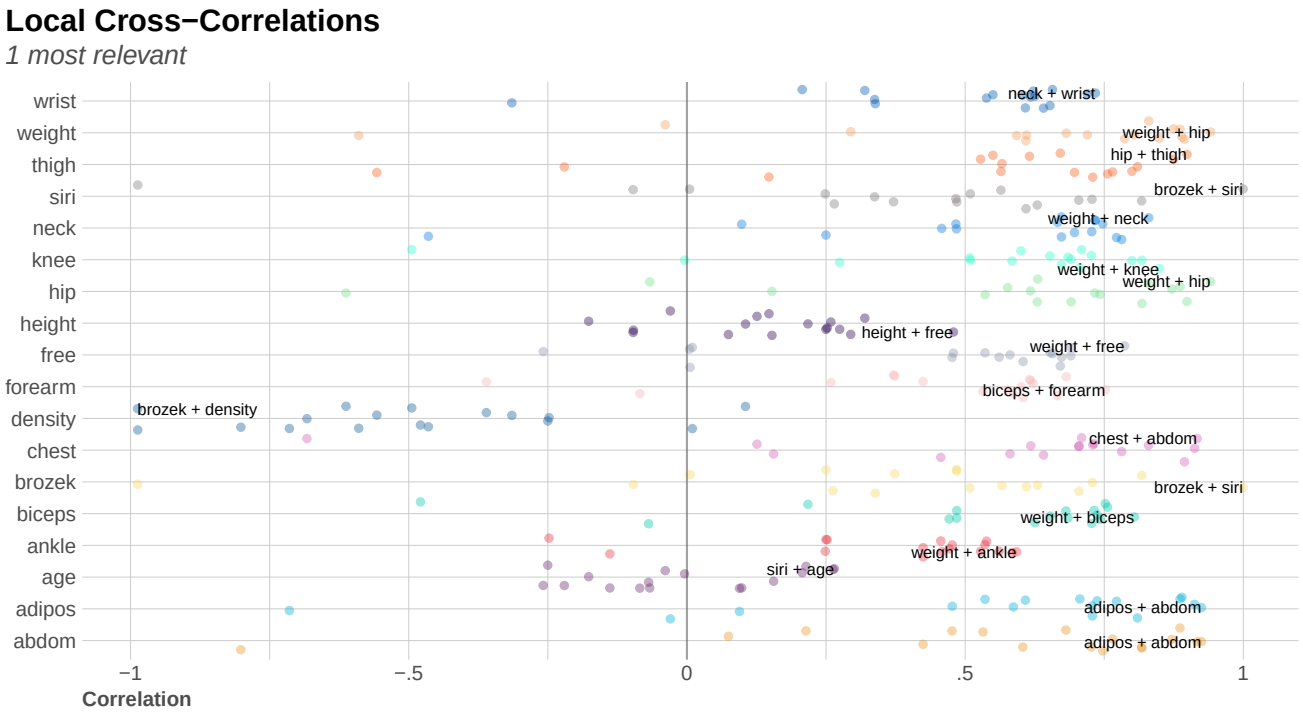


Figure 2: Local Cross-Correlation max values per category

As we can see, several variables are either positively or negatively correlated. We have a strong negative correlation between **density** and **brozek** as well as **density** and **siri**. There is lot of multicollinearity among the predictors. Here, the brozek + density have strongest negative correlation, while most of the other pair of variables have a somewhat strong to very strong positive correlation. This makes this dataset is a great candidate for variable selection.

Even though, we noticed several features with potential outliers from the distribution plots (histograms). It was somewhat tricky to make comparisons between histograms. The following chart shows the box plot of each column in the training data.

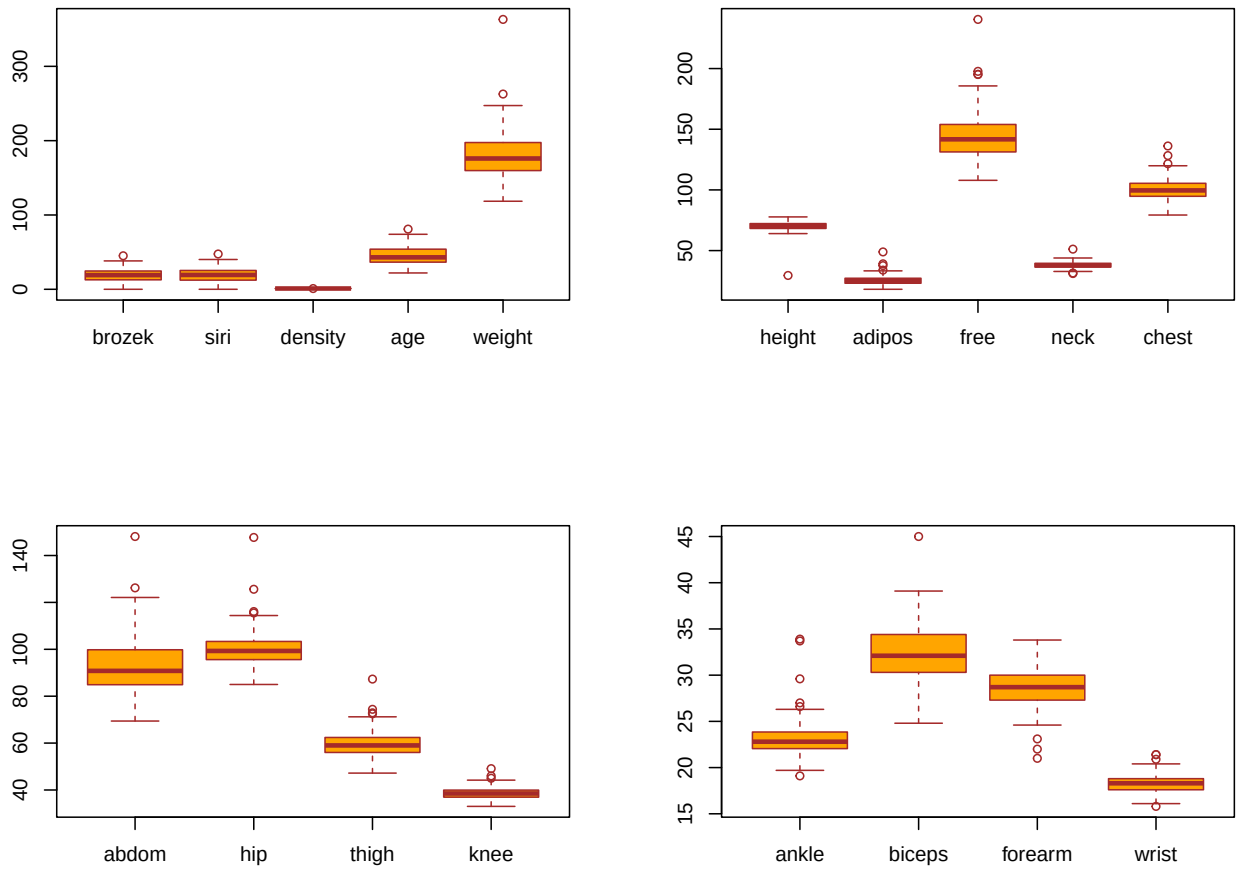


Figure 3: Box plot of each subset of features

We can clearly notice, the training data has a lot of data points that could be outliers. We see noticeable outlier data points in the following predictors: weight, free, abdom, hip, ankle and bicepts. Without a further investigation, we really can't say if those outliers are actual data or some data entry mistakes.

For the purpose of this assignment, we assume that these data points are real data and therefore they will be kept.

Methodology

As previously mentioned, we will be building several regression models then use these models to predict response variable in the testing set.

To predict **brozek**, we will compare the results from Linear regression with all predictors, Linear regression with the best subset of k = 5 predictors variables, Linear regression with variables (step-wise) selected using AIC, Ridge regression, LASSO regression, Principal component regression, and Partial least squares regression.

We started by fitting a linear regression model on all of the available features in the training data set. To calculate the training errors, we simply took the mean squared of the residuals from the model. For testing errors, we used the predict function to get the predicted values, then took the mean of the difference of that and the true value squared as the testing MSEs.

To fit Linear regression with the best subset of k = 5 predictors, **regsubsets()** was used to select the best five variables from the training data. **regsubsets()** performed an exhaustive search algorithm on **100** subsets of data to select five the best variables. To auto-fit the best five variables to the data, a new regression model was built. Similar to the full model, training and testing errors were calculated to assess model performance.

To fit a Linear regression with variables (step-wise) selected using AIC, **step()** R function was used which performs a step-wise variable selection. The method is a combination of forward and backward selection. At each step, the model can either add or delete an existing variable. Then chooses the best options among all possible options. At each step, the model's performance is measured using the Information Criterion (AIC). The best model is the one that has the smallest AIC score. Similar to previous models, the training and testing errors were calculated to assess model performance.

Given how collinear the data features are, Shrinkage methods mainly Ridge and Lasso regression were explored as well. To fit the Ridge Regression model, **lm.ridge** was used from MASS package. To fine tune the ridge regression parameter λ, a generalized cross validation was used on a grid of values from 0 to 100 with an increment of 0.001. then, the corresponding coefficients with respect to the optimal λ index was extracted. However, these coefficients are for the scaled/normalized data instead of original raw data. We had to transfer the data to the original data $Y = X\beta + \epsilon$, and find the estimated β value for this "optimal" Ridge Regression Model. For the estimated β, we needed to separate β_0 (i.e intercept) from the other β's. We then got $\hat{y} = X\beta + \beta_0$. After that, the training errors and testing errors were calculated to assess model performance.

For Lasso regression, **lars** package was used and followed by a similar process to calculate the training and testing errors.

To fit Principal component regression, we first needed to extract the top principal components from the data using principal component analysis. Principal component analysis is an integral part of dimensionality reduction

in machine learning. The main idea is to find the linear combination of X variables that express as much of the variability in response variable as possible. If the most variations of the predictors come out from the first few principal components, then there is enough to use them to build models. Using the training data, we can visualize the variance explained by each principal component by plotting the scree plot (plot the eigenvalues) using **fviz_eig** R function.

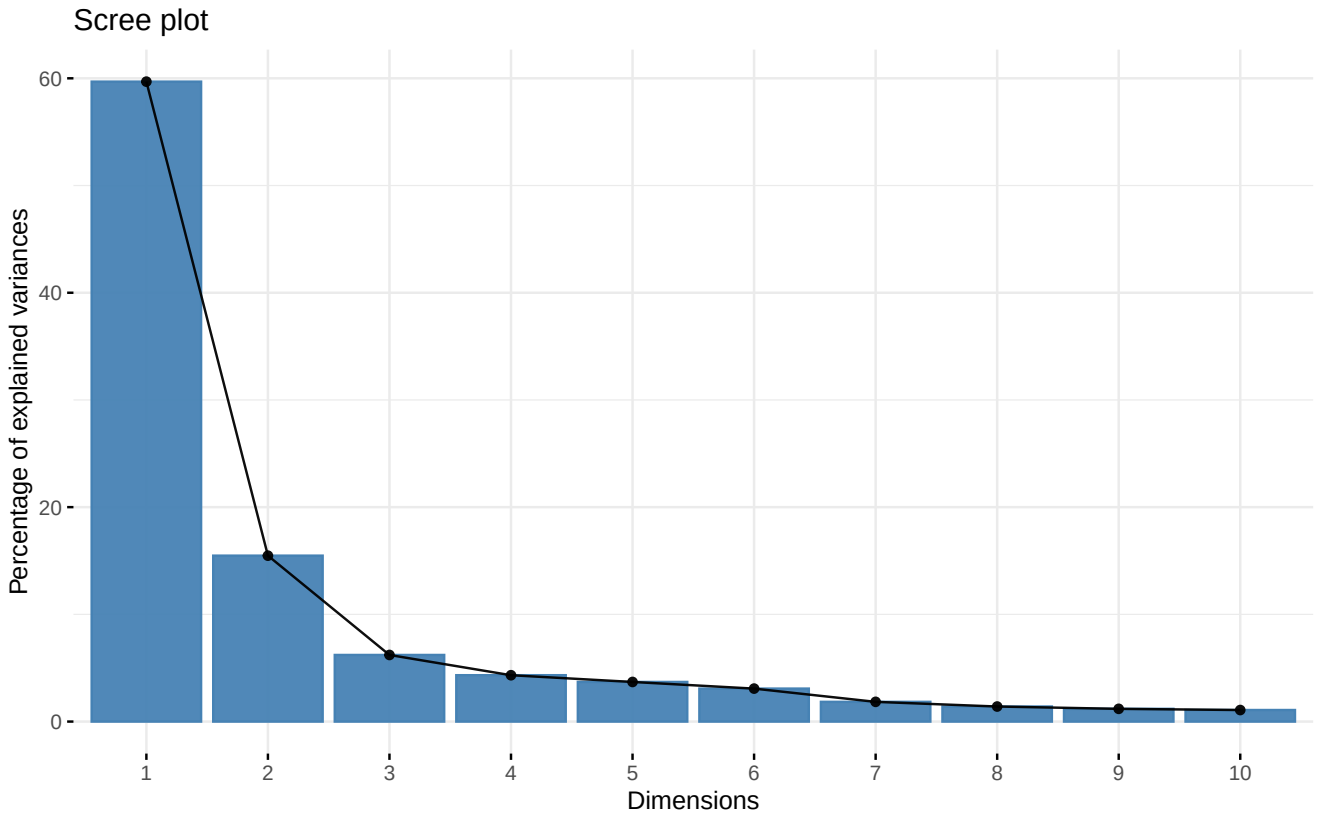


Figure 4: The explained variance of each principal component

We clearly notice from the scree plot that the first principal component has the highest explained variance (60%). To have a better understanding of which variables are contributing to the explained variance, we can plot the PCA variables using **fviz_pca_var** R function (see below).

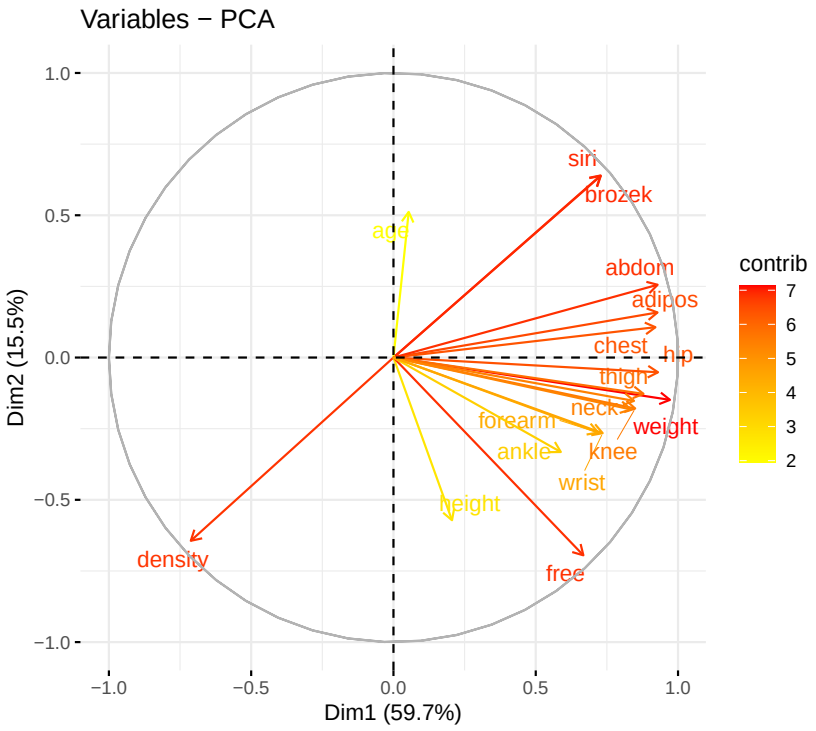


Figure 5: Variables that contributed the most to the explained variance

From the figure above, we can see the variables that contributed the most to the top six principal components.

We can technically use the first six principal components to build a linear regression model. However, in practice, one must choose the number of Principal components carefully. It is always recommended to use a validation set or cross validation.

For our analysis, we used the **pcr()** function in *pls* R package with cross validation to build a principal component regression model. Once we got the number of components, we then calculated the training and testing errors using the optimal choice of PCs (17)

For Partial Least Squares (PLS) Regression, **plsr** function in *pls* R package was used. Similar to pcr, the training and testing errors were calculated using using the optimal choice of PCs (17).

To decide which method produced the best results, Monte Carlo Cross-Validation was used to compare the average testing errors of each of the models. Monte Carlo Cross Validation algorithm splits the observed data points into training and testing subsets, and repeats the above computation for $B = 100$ and $B = 800$ times for comparison. For the purpose of this homework, we are going to combine $n_1 = 227$ training data and $n_2 = 25$ testing data together into a larger data set. Then for each loop $b = 1, \ldots, B$ we will randomly select $n_1 = 227$ as a new training subset and use the remaining $n_2 = 25$ data as the new testing subset. Within each loop, we first build different models from the training data of that specific loop and then evaluate their performances on the corresponding testing data. Therefore, for each model, we will obtain B values of the testing errors on B different subsets of testing data, denote by TE_b for $b = 1, \ldots, B$. Then **the average** performances of each model will be summarized by the sample mean

and sample variances of these B values

Results

The training and testing mean squared errors for each model are as follows

Table 4: MSE errors for all seven models

	mod1	mod2	mod3	mod4	mod5	mod6	mod7
MSEtrain	0.02931	0.03147	0.02946	0.02931	0.03086	0.02931	0.02931
MSEtest	0.00876	0.00279	0.00896	0.00886	0.00316	0.00876	0.00876

While all seven models performed fairly similarly on training data, we notice that the testing errors are much much smaller than what one would expect (compared to training errors). Model 2 and model 5 (best 5 model and Lasso regression respectively) have performed the best on the test data set.

The results from the Monte Carlo Cross-Validation with $B = 100$ are as follows:

Table 5: Sample mean/variance of MCCV Errors for B = 100

	mod1	mod2	mod3	mod4	mod5	mod6	mod7
Mean	0.07435	0.06125	0.02402	0.07637	0.06267	0.07058	0.07251
Variance	0.06979	0.04807	0.00090	0.07784	0.05833	0.06995	0.06986

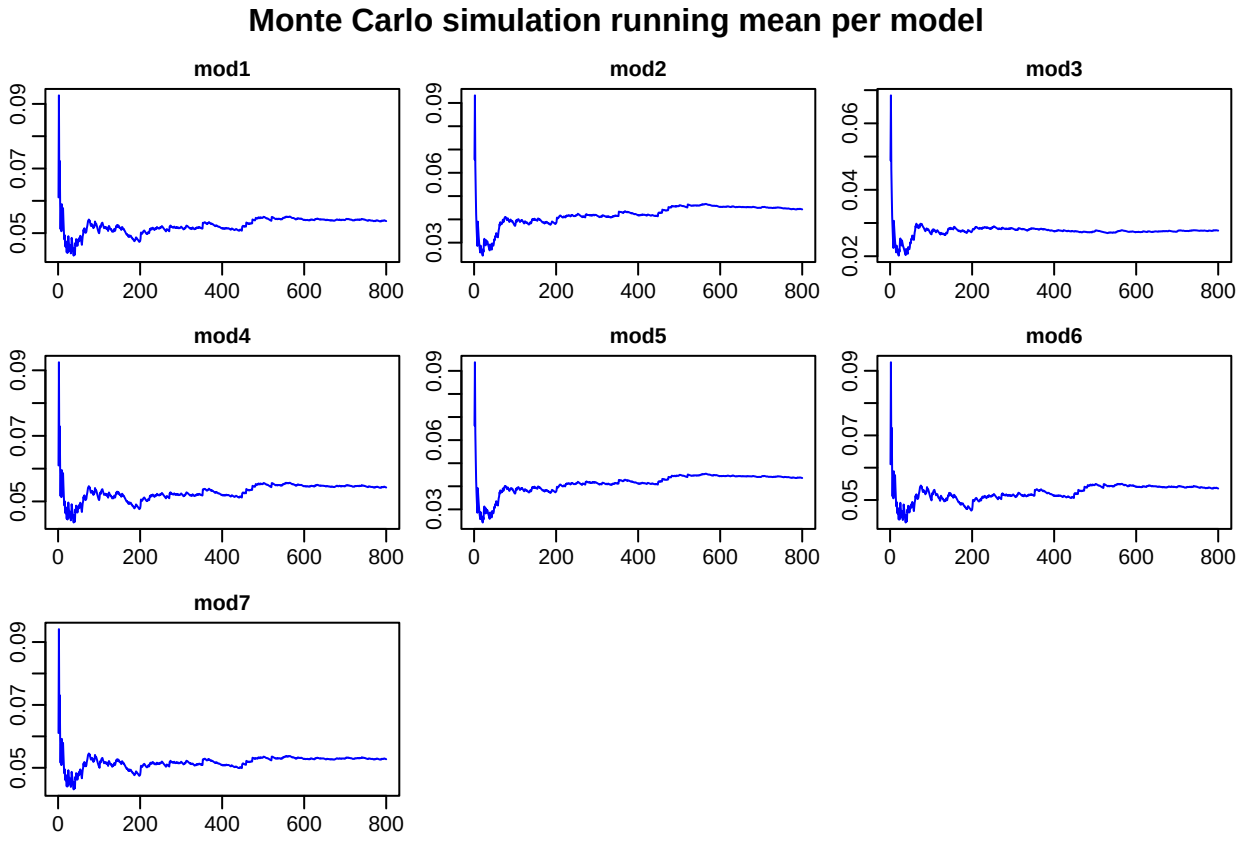
Unlike the initial testing, the models performed similarly except model 3 (step-wise regression) which performed much better. The step-wise (model 3) regression model had the lowest error rate, followed by the linear regression model with best five predictors. The variances were all low except for model 3 which had the smallest variance.
To determine the expected error that each model is converging to, I increased the Monte Carlo run to $B = 800$

Table 6: Sample mean/variance of MCCV Errors for B = 800

	mod1	mod2	mod3	mod4	mod5	mod6	mod7
Mean	0.05372	0.04429	0.02771	0.05424	0.04359	0.05355	0.05272
Variance	0.00539	0.00524	0.00103	0.00560	0.00450	0.00550	0.00534

After running Monte Carlo cross validation for $B = 800$ iteration, we conclude that model 3 (step-wise regression) is by far the best model based on the mean and variance of testing errors, followed by model 2 (best five model) and model 5 (LASSO)

From the running mean plots below, it is evident that each model's testing error is converging to its expected prediction error



Conclusion

In this homework, we explored different regression methods including methods that are used for variable selection as well as methods used for shrinkage/regularization. Each model had an accuracy rate of $\approx 97\%$ on training data and an extremely high accuracy $\approx 99\%$ on testing data. The low testing errors were not expected and probably caused by the small testing data ("25 carefully selected rows"). Unlike the initial training/testing results, Monte Carlo cross validation models had an accuracy rate on testing data ranging from $\approx 93\%$ up to $\approx 97\%$. Step-wise linear regression was by far the best model after cross validation (for $B = 100$ and $B = 800$).

Reference

- HW2 sample code, HW1 code (own code), and Supplemental R code from week 3 module

Appendix

R Code

```
# Settings
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, fig.pos = 'H')


# Check if packages are installed. If not, install them.
install.packages <- function(pkg) {
    new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
    if (length(new.pkg)) {
        install.packages(new.pkg)
    }
}


packages <- c("ggplot2", "mcmcplots", "knitr","xtable", "kableExtra",
              "reshape2", "devtools","MASS","glmnet", "lares", "caret",
              "splines", "class", "dplyr", "PerformanceAnalytics",  "factoextra")


install.packages(packages)
library(mcmcplots)
library(knitr)
library(kableExtra)
library(ggplot2)
library(reshape2)
library(devtools)
library(lares)
library(caret)
library(splines)
library(class)
library(extrafont)
library(xtable)
library(dplyr)
library(leaps)
library(MASS)
```

```r
library(lars)
library(pls)
library(glmnet)
library(tidyverse)
library(gtsummary)
library(tidyr)
library(PerformanceAnalytics)
library(corrplot)
library(factoextra)
options(kableExtra.latex.load_packages = FALSE)
loadfonts()

devtools::install_github("haozhu233/kableExtra")
# Reading the data
fat <- read.table(file = "fat.csv", sep=",", header=TRUE);

### Split the data as in Part (a)
n = dim(fat)[1]; ### total number of observations
n1 = round(n/10); ### number of observations randomly selected for testing data

## To fix our ideas, let the following 25 rows of data as the testing subset:
flag0 = c(1, 21, 22, 57, 70, 88, 91, 94, 121, 127, 149, 151, 159, 162,
164, 177, 179, 194, 206, 214, 215, 221, 240, 241, 243);
fat1train = fat[-flag0,];
fat1test = fat[flag0,];

knitr::kable(summary(fat[1:6]),"pipe", digit=3,
             caption = paste("Summary statistics 1"))%>%
          kable_styling(position = "center", full_width = F)

knitr::kable(summary(fat[7:12]),"pipe", digit=3,
             caption = paste("Summary statistics 2"))%>%
          kable_styling(position = "center", full_width = F)
```

```
knitr::kable(summary(fat[13:18]),"pipe", digit=3,
             caption = paste("Summary statistics 3"))%>%
          kable_styling(position = "center", full_width = F)
ggplot(gather(fat), aes(value)) +
    geom_histogram(bins = 35, fill = 'orange', color='black') +
    facet_wrap(~key, scales = 'free') +
    theme_classic()
# Cross-Correlation max values per category
corr_cross(fat1train, type = 2, top = NA)
par(mfrow=c(2,2))
boxplot(fat1train[,1:5], col="orange",border="brown")
boxplot(fat1train[,6:10], col="orange",border="brown")
boxplot(fat1train[,11:14], col="orange",border="brown")
boxplot(fat1train[,15:18], col="orange",border="brown")
# Set a seed
set.seed(42)
# Apply PCA
pca <- prcomp(fat1train, scale = TRUE)
fviz_eig(pca)

# pca variables plot
fviz_pca_var(pca, col.var = "contrib", gradient.cols = c("yellow", "orange",
"red"),repel = TRUE)

ytrue    <- fat1test$brozek;
MSEtrain <- NULL;
MSEtest  <- NULL;

## Model 1: Linear regression with all predictors (Full Model)
model1 <- lm( brozek ~ ., data = fat1train);

## Model 1: fat1train error
MSEmod1train <-   mean( (resid(model1) )^2);
MSEtrain <- c(MSEtrain, MSEmod1train);
```

```r
# Model 1: testing error
pred1a <- predict(model1, fat1test[,-1]);
MSEmod1test <-   mean((pred1a - ytrue)^2);
MSEtest <- c(MSEtest, MSEmod1test);

## model 2:  Linear regression with the best subset of k = 5 predictors variables
fat.leaps <- regsubsets(brozek ~ ., data= fat1train, nbest= 100, really.big= TRUE);
fat.models <- summary(fat.leaps)$which;
fat.models.size <- as.numeric(attr(fat.models, "dimnames")[[1]]);
fat.models.rss <- summary(fat.leaps)$rss;

op2 <- which(fat.models.size == 5);
flag2 <- op2[which.min(fat.models.rss[op2])]

mod2selectedmodel <- fat.models[flag2,];
mod2Xname <- paste(names(mod2selectedmodel)[mod2selectedmodel][-1], collapse="+");
mod2form <- paste ("brozek ~", mod2Xname);

model2 <- lm( as.formula(mod2form), data= fat1train)

MSEmod2train <- mean(resid(model2)^2);
MSEtrain <- c(MSEtrain, MSEmod2train);
pred2 <- predict(model2, fat1test[,-1]);
MSEmod2test <-   mean((pred2 - ytrue)^2);
MSEtest <- c(MSEtest, MSEmod2test)
model3  <- step(model1);
## Model 3: Linear regression with variables (stepwise) selected using AIC
MSEmod3train <- mean(resid(model3)^2);
pred3 <- predict(model3, fat1test[,-1]);
MSEmod3test <-   mean((pred3 - ytrue)^2);
MSEtrain <- c(MSEtrain, MSEmod3train);
MSEtest <- c(MSEtest, MSEmod3test);
## Model 4: Ridge regression
fat.ridge <- lm.ridge( brozek ~ ., data = fat1train, lambda= seq(0,100,0.001));
```

```
indexopt <-  which.min(fat.ridge$GCV);

fat.ridge$coef[,indexopt]
ridge.coeffs = fat.ridge$coef[,indexopt]/ fat.ridge$scales;
intercept = -sum( ridge.coeffs  * colMeans(fat1train[,2:18] )  )+ mean(fat1train[,1]);

yhat4train <- as.matrix(fat1train[,2:18]) %*% as.vector(ridge.coeffs) + intercept;
MSEmod4train <- mean((yhat4train - fat1train$brozek)^2);
MSEtrain <- c(MSEtrain, MSEmod4train);
pred4test <- as.matrix( fat1test[,2:18]) %*% as.vector(ridge.coeffs) + intercept;
MSEmod4test <-  mean((pred4test - ytrue)^2);
MSEtest <- c(MSEtest, MSEmod4test);
## Model 5: LASSO
fat.lars <- lars( as.matrix(fat1train[,2:18]), fat1train[,1], type= "lasso", trace= TRUE);

Cp1  <- summary(fat.lars)$Cp;
index1 <- which.min(Cp1);

coef(fat.lars)[index1,]
fat.lars$beta[index1,]

##   the third way is to get the coefficients via prediction function
lasso.lambda <- fat.lars$lambda[index1]
coef.lars1 <- predict(fat.lars, s=lasso.lambda, type="coef", mode="lambda")
coef.lars1$coef

LASSOintercept = mean(fat1train[,1]) -sum( coef.lars1$coef  * colMeans(fat1train[,2:18] ));
c(LASSOintercept, coef.lars1$coef)

## Model 5:  fat1train error for lasso
pred5train  <- predict(fat.lars, as.matrix(fat1train[,2:18]), s=lasso.lambda, type="fit", mode="lambda");
yhat5train <- pred5train$fit;
MSEmod5train <- mean((yhat5train - fat1train$brozek)^2);
MSEtrain <- c(MSEtrain, MSEmod5train);
```

```r
## Model 5:  fat1train error for lasso
pred5test <- predict(fat.lars, as.matrix(fat1test[,2:18]), s=lasso.lambda, type="fit", mode="lambda");
yhat5test <- pred5test$fit;
MSEmod5test <- mean( (yhat5test - fat1test$brozek)^2);
MSEtest <- c(MSEtest, MSEmod5test);
## Model 6: Principal component regression
fat.pca <- pcr(brozek~., data=fat1train, validation="CV");
ncompopt <- which.min(fat.pca$validation$adj); # optimal number of components is 17
# ncompopt
## fat1train Error with the optimal choice of PCs
ypred6train <- predict(fat.pca, ncomp=ncompopt, newdata =fat1train[2:18]);
MSEmod6train <- mean( (ypred6train - fat1train$brozek)^2);
MSEtrain <- c(MSEtrain, MSEmod6train);

## Testing Error with the optimal choice of PCs
ypred6test <- predict(fat.pca, ncomp = ncompopt, newdata =fat1test[2:18]);
MSEmod6test <- mean( (ypred6test - fat1test$brozek)^2);
MSEtest <- c(MSEtest, MSEmod6test)

## Model 7: Partial least squares
fat.pls <- plsr(brozek ~ ., data = fat1train, validation="CV");

mod7ncompopt <- which.min(fat.pls$validation$adj);
# mod7ncompopt
ypred7train <- predict(fat.pls, ncomp = mod7ncompopt, newdata = fat1train[2:18]);
MSEmod7train <- mean( (ypred7train - fat1train$brozek)^2);
MSEtrain <- c(MSEtrain, MSEmod7train);

## Testing Error with the optimal choice of "mod7ncompopt"
ypred7test <- predict(fat.pls, ncomp = mod7ncompopt, newdata = fat1test[2:18]);
MSEmod7test <- mean( (ypred7test - fat1test$brozek)^2);
MSEtest <- c(MSEtest, MSEmod7test);
error_summary <- rbind(MSEtrain, MSEtest)
colnames(error_summary) <- c( "mod1","mod2", "mod3", "mod4","mod5","mod6", "mod7")
```

```
# displaying the tables
knitr::kable(error_summary,"pipe", digit=5,
             caption = paste("MSE errors for all seven models"))%>%
          kable_styling(position = "center", full_width = F)
fatfull = rbind(fat1train, fat1test)

### combine to a full data set
n1 = 227; # training set sample size

n = dim(fatfull)[1]; ## the total sample size
set.seed(7406); ### set the seed for randomization

B = 100; ## number of loops

TEALL = NULL; ### Final TE values for testing errors

for (b in 1:B){
  ## randomly select n1 observations as new training subset in each loop
  flag <- sort(sample(1:n,n1));
  fat1traintempset <- fatfull[flag, ];
  fat1testtempset <- fatfull[-flag, ];

  fattest <- fat1testtempset[,2:18]
  ytrue <- fat1testtempset[,1]

  #--------------------LR----------------------#
  fat.lm <- lm(brozek~ . , data= fat1traintempset);

  pred1 <- predict(fat.lm, fattest);
  te1 <-   mean((pred1 - ytrue)^2);

  #----------------- subset LR ----------------------#
  fat.leaps <- regsubsets(brozek ~ ., data=fat1traintempset, really.big= TRUE);
```

```r
## Record useful information from the output
fat.models <- summary(fat.leaps)$which;
fat.models.size <- as.numeric(attr(fat.models, "dimnames")[[1]]);
fat.models.rss <- summary(fat.leaps)$rss;

op2 <- which(fat.models.size == 5);
flag2 <- op2[which.min(fat.models.rss[op2])]

mod2selectedmodel <- fat.models[flag2,];
mod2Xname <- paste(names(mod2selectedmodel)[mod2selectedmodel][-1], collapse="+");
mod2form <- paste ("brozek ~", mod2Xname);

model2 <- lm(as.formula(mod2form), data= fat1traintempset)

pred2 <- predict(model2, fattest);
te2 <- mean((pred2 - ytrue)^2);

#------------------------------Set-wise LR -------------------#
model3  <- step(model1);
pred3 <- predict(model3, fattest);
te3 <-   mean((pred3 - ytrue)^2);

#------------------------------Ridge-----------------#
fat.ridge <- lm.ridge( brozek ~ ., data = fat1traintempset, lambda= seq(0,100,0.001));
indexopt <-  which.min(fat.ridge$GCV);
fat.ridge$coef[,indexopt]
ridge.coeffs = fat.ridge$coef[,indexopt]/ fat.ridge$scales;
intercept = -sum( ridge.coeffs  * colMeans(fat1traintempset[,2:18] )  )+ mean(fat1traintempset[,1]);
pred4test <- as.matrix(fattest)%*% as.vector(ridge.coeffs) + intercept;
te4 <-  mean((pred4test - ytrue)^2);

#------------------------------Lasso----------------------#
fat.lars <- lars(as.matrix(fat1traintempset[,2:18]), fat1traintempset[,1], type= "lasso", trace= TRUE);
```

```
  Cp1  <- summary(fat.lars)$Cp;
  index1 <- which.min(Cp1);

  lasso.lambda <- fat.lars$lambda[index1]
  lasso.lambda[is.na(lasso.lambda)] <- 0

  pred5test <- predict(fat.lars, as.matrix(fattest), s=lasso.lambda, type="fit", mode="lambda");
  yhat5test <- pred5test$fit;
  te5 <- mean((yhat5test - ytrue)^2);

  #--------------------------PCR----------------------#
  fat.pca <- pcr(brozek~., data=fat1traintempset, validation="CV");
  ncompopt <- which.min(fat.pca$validation$adj); # optimal number of components is 17

  ypred6test <- predict(fat.pca, ncomp = ncompopt, newdata =fattest);
  te6 <- mean( (ypred6test - ytrue)^2);

  #--------------------------PLSR----------------------#
  fat.pls <- plsr(brozek ~ ., data = fat1traintempset, validation="CV");

  mod7ncompopt <- which.min(fat.pls$validation$adj);

  ypred7test <- predict(fat.pls, ncomp = mod7ncompopt, newdata = fattest);
  te7 <- mean( (ypred7test - ytrue)^2);
  TEALL = rbind( TEALL, cbind(te1, te2, te3, te4, te5, te6, te7));
}

colnames(TEALL) <- c( "mod1","mod2", "mod3", "mod4","mod5","mod6", "mod7")

Mean = apply(TEALL, 2, mean);
Variance  = apply(TEALL, 2, var);

data = rbind(Mean, Variance)
```

```r
# displaying the tables
knitr::kable(data,"pipe", digit=5,
             caption = paste("Sample mean/variance of MCCV Errors for B = ", B))%>%
          kable_styling(position = "center", full_width = F)
fatfull = rbind(fat1train, fat1test)

### combine to a full data set
n1 = 227; # training set sample size

n = dim(fatfull)[1]; ## the total sample size
set.seed(7405); ### set the seed for randomization

B = 800; ## number of loops

TEALL = NULL; ### Final TE values for testing errors

for (b in 1:B){
  ## randomly select n1 observations as new training subset in each loop
  flag <- sort(sample(1:n,n1));
  fat1traintempset <- fatfull[flag, ];
  fat1testtempset <- fatfull[-flag, ];

  fattest <- fat1testtempset[,2:18]
  ytrue <- fat1testtempset[,1]

  #---------------------LR-----------------------#
  fat.lm <- lm(brozek~ . , data= fat1traintempset);

  pred1 <- predict(fat.lm, fattest);
  te1 <-   mean((pred1 - ytrue)^2);

  #------------------ subset LR ------------------------#
  fat.leaps <- regsubsets(brozek ~ ., data=fat1traintempset, really.big= TRUE);
```

```
fat.models <- summary(fat.leaps)$which;
fat.models.size <- as.numeric(attr(fat.models, "dimnames")[[1]]);
fat.models.rss <- summary(fat.leaps)$rss;

op2 <- which(fat.models.size == 5);
flag2 <- op2[which.min(fat.models.rss[op2])]

mod2selectedmodel <- fat.models[flag2,];
mod2Xname <- paste(names(mod2selectedmodel)[mod2selectedmodel][-1], collapse="+");
mod2form <- paste ("brozek ~", mod2Xname);
model2 <- lm(as.formula(mod2form), data= fat1traintempset)

## Model 2:  testing error
pred2 <- predict(model2, fattest);
te2 <- mean((pred2 - ytrue)^2);

#--------------------------------Set-wise LR -------------------#
model3  <- step(model1);
pred3 <- predict(model3, fattest);
te3 <-   mean((pred3 - ytrue)^2);

#--------------------------------Ridge-----------------#
fat.ridge <- lm.ridge( brozek ~ ., data = fat1traintempset, lambda= seq(0,100,0.001));
indexopt <-  which.min(fat.ridge$GCV);
fat.ridge$coef[,indexopt]
ridge.coeffs = fat.ridge$coef[,indexopt]/ fat.ridge$scales;
intercept = -sum( ridge.coeffs  * colMeans(fat1traintempset[,2:18] )  )+ mean(fat1traintempset[,1]);
pred4test <- as.matrix(fattest)%*% as.vector(ridge.coeffs) + intercept;
te4 <-  mean((pred4test - ytrue)^2);

#------------------------------Lasso-----------------------#
fat.lars <- lars(as.matrix(fat1traintempset[,2:18]), fat1traintempset[,1], type= "lasso", trace= TRUE);

Cp1  <- summary(fat.lars)$Cp;
```

```
  index1 <- which.min(Cp1);
  lasso.lambda <- fat.lars$lambda[index1]
  lasso.lambda[is.na(lasso.lambda)] <- 0

  pred5test <- predict(fat.lars, as.matrix(fattest), s=lasso.lambda, type="fit", mode="lambda");
  yhat5test <- pred5test$fit;
  te5 <- mean((yhat5test - ytrue)^2);

  #--------------------------PCR----------------------#
  fat.pca <- pcr(brozek~., data=fat1traintempset, validation="CV");
  ncompopt <- which.min(fat.pca$validation$adj); # optimal number of components is 17

  ypred6test <- predict(fat.pca, ncomp = ncompopt, newdata =fattest);
  te6 <- mean( (ypred6test - ytrue)^2);

  #--------------------------PLSR----------------------#
  fat.pls <- plsr(brozek ~ ., data = fat1traintempset, validation="CV");

  mod7ncompopt <- which.min(fat.pls$validation$adj);

  ypred7test <- predict(fat.pls, ncomp = mod7ncompopt, newdata = fattest);
  te7 <- mean( (ypred7test - ytrue)^2);
  TEALL = rbind( TEALL, cbind(te1, te2, te3, te4, te5, te6, te7));
}

colnames(TEALL) <- c( "mod1","mod2", "mod3", "mod4","mod5","mod6", "mod7")

Mean = apply(TEALL, 2, mean);
Variance  = apply(TEALL, 2, var);

data = rbind(Mean, Variance)

# displaying the tables
knitr::kable(data,"pipe", digit=5,
```

```
            caption = paste("Sample mean/variance of MCCV Errors for B = ", B))%>%
          kable_styling(position = "center", full_width = F)
par(cex.main=1)
rmeanplot(TEALL, style="plain", col="blue",
          plot.title = "Monte Carlo simulation running mean per model")
```