PENNSTATE

December 14, 2015

Evan Agovino / Haiteng Xu / Xiang Wang

Department of Statistics
Eberly College of Science
Penn State University

1. Introduction

Today we are going to be doing a study of marketing data for a charitable organization. This organization wants to develop a data-mining model to improve the cost-effectiveness of their direct marketing campaigns to their donors. Recent mailing records show a typical overall response rate of 10%, an average donation of $14.50 and a cost of $2 per mailing. Because the expected profit for each mailing is $14.50*10% - $2 = -$0.55, it is not cost effective to mail everyone.

Thus, we want to develop a classification model using our data to capture our most likely donors and maximize expected net profit. We would also like to build a prediction model to predict donation amounts for donors, using data from only our customers who have previously donated. The entire dataset consists of 3,984 training observations, 2,018 validation observations and 2,007 test observations. The responders have been over-represented in the training and validation observations so that responders are closer to 50% of the data instead of the typical 10%.

2. Exploratory Data Analysis

Removing Extraneous Variables

First, let's comb through the data to see if there are any predictors we need to transform or remove from our analysis. There are twenty total predictor variables and two response variables: DONR for our classification model and DAMT for our regression model.

The variables INCM (Median Family Income), INCA (Average Family Income) and PLOW (Neighborhood Low Income %) are all strongly correlated with each other and AVHV (Average Home Value). For this reason, we will remove them from both models. The TGIF variable ($ amount of lifetime gifts to date) is strongly correlated with NPRO (Lifetime # of Promotions Received), so we will remove it from both models. The LGIF ($ amount of largest gift to date) and AGIF (average $ amount of gifts to date) variables strongly correlate with RGIF ($ amount of recent gift), so we will remove them from both models.

Transformations

Next, we find that the AVHV (Average Home Value), RGIF ($ amount of Recent Gift) and TLAG (# of months between first and second gift) variables are skewed to the right. We will log-transform these so they are normally distributed (see **Exhibit 3** and **Exhibit 4** for histograms of these variables before and after the log transformation). We will also add **HINC^2** to the model as we find that the profit with it included is generally higher than the profit without it for most of our classification models. Now we have 15 total predictor variables, after removing six and adding one.

3. Models

First we would like to define some common steps taken for all the models.

a) Develop a classification model for the DONR variable using other variables as predictors (except ID and DAMT). Fit all candidate models using the 3984 training observations and evaluate the fitted models using 2018 validation observations.

b) Then we will use “maximum profit” as the evaluation criteria to select the final classification model to classify DONR responses in the test dataset. The calculation of “maximum profit” as follows

- Calculate the posterior probabilities for the validation dataset;
- Sort DONR in order of the posterior probabilities from highest to lowest;
- Calculate the cumulative sum of (14.5 * DONR – 2);
- Then find the maximum of this profit function.

c) Develop a prediction model for the DAMT variable using other variables as predictors (except ID and DONR). We will use only the data records for which DONR=1. Fit all candidate models using the 3984 training observations and evaluate the fitted models using the 2018 validation observations.

d) Then we will use “mean prediction error” as the evaluation criteria to select the final prediction model to predict DAMT responses in the test dataset.

mean prediction error = mean((valid.y-pred.value)^2)
std error = sd((valid.y-pred.value)^2)/sqrt(n.valid.y)

4. Classification Models

Classification Model 1 – Linear Discriminant Analysis

	Validation Leads			
LDA		0	1	
Predictions	0	661	12	673
	1	358	987	1345
		1019	999	2018

Number of Predicted Donors: 1345

Number of Correctly Predicted Donors: 987

Profit: (\$14.5 * 987) – (\$2 * 1345) = **\$11,621.50**

Classification Model 2 – Logistic Regression

	Validation Leads			
Logistic Regression Predictions		0	1	
	0	724	21	745
	1	295	978	1273
		1019	999	2018

Number of Predicted Donors: 1273

Number of Correctly Predicted Donors: 978

Profit: ($14.5 * 978) – ($2 * 1273) = **$11,635.00**

Classification Model 3 – Quadratic Discriminant Analysis

	Validation Leads			
QDA Predictions		0	1	
	0	593	34	627
	1	426	965	1391
		1019	999	2018

Number of Predicted Donors: 1391

Number of Correctly Predicted Donors: 965

Profit: ($14.5 * 965) – ($2 * 1391) = **$11,210.50**

Classification Model 4 – K-Nearest Neighbors

This was modeled using all twenty predictors

K=29 (found via Leave-One-Out Cross Validation)

(We are unable to see the results of leads versus predictions, just the final result)

Number of Predicted Donors: 1305

Number of Correctly Predicted Donors: 953

Profit: ($14.5 * 953) – ($2 * 1305) = **$11,208.50**

Classification Model 5 – Generalized Additive Model

This was modeled using all twenty predictors**.** The predictors determined to be non-significant (GENF, LGIF, RGIF, AGIF) were removed.

Each ordinal and quantitative predictor was modeled using a smoothing spline with five degrees of freedom.

	Validation Leads			
GAM Predictions		0	1	
	0	776	5	781
	1	243	994	1237
		1019	999	2018

Number of Predicted Donors: 1237

Number of Correctly Predicted Donors: 994

Profit: ($14.5 * 994) – ($2 * 1237) = **$11,939.00**

Classification Model 6 – Regression Tree

This was modeled using all twenty predictors plus the squared value of the "HINC" predictor.

	Validation Leads			
Regression		0	1	
Tree	0	645	37	692
Predictions	1	374	962	1336
		1019	999	2018

Number of Predicted Donors: 1336

Number of Correctly Predicted Donors: 962

Profit: ($14.5 * 962) – ($2 * 1336) = **$11,277.00**

Classification Model 7 – Random Forest

We found that the random forest model with fifteen variables performed better than the model with all twenty predictors plus the squared value of the "HINC" predictor.

	Validation Leads			
Random		0	1	
Forest	0	750	12	762
Predictions	1	269	987	1256
		1019	999	2018

Number of Predicted Donors: 1256

Number of Correctly Predicted Donors: 987

Profit: ($14.5 * 987) – ($2 * 1256) = **$11,799.50**

Classification Model 8 – Support Vector Machine

This was modeled using all twenty predictors plus the squared value of the "HINC" predictor.

	Validation Leads			
Random		0	1	
Forest	0	686	18	704
Predictions	1	333	981	1314
		1019	999	2018

Number of Predicted Donors: 1314

Number of Correctly Predicted Donors: 981

Profit: ($14.5 * 981) – ($2 * 1314) = **$11,596.50**

5. Prediction Models

Prediction Model 1 – Least Squares Regression

We begin with our fifteen predictors and drop REG1, REG2, WRAT and TLAG from the model as they are not significant.

Mean Prediction Error: 1.880

Standard Deviation: 0.169

Prediction Model 2 – Stepwise Regression Using AIC
The model with the smallest AIC found via stepwise regression ends up having the same predictors as our first model with REG2 added.
Mean Prediction Error: 1.872
Standard Deviation: 0.169

Prediction Model 3 – Best Subset Selection Using BIC
The eight-variable model with REG3, REG4, HOME, CHLD, HINC, NPRO, RGIF and TDON has the lowest BIC value.
Mean Prediction Error: 1.895
Standard Deviation: 0.170

Prediction Model 4 – Ridge Regression
$\Lambda = 0.787$ (found via 10-fold cross validation: See **Exhibit 5**)
Mean Prediction Error: 1.836
Standard Deviation: 0.173

Prediction Model 5 – Lasso Regression (with 10-Fold Cross Validation)
$\Lambda = 0.099$ (found via 10-fold cross validation: See **Exhibit 6**)
Mean Prediction Error: 1.819
Standard Deviation: 0.164

Prediction Model 6 – Principal Components Regression
The lowest cross-validation error occurs when 15 components are used. Even though the CV error only reduces slightly when less components are used, we find that the 15 component model has the lowest CV error.
Mean Prediction Error: 1.870
Standard Deviation: 0.169

Prediction Model 7 – Partial Least Squares
The lowest cross-validation error occurs when 5 components are used.
Mean Prediction Error: 1.873
Standard Deviation: 0.169

6. Summary Results

Classification Model Results:

<u>Number of Predicted Donors</u>	<u>Profit</u>	<u>Model</u>
1345	$11,621.50	Linear Discriminant Analysis
1273	$11,635.00	Logistic Regression
1391	$11,210.50	Quadratic Discriminant Analysis
1305	$11,208.50	K-Nearest Neighbors
1237	**$11,939.00**	**Generalized Additive Model**
1391	$11,312.00	Regression Tree
1314	$11,799.50	Random Forest
1314	$11,596.50	Support Vector Machine

The success of the linear discriminant analysis and logistic regression models relative to the quadratic discriminant analysis shows that the observations within each class are something close to uncorrelated random normal variables, which is to be expected after we removed the correlated variables and transformed the skewed variables. Since there only two response categories, donor (1) or non-donor (0), the logistic regression performs better than the LDA model (although only slightly).

Still, the generalized additive model and random forest are superior to linear discriminant analysis and logistic regression. The generalized additive model may be the most superior because it allows us to automatically model non-linear relationships, meaning that we don't need to worry about whether the transformations we made impacted the model. This helps when there are over a dozen predictors we can transform! From the earlier data analysis we could tell that some predictors had non-linear relationship with the response, so GAM's non-linear fits can potentially make more accurate predictions for the response *Y.* And this is the case here, GAM model gives the most predictive power.

Prediction Model Results:

<u>Mean Prediction Error</u>	<u>Standard Deviation</u>	<u>Model</u>
1.880	0.169	Least Squares Regression
1.872	0.169	Stepwise Regression (AIC)
1.895	0.170	Best Subset Selection (BIC)
1.832	0.173	Ridge Regression
1.819	**0.164**	**Lasso**
1.870	0.169	Principal Components Regression
1.873	0.169	Partial Least Sqaures

We can see that the ridge regression and lasso model perform better than the least squares models. This could mean that the least squares estimates have high variance. We also see that the lasso performs better than the ridge regression, meaning that there are some extraneous predictors that the lasso model removed. There were only 10 predictors in the lasso regression as opposed to the twenty in ridge regression. If the response was a function of more predictors, the ridge regression would have performed better.

7. Conclusion

Based on maximum profit, the **generalized additive model** scores the best as it nets us a profit of $11,939.00 on the validation data.

Based on mean prediction error, the **lasso model** scores the best as it gives us a mean prediction error of **1.819**.

Appendix

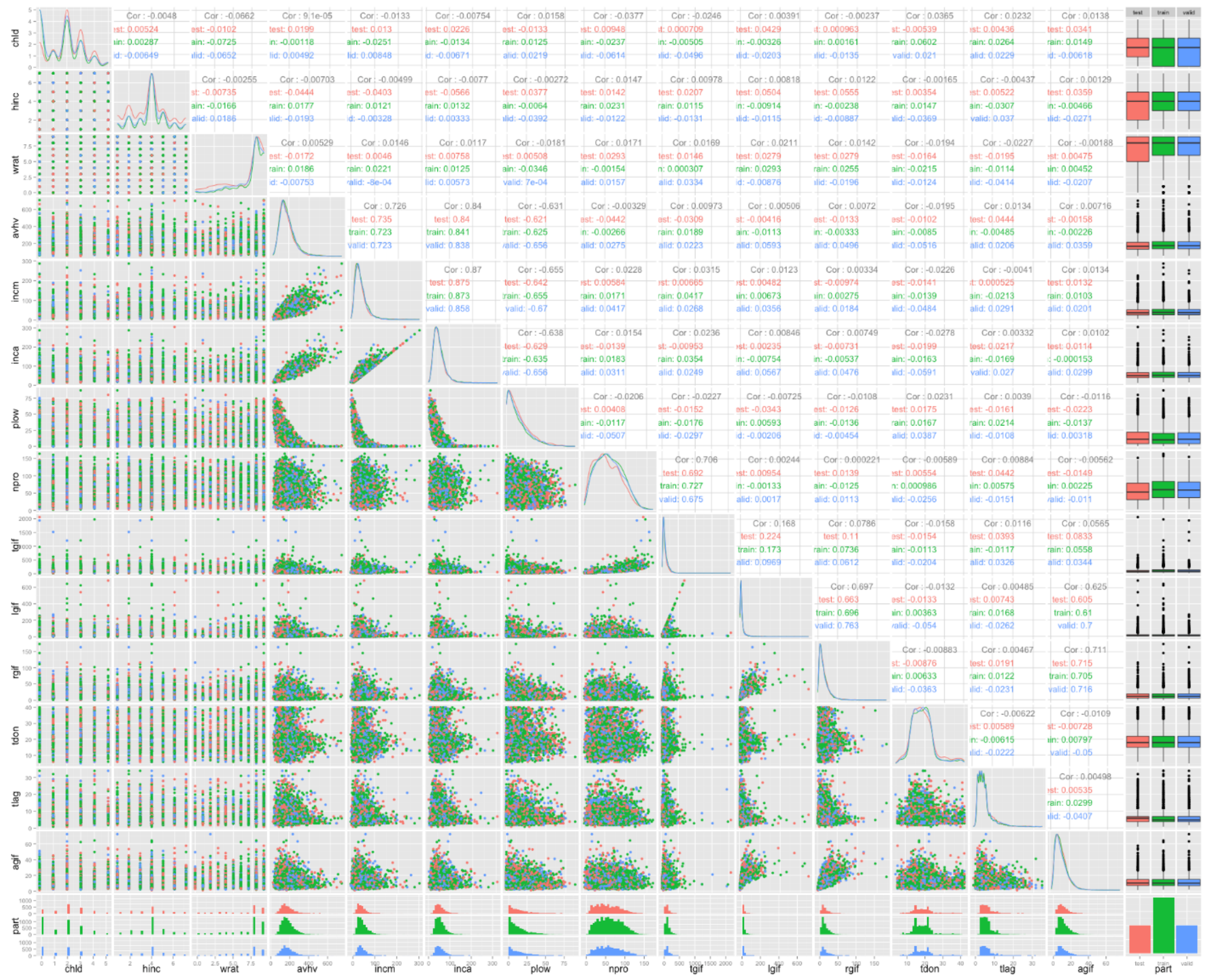
Exhibit 1:
<u>List of Variables (variables eliminated from the model are *italicized*):</u>

ID	Customer # - Not a predictor variable	Ordinal
REG1/REG2 /REG3/REG4	Region (Five Regions Total)	Binary (0 for all four responses means customer is in Reg 5)
HOME	Homeowner	Binary
CHLD	Number of Children	Ordinal
HINC	Household Income	Ordinal
GENF	Gender	Binary
WRAT	Wealth Rating	Ordinal
AVHV	Average Home Value	Quantitative
INCM	*Median Family Income*	*Quantitative*
INCA	*Average Family Income*	*Quantitative*
PLOW	*Neighborhood Low Income %*	*Quantitative*
NPRO	Lifetime Number of Promotions Received	Quantitative
TGIF	*$ amount of lifetime gifts to date*	*Quantitative*
LGIF	*$ amount of largest gift to date*	*Quantitative*
RGIF	$ amount of most recent gift	Quantitative
TDON	# of Months since Last Donation	Quantitative
TLAG	# of months between first and second gift	Quantitative
AGIF	*Average $ amount of gifts to date*	*Quantitative*
DONR	RESPONSE VARIABLE: (Donor)	Binary
DAMT	RESPONSE VARIABLE: (Predicted Donation Amount)	Quantitative

RStudio Version 0.98.1103 – © 2009-2014 in Mac OS X 10_11_1

Exhibit 2:

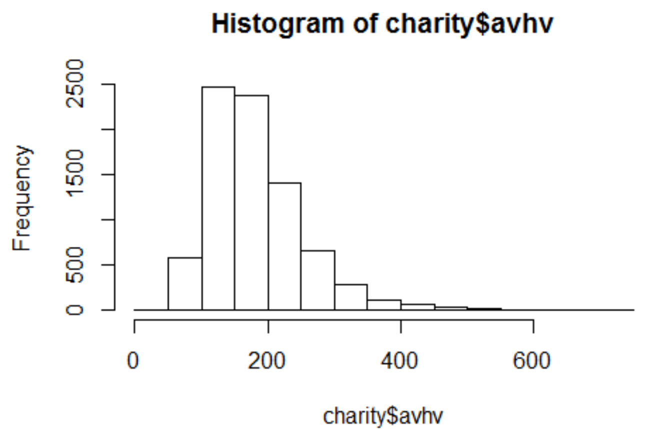
Scatterplots of Predictors

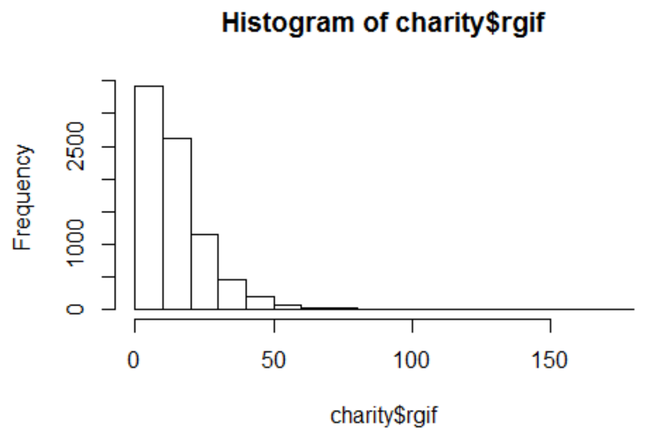


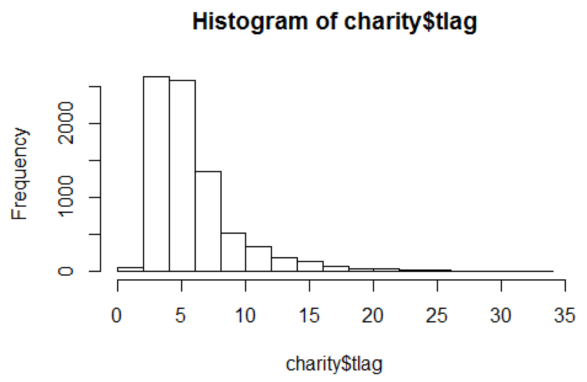
RStudio Version 0.98.1103 – © 2009-2014 in Mac OS X 10_11_1

Exhibit 3:

Histograms of AVHV, RGIF and TLAG Variables (Before Log Transformation)



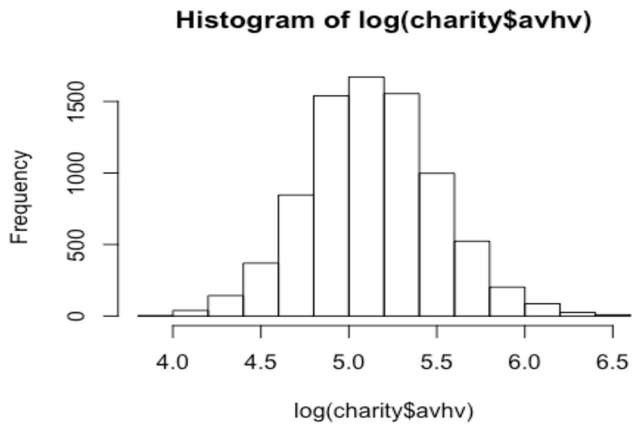


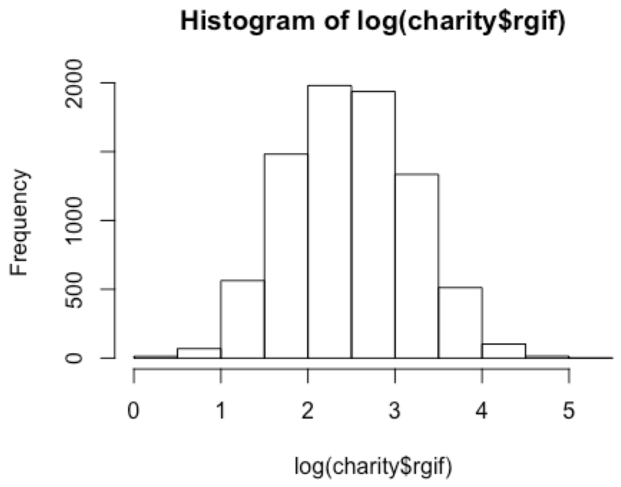


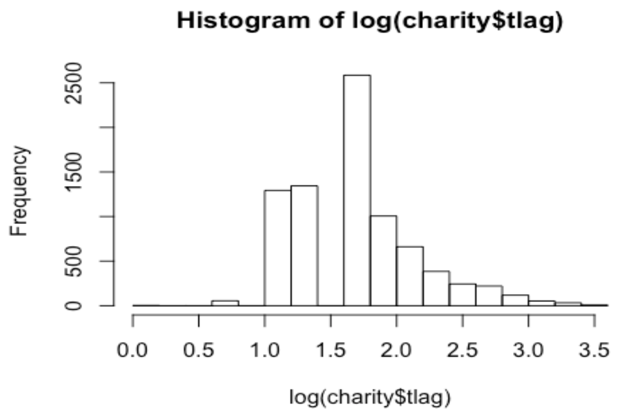
RStudio Version 0.98.1103 – © 2009-2014 in Mac OS X 10_11_1

Exhibit 4:

Histograms of AVHV, RGIF and TLAG Variables (After Log Transformation)







RStudio Version 0.98.1103 – © 2009-2014 in Mac OS X 10_11_1

Exhibit 5:

Lambda Plot for Ridge Regression Model

(Note the 20s on top of the graph mean that the ridge regression model will always have twenty predictors)

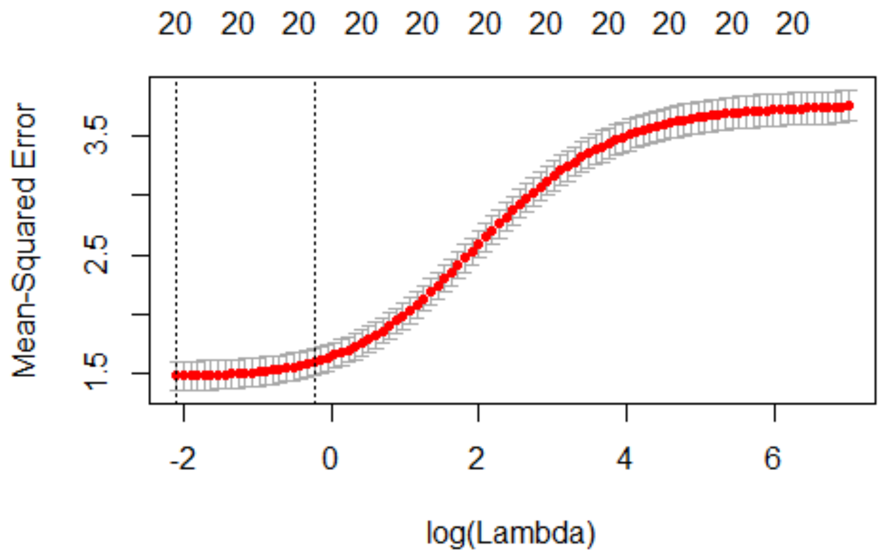
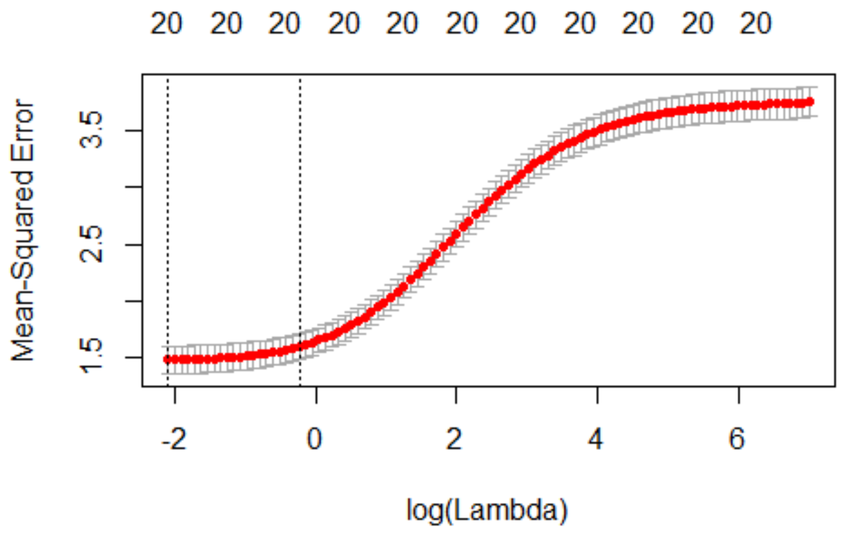


Exhibit 6:

Lambda Plot for Lasso Model

(Note the numbers between one and twenty on top of the graph mean that the ridge regression model will always have twenty predictors)



RStudio Version 0.98.1103 – © 2009-2014 in Mac OS X 10_11_1

R Code and some output

Exploratory Data Analysis

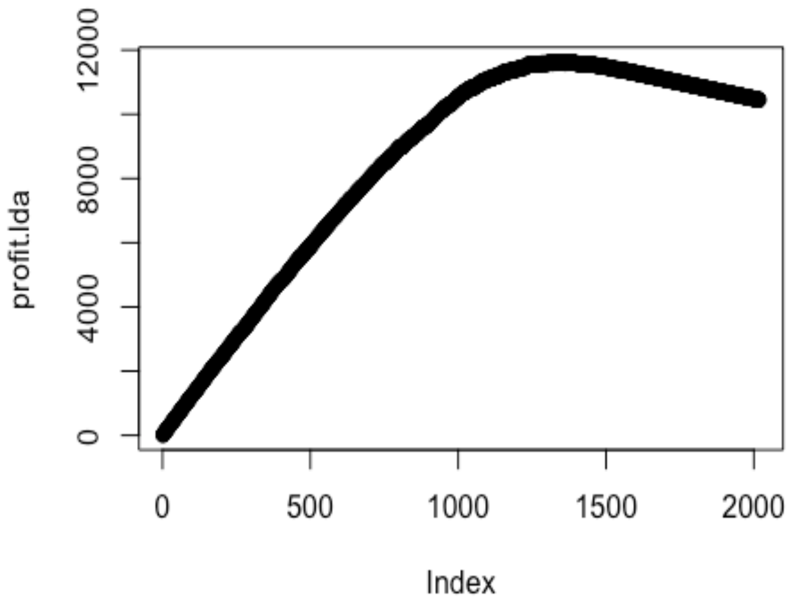
```
charity <- read.csv("charity.csv")
charity1 <- charity[, c(7:8, 10:21, 24)]
## Matrix scatter plot using ggpairs()
library(GGally); # ggpairs(charity1, colour='part')
hist(log(charity$avhv))
hist(log(charity$rgif))
hist(log(charity$tlag))

# transformations
charity.t <- charity
charity.t$avhv <- log(charity.t$avhv)
charity.t$rgif <- log(charity.t$rgif)
charity.t$tlag <- log(charity.t$tlag)

# set up data for analysis
data.train <- charity.t[charity$part=="train",]
x.train <- data.train[,2:21]
c.train <- data.train[,22] # donr
n.train.c <- length(c.train) # 3984
y.train <- data.train[c.train==1,23] # damt for observations with donr=1
n.train.y <- length(y.train) # 1995

data.valid <- charity.t[charity$part=="valid",]
x.valid <- data.valid[,2:21]
c.valid <- data.valid[,22] # donr
n.valid.c <- length(c.valid) # 2018
y.valid <- data.valid[c.valid==1,23] # damt for observations with donr=1
n.valid.y <- length(y.valid) # 999

data.test <- charity.t[charity$part=="test",]
n.test <- dim(data.test)[1] # 2007
x.test <- data.test[,2:21]

x.train.mean <- apply(x.train, 2, mean)
x.train.sd <- apply(x.train, 2, sd)
x.train.std <- t((t(x.train)-x.train.mean)/x.train.sd) # standardize to have zero mean and unit sd
apply(x.train.std, 2, mean) # check zero mean
apply(x.train.std, 2, sd) # check unit sd
```

```
data.train.std.c <- data.frame(x.train.std, donr=c.train) # to classify donr
data.train.std.y <- data.frame(x.train.std[c.train==1,], damt=y.train) # to predict damt when
donr=1

x.valid.std <- t((t(x.valid)-x.train.mean)/x.train.sd) # standardize using training mean and sd
data.valid.std.c <- data.frame(x.valid.std, donr=c.valid) # to classify donr
data.valid.std.y <- data.frame(x.valid.std[c.valid==1,], damt=y.valid) # to predict damt when
donr=1

x.test.std <- t((t(x.test)-x.train.mean)/x.train.sd) # standardize using training mean and sd
data.test.std <- data.frame(x.test.std)
```

Classification modeling

Model 1: linear discriminant analysis (LDA)

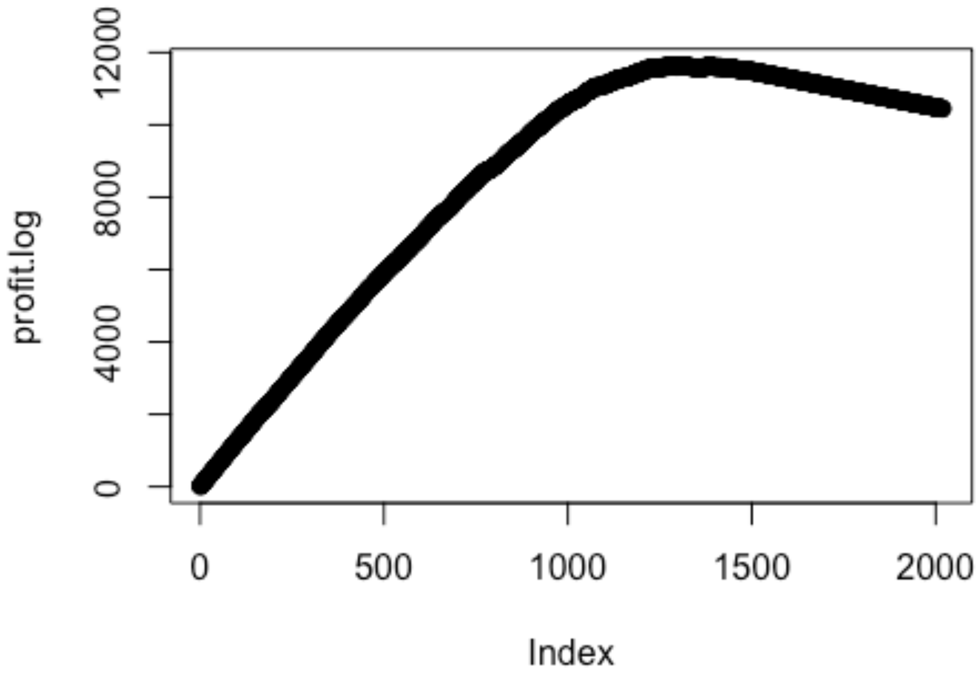
```
library(MASS)
model.lda <- lda(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat
+ avhv + npro + rgif + tdon + tlag, data.train.std.c) # include additional terms on the fly using I()
# Note: strictly speaking, LDA should not be used with qualitative predictors, but in practice it
often is if the goal is simply to find a good predictive model
post.valid.lda <- predict(model.lda, data.valid.std.c)$posterior[,2] # n.valid.c post probs
# calculate ordered profit function using average donation = $14.50 and mailing cost = $2
profit.lda <- cumsum(14.5*c.valid[order(post.valid.lda, decreasing=T)]-2)
plot(profit.lda) # see how profits change as more mailings are made
```



```
n.mail.valid <- which.max(profit.lda) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.lda)) # report number of mailings and maximum profit; 1345, 11621.5
## [1]  1345.0 11621.5
cutoff.lda <- sort(post.valid.lda, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.lda <- ifelse(post.valid.lda > cutoff.lda, 1, 0) # mail to everyone above the cutoff
table(chat.valid.lda, c.valid) # classification table
#              c.valid
#chat.valid.lda    0   1
#                0 661  12
#                1 358 987
# check n.mail.valid = 358+987 = 1345
# check profit = 14.5*987-2*1345 = 11621.5
```

Model 2: logistic regression (LR)

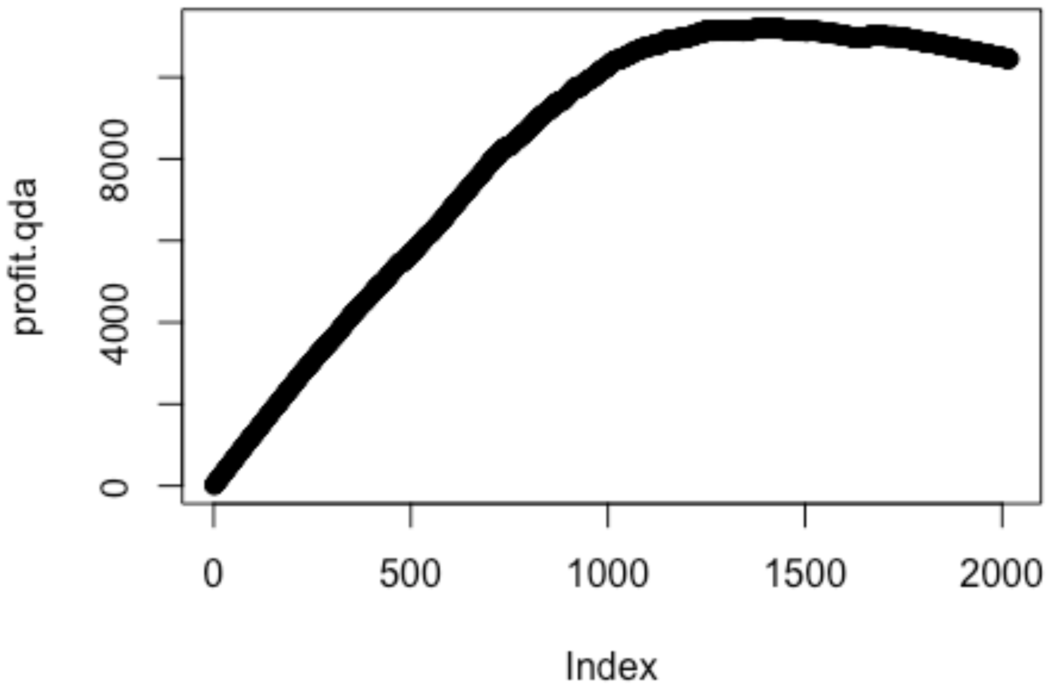
```
model.log <- glm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat
+ avhv + npro + rgif + tdon + tlag, data.train.std.c, family=binomial("logit"))
post.valid.log <- predict(model.log, data.valid.std.c, type="response") # n.valid post probs

# calculate ordered profit function using average donation = $14.50 and mailing cost = $2
profit.log <- cumsum(14.5*c.valid[order(post.valid.log, decreasing=T)]-2)
plot(profit.log) # see how profits change as more mailings are made
```



```
n.mail.valid <- which.max(profit.log) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.log)) # report number of mailings and maximum profit; 1273, 11635
## [1]  1273 11635
cutoff.log <- sort(post.valid.log, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.log <- ifelse(post.valid.log > cutoff.log, 1, 0) # mail to everyone above the cutoff
table(chat.valid.log, c.valid) # classification table
#              c.valid
#chat.valid.log    0   1
#             0 724  21
#             1 295 978
# check n.mail.valid = 295+978 = 1273
# check profit = 14.5*978-2*1273 = 11635
```

Model 3: Quadratic discriminant analysis (QDA)

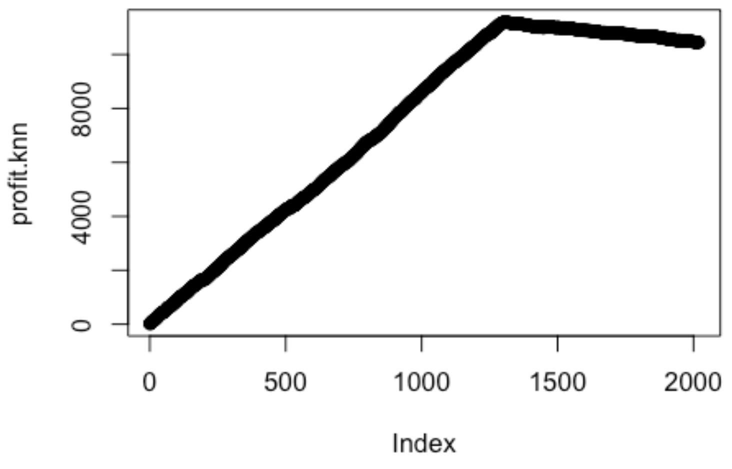
```
model.qda <- qda(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf +
wrat + avhv + npro + rgif + tdon + tlag, data.train.std.c)
post.valid.qda <- predict(model.qda, data.valid.std.c)$posterior[,2] # n.valid.c post probs
# calculate ordered profit function using average donation = $14.50 and mailing cost = $2
profit.qda <- cumsum(14.5*c.valid[order(post.valid.qda, decreasing=T)]-2)
plot(profit.qda) # see how profits change as more mailings are made
```



```
n.mail.valid <- which.max(profit.qda) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.qda)) # report number of mailings and maximum profit
## [1]  1391.0 11210.5

cutoff.qda <- sort(post.valid.qda, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.qda <- ifelse(post.valid.qda>cutoff.qda, 1, 0) # mail to everyone above the cutoff
table(chat.valid.qda, c.valid) # classification table
#              c.valid
#chat.valid.qda    0   1
#             0 593  34
#             1 426 965
# check n.mail.valid = 426+965 = 1391
# check profit = 14.5*965-2*1391 = 11210.5
```

Model 4: K-Nearest Neighbors (KNN)

```
library(class)
mer <- rep(NA, 30) # misclassification error rates based on leave-one-out cross-validation
set.seed(2014) # seed must be set because R randomly breaks ties
for (i in 1:30) mer[i] <- sum((c.train-(c(knn.cv(train=x.train, cl=c.train, k=i))-1))^2)/n.train.c
which.min(mer) # minimum occurs at k=29
set.seed(2014)
post.valid.knn <- knn(data.train.std.c[,-21], data.valid.std.c[,-21], data.train.std.c[,21], k=29, prob=T)

# calculate ordered profit function using average donation = $14.50 and mailing cost = $2
profit.knn <- cumsum(14.5*c.valid[order(post.valid.knn, decreasing=T)]-2)
plot(profit.knn) # see how profits change as more mailings are made
```



```
n.mail.valid <- which.max(profit.knn) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.knn)) # report number of mailings and maximum profit; 1305, 11208.5
## [1]  1305.0 11208.5
```

Model 5: Logistic regression generalized additive model (GAM) with each predictor modeled using a smoothing spline with 5 degrees of freedom

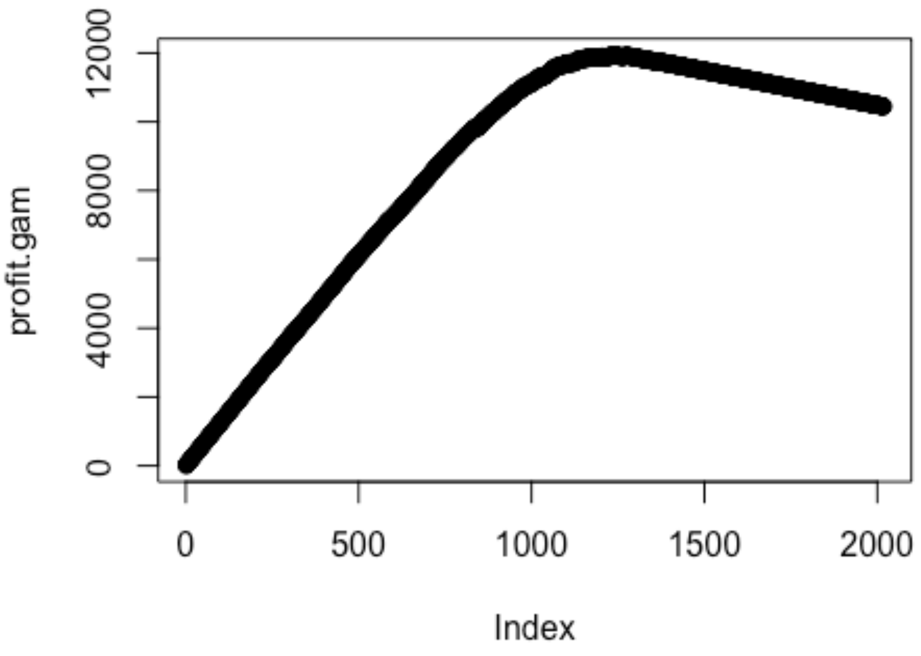
```
library(gam)
model.gam <- gam(donr ~ reg1 + reg2 + reg3 + reg4 + home + s(chld,df=5) + s(hinc,df=5) + genf
+ s(wrat,df=5) + s(avhv,df=5) + s(incm,df=5) + s(inca,df=5) + s(plow,df=5) + s(npro,df=5) +
s(tgif,df=5) + s(lgif,df=5) + s(rgif,df=5) + s(tdon,df=5) + s(tlag,df=5) + s(agif,df=5),
data.train.std.c, family=binomial)
summary(model.gam)
##
## Call: gam(formula = donr ~ reg1 + reg2 + reg3 + reg4 + home + s(chld,
##     df = 5) + s(hinc, df = 5) + genf + s(wrat, df = 5) + s(avhv,
##     df = 5) + s(incm, df = 5) + s(inca, df = 5) + s(plow, df = 5) +
##     s(npro, df = 5) + s(tgif, df = 5) + s(lgif, df = 5) + s(rgif,
##     df = 5) + s(tdon, df = 5) + s(tlag, df = 5) + s(agif, df = 5),
##     family = binomial, data = data.train.std.c)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -3.154918 -0.130361  0.001612  0.238887  2.983810
##
## (Dispersion Parameter for binomial family taken to be 1)
##
##     Null Deviance: 5522.988 on 3983 degrees of freedom
## Residual Deviance: 1521.23 on 3907 degrees of freedom
## AIC: 1675.231
##
## Number of Local Scoring Iterations: 11
##
## Anova for Parametric Effects
##                  Df  Sum Sq Mean Sq   F value    Pr(>F)
## reg1              1    1.24    1.24    1.9512   0.16254
## reg2              1  103.67  103.67  163.3938 < 2.2e-16 ***
## reg3              1    0.96    0.96    1.5068   0.21970
## reg4              1    0.01    0.01    0.0088   0.92535
## home              1   54.45   54.45   85.8285 < 2.2e-16 ***
## s(chld, df = 5)   1  788.22  788.22 1242.3511 < 2.2e-16 ***
## s(hinc, df = 5)   1    0.01    0.01    0.0123   0.91174
## genf              1    0.93    0.93    1.4585   0.22724
## s(wrat, df = 5)   1  169.14  169.14  266.5948 < 2.2e-16 ***
## s(avhv, df = 5)   1   61.24   61.24   96.5292 < 2.2e-16 ***
## s(incm, df = 5)    1   43.79   43.79   69.0119 < 2.2e-16 ***
## s(inca, df = 5)    1    1.90    1.90    2.9964   0.08353 .
## s(plow, df = 5)    1    2.04    2.04    3.2103   0.07325 .
## s(npro, df = 5)    1   71.01   71.01  111.9167 < 2.2e-16 ***
## s(tgif, df = 5)    1   17.91   17.91   28.2284 1.138e-07 ***
## s(lgif, df = 5)    1    2.88    2.88    4.5365   0.03324 *
## s(rgif, df = 5)    1    0.62    0.62    0.9819   0.32178
## s(tdon, df = 5)    1   22.72   22.72   35.8094 2.372e-09 ***
## s(tlag, df = 5)    1  109.10  109.10  171.9609 < 2.2e-16 ***
## s(agif, df = 5)    1    0.92    0.92    1.4492   0.22873
## Residuals       3907 2478.82    0.63
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Anova for Nonparametric Effects
##                  Npar Df Npar Chisq    P(Chi)
## (Intercept)
## reg1
## reg2
## reg3
## reg4
## home
## s(chld, df = 5)       4     251.30 < 2.2e-16 ***
## s(hinc, df = 5)       4     599.56 < 2.2e-16 ***
## genf
## s(wrat, df = 5)       4      75.22 1.776e-15 ***
## s(avhv, df = 5)       4       6.30  0.177869
## s(incm, df = 5)       4      29.25 6.941e-06 ***
## s(inca, df = 5)       4      26.18 2.913e-05 ***
## s(plow, df = 5)       4      16.63  0.002284 **
## s(npro, df = 5)       4       7.75  0.101320
## s(tgif, df = 5)       4      10.17  0.037640 *
## s(lgif, df = 5)       4       5.75  0.218428
## s(rgif, df = 5)       4       3.08  0.543850
## s(tdon, df = 5)       4     143.14 < 2.2e-16 ***
## s(tlag, df = 5)       4      36.78 1.997e-07 ***
## s(agif, df = 5)       4       3.58  0.465298
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#remove spline for linear predictors
model.gam2 <- gam(donr ~ reg1 + reg2 + reg3 + reg4 + home + s(chld,df=5) + s(hinc,df=5) + genf
+ s(wrat,df=5) + avhv + s(incm,df=5) + s(inca,df=5) + s(plow,df=5) + npro + s(tgif,df=5) + lgif +
rgif + s(tdon,df=5) + s(tlag,df=5) + agif, data.train.std.c, family=binomial)
```

```
summary(model.gam2)
##
## Call: gam(formula = donr ~ reg1 + reg2 + reg3 + reg4 + home + s(chld,
##     df = 5) + s(hinc, df = 5) + genf + s(wrat, df = 5) + avhv +
##     s(incm, df = 5) + s(inca, df = 5) + s(plow, df = 5) + npro +
##     s(tgif, df = 5) + lgif + rgif + s(tdon, df = 5) + s(tlag,
##     df = 5) + agif, family = binomial, data = data.train.std.c)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -3.142273 -0.131625  0.002241  0.240759  2.909445
##
## (Dispersion Parameter for binomial family taken to be 1)
##
##     Null Deviance: 5522.988 on 3983 degrees of freedom
## Residual Deviance: 1542.91 on 3926.999 degrees of freedom
## AIC: 1656.912
##
## Number of Local Scoring Iterations: 10
##
## Anova for Parametric Effects
##                   Df  Sum Sq Mean Sq   F value    Pr(>F)
## reg1               1    1.47    1.47    2.3158   0.12815
## reg2               1  104.73  104.73  165.2480 < 2.2e-16 ***
## reg3               1    0.99    0.99    1.5571   0.21216
## reg4               1    0.00    0.00    0.0019   0.96487
## home               1   53.03   53.03   83.6731 < 2.2e-16 ***
## s(chld, df = 5)    1  780.09  780.09 1230.8992 < 2.2e-16 ***
## s(hinc, df = 5)    1    0.01    0.01    0.0136   0.90733
## genf               1    0.97    0.97    1.5232   0.21721
## s(wrat, df = 5)    1  167.23  167.23  263.8786 < 2.2e-16 ***
## avhv               1   57.87   57.87   91.3060 < 2.2e-16 ***
## s(incm, df = 5)    1   41.81   41.81   65.9678 6.078e-16 ***
## s(inca, df = 5)    1    0.73    0.73    1.1557   0.28243
## s(plow, df = 5)    1    2.83    2.83    4.4686   0.03459 *
## npro               1   68.96   68.96  108.8102 < 2.2e-16 ***
## s(tgif, df = 5)    1   15.78   15.78   24.9056 6.282e-07 ***
## lgif               1    0.76    0.76    1.1999   0.27342
## rgif               1    0.23    0.23    0.3693   0.54341
## s(tdon, df = 5)    1   20.38   20.38   32.1554 1.526e-08 ***
## s(tlag, df = 5)    1  107.42  107.42  169.5032 < 2.2e-16 ***
## agif               1    1.73    1.73    2.7276   0.09871 .
## Residuals       3927 2488.75    0.63
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Anova for Nonparametric Effects
##                Npar Df Npar Chisq    P(Chi)
## (Intercept)
## reg1
## reg2
## reg3
## reg4
## home
## s(chld, df = 5)      4     252.77 < 2.2e-16 ***
## s(hinc, df = 5)      4     590.91 < 2.2e-16 ***
## genf
## s(wrat, df = 5)      4      74.40 2.665e-15 ***
## avhv
## s(incm, df = 5)      4      25.72 3.599e-05 ***
## s(inca, df = 5)      4      25.59 3.819e-05 ***
## s(plow, df = 5)      4      18.02  0.001223 **
## npro
## s(tgif, df = 5)      4      12.14  0.016313 *
## lgif
## rgif
## s(tdon, df = 5)      4     140.96 < 2.2e-16 ***
## s(tlag, df = 5)      4      37.76 1.258e-07 ***
## agif
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#remove non-significant predictors
model.gam3 <- gam(donr ~ reg1 + reg2 + reg3 + reg4 + home + s(chld,df=5) + s(hinc,df=5) +
s(wrat,df=5) + avhv + s(incm,df=5) + s(inca,df=5) + s(plow,df=5) + npro + s(tgif,df=5) +
s(tdon,df=5) + s(tlag,df=5), data.train.std.c, family=binomial)
summary(model.gam3)
##
## Call: gam(formula = donr ~ reg1 + reg2 + reg3 + reg4 + home + s(chld,
##     df = 5) + s(hinc, df = 5) + s(wrat, df = 5) + avhv + s(incm,
##     df = 5) + s(inca, df = 5) + s(plow, df = 5) + npro + s(tgif,
##     df = 5) + s(tdon, df = 5) + s(tlag, df = 5), family = binomial,
##     data = data.train.std.c)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -3.154742 -0.132956  0.002474  0.240161  2.854841
##
## (Dispersion Parameter for binomial family taken to be 1)
##
```

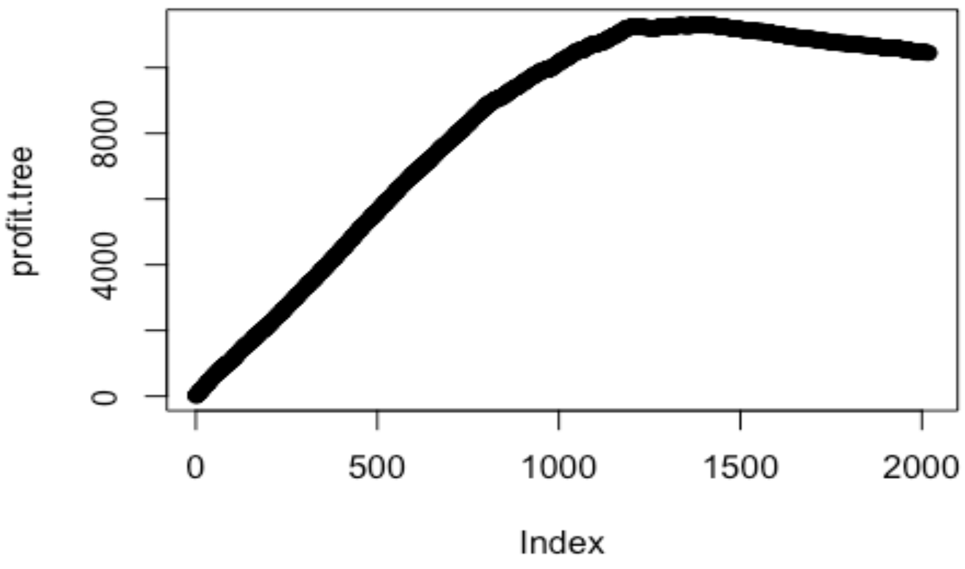
```
##     Null Deviance: 5522.988 on 3983 degrees of freedom
## Residual Deviance: 1546.558 on 3930.999 degrees of freedom
## AIC: 1652.56
##
## Number of Local Scoring Iterations: 10
##
## Anova for Parametric Effects
##                   Df  Sum Sq Mean Sq   F value    Pr(>F)
## reg1               1    1.30    1.30    2.0499   0.15229
## reg2               1  104.57  104.57  165.5169 < 2.2e-16 ***
## reg3               1    0.90    0.90    1.4176   0.23387
## reg4               1    0.00    0.00    0.0001   0.99359
## home               1   53.23   53.23   84.2536 < 2.2e-16 ***
## s(chld, df = 5)    1  783.32  783.32 1239.8088 < 2.2e-16 ***
## s(hinc, df = 5)    1    0.03    0.03    0.0535   0.81703
## s(wrat, df = 5)    1  166.18  166.18  263.0252 < 2.2e-16 ***
## avhv               1   57.29   57.29   90.6775 < 2.2e-16 ***
## s(incm, df = 5)    1   41.68   41.68   65.9752 6.054e-16 ***
## s(inca, df = 5)    1    0.71    0.71    1.1183   0.29035
## s(plow, df = 5)    1    2.90    2.90    4.5958   0.03211 *
## npro               1   71.05   71.05  112.4487 < 2.2e-16 ***
## s(tgif, df = 5)    1   14.98   14.98   23.7154 1.161e-06 ***
## s(tdon, df = 5)    1   19.79   19.79   31.3284 2.327e-08 ***
## s(tlag, df = 5)    1  108.78  108.78  172.1725 < 2.2e-16 ***
## Residuals       3931 2483.63    0.63
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                 Npar Df Npar Chisq    P(Chi)
## (Intercept)
## reg1
## reg2
## reg3
## reg4
## home
## s(chld, df = 5)       4     256.12 < 2.2e-16 ***
## s(hinc, df = 5)       4     591.54 < 2.2e-16 ***
## s(wrat, df = 5)       4      74.05 3.109e-15 ***
## avhv
## s(incm, df = 5)       4      25.47 4.037e-05 ***
## s(inca, df = 5)       4      25.89 3.323e-05 ***
## s(plow, df = 5)       4      18.14  0.001157 **
## npro
## s(tgif, df = 5)       4      11.39  0.022518 *
## s(tdon, df = 5)       4     140.41 < 2.2e-16 ***
## s(tlag, df = 5)       4      37.90 1.177e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

post.valid.gam <- predict(model.gam3, data.valid.std.c, type="response") # n.valid.c post probs
# calculate ordered profit function using average donation = $14.50 and mailing cost = $2
profit.gam <- cumsum(14.5*c.valid[order(post.valid.gam, decreasing=T)]-2)
plot(profit.gam) # see how profits change as more mailings are made
```



```
n.mail.valid <- which.max(profit.gam) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.gam)) # report number of mailings and maximum profit
## [1]  1237 11939

cutoff.gam <- sort(post.valid.gam, decreasing=T)[n.mail.valid+1]
chat.valid.gam <- ifelse(post.valid.gam>cutoff.gam, 1, 0) # mail to everyone above the cutoff
table(chat.valid.gam, c.valid) # classification table
#             c.valid
#chat.valid.gam    0   1
#            0 776   5
#            1 243 994
# check n.mail.valid = 243+994 = 1237
# check profit = 14.5*994-2*1237 = 11939
```

Model 6: Regression Tree

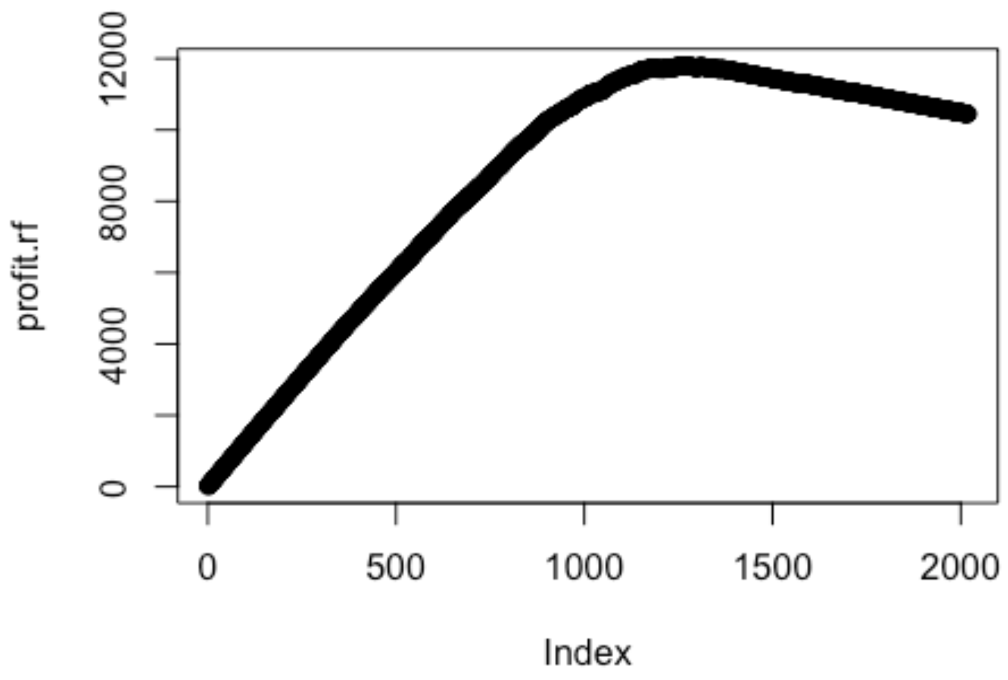
```
library(tree)
tree.charity =tree(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf +
wrat + avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif ,data.train.std.c)
summary(tree.charity)
## Regression tree:
## tree(formula = donr ~ reg1 + reg2 + reg3 + reg4 + home + chld +
##     hinc + I(hinc^2) + genf + wrat + avhv + incm + inca + plow +
##     npro + tgif + lgif + rgif + tdon + tlag + agif, data = data.train.std.c)
## Variables actually used in tree construction:
## [1] "chld"      "home"      "I(hinc^2)" "reg2"      "wrat"      "tlag"
## [7] "tdon"
## Number of terminal nodes:  14
## Residual mean deviance:  0.1088 = 431.8 / 3970
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.92720 -0.05339  0.07277  0.00000  0.07277  0.97560

plot(tree.charity); text(tree.charity ,pretty=0)
tree.pred <- predict (tree.charity ,data.valid.std.c)
profit.tree <- cumsum(14.5*c.valid[order(tree.pred, decreasing=T)]-2)
plot(profit.tree) # see how profits change as more mailings are made
```



```
n.mail.valid <- which.max(profit.tree) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.tree)) # report number of mailings and maximum profit
## [1]  1391 11312

cutoff.tree <- sort(tree.pred, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.tree <- ifelse(tree.pred>cutoff.tree, 1, 0) # mail to everyone above the cutoff
table(chat.valid.tree, c.valid) # classification table
#             c.valid
#chat.valid.tree    0   1
#             0 645  37
#             1 374 962
# check: 374+962 = 1336
# check: 14.5*962 - 2*1336 = 11277
```

Model 7: Random forest (RF)

```
library(randomForest); set.seed(1)
model.rf <- randomForest(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) +
genf + wrat + avhv + npro + rgif + tdon + tlag, data.train.std.c, importance=TRUE)
importance(model.rf) # chld, home, reg2, wrat are among the most influence factors.
##             %IncMSE IncNodePurity
## reg1      27.9991851     15.474573
## reg2     105.8628435     70.787351
## reg3      10.7852658      6.619042
## reg4       8.8438649      5.420146
## home     120.8442012     66.621204
## chld     281.3790597    295.296690
## hinc      21.0531978     28.922596
## I(hinc^2)  74.3146838   102.607126
## genf      -0.8239311      7.022072
## wrat      84.7129643     70.799193
## avhv      18.1592399     63.163948
## npro      24.9536369     62.323803
## rgif       2.3211371     40.715840
## tdon      46.6740572     60.266954
## tlag      33.6730735     44.637939

post.valid.rf <- predict(model.rf, data.valid.std.c) # n.valid post probs

# calculate ordered profit function using average donation = $14.50 and mailing cost = $2
profit.rf <- cumsum(14.5*c.valid[order(post.valid.rf, decreasing=T)]-2)
plot(profit.rf) # see how profits change as more mailings are made
```

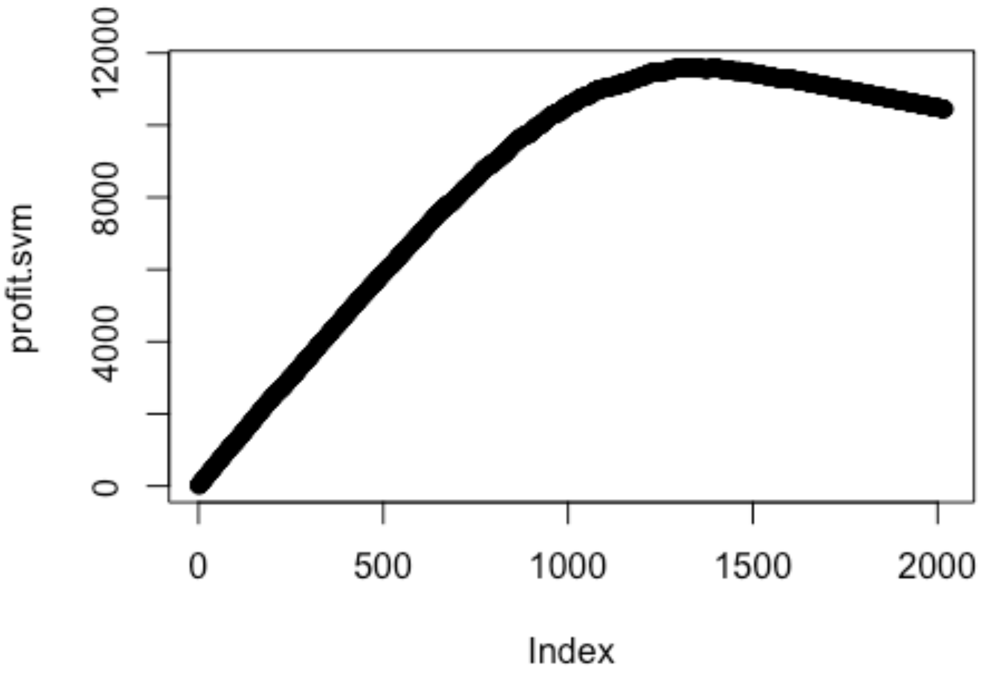


```
n.mail.valid <- which.max(profit.rf) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.rf)) # report number of mailings and maximum profit; 1256, 11799.5
## [1]  1256.0 11799.5
cutoff.rf <- sort(post.valid.rf, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.rf <- ifelse(post.valid.rf > cutoff.rf, 1, 0) # mail to everyone above the cutoff
table(chat.valid.rf, c.valid) # classification table
#             c.valid
#chat.valid.rf    0   1
#            0 750  12
#            1 269 987
# check n.mail.valid = 269+987 = 1256
# check profit = 14.5*987-2*1256 = 11799.5
```

Model 8: Support Vector Machine (SVM)

```
library(e1071)
svm.charity <- svm(donr ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf +
wrat + avhv + incm + inca + plow + npro + tgif + lgif + rgif + tdon + tlag + agif,
data=data.train.std.c, kernel="linear",cost=10)
```

```
summary(svm.charity)
## Call:
## svm(formula = donr ~ reg1 + reg2 + reg3 + reg4 + home + chld +
##     hinc + I(hinc^2) + genf + wrat + avhv + incm + inca + plow +
##     npro + tgif + lgif + rgif + tdon + tlag + agif, data = data.train.std.c,
##     kernel = "linear", cost = 10)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.04761905
##     epsilon:  0.1
##
##
## Number of Support Vectors:  3527

svm.pred <- predict (svm.charity ,data.valid.std.c)
profit.svm <- cumsum(14.5*c.valid[order(svm.pred, decreasing=T)]-2)
plot(profit.svm) # see how profits change as more mailings are made
```

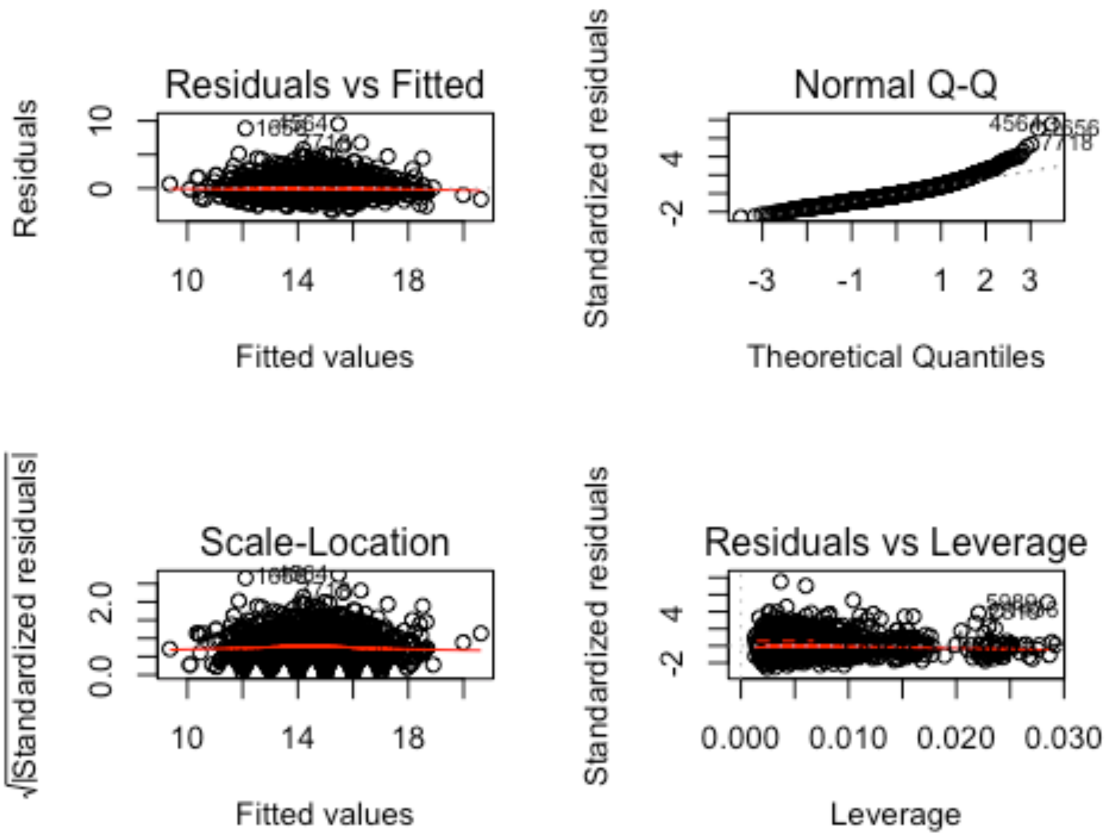


```
n.mail.valid <- which.max(profit.svm) # number of mailings that maximizes profits
c(n.mail.valid, max(profit.svm)) # report number of mailings and maximum profit
## [1]  1314.0 11596.5

cutoff.svm <- sort(svm.pred, decreasing=T)[n.mail.valid+1] # set cutoff based on n.mail.valid
chat.valid.svm <- ifelse(svm.pred>cutoff.svm, 1, 0) # mail to everyone above the cutoff
table(chat.valid.svm, c.valid) # classification table
#             c.valid
#chat.valid.svm     0   1
#             0  686  18
#             1  333 981
# Check 333+981 = 1314
# Check 14.5*981 - 2*1314 = 11596.5
```

Prediction modeling

Model 1: Least squares regression

```
model.ls <- lm(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +
avhv + npro + rgif + tdon + tlag, data.train.std.y)
summary(model.ls) # we can see some predictors are not significant; so I drop them (reg1, reg2,
wrat and tlag); also note that the predictor error and std error will not change much after
dropping them
##
## Call:
## lm(formula = damt ~ reg1 + reg2 + reg3 + reg4 + home + chld +
##     hinc + I(hinc^2) + genf + wrat + avhv + npro + rgif + tdon +
##     tlag, data = data.train.std.y)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1922 -0.8254 -0.1987  0.6055  9.6137
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.23009    0.04664 305.092  < 2e-16 ***
## reg1        -0.04762    0.03935  -1.210  0.22632
## reg2        -0.08167    0.04312  -1.894  0.05838 .
## reg3         0.32672    0.04010   8.148 6.48e-16 ***
## reg4         0.63190    0.04124  15.324  < 2e-16 ***
## home         0.25206    0.06076   4.149 3.48e-05 ***
## chld        -0.59628    0.03921 -15.209  < 2e-16 ***
## hinc         0.50312    0.03964  12.692  < 2e-16 ***
## I(hinc^2)   -0.05300    0.02946  -1.799  0.07215 .
## genf       -0.06498    0.02833  -2.293  0.02193 *
## wrat        0.01691    0.04139   0.409  0.68286
## avhv        0.06684    0.02889   2.313  0.02080 *
## npro        0.16860    0.02874   5.867 5.19e-09 ***
## rgif        1.11487    0.02824  39.484  < 2e-16 ***
## tdon        0.09491    0.03472   2.733  0.00633 **
## tlag        0.03497    0.03079   1.136  0.25618
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.267 on 1979 degrees of freedom
## Multiple R-squared:  0.5753, Adjusted R-squared:  0.5721
## F-statistic: 178.7 on 15 and 1979 DF,  p-value: < 2.2e-16

model.ls <- lm(damt ~ reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + avhv + npro + rgif +
tdon, data.train.std.y)
summary(model.ls)
##
## Call:
## lm(formula = damt ~ reg3 + reg4 + home + chld + hinc + I(hinc^2) +
##     genf + avhv + npro + rgif + tdon, data = data.train.std.y)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2823 -0.8212 -0.1829  0.6087  9.5277
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.20454    0.04221 336.551  < 2e-16 ***
## reg3         0.36560    0.03341  10.942  < 2e-16 ***
## reg4         0.67162    0.03431  19.574  < 2e-16 ***
## home         0.26788    0.06025   4.446 9.24e-06 ***
## chld        -0.61768    0.03700 -16.694  < 2e-16 ***
## hinc         0.50324    0.03954  12.727  < 2e-16 ***
## I(hinc^2)   -0.06439    0.02882  -2.234  0.02558 *
## genf        -0.06545    0.02833  -2.310  0.02099 *
## avhv         0.06883    0.02875   2.394  0.01675 *
## npro         0.17356    0.02861   6.067 1.56e-09 ***
## rgif         1.11448    0.02824  39.465  < 2e-16 ***
## tdon         0.09481    0.03473   2.730  0.00638 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.267 on 1983 degrees of freedom
## Multiple R-squared:  0.5742, Adjusted R-squared:  0.5719
## F-statistic: 243.1 on 11 and 1983 DF,  p-value: < 2.2e-16

pred.valid.ls <- predict(model.ls, data.valid.std.y) # validation predictions
mean((y.valid - pred.valid.ls)^2) # mean prediction error
## [1] 1.880341
sd((y.valid - pred.valid.ls)^2)/sqrt(n.valid.y) # std error
## [1] 0.1694115

# Check assumptions
par(mfrow=c(2,2)); plot(model.ls)
```

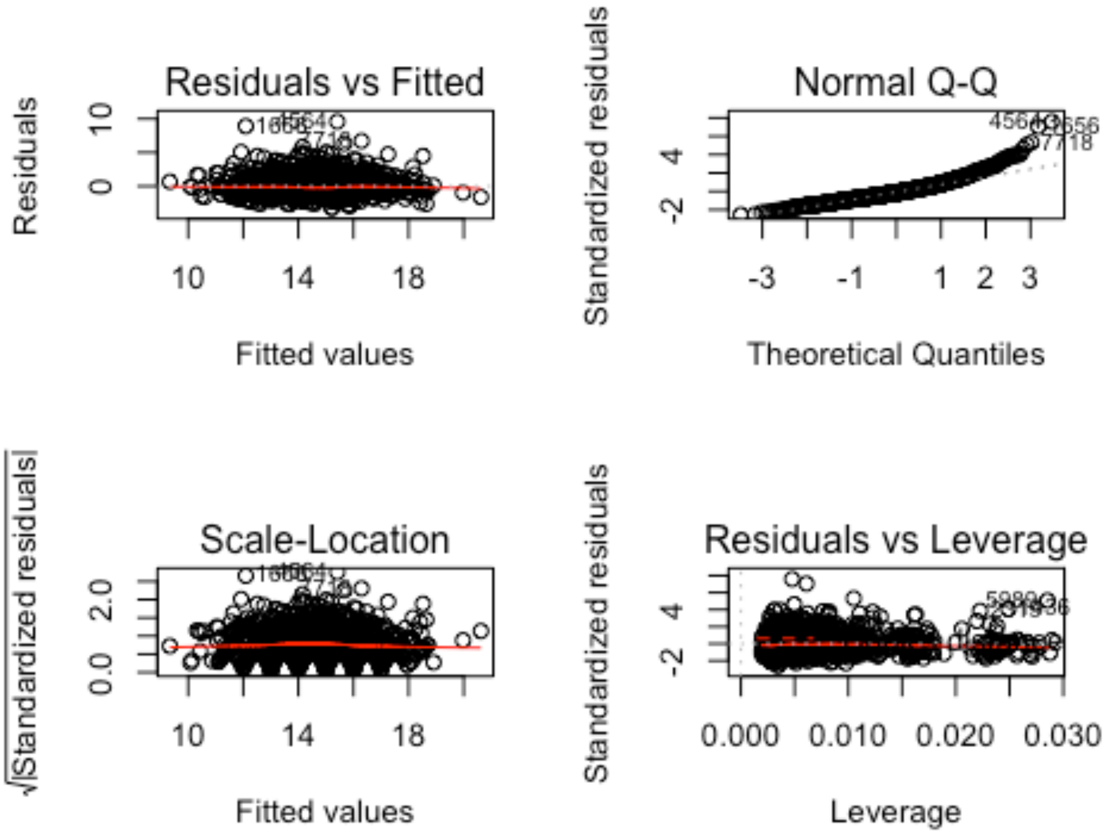


```
library(car); vif(model.ls)
##      reg3      reg4      home      chld      hinc I(hinc^2)      genf
##  1.041498  1.045173  1.027610  1.142138  1.004954  1.083112  1.004724
##      avhv      npro      rgif      tdon
##  1.026368  1.020440  1.003911  1.007183
# all ok around 1 - multicollinearity not an issue
```

Model 2: Stepwise regression using AIC

```
library(MASS)
model.AIC <- lm(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat
+ avhv + npro + rgif + tdon + tlag, data.train.std.y)
step <- stepAIC(model.AIC, direction="both")
step$anova # note that the model with smallest AIC is exactly the same as model 1
summary(step)
##
## Call:
## lm(formula = damt ~ reg2 + reg3 + reg4 + home + chld + hinc +
##     I(hinc^2) + genf + avhv + npro + rgif + tdon, data = data.train.std.y)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2263 -0.8147 -0.1903  0.6028  9.5868
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.21978    0.04354 326.610  < 2e-16 ***
## reg2        -0.04429    0.03117  -1.421  0.15549
## reg3         0.35091    0.03497  10.035  < 2e-16 ***
## reg4         0.65658    0.03590  18.291  < 2e-16 ***
## home         0.25880    0.06058   4.272 2.03e-05 ***
## chld        -0.60302    0.03840 -15.703  < 2e-16 ***
## hinc         0.50505    0.03955  12.769  < 2e-16 ***
## I(hinc^2)   -0.05790    0.02917  -1.985  0.04730 *
## genf        -0.06474    0.02833  -2.285  0.02240 *
## avhv         0.06591    0.02881   2.288  0.02226 *
## npro         0.17017    0.02870   5.929 3.58e-09 ***
## rgif         1.11447    0.02823  39.475  < 2e-16 ***
## tdon         0.09413    0.03472   2.711  0.00676 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.267 on 1982 degrees of freedom
## Multiple R-squared:  0.5747, Adjusted R-squared:  0.5721
## F-statistic: 223.1 on 12 and 1982 DF,  p-value: < 2.2e-16

model.AIC <- lm(damt ~ reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + avhv + npro
+ rgif + tdon, data.train.std.y)
summary(model.AIC)
##
## Call:
## lm(formula = damt ~ reg2 + reg3 + reg4 + home + chld + hinc +
##     I(hinc^2) + genf + avhv + npro + rgif + tdon, data = data.train.std.y)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2263 -0.8147 -0.1903  0.6028  9.5868
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.21978    0.04354 326.610  < 2e-16 ***
## reg2        -0.04429    0.03117  -1.421  0.15549
## reg3         0.35091    0.03497  10.035  < 2e-16 ***
## reg4         0.65658    0.03590  18.291  < 2e-16 ***
## home         0.25880    0.06058   4.272 2.03e-05 ***
## chld        -0.60302    0.03840 -15.703  < 2e-16 ***
## hinc         0.50505    0.03955  12.769  < 2e-16 ***
## I(hinc^2)   -0.05790    0.02917  -1.985  0.04730 *
## genf        -0.06474    0.02833  -2.285  0.02240 *
## avhv         0.06591    0.02881   2.288  0.02226 *
## npro         0.17017    0.02870   5.929 3.58e-09 ***
## rgif         1.11447    0.02823  39.475  < 2e-16 ***
## tdon         0.09413    0.03472   2.711  0.00676 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.267 on 1982 degrees of freedom
## Multiple R-squared:  0.5747, Adjusted R-squared:  0.5721
## F-statistic: 223.1 on 12 and 1982 DF,  p-value: < 2.2e-16

pred.valid.aic <- predict(model.AIC, data.valid.std.y) # validation predictions
mean((y.valid - pred.valid.aic)^2) # mean prediction error
## [1] 1.871926
sd((y.valid - pred.valid.aic)^2)/sqrt(n.valid.y) # std error
## [1] 0.1687491
```

```
# Check assumptions
par(mfrow=c(2,2)); plot(model.AIC)
```



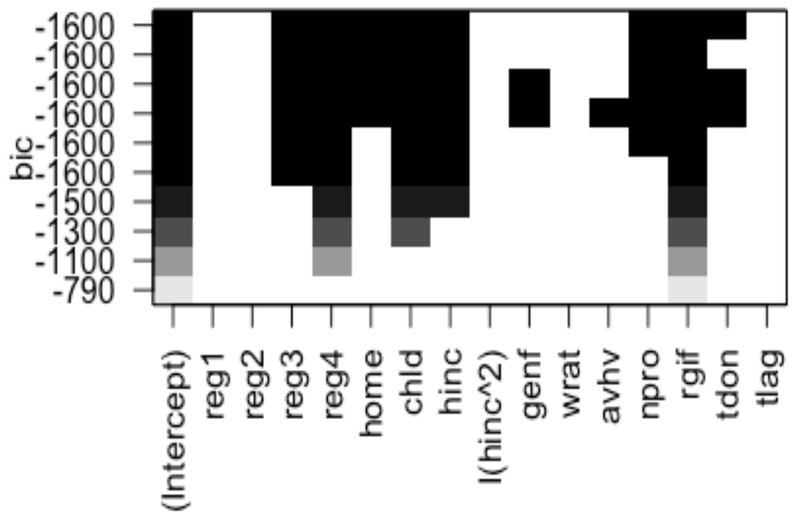
```
library(car); vif(model.AIC)
##      reg2      reg3      reg4      home      chld      hinc I(hinc^2)
##  1.341120  1.141290  1.144626  1.039159  1.230919  1.005995  1.110321
##      genf      avhv      npro      rgif      tdon
##  1.005032  1.031588  1.027540  1.003911  1.007373
# all ok around 1 - multicollinearity not an issue
```

Model 3: Best subset selection using BIC

```
library(leaps)
BIC <- regsubsets(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf +
wrat + avhv + npro + rgif + tdon + tlag, data.train.std.y, nvmax=10)
summary(BIC)
## Subset selection object
## Call: regsubsets.formula(damt ~ reg1 + reg2 + reg3 + reg4 + home +
##     chld + hinc + I(hinc^2) + genf + wrat + avhv + npro + rgif +
##     tdon + tlag, data.train.std.y, nvmax = 10)
## 15 Variables  (and intercept)
##           Forced in Forced out
## reg1          FALSE      FALSE
## reg2          FALSE      FALSE
## reg3          FALSE      FALSE
## reg4          FALSE      FALSE
## home          FALSE      FALSE
## chld          FALSE      FALSE
## hinc          FALSE      FALSE
## I(hinc^2)     FALSE      FALSE
## genf          FALSE      FALSE
## wrat          FALSE      FALSE
## avhv          FALSE      FALSE
## npro          FALSE      FALSE
## rgif          FALSE      FALSE
## tdon          FALSE      FALSE
## tlag          FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           reg1 reg2 reg3 reg4 home chld hinc I(hinc^2) genf wrat avhv npro
## 1  ( 1 )  " "  " "  " "  " "  " "  " "  " "  " "       " "  " "  " "  " "
## 2  ( 1 )  " "  " "  " "  "*"  " "  " "  " "  " "       " "  " "  " "  " "
## 3  ( 1 )  " "  " "  " "  "*"  " "  "*"  " "  " "       " "  " "  " "  " "
## 4  ( 1 )  " "  " "  " "  "*"  " "  "*"  "*"  " "       " "  " "  " "  " "
## 5  ( 1 )  " "  " "  "*"  "*"  " "  "*"  "*"  " "       " "  " "  " "  " "
## 6  ( 1 )  " "  " "  "*"  "*"  " "  "*"  "*"  " "       " "  " "  " "  "*"
## 7  ( 1 )  " "  " "  "*"  "*"  "*"  "*"  "*"  " "       " "  " "  " "  "*"
## 8  ( 1 )  " "  " "  "*"  "*"  "*"  "*"  "*"  " "       " "  " "  " "  "*"
## 9  ( 1 )  " "  " "  "*"  "*"  "*"  "*"  "*"  " "       "*"  " "  " "  "*"
## 10  ( 1 ) " "  " "  "*"  "*"  "*"  "*"  "*"  " "       "*"  " "  "*"  "*"
##           rgif tdon tlag
## 1  ( 1 )  "*"  " "  " "
## 2  ( 1 )  "*"  " "  " "
## 3  ( 1 )  "*"  " "  " "
## 4  ( 1 )  "*"  " "  " "
## 5  ( 1 )  "*"  " "  " "
## 6  ( 1 )  "*"  " "  " "
## 7  ( 1 )  "*"  " "  " "
## 8  ( 1 )  "*"  "*"  " "
## 9  ( 1 )  "*"  "*"  " "
## 10  ( 1 ) "*"  "*"  " "
```

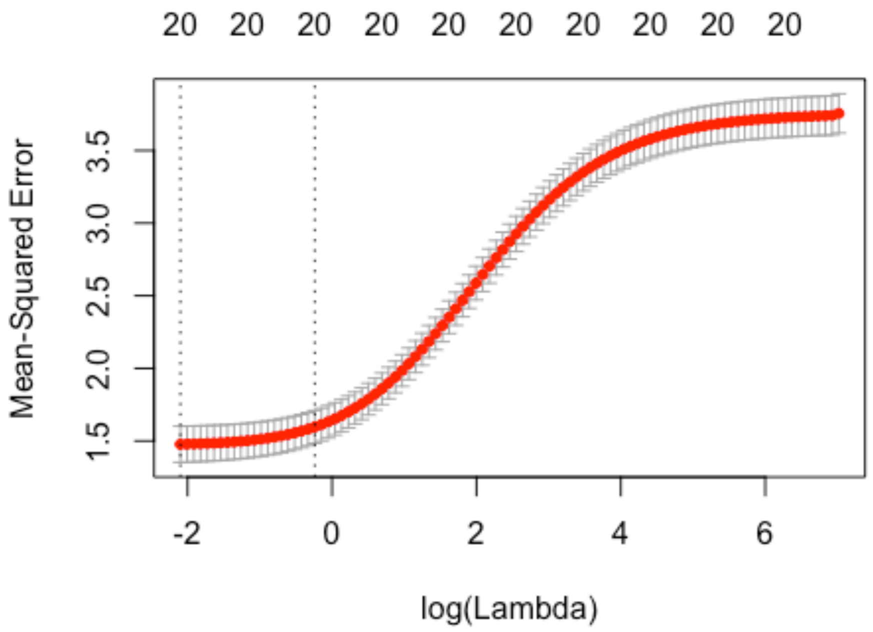
```
# determine the number of variables with the lowest BIC value
which.min(summary(BIC)$bic)
## [1] 8
```

```
coef(BIC, id=8)
## (Intercept)       reg3       reg4       home       chld       hinc
##  14.2031999  0.3767497  0.6793971  0.2554029 -0.5890471  0.4973662
##       npro       rgif       tdon
##   0.1666014  1.1119596  0.0977098

# Create the function to use predict() with regsubsets
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}

pred.BIC <- predict(BIC, data.valid.std.y, id=8)  # validation predictions
mean((y.valid-pred.BIC)^2) # mean prediction error 1.895
## [1] 1.895405
sd((y.valid-pred.BIC)^2)/sqrt(n.valid.y) # std error 0.170
## [1] 0.1697195
plot(BIC, scale='bic')
```



Model 4: Ridge regression using 10-fold cross-validation

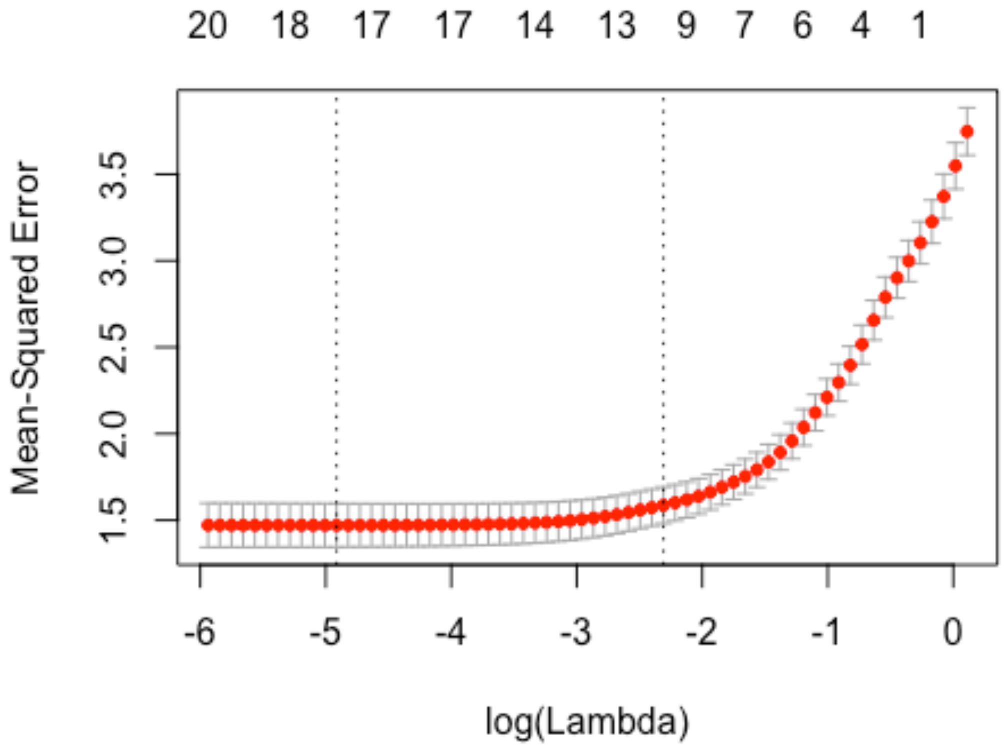
```
library(glmnet) # The package "glmnet" was used to perform Ridge Regression and Lasso
set.seed(2015)
x <- as.matrix(data.train.std.y[,1:20]); y <- as.matrix(data.train.std.y[,21])
valid.x <- as.matrix(data.valid.std.y[,1:20])
valid.y <- as.matrix(data.valid.std.y[,21])
cv.out <- cv.glmnet(x, y, alpha=0, nfolds=10)
bestlambda <- cv.out$lambda.1se; bestlambda
## [1] 0.7873924
ridge <- glmnet(x, y, alpha=0)
pred.ridge <- predict(ridge, s=bestlambda, valid.x)
mean((valid.y-pred.ridge)^2) # mean prediction error 1.832
## [1] 1.832544
sd((valid.y-pred.ridge)^2)/sqrt(n.valid.y) # std error 0.173
## [1] 0.1728453
plot(cv.out)
```



Model 5: Lasso regression using 10-fold cross-validation

```
set.seed(2015)
cv.out <- cv.glmnet(x, y, alpha=1, nfolds=10)
bestlambda <- cv.out$lambda.1se; bestlambda
## [1] 0.09935765
```

```
lasso <- glmnet(x, y, alpha=1)
pred.lasso <- predict(lasso, s=bestlambda, valid.x, exact=T)
mean((valid.y-pred.lasso)^2) # mean prediction error 1.819
## [1] 1.819467
sd((valid.y-pred.lasso)^2)/sqrt(n.valid.y) # std error 0.164
## [1] 0.1641376
plot(cv.out)
```



Model 6: Principal Components Regression

```
library(pls)
set.seed(2015)
pcr.fit=pcr(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +
avhv + npro + rgif + tdon + tlag, data=data.train.std.y, validation="CV")

summary(pcr.fit)
## Data:    X dimension: 1995 15
##  Y dimension: 1995 1
## Fit method: svdpc
## Number of components considered: 15
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.937    1.908    1.873    1.813    1.726    1.669    1.624
## adjCV        1.937    1.907    1.876    1.839    1.706    1.675    1.651
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       1.370    1.361    1.345     1.342     1.327     1.304     1.293
## adjCV    1.362    1.358    1.344     1.342     1.327     1.304     1.293
##        14 comps  15 comps
## CV        1.293     1.275
## adjCV     1.293     1.274
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X      14.045   23.700    32.45     41.1    49.57    57.68    65.51
## damt    2.994    6.525    11.45     23.8    26.38    28.90    50.82
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       72.68    79.26     84.78     89.24     93.42     96.75     98.54
## damt    51.18    52.40     52.67     53.89     55.40     56.19     56.25
##       15 comps
## X       100.00
## damt     57.53

pred.valid.pcr <- predict(pcr.fit, data.valid.std.y,ncomp=15)
mean((y.valid - pred.valid.pcr)^2) # mean prediction error
## [1] 1.869997
sd((y.valid - pred.valid.pcr)^2)/sqrt(n.valid.y) # std error
## [1] 0.1689471
```

Model 7: Partial Least Squares

```
set.seed(2015)
pls.fit=plsr(damt ~ reg1 + reg2 + reg3 + reg4 + home + chld + hinc + I(hinc^2) + genf + wrat +
avhv + npro + rgif + tdon + tlag, data=data.train.std.y, validation="CV")

summary(pls.fit)
## Data:    X dimension: 1995 15
##  Y dimension: 1995 1
## Fit method: kernelpls
## Number of components considered: 15
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.937    1.322    1.289    1.280    1.277    1.275    1.275
## adjCV        1.937    1.321    1.289    1.279    1.276    1.274    1.274
##       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      1.275    1.275    1.275     1.275     1.275     1.275     1.275
## adjCV   1.274    1.274    1.274     1.274     1.274     1.274     1.274
##       14 comps  15 comps
## CV       1.275     1.275
## adjCV    1.274     1.274
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       9.113    21.22    27.07    33.35    37.72    44.60    50.08
## damt   54.082    56.40    57.19    57.41    57.51    57.53    57.53
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       55.65    59.89     65.15     71.39     77.21     84.00     91.64
## damt    57.53    57.53     57.53     57.53     57.53     57.53     57.53
##       15 comps
## X       100.00
## damt     57.53

pred.valid.pls <- predict(pls.fit, data.valid.std.y,ncomp=5)
mean((y.valid - pred.valid.pls)^2) # mean prediction error
## [1] 1.873125
sd((y.valid - pred.valid.pls)^2)/sqrt(n.valid.y) # std error
## [1] 0.1693118
```