PHP 2550: HW #5

Blain Morin

November 26, 2018

Data Cleaning

In this assignment, we compare step, ridge and lasso models using bootstraping and cross validation. Our data comes from two kidney disease studies. Our aim is to find the variables that are predictive of glomerular filtration rate (GFR).

The raw data contain 2749 observations of 57 variables. Some of the variables, such as LDL cholesterol, had over 50% missing data. For our analysis we decided to exclude variables with more than 10% missing observations. Excluding these variables comes at a cost of some predictive power, but allows us to use more complete cases in our regressions. We then filtered out rows that had missing data (455 cases). Removing these data makes a strong assumption that they were missing completely at random. For further analysis, we could use multiple imputation to estimate these missing values.

Overall, we were left with 2294 observations of 31 variables:

Table 1: Summary Statistics

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
X1	2,294	1,435.800	780.993	1	773	2,112.8	2,749
WEIGHT	2,294	85.140	19.495	38.102	71.668	96.616	177.000
BMI	2,294	28.828	6.003	14.694	24.631	32.091	58.462
GFR	2,294	53.441	27.664	5.867	31.890	70.943	169.000
SUN	2,294	25.178	14.558	0	15	32	107
SCR	2,294	1.929	1.036	0.407	1.278	2.300	8.300
BLACK	2,294	0.515	0.500	0	0	1	1
HEIGHT	2,294	171.775	10.033	132.080	164.862	179.300	203.800
AGE	2,294	51.193	12.675	18.000	42.188	61.664	86.000
FEMALE	2,294	0.385	0.487	0	0	1	1
cys	2,294	1.706	0.803	0.460	1.090	2.150	5.280
DONOR	2,294	0.039	0.194	0	0	0	1
Tx	2,294	0.095	0.294	0	0	0	1
Diabetes	2,294	0.043	0.203	0	0	0	1
c.scr	2,294	1.804	0.973	0.401	1.167	2.092	8.061
c.cys	2,294	1.706	0.803	0.460	1.090	2.150	5.280
MDRD	2,294	0.283	0.451	0	0	1	1
AASK	2,294	0.478	0.500	0	0	1	1
CSG	2,294	0.000	0.000	0	0	0	0
gronigen	2,294	0.083	0.276	0	0	0	1
gronigendonor	2,294	0.010	0.102	0	0	0	1
crisp	2,294	0.056	0.230	0	0	0	1
ccfp	2,294	0.021	0.145	0	0	0	1
ccpdonor	2,294	0.027	0.162	0	0	0	1
drds	2,294	0.000	0.000	0	0	0	0
mayo	2,294	0.039	0.193	0	0	0	1
mayodonor	2,294	0.002	0.042	0	0	0	1
rass1	2,294	0.000	0.000	0	0	0	0
..2	2,294	0.432	0.449	−0.777	0.086	0.765	1.664
..2_1	2,294	0.473	0.467	−0.913	0.155	0.738	2.087

We see from the above table that the variables CSG, drds, and rass1 contain only 0s. Since they have no variance, we excluded them from our model selection. The "X1" and patient ID columns were removed because they are unique identifiers for each row. We also removed the "..2" and "..2_1" columns because there was no documentation explaining what these were. We then checked for collinearity using the alias() function on a fully specified model:

Table 2: Collinearity Check

	c.cys	mayo	mayodonor
(Intercept)	0	1	0
WEIGHT	0	0	0
BMI	0	0	0
SUN	0	0	0
SCR	0	0	0
BLACK	0	0	0
HEIGHT	0	0	0
AGE	0	0	0
FEMALE	0	0	0
cys	1	0	0
DONOR	0	-1	1
Tx	0	0	0
Diabetes	0	0	0
c.scr	0	0	0
MDRD	0	-1	0
AASK	0	-1	0
gronigen	0	-1	0
gronigendonor	0	0	-1
crisp	0	-1	0
ccfp	0	-1	0
ccpdonor	0	0	-1

In the table above, non zero values indicate that there is collinearity between the corresponding variables. We thus excluded c.cys, mayodonor, and mayo. Also, from intuition we decided to omit height and weight because we felt these variables were captured by BMI. We also eliminated all of the alternate "donor" variables and keep only the main "donor" column. Our final cleaned data set contains 2294 observations of 16 variables.

Question 1: Bootstrap Step Regression

To start, we use the stepAIC function to do forward and backward regression on the cleaned data set. Here are the regression results:

Table 3: StepAIC on Observed Data

	Dependent variable:
	GFR
SUN	-0.142^{***}
	(0.031)
SCR	-4.262^{***}
	(0.630)
AGE	-0.249^{***}
	(0.026)
FEMALE	-5.156^{***}
	(0.636)
cys	-16.212^{***}
	(0.839)
DONOR	32.818^{***}
	(1.682)
Tx	-5.886^{***}
	(1.947)
Diabetes	3.791^{**}
	(1.511)
AASK	9.043^{***}
	(0.753)
gronigen	12.950^{***}
	(2.060)
crisp	26.537^{***}
	(1.531)
ccfp	9.170^{***}
	(2.108)
Constant	99.619^{***}
	(1.734)
Observations	2,294
R^2	0.746
Adjusted R^2	0.744
Residual Std. Error	13.983 (df = 2281)
F Statistic	557.785^{***} (df = 12; 2281)
Note:	*p<0.1; **p<0.05; ***p<0.01

We see that the backward and forward step regression selects 12 variables and that the r-squared for the

regression is .746. An r-squared of .746 means that the model accounts for 74.6% of the variability in GFR. However, this r-squared value only tells us how well the model fits within the data in which it was trained. We want to know how well the model would fit on a testing set. In other words, we want to adjust for the optimism in the training fit. To do this, we simulate new data sets using the bootstrap.

To simulate data, we take random samples from the original data set until we have a data frame that is the same size as the original data (the bootstrap sample). We then run stepAIC on the bootstrap sample to get the training r squared. We can use the observations not selected into the bootstrap sample as a testing set, which allows to calculate the test r-squared. The difference between the training and test r squareds is the optimism. We can repeat this process over and over and use the average optimism as an estimate for the true optimism of the fitting process. The underlying assumption of this method is that our data is a truly random sample from the population.

We bootstrapped 1000 samples and calculated an average optimism of .009.

We can thus calculate an estimate for our model's test r-squared:

$$\hat{r}^2_{test} = r^2_{train} - \hat{optimism} = .746 - .009 = .737$$

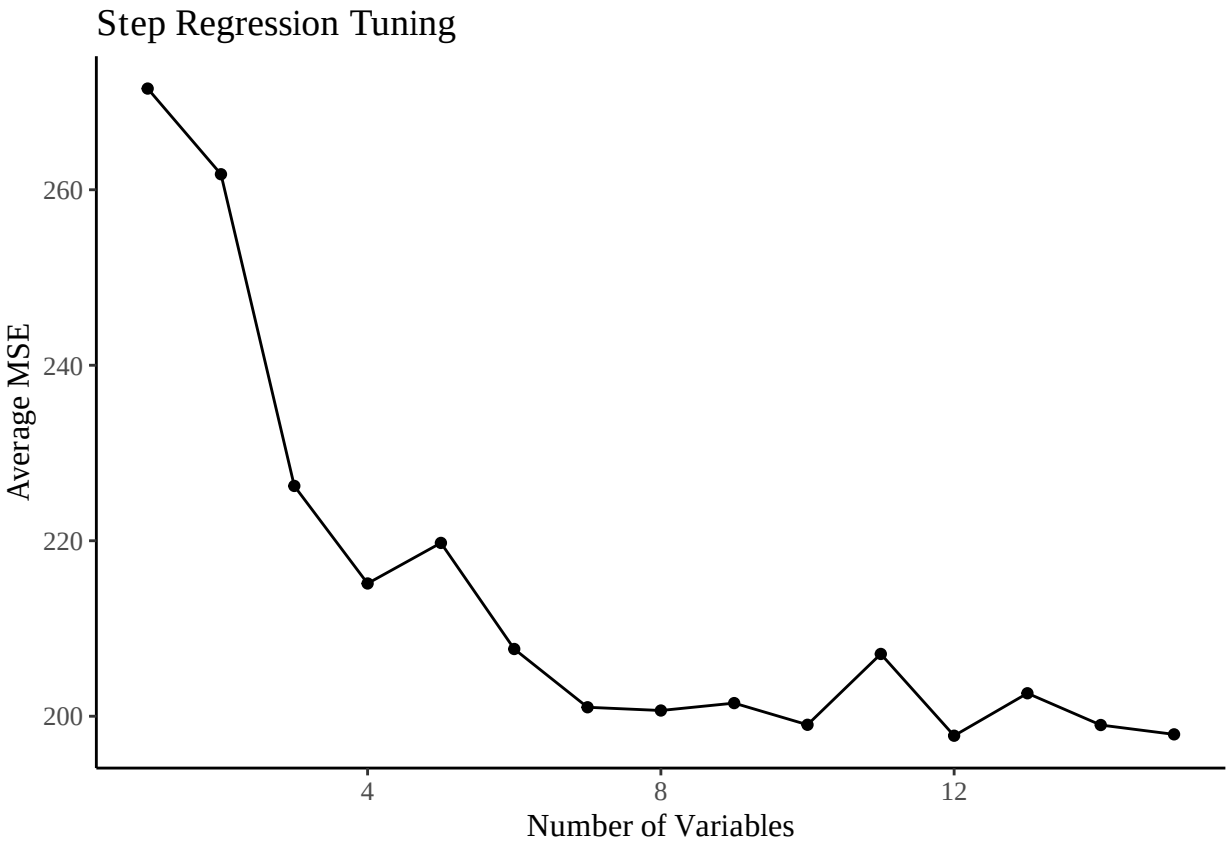
Question 2 and 3:

We use cross validation with 10 folds to tune the parameters for step, ridge, and lasso regression. Like in Question 1, we calculate optimism by taking the difference between the training and testing r-squared values. We use the average optimism over the 10 folds as an estimate for the true optimism of the fitting process.

We run the regression using the tuned parameters found in cross validation on the whole data set to get the training r-squared. We then subtract the estimated optimism from the training r-squared to obtain an estimate of the test r-squared.

a.) Step Regression Cross Validation:

For forward and backward step regression, the parameter we would like to tune is the number of variables in the final model. Using the cross validation procedure described above, here are the average mean squared errors from the testing set versus the number of variables included in the model:



We see from the graph above that the backward and forward step regression with 12 variables has the lowest average mean squared error. Here are the step regression results for a size 12 model on the whole data set:

Table 4: Final Step Model

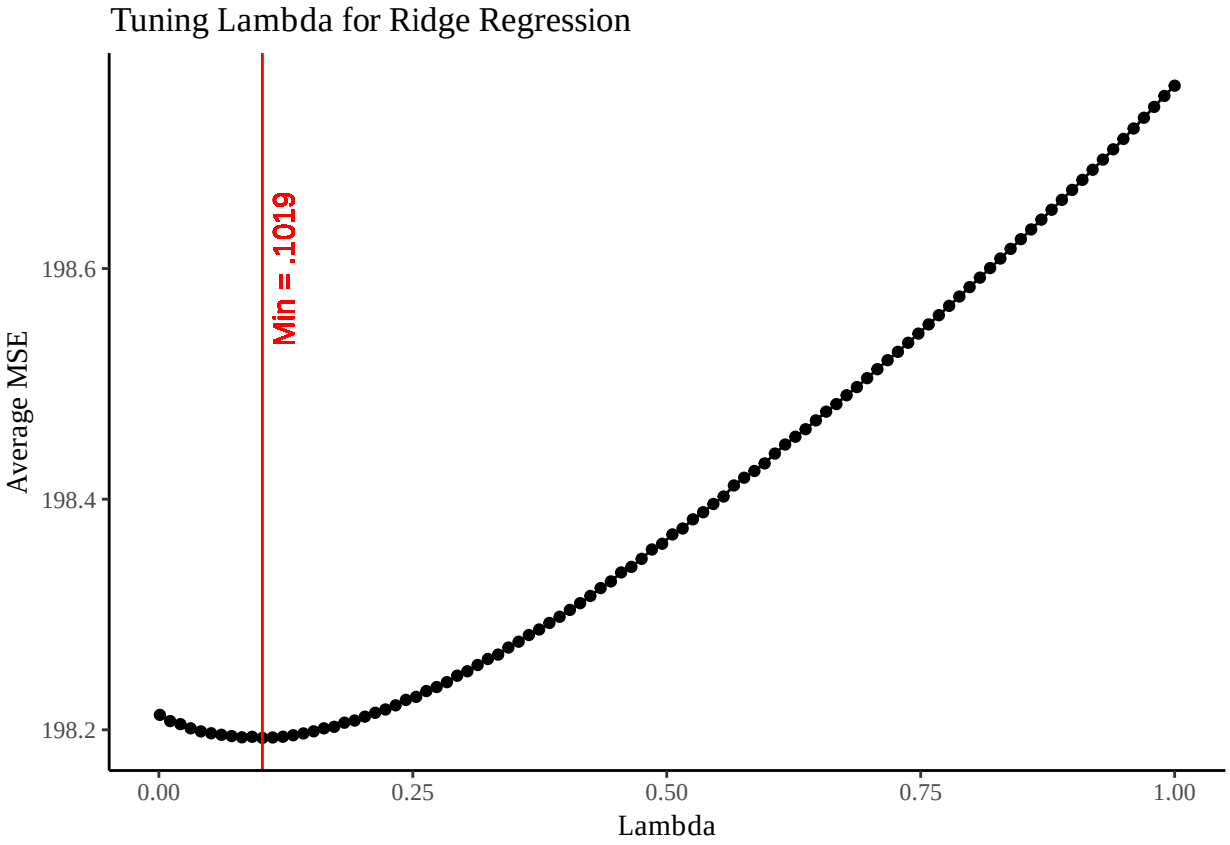
	Dependent variable:
	GFR
SUN	-0.142^{***}
	(0.031)
SCR	-4.262^{***}
	(0.630)
AGE	-0.249^{***}
	(0.026)
FEMALE	-5.156^{***}
	(0.636)
cys	-16.212^{***}
	(0.839)
DONOR	32.818^{***}
	(1.682)
Tx	-5.886^{***}
	(1.947)
Diabetes	3.791^{**}
	(1.511)
AASK	9.043^{***}
	(0.753)
gronigen	12.950^{***}
	(2.060)
crisp	26.537^{***}
	(1.531)
ccfp	9.170^{***}
	(2.108)
Constant	99.619^{***}
	(1.734)
Observations	2,294
R^2	0.746
Adjusted R^2	0.744
Residual Std. Error	13.983 (df = 2281)
F Statistic	557.785^{***} (df = 12; 2281)
Note:	*p<0.1; **p<0.05; ***p<0.01

To estimate the test r-squared of our final model, we subtract the average optimism for a fit of size 12:

$$\hat{r}^2_{test} = r^2_{train} - \hat{optimism} = .746 - .003 = .743$$

b.) Ridge Regression Cross Validation:

For ridge regression, we are tuning the penalization parameter lambda. Using the cross validation procedure described above, here are the average mean squared errors for lambda values between .001 and 1 (we checked lambda values up to 10000, but the minimum was found in this range):



We see from the plot above that the lambda value that minimizes the average MSE is .1019. Using this lambda, we reran the ridge regression on the entire data set. Here are the beta coefficients for the final ridge model:

Table 5: Final Ridge Model

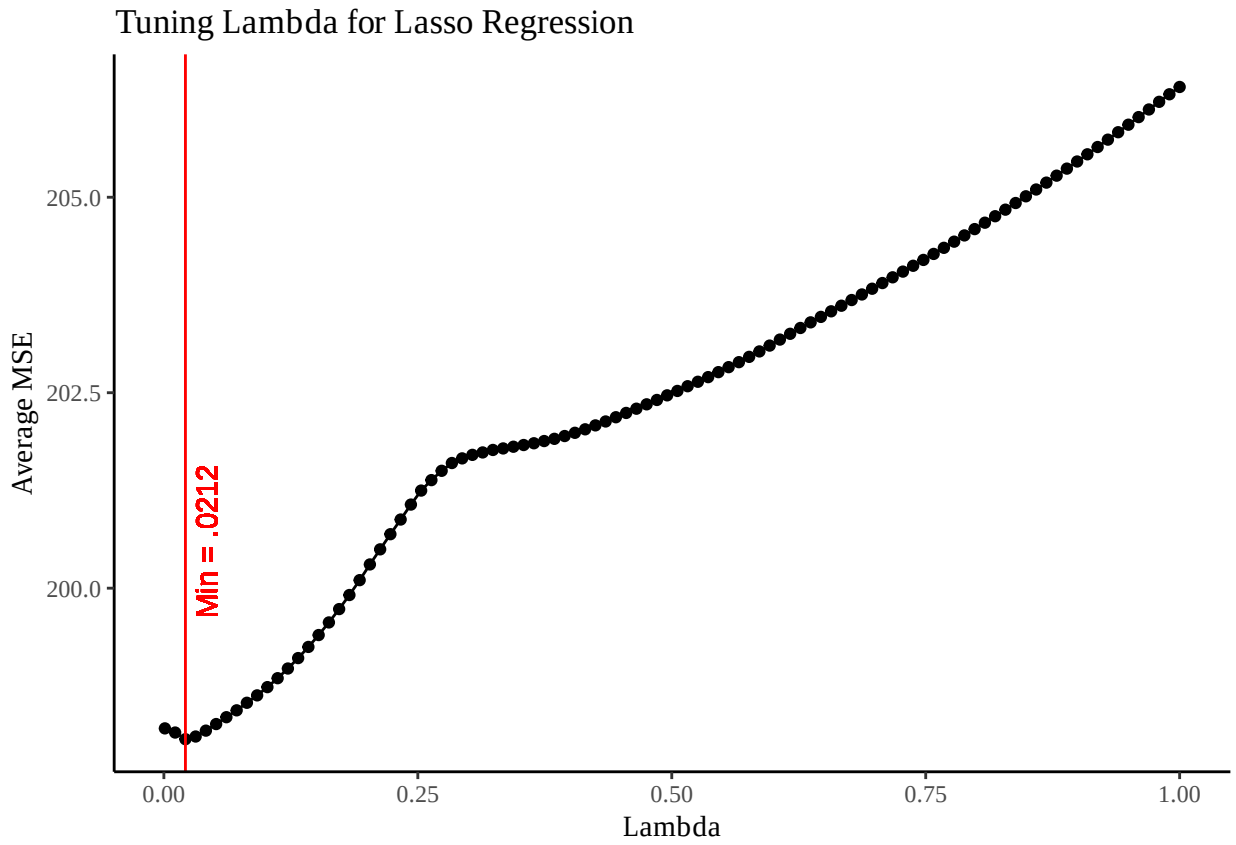
Betas	Coef
(Intercept)	98.854
BMI	0.017
SUN	-0.146
SCR	-4.349
BLACK	0.621
AGE	-0.248
FEMALE	-5.164
cys	-16.024
DONOR	32.923
Tx	-5.606
Diabetes	3.633
MDRD	0.263
AASK	8.603
gronigen	12.875
crisp	26.676
ccfp	9.340

Again, we calculate the estimated test r-squared by subtracting the optimism of the fitting process from the r-squared of the training model:

$$\hat{r}^2_{test} = r^2_{train} - \hat{optimism} = .746 - .006 = .740$$

c.) Lasso Regression Cross Validation:

For lasso regression, we are also tuning the penalization parameter lambda. Using the cross validation procedure described above, here are the average mean squared errors for lambda values between .001 and 1 (we checked lambda values up to 10000, but the minimum was found in this range):



We see from the plot above that the lambda value that minimizes the average MSE is .0212. Using this lambda, we reran the lasso regression on the entire data set. Here are the beta coefficients for the final lasso model:

Table 6: Final Lasso Model

Betas	Coef
(Intercept)	99.114
BMI	0.015
SUN	-0.139
SCR	-4.238
BLACK	0.441
AGE	-0.246
FEMALE	-5.106
cys	-16.285
DONOR	32.643
Tx	-5.361
Diabetes	3.515
AASK	8.476
gronigen	12.412
crisp	26.400
ccfp	8.939

Again, we calculate the estimated test r-squared for lasso by subtracting the optimism of the fitting process from the r-squared of the training model:

$$\hat{r}^2_{test} = r^2_{train} - \hat{optimism} = .746 - .006 = .740$$

d.) Compare the estimate of test r-squared between the bootstrap approach and the stepwise, ridge and lasso approaches.

To recap, here are the estimated test r-squareds for the bootstrap and stepwise, ridge, and lasso cross validations:

Table 7: Estimated Test r-squareds

Step.boot	Step.cv	Ridge	Lasso
0.7363	0.7427	0.7395	0.7397

We see that the estimated test r-squared values are similar for each approach. This means that all the models are predicting GFR equally well.

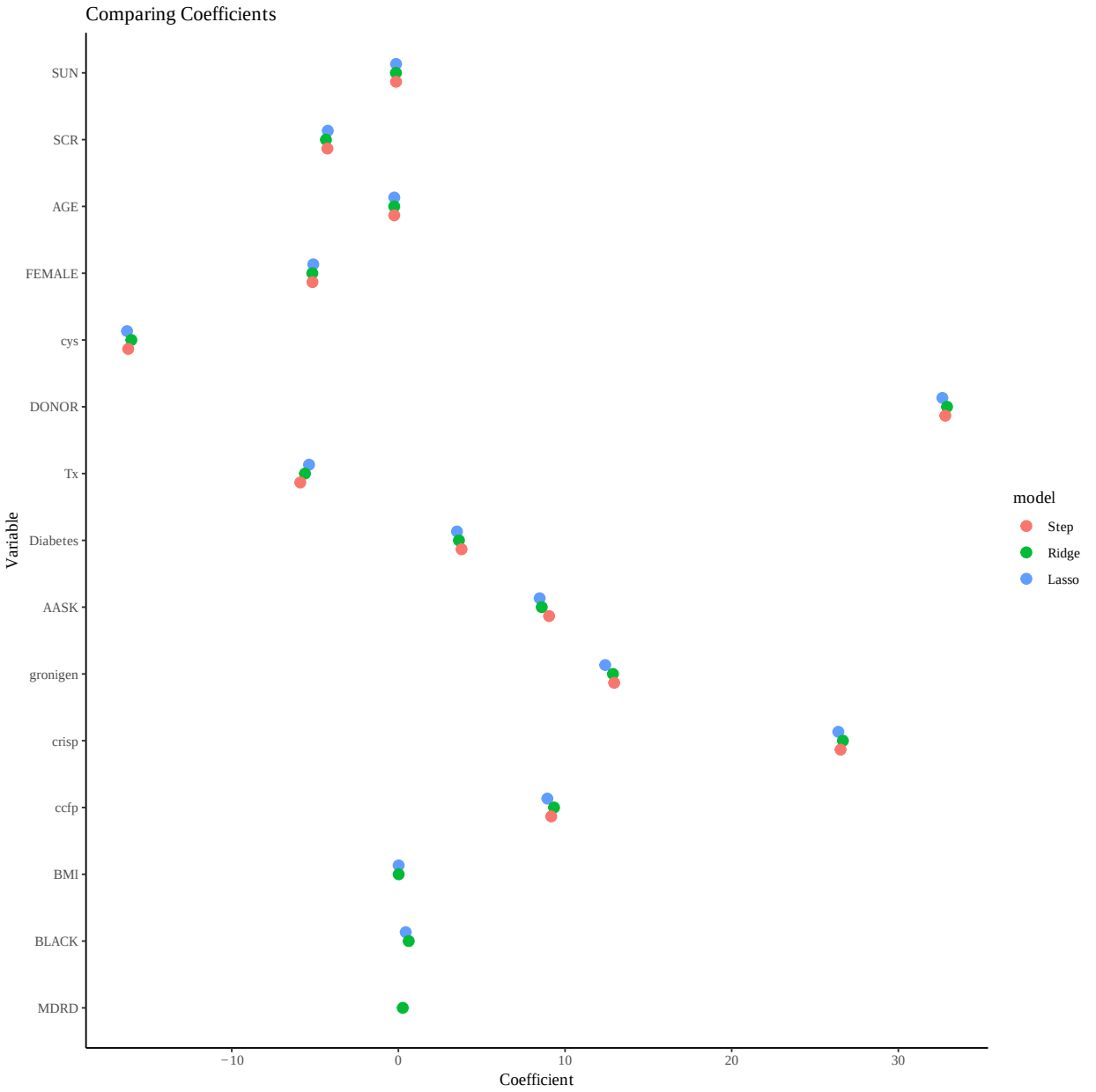
We also compare the average optimism for the best models:

Table 8: Average Optimism

Step.boot	Step.cv	Ridge	Lasso
0.0095	0.0031	0.0063	0.0061

We see that the average optimism is less than .01 for all of the models. We notice that the optimism estimated from the bootstrap approach is higher than the estimated optimism from the cross validation approach. Since the bootstrap method used more simulations, we may believe its value more than the cross validation method. However, the difference between the two is small: it is unclear if they are significantly different.

Lastly, we compared the coefficients for the step, ridge, and lasso final models (the final model for the bootstrap and cross validation step approach were the same):



We see from the plot above that the coefficients between the models agree well with eachother. We see that ridge uses all 15 variables (which is expected because the ridge approach does not do selection). Our lasso regression drops one variable, MDRD. Step selection drops MDRD, BLACK, and BMI. The coefficient with with the largest single effect in all models is DONOR. Since the performance is similar between all of the models, the step approach appears to give the most parsimonious model for this analysis.

Apendix: R code

```
### Load Libraries

library(knitr)
library(readr)
```

```r
library(MASS)
library(leaps)
library(dplyr)
library(stargazer)
library(caret)
library(extrafont)
library(glmnet)
library(dotwhisker)
library(broom)



# Data Cleaning



### Load data
iod = read_csv("iodatadev.csv")

### Filter out columns with more that 10% missing data
iod.clean = Filter(function(x) mean(is.na(x)) < 0.1, iod)

### Get complete cases
iod.clean = iod.clean %>%
  filter(complete.cases(.))



stargazer(as.data.frame(iod.clean), header = FALSE,
          table.placement = 'H',
          title = "Summary Statistics")




iod.clean = iod.clean %>%
  select(-X1, -ID, -"..2", -"..2_1", -CSG, -drds, - rass1)


for.star = alias(GFR~., iod.clean)

stargazer(as.data.frame(t(for.star$Complete)),
          header = FALSE,
          summary = FALSE,
          table.placement = 'H',
          title = "Collinearity Check")




### Filter out collinear terms
iod.clean = iod.clean %>%
  select(-WEIGHT, -HEIGHT, -c.cys, -mayodonor, -gronigendonor, - ccpdonor, - c.scr, - mayo)
```

```r
# Question 1: Bootstrap Step Regression



set.seed(1)

### Forward and backward step regression
full = lm(GFR ~ ., iod.clean)
base = lm(GFR ~ 1, iod.clean)
fulltest = stepAIC(full, scope = list(upper=full,lower=base), direction = "both", trace = FALSE)
summary.fulltest = summary(fulltest)
rsquared.fulltest = summary.fulltest$r.squared


stargazer(fulltest,
          header = FALSE,
          title = "StepAIC on Observed Data",
          table.placement = 'H')



### Bootstrap 1000 samples and get rsquares
set.seed(1)
nsims = 1000

r2boot.train = rep(NA, nsims)
r2boot.test = rep(NA, nsims)

for (i in 1:nsims) {

  train = sample_n(iod.clean, size = nrow(iod.clean), replace = TRUE)
  test = anti_join(iod.clean, train)
  mod = stepAIC(lm(GFR ~ ., data = train), direction = "both", trace = FALSE)
  summary.mod = summary(mod)
  r2boot.train[i] = summary.mod$r.squared
  preds = predict(mod, test)
  tss = sum((test$GFR - mean(test$GFR))^2)
  ess = sum((test$GFR - preds)^2)
  r2boot.test[i] = 1 - (ess/tss)


}


### Calculate average optimism
optimism = r2boot.train - r2boot.test
ave.optimism = mean(optimism)

### Calculate Estimated Test r2
est.test.r2 = rsquared.fulltest - ave.optimism
```

```r
# Question 2 and 3:



## a.) Step Regression Cross Validation:



### Function to get predictions for leap package
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
}

### Crossvalidate number of variables for step

set.seed(15)

folds = 10

test = createFolds(iod.clean$GFR, k = folds)

cv.errors = matrix(NA, nrow = folds, ncol = ncol(iod.clean) - 1 )
step.r2.train = matrix(NA, nrow = folds, ncol = ncol(iod.clean) - 1)
step.r2.test = matrix(NA, nrow = folds, ncol = ncol(iod.clean) - 1)

for (i in 1:folds) {

  training = iod.clean[-test[[i]], ]
  testing = iod.clean[test[[i]], ]
  best.model = regsubsets(GFR ~ .,
                          data = training,
                          nvmax = ncol(iod.clean),
                          method = "seqrep")

  for (j in 1:(ncol(iod.clean)-1)) {

    preds.training = predict.regsubsets(best.model, newdata = training, id = j)
    preds = predict.regsubsets(best.model, newdata = testing, id = j)


    #R2 for training folds
    train.tss = sum((training$GFR - mean(training$GFR))^2)
    train.ess = sum((training$GFR - preds.training)^2)
    r2train =  1 - (train.ess/train.tss)
    step.r2.train[i,j] = r2train
```

```
    #R2 for testing folds
    test.tss = sum((testing$GFR - mean(testing$GFR))^2)
    test.ess = sum((testing$GFR - preds)^2)
    r2test =  1 - (test.ess/test.tss)
    step.r2.test[i,j] = r2test



    MSE = mean((testing$GFR - preds)^2)
    cv.errors[i, j] = MSE



  }


}


mean.cv.errors = apply(cv.errors, 2, mean)

step.plot.frame = data.frame(variables = seq(1:(ncol(iod.clean)-1)), error = mean.cv.errors)

step.plot = step.plot.frame %>%
  ggplot(aes(x = variables, y = mean.cv.errors)) +
  geom_point() +
  geom_line() +
  xlab("Number of Variables") +
  ylab("Average MSE") +
  ggtitle("Step Regression Tuning") +
  theme_classic() +
  theme(text=element_text(size=12,  family="CM Sans"))


### Use best model on full data

best.number = which.min(mean.cv.errors) ## 12 variables is best

### Run models
final.step = regsubsets(GFR ~ .,
                        data = iod.clean,
                        nvmax = ncol(iod.clean),
                        method = "seqrep")

### Extract 12 variable model's rsquared
final.step.r2 = summary(final.step)$rsq[which.min(mean.cv.errors)]

### Adjust the r2 by the optimism
step.optimisms = step.r2.train - step.r2.test
mean.step.optimisms = apply(step.optimisms, 2, mean)

### Adjusted r2
step.adj.r2 = final.step.r2 - mean.step.optimisms[which.min(mean.cv.errors)]
```

```
step.plot



step.stargazer = lm(GFR ~ SUN + SCR + AGE + FEMALE + cys + DONOR + Tx + Diabetes +AASK + gronigen + cris

stargazer(step.stargazer,
          header = FALSE,
          title = "Final Step Model",
          table.placement = 'H')



## b.) Ridge Regression Cross Validation:



set.seed(11)

folds = 10

### Create folds indexes for the observations left out
test = createFolds(iod.clean$GFR, k = folds)

### Grid of lambdas to check
grid=seq(1, .001, length=100)


### Set up data

x = model.matrix(GFR ~ . - 1, data = iod.clean)
y = iod.clean$GFR


### Initialize matrices for mse, r2train, r2test
ridge.cv.errors = matrix(NA, nrow = folds, ncol = length(grid))
ridge.r2.train = matrix(NA, nrow = folds, ncol = length(grid))
ridge.r2.test = matrix(NA, nrow = folds, ncol = length(grid))


### For each fold, make test and training x and y
### For each lambda, calculate r2 and mse
for (i in 1:folds) {

  trainingx = x[-test[[i]], ]
  trainingy = y[-test[[i]]]
  testingx = x[test[[i]], ]
  testingy = y[test[[i]]]

  for (j in 1:length(grid)){

    best.model = glmnet(x = trainingx, y = trainingy, alpha = 0, lambda = grid)
```

```
    trainpreds = predict(best.model, newx = trainingx, s = grid[j])
    preds = predict(best.model, newx = testingx, s = grid[j])


    #R2 for training folds
    train.tss = sum((trainingy - mean(trainingy))^2)
    train.ess = sum((trainingy - trainpreds)^2)
    r2train =  1 - (train.ess/train.tss)
    ridge.r2.train[i,j] = r2train


    #R2 for testing folds
    test.tss = sum((testingy - mean(testingy))^2)
    test.ess = sum((testingy - preds)^2)
    r2test =  1 - (test.ess/test.tss)
    ridge.r2.test[i,j] = r2test

    #MSE
    MSE = mean((preds - testingy)^2)
    ridge.cv.errors[i, j] = MSE

  }

}


### Get average Optimism for each lambda
ridge.optimism = ridge.r2.train - ridge.r2.test
est.ridge.optimism = apply(ridge.optimism, 2, mean)

### Get average MSE for each lambda
mean.ridge.errors = apply(ridge.cv.errors, 2, mean)

ridge.plot.frame = data.frame(lambda = grid, error = mean.ridge.errors)

ridge.plot = ridge.plot.frame %>% ggplot(aes(x = lambda, y = error)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept = .1019, color = "red") +
  geom_text(aes(x=.1019, label="\nMin = .1019", y=198.6), colour="red", angle=90) +
  ylab("Average MSE") +
  xlab("Lambda") +
  ggtitle("Tuning Lambda for Ridge Regression") +
  theme_classic() +
  theme(text=element_text(size=11,  family="CM Sans"))

### Get min lambda and its optimism
best.ridge.opt = est.ridge.optimism[which.min(ridge.plot.frame$error)]


### Fit model with best lambda
final.ridge = glmnet(x = x, y = y, alpha = 0, lambda = grid[which.min(ridge.plot.frame$error)])
```

```
### Get final model preds
final.ridge.preds = predict(final.ridge, newx = x)


### Get final model r2
final.ridge.tss = sum((y - mean(y))^2)
final.ridge.ess = sum((y - final.ridge.preds)^2)
final.ridge.r2 =  1 - (final.ridge.ess/final.ridge.tss)


### Get adj final model r2
final.ridge.adj.r2 = final.ridge.r2 - best.ridge.opt

ridge.plot



ridge.coefs = coef(final.ridge)
ridge.stargazer = summary(coef(final.ridge))

ridge.stargazer2 = data.frame(Betas      = rownames(ridge.coefs)[ridge.stargazer$i],
           Destination = colnames(ridge.coefs)[ridge.stargazer$j],
           Coef      = ridge.stargazer$x)

ridge.stargazer2 = ridge.stargazer2 %>%
  select(-Destination)

stargazer(ridge.stargazer2, header = FALSE,
          title = "Final Ridge Model", summary = FALSE, table.placement = 'H', rownames = FALSE)



## c.) Lasso Regression Cross Validation:




set.seed(11)

folds = 10

### Create folds makes indexes for left out observations
test = createFolds(iod.clean$GFR, k = folds)

### Grid of lambdas to check
grid=seq(1, .001, length=100)


### Set up data

x = model.matrix(GFR ~ . - 1, data = iod.clean)
y = iod.clean$GFR
```

```
### Initialize matrices for mse, r2train, r2test
lasso.cv.errors = matrix(NA, nrow = folds, ncol = length(grid))
lasso.r2.train = matrix(NA, nrow = folds, ncol = length(grid))
lasso.r2.test = matrix(NA, nrow = folds, ncol = length(grid))


### For each fold, make test and training x and y
### For each lambda, calculate r2 and mse
for (i in 1:folds) {

  trainingx = x[-test[[i]], ]
  trainingy = y[-test[[i]]]
  testingx = x[test[[i]], ]
  testingy = y[test[[i]]]

  for (j in 1:length(grid)){

    best.model = glmnet(x = trainingx, y = trainingy, alpha = 1, lambda = grid)

    trainpreds = predict(best.model, newx = trainingx, s = grid[j])
    preds = predict(best.model, newx = testingx, s = grid[j])


    #R2 for training folds
    train.tss = sum((trainingy - mean(trainingy))^2)
    train.ess = sum((trainingy - trainpreds)^2)
    r2train =  1 - (train.ess/train.tss)
    lasso.r2.train[i,j] = r2train


    #R2 for testing folds
    test.tss = sum((testingy - mean(testingy))^2)
    test.ess = sum((testingy - preds)^2)
    r2test =  1 - (test.ess/test.tss)
    lasso.r2.test[i,j] = r2test

    #MSE
    MSE = mean((preds - testingy)^2)
    lasso.cv.errors[i, j] = MSE

  }

}


### Get average Optimism for each lambda
lasso.optimism = lasso.r2.train - lasso.r2.test
est.lasso.optimism = apply(lasso.optimism, 2, mean)

### Get average MSE for each lambda
mean.lasso.errors = apply(lasso.cv.errors, 2, mean)

lasso.plot.frame = data.frame(lambda = grid, error = mean.lasso.errors)
```

```
laso.plot = lasso.plot.frame %>% ggplot(aes(x = lambda, y = error)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept = .0212, color = "red") +
  geom_text(aes(x=.0212, label="\nMin = .0212", y=200.6), colour="red", angle=90) +
  ylab("Average MSE") +
  xlab("Lambda") +
  ggtitle("Tuning Lambda for Lasso Regression") +
  theme_classic() +
  theme(text=element_text(size=11,  family="CM Sans"))

### Get min lambda and its optimism
best.lasso.opt = est.lasso.optimism[which.min(lasso.plot.frame$error)]

### Fit model with best lambda
final.lasso = glmnet(x = x, y = y, alpha = 1, lambda = grid[which.min(lasso.plot.frame$error)])

### Get final model preds
final.lasso.preds = predict(final.lasso, newx = x)


### Get final model r2
final.lasso.tss = sum((y - mean(y))^2)
final.lasso.ess = sum((y - final.lasso.preds)^2)
final.lasso.r2 =  1 - (final.lasso.ess/final.ridge.tss)


### Get adj final model r2
final.lasso.adj.r2 = final.lasso.r2 - best.lasso.opt

laso.plot



lasso.coefs = coef(final.lasso)
lasso.stargazer = summary(coef(final.lasso))

lasso.stargazer2 = data.frame(Betas      = rownames(lasso.coefs)[lasso.stargazer$i],
           Destination = colnames(lasso.coefs)[lasso.stargazer$j],
           Coef      = lasso.stargazer$x)

lasso.stargazer2 = lasso.stargazer2 %>%
  select(-Destination)

stargazer(lasso.stargazer2, header = FALSE,
          title = "Final Lasso Model", summary = FALSE, table.placement = 'H', rownames = FALSE)




## d.) Compare the estimate of test r-squared between the bootstrap approach and the stepwise, ridge an
```

```
forkable = round(data.frame( Step.boot = est.test.r2, Step.cv = step.adj.r2, Ridge = final.ridge.adj.r2
kable(forkable, caption = "Estimated Test r-squareds")



forkable = round(data.frame( Step.boot = ave.optimism, Step.cv = mean.step.optimisms[which.min(mean.cv.
kable(forkable, caption = "Average Optimism")

### Set up coefficient plot
step.df = tidy(step.stargazer) %>%
  select(term, estimate, std.error) %>%
  mutate(std.error = 0) %>%
  mutate(model = "Step")

ridge.df = tidy(final.ridge) %>%
  select(term, estimate) %>%
  mutate(std.error = 0) %>%
  mutate(model = "Ridge")

lasso.df = tidy(final.lasso) %>%
  select(term, estimate) %>%
  mutate(std.error = 0) %>%
  mutate(model = "Lasso")

fordwplot = rbind(step.df, ridge.df, lasso.df)


### Coef plot
dwplot(fordwplot, dot_args = list(size = 3)) +
  ylab("Variable") +
  xlab("Coefficient") +
  ggtitle("Comparing Coefficients") +
  theme_classic() +
  theme(text=element_text(size=11,  family="CM Sans"))
```