MLStats: Linear Model Selection and Regularization

Darshan Patel

2/12/2019

In this assignment, mimic the lab exercises from ISLR Chapter 6: Linear Model Selection and Regularization

Libraries

Load the following libraries.

```
rm(list = ls())
library(tidyverse)
```

```
## Warning: package 'tibble' was built under R version 3.4.4
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
library(scales)
```

```
## Warning: package 'scales' was built under R version 3.4.4
```

```
library(gridExtra)
library(RColorBrewer)
library(leaps)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.4
```

```
## Warning: package 'Matrix' was built under R version 3.4.4
```

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 3.4.4
```

Dataset

In this assignment, the dataset that will be used is `student-mat.csv`. It is a collection of attributes on student achievement in secondary math education of two Portuguese schools.

(Source: https://archive.ics.uci.edu/ml/datasets/Student+Performance)

This data approach student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school related features) and it was collected by using school reports and questionnaires. Two datasets are provided regarding the performance in two distinct subjects: Mathematics (mat) and Portuguese language (por). In [Cortez and Silva, 2008], the two datasets were modeled under binary/five-level classification and regression tasks. Important note: the target attribute G3 has a strong correlation with attributes G2 and G1. This occurs because G3 is the final year grade (issued at the 3rd period), while G1 and G2 correspond to the 1st and 2nd period grades. It is more difficult to predict G3 without G2 and G1, but such prediction is much more useful (see paper source for more details).

```
df = read_delim("student-mat.csv", delim = ";")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   age = col_integer(),
##   Medu = col_integer(),
##   Fedu = col_integer(),
##   traveltime = col_integer(),
##   studytime = col_integer(),
##   failures = col_integer(),
##   famrel = col_integer(),
##   freetime = col_integer(),
##   goout = col_integer(),
##   Dalc = col_integer(),
##   Walc = col_integer(),
##   health = col_integer(),
##   absences = col_integer(),
##   G1 = col_integer(),
##   G2 = col_integer(),
##   G3 = col_integer()
## )
```

```
## See spec(...) for full column specifications.
```

The size of the dataset is

```
nrow(df)
```

```
## [1] 395
```

The number of variables in the dataset is

```
ncol(df)
```

```
## [1] 33
```

The variables are listed below:

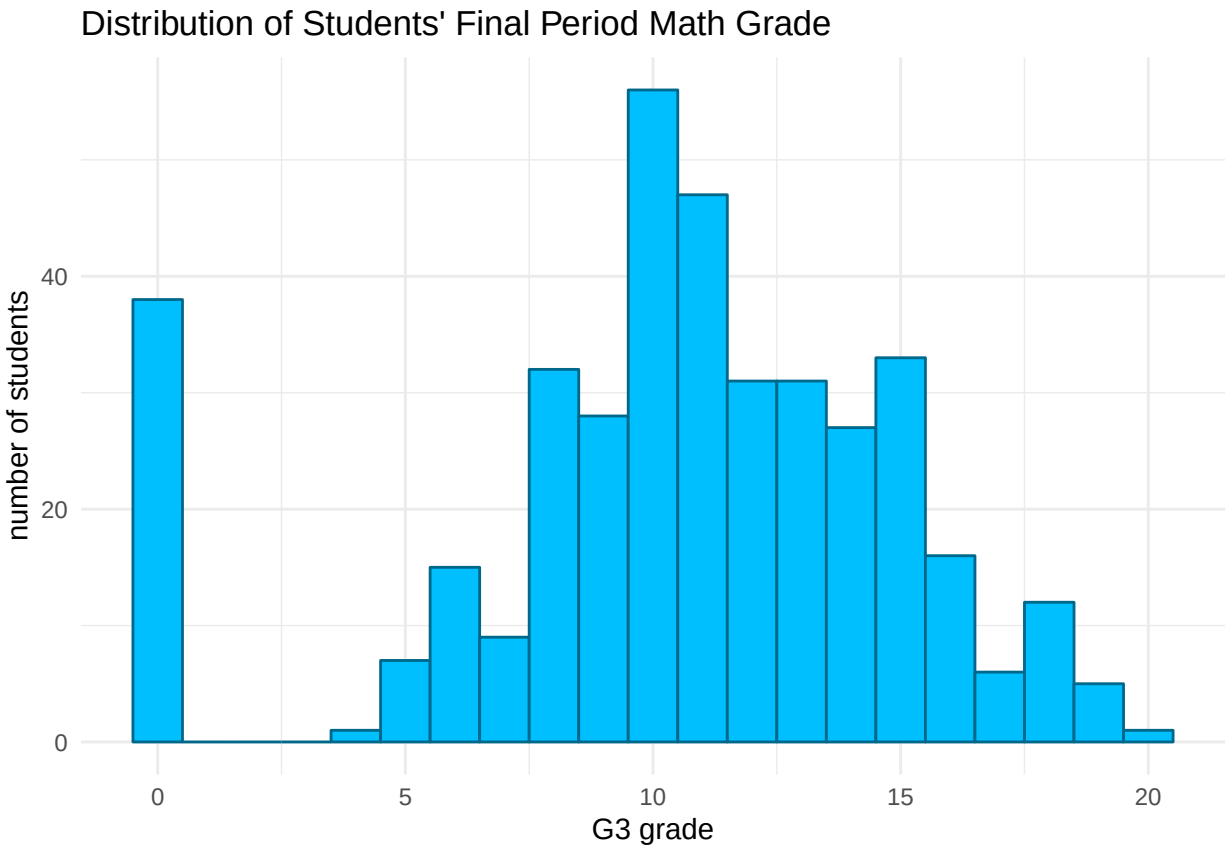
```
colnames(df)
```

```
##  [1] "school"     "sex"        "age"        "address"    "famsize"
##  [6] "Pstatus"    "Medu"       "Fedu"       "Mjob"       "Fjob"
## [11] "reason"     "guardian"   "traveltime" "studytime"  "failures"
## [16] "schoolsup"  "famsup"     "paid"       "activities" "nursery"
## [21] "higher"     "internet"   "romantic"   "famrel"     "freetime"
## [26] "goout"      "Dalc"       "Walc"       "health"     "absences"
## [31] "G1"         "G2"         "G3"
```

According to the data description, `G1` and `G2` are first and second period grades while `G3` is the third and final period grade. In this assignment, `G3` will be predicted using all variables including `G1` and `G2`.

The distribution of the students' final period math grade is shown below.

```
ggplot(df, aes(x = G3)) + geom_histogram(binwidth = 1,
                                         fill = "deepskyblue",
                                         color = "deepskyblue4") +
  ggtitle("Distribution of Students' Final Period Math Grade") +
  labs(x = "G3 grade", y = "number of students") +
  theme_minimal()
```



The distribution of students' final math grades is skewed right. Furthermore, there is an arbitrary large number of 0s received by students.

With so many variables, it is impossible to look at all variables to see visually which ones more or less correlate with `G3`. Therefore use different techniques to find the best combinations of variables to predict `G3`.

Just to check, are there null/missing values?

```
any(is.na(df))
```

```
## [1] FALSE
```

No nulls.

Best Subset Selection

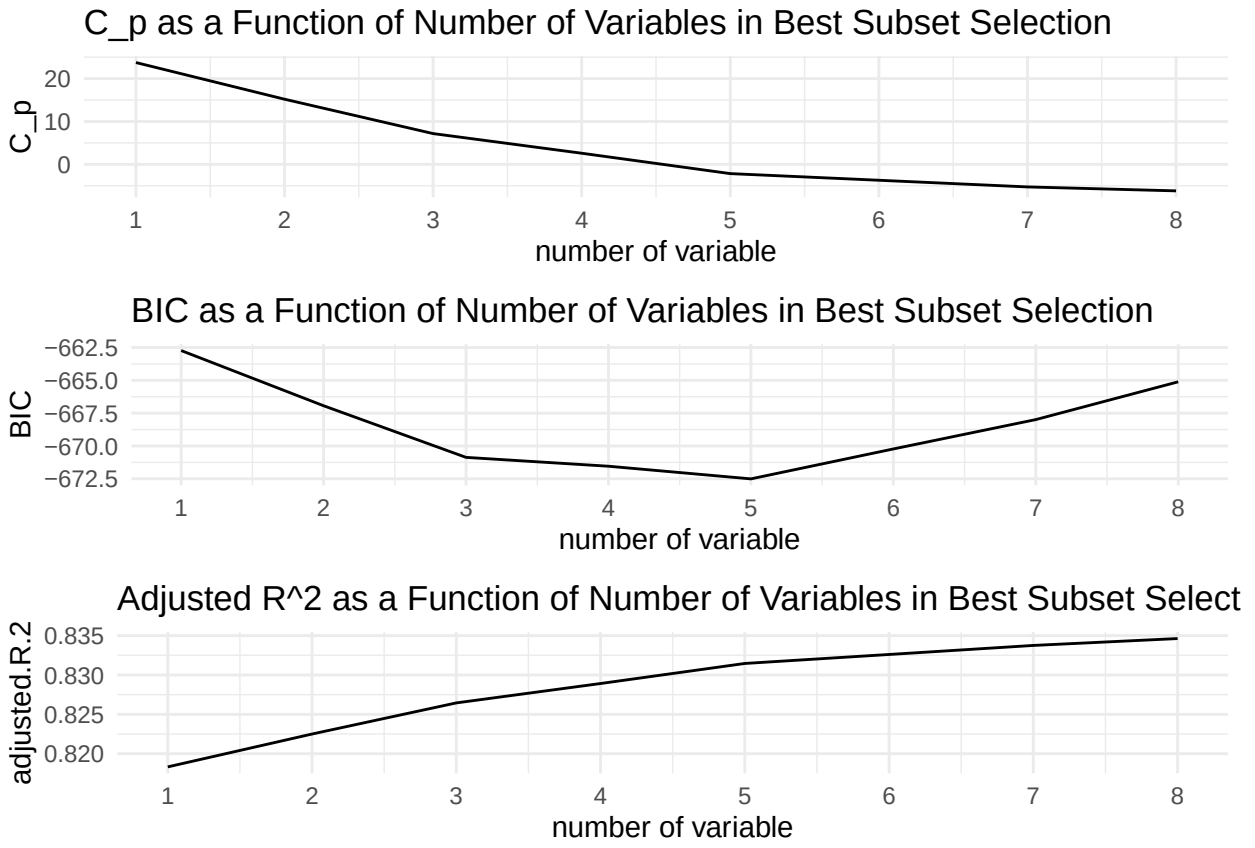
Fit a model using best subset selection. Since best subset selection creates 2^p models, and $p = 32$ here, that is a large number of models to create. Therefore, set a max of 8 variables to use.

```
model_bss = regsubsets(G3~., df, nvmax = 8, really.big = TRUE)
```

Plot the model statistics as a function of number of variables added to the model.

```
model_bss_stats = data.frame("num_vars" = seq(1,8),
                             "C_p" = summary(model_bss)$cp,
                            "BIC" = summary(model_bss)$bic,
                            "adjusted R^2" = summary(model_bss)$adjr2)

g1 = ggplot(model_bss_stats, aes(x = num_vars, y = C_p)) + geom_path() +
```

```
  scale_x_continuous(breaks = seq(1, 8, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("C_p as a Function of Number of Variables in Best Subset Selection") +
  theme_minimal()
g2 = ggplot(model_bss_stats, aes(x = num_vars, y = BIC)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 8, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("BIC as a Function of Number of Variables in Best Subset Selection") +
  theme_minimal()
g3 = ggplot(model_bss_stats, aes(x = num_vars, y = adjusted.R.2)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 8, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("Adjusted R^2 as a Function of Number of Variables in Best Subset Selection") +
  theme_minimal()

grid.arrange(g1,g2,g3,ncol=1)
```



Best subset selection does not do well for modeling in this scenario. The R^2 value is decent too. The best number of variables to use is

```
min(which.min(summary(model_bss)$cp),
    which.min(summary(model_bss)$bic))
```

```
## [1] 5
```

Only 5 variables are needed. The coefficients of the model are

```
coef(model_bss, 5)
```

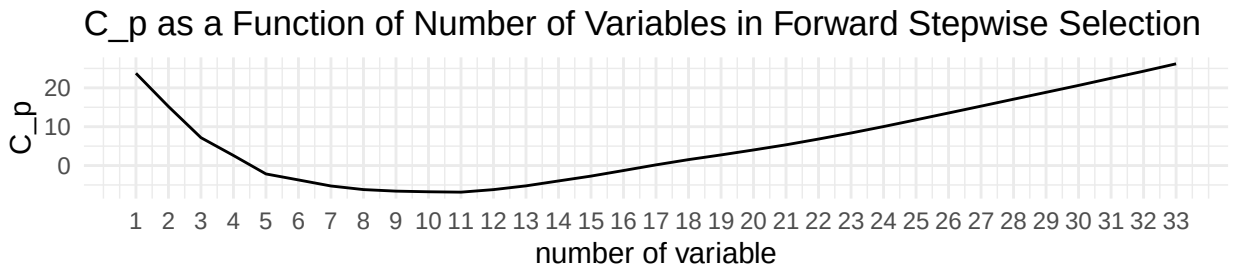
```
## (Intercept)         age      famrel    absences          G1          G2
## -0.07765375 -0.20167083  0.35724740  0.04365321  0.15794465  0.97804334
```

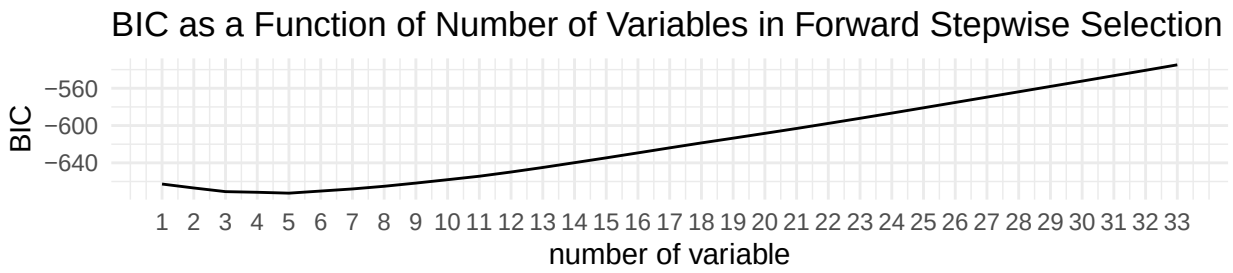
Now try using forward and backward stepwise selection.

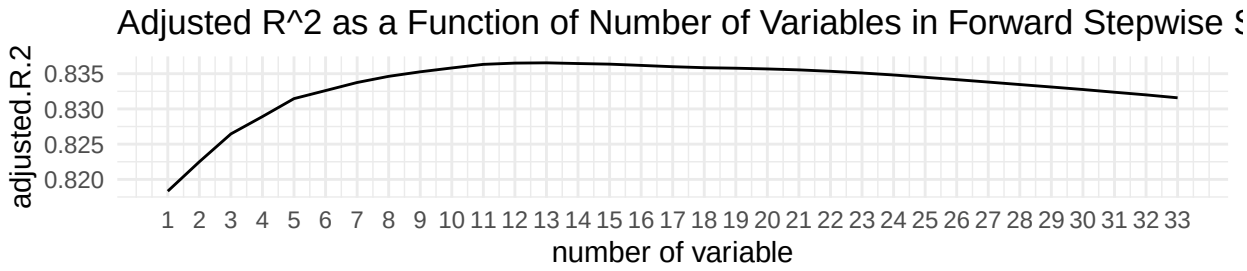
Foward and Backward Stepwise Selection

Perform forward stepwise selection first and report the model statistics. This method is not computationally expensive therefore all variables sizes will be considered.

```
model_fss = regsubsets(G3~., df, nv = 33, method = "forward")

model_fss_stats = data.frame("num_vars" = seq(1,33),
                             "C_p" = summary(model_fss)$cp,
                            "BIC" = summary(model_fss)$bic,
                            "adjusted R^2" = summary(model_fss)$adjr2)

g4 = ggplot(model_fss_stats, aes(x = num_vars, y = C_p)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 33, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("C_p as a Function of Number of Variables in Forward Stepwise Selection") +
  theme_minimal()
g5 = ggplot(model_fss_stats, aes(x = num_vars, y = BIC)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 33, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("BIC as a Function of Number of Variables in Forward Stepwise Selection") +
  theme_minimal()
g6 = ggplot(model_fss_stats, aes(x = num_vars, y = adjusted.R.2)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 33, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("Adjusted R^2 as a Function of Number of Variables in Forward Stepwise Selection") +
  theme_minimal()

grid.arrange(g4,g5,g6,ncol=1)
```







Nice low adjusted R^2 value. The best number of variables to use is

```
min(which.min(summary(model_fss)$cp),
    which.min(summary(model_fss)$bic))
```

```
## [1] 5
```

The best model uses 5 variables as oppose to 33 variables. This is the same result as best subset selection. But do the coefficients and specific variables match? The coefficients of the model are

```
coef(model_fss, 5)
```

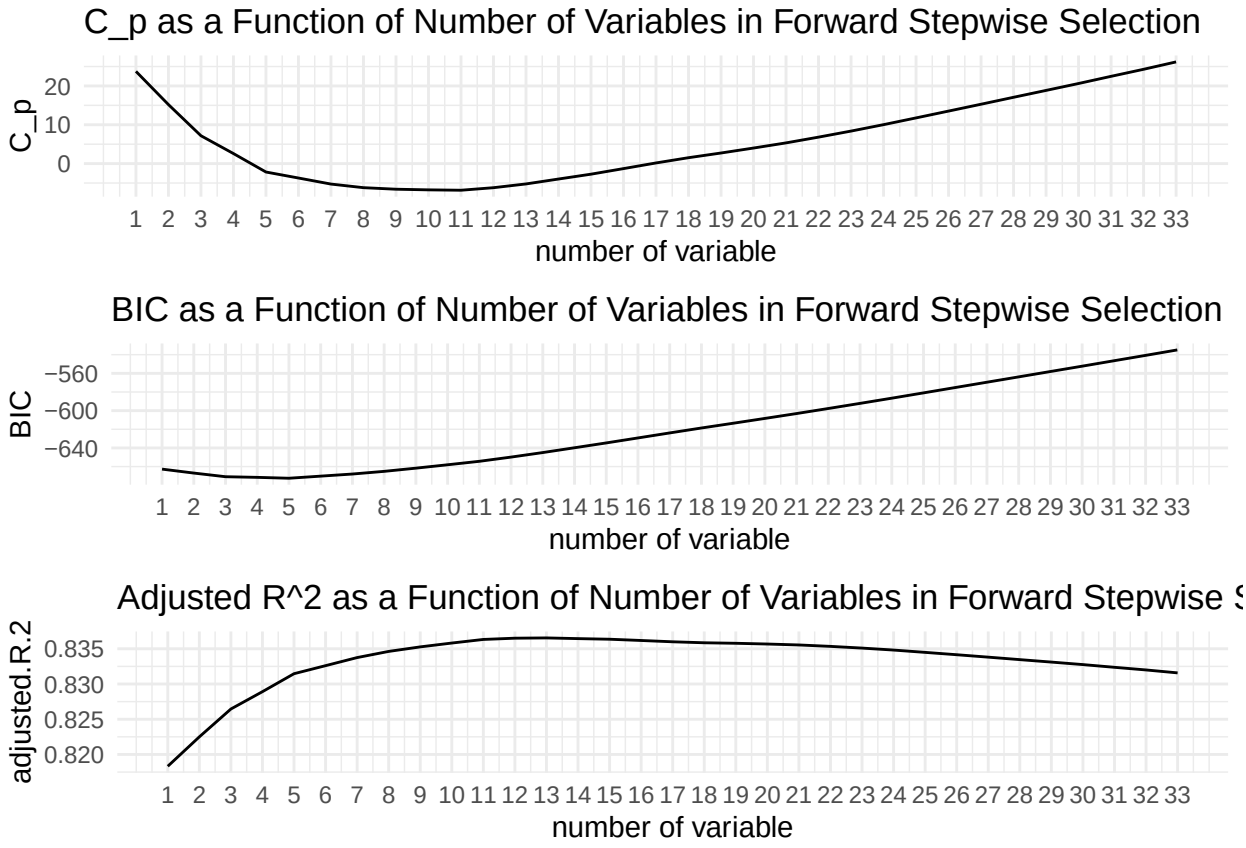
```
## (Intercept)         age      famrel    absences          G1          G2
## -0.07765375 -0.20167083  0.35724740  0.04365321  0.15794465  0.97804334
```

Best subset selection and forward stepwise selection created the same model.

Now try backward stepwise selection and report the model statistics.

```
model_bwss = regsubsets(G3~., df, nv = 33, method = "backward")

model_bwss_stats = data.frame("num_vars" = seq(1,33),
                              "C_p" = summary(model_bwss)$cp,
                             "BIC" = summary(model_bwss)$bic,
                             "adjusted R^2" = summary(model_bwss)$adjr2)

g7 = ggplot(model_bwss_stats, aes(x = num_vars, y = C_p)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 33, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("C_p as a Function of Number of Variables in Backward Stepwise Selection") +
```

```
  theme_minimal()
g8 = ggplot(model_bwss_stats, aes(x = num_vars, y = BIC)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 33, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("BIC as a Function of Number of Variables in Backward Stepwise Selection") +
  theme_minimal()
g9 = ggplot(model_bwss_stats, aes(x = num_vars, y = adjusted.R.2)) + geom_path() +
  scale_x_continuous(breaks = seq(1, 33, by = 1)) +
  labs(x = "number of variable") +
  ggtitle("Adjusted R^2 as a Function of Number of Variables in Backward Stepwise Selection") +
  theme_minimal()

grid.arrange(g4,g5,g6,ncol=1)
```



The best number of variables to use is

```
min(which.min(summary(model_bwss)$cp),
    which.min(summary(model_bwss)$bic))
```

```
## [1] 5
```

The best model uses 5 variables. The coefficients of the model are

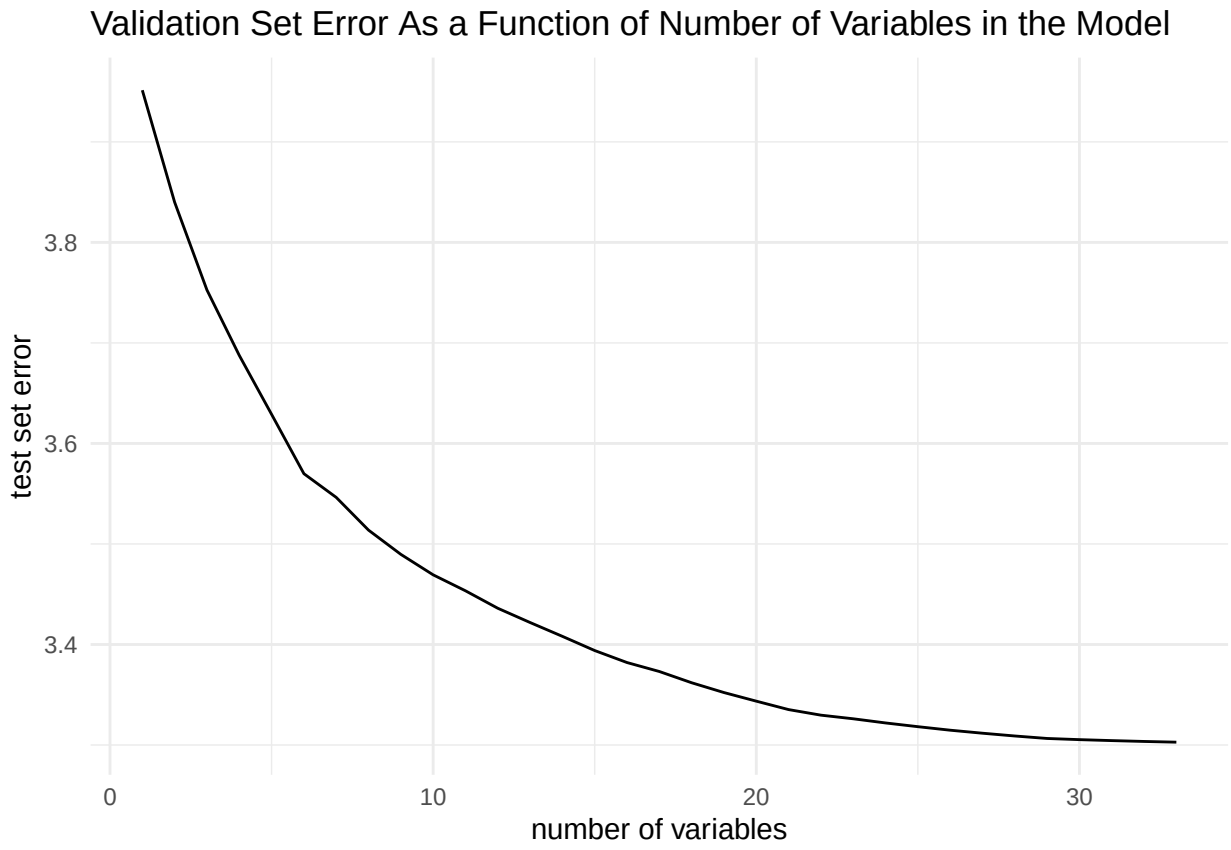
```
coef(model_fss, 5)
```

```
## (Intercept)         age      famrel    absences          G1          G2
## -0.07765375 -0.20167083  0.35724740  0.04365321  0.15794465  0.97804334
```

The same coefficients and coefficient estimates are found again.

All of the above models appear to be doing ok looking at the adjusted R^2 statistic. Try to improve on it using the validation set approach and cross validation and best subset selection.

```
set.seed(25)
indices = sample(1:nrow(df), size = 0.7*nrow(df))
train = df[indices,]
test = df[indices,]
model_bss_vs = regsubsets(G3~., train, nvmax = 33)
test_matrix = model.matrix(G3~., test)
errors_bss = c()

for(i in 1:33){
  c = coef(model_bss_vs, id = i)
  pred = test_matrix[, names(c)] %*% c
  errors_bss = c(errors_bss, mean((test$G3 - pred)^2))
}

error_vs_df = data.frame("num_vars" = seq(1,33, by=1),
                         "error_bss" = errors_bss)

ggplot(error_vs_df, aes(x = num_vars, y = error_bss)) + geom_path() +
  ggtitle("Validation Set Error As a Function of Number of Variables in the Model") +
  labs(x = "number of variables", y = "test set error") +
  theme_minimal()
```



As opposed to running best subset selection on the entire dataset, running it and testing on a smaller data set resulted in the best number of variables to be 33. When the entire dataset is used, it is 5. However creating the model using a split will result in better model performance for the future even if all variables seem to be

needed. Note that the test set error do not change as much after 25 variables. Use this value.

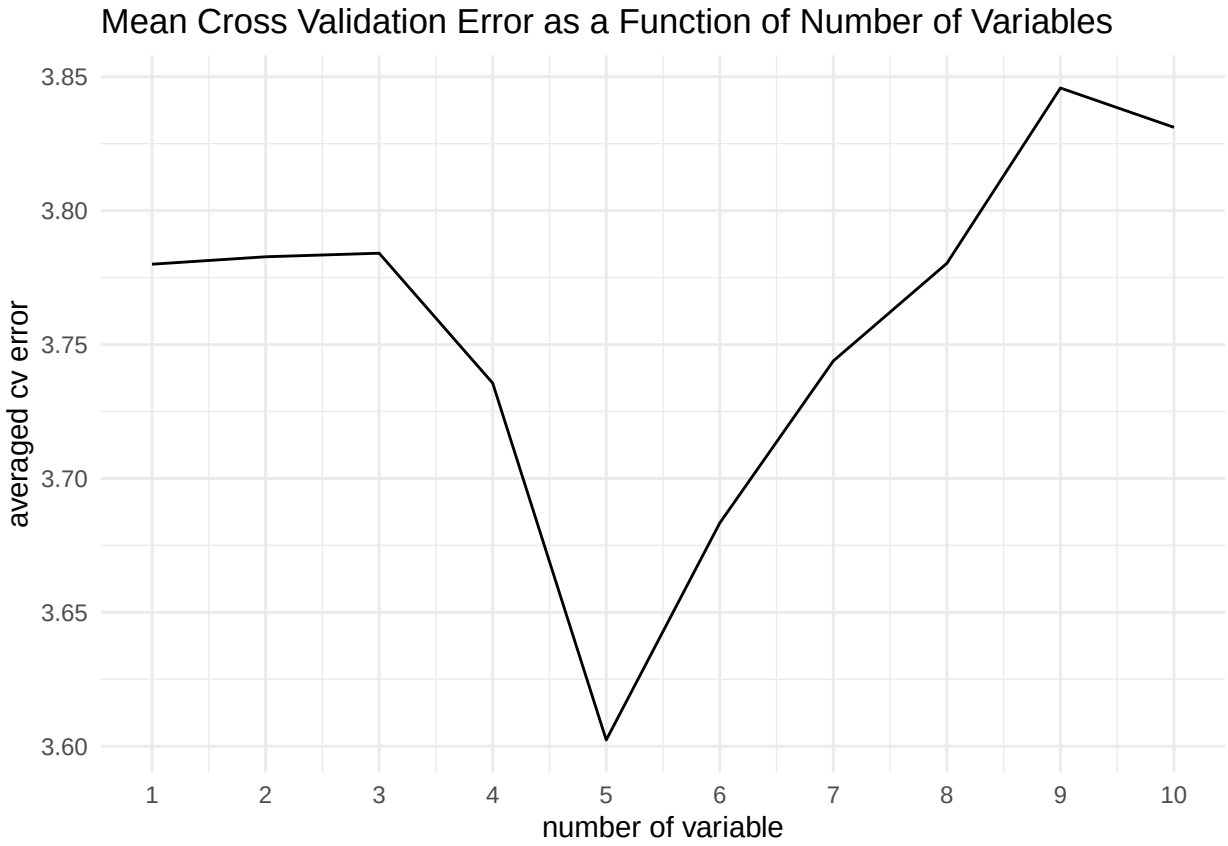
The coefficients of the 25 variable model are

```
coef(model_bss_vs, 25)
```

```
##   (Intercept)      schoolMS           age    famsizeLE3      PstatusT
##   -0.80089180    0.40363239   -0.16375694    0.25855009   -0.30221189
##          Medu    Mjobhealth     Fjobother  Fjobservices   Fjobteacher
##    0.25677711   -0.26915937   -0.43578178   -0.91345563   -0.89953838
##   reasonother guardianother    traveltime     studytime  schoolsupyes
##    0.59525243    0.36290578    0.20577050   -0.24824395    0.36168315
##     famsupyes       paidyes activitiesyes    nurseryyes   romanticyes
##    0.19594834   -0.13720961   -0.23397989   -0.18291394   -0.28475137
##        famrel          Dalc          Walc      absences            G1
##    0.33340923   -0.20572229    0.23052154    0.03374177    0.21199998
##            G2
##    0.95966448
```

Can it be improved using cross-validation? (Note: A smaller `nvmax` parameters was passed since the algorithm was time-consuming.)

```
k = 5
set.seed(25)
folds = sample(1:k, nrow(df), replace = TRUE)
cv_errors = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
predict.regsubsets = function(object, newdata, id, ...){
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id = id)
  xvars = names(coefi)
  return(mat[,xvars] %*% coefi)
}
for(i in 1:k){
  train = df[folds !=i ,]
  test = df[folds == i ,]
  temp_model = regsubsets(G3~., data = train, nvmax = 10)
  for(j in 1:10){
    pred = predict(temp_model, test, id = j)
    cv_errors[i,j] = mean((test$G3 - pred)^2)
  }
}
mean_cv_errors = apply(cv_errors, 2, mean)
mean_cv_df = data.frame("num_vars" = seq(1,10,by=1),
                        "cv_error" = mean_cv_errors)

ggplot(mean_cv_df, aes(x = num_vars, y = cv_error)) + geom_path() +
  ggtitle("Mean Cross Validation Error as a Function of Number of Variables") +
  labs(x = "number of variable", y = "averaged cv error") +
  scale_x_continuous(breaks = 1:10, labels = 1:10) +
  theme_minimal()
```



By using cross validation, the "best" model appears to be one that has 5 variables.

Using this information, perform best subset selection on the full data set in order to find the 5-variable model.

```
model_best_cv = regsubsets(G3~., df, nvmax = 5)
coef(model_best_cv, 5)
```

```
## (Intercept)         age      famrel    absences          G1          G2
## -0.07765375 -0.20167083  0.35724740  0.04365321  0.15794465  0.97804334
```

These coefficients have appeared before.

All of the selection methods have performed similarly on the same train/test split, giving the same give coefficients and coefficient estimates. Using forward/backward stepwise selection was preferred over best subset selection due to its lower computational cost.

Let's see if new estimates can be found via another method.

Ridge Regression

Fit a ridge regression model on a train/test split to predict `G3`.

```
set.seed(25)
x = model.matrix(G3~., df)[,-1]
y = df$G3
train = sample(1:nrow(x), size = 0.7*nrow(x))
test = setdiff(x, train)

model_rr = glmnet(x[train,], y[train],
```

```
                  alpha = 0,
                  lambda = 10^seq(10, -5, length = 1000),
                  thresh = 1e-12)
```

Predict the G3 scores for when $\lambda = 3$ in the ridge regression model and calculate its test set MSE.

```
pred = predict(model_rr, s = 3, newx = x[test,])
mean((pred - y[test])^2)
```

```
## [1] 1.784625
```

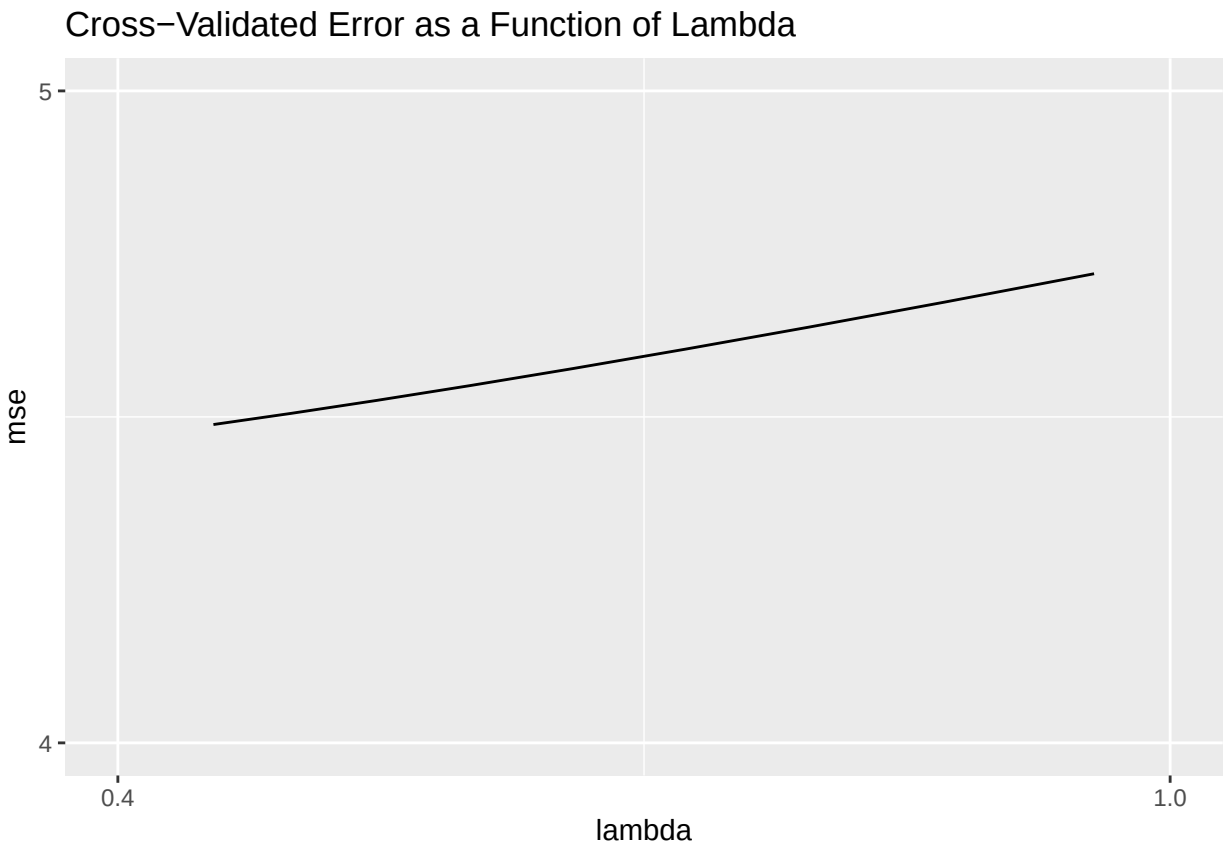
Instead of arbitrarily picking a λ value, use cross-validation to pick the optimal value.

```
set.seed(25)
cv_model_rr = cv.glmnet(x[train,], y[train], alpha = 0)
```

The following plot shows how cross-validation error changes as λ increases.

```
cv_rr_df = data.frame("lambda" = cv_model_rr$lambda, "mse" = cv_model_rr$cvm)
ggplot(cv_rr_df, aes(x = lambda, y = mse)) + geom_path() +
  ggtitle("Cross-Validated Error as a Function of Lambda") +
  scale_x_continuous(limits = c(0.4, 1), breaks = c(0.4,1)) +
  scale_y_continuous(limits = c(4,5), breaks = c(4,5))
```

```
## Warning: Removed 90 rows containing missing values (geom_path).
```



The optimal λ value is

```
cv_model_rr$lambda.min
```

```
## [1] 0.4544779
```

Using this value, predict on the test set.

```
pred = predict(model_rr, s = cv_model_rr$lambda.min, newx = x[test,])
mean((pred - y[test])^2)
```

```
## [1] 1.756583
```

This is a lower test set MSE than the one received by randomly choosing a λ value.

Now refit the ridge regression model on the full data set and predict using the value of λ above to attain the coefficient estimates.

```
model_rr_best = glmnet(x, y, alpha = 0)
predict(model_rr_best, type = "coefficients", s = cv_model_rr$lambda.min)
```

```
## 42 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)      -0.35841911
## schoolMS          0.40092821
## sexM              0.20758479
## age              -0.17119792
## addressU          0.14238575
## famsizeLE3        0.08954966
## PstatusT         -0.20640913
## Medu              0.13157960
## Fedu             -0.11065787
## Mjobhealth       -0.02279771
## Mjobother         0.12218872
## Mjobservices      0.09849061
## Mjobteacher      -0.06011624
## Fjobhealth        0.41666673
## Fjobother         0.07489788
## Fjobservices     -0.11916946
## Fjobteacher      -0.03232873
## reasonhome       -0.16029583
## reasonother       0.41154313
## reasonreputation  0.14028598
## guardianmother    0.14635610
## guardianother    -0.07277865
## traveltime        0.02944006
## studytime        -0.08004083
## failures         -0.22490302
## schoolsupyes      0.43772284
## famsupyes         0.11677830
## paidyes           0.14672928
## activitiesyes    -0.30025168
## nurseryyes       -0.19692866
## higheryes         0.25682664
## internetyes      -0.03867757
## romanticyes      -0.37451723
## famrel            0.28847693
## freetime          0.04910884
## goout            -0.02891537
## Dalc             -0.12674936
## Walc              0.14877169
## health            0.03920464
```

```
## absences           0.04052797
## G1                 0.34096165
## G2                 0.74827545
```

Note that some coefficient estimates are close to 0 but none are actually 0. This shows that ridge regression does not perform variable selection.

Now lasso.

Lasso Regression

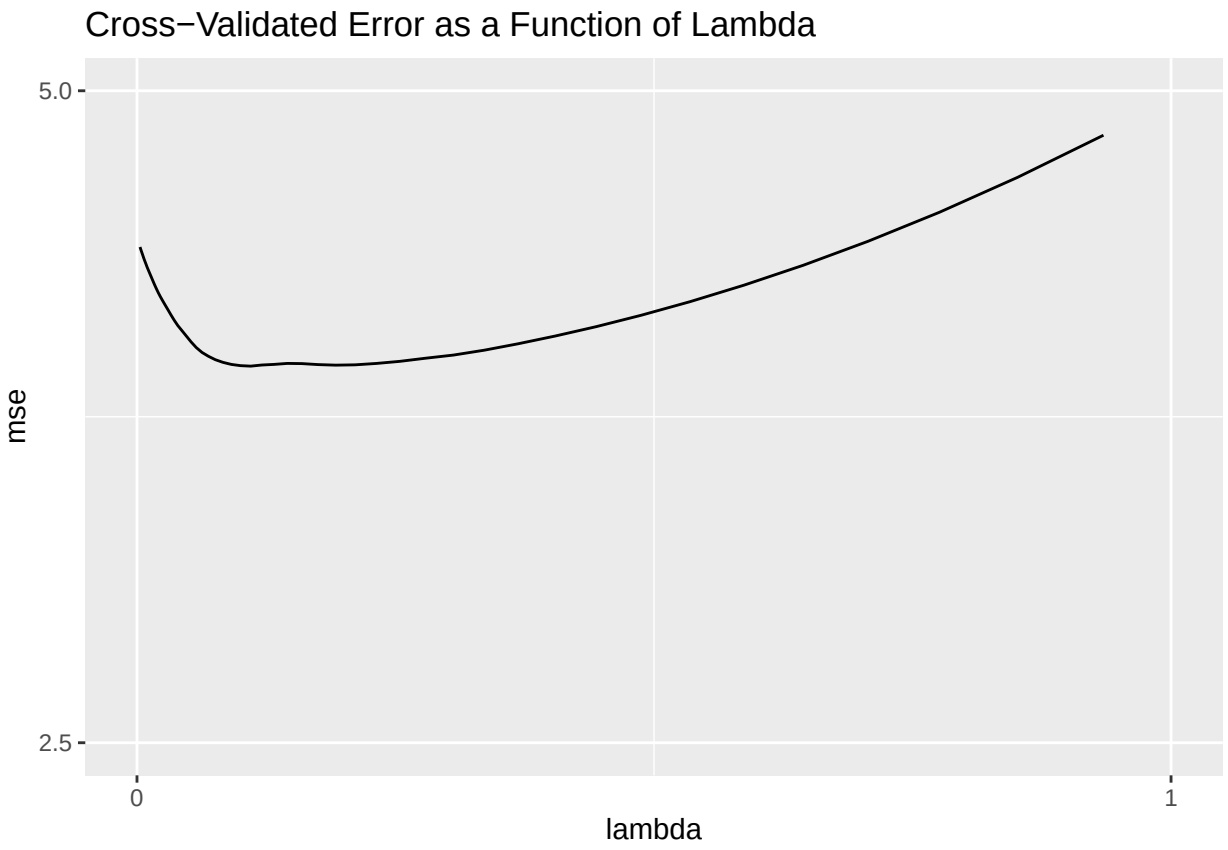
Fit a lasso model on the same train/test split as above and perform cross validation to find the optimal λ value.

```
set.seed(25)
model_lasso = glmnet(x[train,], y[train], alpha = 1, lambda = 10^seq(10,-5, length = 1000))
cv_model_lasso = cv.glmnet(x[train,], y[train], alpha = 1)
```

The following plot shows how cross-validation error changes as λ increases.

```
cv_lasso_df = data.frame("lambda" = cv_model_lasso$lambda, "mse" = cv_model_lasso$cvm)
ggplot(cv_lasso_df, aes(x = lambda, y = mse)) + geom_path() +
  ggtitle("Cross-Validated Error as a Function of Lambda") +
  scale_x_continuous(limits = c(0, 1), breaks = c(0,1)) +
  scale_y_continuous(limits = c(2.5,5), breaks = c(2.5,5))
```

```
## Warning: Removed 16 rows containing missing values (geom_path).
```



The optimal λ value is

```
cv_model_lasso$lambda.min
```

```
## [1] 0.1099895
```

That is really close to 0.. The λ value obtained from ridge regression was slightly larger. Goes to show how specifiying which norm to use in the regression equation will impact the model.

Using this value, predict on the test set.

```
pred = predict(model_lasso, s = cv_model_lasso$lambda.min, newx = x[test,])
mean((pred - y[test])^2)
```

```
## [1] 1.220111
```

This is a lower test set MSE than the one obtained by ridge regression!

Now refit the lasso regression model on the full data set and predict using the value of λ above to attain the coefficient estimates.

```
model_lasso_best = glmnet(x, y, alpha = 1)
predict(model_lasso_best, type = "coefficients", s = cv_model_lasso$lambda.min)
```

```
## 42 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)      -0.93375394
## schoolMS          .
## sexM              .
## age              -0.08406453
## addressU          .
## famsizeLE3        .
## PstatusT          .
## Medu              .
## Fedu              .
## Mjobhealth        .
## Mjobother         .
## Mjobservices      .
## Mjobteacher       .
## Fjobhealth        .
## Fjobother         .
## Fjobservices     -0.13182678
## Fjobteacher       .
## reasonhome       -0.03143406
## reasonother       .
## reasonreputation  .
## guardianmother    0.02990890
## guardianother     .
## traveltime        .
## studytime         .
## failures         -0.08779662
## schoolsupyes      .
## famsupyes         .
## paidyes           .
## activitiesyes    -0.07991598
## nurseryyes        .
## higheryes         .
## internetyes       .
## romanticyes      -0.11795859
```

```
## famrel            0.22210593
## freetime          .
## goout             .
## Dalc              .
## Walc              0.02713614
## health            .
## absences          0.02660656
## G1                0.13354232
## G2                0.96510497
```

Just as expected! The lasso regression model allows some coefficient estimates to be zero and it did as so. 29 of the 41 variables have been effectively wiped out. This is better than what subset selection gave. Hence the lasso model with λ chosen by cross validation contains only 12 variables.

But there's two more techniques to consider.

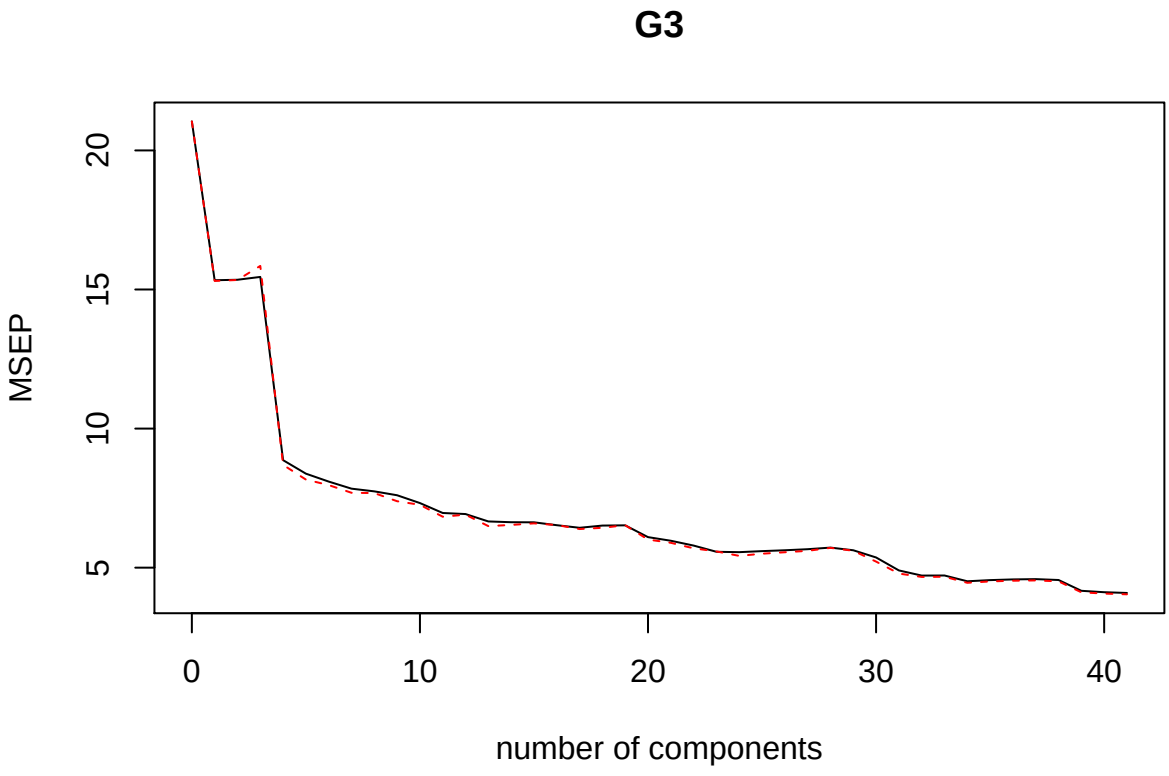
Principal Components Regression

Perform PCR on the entire dataset to predict `G3` grades.

```
set.seed(25)
model_pcr = pcr(G3 ~ ., data = df, scale = TRUE, validation = "CV")
```

The cross validation score for each possible number of components is plotted below.

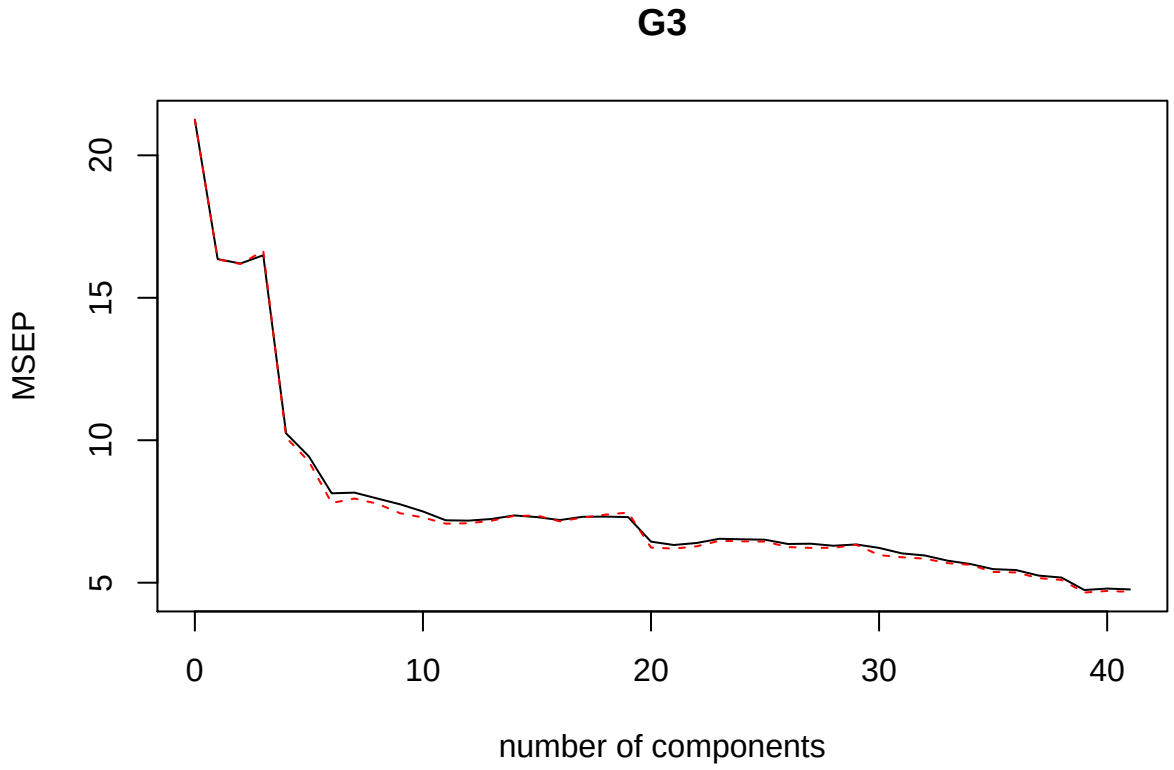
```
validationplot(model_pcr, val.type = "MSEP")
```



The smallest cross-validation error occurs when $M = 40$.

Now perform PCR on the same train/test split as before. Plot the cross-validation errors.

```
set.seed(25)
cv_model_pcr = pcr(G3~., data = df[train,], scale = TRUE, validation = "CV")
validationplot(cv_model_pcr, val.type = "MSEP")
```



The lowest cross-validation occur occurs when $M = 39$. Compute the test set MSE using this M value.

```
pred = predict(cv_model_pcr, x[test,], ncomp = 39)
mean((pred - y[test])^2)
```

```
## [1] 1.96327
```

This model has performed slightly worse than the ridge and lasso regression.

Now fit PCR on the full data set using $M = 39$.

```
model_pcr_full = pcr(y~x, scale = TRUE, ncomp = 39)
summary(model_pcr_full)
```

```
## Data:    X dimension: 395 41
##  Y dimension: 395 1
## Fit method: svdpc
## Number of components considered: 39
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    9.101    15.55    20.73    25.57    29.95    34.08    38.01    41.65
## y   27.147    27.68    27.89    60.20    63.01    64.46    65.53    66.06
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    45.14     48.55     51.76     54.85     57.73     60.48     63.12
## y    67.75     68.07     70.08     70.14     71.72     71.75     72.14
##    16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X     65.61     68.05     70.37     72.66     74.81     76.85     78.78
## y     72.40     72.92     73.21     73.22     75.26     75.78     76.44
```

```
##    23 comps  24 comps  25 comps  26 comps  27 comps  28 comps  29 comps
## X     80.63     82.44     84.17     85.81     87.38     88.93     90.34
## y     76.58     77.61     77.61     77.64     77.66     77.68     78.32
##    30 comps  31 comps  32 comps  33 comps  34 comps  35 comps  36 comps
## X     91.60     92.84     94.04     95.18     96.18     97.11     97.90
## y     79.97     81.46     81.46     81.60     82.24     82.24     82.28
##    37 comps  38 comps  39 comps
## X     98.66     99.26     99.56
## y     82.29     82.47     84.14
```

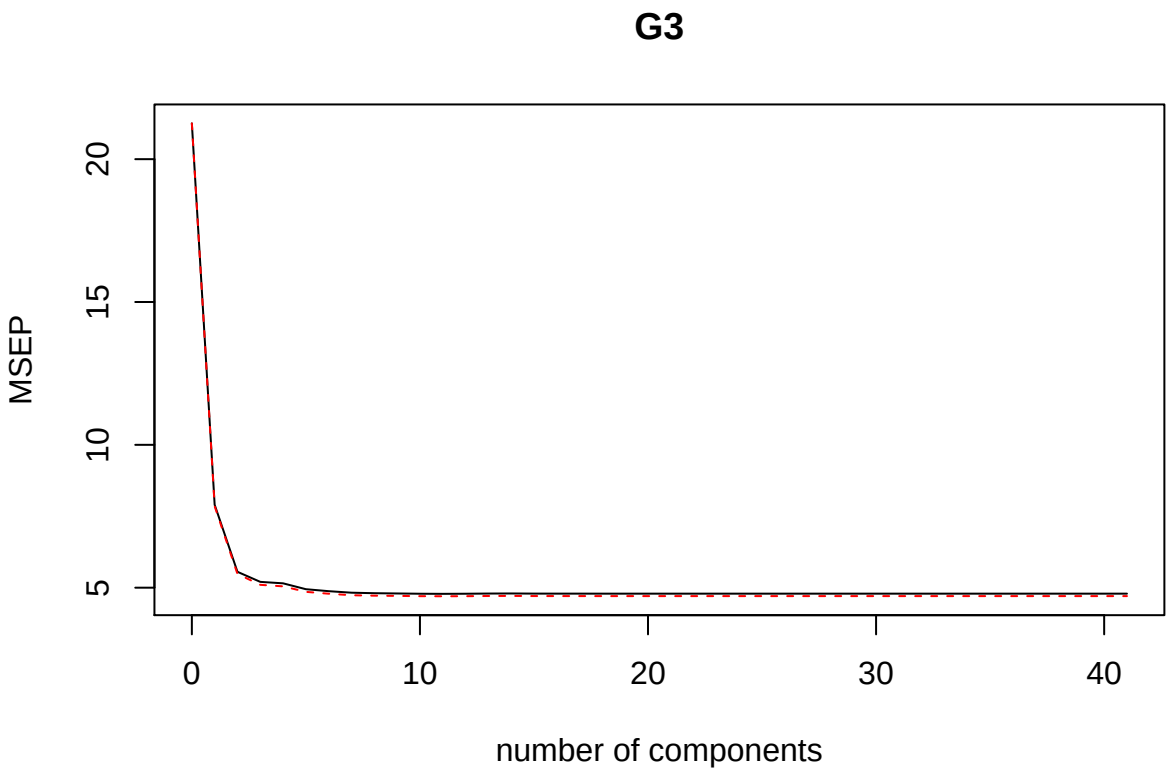
Using 39 components, 99.56% of the variation in the predictors and 84.14% of the variation in the response is explained by the 39 components. Note that the PCR does not try to explain the variance in the response. That's what the next method will do.

Last method, PLS (no pun intended).

Partial Least Squares

Fit a PLS model on training set and plot the cross-validation error.

```
cv_model_pls = plsr(G3~., data = df[train,], scale = TRUE, validation = "CV")
validationplot(cv_model_pls, val.type = "MSEP")
```



The lowest cross-validation error occurs when..

```
summary(cv_model_pls)
```

```
## Data:    X dimension: 276 41
##  Y dimension: 276 1
## Fit method: kernelpls
## Number of components considered: 41
```

```
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4.61    2.811    2.357    2.281    2.270    2.224    2.209
## adjCV         4.61    2.802    2.341    2.258    2.246    2.203    2.188
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       2.196    2.192    2.190     2.188     2.187     2.188      2.19
## adjCV    2.176    2.172    2.171     2.169     2.168     2.168      2.17
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV         2.19     2.189     2.189     2.189     2.189     2.189
## adjCV      2.17     2.170     2.169     2.169     2.169     2.169
##        20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## CV        2.189     2.189     2.189     2.189     2.189     2.189
## adjCV     2.169     2.169     2.169     2.169     2.169     2.169
##        26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
## CV        2.189     2.189     2.189     2.189     2.189     2.189
## adjCV     2.169     2.169     2.169     2.169     2.169     2.169
##        32 comps  33 comps  34 comps  35 comps  36 comps  37 comps
## CV        2.189     2.189     2.189     2.189     2.189     2.189
## adjCV     2.169     2.169     2.169     2.169     2.169     2.169
##        38 comps  39 comps  40 comps  41 comps
## CV        2.189     2.189     2.189     2.189
## adjCV     2.169     2.169     2.169     2.169
##
## TRAINING: % variance explained
##     1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X     7.958    14.01    16.80    20.55    24.00    27.36    29.76    33.06
## G3   66.895    79.00    82.36    83.27    83.77    84.01    84.21    84.28
##     9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X     36.16     38.88     41.33     43.98     46.26     47.92     50.47
## G3    84.32     84.34     84.35     84.35     84.35     84.36     84.36
##     16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X      52.13     54.72     56.68     58.67     60.26     62.21     63.91
## G3     84.36     84.36     84.36     84.36     84.36     84.36     84.36
##     23 comps  24 comps  25 comps  26 comps  27 comps  28 comps  29 comps
## X      66.40     68.71     70.83     73.18     75.49     77.20     78.85
## G3     84.36     84.36     84.36     84.36     84.36     84.36     84.36
##     30 comps  31 comps  32 comps  33 comps  34 comps  35 comps  36 comps
## X      80.89     82.59     84.08     86.35     88.06     89.84     91.79
## G3     84.36     84.36     84.36     84.36     84.36     84.36     84.36
##     37 comps  38 comps  39 comps  40 comps  41 comps
## X      93.38     95.42     96.83     98.18    100.00
## G3     84.36     84.36     84.36     84.36     84.36
```

When $M = 10$! 10 partial least squares direction is needed to explain the variation in the response and predictors.

Using this M value, predict on the test set and report the MSE.

```
pred = predict(cv_model_pls, x[test,], ncomp = 10)
mean((pred - y[test])^2)
```

```
## [1] 1.883517
```

The test set error is lower here than the one obtained by PCR.

Perform PLS on the entire data set using $M = 10$.

```
model_pls_full = plsr(y~x, data = df, scale = TRUE, ncomp = 10)
summary(model_pls_full)
```

```
## Data:    X dimension: 395 41
##  Y dimension: 395 1
## Fit method: kernelpls
## Number of components considered: 10
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    7.739    13.40    16.18    19.86    24.06    26.68    28.99    31.70
## y   68.318    79.88    82.80    83.52    83.89    84.26    84.48    84.55
##    9 comps  10 comps
## X    34.91     37.89
## y    84.57     84.57
```

Note that with PLS, 84.57% of the variance in the final math grade (and 37.89% of the variance in the predictors) is explained by 10 components. Contrasting with that, 84.14% of the variance in the final math grade was explained by 39 components in the PCR model. The PLS outperformed the PCR here in explaining the variance in the final math grade. This came at a cost of a huge decrease in the percent of variance explained explained in the predictors.

All of the lab instructions in this document are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.