Computer Assignment 10 - Linear Regression

Machine Learning, Spring 2020

Rui Li

In this assignment, we will learn how to implement three important regression techniques - linear regression, ridge regression and the LASSO - in **R**. We'll apply these three modeling techniques to the preloaded **R** dataset *mtcars*.

1.Regression overview and loading the data

Linear regression is used to model a continuous response variable y as a linear combination of p predictors taking values $x_1, \dots, x_p$. Suppose that one observes n samples of y and associated predictors (in this case $y, x_1, \dots, x_p$ are all n dimensional vectors). Define $X: = (x_1 \hskip .1pc x_2 \ldots x_p)$ as the $n \times p$ design matrix. The stochastic linear regression model of y on $x_1, \dots x_p$ is given by

$$(1): \quad y = X\beta + \epsilon$$

where $\epsilon = (\epsilon_1, \dots, \epsilon_p)^T$ is a vector of uncorrelated errors with mean 0 and variance 1 and $\beta = (\beta_1, \dots, \beta_p)^T$ is the coefficient vector of unknown parameters. We typically assume a stronger condition that $\epsilon_i \overset{iid}{\sim} N(0,1)$ to simplify statistical inference on $\beta_0, \dots, \beta_p$. One should be careful when applying model (1) as there are many conditions that should be verified. We don't discuss these conditions here, though we recommend reading more about model selection for linear regression.

Recall from class that the least squares estimates $\hat{\beta}$ is given by the *normal equations*:

$$(2): \quad \hat{\beta} = (X^T X)^{-1} X^T y$$

As we can see from (2), the calculation of $\hat{\beta}$ relies upon the invertibility of $X^T X$. Even if we assume $p < n$, we still require that there is no perfectly linear dependence between the predictor vectors $x_1, \dots, x_p$. In other words, we require the design matrix X to have full rank p. When $rank(X) < p$, then X suffers from *multicollinearity* in which case additional tools are needed for estimation of β. For example, penalization methods like ridge regression (squared penalization) or Lasso (L1 penalization) can be used to ``shrink" the estimates of β towards the origin.

We will use the *mtcars* dataset available in **R** as an example throughout this assignment. This dataset describes various quantitative features of 32 different automobiles. There are 11 total variables in this dataset. We will study *miles per gallon (mpg)* as a function of four other predictors:

1. disp: displacement (cu.in.)

2. hp: gross horsepower
3. drat: rear axle ratio
4. wt: weight (lb/1000)

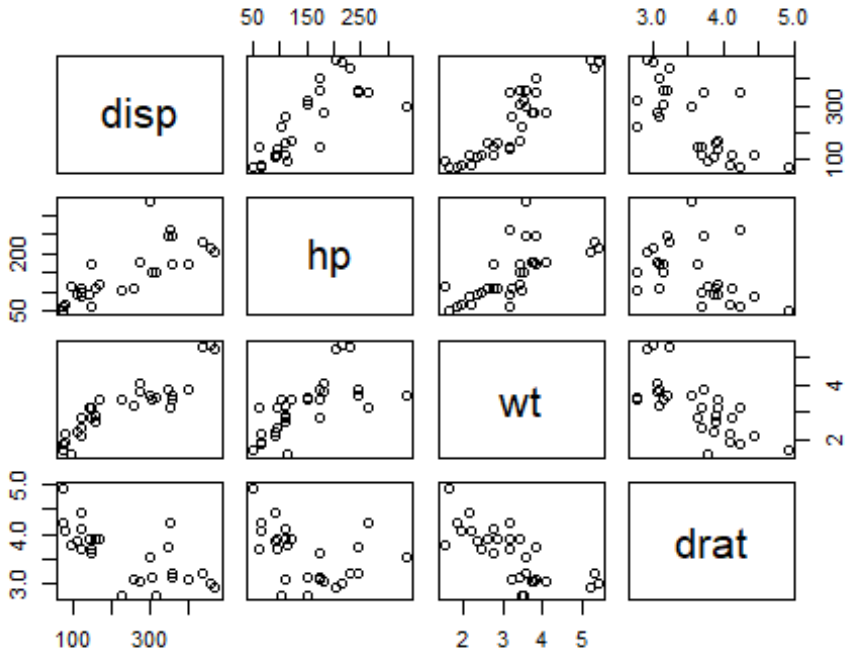
Load and parse the data with the following code:

```
#load the data
data(mtcars)

#create design matrix
X = data.frame(disp = mtcars$disp, hp = mtcars$hp, wt = mtcars$wt, drat = mtc
ars$drat)

#create response
Y = mtcars$mpg
```

Questions:

1. To get an initial idea of pairwise relationships among the predictors, plot a grid of pairwise scatterplots on *X* using the *pairs()* command. Comment on the grid of pairwise scatterplots. Do any pair of predictors appear to share a linear relationship with one another?

```
pairs(X)
```



There is a general positive relationship between pairs (disp,hp); (disp, wt); (hp,wt), and a general negative relationship between pairs (disp, drat); (hp, drat); (wt, drat).

2. What if a pair of predictors have a perfect linear relationship (i.e their correlation is 1 or -1)? Can we still estimate β using non-penalized linear regression? In this case, is $X^T X$ invertible? Why or why not?

```
If a pair of predictors have a perfect linear relationship, then X does not h
ave full rank p, and X suffers from multicollinearity. In this case, we can n
ot only use non-penalized linear regression to estimate beta, and because X i
s not in full rank, X^TX is not invertible.
```

3. What can you say about the least squares estimates $\hat{\beta}$ when the correlation between a pair of predictors gets close to 1 or -1? (HINT: think about the what happens to the empirical variance of X in this case. How does this affect the estimates $\hat{\beta}$?)

```
When the correlation between a pair of predictors gets close to 1 or -1, it's
 not possible to get the estimate value if X^TX is not invertible. Also, the
variance of the estimate will be very huge, showing that the estimate value i
s not reliable.
```

2. Linear Regression:

The `lm(y ~ x, data)` command can be used to run a linear regression of y on x. Here, y and x are both n dimensional vectors. The *data* argument is optional and specifies the source (a data frame) which x is contained. Once a linear regression has been run, we can use the *summary( )* command to obtain coefficient estimates, standard errors of estimates, and p-values measuring the significance of each coefficient in the fitted model. Please type *?lm* for more details. Fit and summarize a linear model of mpg on the remaining variables using the following code:

```
linear.reg = lm(Y ~ disp + hp + wt + drat, data = X)
summary(linear.reg)
```

```
##
## Call:
## lm(formula = Y ~ disp + hp + wt + drat, data = X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5077 -1.9052 -0.5057  0.9821  5.6883
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.148738   6.293588   4.631  8.2e-05 ***
## disp         0.003815   0.010805   0.353  0.72675
## hp          -0.034784   0.011597  -2.999  0.00576 **
## wt          -3.479668   1.078371  -3.227  0.00327 **
## drat         1.768049   1.319779   1.340  0.19153
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.602 on 27 degrees of freedom
```

```
## Multiple R-squared:  0.8376, Adjusted R-squared:  0.8136
## F-statistic: 34.82 on 4 and 27 DF,  p-value: 2.704e-10
```

Now to demonstrate a situation of perfect collinearity, consider including a fifth covariate which is exactly twice the value of the *hp* variable. Construct a new design matrix and try fitting a linear model using the following code:

```
#construct a new design matrix
X.new = data.frame(X, two.hp = 2*X$hp)
#attempt to fit a linear model
linear.reg.fail = lm(Y ~ disp + hp + wt + drat + two.hp, data = X.new)
#summarize the regression
summary(linear.reg.fail)

##
## Call:
## lm(formula = Y ~ disp + hp + wt + drat + two.hp, data = X.new)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5077 -1.9052 -0.5057  0.9821  5.6883
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.148738   6.293588   4.631  8.2e-05 ***
## disp         0.003815   0.010805   0.353  0.72675
## hp          -0.034784   0.011597  -2.999  0.00576 **
## wt          -3.479668   1.078371  -3.227  0.00327 **
## drat         1.768049   1.319779   1.340  0.19153
## two.hp             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.602 on 27 degrees of freedom
## Multiple R-squared:  0.8376, Adjusted R-squared:  0.8136
## F-statistic: 34.82 on 4 and 27 DF,  p-value: 2.704e-10
```

Questions:

1. What are the estimated coefficients on each predictor in the *linear.reg* model?

```
The estimated coefficients of `disp`, `hp`, `wt`, `drat` are 0.0038, -0.0348,
 -3.4797, 1.7680 respectively.
```

2. Which of these coefficients are statistically significant (at a 0.05 level)? Write the p-value for each of the estimated coefficients.

```
The p-value of `disp`, `hp`, `wt`, `drat` are 0.73, 0.005, 0.003, and 0.19 re
spectively. Among these p-value, `hp` is 0.00576 p-value and `wt` is 0.00327
p-value. These two p-value are statistically significant.
```

3. What did **R** do when we introduced perfect collinearity in *linear.reg.fail*? Since we know that $X^T X$ is not invertible in this case, how did **R** still fit the model? (HINT: note

that the coefficients on the original 4 variables remained the same as those in *linear.reg*.)

```
R ignores the variable `two.hp`, and print NA on its statistical values. Also,
 R still operates the same linear regression model on the rest of the variabl
es.
```

3.Ridge Regression:

The *lm.ridge(y ~ x, data, lambda)* command is used to run a ridge regression of y on x with ridge parameter *lambda*. Here, y, x and *data* have the same interpretation as in the *lm()* function described in question 2. The *lm.ridge()* command is available in the *MASS* package in **R**, so be sure to load this package before use. Importantly, one must specify the ridge parameter λ for his/her/their choice of model. One principled way to choose λ is through cross validation.

For the moment, let's try a few fixed values of λ. First, fit a ridge regression for a fixed value of $\lambda = 1$. Also, calculate the distance $\hat{\beta}$ is from the origin (a.k.a the magnitude of $\hat{\beta}$) using the following code:

```
library('MASS')
#run ridge regression with lambda = 1
ridge.reg.1 = lm.ridge(Y ~ disp + hp + wt + drat, data = X, lambda = 1)

#get a summary of the fit
coef.ridge.1 = coef(ridge.reg.1)

#calculate the magnitude of the estimated coefficient vector
dist.reg.1 = sum(abs(coef.ridge.1))
```

Questions:

1. Write the estimated coefficients for the fitted model with $\lambda = 1$. What is magnitude of $\hat{\beta}$ in this model?

```
estimated_coefficients_lambda1 = as.double(coef.ridge.1)
magnitude_lambda1 = sum(abs(coef.ridge.1)*abs(coef.ridge.1))
cat("The estimated coefficients for the fitted model with lambda = 1 is ",est
imated_coefficients_lambda1)

## The estimated coefficients for the fitted model with lambda = 1 is  28.481
74 -0.001067913 -0.03138726 -3.014687 1.712299

cat(", and the magnitude is ", magnitude_lambda1,".")

## , and the magnitude is  823.231 .
```

2. Fit a ridge regression with $\lambda = 0$. Write down the estimated coefficients and the magnitude of $\hat{\beta}$. How do the estimated coefficients here compare to those found with non-penalized linear regression? Explain why your observation makes sense.

```
library('MASS')
#run ridge regression with lambda = 0
```

```
ridge.reg.0 = lm.ridge(Y ~ disp + hp + wt + drat, data = X, lambda = 0)
#get a summary of the fit
coef.ridge.0 = coef(ridge.reg.0)
#Calculate the coefficients and magnitude
estimated_coefficients_lambda0 = as.double(coef.ridge.0)
magnitude_lambda0 = sum(abs(coef.ridge.0)*abs(coef.ridge.0))
cat("The estimated coefficients for the fitted model with lambda = 0 is ",est
imated_coefficients_lambda0)
```

```
## The estimated coefficients for the fitted model with lambda = 0 is  29.148
74 0.003815241 -0.03478353 -3.479668 1.768049
```

```
cat(", and the magnitude is ", magnitude_lambda0,".")
```

```
## , and the magnitude is  864.8842 .
```

```
The estimated coefficients here are the same as those found with non-penalize
d linear regression. Because when lambda is zero, the equation of the estimat
ed coefficients is exactly the same as the ordinary equation. When lambda is
zero, indicating that there is no penalized in the model. Since they are the
same model, the coefficients should be equal as well.
```

3. Fit a ridge regression with $\lambda = 10{,}50{,}100$, and 1000. For each value, calculate the magnitude of the estimated coefficient vector. What happens to the magnitude of $\hat{\beta}$ as you increase λ? This is an example of the "shrinkage" effect of ridge regression.

```
for (lambda in c(10, 50, 100, 1000)) {
  library('MASS')
  ridge.reg = lm.ridge(Y ~ disp + hp + wt + drat, data = X, lambda = lambda)
  coef.ridge= coef(ridge.reg)
  #Calculate the coefficients and magnitude
  estimated_coefficients_lambda = as.double(coef.ridge)
  magnitude_lambda = sum(abs(coef.ridge)*abs(coef.ridge))
  print(paste0("The estimated coefficient vector magnitude of lambda ",lambda,
" is ",magnitude_lambda,"."))
}
```

```
## [1] "The estimated coefficient vector magnitude of lambda 10 is 663.985020
669847."
## [1] "The estimated coefficient vector magnitude of lambda 50 is 523.947445
503118."
## [1] "The estimated coefficient vector magnitude of lambda 100 is 480.60601
3547745."
## [1] "The estimated coefficient vector magnitude of lambda 1000 is 414.6841
2410335."
```

```
When the value of lambda is increasing, the estimated coefficient vector magn
itude is decreasing.
```

4.LASSO:

The *glmnet(X, y, lambda)* command can be used to fit the LASSO model of y on X. The *glmnet* package contains the functions required to conduct LASSO. Download this package before proceeding using the *install.packages()* and *library()* commands. Like ridge regression, the LASSO relies on the choice of a penalty parameter, (call it λ_1). The *glmnet(X, y, lambda)* command is used to fit a LASSO regression with specified parameter *lambda*. In contrast to the *lm()* and *ridge.lm()* commands, the *glmnet(X, y, lambda)* command requires X to be an $n \times p$ design matrix. As usual, y is the n dimensional vector of responses. Once again, we can choose the ``best" λ_1 using cross validation but we'll come back to this later. The *coef()* command is used to summarize the estimated coefficients of the model. Fit a LASSO model with $\lambda_1 = 1$ to the *mtcars* data and calculate the magnitude of the estimated coefficients using the following commands:

```
library('glmnet')

## Warning: package 'glmnet' was built under R version 3.6.3

## Loading required package: Matrix

## Loaded glmnet 3.0-2

#conduct a cross validation study to fit minimum MSE model
lasso.fit.1 = glmnet(as.matrix(X),Y,lambda = 1)

#summarize the estimated coefficients
coef.lasso.1 = coef(lasso.fit.1)

#calculate the magnitude of estimated coefficients
dist.lasso.1 = sum(abs(coef.lasso.1))
```

Questions:

1. Write the estimated coefficients for the fitted model with $\lambda_1 = 1$. What is the magnitude of $\hat{\beta}$ in this model?

```
lasso_coefficients_lambda1 = as.double(coef.lasso.1)
lasso_magnitude_lambda1 = dist.lasso.1
cat("The estimated coefficients for the fitted model with lambda = 1 is ",las
so_coefficients_lambda1)

## The estimated coefficients for the fitted model with lambda = 1 is  31.733
73 -0.0005284494 -0.02260053 -3.035119 0.4334064

cat(", and the magnitude is ", lasso_magnitude_lambda1,".")

## , and the magnitude is  35.22538 .
```

2. Fit the LASSO with $\lambda_1 = 0$. Write down the estimated coefficients and the magnitude of $\hat{\beta}$. Do these estimates match those found in the non-penalized linear regression?

```
#conduct a cross validation study to fit minimum MSE model
lasso.fit.0 = glmnet(as.matrix(X),Y,lambda = 0)
#summarize the estimated coefficients
coef.lasso.0 = coef(lasso.fit.0)
#calculate the magnitude of estimated coefficients
dist.lasso.0 = sum(abs(coef.lasso.0))

lasso_coefficients_lambda0 = as.double(coef.lasso.0)
lasso_magnitude_lambda0 = dist.lasso.0
cat("The estimated coefficients for the fitted model with lambda = 0 is ",las
so_coefficients_lambda0)

## The estimated coefficients for the fitted model with lambda = 0 is  29.155
18 0.003763027 -0.03476093 -3.476059 1.765458

cat(", and the magnitude is ", lasso_magnitude_lambda0,".")

## , and the magnitude is  34.43522 .

The estimates are very close to those found in the non-penalized linear regre
ssion, but they are not exactly the same.
```

3. Fit the LASSO with $\lambda_1 = 10,50,100$, and 1000. For each value, calculate the magnitude of the estimated coefficient vector. What happens to the magnitude of $\hat{\beta}$ as you increase λ?

```
for (lambda in c(10,50,100,1000)) {
  #conduct a cross validation study to fit minimum MSE model
  lasso.fit = glmnet(as.matrix(X),Y,lambda = lambda)
  #summarize the estimated coefficients
  coef.lasso = coef(lasso.fit)
  #calculate the magnitude of estimated coefficients
  dist.lasso = sum(abs(coef.lasso))

   print(paste0("The estimated coefficient vector magnitude of lambda ",lambd
a," is ",dist.lasso,"."))
}

## [1] "The estimated coefficient vector magnitude of lambda 10 is 20.090625.
"
## [1] "The estimated coefficient vector magnitude of lambda 50 is 20.090625.
"
## [1] "The estimated coefficient vector magnitude of lambda 100 is 20.090625.
"
## [1] "The estimated coefficient vector magnitude of lambda 1000 is 20.09062
5."

The magnitude of estimated coefficient vector does not change when lambda is
increasing.
```

4. In this assignment, we considered a dataset where $n > p$. If we had a situation where $p > n$, what modeling framework would you consider using to fit a linear regression?

If you do not remove any of the variables, can we use non-penalized linear regression when $p > n$?

```
When p>>n, the LASSO model should be considered to fit a linear regression mo
del. Also, ridge regression can apply to p>n as well. If we are not to remove
 any of the variables, we can not use non-penalized linear regression, becaus
e "inverse does not exist if p > n, and small eigenvalues resulting from coll
inearity among features can lead to unstable estimates, unreliable prediction
s" ~~ quote from lecture slides.
```

5. High Dimensional LASSO:

We will repeat the analysis that was done in class. Start by uncommenting and running the following code to install the `BicocManager` and `bcellViper` packages. If you are having any difficulties installing these packages, you can seek further instruction here.

```
# if (!requireNamespace("BiocManager", quietly = TRUE))
#   install.packages("BiocManager")
#
# BiocManager::install("bcellViper")
# install.packages("HDCI")
library(bcellViper)
```

```
## Loading required package: Biobase
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following object is masked from 'package:Matrix':
##
##     which
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
```

```
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
data(bcellViper)
gene_expressions = data.frame(t(assayDataElement(dset,'exprs')))
```

The installation of the above packages needs only to be done once, and can be re-commented after this initial run.

Questions

1. Run an OLS model on the `gene_expressions` data, using `ADA` as your response variables, and the remaining variables as your predictors. Comment on any irregularities in the model output. Explain why you are not getting a reasonable number for your degrees of freedom in the model summary.

```
linear.OLS = lm(ADA ~., data = gene_expressions)
sum.OLS = summary(linear.OLS)
```

```
The `Error`, `t-value` and `p-value` of all variables are NA. Also, the degre
es of freedom is zero. The reason that I did not get a resonable degrees of f
reedom is because the number of variable is much larger than the sample numbe
r, and there can be collinearity between the variables. OLS model is not appr
opriate for this data.
```

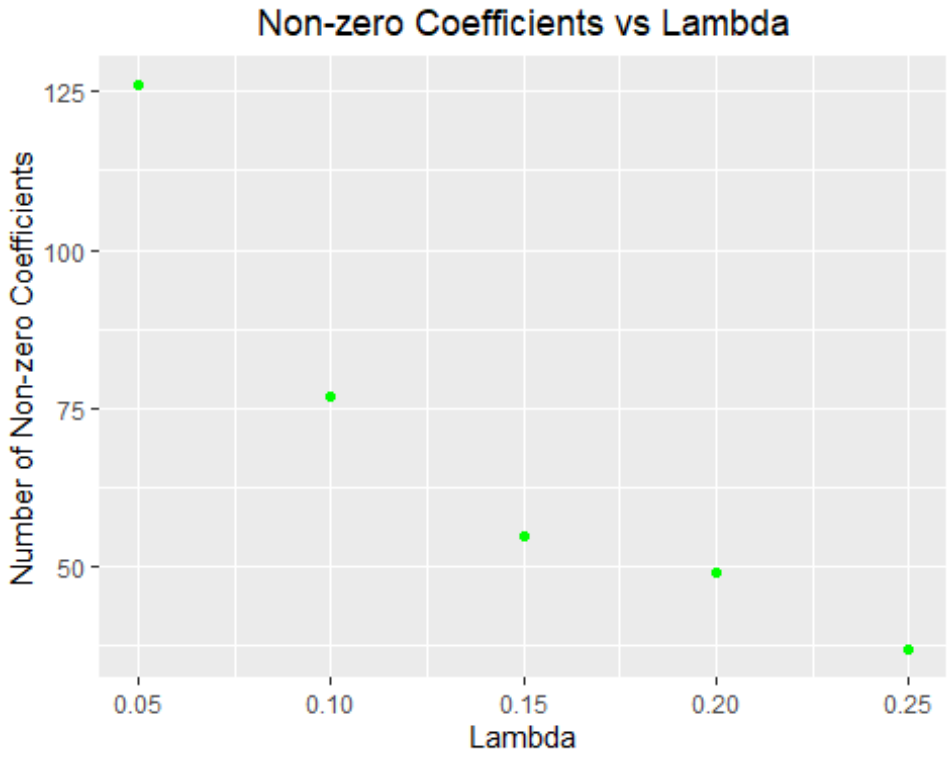
2. Now run 5 different LASSO models on this data, using any λ values of your choosing. Report how many non-zero β coefficients each model has. Plot the number of non-zero coefficients as a function of λ. Comment on any trend you observe.

```
Y = gene_expressions$ADA
X = gene_expressions[,-1]
lambdas = c(0.05, 0.10, 0.15, 0.20, 0.25)
nonzeros = c()
for (lambda in lambdas) {
  #conduct a cross validation study to fit minimum MSE model
  lasso.fit = glmnet(as.matrix(X),Y,lambda = lambda)
  #summarize the estimated coefficients
  coef.lasso = coef(lasso.fit)
  nonzero_num = sum(coef.lasso!=0)
  nonzeros = c(nonzeros,nonzero_num)
}
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
ggplot(data =as.data.frame(t(rbind(nonzeros,lambdas))), aes(y=nonzeros, x = l
ambdas), formula = y~x)+geom_point(col = 'green') + ggtitle("Non-zero Coeffic
ients vs Lambda") + labs(x = 'Lambda', y = 'Number of Non-zero Coefficients')
 + theme(plot.title = element_text(hjust = 0.5))
```



When the lambda is decreasing and close to zero, the number of non-zero coefficients increases dramatically. When the lambda is large, the number of non-zero coefficients stay constant.