Regression and Large Dimensional Data

let's make an example that uses a lot of variables but only a few that are important

100 variables, only 5 matter

```
require(magrittr)
```

```
## Loading required package: magrittr
```

```
require(dplyr)
```

```
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(glmnet)
```

```
## Loading required package: glmnet
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-2
```

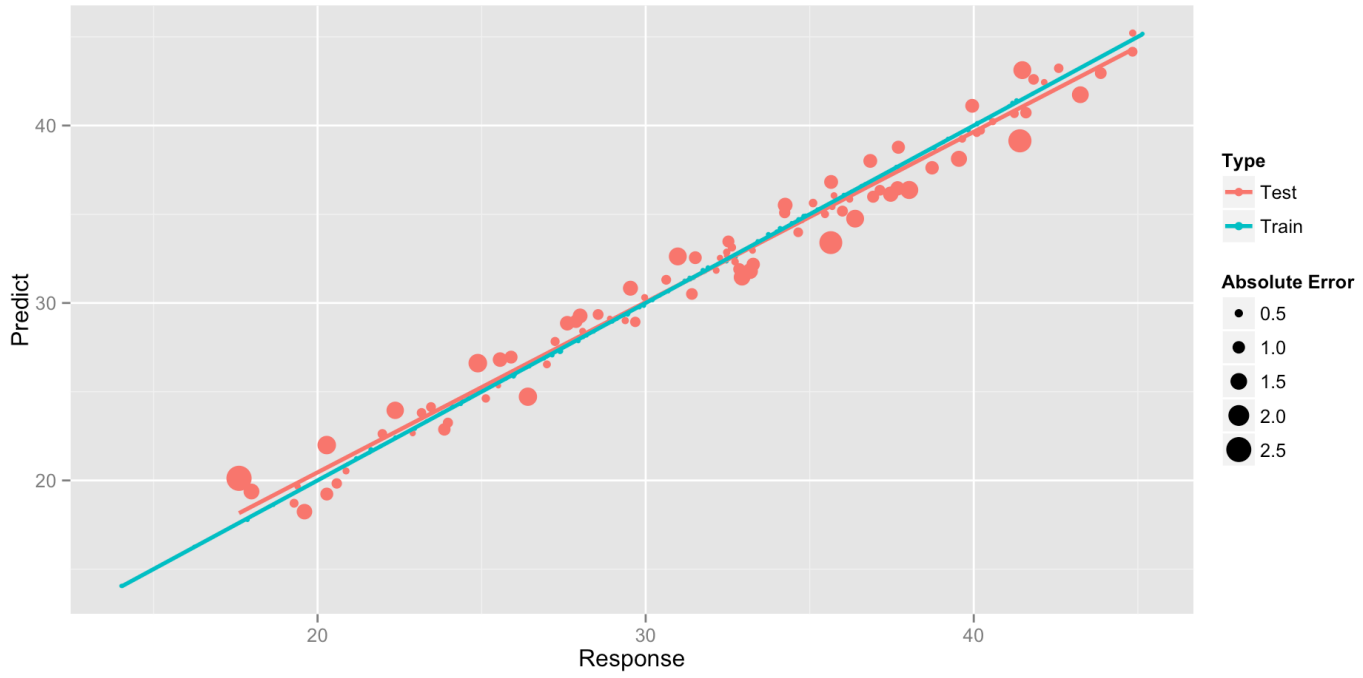
```
bigDF <- data.frame()
N = 200
for (i in 1:100)
  bigDF[1:N,i] <- runif(N)

Response2 = bigDF[,1] * 20 + bigDF[,2]^1.2 * 10 + bigDF[,3]^0.9*5+ bigDF[,4]*2.5+ bigDF[,5] + ro
wSums(0.1 * bigDF[,6:100]^0.5) + 5 + 0.01*runif(N)



sensDF <- data.frame(Method = 0, Var = 0, Value=0)

#ggpairs(bigDF[,c(101,1:10)], lower = list(continuous = "smooth"))

bigDF$Response <- Response2

summary(bigMod<-lm(Response ~ ., data=bigDF[1:110,]))
```

```
##
## Call:
## lm(formula = Response ~ ., data = bigDF[1:110, ])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.116980 -0.031672 -0.003922  0.030363  0.138537
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.738614   1.393184   7.708 2.98e-05 ***
## V1          19.385742   0.423910  45.731 5.72e-12 ***
## V2          10.107444   0.261818  38.605 2.61e-11 ***
## V3           5.426108   0.230121  23.579 2.12e-09 ***
## V4           2.381487   0.154284  15.436 8.80e-08 ***
## V5           0.422572   0.189320   2.232   0.0525 .
## V6           0.406301   0.314982   1.290   0.2292
## V7          -0.047667   0.474756  -0.100   0.9222
## V8          -0.066486   0.192609  -0.345   0.7379
## V9           0.054468   0.209459   0.260   0.8007
## V10         -0.405001   0.284678  -1.423   0.1886
## V11          0.351025   0.257204   1.365   0.2055
## V12          0.263723   0.281342   0.937   0.3730
## V13         -0.194444   0.477929  -0.407   0.6936
## V14         -0.118279   0.257905  -0.459   0.6574
## V15          0.355390   0.211268   1.682   0.1268
## V16         -0.430293   0.235396  -1.828   0.1008
## V17          0.320491   0.196228   1.633   0.1368
## V18         -0.393040   0.190530  -2.063   0.0692 .
## V19          0.198644   0.262043   0.758   0.4678
## V20         -0.538241   0.234169  -2.299   0.0471 *
## V21          0.120725   0.218968   0.551   0.5948
## V22          0.569023   0.203356   2.798   0.0208 *
## V23         -0.244563   0.260295  -0.940   0.3720
## V24          0.183754   0.230168   0.798   0.4452
## V25         -0.487774   0.206723  -2.360   0.0426 *
## V26          0.041586   0.399422   0.104   0.9194
## V27          0.145044   0.238968   0.607   0.5589
## V28          0.051078   0.168102   0.304   0.7681
## V29          0.560813   0.271092   2.069   0.0685 .
## V30         -0.203944   0.255220  -0.799   0.4448
## V31          0.399165   0.239159   1.669   0.1295
## V32         -0.072174   0.268884  -0.268   0.7944
## V33          0.350469   0.340751   1.029   0.3306
## V34         -0.229913   0.219836  -1.046   0.3229
## V35          0.170591   0.290170   0.588   0.5711
## V36         -0.838592   0.263920  -3.177   0.0112 *
## V37          0.140754   0.260996   0.539   0.6028
## V38          0.265898   0.272163   0.977   0.3541
## V39         -0.246520   0.327810  -0.752   0.4712
## V40          0.135001   0.213844   0.631   0.5435
## V41          0.119291   0.405038   0.295   0.7750
## V42         -0.059463   0.235529  -0.252   0.8064
```

```
## V43          0.214696   0.205067   1.047   0.3224
## V44         -0.192265   0.260117  -0.739   0.4787
## V45         -0.286316   0.375071  -0.763   0.4648
## V46          0.309247   0.285405   1.084   0.3067
## V47         -0.244709   0.188215  -1.300   0.2259
## V48         -0.549418   0.413928  -1.327   0.2171
## V49          0.052047   0.225736   0.231   0.8228
## V50          0.171071   0.176969   0.967   0.3590
## V51          0.028949   0.309515   0.094   0.9275
## V52         -0.062347   0.183038  -0.341   0.7412
## V53          0.585268   0.191364   3.058   0.0136 *
## V54          0.075297   0.413437   0.182   0.8595
## V55          0.467507   0.273097   1.712   0.1211
## V56         -0.245959   0.253927  -0.969   0.3580
## V57         -0.328750   0.296991  -1.107   0.2970
## V58          0.527172   0.248271   2.123   0.0627 .
## V59          0.023034   0.270560   0.085   0.9340
## V60          0.270628   0.263344   1.028   0.3309
## V61          0.374267   0.320685   1.167   0.2732
## V62         -0.119485   0.245748  -0.486   0.6384
## V63          0.248703   0.204699   1.215   0.2553
## V64         -0.220934   0.198117  -1.115   0.2937
## V65          0.280184   0.222666   1.258   0.2399
## V66         -0.092208   0.221112  -0.417   0.6864
## V67         -0.069457   0.161869  -0.429   0.6779
## V68         -0.038948   0.273400  -0.142   0.8899
## V69          0.327411   0.216473   1.512   0.1647
## V70         -0.496636   0.194497  -2.553   0.0310 *
## V71         -0.315681   0.256237  -1.232   0.2492
## V72         -0.413758   0.240120  -1.723   0.1190
## V73          0.597728   0.232357   2.572   0.0301 *
## V74          0.416578   0.333075   1.251   0.2426
## V75         -0.277243   0.213930  -1.296   0.2272
## V76         -0.416387   0.220429  -1.889   0.0915 .
## V77          0.147349   0.207527   0.710   0.4957
## V78         -0.237233   0.240912  -0.985   0.3505
## V79          0.033516   0.181003   0.185   0.8572
## V80         -0.149220   0.334485  -0.446   0.6661
## V81         -0.083760   0.281471  -0.298   0.7728
## V82          0.483341   0.286372   1.688   0.1257
## V83          0.549855   0.290017   1.896   0.0905 .
## V84          0.204970   0.235945   0.869   0.4076
## V85         -0.588766   0.197263  -2.985   0.0153 *
## V86          0.506525   0.186599   2.715   0.0238 *
## V87          0.388935   0.238843   1.628   0.1379
## V88         -0.188174   0.299135  -0.629   0.5449
## V89          0.658756   0.235388   2.799   0.0208 *
## V90         -0.375364   0.475720  -0.789   0.4504
## V91         -0.109807   0.193565  -0.567   0.5844
## V92         -0.049334   0.268519  -0.184   0.8583
## V93         -0.373771   0.276088  -1.354   0.2088
## V94         -0.085667   0.149673  -0.572   0.5811
## V95         -0.362715   0.471225  -0.770   0.4612
## V96          0.141340   0.218246   0.648   0.5334
```

```
## V97          0.082951   0.278825   0.298   0.7728
## V98         -0.418090   0.221210  -1.890   0.0913 .
## V99          0.043668   0.252971   0.173   0.8668
## V100         0.007768   0.303731   0.026   0.9802
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1866 on 9 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9991
## F-statistic:  1175 on 100 and 9 DF,  p-value: 2.956e-13
```

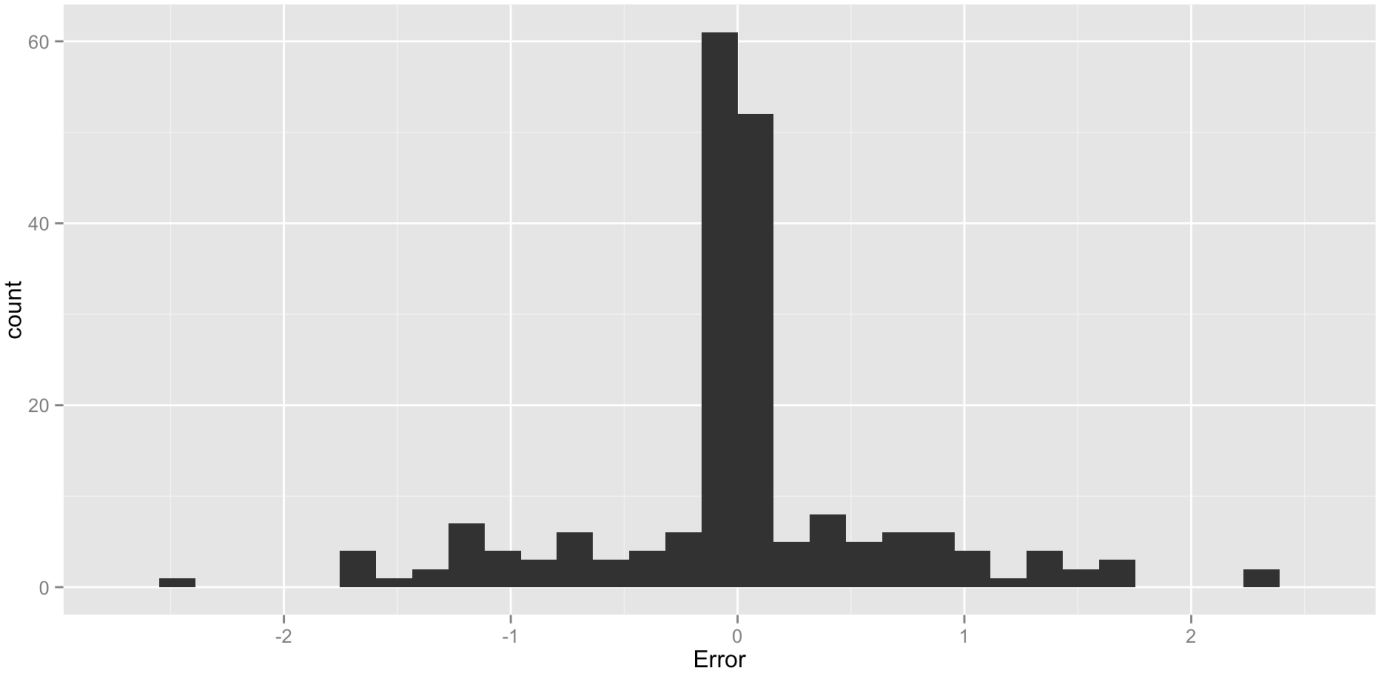
```
plotDF <- bigDF
plotDF[1:110,'Type'] <- 'Train'
plotDF[111:200,'Type'] <- 'Test'
plotDF$Predict <- predict(bigMod,plotDF)
plotDF$Error <- plotDF$Response-plotDF$Predict
ggplot(plotDF,aes(x=Response,y=Predict,color=Type,size=abs(Error))) + geom_point() +
scale_size("Absolute Error") + geom_smooth(method="lm",se=F,size=1)
```




```
sqrt(var(data.frame(plotDF %>% filter(Type=="Test") %>% select(Error))))/110
```

```
##             Error
## Error 0.009294262
```

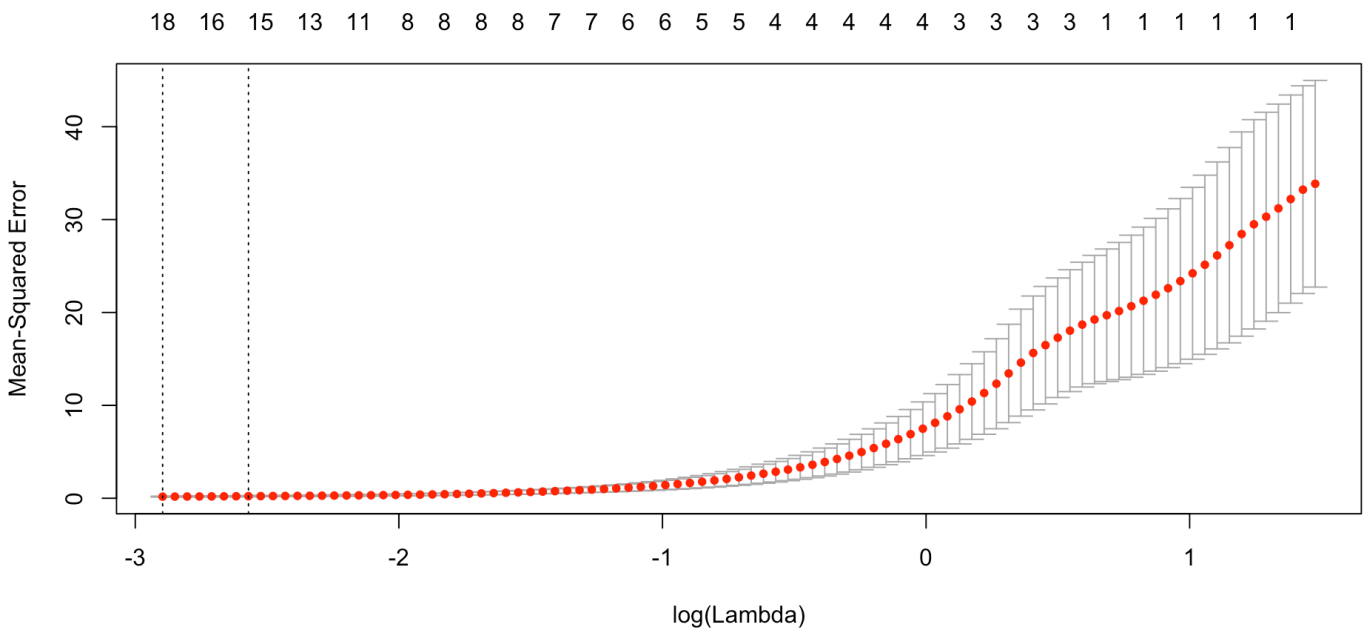
```
ggplot(plotDF,aes(x=Error)) + geom_histogram()
```



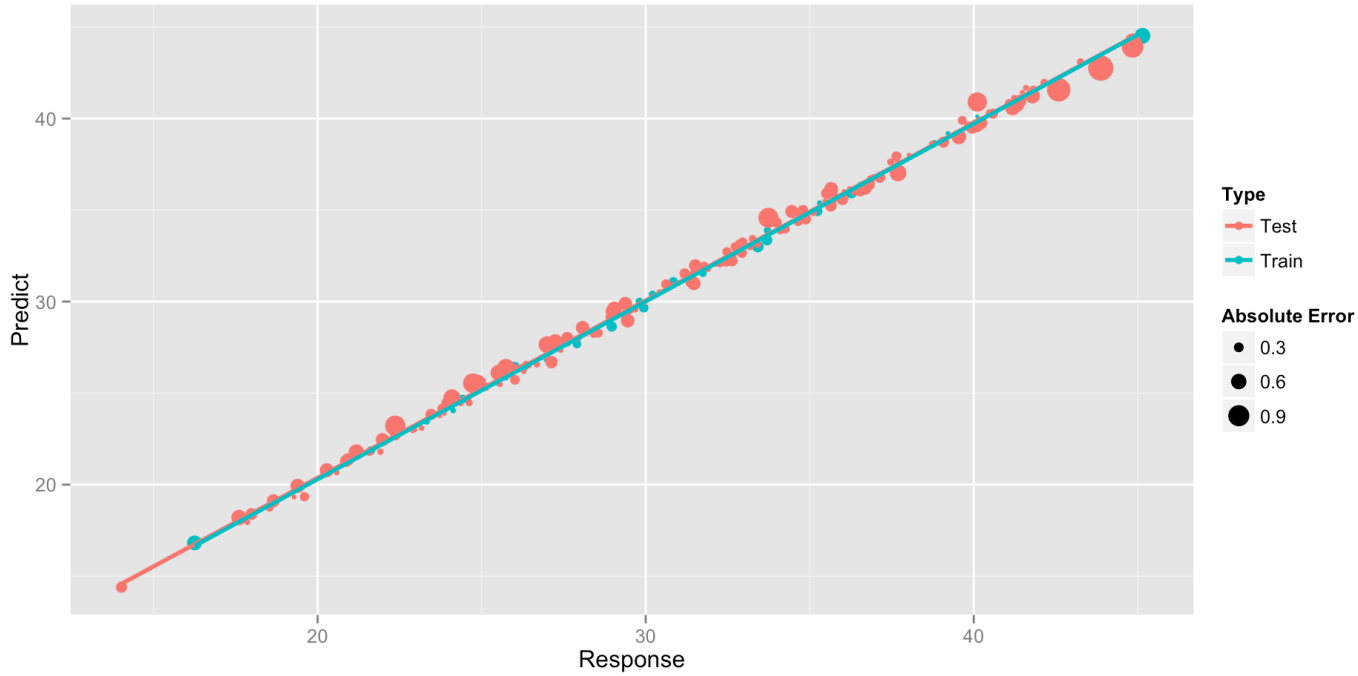

```
sensitivities <- coef(bigMod)
require(glmnet)

sensDF[1:length(sensitivities),'Method'] <- "Least-Squares"
sensDF[1:length(sensitivities),'Var'] <- names(sensitivities)
sensDF[1:length(sensitivities),'Value'] <- (sensitivities)
rowStart <- length(sensitivities)+1
```

Lasso regression

```
crossValid <- cv.glmnet(as.matrix(bigDF[1:40,1:100]),as.matrix(bigDF$Response[1:40]),alpha = 1)
plot(crossValid)
```



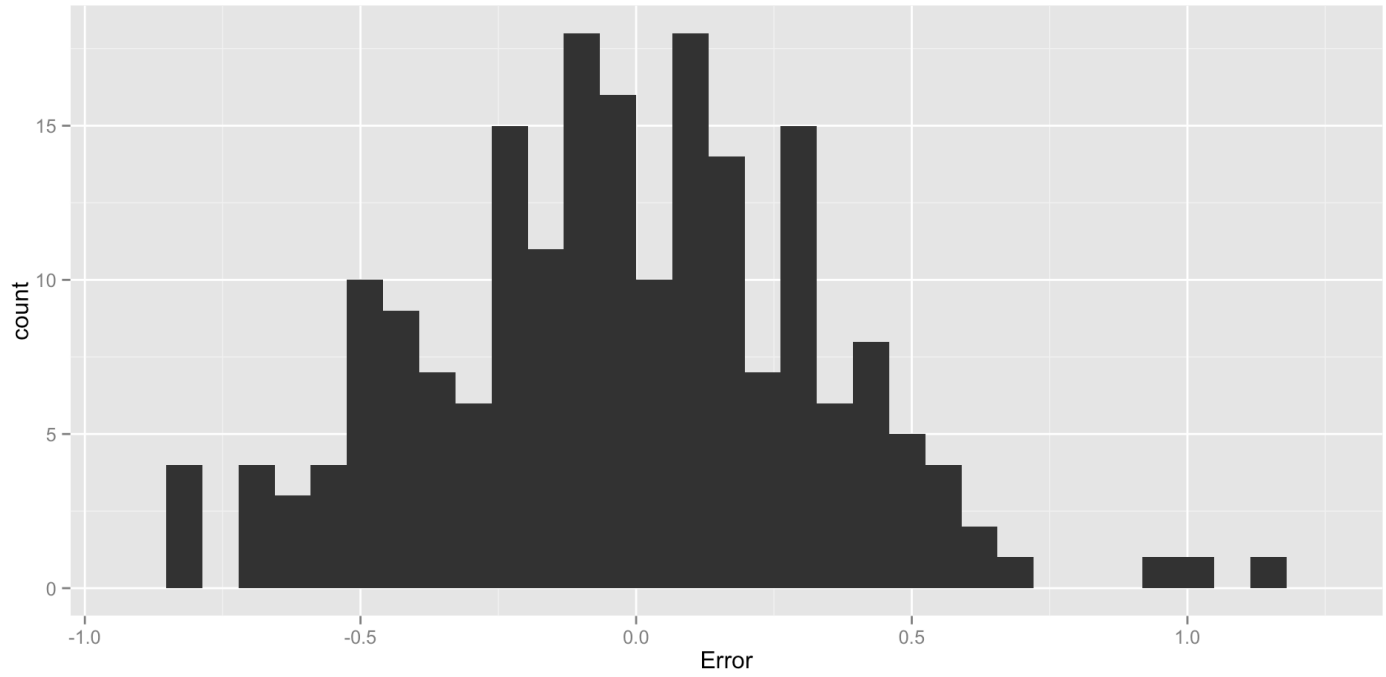

```
lambda <- crossValid$lambda.min
sensitivities <- coef(crossValid)
plotDF <- bigDF
plotDF[1:40,'Type'] <- 'Train'
plotDF[41:200,'Type'] <- 'Test'
plotDF[,"Predict" ]<- data.frame(Predict=predict(crossValid,as.matrix(plotDF[,1:100]),lambda=lam
bda))
plotDF$Error <- plotDF$Response-plotDF$Predict
ggplot(plotDF,aes(x=Response,y=Predict,color=Type,size=abs(Error))) + geom_point() +
scale_size("Absolute Error") + geom_smooth(method="lm",se=F,size=1)
```



```
sqrt(var(data.frame(plotDF %>% filter(Type=="Test") %>% select(Error))))/40
```

```
##             Error
## Error 0.009364811
```

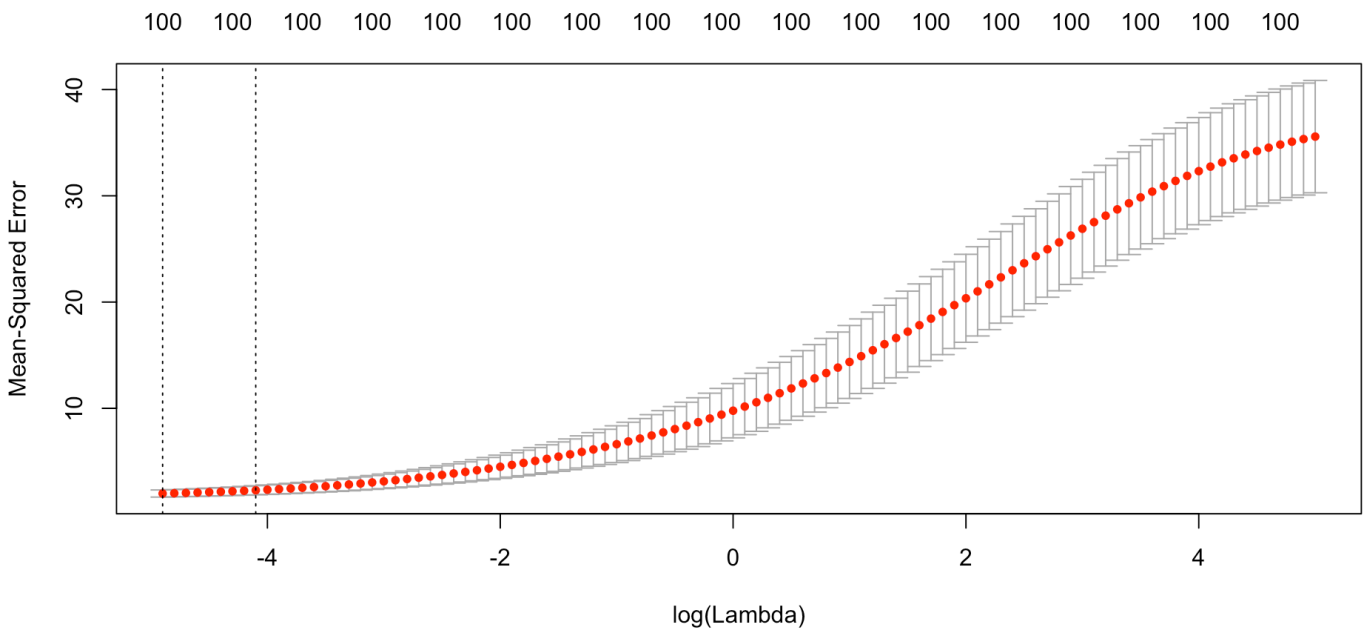
```
ggplot(plotDF,aes(x=Error)) + geom_histogram()
```



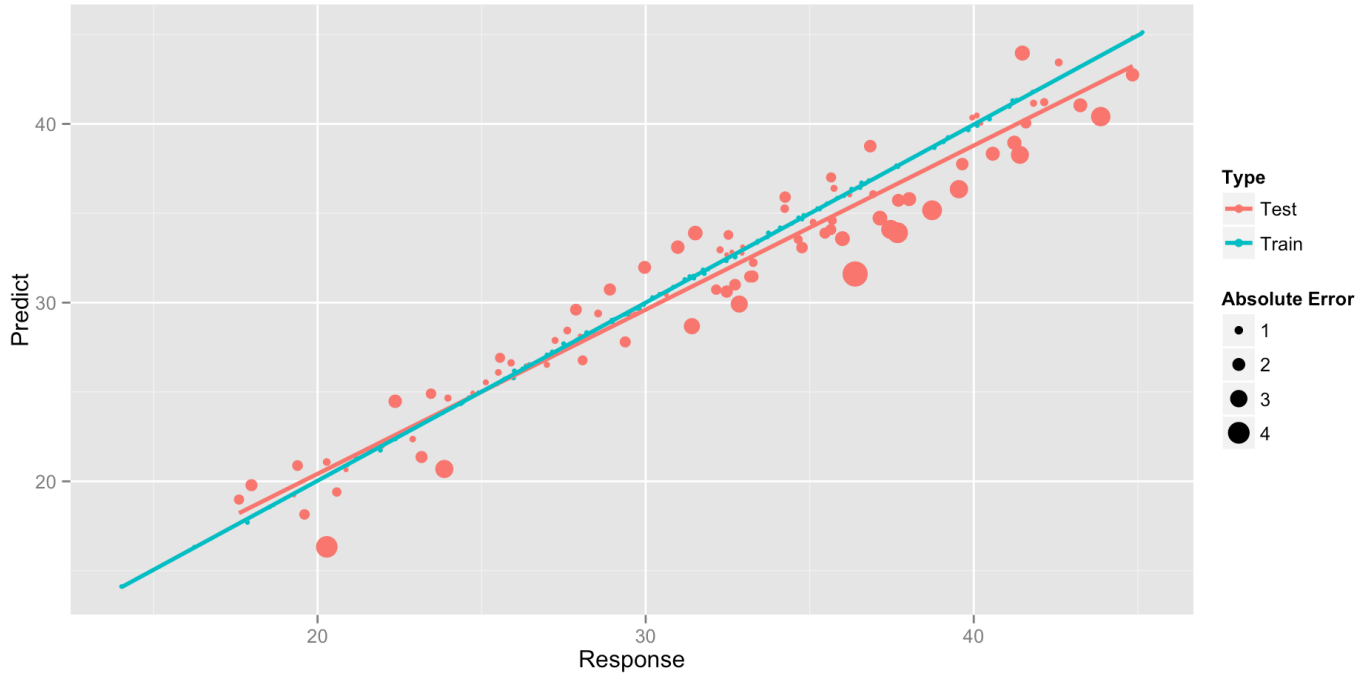

```
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Method'] <- "Lasso"
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Var'] <- t(t(rownames(sensitivities)))
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Value'] <- as.numeric(sensitivities)
rowStart <- rowStart + length(sensitivities)
```

Ridge regression

```
crossValid <- cv.glmnet(as.matrix(bigDF[1:110,1:100]),as.matrix(bigDF$Response[1:110]),alpha = 0,l
mbda=exp(seq(-5,5,by=0.1)))
plot(crossValid)
```



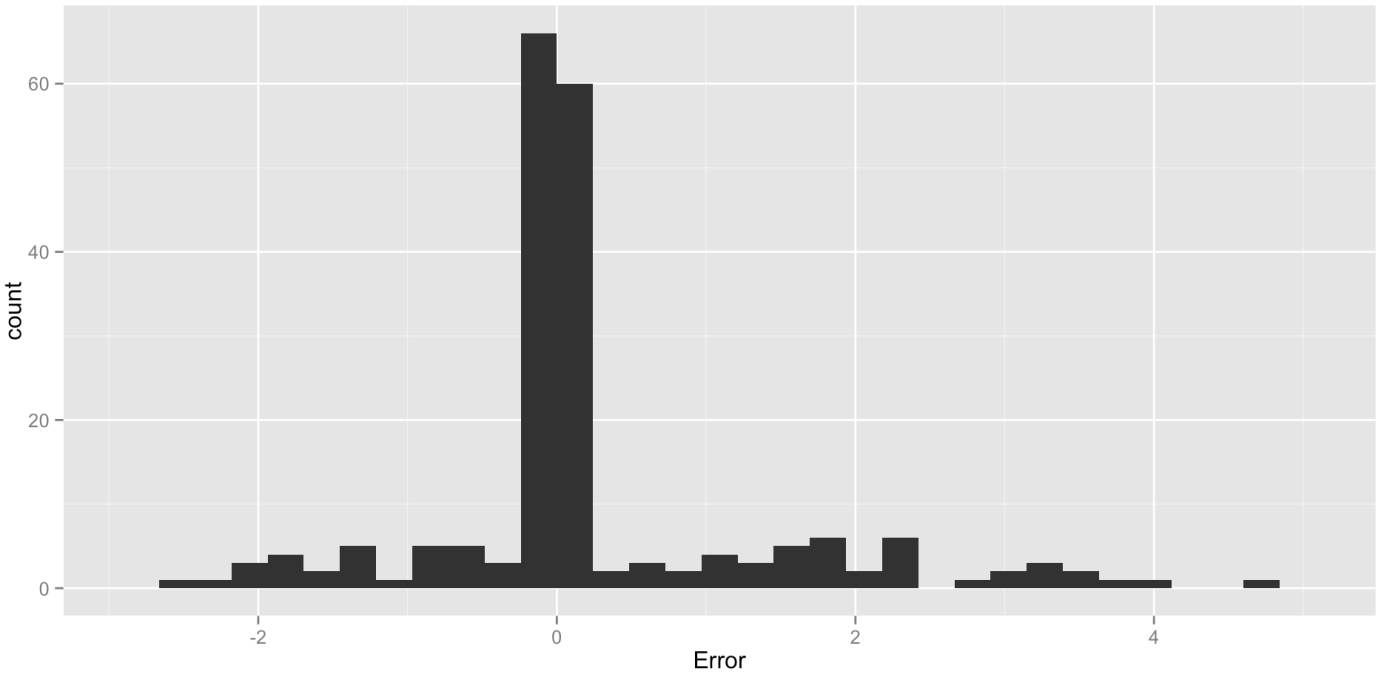

```
lambda <- crossValid$lambda.min
sensitivities <- coef(crossValid)
plotDF <- bigDF
plotDF[1:110,'Type'] <- 'Train'
plotDF[111:200,'Type'] <- 'Test'
plotDF[,"Predict" ]<- data.frame(Predict=predict(crossValid,as.matrix(plotDF[,1:100]),lambda=lam
bda))
plotDF$Error <- plotDF$Response-plotDF$Predict
ggplot(plotDF,aes(x=Response,y=Predict,color=Type,size=abs(Error))) + geom_point() +
scale_size("Absolute Error") + geom_smooth(method="lm",se=F,size=1)
```



```
sqrt(var(data.frame(plotDF %>% filter(Type=="Test") %>% select(Error))))/110
```

```
##           Error
## Error 0.0153188
```

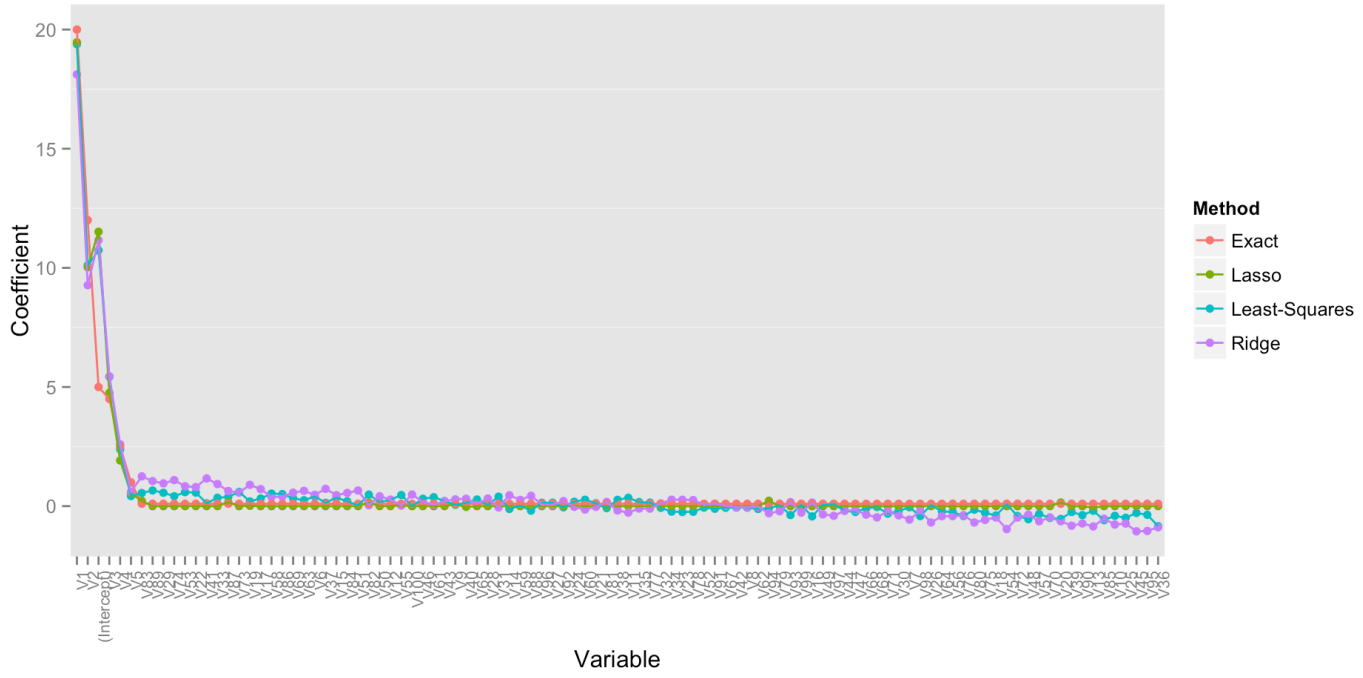
```
ggplot(plotDF,aes(x=Error)) + geom_histogram()
```



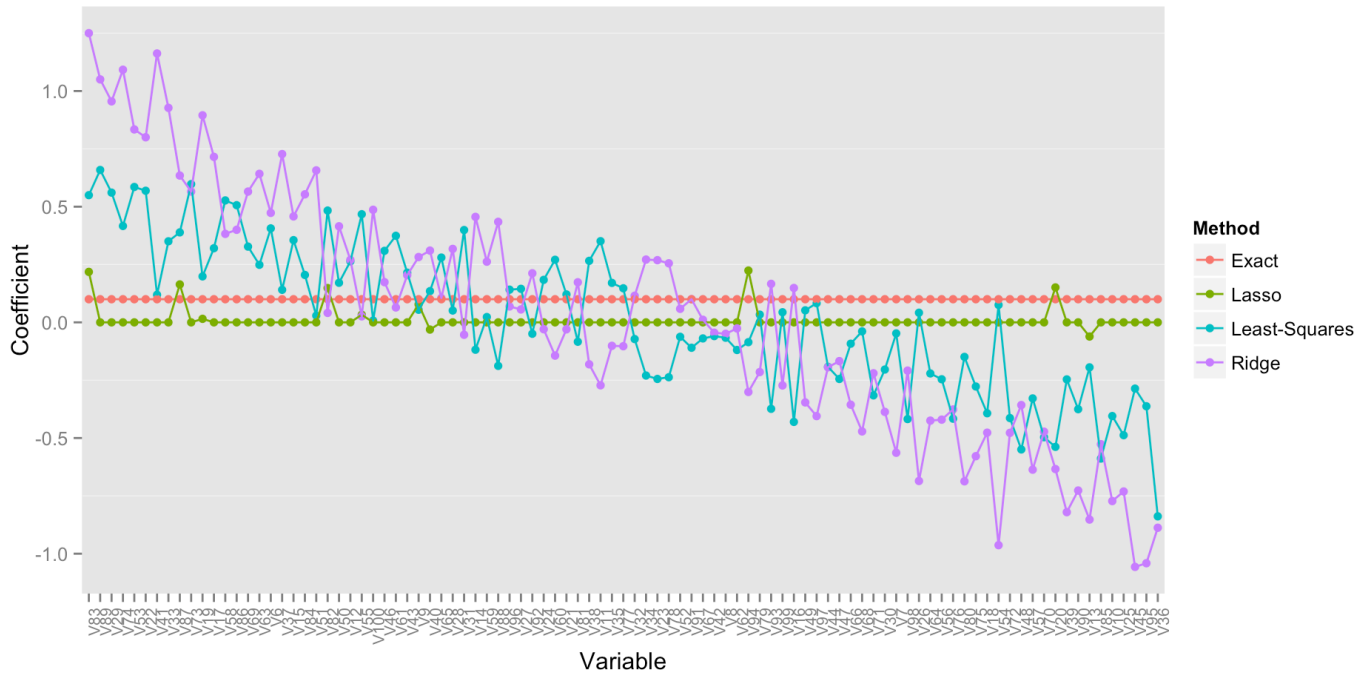

```
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Method'] <- "Ridge"
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Var'] <- t(t(rownames(sensitivities)))
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Value'] <- as.numeric(sensitivities)
rowStart <- rowStart + length(sensitivities)

#good values
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Method'] <- "Exact"
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Var'] <- t(t(rownames(sensitivities)))
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Value'] <- c(5,20,12,5*.9,2.5,1,rep(0.1,95))


sensDF$Exact = rep(c(5,20,12,5*.9,2.5,1,rep(0.1,95)),4)
```

Compare Methods

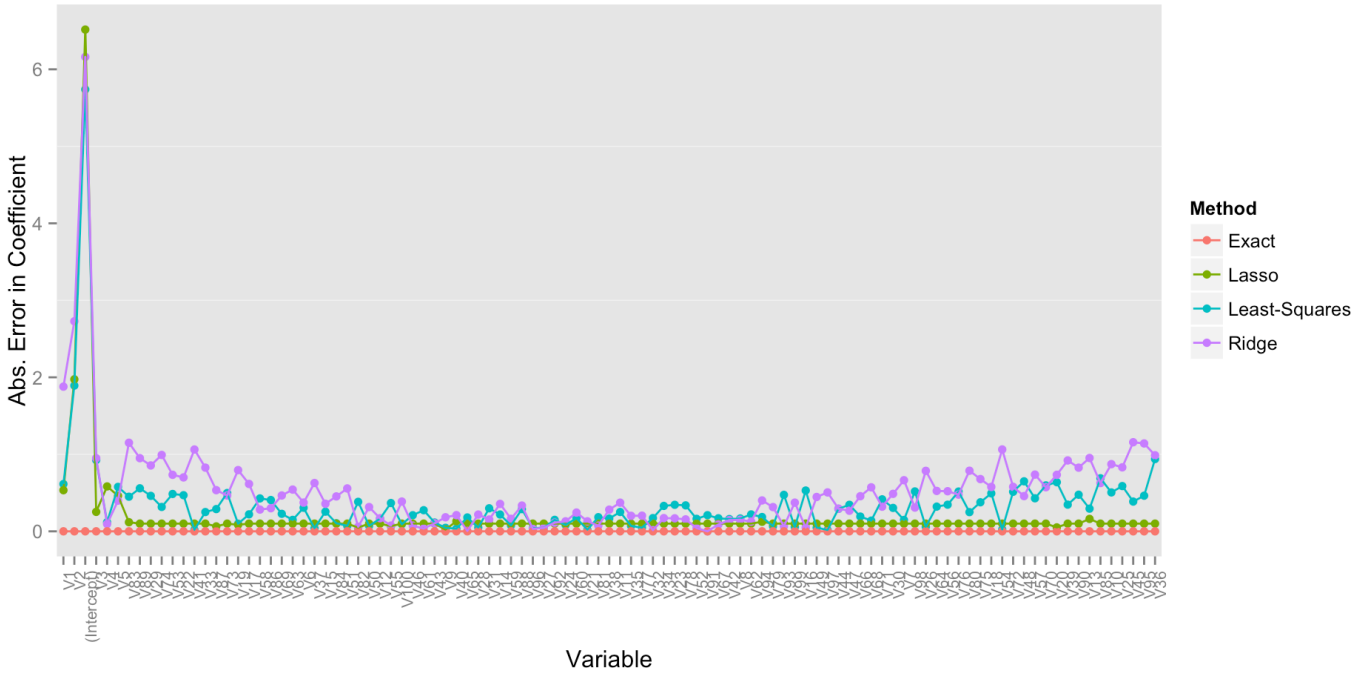
```
ggplot(sensDF,aes(x=reorder(Var,-Value),y=Value,color=Method,group=Method)) + geom_point() + geo
m_line() + theme(panel.grid.major = element_blank(),axis.text.x = element_text(angle = 90, hjust
 = 1, size=8))+ scale_x_discrete("Variable") + scale_y_continuous("Coefficient")
```




```
ggplot(sensDF[sensDF$Exact<1,],aes(x=reorder(Var,-Value),y=Value,color=Method,group=Method)) + g
eom_point() + geom_line() + theme(panel.grid.major = element_blank(),axis.text.x =
element_text(angle = 90, hjust = 1, size=8))+ scale_x_discrete("Variable") +
scale_y_continuous("Coefficient")
```




```
ggplot(sensDF,aes(x=reorder(Var,-Value),y=abs(Value-Exact),color=Method,group=Method)) + geom_po
int() + geom_line() + theme(panel.grid.major = element_blank(),axis.text.x = element_text(angle
= 90, hjust = 1, size=8)) + scale_x_discrete("Variable") + scale_y_continuous("Abs. Error in Coe
fficient")
```


Abs. Error in Coefficient
6
4
2
0
Variable
Method
Exact
Lasso
Least-Squares
Ridge