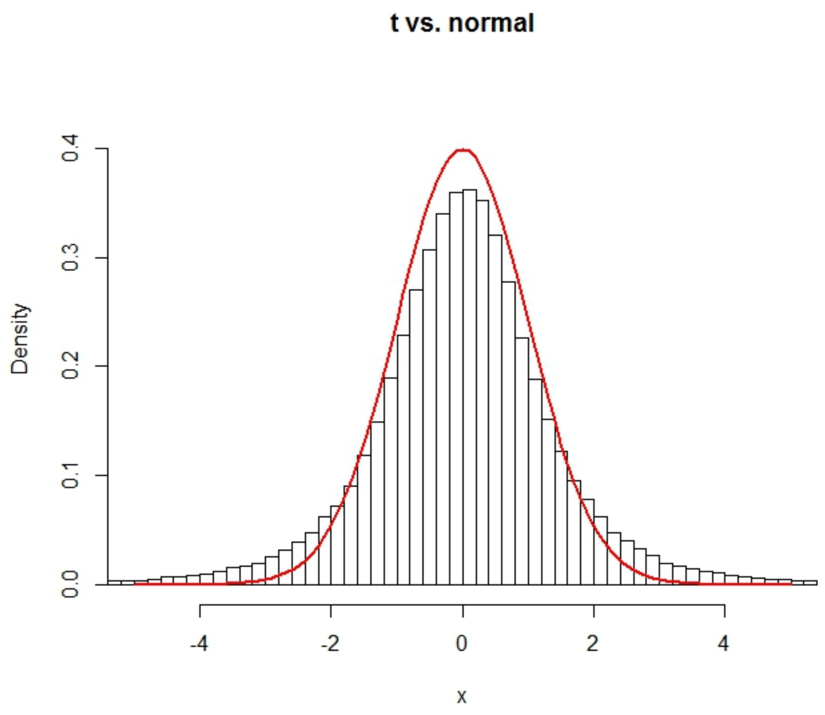
Introduction

We are asked to conduct Monte Carlo experiments to implement a simulation study with 5 different scenario. And the 5 scenario should be based on the multivariate t, not multivariate normal distribution. In each scenario, we obtain the estimated coefficients with 150 loops for each model, and construct some plots to visualize the results.

Data

We generated the data from multivariate t distribution, which is more flexible than multivariate normal distribution, especially in the tails which are heavier for t than normal distribution. The following plot used to illustrate the difference of the two distributions by using univariate case. The histogram of t distribution is with mean 0 and degree of freedom 3, and the red line is the standard normal distribution. We can see that the univariate t distribution have heavier tails than the normal, so do the multivariate case.



In the 5 scenarios, we have two different types of data, three have independent data and the other two have strong correlated data. So we created two different data generating functions for multivariate t distributions. The code for the functions showed below.

```
##### Generate independent data
genData<-function(n, p, beta){
x<- matrix(rmvt(n, delta=rep(0,length(beta)), sigma=diag(p), df=3 ),ncol=p,nrow=n)
y<- rnorm(n, x%*%beta, 2)
return(list(X=x,y=y))
}
#####Generate correlated data
genData2 <- function( n, p, beta, rho ){
s= c(1, rho^seq(1:(length(beta)-1)))
s1= toeplitz(s)
x<- matrix(rmvt(n, delta=rep(0,length(beta)), sigma=s1, df= 3), nrow=n, ncol=p)
y<- rnorm(n, x%*%beta, 2)
return(list(X=x,y=y))
}
```

Methods

0.1 Elastic Net with α and λ decided via cross validation

Elastic Net is a combination of ridge and lasso regression, so we obtain the coefficients by minimize

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}(\alpha\beta_j^2 + (1-\alpha)|\beta_j|)$$

The elastic net selects variables like the lasso, and shrinks together the coefficients of correlated predictors like ridge. It also has considerable computational advantages over the L_q penalties. In this problem, we are asked to use cross validation to choose α and the tuning parameter λ, so we do a for loop to run the cross validation process to get the minimum mean square error to obtain these two parameters. The code showed below.

```
enet<-function(Data)
{
require(glmnet)
k=10
#### Obtain fold for cross validation
fold<- sample(1:k, size=length(Data$y), replace=TRUE)
#### The alpha taht would be chosen
alpha = as.vector(seq(0,1, length=100))
best.lambda = vector(length=length(alpha))
#### The vector to store minimal MSE
min.mse = vector(length=length(alpha))
lowest.coef = list()
for(i in 1:length(alpha))
{
#### For each alpha choose lambda
cvout= cv.glmnet(Data$X, Data$y, foldid=fold,alpha=alpha[i],parallel=TRUE)
best.lambda[i] = cvout$lambda.min
lowest.coef[[i]]=as.numeric(coef(cvout,s=cvout$lambda.min))
min.mse[i] = cvout$cvm[which(cvout$lambda==cvout$lambda.min)]
}
index = which.min(min.mse)
return(lowest.coef[[index]])
}
```

0.2 Lad Lasso based on flare package

In this problem, we use **flare** package. The **flare** package implements a family of high dimensional methods (LAD Lasso, SQRT Lasso, l_q Lasso, and Dantzig Selector).
The least absolute deviation (LAD) regression is a useful method for robust regression. LAD Lasso can do parameter estimation and variable selection simultaneously and it is robust to heavy tail random noise and outliers compared with traditional Lasso. We have the closed form solution of LAD Lasso is

$$\alpha^{t+1} = S_{\frac{1}{n\rho\lambda}}(u^t + r - A\beta^t)$$

And we will have some discussion in later section to say why we choose the coefficient from the minimum λ. The code to implement Lad Lasso showed as fallows.

```
#### b) lad lasso based on flare package
ladlasso <-function(Data){
x1=Data$X
y1=Data$y
out1=slim(x1, y1,method="lq", q=1,nlambda =5,verbose=FALSE)
return(out1$beta0[[5]])
}
```

0.3 SQRT Lasso based on flare package

The SQRT Lasso is a method for estimating high-dimensional sparse linear regression models, where the overall number of regressors p is large, possibly much larger than n, but only s regressors are significant. The closed form solution is

$$\alpha^{t+1} = G_{\frac{1}{\sqrt{n}\rho\lambda}}(u^t + r - A\beta^t)$$

The code for this method showed in the next box, which is similar to LAD Lasso, only small changes.

```
#### c) SQRT lasso based on flare package
sqrtlasso <−function(Data)
{
x1=Data$X
y1=Data$y
out2=slim(x1, y1, method=”lq”, q=2, nlambda=5, verbose=FALSE,
      lambda.min.value=sqrt(log(ncol(x1))/nrow(x1)))
return(out2$beta0[[5]])
}
```

0.4 l_q Lasso based on flare package

l_q Lasso shares the advantage of LAD Lasso and SQRT Lasso. When the parameter $q = 1$ in l_q Lasso, it becomes Lad Lasso. And if $q = 2$, it is SQRT Lasso. The code for l_q Lasso listed in the follow box, usually $1<q<2$, we choose $q = 1.5$ in this case.

```
#### d) Lq lasso based on flare package
lqlasso <−function(Data)
{
x1=Data$X
y1=Data$y
out3=slim(x1, y1, method=”lq”, q=1.5, nlambda=5,verbose=FALSE)
return(out3$beta0[[5]])
}
```

0.5 Dantzig Lasso based on flare package

The Dantzig Lasso have many important statistical application, it can be applied to the cases where the number of variables or parameter p is much larger than the number of observations n. The selector can tolerate missing values in the design matrix and response vector. The closed form solution of this Lasso is

$$\alpha^{t+1} = W_\lambda(u^t + r - A\beta^t)$$

. The code is in following box.

```
#### e) Dantzig lasso based on flare package
dantziglasso <−function(Data)
{
x1=Data$X
y1=Data$y
out4=slim(x1, y1, method='dantzig', nlambda=5, verbose=FALSE)
return(out4$beta0[[5]])
}
```

0.6 Adaptive Lasso based on the least square approximation

Wang and Leng (2007, JASA) proposed a method of least squares approximation for unified yet simple Lasso estimation. The adaptive lasso can transfer many different type of Lasso objective function into their asymptotically equivalent least squares problems. So the resulting estimator can be as efficient as the oracle. We use the code as the link showed but a little change, due to it is too long, we won't put the functions here, only the main part.

```
#### f) Adaptive lasso based on the LS approximation
adaptivelasso <− function(Data)
{
x=Data$X
y1=Data$y
out5=lsa.linear(x=x,y=y1)
return(out5$coeff)
}
```

0.7 Adaptive lad lasso with 4 different penalty terms

The Adaptive Lad Lasso was introduced by Wang (2013), which investigate the effect of four different penalty terms, $\lambda_1 = \sqrt{1.5 * n * \log(p)}$, $\lambda_2 = \sqrt{2 * n * \log(p)}$, $\lambda_3 = \sqrt{4 * n * \log(p)}$, $\lambda_4 = \sqrt{10 * n * \log(p)}$. They are all fixed choices and do not depend on any assumption or parameters. Larger λ can cause more bias to the estimator. Usually λ_1 has the best estimators and λ_3 has the best selection properties. For comparison, we have four functions for the four penalty terms.

```
#### g) 4 variants of adaptive lad lasso
adapladlasso1<- function(Data)
{
#### penalty=sqrt(1.5*n*log(p))
require(quantreg)
QR1<- coef(rq(y ~ X, tau=0.5, data=Data, method="lasso", lambda=1 ))
lam= sqrt(1.5*nrow(Data$X)*log(ncol(Data$X)))/abs(QR1)
lam[1]=0
QR.coef1<- coef(rq(y~ X, tau=.5, data=Data, method="lasso", lambda=lam))
return(QR.coef1)
}
#### penalty=sqrt(2*n*log(p))
adapladlasso2<- function(Data)
{
require(quantreg)
QR1<- coef(rq(y ~ X, tau=0.5, data=Data, method="lasso", lambda=1 ))
lam= sqrt(2*nrow(Data$X)*log(ncol(Data$X)))/abs(QR1)
lam[1]=0
QR.coef2<- coef(rq(y~ X, tau=.5, data=Data, method="lasso", lambda=lam))
return(QR.coef2)
}
#### penalty=sqrt(4*n*log(p))
adapladlasso3<- function(Data)
{
require(quantreg)
QR1<- coef(rq(y ~ X, tau=0.5, data=Data, method="lasso", lambda=1 ))
lam= sqrt(4*nrow(Data$X)*log(ncol(Data$X)))/abs(QR1)
lam[1]=0
QR.coef3<- coef(rq(y~ X, tau=.5, data=Data, method="lasso", lambda=lam))
return(QR.coef3)
}
#### penalty=sqrt(10*n*log(p))
adapladlasso4<- function(Data){
require(quantreg)
QR1<- coef(rq(y ~ X, tau=0.5, data=Data, method="lasso", lambda=1 ))
lam= sqrt(10*nrow(Data$X)*log(ncol(Data$X)))/abs(QR1)
lam[1]=0
```

```
QR.coef4<- coef(rq(y~X, tau=.5, data=Data, method="lasso", lambda=lam))
return(QR.coef4)}
```

0.8 Conventional Lasso

We also use the conventional Lasso to illustrate the differences of the flare package and glmnet package. The code for the Conventional lasso is the same as the notes provided.

```
lasso <- function(Data)
{
require(glmnet)
cvfit <- cv.glmnet(Data$X,Data$y, parallel=TRUE)
return(as.numeric(coef(cvfit,s=cvfit$lambda.min)))
}
```

Results

We have five different scenarios to carry out the different methods. In each simulation, we have n observations and p predictors. In each scenario, we repeat the simulation N=150 times. After the simulations, we calculate the bias, variance and mean square error for each method.

Scenario 1: independent predictors with 16 zeros

In the first scenario, we have 20 independent predictors, and 16 predictors have coefficients zeros, in this case, $p = 20$.

```
n=100
N=150
beta=c(2, −2, 1, −1,rep(0,16))
p=length(beta)
#### results used to store the coefficients for each loop and with names
results = array(NA, dim=c(N, p, 12),dimnames =list(1:N, 1:p,
          c("Elastic Net", "Lad Lasso", "Sqrt Lasso", "lq Lasso", "Dantzig Selector",
        "Adaptive Lasso", "Ad.Lad.Lasso 1", "Ad.Lad.Lasso 2", "Ad.Lad.Lasso 3",
        "Ad.Lad.Lasso 4", "OLS","Lasso")))
#### The names for box plots and heat maps.(Scenarios 1 to 3 share the same names)
dimnames =c("Elastic Net", "Lad Lasso", "Sqrt Lasso", "lq Lasso",
       "Dantzig Selector", "Adaptive Lasso", "Ad.Lad.Lasso 1",
       "Ad.Lad.Lasso 2", "Ad.Lad.Lasso 3", "Ad.Lad.Lasso 4", "OLS")
#### Carrying out the methods in 150 loops.
for(i in 1:N){
Data = genData(n, p, beta)
results[i,,1]<− enet(Data)[−1]
results[i,,2]<− ladlasso(Data)
results[i,,3]<− sqrtlasso(Data)
print(i) ####To trace the loop
results[i,,4]<− lqlasso(Data)
results[i,,5]<− dantziglasso(Data)
results[i,,6]<− adaptivelasso(Data)
results[i,,7]<− adapladlasso1(Data)[−1]
results[i,,8]<− adapladlasso2(Data)[−1]
results[i,,9]<− adapladlasso3(Data)[−1]
results[i,,10]<− adapladlasso4(Data)[−1]
results[i,,11]<− coef(lm(y˜X,Data))[−1]
results[i,,12]<− lasso(Data)[−1]
}
#### Obtain the bias variance and MSE for scenario 1.
B1<− apply(results, 2:3, mean)−beta
V1<− apply(results, 2:3, var)
MSE1<− B1ˆ2+V1
s1 = apply(MSE2, 2, sum)
```

Under this scenario, we obtained the MSE for the 12 methods. They are showed in the Table 1.

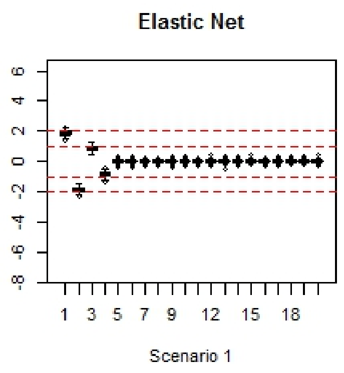
Table 1: The MSE for Scenario 1

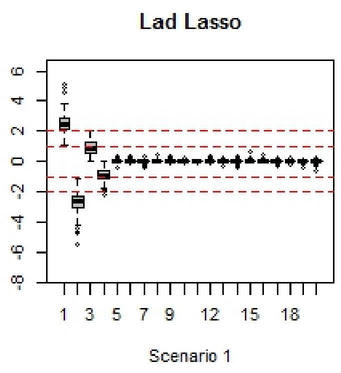
Elastic Net	**Lad Lasso**	**SQRT Lasso**	l_q **Lasso**	**Dantzig**	**Ad.Lasso**
0.286977	2.214455	3.155539	2.854081	2.371748	0.140569
Ad.La.Lasso1	**Ad.La.Lasso2**	**Ad.La.Lasso3**	**Ad.La.Lasso4**	**OLS**	**Lasso**
0.471701	0.588352	1.079873	2.462427	0.592076	0.282087

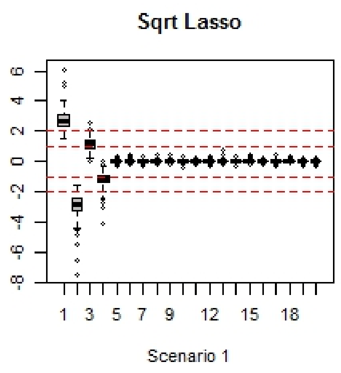
From the table, we can see that the adaptive lasso have the smallest mean square error, so it is small for each loop.And for adaptive lad lasso, with the increase of the penalty term, the mean square error increases, too, which means large penalty term may kill significant variable. Besides, the Lasso of flare package don't do well than the conventional Lasso and ordinary least square. But the conventional lasso does better than the least square, which is the same as what we have in the class.

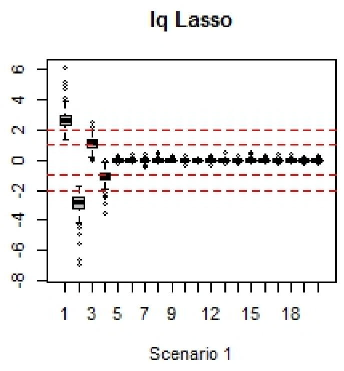
For visualization, we obtain the box plot for the coefficients, and heat map for the bias and variance. The code is in the following box.In other scenarios, we have almost the same code for the plots, so the code for the later scenarios will not be presented.

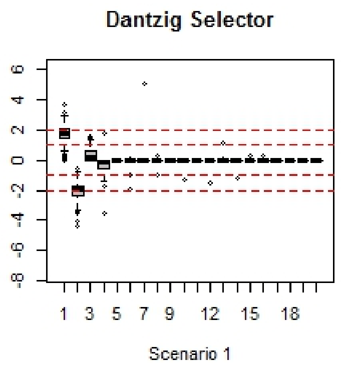
```
#### Draw the heat map for the bias and variance
pheatmap(B1, cluster_rows=F, cluster_cols=F, cellwidth =30, cellheight = 15,
          labels_row = c(1:20),labels_col =dimnamesh, main="Bias Scenario 1")
pheatmap(V1, cluster_rows=F, cluster_cols=F, cellwidth =30, cellheight = 15,
          labels_row = c(1:20),labels_col =dimnamesh, main="Variance Scenario 1")
#### Save the MSE in my drive
write.csv(s1,row.names=dimnames,file="C:/Users/Lanlan/Downloads/STP 598
    Computational Statistics/hwk/HWK4/s1.csv")

ylim=range(results)
par(mfrow=c(2:3))
for(i in 1:12)
{
boxplot(results[,,i], col="gray", ylim=ylim, main=dimnames[i], xlab="Scenario 1")
abline(h=c(2,-2,1,-1),lty=2, col="red")
}
```

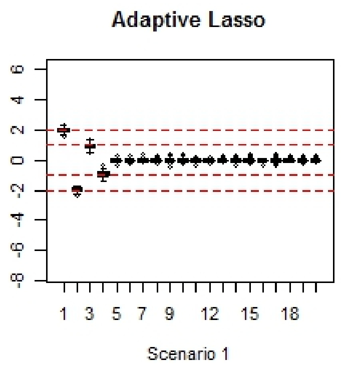


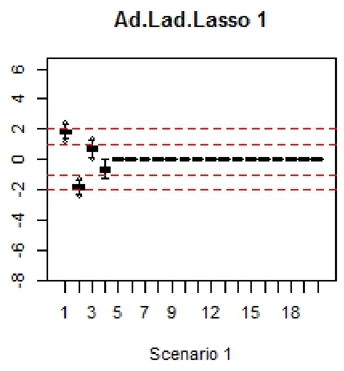


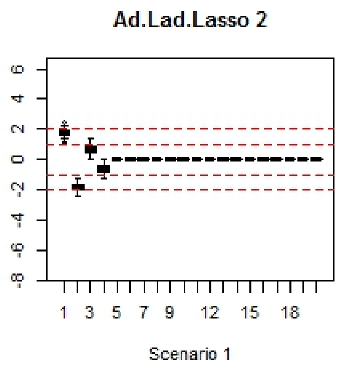


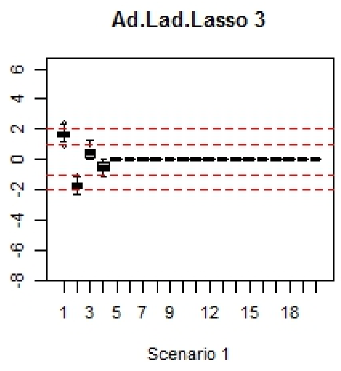


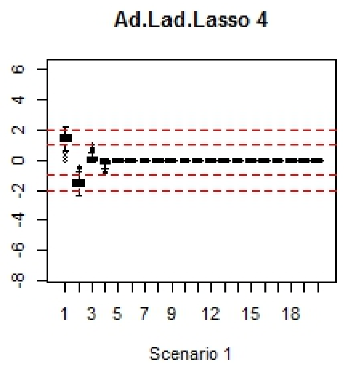


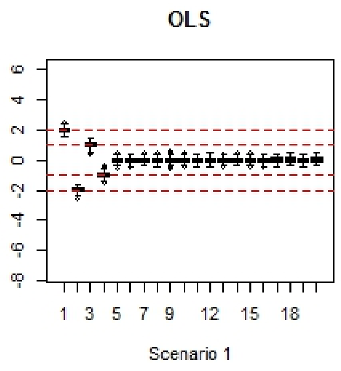


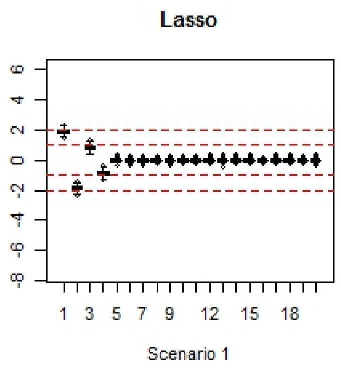




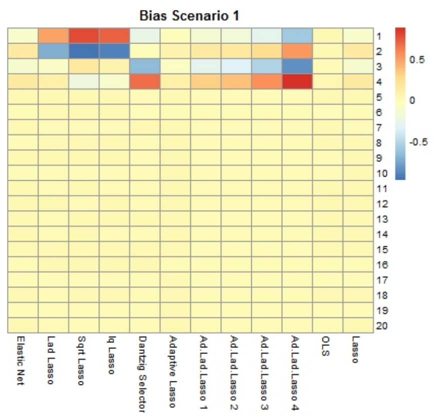


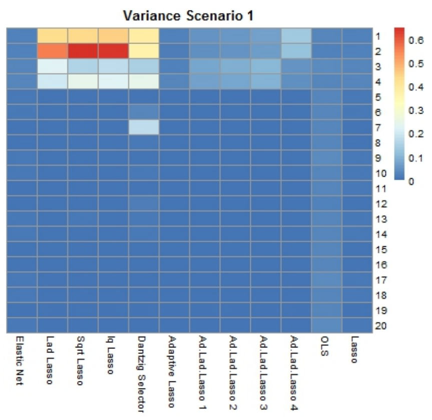






From the above plots, we can see that the coefficients in lad lasso, SQRT lasso and l_q are a bit overfitting, and Elastic Net with α and λ chosen by cross validation and Adaptive Lasso performs significantly better than the Lasso by using the flare package. Besides, the ordinary least square also performs well than the Lasso in flare package.





From the heat map for bias in scenario 1, the least square and adaptive lasso based on least square approximation have nearly zero bias, and the square lasso and l_q lasso have large bias for the first and second betas. And all the methods shrink the estimate to zero when the true beta is zero.
In the above map for variance, the Lad Lasso, SQRT Lasso, l_q Lasso and Dantzig Lasso have high variance for the first four betas compared with other methods. So the MSE's are larger i Lad Lasso, SQRT Lasso, l_q Lasso and Dantzig Lasso with respect with the bias and Variance, and this is in accordance with Table 1 we have before.

Scenario 2: independent predictors with no zeros

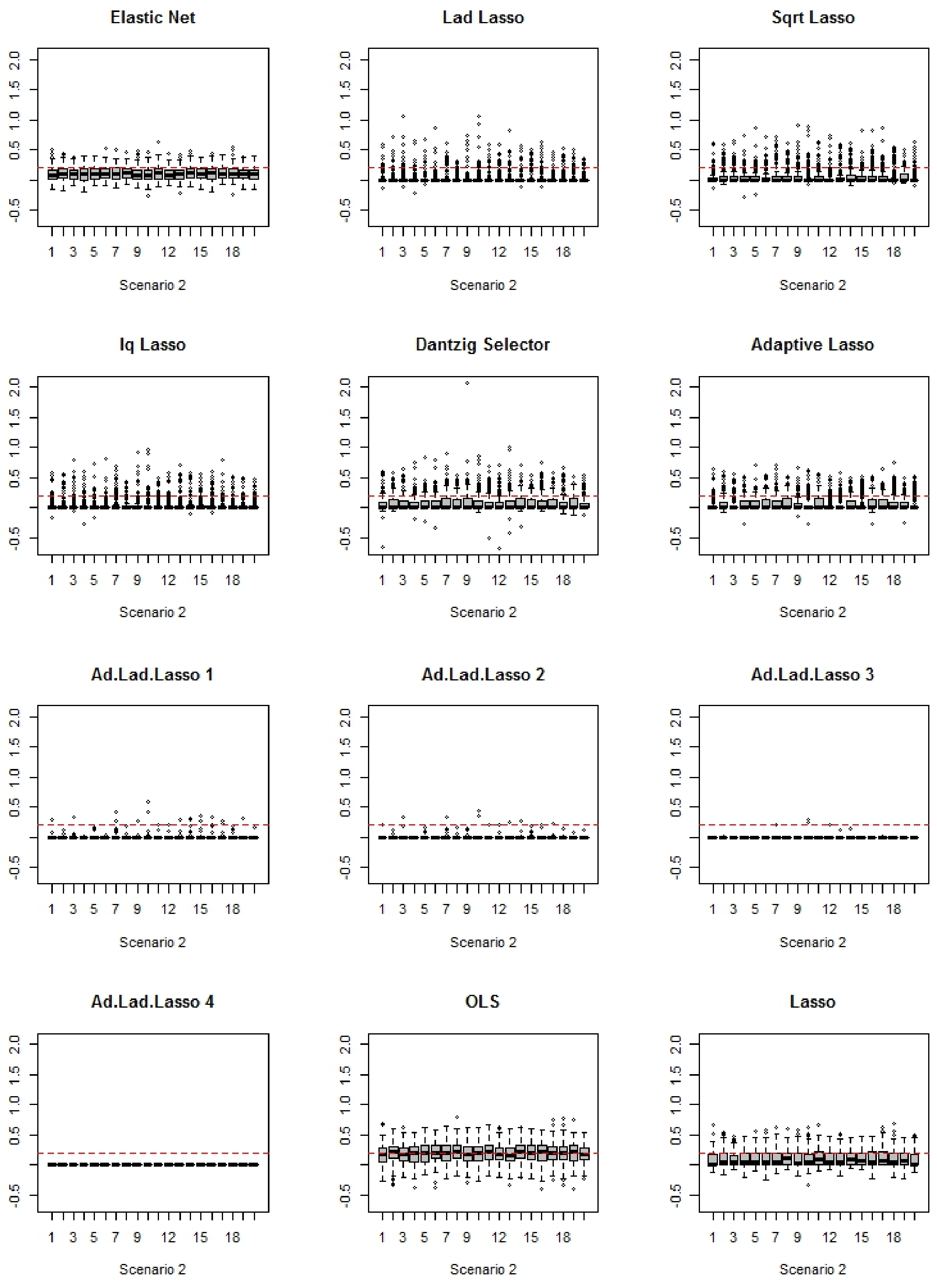
Under Scenario 2, we have 20 independent predictors all with coefficients 0.2. Then we do simulation 150 times and obtain the bias, variance and MSE. The code for this scenario is in the following box.

```
#### Scenario 2.
n=100
N=150
#### The betas are different from scenario 1.
beta2=rep(0.2,20)
p=length(beta2)
#### results used to store the coefficients for each loop and with names
results = array(NA, dim=c(N, p, 12),dimnames =list(1:N, 1:p,
    c("Elastic Net", "Lad Lasso", "Sqrt Lasso", "lq Lasso", "Dantzig Selector",
    "Adaptive Lasso", "Ad.Lad.Lasso 1", "Ad.Lad.Lasso 2", "Ad.Lad.Lasso 3",
    "Ad.Lad.Lasso 4", "OLS","Lasso")))
#### Do the simulation
for(i in 1:N)
{
Data = genData(n, p, beta2)
results2[i,,1]<− enet(Data)[−1]
results2[i,,2]<− ladlasso(Data)
results2[i,,3]<− sqrtlasso(Data)
print(i)
results2[i,,4]<− lqlasso(Data)
results2[i,,5]<− dantziglasso(Data)
results2[i,,6]<− adaptivelasso(Data)
results2[i,,7]<− adapladlasso1(Data)[−1]
results2[i,,8]<− adapladlasso2(Data)[−1]
results2[i,,9]<− adapladlasso3(Data)[−1]
results2[i,,10]<− adapladlasso4(Data)[−1]
results2[i,,11]<− coef(lm(y˜X,Data))[−1]
results2[i,,12]<− lasso(Data)[−1]
}
#### Get the statistics for the simulatio
B2<− apply(results2, 2:3, mean)−beta2
V2<− apply(results2, 2:3, var)
MSE2<− B2ˆ2+V2
s2 = apply(MSE2, 2, sum)
```

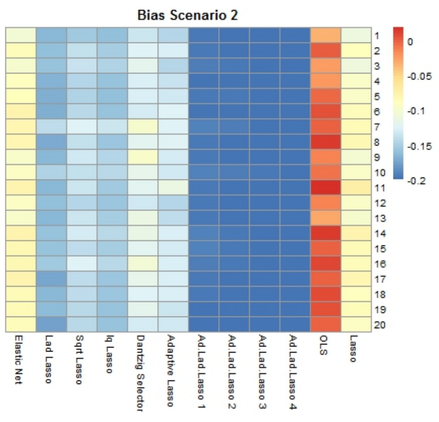
Under this scenario, we obtained the MSE for the 12 methods in Table 2. From the table, we can see that elastic net performs best, the next best is conventional lasso. And least square does better than the lasso except the elastic net and conventional lasso. But all the MSE are relatively small.

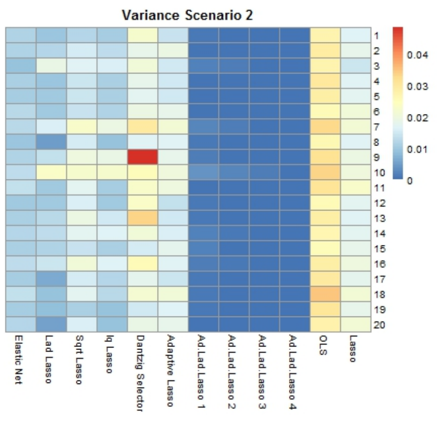
Table 2: The MSE for Scenario 2

Elastic Net	**Lad Lasso**	**SQRT Lasso**	**l_q Lasso**	**Dantzig**	**Ad.Lasso**
0.403121	0.752596	0.770536	0.771891	0.789228	0.675153
Ad.La.Lasso1	**Ad.La.Lasso2**	**Ad.La.Lasso3**	**Ad.La.Lasso4**	**OLS**	**Lasso**
0.790491	0.79469	0.798615	0.799819	0.58539	0.542296



The above are box plots for coefficients in scenario 2. We know that the true value is 0.2 in this scenario and elastic net, least square, and conventional lasso have a better estimation. And with lad lasso, l_q lasso, adaptive lasso with penalty 2, 3 and 4, the coefficients are shrinking to zero, because the heavy penalty kill the significant variable.





From the heat map of bias, we see the least square is unbiased and the Adaptive Lad Lasso with four different penalty terms have largest bias, maybe because the Adaptive Lasso have big penalty terms to kill the variables. In heat map of variance, the Adaptive Lad Lasso have smallest variance and least square have largest variance.

Scenario 3: strongly correlated predictors

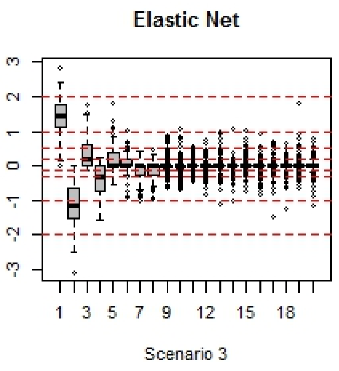
Under scenario 3, we are going to deal with strongly correlated data, where the correlation is 0.9. With correlated data, we have different data generating function, the code showed in the data section. The code for this scenario showed below.

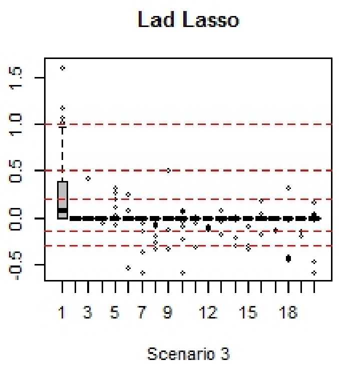
```
n=100
N=150
beta3=c(2, −2, 1, −1, 0.5, 0.2, −0.3, −0.15, rep(0,12))
p=length(beta3)
#### rho3 is the correlation for correlated data.
rho3=0.9
#### results used to store the coefficients for each loop and with names
results = array(NA, dim=c(N, p, 12),dimnames =list(1:N, 1:p,
        c(”Elastic Net”, ”Lad Lasso”, ”Sqrt Lasso”, ”lq Lasso”, ”Dantzig Selector”,
       ”Adaptive Lasso”, ”Ad.Lad.Lasso 1”, ”Ad.Lad.Lasso 2”, ”Ad.Lad.Lasso 3”,
       ”Ad.Lad.Lasso 4”, ”OLS”,”Lasso”)))

for(i in 1:N)
{
Data = genData2(n, p, beta3, rho3)
results3[i,,1]<− enet(Data)[−1]
results3[i,,2]<− ladlasso(Data)
results3[i,,3]<− sqrtlasso(Data)
print(i)
results3[i,,4]<− lqlasso(Data)
results3[i,,5]<− dantziglasso(Data)
results3[i,,6]<− adaptivelasso(Data)
results3[i,,7]<− adapladlasso1(Data)[−1]
results3[i,,8]<− adapladlasso2(Data)[−1]
results3[i,,9]<− adapladlasso3(Data)[−1]
results3[i,,10]<− adapladlasso4(Data)[−1]
results3[i,,11]<− coef(lm(y˜X,Data))[−1]
results3[i,,12]<− lasso(Data)[−1]
}

B3<− apply(results3, 2:3, mean)−beta3
V3<− apply(results3, 2:3, var)
MSE3<− B3ˆ2+V3
s3 = apply(MSE3, 2, sum)
```

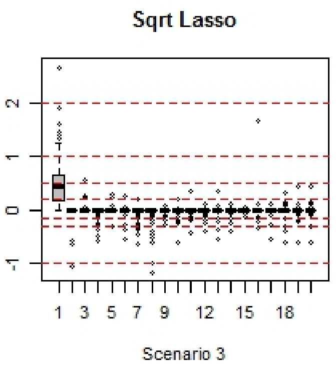
Under scenario 3, I obtained the MSE for the 12 methods in Table 3. From the table, we can see that adaptive lasso with least square approximation performs best, next is the elastic net then is least square. But all the MSE's are larger than the independent case, because the correlated data have greater variance than independent data. So do the MSE for these methods.

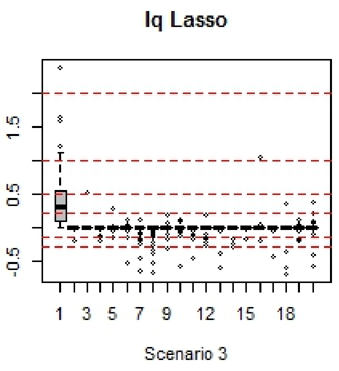
Table 3: The MSE for Scenario 3

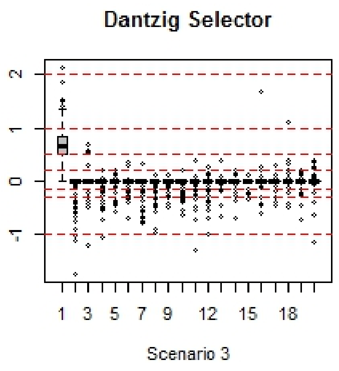
Elastic Net	**Lad Lasso**	**SQRT Lasso**	l_q **Lasso**	**Dantzig**	**Ad.Lasso**
4.039673	9.614989	8.883889	9.228446	8.143745	3.351744
Ad.La.Lasso1	**Ad.La.Lasso2**	**Ad.La.Lasso3**	**Ad.La.Lasso4**	**OLS**	**Lasso**
7.539734	8.186295	9.468256	9.930285	5.280785	3.897252

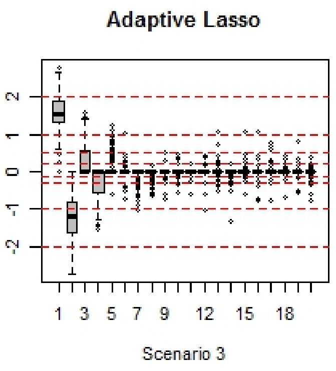


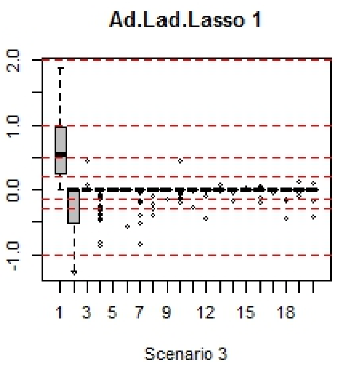


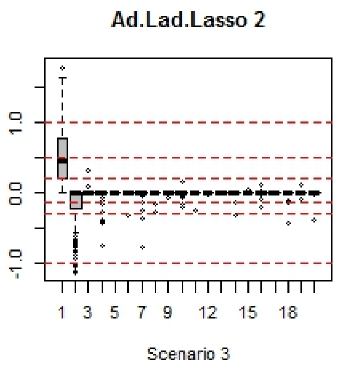


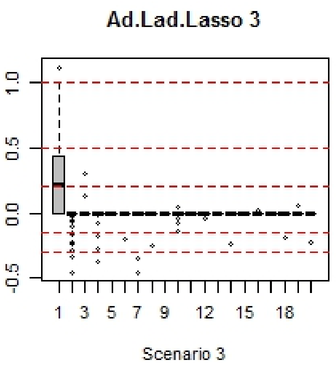


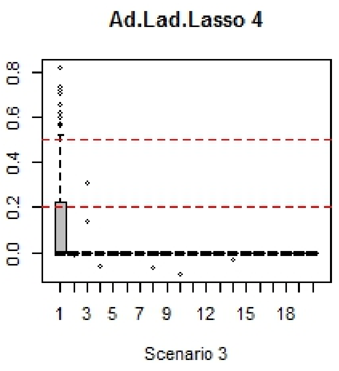


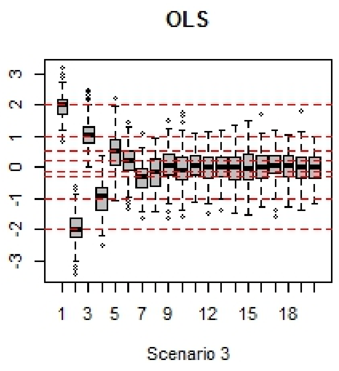


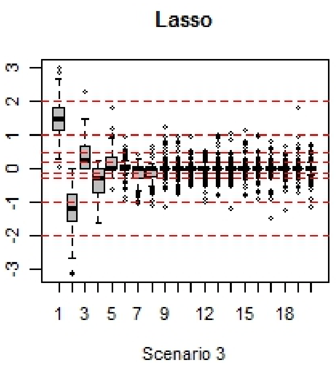




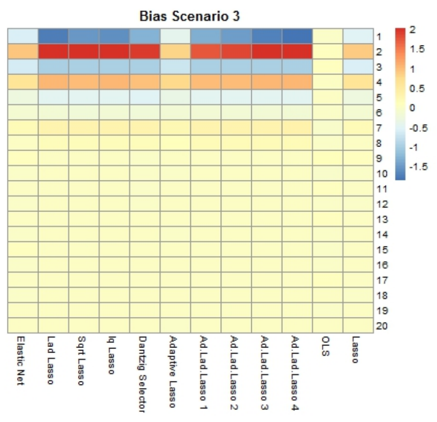


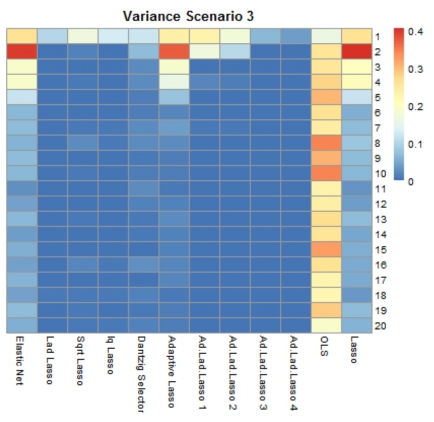






From the box plots, we see that the first eight coefficients are far away from the true values in all methods, and we get stronger variance than independent case. The lad lasso, sqrt lasso, l_q lasso, adaptive lad lasso with penalty 3 and 4 all the coefficients except the first one are shrinking to zero, which cause the variance big.





In bias heat map, least square gets almost unbiased estimator, and when the betas are not zero, all other methods obtain biased coefficients, but the absolute biases are decreasing as the absolute values of the coefficients decrease. Also from heat map of variance, the least square have biggest variance. The other methods only have variance for the first and second cases.

Scenario 4: high dimensional setting

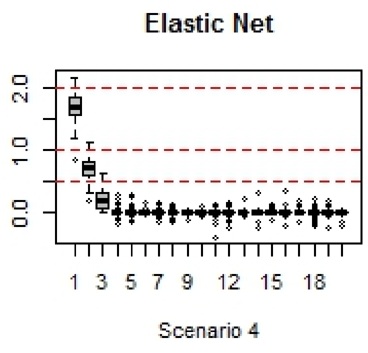
In scenario 4, we are doing to deal with high dimensional data set, where the number of predictors p are greater than the observations n. In this case, we find the least square cannot get unique solution, so we can’t get the results from the ordinary least square. We also can’t find solutions for the adaptive Lasso based on the least square approximation. So we only need to compare 10 methods here. To decrease the time for running the code, I only have 220 betas here. The code are provided in the following box.

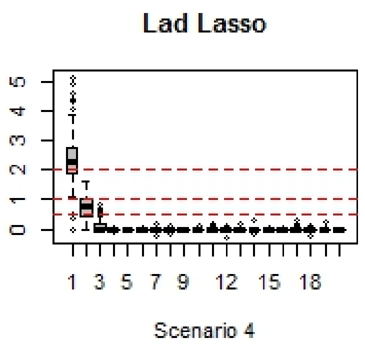
```
n=100
N=150
#### High dimensional case.
beta4=c(2, 1, 0.5, rep(0, 217))
p=length(beta4)
results4 = array(NA, dim=c(N, p, 10),dimnames = list(1:N,1:p,
      c(”Elastic Net”, ”Lad Lasso”, ”Sqrt Lasso”, ”lq Lasso”,
      ”Dantzig Selector”, ”Ad.Lad.Lasso 1”, ”Ad.Lad.Lasso 2”, ”Ad.Lad.Lasso 3”,
      ”Ad.Lad.Lasso 4”,”Lasso”)
dimnames2 =c(”Elastic Net”, ”Lad Lasso”, ”Sqrt Lasso”, ”lq Lasso”, ”Dantzig
            Selector”, ”Ad.Lad.Lasso 1”, ”Ad.Lad.Lasso 2”, ”Ad.Lad.Lasso 3”,
        "Ad.Lad.Lasso 4","Lasso")

for(i in 1:N)
{
Data = genData(n, p, beta4)
results4[i,,1]<− enet(Data)[−1]
results4[i,,2]<− ladlasso(Data)
results4[i,,3]<− sqrtlasso(Data)
print(i)
results4[i,,4]<− lqlasso(Data)
results4[i,,5]<− dantziglasso(Data)
results4[i,,6]<− adapladlasso1(Data)[−1]
results4[i,,7]<− adapladlasso2(Data)[−1]
results4[i,,8]<− adapladlasso3(Data)[−1]
results4[i,,9]<− adapladlasso4(Data)[−1]
results4[i,,10]<− lasso(Data)[−1]

}
B4<− apply(results4, 2:3, mean)−beta4
V4<− apply(results4, 2:3, var)
MSE4<− B4ˆ2+V4
s4 = apply(MSE4, 2, sum)
```

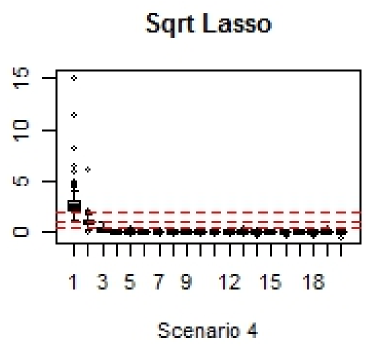
Under scenario 4, we obtained the MSE for 10 methods in Table 4. Because we can't get the unique solutions for least square and least square approximation. From the table, we can see that elastic net performs best, next best is regular lasso then adaptive lad lasso. But the SQRT and l_q Lasso have relative large MSE. In this case, the adaptive lad lasso with small penalty terms may work better in high dimensional case when compared with other cases.

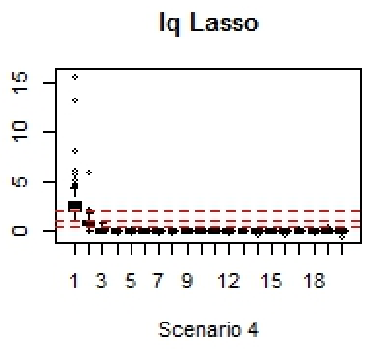
Table 4: The MSE for Scenario 4

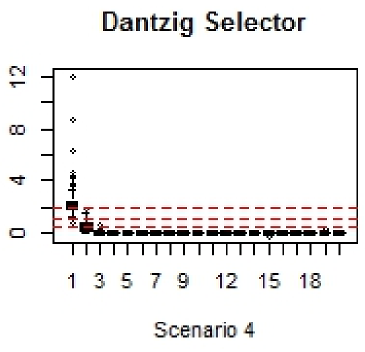
Elastic Net	**Lad Lasso**	**SQRT Lasso**	l_q **Lasso**	**Dantzig**
0.537846	1.267688	3.759697	3.933184	2.244374
Ad.La.Lasso1	**Ad.La.Lasso2**	**Ad.La.Lasso3**	**Ad.La.Lasso4**	**Lasso**
0.767141	0.896543	1.303894	2.119386	0.546357

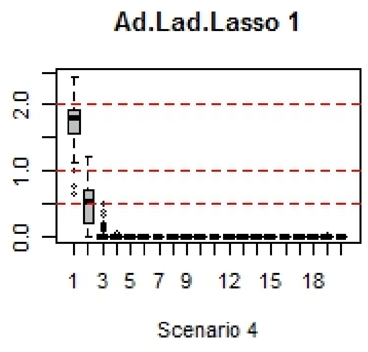


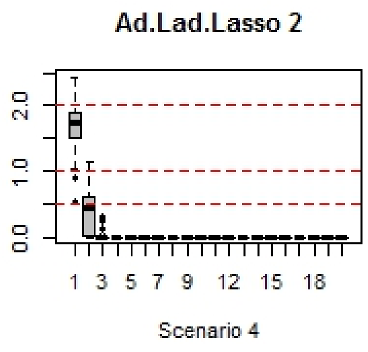


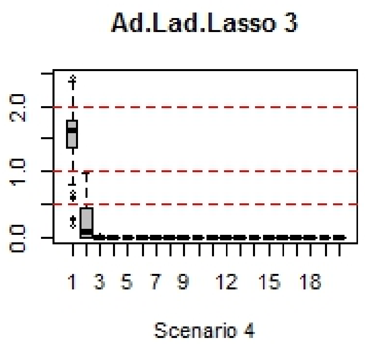


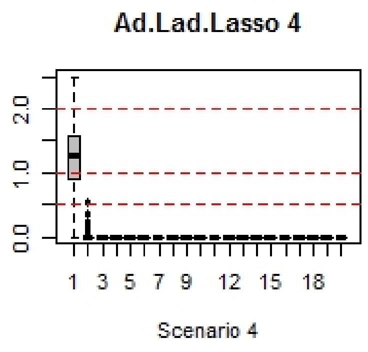


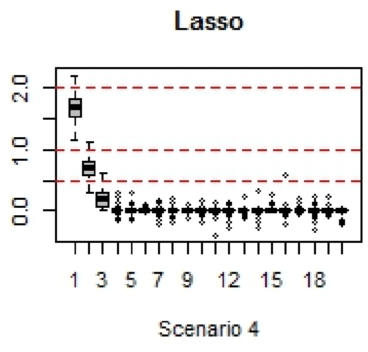




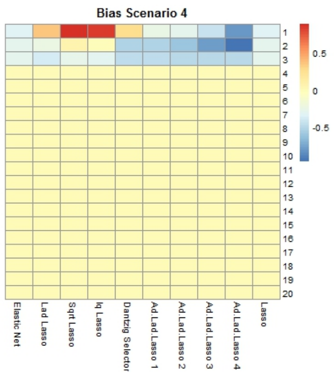


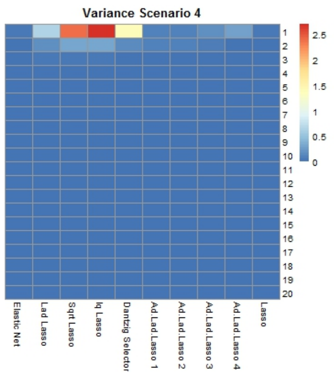






In the box plot I only use the first 20 coefficients to make the plots look better. From the box plots, we see that there are some large valued outliers in the SQRT Lasso, l_q Lasso, and Dantzig selector, which may be the reason why the MSE for these methods are much larger than other methods.





In bias heat map, the SQRT Lasso and l_q Lasso have highest bias, and only the first three coefficients are biased. Also from heat map of variance, the SQRT Lasso, l_q Lasso and Dantzig selector have biggest variance.

Scenario 5: correlated predictors in high dimensional setting

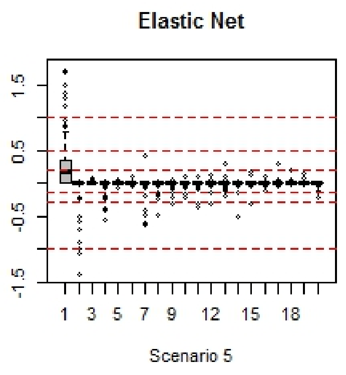
In this scenario, we are going to deal with high dimensional setting and strongly correlated data. The code are showed in the following box.

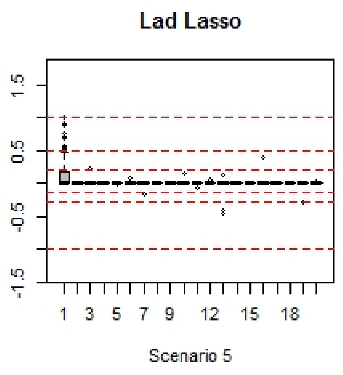
```
n=100
N=150
beta5=c(2, −2, 1, −1, 0.5, 0.2, −0.3, −0.15, rep(0,212))
p=length(beta5)
rho5=0.9
results5 = array(NA, dim=c(N, p, 10), dimnames = list(1:N,1:p,
            c("Elastic Net", "Lad Lasso", "Sqrt Lasso", "lq Lasso", "Dantzig Selector",
            "Ad.Lad.Lasso 1", "Ad.Lad.Lasso 2", "Ad.Lad.Lasso 3",
            "Ad.Lad.Lasso 4","Lasso")))

for(i in 1:N)
{
Data = genData2(n, p, beta5, rho5)
results5[i,,1]<− enet(Data)[−1]
results5[i,,2]<− ladlasso(Data)
results5[i,,3]<− sqrtlasso(Data)
print(i)
results5[i,,4]<− lqlasso(Data)
results5[i,,5]<− dantziglasso(Data)
results5[i,,6]<− adapladlasso1(Data)[−1]
results5[i,,7]<− adapladlasso2(Data)[−1]
results5[i,,8]<− adapladlasso3(Data)[−1]
results5[i,,9]<− adapladlasso4(Data)[−1]
results5[i,,10]<− lasso(Data)[−1]
}

B5<− apply(results5, 2:3, mean)−beta5
V5<− apply(results5, 2:3, var)
MSE5<− B5ˆ2+V5
s5 = round(apply(MSE5, 2, sum))
```

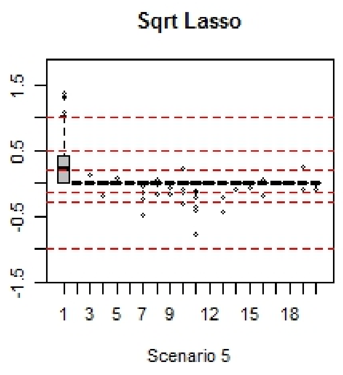
Table 5: The MSE for Scenario 5

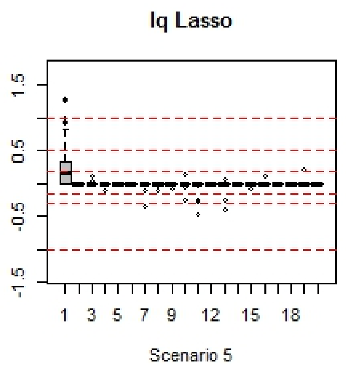
Elastic Net	**Lad Lasso**	**SQRT Lasso**	l_q **Lasso**	**Dantzig**
9.497539	9.978904	9.560686	9.674235	9.418479
Ad.La.Lasso1	**Ad.La.Lasso2**	**Ad.La.Lasso3**	**Ad.La.Lasso4**	**Lasso**
9.642956	9.748485	10.05788	10.28157	9.333704

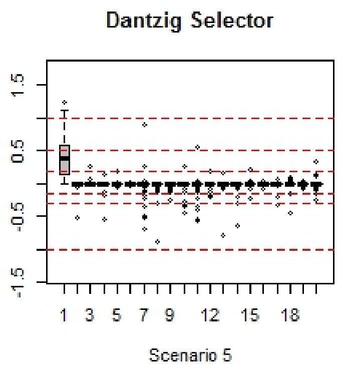
From the table, we see that the MSE's are much bigger than the independent case. And the conventional lasso has smallest MSE, the next best is elastic net, the next is Dantzig Lasso. But the adaptive lad lasso have largest value, maybe it has too much penalty.

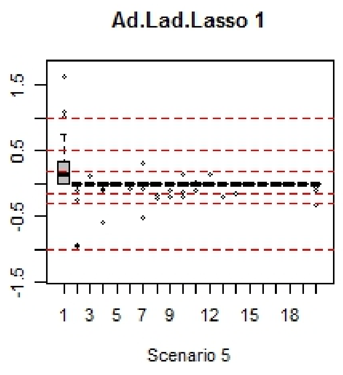


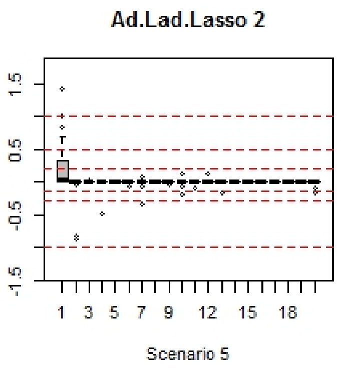


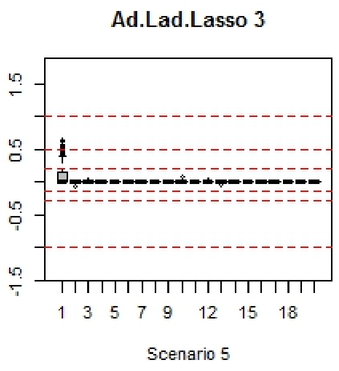


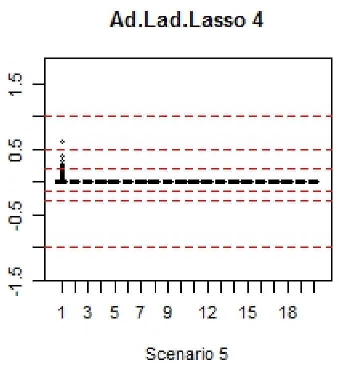


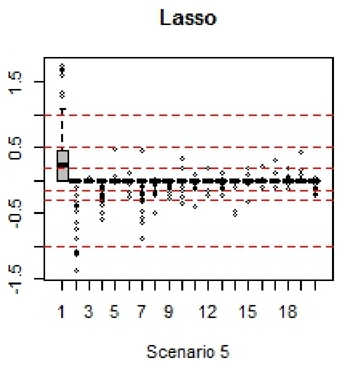




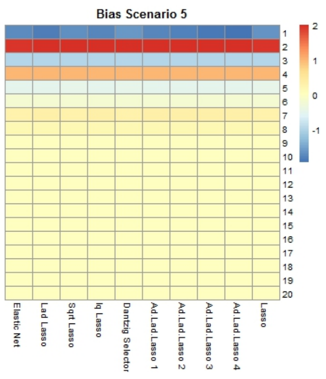


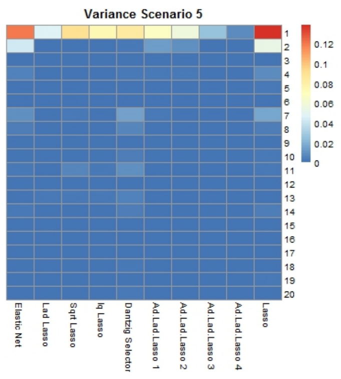






In the box plot I only get the first 20 coefficients to make the plots look better. From the box plots, we see that all the coefficients except the first one are shrinking to zero, which make the MSE big in the correlated data set.





In bias heat map, all the methods have almost the same trend of bias. The first beta have heaviest negative bias and second have positive bias. Also from heat map of variance, the first beta have large variance and other beta have almost zero variance.

Discussion

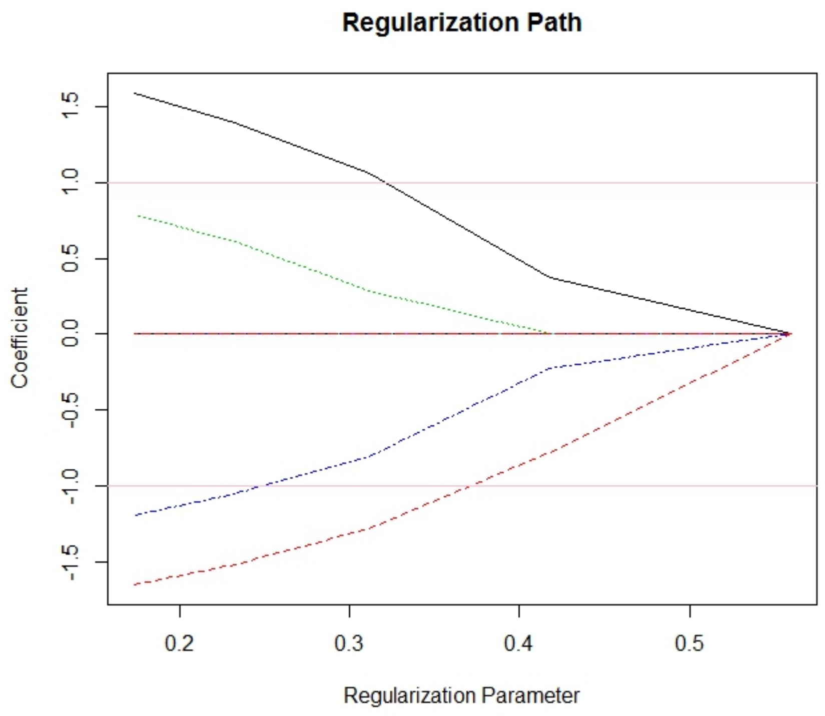
Why choose coefficient with smallest λ

In the flare package paper, written by Xingguo Li, etc., we see that the coefficients are solved by iteration with a sequence of λ by descending order. So we manually choose its minimum regularization parameter to get the coefficients in the flare package, which is the same idea in the GLMNET package. Or we can get the same conclusion from data analysis.

We used the beta in scenario 1 with SQRT Lasso to specify why we choose the last lambda in our code. From the results of first ten betas with five λ, we see that the last λ, gives us the best results that are close the true values, and with other λ, some betas are shrinking toward zero, thus we choose the last λ and have the smallest mean square error.

```
> coef(out2,lambda.idx = (1:5), beta.idx = (1:10))
Values of estimated coefficients:
index              1          2          3          4          5
lambda        0.5598     0.4174     0.3113     0.2321     0.1731
intercept      2.143      1.506     0.7893     0.4263     0.2238
beta 1             0     0.3721      1.056      1.395      1.585
beta 2    -4.835e-18    -0.7791      -1.28     -1.515     -1.645
beta 3             0          0      0.282     0.6063     0.7872
beta 4             0    -0.2243    -0.8017     -1.051     -1.191
beta 5             0          0          0          0          0
beta 6             0          0          0          0          0
beta 7             0          0          0          0          0
beta 8             0          0          0          0          0
beta 9             0          0          0          0          0
beta 10            0          0          0          0          0
```

We also use the plot of coefficients versus lambda to illustrate our conclusions. In the plot, the horizontal line is the true value of the parameter(2 and -2 outside the scope of y), and we see that the coefficients are all shrinking to zero as the increasing of the λ, which means the coefficients are going away from the true value, we can also see this trend in last box, when $\lambda = 0.5598$, all the coefficients are almost zero. So from this two aspects, we see that we choose the minimum λ to get the coefficients. We also present other cases and have the same conclusions, but we don't show all the cases.



<u>Results Discussion</u>

For a better understanding, I drawn a table for the overall results, which contains MSE's for all the scenarios and the methods. NA stands for the methods are not available under that scenario.

Table 6: Overall results

Methods	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
Elastic Net	0.28698	0.40312	4.03967	0.53784	9.49754
Lad Lasso	2.21445	0.75260	9.61499	1.26769	9.9789
Sqrt Lasso	3.15554	0.77054	8.88389	3.7597	9.56069
lq Lasso	2.854081	0.77189	9.22845	3.93318	9.67423
Dantzig Selector	2.37175	0.78923	8.14374	2.24437	9.41848
Adaptive Lasso	0.14057	0.67515	3.35174	NA	NA
Ad.Lad.Lasso 1	0.4717	0.79049	7.53973	0.76714	9.64296
Ad.Lad.Lasso 2	0.58835	0.79469	8.1863	0.89654	9.74849
Ad.Lad.Lasso 3	1.07987	0.79861	9.46826	1.30389	10.05788
Ad.Lad.Lasso 4	2.46243	0.799829	9.93029	2.11939	10.28157
OLS	0.59208	0.58539	5.28078	NA	NA
Lasso	0.28209	0.54229	3.89725	0.546357	9.3337

It is clear that there is no methods is absolutely better than the others in all situations in this case. But our results can be summarized that:

- The elastic net with α and λ chosen by cross validation performs better than the other methods in most of the cases.
- In independent case, the lad lasso performs better than SQRT lasso, and SQRT does better than l_q lasso. But for correlated data, SQRT has smaller MSE than l_q lasso, and l_q lasso has smaller MSE than lad lasso.
- The lasso in the flare package (Lad Lasso, SQRT Lasso, l_q Lasso and Dantzig Selector) don't do well than the conventional lasso in all the scenario, but the lasso in flare package are more robust to outliers or heavy tails. Maybe the data in our code are not good enough to show the advantage of the lasso in flare package. Or we need to cross validation to choose the tuning parameter λ to have a better solutions.
- The Adaptive Lasso from least square approximation does well in scenario 1 and 3, also the least square has a better performance than the lasso in flare package. But in high dimensional case, we can't get unique solutions for Adaptive Lasso and least square.
- For the four penalty terms in adaptive lad lasso, the larger the λ, the larger the penalty, which can kill more significant variables, as we get showed in the 5 scenarios.
- When compared the independent data with correlated data, we can find the mean square error in correlated setting is much larger than the independent case. This is because the correlation within the variables can lead to phenomenon of over-dispersion, which can make something significant actually they are not. So for the correlated data set, we need to consider other models for variance correction or the model used for correlated data set to get more accurate conclusions.