BIOS 664 HW2

Meng-Ni Ho

3/11/2019

Instruction

The primary goal of this assignment is to construct linear prediction algorithms to predict gene expression levels using genetic variants.

Read in data

Two genes:

1. NSG00000146574.15:

training sample size: 570 x 7788 (`gene_train1`)
testing sample size: 100 x 7788 (`gene_test1`)
normalized expression level data for training samples: 570 x 1 (`pheno_train1`)

2. ENSG00000238142.1

training sample size: 570 x 5230 (`gene_train2`)
testing sample size: 100 x 5230 (`gene_test2`)
normalized expression level data for training samples: 570 x 1 (`pheno_train2`)

```
# gene 574
gene_train1 = as.matrix(read.table("ENSG00000146574.15.training_geno.dat", head = T))
pheno_train1 = as.matrix(read.table("ENSG00000146574.15.training_pheno.dat", head = F))
gene_test1 = as.matrix(read.table("ENSG00000146574.15.testing_geno.dat", head = T))

# gene 142
gene_train2 = as.matrix(read.table("ENSG00000238142.1.training_geno.dat", head = T))
pheno_train2 = as.matrix(read.table("ENSG00000238142.1.training_pheno.dat", head = F))
gene_test2 = as.matrix(read.table("ENSG00000238142.1.testing_geno.dat", head = T))
```

Problems

Question 1

Fit the two training datasets by finding a prediction function containing only a single best predictor/SNP. Record this set of the prediction functions as `predict_single`.

1. Gene1: ENSG00000146574.15.

```
gene1_snp = apply(gene_train1, 2, function(x) summary(lm(pheno_train1 ~ x))$r.squared)
gene1_best_snp = which.max(gene1_snp) #chr7_6845133_A_C_b38: 3239
# fit using the best SNP: chr7_6845133_A_C_b38
```

```
gene1_fit = lm(pheno_train1 ~ chr7_6845133_A_C_b38, data = as.data.frame(gene_train1))
predict_single_gene1 = predict(gene1_fit, as.data.frame(gene_test1), type = 'response')

print(paste0("Best SNP for ENSG00000146574.15. is ", names(gene1_best_snp)))
```

```
## [1] "Best SNP for ENSG00000146574.15. is chr7_6845133_A_C_b38"
```

2. Gene2: ENSG00000238142.1.

```
gene2_snp = apply(gene_train2, 2, function(x) summary(lm(pheno_train2 ~ x))$r.squared)
gene2_best_snp = which.max(gene2_snp) #chr1_16927653_T_C_b38: 2258
gene2_fit = lm(pheno_train2 ~ chr1_16927653_T_C_b38, data = as.data.frame(gene_train2))
predict_single_gene2 = predict(gene2_fit, as.data.frame(gene_test2), type = 'response')

print(paste0("Best SNP for ENSG00000238142.1. is ", names(gene2_best_snp)))
```

```
## [1] "Best SNP for ENSG00000238142.1. is chr1_16927653_T_C_b38"
```

Question 2

Find the best subset of predictors using forward (or backward, or forward-backword) subset selection algorithms. Fit the two training datasets with your best subset predictors. Record this set of prediction functions as `predict_stepwise`.

1. Gene1: ENSG00000146574.15.

```
set.seed(15)

# Extract and remove highly correlated SNPs (correlation > 0.7) in training datasets
gene1_cor = cor(gene_train1, method = 'pearson')
gene1_cor[upper.tri(gene1_cor)] = 0
diag(gene1_cor) = 0

# new training sets with corr < 0.7
new_gene_train1 = gene_train1[, !apply(gene1_cor, 2, function(x) any(x > 0.7))] # 570 x 62
# get p-value
pval_gene1 = apply(new_gene_train1, 2, function(x) summary(lm(pheno_train1 ~ x))$coef[2, 4])
# extract SNP with p-value < 0.1
new_gene_train1 = new_gene_train1[, -which(pval_gene1 > 0.1)] # 570 x 254

# set cross-validation
control = trainControl(method = 'cv', number = 10)
# new training set
train_bind_gene1 = cbind(new_gene_train1, pheno_train1)

# perform stepwise
gene1_stepwise = train(V1 ~ ., data = train_bind_gene1, method = 'leapSeq', trControl = control)
# get covariates
gene1_beta = coef(gene1_stepwise$finalModel, 3)

# perform linear regression using covariates extracted from stepwise
```

```r
gene1_fit_stepwise = lm(V1 ~ chr7_6841876_C_T_b38 + chr7_6843271_A_G_b38 + chr7_6844357_G_A_b38, data =
# get predicted values
predict_stepwise_gene1 = predict(gene1_fit_stepwise, as.data.frame(gene_test1), type = 'response')
```

Results from stepwise:

```r
print(gene1_stepwise)
```

```
## Linear Regression with Stepwise Selection
##
## 570 samples
## 254 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 513, 514, 513, 513, 513, 514, ...
## Resampling results across tuning parameters:
##
##   nvmax  RMSE       Rsquared   MAE
##   2      0.5187703  0.3210325  0.4114848
##   3      0.4759350  0.4314424  0.3710241
##   4      0.5049845  0.3353188  0.3930751
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was nvmax = 3.
```

```r
kable(as.data.frame(gene1_beta))
```

	gene1_beta
(Intercept)	-0.0020457
chr7_6841876_C_T_b38	0.7052655
chr7_6843271_A_G_b38	-0.6607898
chr7_6844357_G_A_b38	0.5508647

Results from linear regression using covariates extracted from stepwise

```r
gene1_fit_stepwise
```

```
##
## Call:
## lm(formula = V1 ~ chr7_6841876_C_T_b38 + chr7_6843271_A_G_b38 +
##     chr7_6844357_G_A_b38, data = as.data.frame(train_bind_gene1))
##
## Coefficients:
##          (Intercept)  chr7_6841876_C_T_b38  chr7_6843271_A_G_b38
##            -0.002046              0.705266             -0.660790
## chr7_6844357_G_A_b38
##             0.550865
```

2. Gene2: ENSG00000238142.1.

```
# Extract and remove highly correlated SNPs (correlation > 0.7) in training datasets
gene2_cor = cor(gene_train2, method = 'pearson')
gene2_cor[upper.tri(gene2_cor)] = 0
diag(gene2_cor) = 0
# new training sets with corr < 0.7
new_gene_train2 = gene_train2[, !apply(gene2_cor, 2, function(x) any(x > 0.7))] # 570 x 616
# get p-value
pval_gene2 = apply(new_gene_train2, 2, function(x) summary(lm(pheno_train2 ~ x))$coef[2, 4])
# extract SNP with p-value < 0.1
new_gene_train2 = new_gene_train2[, -which(pval_gene2 > 0.1)] # 570 x 141

train_bind_gene2 = cbind(new_gene_train2, pheno_train2)

# perform stepwise
gene2_stepwise = train(V1 ~ ., data = train_bind_gene2, method = 'leapSeq', trControl = control)
# get covariates
gene2_beta = coef(gene2_stepwise$finalModel, 2)

# perform linear regression using covariates extracted from stepwise
gene2_fit_stepwise = lm(V1 ~ chr1_16533405_G_A_b38 + chr1_16948523_C_A_b38
                        , data = as.data.frame(train_bind_gene2))
# get predicted values
predict_stepwise_gene2 = predict(gene2_fit_stepwise, as.data.frame(gene_test2), type = 'response')
```

Results from stepwise:

```
print(gene2_stepwise)
```

```
## Linear Regression with Stepwise Selection
##
## 570 samples
## 141 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 514, 514, 513, 513, 511, 512, ...
## Resampling results across tuning parameters:
##
##   nvmax  RMSE       Rsquared   MAE
##   2      0.5779090  0.1241325  0.4604960
##   3      0.5796133  0.1225383  0.4633878
##   4      0.5775184  0.1338229  0.4547286
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was nvmax = 4.
```

```
kable(as.data.frame(gene2_beta))
```

	gene2_beta
(Intercept)	-0.0097652

	gene2_beta
chr1_16533405_G_A_b38	0.3756549
chr1_16948523_C_A_b38	0.6175109

Results from linear regression using covariates extracted from stepwise

```r
print(gene2_fit_stepwise)
```

```
##
## Call:
## lm(formula = V1 ~ chr1_16533405_G_A_b38 + chr1_16948523_C_A_b38,
##     data = as.data.frame(train_bind_gene2))
##
## Coefficients:
##           (Intercept)  chr1_16533405_G_A_b38  chr1_16948523_C_A_b38
##             -0.009765               0.375655               0.617511
```

Question 3

Fit the two training datasets by the Lasso regression algorithm. Describe how the tuning parameters are selected. Record this set of prediction functions as `predict_lasso`.

(1) Tuning parameters: `alpha = 1`

(2) Use minimized mean squared error as a critera when selecting predictors: set `type.measure = 'mse'`

(3) Choose the model with the smallest lambda to predict testing samples: set `s = 'lambda.min'`

1. Gene1: ENSG00000146574.15.

```r
# in `cv.glmnet`: alpha=1 is the lasso penalty, and alpha=0 the ridge penalty.
gene1_lasso = cv.glmnet(gene_train1, pheno_train1, family = 'gaussian',
                        type.measure = 'mse', alpha = 1)
predict_lasso_gene1 = predict(gene1_lasso, gene_test1, type = 'response', s = 'lambda.min')
print(gene1_lasso)
```

```
##
## Call:  cv.glmnet(x = gene_train1, y = pheno_train1, type.measure = "mse",      family = "gaussian",
##
## Measure: Mean-Squared Error
##
##      Lambda Measure       SE Nonzero
## min 0.01067  0.1020 0.004436     207
## 1se 0.02582  0.1062 0.005827      83
```

```r
# dimension reduction: select only non-zero coefficients
lasso_coef = predict(gene1_lasso, type="coefficient", s = 'lambda.min')
col_reduce = colnames(gene_train1)[which(lasso_coef!=0)]
```

2. Gene2: ENSG00000238142.1.

```
gene2_lasso = cv.glmnet(gene_train2, pheno_train2, family = 'gaussian',
                        type.measure = 'mse', alpha = 1)
predict_lasso_gene2 = predict(gene2_lasso, gene_test2, type = 'response', s = 'lambda.min')
```

Question 4

Fit the two training datasets by the ridge regression algorithm. Describe how the tuning parameters areselected. Record this set of prediction functions as `predict_ridge`.

(1) Tuning parameters: `alpha = 0`

(2) Use minimized mean squared error as a critera when selecting predictors: set `type.measure = 'mse'`

(3) Choose the model with the smallest lambda to predict testing samples: set `s = 'lambda.min'`

1. Gene1: ENSG00000146574.15.

```
gene1_ridge = cv.glmnet(gene_train1, pheno_train1, family = 'gaussian',
                         type.measure = 'mse', alpha = 0)
predict_ridge_gene1 = predict(gene1_ridge, gene_test1, type = 'response', s = 'lambda.min')
print(gene1_ridge)
```

```
##
## Call:  cv.glmnet(x = gene_train1, y = pheno_train1, type.measure = "mse",      family = "gaussian", a
##
## Measure: Mean-Squared Error
##
##     Lambda Measure       SE Nonzero
## min  4.617   0.124 0.009286    7788
## 1se  7.702   0.133 0.009656    7788
```

2. Gene2: ENSG00000238142.1.

```
gene2_ridge = cv.glmnet(gene_train2, pheno_train2, family = 'gaussian',
                        type.measure = 'mse', alpha = 0)
predict_ridge_gene2 = predict(gene2_ridge, gene_test2, type = 'response', s = 'lambda.min')
print(gene2_ridge)
```

```
##
## Call:  cv.glmnet(x = gene_train2, y = pheno_train2, type.measure = "mse",      family = "gaussian", a
##
## Measure: Mean-Squared Error
##
##     Lambda Measure      SE Nonzero
## min   2.25  0.3079 0.02341    5230
## 1se  48.39  0.3311 0.02727    5230
```

Question 5

Fit the two training datasets by the elastic-net regression algorithm. Describe how the tuning parametersare selected. Record this set of prediction functions as `predict_enet`.

(1) Tuning parameters: `alpha = 0.5`

(2) Use minimized mean squared error as a critera when selecting predictors: set `type.measure = 'mse'`

(3) Choose the model with the smallest lambda to predict testing samples: set `s = 'lambda.min'`

1. Gene1: ENSG00000146574.15.

```
gene1_elastic = cv.glmnet(gene_train1, pheno_train1, family = 'gaussian',
                          type.measure = 'mse', alpha = 0.5)
predict_elastic_gene1 = predict(gene1_elastic, gene_test1, type = 'response', s = 'lambda.min')
print(gene1_elastic)
```

```
##
## Call:  cv.glmnet(x = gene_train1, y = pheno_train1, type.measure = "mse",      family = "gaussian", a
##
## Measure: Mean-Squared Error
##
##       Lambda Measure       SE Nonzero
## min 0.01470 0.09863 0.007794     299
## 1se 0.05409 0.10615 0.008889      92
```

2. Gene2: ENSG00000238142.1.

```
gene2_elastic = cv.glmnet(gene_train2, pheno_train2, family = 'gaussian',
                         type.measure = 'mse', alpha = 0.5)
predict_elastic_gene2 = predict(gene2_elastic, gene_test2, type = 'response', s = 'lambda.min')
print(gene2_elastic)
```

```
##
## Call:  cv.glmnet(x = gene_train2, y = pheno_train2, type.measure = "mse",      family = "gaussian", a
##
## Measure: Mean-Squared Error
##
##      Lambda Measure      SE Nonzero
## min 0.0767  0.3189 0.01579      85
## 1se 0.1541  0.3336 0.01510      23
```

Question 6

Comments on the prediction functions you have constructed from 1 - 5. Which prediction function do you expect to have the best performance on the testing data? Why?

1. Compare MSE using single SNP predictors (*simple linear regression*) to predictors resulted from *stepwise selection*:

(1) gene1: MSE_single is smaller

```r
c(MSE_single_gene1 = sum(gene1_fit$residuals^2),
  MSE_stepwise_gene1 = sum(gene1_fit_stepwise$residuals^2))
```

```
##   MSE_single_gene1 MSE_stepwise_gene1
##           102.4834           122.6349
```

(2) gene2: MSE_stepwise is smaller

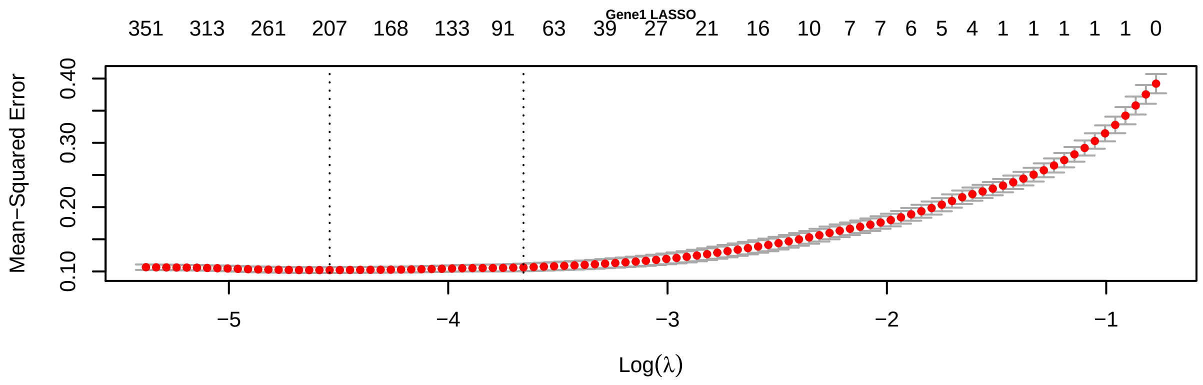
```r
c(MSE_single_gene2 = sum(gene2_fit$residuals^2),
  MSE_stepwise_gene2 = sum(gene2_fit_stepwise$residuals^2))
```

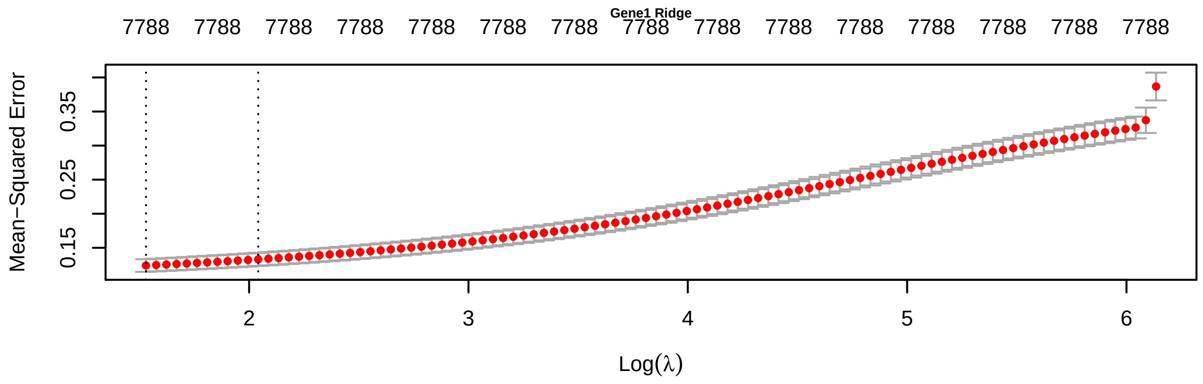
```
##   MSE_single_gene2 MSE_stepwise_gene2
##           197.2943           186.3019
```

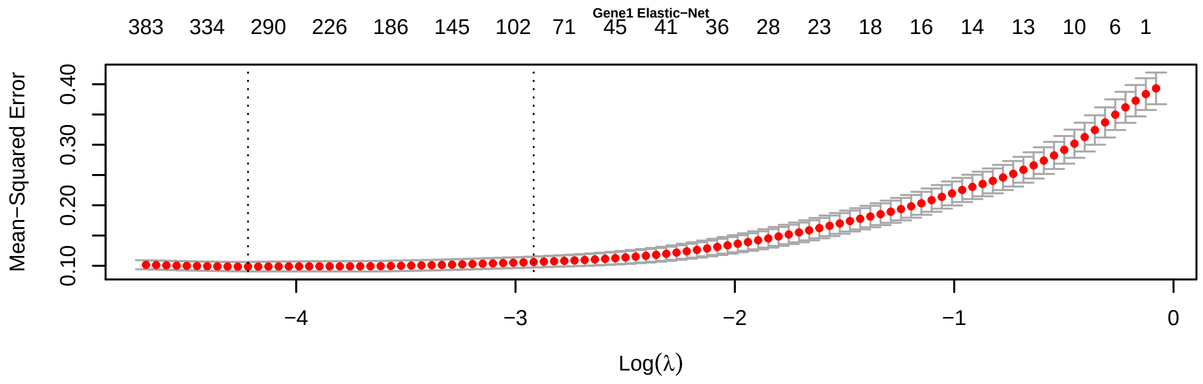
2. Compare MSE: lasso, ridge, elastic-net

(1) gene1: Lasso is better
Lasso contains least predictors (around 344) reaching MSE around 0.1, elastic-net contains more predictors (374) but still performs better than ridge (ridge contains all predictors to reach MSE 0.15)

```r
par(mfrow=c(3,1))
plot(gene1_lasso, main = 'Gene1 LASSO', cex.main = 0.6)
plot(gene1_ridge, main = 'Gene1 Ridge', cex.main = 0.6)
plot(gene1_elastic, main = 'Gene1 Elastic-Net', cex.main = 0.6)
```

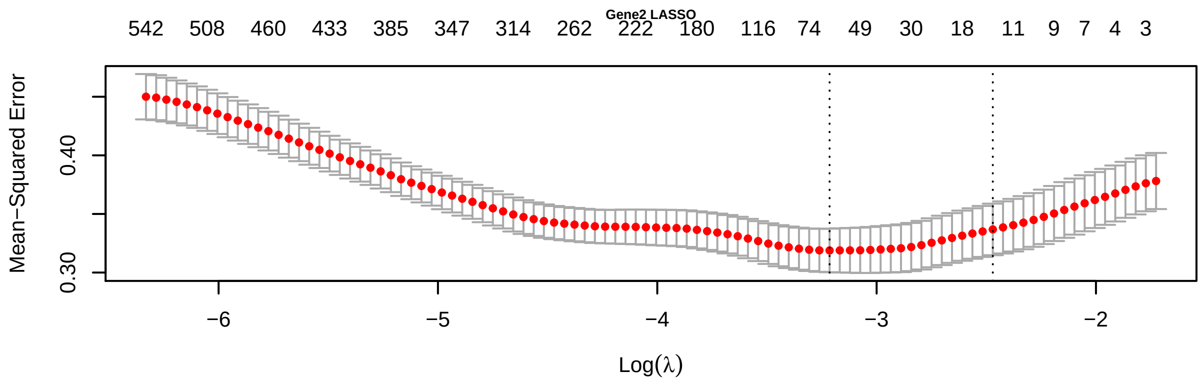


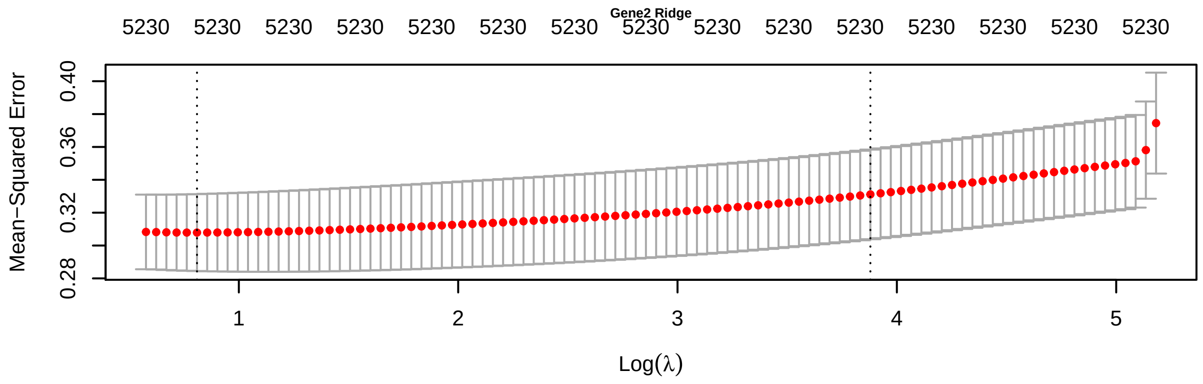


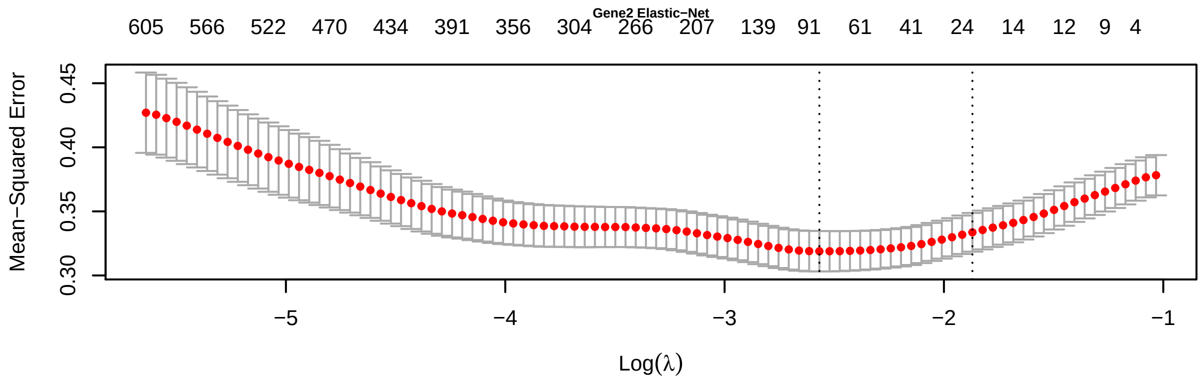


(2) gene2: Lasso is better Lasso contains least predictors (around 40~67) reaching MSE around 0.3, elastics-net needs more predictors (around 74~121), but still performs better than ridge (ridge contains all predictors and reach MSE to 0.40)

```r
par(mfrow=c(3,1))
plot(gene2_lasso, main = 'Gene2 LASSO', cex.main = 0.6)
plot(gene2_ridge, main = 'Gene2 Ridge', cex.main = 0.6)
plot(gene2_elastic, main = 'Gene2 Elastic-Net', cex.main = 0.6)
```







Question 7

Apply the prediction functions constructed from 1 - 5 to corresponding testing data set.

1. Gene1: ENSG00000146574.15.

```
# gene 1
test_result_gene1 = as.data.frame(cbind(predict_single_gene1, predict_stepwise_gene1,
                                        predict_lasso_gene1, predict_ridge_gene1, predict_elastic_gene1

colnames(test_result_gene1) = c("predict_single", "predict_stepwise", "predict_lasso", "predict_ridge",
```

```r
write.table(test_result_gene1, file = "test_result_gene1.txt", sep = "\t",
            row.names = TRUE, col.names = TRUE)
```

2. Gene2: ENSG00000238142.1.

```r
# gene 2
test_result_gene2 = as.data.frame(cbind(predict_single_gene2, predict_stepwise_gene2,
                                        predict_lasso_gene2, predict_ridge_gene2, predict_elastic_gene2)
colnames(test_result_gene2) = c("predict_single", "predict_stepwise", "predict_lasso", "predict_ridge",
write.table(test_result_gene2, file = "test_result_gene2.txt", sep = "\t",
            row.names = TRUE, col.names = TRUE)
```