

Practical 3: The epigenetic clock: Building a prediction rule

Verena Zuber, Jelena Besovic, Alpha Forna, and Saredo Said

7/2/2019

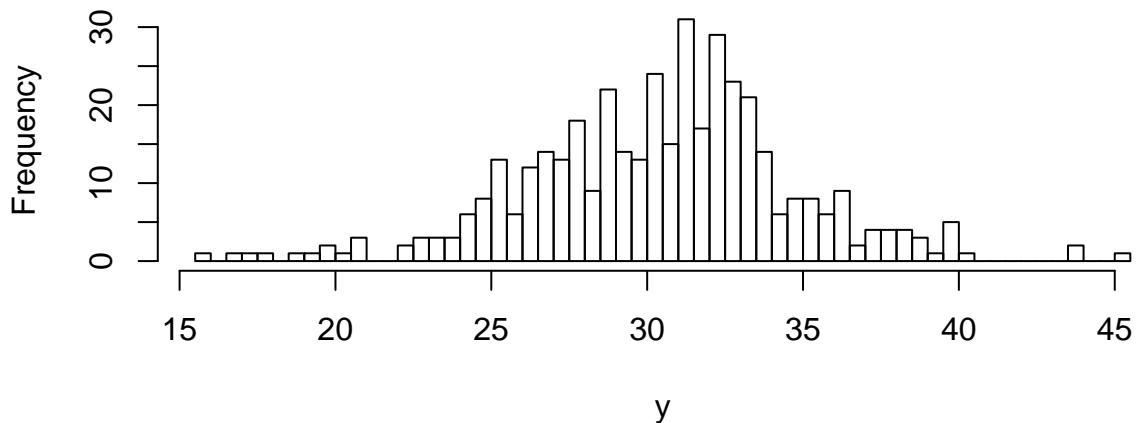
Part 1: The epigenetic clock: A predictive signature for ageing using penalised regression

In this practical we consider again the same data on $n = 409$ healthy mice and methylation of $p = 3,663$ conserved methylation sites as last week. Our goal is to train our own epigenetic clock and use the methylation data to predict the biological age of mice.

Load the dataset, that contains the methylation matrix as predictor matrix and the age of the mice (in months) stored in the vector y . Familiarise yourself with the dataset using the following commands

```
load("data_epigenetic_clock_control")
#alternatively try load("data_epigenetic_clock_control.dms")
y = control_mice$y_control
hist(y,breaks=50)
```

Histogram of y



```
x = control_mice$x_control
dim(x)
```

```
## [1] 409 3663
```

Question 1.1

First load the `glmnet` package and fit a lasso regression, where you use y as the outcome and x as predictor matrix (always make sure x is a matrix and not a data frame). For the first question to get the `glmnet` function to run use a fixed regularisation parameter and set lambda to 0.9. Run the lasso and see how many beta-coefficients are unequal zero and thus included in the model.

```
library(glmnet)
```

Reply: There are 7 methylation sites that are included in the model.

```

is.matrix(x)

## [1] TRUE

lasso.out11 = glmnet(x,y,family="gaussian",alpha=1,lambda=0.9)
sum(abs(coef(lasso.out11))>0)

```

```
## [1] 7
```

Question 1.2

When performing penalised regression it is not advised to set the regularisation parameter before seeing the data. It is good practice to perform cross-validation (cv) to set the regularisation parameter. Use the cv.glmnet function and the option type.measure = "mse" to optimise the mean squared error (mse). Find the lambda parameter that minimises the cv mse using the value \$lambda.min. What is the lambda parameter that is largest, but has a mse that is within one standard error of the minimum mse using the value \$lambda.1se?

Reply: lambda = 0.023 minimises the mse, but lambda = 0.155 is the largest lambda (smallest model) that has a mse that is within 1 standard error of the minimum mse.

```

set.seed(12)
lasso.cv = cv.glmnet(x,y,family="gaussian",alpha=1, type.measure="mse")
lasso.cv$lambda.min

## [1] 0.02304228
lasso.cv$lambda.1se

## [1] 0.1551702

```

Question 1.3

Redo the cv and check if the optimal lambda parameters are the same. What can you do to control the random number generator in R?

Reply: Without the set.seed the results are different. Always use the set.seed() to reproduce specific cv results.

```

lasso.cv.new = cv.glmnet(x,y,family="gaussian",alpha=1, type.measure="mse")
lasso.cv.new$lambda.min

## [1] 0.02304228
lasso.cv.new$lambda.1se

## [1] 0.2051258

```

Question 1.4

Fit the two lasso models, one with the lambda that optimises the mse, the second with the largest lambda that is within one standard error of the minimum mse. How many variables are included in each model and discuss the impact of the regularisation.

Reply: The first model with minimum mse includes 373 variables.

```

lasso.out.min = glmnet(x,y,family="gaussian",alpha=1,lambda=lasso.cv$lambda.min)
sum(abs(coef(lasso.out.min))>0)

## [1] 373

```

Reply: The second model with lambda.1se includes 201 variables, and has almost half of the variables as the first model trained on the minimum mse. The second model is the sparsest model that has nearly the same mse (within 1 standard error) as the first model with the minimum mse.

```
lasso.out.1se = glmnet(x,y,family="gaussian",alpha=1,lambda=lasso.cv$lambda.1se)
sum(abs(coef(lasso.out.1se))>0)
```

```
## [1] 201
```

Question 1.5

Fit a cv to define the optimal regularisation for the ridge regression. What are the optimal lambda parameter for the minimum cv mse and 1 standard error within the minimum? Fit a ridge regression with the respective parameter.

Reply: This is how to fit cv for ridge. Note alpha=0 as option is ridge regression, alpha=1 is lasso and alpha in between is elastic net.

```
set.seed(123)
ridge.cv = cv.glmnet(x,y,family="gaussian",alpha=0, type.measure="mse")
ridge.cv$lambda.min

## [1] 14.47122
ridge.cv$lambda.1se
```

```
## [1] 102.0916
```

This is the final ridge regression fit.

```
ridge.out.min = glmnet(x,y,family="gaussian",alpha=0,lambda=ridge.cv$lambda.min)
ridge.out.1se = glmnet(x,y,family="gaussian",alpha=0,lambda=ridge.cv$lambda.1se)
sum(abs(coef(ridge.out.1se))>0)
```

```
## [1] 3664
```

Ridge regression does not perform variable selection, all regression coefficients are unequal to zero and thus included into the model. In contrast lasso and elastic net set regression coefficient to exactly zero and exclude them from the model.

Question 1.6

Fit a cv to define the two optimal regularisation parameter for elastic net regression. Focus on the largest lambda within 1 standard error of the minimum. This will provide the sparsest model (fewest predictors) that is almost as good as the one with the minimum cv mse. What are the optimal lambda and alpha parameter? Use the following code to search the optimal combination of lambda and alpha on a grid.

```
set.seed(1234)
a = seq(0.05, 0.95, 0.05)
search = foreach(i = a, .combine = rbind)%do%{
  cv = cv.glmnet(x,y,family = "gaussian", type.measure = "mse", alpha = i)
  data.frame(cvm = cv$cvm[cv$lambda == cv$lambda.1se], lambda.1se = cv$lambda.1se, alpha = i)
}
elasticnet.cv = search[search$cvm == min(search$cvm), ]
elasticnet.cv

##          cvm lambda.1se alpha
## 16 10.84984 0.08795974 0.8
```

Reply: CV in elastic net defines lambda.1se as 0.088 and alpha as 0.8. Note for elastic net alpha is between 0 and 1 and needs to be specified as well.

Finally fit an elastic net model using the optimal lambda and alpha regularisation parameter.

```
enet.out = glmnet(x, y, family = "gaussian", lambda = elasticnet.cv$lambda.1se, alpha = elasticnet.cv$alpha)
sum(abs(coef(enet.out))>0)
```

```
## [1] 294
```

There are 294 methylation sites included into the elastic net model.

Part 2: Which of the 3 models (ridge, lasso and elastic net) builds the better prediction rule?

In the second part we perform a cv to compare how well the three different models can predict new data. To this end we use the

```
library(crossval)
```

package for which we need to write a prediction function. Please see here how to define a prediction function for a linear regression model.

```
predfun.lm = function(train.x, train.y, test.x, test.y){  
  #fit the model and build a prediction rule  
  lm.fit = lm(train.y ~ ., data=train.x)  
  #predict the new observation based on the test data and the prediction rule  
  ynew = predict(lm.fit, test.x )  
  
  #compute mse as squared difference between predicted and observed outcome  
  out = mean( (ynew - test.y)^2 )  
  return( out )  
}
```

Use this code fragment to write a prediction function for the following

Question 2.1

Lasso (with the regularisation parameter lambda as set in Question 1.2 to be the largest lambda that has a mse that is within one standard error of the minimum mse using the value \$lambda.1se)

```
predfun.lasso = function(train.x, train.y, test.x, test.y){  
  
  glmnet.fit = glmnet(x=train.x, y=train.y, lambda = lasso.out.1se, alpha=1)  
  ynew = predict(glmnet.fit , test.x)  
  
  # compute squared error risk (MSE)  
  out = mean( (ynew - test.y)^2 )  
  return( out )  
}
```

Question 2.2

Ridge (with lambda as \$lambda.1se in Question 1.5)

```
predfun.ridge = function(train.x, train.y, test.x, test.y){  
  
  glmnet.fit = glmnet(x=train.x, y=train.y, lambda = ridge.out.1se, alpha=0)  
  ynew = predict(glmnet.fit , test.x)  
  
  # compute squared error risk (MSE)  
  out = mean( (ynew - test.y)^2 )  
  return( out )
```

```
}
```

Question 2.3

Elastic net (with lambda and alpha as \$lambda.1se in Question 1.6)

Reply: Since all three methods essentially build on the glmnet function, it is actually possible to build a more general prediction function, that has both, lambda and alpha as free parameters.

```
predfun.glmnet = function(train.x, train.y, test.x, test.y, lambda = lambda, alpha=alpha){

  glmnet.fit = glmnet(x=train.x, y=train.y, lambda = lambda, alpha=alpha)
  ynew = predict(glmnet.fit , test.x)

  # compute squared error risk (MSE)
  out = mean( (ynew - test.y)^2 )
  return( out )

}
```

Question 2.4

Use the crossval package to perform a k-fold cross validation with k=5 folds. For each of the three methods output the mean and standard error of the cv test error and discuss which method generalises best to new data.

Reply: We use predfun.glmnet as the most general prediction rule and adjust the parameters according to the regularised regression used. We see that ridge regression has the worst prediction performance. The lasso improves over the ridge, while the elastic net slightly improves over the lasso.

```
set.seed(24)
cv.out.ridge = crossval(predfun.glmnet, x, y, lambda=ridge.cv$lambda.1se, alpha=0,
  K=5, B=20, verbose=FALSE)
cv.out.lasso = crossval(predfun.glmnet, x, y, lambda=lasso.cv$lambda.1se, alpha=1,
  K=5, B=20, verbose=FALSE)
cv.out.enet = crossval(predfun.glmnet, x, y, lambda=elasticnet.cv$lambda.1se,
  alpha=elasticnet.cv$alpha, K=5, B=20, verbose=FALSE)

#save to table
table.out = rbind(c(cv.out.ridge$stat, cv.out.ridge$stat.se),
  c(cv.out.lasso$stat, cv.out.lasso$stat.se), c(cv.out.enet$stat, cv.out.enet$stat.se))
rownames(table.out) = c("ridge", "lasso", "elastic net")
colnames(table.out) = c("mse", "se")
table.out

##           mse        se
## ridge      15.90795 0.3050867
## lasso      12.87486 0.2694219
## elastic net 12.38798 0.2891237
```

Part 3 (optional): Differential expression using shrinkage t-score

The third question considers the response of breast cancer patients to treatment. The aim is to identify differentially expressed genes in 2 responder groups (pathologic complete response or minimal residual cancer burden [RCB-I] defining excellent response, vs moderate or extensive residual cancer burden [RCB-II/III] defining lesser response.

For the original publication please see <https://jamanetwork.com/journals/jama/fullarticle/899864>

Load the dataset including $n = 414$ patients and the predictor matrix including expression of $p = 22,283$ genes.

```
load("JAMA2011_breast_cancer")
#alternatively try load("JAMA2011_breast_cancer.dms")
y = data_bc$rcb
table(y)

## y
##   0    1
## 117 297

x = data_bc$x
dim(x)

## [1] 414 22283
```

Question 3.1

Compute first the sample variance estimate of each gene using the var.shrink function in the

```
library(corrplot)
```

package using the option lambda=0 for no shrinkage and save it in a vector. Then compute the shrinkage variance using the same function but do not specify the shrinkage parameter, so that lambda is estimated from the data. Plot two boxplots of the 1000 smallest variances for the sample and the shrinkage variance to visualise the impact of the shrinkage.

Reply: We use the var.shrink function with lambda=0 to estimate the sample variance and var.shrink without specifying lambda to estimate the shrinkage parameter. When there is no lambda specified var.shrink estimates lambda from the data.

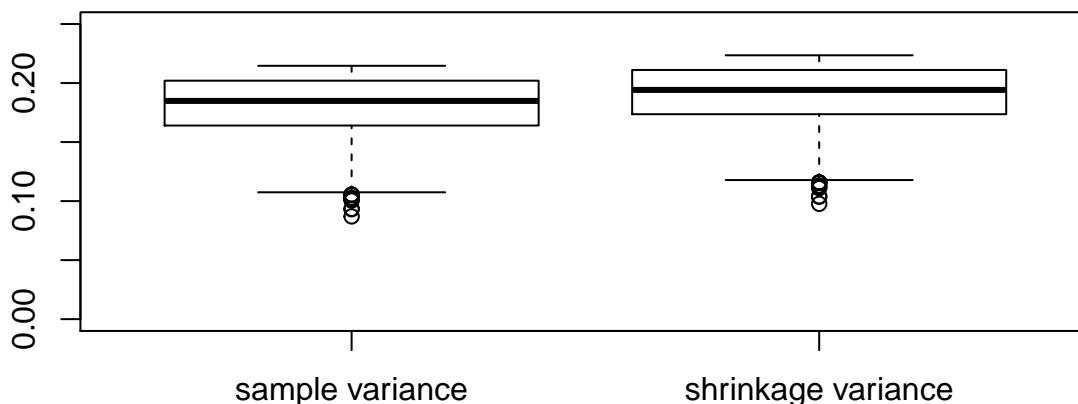
```
sample.var = var.shrink(x,lambda=0)

## Specified shrinkage intensity lambda.var (variance vector): 0
shrink.var = var.shrink(x)

## Estimating optimal shrinkage intensity lambda.var (variance vector): 0.0137
```

The shrinkage effect was minimal (0.0137 is close to 0) but we can see that the shrinkage variance is larger than the sample variance. This is due to the shrinkage towards the median variance as target.

```
boxplot(cbind(sort(sample.var)[1:1000],sort(shrink.var)[1:1000]), ylim=c(0,0.25),
names=c("sample variance", "shrinkage variance"))
```



Question 3.2

Next compute for each of the $p = 22,283$ genes the shrinkage t -score using the function `shrinkt.stat()` in the `library(st)`

st package. Save this to a vector named `shrinkt`.

Reply: The shrinkage t-score can be computed as follows.

```
shrinkt = shrinkt.stat(X=as.matrix(x), L=as.factor(y))
```

```
## Number of variables: 22283
## Number of observations: 414
## Number of classes: 2
##
## Estimating optimal shrinkage intensity lambda.freq (frequencies): 0.0104
## Estimating variances (pooled across classes)
## Estimating optimal shrinkage intensity lambda.var (variance vector): 0.014
```

Question 3.3

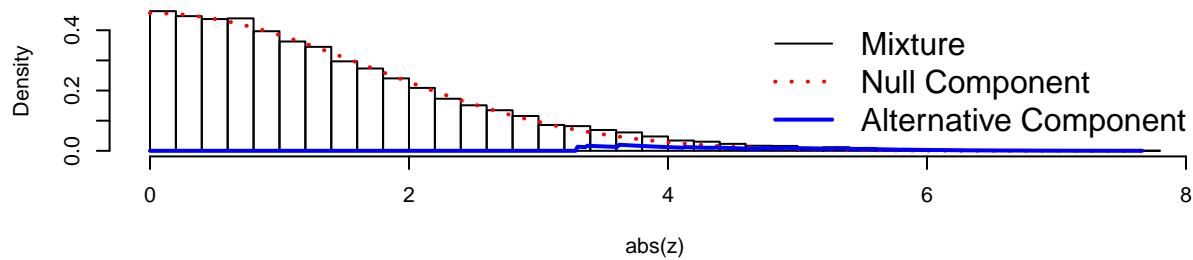
Finally perform a multiple testing correction on the shrinkage t -statistic saved in the `shrinkt` object using the `fdrtool` package. The great advantage of the `fdrtool` is that it works also on t -statistics like the shrinkage t -score. Specify `statistic = "normal"` as option in the `fdrtool` to fit a Normal-distribution as Null distribution for the shrinkage t -score. Use the local `fdr` and a threshold of 0.2.

How many genes do you identify as differentially expressed between excellent and non-response to treatment?

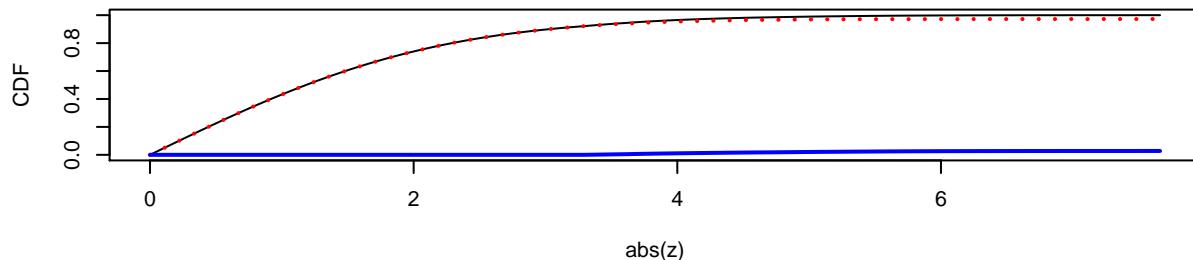
Reply: There are 11 genes that have a local fdr smaller than 0.2. These 11 genes are very likely to be differentially expressed between responders and non-responders to treatment.

```
fdr.out.st = fdrtool(shrinkt, statistic = "normal", verbose =FALSE)
```

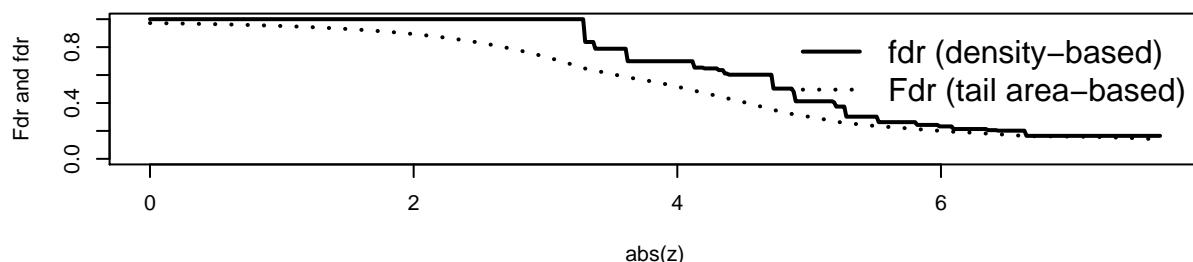
Type of Statistic: z-Score (sd = 1.698, eta0 = 0.9723)



Density (first row) and Distribution Function (second row)



(Local) False Discovery Rate



```
sum(fdr.out.st$lfdr<0.2)
## [1] 11
fdr.out.tab = cbind(shrinkt, fdr.out.st$lfdr)[which(fdr.out.st$lfdr<0.2),]
colnames(fdr.out.tab)=c("shrinkt", "localfdr")
fdr.out.tab[order(abs(fdr.out.tab[,1]),decreasing=TRUE),]

##      shrinkt localfdr
## [1,] -7.661996 0.1646999
## [2,]  7.319288 0.1646999
## [3,]  7.054506 0.1646999
## [4,]  6.940104 0.1646999
## [5,]  6.915592 0.1646999
## [6,]  6.837610 0.1646999
## [7,] -6.822309 0.1646999
## [8,] -6.743898 0.1646999
## [9,] -6.701298 0.1646999
## [10,] -6.695584 0.1646999
## [11,] -6.671170 0.1646999
```