Practical 5: The epigenetic clock: Predicting biological age on new data

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

14/2/2019

Part 1: The epigenetic clock: Non-parametric prediction using random forests

In this practical we consider again the same data on $n = 409$ healthy mice and methylation of $p = 3,663$ conserved methylation sites as in the last two weeks. This time we are interested in random forests and see if random forests will provide a better prediction rule than penalised regression techniques. Install and load the package

```
library("randomForest")
```

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

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

Question 1.1

First compute a random forest with the options ‘ntree=100’ and ‘importance = TRUE’ and save the output to an object call rf.out or similar.

Question 1.2

Random forests have an intrinsic way of assigning variable importance to the predictors. How do random forests with quantitative outcome rank variables according to their importance? Rank the variables according to their importance and display the 10 most important methylation sites for ageing in the random forest algorithm. Use the varImpPlot() function to visualise the variable importance.

Question 1.3

How would you run bagging using the randomForest function? What is the interpretation of the mtry option?

Question 1.4

Next step is to write a prediction function for the random forest algorithm. Re-use your code from last week (Practical 3 Question 2) and see the prediction function for a linear regression model below. Set the number of trees as an open parameter.

```
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 )
}
```

Question 1.5

Compare the prediction performance using ntree = 10 and ntree = 100 random trees to train the random forest. To this end we use the

```r
library(crossval)
```

package and compare the two parameters using k-fold cross-validation (cv) using $k = 5$ folds.

Question 1.6

Finally, we want to know if random forests outperform regularised regression with respect to prediction performance. Re-use the following code from last week(Practical 3 Question 2) as prediction rule for glmnet() and use cv as implemented in crossval() to determine if random forests (ntree = 100) is better than lasso and elastic net as implemented in the glmnet package.

```r
library(glmnet)
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 )

}
```

Part 2: The epigenetic clock: Evaluating the impact of nitrogen dioxide on biological age

An environmental research group gets in contact with us. They have measured the methylation profile of $n = 140$ mice of which 40 mice have been exposed to nitrogen dioxide (NO2) for 10 hours a day for 10 weeks. The remaining 100 mice are healthy controls. Load the data and familiarise yourself with the data structure.

```r
load("data_epigenetic_clock_experimental")
#alternatively try load("data_epigenetic_clock_experimental.dms")
exposed = experimental_mice$exposed
table(exposed)
```

```
## exposed
##   0   1
## 100  40
```

```r
x.test = experimental_mice$x_exp
dim(x.test)
```

```
## [1]  140 3663
```

The working hypothesis is that NO2 exposure reduces the biological age of mice and makes the exposed mice age more quickly. The environmental research group asks us to predict the biological age of the mice using our algorithms to determine the biological age of a mouse.

Question 2.1

Predict the biological age of the $n = 140$ new mice using the lasso model (Practical 3: Question 1.4) from last week as implemented in the glmnet package. Use again $lambda.1se as the regularisation parameter and save the predicted age in an object called lasso.hat.

Question 2.2

Predict the biological age of the $n = 140$ new mice using the ridge model (Practical 3: Question 1.5) from last week. Use again $lambda.1se as the regularisation parameter and save the predicted age in an object called ridge.hat.

Question 2.3

Predict the biological age of the $n = 140$ new mice using the elastic net model (Practical 3: Question 1.6) from last week. Use again $lambda.1se as the regularisation parameter and save the predicted age in an object called enet.hat.

Question 2.4

Predict the biological age of the $n = 140$ new mice using the a random forest model. This time use 'ntree=500' to build a reliable prediction rule and save the predicted age in an object called rf.hat.

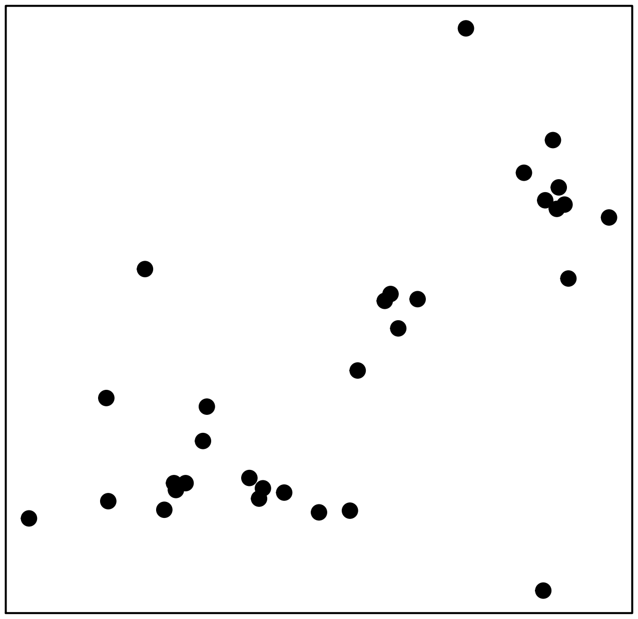
Question 2.5

Can you confirm the working hypothesis that NO2 exposure reduces the biological age of mice? Perform a t-test to see if the predicted biological age of the mice differs significantly between exposed and healthy mice. Perform a $t-$test for all four predictions done in Question 2.1-2.4. How will you advice the environmental research group?

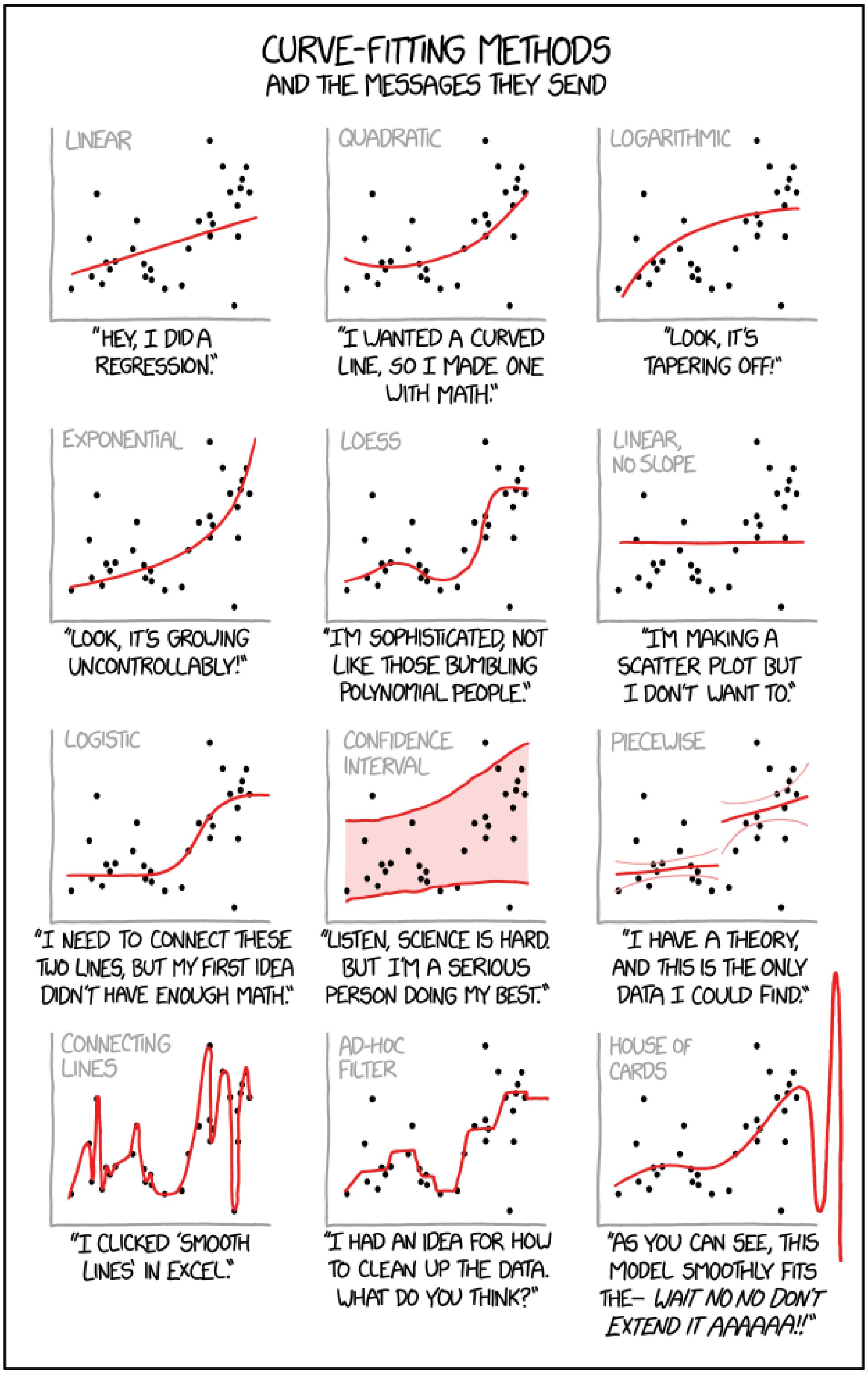
Part 3 (optional): Non-linear methods and curve-fitting

How many of the following curve-fitting methods as shown in Figure 1 can you fit in R? Load the data from blackboard into R using the following lines

```
curve.fitting = read.csv("curve-fitting.txt", sep="\t")
x = curve.fitting$x
y = curve.fitting$y
plot(x,y, pch=19, xlab="", ylab="", xaxt='n', yaxt='n')
```



```r
#Figure 1: Comic on curve-fitting methods from https://xkcd.com/2048/
knitr::include_graphics("curve_fitting.png")
```


CURVE-FITTING METHODS
AND THE MESSAGES THEY SEND
LINEAR
"HEY, I DID A REGRESSION."
QUADRATIC
"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."
LOGARITHMIC
"LOOK, IT'S TAPERING OFF!"
EXPONENTIAL
"LOOK, IT'S GROWING UNCONTROLLABLY!"
LOESS
"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."
LINEAR, NO SLOPE
"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."
LOGISTIC
"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."
CONFIDENCE INTERVAL
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."
PIECEWISE
"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND."
CONNECTING LINES
"I CLICKED 'SMOOTH LINES' IN EXCEL."
AD-HOC FILTER
"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"
HOUSE OF CARDS
"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE– WAIT NO NO DON'T EXTEND IT AAAAAA!!"

Question 3.1

Uncomment the following lines (Remove the hash tag) and discuss how linear regression can be used for curve-fitting.

```
#par(mfrow=c(1,3)) #How many figures next to each other?
#lm.object = lm(y~x) #Linear
#plot(x,y, pch=19, xlab="", ylab="")
#abline(lm.object, col="red", lwd = 2)
#lm.intercept = lm(y~1) #Linear, no slope
#plot(x,y, pch=19, xlab="", ylab="")
#abline(lm.intercept, col="red", lwd = 2)
#newx = seq(min(x), max(x), length.out=100)
#lm.confidence = predict(lm.object, newdata = data.frame(x=newx), interval = 'confidence') #Build confi
#plot(x,y, pch=19, xlab="", ylab="")
#abline(lm.object, col="red", lwd = 2)
#lines(newx, lm.confidence[ ,3], lty = 'dashed', col = 'red')
#lines(newx, lm.confidence[ ,2], lty = 'dashed', col = 'red')
```

Question 3.2

How can we fit non-linear relationships with the linear model?

```
#par(mfrow=c(1,3)) #How many figures next to each other?
#lm.quad = lm(y~x+I(x^2))
#plot(x,y, pch=19, xlab="", ylab="")
#curve(predict(lm.quad,data.frame(x=x)),col='red',lwd=2,add=TRUE)
#lm.log = lm(y~log(x))
#fitted=predict(lm.log,newdata=list(x=seq(from=0.5,to=4,length.out=100)))
#plot(x,y, pch=19, xlab="", ylab="")
#matlines(x=seq(from=0.5,to=4,length.out=100),fitted,lwd=2, col='red')
#lm.exp = lm(y~exp(x))
#fitted=predict(lm.exp,newdata=list(x=seq(from=0.5,to=4,length.out=100)))
#plot(x,y, pch=19, xlab="", ylab="")
#matlines(x=seq(from=0.5,to=4,length.out=100),fitted,lwd=2, col='red')
```

Question 3.3

Can we use a glm to fit a logistic function? And how does smoothing using the loess function work?

```
#par(mfrow=c(1,3))
#y.bin = as.numeric(y>mean(y))
#table(y.bin)
#glm.object = glm(y.bin~x, family = binomial)
#glm.fitted = fitted(glm.object)
#plot(x,y, pch=19, xlab="", ylab="")
#lines(glm.fitted, x=x, col="red")
#loess = loess(y~x, span=0.75)
#smoothed75 = predict(loess)
#plot(x,y, pch=19, xlab="", ylab="")
#lines(smoothed75, x=x, col="red")
#loess = loess(y~x, span=0.1)
#connectinglines = predict(loess)
#plot(x,y, pch=19, xlab="", ylab="")
#lines(connectinglines, x=x, col="red")
```

Question 3.4

The last three curve-fitting methods are advanced and are beyond the scope of the module. House of Cards is a mistery and we are not sure if it is implemented in R . . .

```
#par(mfrow=c(1,3))
#nls.out = nls(y ~ ifelse(x < median(x),ya+A*x,yb+B*x), data = data.frame(x,y))
#plot(x,y, pch=19, xlab="", ylab="")
#lines(x[1:15], fitted(nls.out)[1:15], lwd=2, col='red')
#lines(x[17:31], fitted(nls.out)[17:31], lwd=2, col='red')
#ymed = runmed(y,5)
#plot(x,y, pch=19, xlab="", ylab="")
#lines(x, ymed, lwd=2, col='red')
#House of Cards??
#plot(x,y, pch=19, xlab="", ylab="")
```