This protocol describes tools for analyzing and visualizing Illumina methylation 450k and EPIC array data with the minfi-package (the 27k array is not supported at the moment). The following codes were generated by Xinyang Yu, Tianye Jia and Sylvane Desrivières.

Please let us know if you run into problems along the way, contact us at: sylvane.desrivieres@kcl.ac.uk or xinyang.1.yu@kcl.ac.uk.

#####################################################################**Before you start, you need to download and install some required programs (which you may already have). The required programs are: R and the minfi bioconductor package (see** [**http://bioconductor.org/packages/release/bioc/vignettes/minfi/inst/doc/minfi.pdf**](http://bioconductor.org/packages/release/bioc/vignettes/minfi/inst/doc/minfi.pdf)**).** An SSH client, e.g. PuTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html), is required to if your data is stored on a linux/unix server.

#####################################################################

Windows and Mac users most likely want one of these versions of R:

* [Download R for Linux](https://cran.r-project.org/bin/linux/)
* [Download R for (Mac) OS X](https://cran.r-project.org/bin/macosx/)
* [Download R for Windows](https://cran.r-project.org/bin/windows/)

To install the minfi package, start R and enter:

if (!require("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("minfi")

To install the FlowSorted.Blood.450k package, which is necessary when using the function of “estimateCellCounts”, enter:

if (!require("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("FlowSorted.Blood.450k")

##################################################################### **Reading the methylation data**

#####################################################################

The procedure below requires IDAT files, containing the summarised beads information and the "Sample Sheet".csv file that describes the layout of the experiment.

Highlighted portions of the instructions require you to make changes so that the commands work on your system and data.

##open R and copy the lines below

require(minfi)

require(minfiData)

library(RColorBrewer)

library(minfi)

library(limma)

library(FlowSorted.Blood.450k)

setwd("/home/ENIGMAepigenetics/ ") # replace with your local working directory

## #################################

### Reading the methylation data ####

#################################

## Creating the initial object of the minfi analysis that contains the raw intensities in the green and red channels

datadir <- "/home/imagen/wave1/idat450k" #specify your local directory, which contains both raw IDAT files and the sample sheets (i.e., a .csv file, which includes the path for each sample’s IDAT file). The following scripts expect the sample sheet to include column names as shown in the figure below, where the header information (all lines up to [Data]) could be omitted (Variables such as sex, age, site and disease status can also be included as extra columns):

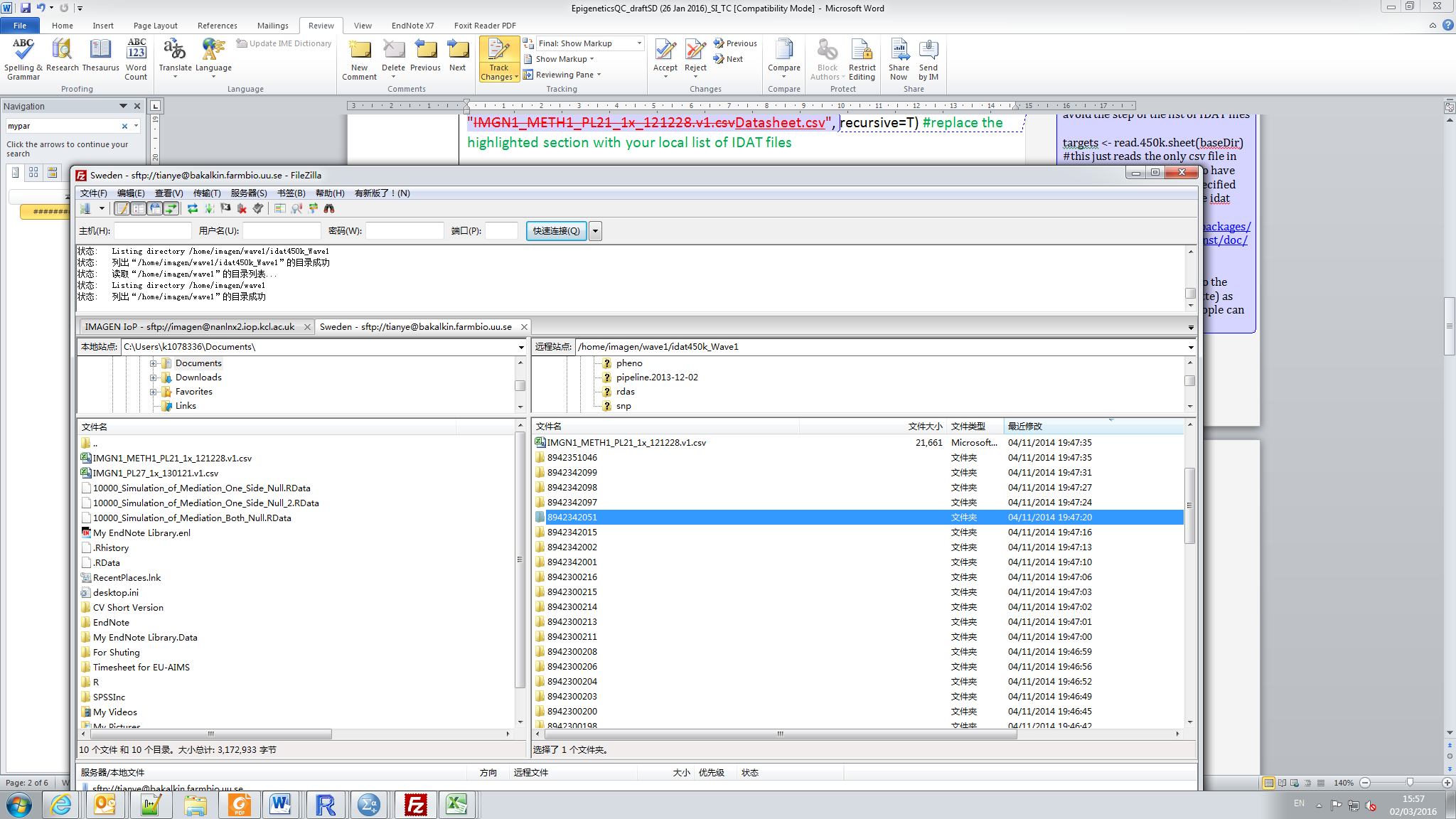


# We recommend combining information about all samples in a single sample sheet, if there are multiple sample sheets. The following code shows how to combine 3 sample sheets:

#targets=rbind(targets, read.metharray.sheet(datadir, "First Sheet.csv", recursive=T) #targets=rbind(targets, read.metharray.sheet(datadir, "Second Sheet.csv", recursive=T)

#targets=rbind(targets, read.metharray.sheet(datadir, "Third Sheet.csv", recursive=T)

# Your local directory should look like this:



targets=read.metharray.sheet(datadir, "IMGN1\_METH1\_PL21\_1x\_121228.v1.csv", recursive=T) #loads the corresponding .csv sample sheet file

RGset <- read.metharray.exp(base = datadir, targets = targets, verbose=T) #creates an object, which contains the raw data from the IDAT files

save(RGset,file="./RGset.rda") #saves the object

### If the initial object with raw methylation data had already been created as above, it can be directly loaded with the command below (if the file hasn’t been generated yet or if you don’t know what it means, please follow the procedure above):

#load("./RGset.rda")

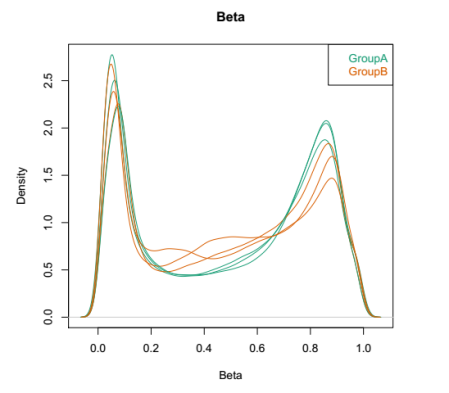
### Producing Quality Control plots

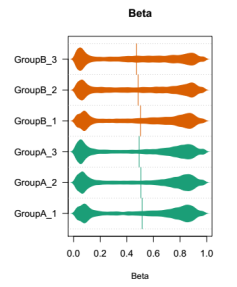
pd <- pData(RGset) #extracting the sample information (phenotype data) from the sample sheet

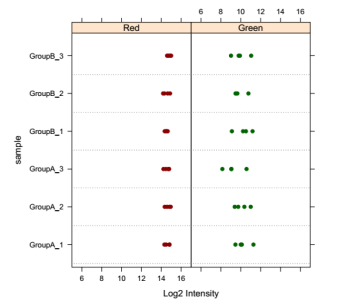
qcReport(RGset, sampNames = pd$Sample.ID, sampGroups=pd$Experimenter, pdf = "./qcReport.pdf") # produces a PDF QC report of common plots, colored by groups of samples. Check that the highlighted variables correspond to column names in your sample sheet, as illustrated above. In this example, the column named ‘Experimenter’ corresponds to the different waves by which our DNA samples were processed and hybridized.

#These plots are useful for identifying samples with data quality. They display summaries of signals from the array (e.g. density plots) as well as the values of several types of control probes included on the array. A good rule of thumb is to be wary of samples whose behaviour deviates from that of others in the same or similar experiments. The following figures display different types of plots, with no evidence for outliers in 6 samples.

# In case outliers are detected, their individuals’ Sample.ID should registered into the file ‘Outliers’, which will be used to remove outliers in a later step.







##################################

### Quality assessment of methylation data ####

##################################

object=preprocessIllumina(RGset) #We prefer preprocessIllumina because it considers the background correction as well as normalization to internal controls as described in the Illumina documentation. This will minimize the amount of variation between arrays.

object<-mapToGenome(object) #assign probes with its physical location on the genome

object=ratioConvert(object, type="Illumina") #convert raw methylation data

beta <- getBeta(object) #get the beta value for each probe

dat <- object #rename ‘object’ to ‘dat’

pd=pData(dat) #get phenotypes of methylation data

####Removing data from the X and Y chromosomes#######

keepIndex=which(!seqnames(dat)%in%c("chrX","chrY"))

beta <- beta[keepIndex,]

### Use MDS to check bad samples ####

mydist=dist(t(beta[sample(1:nrow(beta),10000),]))

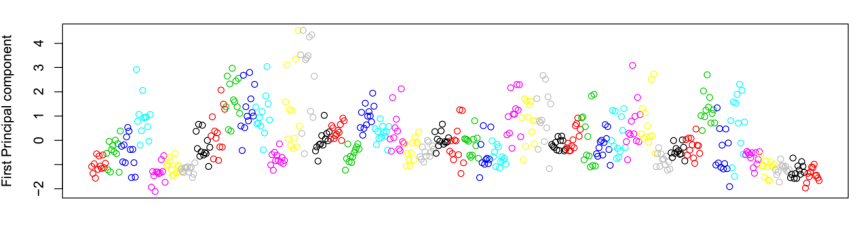
mds=cmdscale(mydist)

pdf("./Batch\_effect\_mds.pdf",width=11,height=4)

plot(mds[,1],col=as.numeric(as.factor(pd[,"Slide"]))+1,xlab="",ylab="First Principal component",xaxt="n") #can replace with your own potential confounder variables.

dev.off()

#An example of the output:



#the figure shows some, but not gigantic, impact of Slide information on the first component of methylation data.

### Use PCA to check batch effects ####

b <- beta - rowMeans(beta)

#Performing singular value decomposition (SVD), equivalent to principal component analysis with covariance matrix.

library(corpcor)

ss <- fast.svd(b,tol=0) # this may take a while

#Looking at the percent of variance explained by each principal component

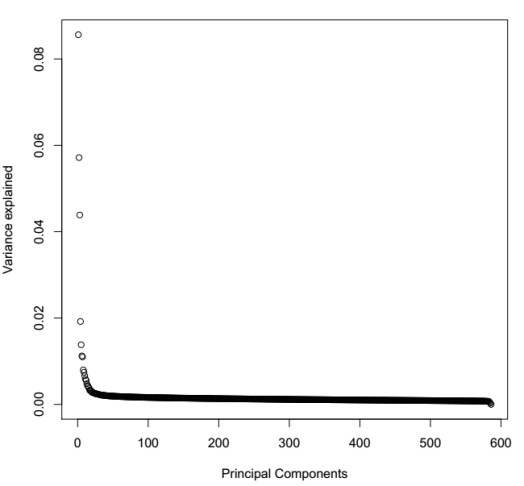
percvar <- ss$d^2/sum(ss$d^2) #calculates the variance explained by each component

pdf(file="./PCA\_distribution.pdf")

plot(percvar,xlab="Principal Components",ylab="Variance explained")

dev.off()

#example of output



## Plotting Batch Effect over Methylation Components

for (i in c("Array","Slide")){ #can add your own potential confounder variables.

pdf(file=paste("./batch\_",i,".pdf",sep="")) #output filename

variabletocolor=pd[,i] #assign different colours to batches

bob=levels(factor(variabletocolor))

colors=match(variabletocolor,bob)

pairs(ss$v[,1:4], col=colors, labels=c("PC1", "PC2", "PC3", "PC4"))

par(mfrow=c(2,2))

boxplot(ss$v[,1]~pd[,i],ylab="PC1",xlab=i)

boxplot(ss$v[,2]~pd[,i],ylab="PC2",xlab=i)

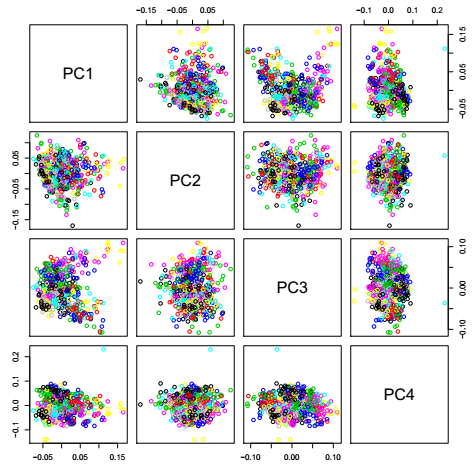
boxplot(ss$v[,3]~pd[,i],ylab="PC3",xlab=i)

boxplot(ss$v[,4]~pd[,i],ylab="PC4",xlab=i)

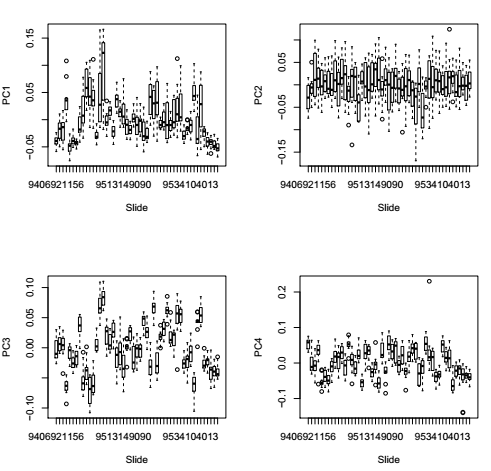
dev.off()

}

#An example of the outputs:



#this figure shows no clear subpopulation based on the first 4 principal components



#this figure shows the box plot of slides effect against the first 4 principal components. It appears that the slide has some, but no gigantic, impact on PC1, but a huge effect on PC3, i.e. individuals on different slides show significantly different distributions on PC3.

### Mark individuals out of normal range (i.e. median+3SD or median-3SD) based on first 4 components ###

RM <- rep(FALSE, nrow(ss$v))

for (i in 1:4) {

Median <- median(ss$v[,i]) #calculate median of each component

SD <- sd(ss$v[,i]) #calculate the standard deviation of each component

RM <- RM|(abs(ss$v[,i]-Median)>3\*SD) #mark individuals outside 3SD range as TRUE

}

# RM will be used to remove outliers at a later stage

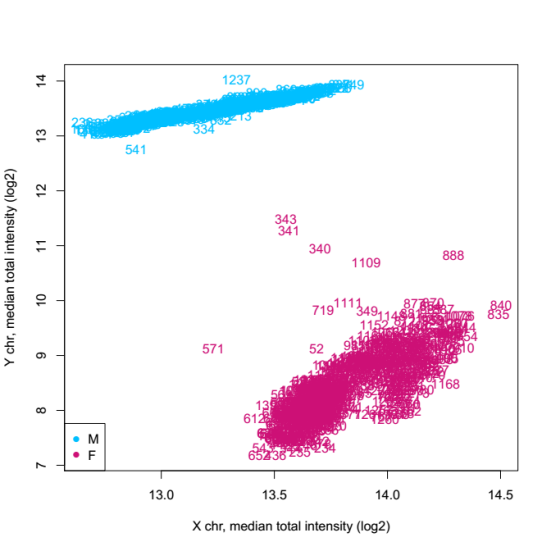
###Predicting sex using methylation data###

predictedSex <- getSex(dat, cutoff = -2)$predictedSex # N.B., this function does not handle datasets with only females or only males)

pdf(file="./Sex\_Plot.pdf")

plotSex(getSex(dat, cutoff = -2)) #Plot of predicted gender information as follows

dev.off()



pdf(file="./predicted\_gender.pdf")

Jitter1 <- jitter(as.numeric(as.factor(pd$Sex))) #Here we assume the self-reported gender information is integrated in the RGset and therefore is available as pd$Sex. Otherwise, please replace pd$Sex with a vector containing the self-reported gender information. Ideally, both self-reported and predictedSex should take the same format, i.e. “M” for male and “F” for female.

Jitter2 <- jitter(as.numeric(as.factor(predictedSex)))

plot(Jitter1, Jitter2,xlab="Sex (self reported)",ylab="Sex(predicted)",xaxt="n",yaxt="n")

axis(2,c(1,2),c("Female", "Male"))

axis(1,c(1,2),c("Female", "Male"))

idx <- as.factor(pd[,"Sex"])==as.factor(predictedSex) #Change to your own vector of self-reported sex if pd$Sex is not available.

Outliers <- pd[!idx, “Sample.ID”] #mark IDs with mis-matched gender information

# text(Jitter1[!idx],Jitter2[!idx]-0.05,Outliers,cex=0.5) #If included, this line will add IDs to the mis-matched individuals. However, if there are too many outliers, the figure might look a bit messy.

dev.off()

# This figure shows one individual with wrong sex information to be removed from downstream analyses. The IDs of individuals with wrongly assigned sex information were registered into the variable ‘Outliers’.

predicted_gender.pdf

#####Removing outlier individuals####

##We have listed outlier individuals based on gender discrepancy (graph above) and qcReport in variable ‘Outliers’, and we have also marked outlier individuals outside of 3 standard deviation (3SD) in variable ‘RM’.

##We also need to remove methylation control samples. If a ‘Note’ column in the Sample sheet provides such information, we can use the following code to mark these control individuals:

MC <- pd[,"Notes"] == "Methylation control" #mark control individuals as TRUE

#Change the highlighted portion to the corresponding column name and identifier in your local Sample sheet .csv file that suits the type of samples you want to remove.

##However it could be the case that the ‘Note’ column does not exist, and therefore one needs to remove Methylation controls based on the ‘Sample.ID’ column, for example, if we know all control individuals share the same string ‘Meth’, we could use the following code:

Index <- grep("Meth", pd[,"Sample.ID"]) #find individuals as methylation control

MC <- rep(FALSE, length(pd[,"Sample.ID"])) #generate an all FALSE variable with length of sample size

MC[Index] <- TRUE #mark control individuals as TRUE

#Change the highlighted portion to the corresponding identifier in your local Sample sheet .csv file.

##Mark and remove all outliers:

RM <- RM|(pd[,"Sample.ID"]%in%Outliers) #combine outliers marked in RM and Outlier

RM <- RM|MC #merge outliers info from MC

pd = pd[-which(RM),] #remove outliers

RGset = read.metharray.exp(base = datadir, targets=pd, verbose=TRUE) #reload data without outliers

save(RGset, file="./RGset.rda") #save raw data with outliers removed

###if samples have been processed in batches, each batch can be assessed separately as above, but the data should be merged before further preprocessing. Example for merging 2 waves of data, and having been saved as ‘RGset1’ and ‘RGset2’ through save(RGset1, file="./RGset1.rda") and save(RGset2, file="./RGset2.rda"), respectively.

p1 <- pData(RGset1)

p2 <- pData(RGset2)

Match <- match(colnames(p2), colnames(p1)) #match columns between p2 and p1

p1 <- p1[,Match[!is.na(Match)]] #in p1, only keep columns where both p1 and p2 have values.

Match <- match(colnames(p1), colnames(p2)) #match columns between p1 and p2

p2 <- p2[,Match[!is.na(Match)]] # in p2, only keep columns where both p1 and p2 have values.

stopifnot(identical(colnames(p1),colnames(p2))) #compare if p1 and p2 have the same column names.

pd <- rbind(p1,p2) #merge p1 and p2 to form the combined phenotype file.

RGset = read.metharray.exp(targets=pd, verbose=TRUE) #reload databased on combined Sample sheet information.

save(RGset, file="./RGset.rda") #save combined RGsets after QC.

################################################

### Preprocessing (Quantile normalization) after initial QC ####

################################################

#We have used the preprocessQuantile function that implements stratified quantile normalization preprocessing for Illumina methylation microarrays (Epigenomics 4, 325-341 (2012). The algorithm used for such a normalization method relies on the assumptions necessary for quantile normalization to be applicable and thus is not recommended for cases where global changes are expected, such as in cancer–normal comparisons. When large-scale differences in methylation are expected (cancer/normal studies or between tissue studies) the functional normalization algorithm may be preferable. Our experience shows that using quantile normalization procedure for blood samples without inflammation disorders works well. Subset-quantile within array normalization (SWAN) is another method that worked quite well and could be recommended, but quantile normalization is more popular.

# Low-quality samples that normalization cannot correct are automatically filtered out using the removeBadSamples argument in the preprocessQuantile function.

##Running quantile normalization

object=preprocessQuantile(RGset, fixOutliers = TRUE, removeBadSamples = TRUE, badSampleCutoff = 10.5, quantileNormalize = TRUE, stratified = TRUE, mergeManifest = FALSE, sex = NULL, verbose = TRUE) #Note this step removes samples with low intensity

save(object, file="./Quan-norm.rda") #Saving the normalized data to a new file

## Get beta values of normalized data

beta <- getBeta(object) #Gets the normalized Beta Value from normalized data

keepIndex=which(!seqnames(object)%in%c("chrX","chrY")) #mark probles on X and Y chromosome

beta <- beta[keepIndex,] #only keep probes on autosomes

save(beta, file="./Beta\_Quantile.rda") #saves the normalized beta values to new file

## Generate PCA to control for unknown structure ####

library(corpcor)

ss <- fast.svd(beta,tol=0) # this may take a while

save(ss,file="./fast\_svd.rda") #saves the output as a new file to be used as covariate in downstream analyses

### Estimate Cell-type Proportions ####

fileName <- "./cellcount.rda" #Output file name

if (file.exists(fileName)) {

load(fileName) #load existing output. Otherwise generate a new output

} else {

cellcount <- estimateCellCounts(RGset) #estimate cell count

save(cellcount, file=fileName) #save output

}

################################

#######QC plots after preprocessing######

################################

Cohort <- "Cohort Name" #Please change to your own cohort name.

mydist=dist(t(beta[sample(1:nrow(beta),10000),]))

mds=cmdscale(mydist)

pd=pData(dat)

pdf(paste("./figs/Batch\_effect\_Quantile\_Normalised\_mds\_",Cohort, ".pdf",sep=""),width=11,height=4)

plot(mds[,1],col=as.numeric(as.factor(pd[,"Slide"]))+1,xlab="",ylab="First Principal component",xaxt="n") #highlighted can be changed to other possible batch effects

dev.off()

###Pre-processed data for further analysis#####

"./RGset.rda" #RGset after QC

"./Quan-norm.rda" #Quantile normalized methylation data

"./fast\_svd.rda" #Principle components of beta value

"./cellcount.rda" #Estimated cell-type proportion