Pre-processing, normalisation and differential expression analysis

#All analysis is undertaken in the R programming environment.

#Code assumes that the relevant packages have been obtained (SEE TABLE XX). Each package has many user-defined options and readers are referred to the manuals associated with each package. Code in grey script is presented as an alternative strategy and was not run for the published analysis.

#Raw data files are available from the ArrayExpress database (www.ebi.ac.uk/arrayexpress) using the accession code: **E-MTAB-4800**.

#Series of 40 microarrays are read individually. Files contain raw **B**ead **L**evel **D**ata (BLData).

#Sample abbreviations

#dC – monolayer (dedifferentiated chondrocytes)

#dT – monolayer (dedifferentiated tenocytes)

#dF – monolayer (dedifferentiated fibroblasts)

#nC – native cartilage (whole tissue)

#nT – native tendon (whole tissue)

#ALG – cells encapsulated in alginate beads

#FIB – cells retained in fibrin-based tensional cultures

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

##PRE-PROCESSING#####################

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

setwd(‘/path/to/raw/data/files/’)

library(beadarray)

library(illuminaRatv1.db) #annotation package

BLData.arrayName=readIllumina(

useImages=FALSE,illuminaAnnotation="Ratv1") #change name for each file

#save all loaded raw data text files

#check what has been read

slotNames(BLData. BLData.arrayName)

#first ten rows of data from all columns

BLData.arrayName [[1]][1:10,]

#boxplot data

boxplot(BLData.arrayName,las=2,outline=FALSE,ylim=c(4,12))

#apply BASH algorithm – this takes a while for each array!

BLData.arrayName.bsh =BASH(BLData.arrayName,array=1,useLocs=FALSE)

#save a separate file of only .bsh files so that these may be accessed. Remove all raw files prior to saving .bsh files

rm(BLData.arrayName,…)

#set weights derived from BASH assessment of beads. This needs to be done for each array individually

BLData.arrayName=setWeights(

BLData.arrayName,wts=BLData.arrayName.bsh$wts,array=1,

combine=FALSE,wtName='wts')

#add quality information from BASH to bead level data for each array individually BLData.arrayName=insertSectionData(BLData.arrayName,what="BASHQC",data=BLData.arrayName.bsh$QC)

#check that extended score have been added

BLData.arrayName@sectionData

#plots positive control for housekeeping or biotin

poscontPlot(BLData.arrayName)

png('controlplots.png')

#a general control plot for all data

combinedControlPlot(BLData.arrayName)

dev.off()

#combine all arrays in a single expression set – this has to be done one-by-one in beadarray BLData=combine(BLData.arrayName1,BLData.arrayName2)

BLData=combine(BLData,BLData.arrayName3)

#continue to append the other arrays to BLData object

#Summarise probe data

myMean=function(x)mean(x,na.rm=TRUE)

mySd=function(x)sd(x,na.rm=TRUE) greenChannel=new("illuminaChannel",greenChannelTransform,illuminaOutlierMethod,myMedian,myMad,"Grn")

BSData=summarize(BLData,list(greenChannel),

useSampleFac=TRUE,sampleFac=rep(1:36,each=1),

weightNames="wts",removeUnMappedProbes=TRUE);

det=calculateDetection(BSData,status=fData(BSData)$Status,

negativeLabel="negative")

Detection(BSData)=det

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

###NORMALISATION#######

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

#Transform and normalise across arrays. Both quantile and loess strategies are shown

#QUANTILE

#BSData.q=normaliseIllumina(BSData,method="quantile",transform="log2")

#LOESS

library(limma)

BSData=normaliseIllumina(

BSData,method="none",transform="log2")

BSData.loess<-normalizeCyclicLoess(exprs(BSData),

weights = NULL, span=0.7, iterations = 3, method = "affy") ##no longer an eSet, just a normalised matrix

write.csv(BSData.loess,file=”BSData\_loess.csv”)

#filter probes to retain high quality probes

BSData.genes=BSData.q[

which(fData(BSData)$Status=="regular"), ]

expressed=apply(Detection(BSData.genes)<0.05,1,any)

BSData.filt=BSData.genes[expressed,]

ID=as.character(featureNames(BSData.q))

#addFeatureData

qual=unlist(mget(

ID,illuminaRatv1PROBEQUALITY,ifnotfound=NA))

table(qual)

rem<- qual == "No match" | qual == "Bad" | is.na(qual) #vector of probes to be removed

BSData.filt=BSData.q[!rem, ]

dim(BSData.filt)

###export BSData.filt to WGCNA and GOstats

#Matrix design for differential expression analysis for 36 arrays. Assign abbreviated names to each array based on sample group and replicate

design<model.matrix(~0+factor(

c(1,1,1,1,2,2,3,3,3,3,3,3,3,3,4,4,4,5,5,6,6,6,6,6,6,6,6,7,7,7,7,8,8,8,8,8)))

colnames(design)<-c("dC\_ALG","dC\_FIB","dC","dF","dT\_FIB","dT","nC","nT")

rownames(design)<c("dC\_ALG3","dC\_ALG4","dC\_ALG1","dC\_ALG2","dC\_FIB1","dC\_FIB2","dC1","dC2","dC3","dC4","dC5","dC6","dC7","dC8","dF1","dF2","dF3","dT\_FIB1","dT\_FIB2","dT1","dT2","dT3","dT4","dT5","dT6","dT7","dT8","nC2","nC3","nC4","nC5","nT1","nT2","nT3","nT4","nT5")

###Differential expression and feature data for **loess** normalised matrix

ID<-rownames(BSData.loess)

symbol=mget(ID,illuminaRatv1SYMBOL,ifnotfound=NA)

genename=mget(ID,illuminaRatv1GENENAME,ifnotfound=NA)

entrezID=mget(ID,illuminaRatv1ENTREZID,ifnotfound=NA)

anno=data.frame(Illumina\_ID=ID,Symbol=as.character(symbol),EntrezID=as.numeric(entrezID),

GeneName=as.character(genename))

fit<-lmFit(ajm.loess,design)

contrast.matrix<-makeContrasts(dC-dT,levels=design)

#set up the matrix and then you can include or exclude the samples that you want

fit2<-contrasts.fit(fit,contrast.matrix)

fit2<-eBayes(fit2)

fit2$gene=anno

rankCresults=topTable(

fit2,coef=1,number=1500,lfc=1.4,adjust.method="fdr",

sort.by="logFC",genelist=fit2$gene)

#export results to working directory in ranked format

write.table(rankCresults,file="rankCresults.txt",sep="\t")

###########

aw=arrayWeights(exprs(BSData.filt),design)

#Differential expression analysis for **quantile** normalised data

fit<-lmFit(exprs(BSData.filt),design, weights=aw)

ID=featureNames(BSData.filt)

chr=mget(ID,illuminaRatv1CHR,ifnotfound=NA)

refseq=mget(ID,illuminaRatv1REFSEQ,ifnotfound=NA)

entrezID=mget(ID,illuminaRatv1ENTREZID,ifnotfound=NA)

symbol=mget(ID,illuminaRatv1SYMBOL,ifnotfound=NA)

genename=mget(ID,illuminaRatv1GENENAME,ifnotfound=NA)

probequality=mget(ID,illuminaRatv1PROBEQUALITY,

ifnotfound=NA)

GO=mget(ID,illuminaRatv1GO,ifnotfound=NA)

anno=data.frame(

Illumina\_ID=ID,Chr=as.character(chr),

RefSeq=as.character(refseq),

EntrezID=as.numeric(entrezID),

Symbol=as.character(symbol),

GeneName=as.character(genename),

ProbeQuality=as.character(probequality),

GOterm=as.character(GO))

#linear model fit and contrast matrix

fit<-lmFit(exprs(BSData.filt),design)

contrast.matrix<-makeContrasts(dC\_ALG-dC,levels=design)

#set up the matrix and then you can include or exclude the samples that you want

fit2<-contrasts.fit(fit,contrast.matrix)

fit2<-eBayes(fit2)

fit2$gene=anno

rankCresults=topTable(

fit2,coef=1,number=1500,lfc=1.4,adjust.method="fdr",

sort.by="logFC",genelist=fit2$gene)

write.table(rankCresults,file="rankCresults.txt",sep="\t")

#Hierarchical clustering of quantile normalised data

d=dist(t(exprs(BSData.q)))

plot(hclust(d))

Hierarchical clustering and heatmap

setwd("/Users/XXX")

data<-read.csv("BSData\_loess.csv",header=TRUE)

colnames(data)[1]<-'IlluminaID'

ArrayName=names(data.frame(data[,-1]))

GeneName=data$EntrezID

exprs=data.frame(t(data[,-1]))

names(exprs)=data[,1]

dimnames(exprs)[[1]]=names(data.frame(data[,-1]))

exprs.v=as.vector(apply(as.matrix(exprs),2,var,na.rm=T))

keep=exprs.v>0.8

#Keep the genes showing the greatest evidence for co-expression

library(WGCNA)

filt=exprs[,keep]

GeneName=GeneName[keep]

ADJ1=abs(cor(filt,use="p"))^9 #create adjacency matrix

k=as.vector(apply(ADJ1,2,sum, na.rm=T))

datExpr=filt[, rank(-k,ties.method="first" )<=500]

rename<-t(datExpr)

colnames(rename)<-c(rep("3D",6),rep("2D",11),rep("3D",2),rep("2D",8),

rep("Cartilage",4),rep("Tendon",5))

map<-as.matrix(rename)

#Define and export the heatmap groups

library(gplots)

library(RColorBrewer)

hm <- heatmap.2(map)

hc <- as.hclust(hm$rowDendrogram)

#define the height at which the dendrogram is cut

groups<-cutree(hc, h=25) [hc$order]

names<-names(groups)

groups1<-unname(groups)

groups2<-data.frame("Symbol"=names,"Groups"=groups1)

write.csv(groups2,file="heatmapGroups.csv",row.names=FALSE)

##Create heatmap with the row groups and columns colour-coded

groups<-cutree(hc,h=25)

cols <- brewer.pal(max(groups), "Set3")

setwd(“”)

pdf(file = "Illumina\_heatmap2.pdf", width= 8,

height = 8,useDingbats=F)

par(oma=c(2,2,2,2))

heatmap.2(map,scale="row",col=greenred(100),

colsep=c(4,9,17),sepcolor="white",sepwidth=c(0.1,0.1),

trace="none",density.info="none",RowSideColors=cols[groups],

ColSideColors=c(rep("firebrick1",6),rep("midnightblue",11),

rep("firebrick1",2),rep("midnightblue",8),

rep("lightsteelblue3",4),rep("goldenrod2",5)),

cexRow=0.07,cexCol=1)

dev.off()

##retain ‘map’, a matrix of gene expression values, for PCA

Principal Component Analysis

#PCA for 36 arrays filtered on covariance

library(FactoMineR)

library(RColorBrewer)

#re-order columns from heatmap matrix so that they lie: 2D, #3D, native

map2<-map[,c(7,8,9,10,11,12,13,14,15,16,17,20,21,22,23,24,25,26,27,1,2,3,4,5,6,18,19,28,29,30,31,32,33,34,35,36)];

res.pca<-PCA(t(map2),graph=FALSE,axes=c(1,2))

PC1 <- res.pca$ind$coord[,1]

PC2 <- res.pca$ind$coord[,2]

#define factors

cell.type<-c("chondrocytes", "chondrocytes", "chondrocytes", "chondrocytes", "chondrocytes", "chondrocytes", "chondrocytes", "chondrocytes", "fibroblasts" ,"fibroblasts", "fibroblasts" ,"tenocytes", "tenocytes", "tenocytes", "tenocytes", "tenocytes", "tenocytes","tenocytes", "tenocytes","chondrocytes","chondrocytes", "chondrocytes", "chondrocytes","chondrocytes","chondrocytes", "tenocytes", "tenocytes","chondrocytes","chondrocytes", "chondrocytes", "chondrocytes", "tenocytes","tenocytes","tenocytes", "tenocytes", "tenocytes")

cell.type<-as.data.frame(cell.type)

condition<-c(rep("monolayer",19),rep("model.3D",8),rep("cartilage",4),rep("tendon",5))

condition<-as.data.frame(condition)

PCs <- data.frame(cbind(PC1,PC2,cell.type,condition))

PCA.comp1<-res.pca$eig[1,2]

PCA.comp2<-res.pca$eig[2,2]

#Colours for plot

mypalette<-c("gray0","gray88","gray64","gray40")

#Prepare and export PCA plot

library(ggplot2)

setwd("/Users/XXX")

pdf(file = "Illumina\_PCA\_Figure.pdf", width=8, height=8,useDingbats=F)

par(mar=c(1,1,1,1))

p<-ggplot(PCs)

p<-p+geom\_point(aes(PC1,PC2,color=condition,shape=cell.type),

size=6,alpha=0.6)+

scale\_colour\_manual(values=mypalette)+

labs(list(x=sprintf("PC1(%.1f%%)",PCA.comp1),

y=sprintf("PC2(%.1f%%)",PCA.comp2)))+

theme\_minimal(base\_size=10,base\_family="Helvetica")+

theme(legend.position = c(.85,.7),text = element\_text(size=12),plot.title=element\_text(

lineheight=.8,face="bold"))+

ggtitle("Principal Component Analysis")+

scale\_shape\_discrete(solid=T)

p

dev.off()

Hypergeometric testing of gene ontology analysis

#####Define the universe – all the genes on the Illumina microarray. Remove duplicate entries from Entrez annotations

##Whole chip

setwd("/Users/… ")

universe<-read.csv("RatRefv1.csv",header=TRUE,sep=",",as.is=TRUE)

rem.dups<-universe[!duplicated(universe$Entrez\_Gene\_ID),]

universe.entrez<-as.vector(rem.dups$Entrez\_Gene\_ID)

length(universe.entrez)

table(is.na(universe.entrez))

rem.universe<-universe.entrez=="NA"|is.na(universe.entrez)

filt<-universe.entrez[!rem.universe]

UNIVERSE<-as.numeric(filt)

#setwd("/Users/……. ")

#read in differential expression lists of genes by Entrez ID to functionally annotate

nt.up<-read.csv("working.csv",header=TRUE)

nt.entrez<-nt.up$Entrez

rem.NA<-nt.entrez=="NA"|is.na(nt.entrez)

table(rem.NA)

#filt<-nt.entrez[!rem.NA]

dups<-duplicated(nt.entrez)

table(dups)

no.dups=nt.entrez[!dups]

nt.final<-as.numeric(no.dups)

library(illuminaRatv1.db)

library(GOstats)

hgCutoff <- 0.001 #statistical cut-off

#Perform each in turn for biological process, cellular compartment and metabolic function. Change name of output file on each occasion.

params <- new("GOHyperGParams",geneIds=nt.final,universeGeneIds=UNIVERSE,annotation="illuminaRatv1.db",ontology="BP",pvalueCutoff=hgCutoff,conditional=FALSE,testDirection="over")

#Metabolic function annotation

#params <- new("GOHyperGParams",geneIds=nt.final,universeGeneIds=UNIVERSE,annotation="illuminaRatv1.db",ontology="**MF**",pvalueCutoff=hgCutoff,conditional=FALSE,testDirection="over")

#Cellular compartment annotation

#params <- new("GOHyperGParams",geneIds=nt.final,universeGeneIds=UNIVERSE,annotation="illuminaRatv1.db",ontology="**CC**",pvalueCutoff=hgCutoff,conditional=FALSE,testDirection="over")

hgOver <- hyperGTest(params)

df=summary(hgOver,htmlLinks=FALSE) #TRUE returns links to AmiGO

hgOver

p.value<-df$Pvalue

adjusted.p<-p.adjust(p.value,method="fdr")

df$adj.Pvalue<-adjusted.p

write.csv(file='GO.csv',df,row.names=FALSE)

SPIA pathway topology analysis

#Requires XML files to be downloaded from KEGG and stored within a named folder within the same directory;

library(SPIA)

setwd("/Volumes/XXX/SPIA")

makeSPIAdata(kgml.path="/Volumes/XXX/kegg",organism="rno",out.path="/Volumes/XXX/kegg")

#read in lists of differentially expressed genes as Entrez IDs

top<-read.csv("SPIA.csv",header=TRUE,sep=",")

setwd("/Users/XXX")

#create universe based on microarray probes

universe<-read.csv("RatRefv1.csv",header=TRUE,sep=",",as.is=TRUE)

#ensure that everything in universe is found in top

merged<-merge(top,universe,by.x<-"EntrezID",by.y="Entrez\_Gene\_ID")

dim(merged)

#[1] 2007 37

top<-merged[!duplicated(merged$EntrezID),]

dim(top)

#[1] 1842 37

top<-top[top$adj.P.Val<0.01,]

dim(top)

#[1] 1658 37

de<-as.vector(top$log2FC)

names(de)<-as.vector(top$EntrezID)

head(de)

dim(universe)

#[1] 23405 28

rem.dups<-universe[!duplicated(universe$Entrez\_Gene\_ID),]

dim(rem.dups)

#[1] 21494 28

universe.entrez<-as.vector(rem.dups$Entrez\_Gene\_ID)

#The SPIA algorithm takes as input the two vectors above and produces a table of pathways ranked from the most to the least significant.

res<-spia(de=de, all=universe.entrez, organism="rno",data.dir="/Volumes/XXX/kegg/",nB=2000,

plots=FALSE)

#show first 15 pathways, omit KEGG links

res[1:20,-12]

plotP(res,threshold=0.05)

setwd("/Users/XXX/SPIA\_pathways")

pdf("SPIA\_pathways\_SPIA.pdf")

plotP(res,threshold=0.05)

dev.off()

results<-res[1:20,]

write.table(results,file="SPIA\_pathways\_SPIA.txt")

[END]