This script has been used in EdgeR v 3.12

The dataset consists of 2 cichlid fish species (0. nasuta, O. ventralis) from which 6 parts of the brain had been sequenced over 5 biological replicates. This provides us with 60 samples.

We address two hypotheses:

H0: there is no DE of genes between a brainpart and the average expression in the five other brainparts within each species

H0: there is no difference in expression pattern across brainparts between the two species

We generate a contrastmatrix for each of these two hypotheses.

```{R,echo=FALSE}

library(edgeR)

library(gplots)

library(RColorBrewer)

library(stageR)

library(limma)

library(VennDiagram)

```

# Read in data

sessionInfo()

```{r}

dir="/Users/sderycke/Documents/Genbas/RNA/Control\_Females/OniloticusUMD1\_103/O\_niloticus/count\_files/"

files=list.files(dir)

files=files[grep(x=files,pattern="^Co")]

d=readDGE(paste0(dir,files), nrows=36499,header=FALSE)

sampleNames=unlist(lapply(strsplit(colnames(d),split="/"),function(x) x[11]))

species=factor(substr(sampleNames,4,5))

brain=factor(unlist(lapply(strsplit(sampleNames,split="\_"),function(x) x[2])))

specimen=as.numeric(substr(unlist(lapply(strsplit(sampleNames,split="\_"),function(x) x[3])),1,1))

specimen[31:60]=specimen[31:60]+5

specimen=factor(specimen)

colnames(d)=unlist(lapply(strsplit(sampleNames,split=".",fixed=TRUE),function(x) x[1]))

```

# Remove outlying sample

One sample (Co1Na\_OB\_5) has an extremely low library size: the olfactory bulb from the fifth specimen of the O. nasuta species. We remove this sample, and accordingly change the metadata.

```{r}

barplot(d$samples$lib.size/1e6, ylab="Library size (millions)", names.arg = colnames(d), cex.names = 0.4, las=2)

keepSample=!colnames(d)=="Co1Na\_OB\_5"

species=species[keepSample]

brain=brain[keepSample]

specimen=specimen[keepSample]

d=d[,keepSample]

dim(d)

```

# Filtering, normalization, exploratory data analysis

We filter genes that do not have a CPM of 15 in at least 4 samples. We picked 4 samples because this is the lowest number of samples for any treatment group (brain). For the lowest library size of 137528, a CPM of 15 corresponds to a count of (15/1e6)\*137528 = ca 2 in the lowest sample.After this filter, 11577 genes are kept.

```{r}

min(d$samples$lib.size)

keep <- rowSums(cpm(d)>15)>=4

table(keep)

d=d[keep,,keep.lib.sizes=FALSE]

barplot(d$samples$lib.size/1e6, ylab="Library size (millions)", names.arg = colnames(d), cex.names = 0.4, las=2)

d=calcNormFactors(d)

```

## Gene expression data exploration

to count the number of genes expressed per sample, shared between all samples and occurring in only one sample;

```{r}

NumberGenes = colSums(d$counts != 0)

median(NumberGenes)

setEPS()

postscript(paste("Number of Genes.eps", sep = ""))

barplot(NumberGenes, ylab="Number of genes", ylim = c(0, 12000),names.arg = colnames(NumberGenes), cex.names = 0.4, las=2, )

dev.off()

NumberGenesShared = as.data.frame(rowSums(d$counts != 0))

sum(NumberGenesShared == 59) #7224

sum(NumberGenesShared == 1) #0

```

##The MDS plot clearly shows that the samples cluster by brain, suggesting the largest variability in expression can be explained by brain part.

```{r}

setEPS()

pch = c(8,19)

cols = c("firebrick", "skyblue", "pink3", "darkseagreen", "darkgoldenrod", "mediumpurple")

postscript(paste("MDSplot.eps", sep = ""))

plotMDS(d, pch = pch[species] , col= cols[as.double(brain)], cex = 1.5)

legend('topright', legend = c("O. nasuta", "O. ventralis", "BS", "CE", "DI", "OB", "OT", "TE"), pch = c(8,19,1,1,1,1,1,1), col=c("black", "black", "white", "white", "white", "white", "white", "white"), text.col = c("black", "black", "firebrick", "skyblue", "pink3", "darkseagreen", "darkgoldenrod", "mediumpurple"), cex = 0.8)

dev.off()

```

# Differential expression analysis

We use the Negative binomial model which takes care of multiple factors by fitting generalized linear models (GLM) with a design matrix. We first have to create the design matrix based on our samples.

## Design matrix

We have specimen effects (brainparts within a specimen are linked to each other) and brainpart effects. We first make a matrix with the parameters for the specimens, and then we make a matrix with the parameters for the brainparts.Columns are specimens and brainparts, rows are the samples. Both matrices are then merged to generate the design matrix

```{r}

specimenHlp=model.matrix(~-1+specimen)

colnames(specimenHlp)=paste0("s",1:10)

rownames(specimenHlp)=colnames(d)

specimenHlp

brainSpecies=factor(paste(brain,species,sep="\_"))

brainHlp=model.matrix(~-1+brainSpecies)

colnames(brainHlp)=levels(brainSpecies)

brainHlp

```

#We have separate specimen effect so we can only estimate 4 brain parameters within species. We drop first two brain parameters, BS in Na and BS in Ve. The intercept in our model will be the average expression in BS over the specimens (5 for Na, 5 for Ve). Note, that we can estimate the BS effect in Ve if we take the average over all Ve specimen parameters.

```{r}

design\_2way= cbind(specimenHlp,brainHlp[,-(1:2)])

design\_2way

```

## Dispersion estimation and GLM fitting

The negative binomial dispersion is estimated using the estimateDisp function. This returns the DGEList object with additional entries for the estimated NB dispersions for all genes. These estimates include the common, trended and tagwise dispersion estimates, which can be visualized with plotBCV, which shows the root-estimate, i.e., the biological coefficient of variation for each gene.

```{r}

d=estimateDisp(d,design\_2way,robust=TRUE)

d$common.dispersion

plotBCV(d)

fit\_2way = glmQLFit(d, design\_2way, robust=TRUE)

head(fit\_2way$coefficients)

plotQLDisp(fit\_2way)

```

## III.4. Testing for differential expression

## III.4.2. Hypothesis 1: There is no differential expression between a brain part and the average of all other brain parts in O. nasuta and O. ventralis.

### III.4.2.1. Screening stage

We will first make a contrast matrix containing all contrasts of interest.The rows in the contrastmatrix are the colnames of the designmatrix, and the columns are the contrasts of interest.

Then we use this matrix to perform the screening step, testing whether any of the contrasts of interest are false.

In the first block of code we build a contrast matrix that involves all contrasts within one species. Note, that the average expression of the BS brain part for a species is obtained by averaging the specimen effects for the specimens belonging to that species.

```{r}

## all contrasts within one species.

LBrain=matrix(0,nrow=ncol(fit\_2way$coefficients),ncol=12)

rownames(LBrain)=colnames(fit\_2way$coefficients)

colnames(LBrain)=c("BS-avg\_Na","BS-avg\_Ve",

 "CE-avg\_Na","CE-avg\_Ve",

 "DI-avg\_Na","DI-avg\_Ve",

 "OB-avg\_Na","OB-avg\_Ve",

 "OT-avg\_Na","OT-avg\_Ve",

 "TE-avg\_Na","TE-avg\_Ve")

#O. nasuta

LBrain[c("CE\_Na","DI\_Na","OB\_Na","OT\_Na","TE\_Na"),"BS-avg\_Na"]=-1/5

LBrain[c("CE\_Na","DI\_Na","OB\_Na","OT\_Na","TE\_Na"),"CE-avg\_Na"]=c(1,rep(-1/5,4))

LBrain[c("DI\_Na","CE\_Na","OB\_Na","OT\_Na","TE\_Na"),"DI-avg\_Na"]=c(1,rep(-1/5,4))

LBrain[c("OB\_Na","CE\_Na","DI\_Na","OT\_Na","TE\_Na"),"OB-avg\_Na"]=c(1,rep(-1/5,4))

LBrain[c("OT\_Na","CE\_Na","DI\_Na","OB\_Na","TE\_Na"),"OT-avg\_Na"]=c(1,rep(-1/5,4))

LBrain[c("TE\_Na","CE\_Na","DI\_Na","OB\_Na","OT\_Na"),"TE-avg\_Na"]=c(1,rep(-1/5,4))

#O. ventralis

LBrain[c("CE\_Ve","DI\_Ve","OB\_Ve","OT\_Ve","TE\_Ve"),"BS-avg\_Ve"]=-1/5

LBrain[c("CE\_Ve","DI\_Ve","OB\_Ve","OT\_Ve","TE\_Ve"),"CE-avg\_Ve"]=c(1,rep(-1/5,4))

LBrain[c("DI\_Ve","CE\_Ve","OB\_Ve","OT\_Ve","TE\_Ve"),"DI-avg\_Ve"]=c(1,rep(-1/5,4))

LBrain[c("OB\_Ve","CE\_Ve","DI\_Ve","OT\_Ve","TE\_Ve"),"OB-avg\_Ve"]=c(1,rep(-1/5,4))

LBrain[c("OT\_Ve","CE\_Ve","DI\_Ve","OB\_Ve","TE\_Ve"),"OT-avg\_Ve"]=c(1,rep(-1/5,4))

LBrain[c("TE\_Ve","CE\_Ve","DI\_Ve","OB\_Ve","OT\_Ve"),"TE-avg\_Ve"]=c(1,rep(-1/5,4))

```

#In the screening stage, a global Ftest across all contrasts is performed for each gene to see whether any of the contrasts is significant.This results in a vector pScreen with one pvalue per gene.WARNING: these pvalues are not corrected for multiple testing and can thus not be used!!

```{r}

alpha=0.05

screenTest = glmQLFTest(fit\_2way, contrast=LBrain)

pScreen <- screenTest$table$PValue

names(pScreen) <- rownames(screenTest$table)

```

#In the confirmation stage, the Ftest is performed for each contrast. Pvalues generated here are then corrected with the holm method. This tests for genes with logFC > O

```{r}

confirmationResultsHyp1 <- sapply(1:ncol(LBrain),function(i) glmQLFTest(fit\_2way, contrast = LBrain[,i]), simplify=FALSE) #calculates Ftest for each contrast

confirmationPList <- lapply(confirmationResultsHyp1, function(x) x$table$PValue)#takes the p-values from all genes for each contrast and puts them in a list

confirmationP <- as.matrix(Reduce(f=cbind,confirmationPList))

rownames(confirmationP) <- rownames(confirmationResultsHyp1[[1]]$table)

colnames(confirmationP) <- colnames(LBrain)

stageRObj <- stageR(pScreen=pScreen, pConfirmation=confirmationP) #constructs an object

stageRAdj <- stageWiseAdjustment(object=stageRObj, method="holm", alpha=0.05) # adjusts the p-values using FWER correction; here we choose the holm method

resBrain <- getResults(stageRAdj)

resBrain

SignifGenesBrain = colSums(resBrain) #number of DE genes in every contrast.

SignifGenesBrain

adjusted\_p = getAdjustedPValues(stageRAdj) #p-values to report in manuscript.pAdjScreen = pvalue of the global test, other columns are then the posthoc tests

```

#To find unique DE genes for each brainpart and shared DE between brainparts (global), we first remove genes for which the global test was not significant.Then, we extract the names of the genes that were significant for each contrast.

```{r}

resbrain\_df = as.data.frame(resBrain) #resBrain is a matrix, to be able to subset, change it to a dataframe

resbrain\_df2 = resbrain\_df

resbrain\_df2$gene = rownames(resbrain\_df2)

OnlySignGenes = resbrain\_df[resbrain\_df$padjScreen == 1,] #removes rows for which global test was non significant

dim(OnlySignGenes) # 8748 significant after correction; still includes genes for which all posthoc tests were 0

setEPS()

postscript(paste("SignGenesBrainpart.eps", sep = ""))

barplot(as.matrix(OnlySignGenes), ylab="Number of significant genes", ylim = c(0,9000), names.arg = colnames(OnlySignGenes), cex.names = 0.8, las=2)

dev.off()

#counting unique values:

sum(OnlySignGenes$`BS-avg\_Na`== 1 & OnlySignGenes$`CE-avg\_Na` == 0 & OnlySignGenes$`DI-avg\_Na` == 0 & OnlySignGenes$`OB-avg\_Na` == 0 & OnlySignGenes$`OT-avg\_Na` == 0 & OnlySignGenes$`TE-avg\_Na`== 0) # 340

sum(OnlySignGenes$`BS-avg\_Na`== 0 & OnlySignGenes$`CE-avg\_Na` == 1 & OnlySignGenes$`DI-avg\_Na` == 0 & OnlySignGenes$`OB-avg\_Na` == 0 & OnlySignGenes$`OT-avg\_Na` == 0 & OnlySignGenes$`TE-avg\_Na`== 0) #806

sum(OnlySignGenes$`BS-avg\_Na`== 0 & OnlySignGenes$`CE-avg\_Na` == 0 & OnlySignGenes$`DI-avg\_Na` == 1 & OnlySignGenes$`OB-avg\_Na` == 0 & OnlySignGenes$`OT-avg\_Na` == 0 & OnlySignGenes$`TE-avg\_Na`== 0) #152

sum(OnlySignGenes$`BS-avg\_Na`== 0 & OnlySignGenes$`CE-avg\_Na` == 0 & OnlySignGenes$`DI-avg\_Na` == 0 & OnlySignGenes$`OB-avg\_Na` == 1 & OnlySignGenes$`OT-avg\_Na` == 0 & OnlySignGenes$`TE-avg\_Na`== 0) #236

sum(OnlySignGenes$`BS-avg\_Na`== 0 & OnlySignGenes$`CE-avg\_Na` == 0 & OnlySignGenes$`DI-avg\_Na` == 0 & OnlySignGenes$`OB-avg\_Na` == 0 & OnlySignGenes$`OT-avg\_Na` == 1 & OnlySignGenes$`TE-avg\_Na`== 0) #324

sum(OnlySignGenes$`BS-avg\_Na`== 0 & OnlySignGenes$`CE-avg\_Na` == 0 & OnlySignGenes$`DI-avg\_Na` == 0 & OnlySignGenes$`OB-avg\_Na` == 0 & OnlySignGenes$`OT-avg\_Na` == 0 & OnlySignGenes$`TE-avg\_Na`== 1) #410

sum(OnlySignGenes$`BS-avg\_Ve`== 1 & OnlySignGenes$`CE-avg\_Ve` == 0 & OnlySignGenes$`DI-avg\_Ve` == 0 & OnlySignGenes$`OB-avg\_Ve` == 0 & OnlySignGenes$`OT-avg\_Ve` == 0 & OnlySignGenes$`TE-avg\_Ve`== 0) # 291

sum(OnlySignGenes$`BS-avg\_Ve`== 0 & OnlySignGenes$`CE-avg\_Ve` == 1 & OnlySignGenes$`DI-avg\_Ve` == 0 & OnlySignGenes$`OB-avg\_Ve` == 0 & OnlySignGenes$`OT-avg\_Ve` == 0 & OnlySignGenes$`TE-avg\_Ve`== 0) #1004

sum(OnlySignGenes$`BS-avg\_Ve`== 0 & OnlySignGenes$`CE-avg\_Ve` == 0 & OnlySignGenes$`DI-avg\_Ve` == 1 & OnlySignGenes$`OB-avg\_Ve` == 0 & OnlySignGenes$`OT-avg\_Ve` == 0 & OnlySignGenes$`TE-avg\_Ve`== 0) #197

sum(OnlySignGenes$`BS-avg\_Ve`== 0 & OnlySignGenes$`CE-avg\_Ve` == 0 & OnlySignGenes$`DI-avg\_Ve` == 0 & OnlySignGenes$`OB-avg\_Ve` == 1 & OnlySignGenes$`OT-avg\_Ve` == 0 & OnlySignGenes$`TE-avg\_Ve`== 0) #349

sum(OnlySignGenes$`BS-avg\_Ve`== 0 & OnlySignGenes$`CE-avg\_Ve` == 0 & OnlySignGenes$`DI-avg\_Ve` == 0 & OnlySignGenes$`OB-avg\_Ve` == 0 & OnlySignGenes$`OT-avg\_Ve` == 1 & OnlySignGenes$`TE-avg\_Ve`== 0) #183

sum(OnlySignGenes$`BS-avg\_Ve`== 0 & OnlySignGenes$`CE-avg\_Ve` == 0 & OnlySignGenes$`DI-avg\_Ve` == 0 & OnlySignGenes$`OB-avg\_Ve` == 0 & OnlySignGenes$`OT-avg\_Ve` == 0 & OnlySignGenes$`TE-avg\_Ve`== 1) #370

```

# to make venn diagrams of shared DE genes between the species for each brainpart

```{r}

#to get names of all DE genes unique for each brainpart, and for all brainparts in nasuta:

BS\_avgNa = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Na`== 1 & OnlySignGenes$`CE-avg\_Na` == 0 & OnlySignGenes$`DI-avg\_Na` == 0 & OnlySignGenes$`OB-avg\_Na` == 0 & OnlySignGenes$`OT-avg\_Na` == 0 & OnlySignGenes$`TE-avg\_Na`== 0)))

CE\_avgNa = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Na`== 0 & OnlySignGenes$`CE-avg\_Na` == 1 & OnlySignGenes$`DI-avg\_Na` == 0 & OnlySignGenes$`OB-avg\_Na` == 0 & OnlySignGenes$`OT-avg\_Na` == 0 & OnlySignGenes$`TE-avg\_Na`== 0)))

DI\_avgNa = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Na`== 0 & OnlySignGenes$`CE-avg\_Na` == 0 & OnlySignGenes$`DI-avg\_Na` == 1 & OnlySignGenes$`OB-avg\_Na` == 0 & OnlySignGenes$`OT-avg\_Na` == 0 & OnlySignGenes$`TE-avg\_Na`== 0)))

OB\_avgNa = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Na`== 0 & OnlySignGenes$`CE-avg\_Na` == 0 & OnlySignGenes$`DI-avg\_Na` == 0 & OnlySignGenes$`OB-avg\_Na` == 1 & OnlySignGenes$`OT-avg\_Na` == 0 & OnlySignGenes$`TE-avg\_Na`== 0)))

OT\_avgNa = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Na`== 0 & OnlySignGenes$`CE-avg\_Na` == 0 & OnlySignGenes$`DI-avg\_Na` == 0 & OnlySignGenes$`OB-avg\_Na` == 0 & OnlySignGenes$`OT-avg\_Na` == 1 & OnlySignGenes$`TE-avg\_Na`== 0)))

TE\_avgNa = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Na`== 0 & OnlySignGenes$`CE-avg\_Na` == 0 & OnlySignGenes$`DI-avg\_Na` == 0 & OnlySignGenes$`OB-avg\_Na` == 0 & OnlySignGenes$`OT-avg\_Na` == 0 & OnlySignGenes$`TE-avg\_Na`== 1)))

allSignGenes\_Na = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Na`== 1 | OnlySignGenes$`CE-avg\_Na` == 1 | OnlySignGenes$`DI-avg\_Na` == 1 | OnlySignGenes$`OB-avg\_Na` == 1 | OnlySignGenes$`OT-avg\_Na` == 1 | OnlySignGenes$`TE-avg\_Na`== 1)))

#to get names of all DE genes unique for each brainpart, and for all brainparts in ventralis:

BS\_avgVe = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Ve`== 1 & OnlySignGenes$`CE-avg\_Ve` == 0 & OnlySignGenes$`DI-avg\_Ve` == 0 & OnlySignGenes$`OB-avg\_Ve` == 0 & OnlySignGenes$`OT-avg\_Ve` == 0 & OnlySignGenes$`TE-avg\_Ve`== 0)))

CE\_avgVe = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Ve`== 0 & OnlySignGenes$`CE-avg\_Ve` == 1 & OnlySignGenes$`DI-avg\_Ve` == 0 & OnlySignGenes$`OB-avg\_Ve` == 0 & OnlySignGenes$`OT-avg\_Ve` == 0 & OnlySignGenes$`TE-avg\_Ve`== 0)))

DI\_avgVe = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Ve`== 0 & OnlySignGenes$`CE-avg\_Ve` == 0 & OnlySignGenes$`DI-avg\_Ve` == 1 & OnlySignGenes$`OB-avg\_Ve` == 0 & OnlySignGenes$`OT-avg\_Ve` == 0 & OnlySignGenes$`TE-avg\_Ve`== 0)))

OB\_avgVe = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Ve`== 0 & OnlySignGenes$`CE-avg\_Ve` == 0 & OnlySignGenes$`DI-avg\_Ve` == 0 & OnlySignGenes$`OB-avg\_Ve` == 1 & OnlySignGenes$`OT-avg\_Ve` == 0 & OnlySignGenes$`TE-avg\_Ve`== 0)))

OT\_avgVe = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Ve`== 0 & OnlySignGenes$`CE-avg\_Ve` == 0 & OnlySignGenes$`DI-avg\_Ve` == 0 & OnlySignGenes$`OB-avg\_Ve` == 0 & OnlySignGenes$`OT-avg\_Ve` == 1 & OnlySignGenes$`TE-avg\_Ve`== 0)))

TE\_avgVe = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Ve`== 0 & OnlySignGenes$`CE-avg\_Ve` == 0 & OnlySignGenes$`DI-avg\_Ve` == 0 & OnlySignGenes$`OB-avg\_Ve` == 0 & OnlySignGenes$`OT-avg\_Ve` == 0 & OnlySignGenes$`TE-avg\_Ve`== 1)))

allSignGenes\_Ve = c(rownames (subset (OnlySignGenes, OnlySignGenes$`BS-avg\_Ve`== 1 | OnlySignGenes$`CE-avg\_Ve` == 1 | OnlySignGenes$`DI-avg\_Ve` == 1 | OnlySignGenes$`OB-avg\_Ve` == 1 | OnlySignGenes$`OT-avg\_Ve` == 1 | OnlySignGenes$`TE-avg\_Ve`== 1)))

NamesOnlySignGenes= names(which(resBrain[,"padjScreen"]==1))

NamesOnlySignGenes\_BS\_Na= names(which(resBrain[,"BS-avg\_Na"]==1))

NamesOnlySignGenes\_BS\_Ve= names(which(resBrain[,"BS-avg\_Ve"]==1))

NamesOnlySignGenes\_CE\_Na= names(which(resBrain[,"CE-avg\_Na"]==1))

NamesOnlySignGenes\_CE\_Ve= names(which(resBrain[,"CE-avg\_Ve"]==1))

NamesOnlySignGenes\_DI\_Na= names(which(resBrain[,"DI-avg\_Na"]==1))

NamesOnlySignGenes\_DI\_Ve= names(which(resBrain[,"DI-avg\_Ve"]==1))

NamesOnlySignGenes\_OB\_Na= names(which(resBrain[,"OB-avg\_Na"]==1))

NamesOnlySignGenes\_OB\_Ve= names(which(resBrain[,"OB-avg\_Ve"]==1))

NamesOnlySignGenes\_OT\_Na= names(which(resBrain[,"OT-avg\_Na"]==1))

NamesOnlySignGenes\_OT\_Ve= names(which(resBrain[,"OT-avg\_Ve"]==1))

NamesOnlySignGenes\_TE\_Na= names(which(resBrain[,"TE-avg\_Na"]==1))

NamesOnlySignGenes\_TE\_Ve= names(which(resBrain[,"TE-avg\_Ve"]==1))

write.table (NamesOnlySignGenes\_TE\_Ve, "NamesOnlySignGenes\_TE\_Ve.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_TE\_Na, "NamesOnlySignGenes\_TE\_Na.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_CE\_Ve, "NamesOnlySignGenes\_CE\_Ve.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_CE\_Na, "NamesOnlySignGenes\_CE\_Na.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_OB\_Ve, "NamesOnlySignGenes\_OB\_Ve.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_OB\_Na, "NamesOnlySignGenes\_OB\_Na.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_OT\_Ve, "NamesOnlySignGenes\_OT\_Ve.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_OT\_Na, "NamesOnlySignGenes\_OT\_Na.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_BS\_Ve, "NamesOnlySignGenes\_BS\_Ve.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_BS\_Na, "NamesOnlySignGenes\_BS\_Na.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_DI\_Ve, "NamesOnlySignGenes\_DI\_Ve.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

write.table (NamesOnlySignGenes\_DI\_Na, "NamesOnlySignGenes\_DI\_Na.txt", col.names = FALSE, row.names = FALSE, quote = FALSE)

pdf("venn\_BS.pdf")

venn\_BS <- venn.diagram(list(NamesOnlySignGenes\_BS\_Na, NamesOnlySignGenes\_BS\_Ve), NULL, alpha=c(0.5,0.5), cex = 2, cat.fontface=1, category.names=c("BS\_Na", "BS\_Ve"), cat.pos = c(0,0), cat.cex = c(2,2))

grid.draw(venn\_BS)

dev.off()

pdf("venn\_CE.pdf")

venn\_CE <- venn.diagram(list(NamesOnlySignGenes\_CE\_Na, NamesOnlySignGenes\_CE\_Ve), NULL, alpha=c(0.5,0.5), cex = 2, cat.fontface=1, category.names=c("CE\_Na", "CE\_Ve"),cat.pos = c(0,0), cat.cex = c(2,2), inverted = TRUE)

grid.draw(venn\_CE)

dev.off()

pdf("venn\_TE.pdf")

venn\_TE <- venn.diagram(list(NamesOnlySignGenes\_TE\_Na, NamesOnlySignGenes\_TE\_Ve), NULL, alpha=c(0.5,0.5), cex = 2, cat.fontface=1, category.names=c("TE\_Na", "TE\_Ve"), cat.pos = c(0,0), cat.cex = c(2,2))

grid.draw(venn\_TE)

dev.off()

pdf("venn\_OT.pdf")

venn\_OT <- venn.diagram(list(NamesOnlySignGenes\_OT\_Na, NamesOnlySignGenes\_OT\_Ve), NULL, alpha=c(0.5,0.5), cex = 2, cat.fontface=1, category.names=c("OT\_Na", "OT\_Ve"), cat.pos = c(0,0), cat.cex = c(2,2))

grid.draw(venn\_OT)

dev.off()

pdf("venn\_OB.pdf")

venn\_OB <- venn.diagram(list(NamesOnlySignGenes\_OB\_Na, NamesOnlySignGenes\_OB\_Ve), NULL, alpha=c(0.5,0.5), cex = 2, cat.fontface=1, category.names=c("OB\_Na", "OB\_Ve"), cat.pos = c(0,0), cat.cex = c(2,2), inverted = TRUE)

grid.draw(venn\_OB)

dev.off()

pdf("venn\_DI.pdf")

venn\_DI <- venn.diagram(list(NamesOnlySignGenes\_DI\_Na, NamesOnlySignGenes\_DI\_Ve), NULL, alpha=c(0.5,0.5), cex = 2, cat.fontface=1, category.names=c("DI\_Na", "DI\_Ve"), cat.pos = c(0,0), cat.cex = c(2,2), inverted = TRUE)

grid.draw(venn\_DI)

dev.off()

```

#to get fold changes for the significant gene(s); adjusted P values from stage wise testing are added and only the significant genes are kept;

```{r}

##BSNa

Hyp1SignGenesBSNa=names(which(resBrain[,"BS-avg\_Na"]==1))

logFC\_BSavgNa = confirmationResultsHyp1[[1]]$table[Hyp1SignGenesBSNa,] #adjust column to select correct contrast

BSNa\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesBSNa)

logFC\_BSavgNa\_adjP = merge(logFC\_BSavgNa,BSNa\_adjustedP, by = 0, all = TRUE)

logFC\_BSavgNa\_adjP = logFC\_BSavgNa\_adjP[, c(1,2,7)] #adjust column to select correct contrast

colnames(logFC\_BSavgNa\_adjP)[3] = "adjusted Pvalue"

##CENa

Hyp1SignGenesCENa=names(which(resBrain[,"CE-avg\_Na"]==1))

logFC\_CEavgNa = confirmationResultsHyp1[[3]]$table[Hyp1SignGenesCENa,]

CENa\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesCENa)

logFC\_CEavgNa\_adjP = merge(logFC\_CEavgNa,CENa\_adjustedP, by = 0, all = TRUE)

logFC\_CEavgNa\_adjP = logFC\_CEavgNa\_adjP[, c(1,2,9)]

colnames(logFC\_CEavgNa\_adjP)[3] = "adjusted Pvalue"

##DINa

Hyp1SignGenesDINa=names(which(resBrain[,"DI-avg\_Na"]==1))

logFC\_DIavgNa = confirmationResultsHyp1[[5]]$table[Hyp1SignGenesDINa,]

DINa\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesDINa)

logFC\_DIavgNa\_adjP = merge(logFC\_DIavgNa,DINa\_adjustedP, by = 0, all = TRUE)

logFC\_DIavgNa\_adjP = logFC\_DIavgNa\_adjP[, c(1,2,11)]

colnames(logFC\_DIavgNa\_adjP)[3] = "adjusted Pvalue"

##OBNa

Hyp1SignGenesOBNa=names(which(resBrain[,"OB-avg\_Na"]==1))

logFC\_OBavgNa = confirmationResultsHyp1[[7]]$table[Hyp1SignGenesOBNa,]

OBNa\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesOBNa)

logFC\_OBavgNa\_adjP = merge(logFC\_OBavgNa,OBNa\_adjustedP, by = 0, all = TRUE)

logFC\_OBavgNa\_adjP = logFC\_OBavgNa\_adjP[, c(1,2,13)]

colnames(logFC\_OBavgNa\_adjP)[3] = "adjusted Pvalue"

##OTNa

Hyp1SignGenesOTNa=names(which(resBrain[,"OT-avg\_Na"]==1))

logFC\_OTavgNa = confirmationResultsHyp1[[9]]$table[Hyp1SignGenesOTNa,]

OTNa\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesOTNa)

logFC\_OTavgNa\_adjP = merge(logFC\_OTavgNa,OTNa\_adjustedP, by = 0, all = TRUE)

logFC\_OTavgNa\_adjP = logFC\_OTavgNa\_adjP[, c(1,2,15)]

colnames(logFC\_OTavgNa\_adjP)[3] = "adjusted Pvalue"

##TENa

Hyp1SignGenesTENa=names(which(resBrain[,"TE-avg\_Na"]==1))

logFC\_TEavgNa = confirmationResultsHyp1[[11]]$table[Hyp1SignGenesTENa,]

TENa\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesTENa)

logFC\_TEavgNa\_adjP = merge(logFC\_TEavgNa,TENa\_adjustedP, by = 0, all = TRUE)

logFC\_TEavgNa\_adjP = logFC\_TEavgNa\_adjP[, c(1,2,17)]

colnames(logFC\_TEavgNa\_adjP)[3] = "adjusted Pvalue"

##BSVe

Hyp1SignGenesBSVe=names(which(resBrain[,"BS-avg\_Ve"]==1))

logFC\_BSavgVe = confirmationResultsHyp1[[2]]$table[Hyp1SignGenesBSVe,] #adjust column to select correct contrast

BSVe\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesBSVe)

logFC\_BSavgVe\_adjP = merge(logFC\_BSavgVe,BSVe\_adjustedP, by = 0, all = TRUE)

logFC\_BSavgVe\_adjP = logFC\_BSavgVe\_adjP[, c(1,2,8)] #adjust column to select correct contrast

colnames(logFC\_BSavgVe\_adjP)[3] = "adjusted Pvalue"

##CEVe

Hyp1SignGenesCEVe=names(which(resBrain[,"CE-avg\_Ve"]==1))

logFC\_CEavgVe = confirmationResultsHyp1[[4]]$table[Hyp1SignGenesCEVe,]

CEVe\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesCEVe)

logFC\_CEavgVe\_adjP = merge(logFC\_CEavgVe,CEVe\_adjustedP, by = 0, all = TRUE)

logFC\_CEavgVe\_adjP = logFC\_CEavgVe\_adjP[, c(1,2,10)]

colnames(logFC\_CEavgVe\_adjP)[3] = "adjusted Pvalue"

##DIVe

Hyp1SignGenesDIVe=names(which(resBrain[,"DI-avg\_Ve"]==1))

logFC\_DIavgVe = confirmationResultsHyp1[[6]]$table[Hyp1SignGenesDIVe,]

DIVe\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesDIVe)

logFC\_DIavgVe\_adjP = merge(logFC\_DIavgVe,DIVe\_adjustedP, by = 0, all = TRUE)

logFC\_DIavgVe\_adjP = logFC\_DIavgVe\_adjP[, c(1,2,12)]

colnames(logFC\_DIavgVe\_adjP)[3] = "adjusted Pvalue"

##OBVe

Hyp1SignGenesOBVe=names(which(resBrain[,"OB-avg\_Ve"]==1))

logFC\_OBavgVe = confirmationResultsHyp1[[8]]$table[Hyp1SignGenesOBVe,]

OBVe\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesOBVe)

logFC\_OBavgVe\_adjP = merge(logFC\_OBavgVe,OBVe\_adjustedP, by = 0, all = TRUE)

logFC\_OBavgVe\_adjP = logFC\_OBavgVe\_adjP[, c(1,2,14)]

colnames(logFC\_OBavgVe\_adjP)[3] = "adjusted Pvalue"

##OTVe

Hyp1SignGenesOTVe=names(which(resBrain[,"OT-avg\_Ve"]==1))

logFC\_OTavgVe = confirmationResultsHyp1[[10]]$table[Hyp1SignGenesOTVe,]

OTVe\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesOTVe)

logFC\_OTavgVe\_adjP = merge(logFC\_OTavgVe,OTVe\_adjustedP, by = 0, all = TRUE)

logFC\_OTavgVe\_adjP = logFC\_OTavgVe\_adjP[, c(1,2,16)]

colnames(logFC\_OTavgVe\_adjP)[3] = "adjusted Pvalue"

##TEVe

Hyp1SignGenesTEVe=names(which(resBrain[,"TE-avg\_Ve"]==1))

logFC\_TEavgVe = confirmationResultsHyp1[[12]]$table[Hyp1SignGenesTEVe,]

TEVe\_adjustedP = subset(adjusted\_p, rownames(adjusted\_p)%in%Hyp1SignGenesTEVe)

logFC\_TEavgVe\_adjP = merge(logFC\_TEavgVe,TEVe\_adjustedP, by = 0, all = TRUE)

logFC\_TEavgVe\_adjP = logFC\_TEavgVe\_adjP[, c(1,2,18)]

colnames(logFC\_TEavgVe\_adjP)[3] = "adjusted Pvalue"

```

# to find genes that have logFC significantly > 2 or significantly < -2 => for this, we have to test for DE relative to a threshold in the confirmation stage; this is done with the glmtreat function

```{r}

confirmationResultsFC2 <- sapply(1:ncol(LBrain),function(i) glmTreat(fit\_2way, contrast = LBrain[,i], lfc=2),simplify = FALSE)

confirmationPListFC2 <- lapply(confirmationResultsFC2, function(x) x$table$PValue)#takes the p-values from all genes for each contrast and puts them in a list

confirmationPFC2 <- as.matrix(Reduce(f=cbind,confirmationPListFC2))

rownames(confirmationPFC2) <- rownames(fit\_2way)

colnames(confirmationPFC2) <- colnames(LBrain)

stageRObjFC2 <- stageR(pScreen=pScreen, pConfirmation=confirmationPFC2) #constructs an object

stageRAdjFC2 <- stageWiseAdjustment(object=stageRObjFC2, method="holm", alpha=0.05) # adjusts the p-values using FWER correction; here we choose the holm method

resBrainFC2 <- getResults(stageRAdjFC2)

SignifGenesBrainFC2 = colSums(resBrainFC2) #number of DE genes in every contrast.

SignifGenesBrainFC2

adjusted\_pFC2 = getAdjustedPValues(stageRAdjFC2) #p-values to report in manuscript.pAdjScreen = pvalue of the global test, other columns are then the posthoc tests

resbrainFC2\_df = as.data.frame(resBrainFC2) #resBrain is a matrix, to be able to subset, change it to a dataframe

OnlySignGenesFC2 = resbrainFC2\_df[resbrainFC2\_df$padjScreen == 1,] #removes rows for which global test was non significant

##to get fold changes of the DE genes with logFC >2

##BSNa

Hyp1SignGenesBSNaFC2=names(which(resBrainFC2[,"BS-avg\_Na"]==1))

logFC2\_BSavgNa = confirmationResultsFC2[[1]]$table[Hyp1SignGenesBSNaFC2,] #adjust column to select correct contrast

BSNa\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesBSNaFC2)

logFC2\_BSavgNa\_adjP = merge(logFC2\_BSavgNa,BSNa\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_BSavgNa\_adjP = logFC2\_BSavgNa\_adjP[, c(1,2,7)] #adjust column to select correct contrast

colnames(logFC2\_BSavgNa\_adjP)[3] = "adjusted Pvalue"

##CENa

Hyp1SignGenesCENaFC2=names(which(resBrainFC2[,"CE-avg\_Na"]==1))

logFC2\_CEavgNa = confirmationResultsFC2[[3]]$table[Hyp1SignGenesCENaFC2,] #adjust column to select correct contrast

CENa\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesCENaFC2)

logFC2\_CEavgNa\_adjP = merge(logFC2\_CEavgNa,CENa\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_CEavgNa\_adjP = logFC2\_CEavgNa\_adjP[, c(1,2,9)] #adjust column to select correct contrast

colnames(logFC2\_CEavgNa\_adjP)[3] = "adjusted Pvalue"

##DINa

Hyp1SignGenesDINaFC2=names(which(resBrainFC2[,"DI-avg\_Na"]==1))

logFC2\_DIavgNa = confirmationResultsFC2[[5]]$table[Hyp1SignGenesDINaFC2,] #adjust column to select correct contrast

DINa\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesDINaFC2)

logFC2\_DIavgNa\_adjP = merge(logFC2\_DIavgNa,DINa\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_DIavgNa\_adjP = logFC2\_DIavgNa\_adjP[, c(1,2,11)] #adjust column to select correct contrast

colnames(logFC2\_DIavgNa\_adjP)[3] = "adjusted Pvalue"

##OBNa

Hyp1SignGenesOBNaFC2=names(which(resBrainFC2[,"OB-avg\_Na"]==1))

logFC2\_OBavgNa = confirmationResultsFC2[[7]]$table[Hyp1SignGenesOBNaFC2,] #adjust column to select correct contrast

OBNa\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesOBNaFC2)

logFC2\_OBavgNa\_adjP = merge(logFC2\_OBavgNa,OBNa\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_OBavgNa\_adjP = logFC2\_OBavgNa\_adjP[, c(1,2,13)] #adjust column to select correct contrast

colnames(logFC2\_OBavgNa\_adjP)[3] = "adjusted Pvalue"

##OTNa

Hyp1SignGenesOTNaFC2=names(which(resBrainFC2[,"OT-avg\_Na"]==1))

logFC2\_OTavgNa = confirmationResultsFC2[[9]]$table[Hyp1SignGenesOTNaFC2,] #adjust column to select correct contrast

OTNa\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesOTNaFC2)

logFC2\_OTavgNa\_adjP = merge(logFC2\_OTavgNa,OTNa\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_OTavgNa\_adjP = logFC2\_OTavgNa\_adjP[, c(1,2,15)] #adjust column to select correct contrast

colnames(logFC2\_OTavgNa\_adjP)[3] = "adjusted Pvalue"

##TENa

Hyp1SignGenesTENaFC2=names(which(resBrainFC2[,"TE-avg\_Na"]==1))

logFC2\_TEavgNa = confirmationResultsFC2[[11]]$table[Hyp1SignGenesTENaFC2,] #adjust column to select correct contrast

TENa\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesTENaFC2)

logFC2\_TEavgNa\_adjP = merge(logFC2\_TEavgNa,TENa\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_TEavgNa\_adjP = logFC2\_TEavgNa\_adjP[, c(1,2,17)] #adjust column to select correct contrast

colnames(logFC2\_TEavgNa\_adjP)[3] = "adjusted Pvalue"

##BSVe

Hyp1SignGenesBSVeFC2=names(which(resBrainFC2[,"BS-avg\_Ve"]==1))

logFC2\_BSavgVe = confirmationResultsFC2[[2]]$table[Hyp1SignGenesBSVeFC2,] #adjust column to select correct contrast

BSVe\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesBSVeFC2)

logFC2\_BSavgVe\_adjP = merge(logFC2\_BSavgVe,BSVe\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_BSavgVe\_adjP = logFC2\_BSavgVe\_adjP[, c(1,2,8)] #adjust column to select correct contrast

colnames(logFC2\_BSavgVe\_adjP)[3] = "adjusted Pvalue"

##CEVe

Hyp1SignGenesCEVeFC2=names(which(resBrainFC2[,"CE-avg\_Ve"]==1))

logFC2\_CEavgVe = confirmationResultsFC2[[4]]$table[Hyp1SignGenesCEVeFC2,] #adjust column to select correct contrast

CEVe\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesCEVeFC2)

logFC2\_CEavgVe\_adjP = merge(logFC2\_CEavgVe,CEVe\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_CEavgVe\_adjP = logFC2\_CEavgVe\_adjP[, c(1,2,10)] #adjust column to select correct contrast

colnames(logFC2\_CEavgVe\_adjP)[3] = "adjusted Pvalue"

##DIVe

Hyp1SignGenesDIVeFC2=names(which(resBrainFC2[,"DI-avg\_Ve"]==1))

logFC2\_DIavgVe = confirmationResultsFC2[[6]]$table[Hyp1SignGenesDIVeFC2,] #adjust column to select correct contrast

DIVe\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesDIVeFC2)

logFC2\_DIavgVe\_adjP = merge(logFC2\_DIavgVe,DIVe\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_DIavgVe\_adjP = logFC2\_DIavgVe\_adjP[, c(1,2,12)] #adjust column to select correct contrast

colnames(logFC2\_DIavgVe\_adjP)[3] = "adjusted Pvalue"

##OBVe

Hyp1SignGenesOBVeFC2=names(which(resBrainFC2[,"OB-avg\_Ve"]==1))

logFC2\_OBavgVe = confirmationResultsFC2[[8]]$table[Hyp1SignGenesOBVeFC2,] #adjust column to select correct contrast

OBVe\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesOBVeFC2)

logFC2\_OBavgVe\_adjP = merge(logFC2\_OBavgVe,OBVe\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_OBavgVe\_adjP = logFC2\_OBavgVe\_adjP[, c(1,2,14)] #adjust column to select correct contrast

colnames(logFC2\_OBavgVe\_adjP)[3] = "adjusted Pvalue"

##OTVe

Hyp1SignGenesOTVeFC2=names(which(resBrainFC2[,"OT-avg\_Ve"]==1))

logFC2\_OTavgVe = confirmationResultsFC2[[10]]$table[Hyp1SignGenesOTVeFC2,] #adjust column to select correct contrast

OTVe\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesOTVeFC2)

logFC2\_OTavgVe\_adjP = merge(logFC2\_OTavgVe,OTVe\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_OTavgVe\_adjP = logFC2\_OTavgVe\_adjP[, c(1,2,16)] #adjust column to select correct contrast

colnames(logFC2\_OTavgVe\_adjP)[3] = "adjusted Pvalue"

##TEVe

Hyp1SignGenesTEVeFC2=names(which(resBrainFC2[,"TE-avg\_Ve"]==1))

logFC2\_TEavgVe = confirmationResultsFC2[[12]]$table[Hyp1SignGenesTEVeFC2,] #adjust column to select correct contrast

TEVe\_adjustedPFC2 = subset(adjusted\_pFC2, rownames(adjusted\_pFC2)%in%Hyp1SignGenesTEVeFC2)

logFC2\_TEavgVe\_adjP = merge(logFC2\_TEavgVe,TEVe\_adjustedPFC2, by = 0, all = TRUE)

logFC2\_TEavgVe\_adjP = logFC2\_TEavgVe\_adjP[, c(1,2,18)] #adjust column to select correct contrast

colnames(logFC2\_TEavgVe\_adjP)[3] = "adjusted Pvalue"

##merge files

logFC2\_BS = merge(logFC2\_BSavgNa\_adjP, logFC2\_BSavgVe\_adjP, by.x = "Row.names", by.y = "Row.names")

logFC2\_CE = merge(logFC2\_CEavgNa\_adjP, logFC2\_CEavgVe\_adjP, by.x = "Row.names", by.y = "Row.names")

logFC2\_DI = merge(logFC2\_DIavgNa\_adjP, logFC2\_DIavgVe\_adjP, by.x = "Row.names", by.y = "Row.names")

logFC2\_OB = merge(logFC2\_OBavgNa\_adjP, logFC2\_OBavgVe\_adjP, by.x = "Row.names", by.y = "Row.names")

logFC2\_OT = merge(logFC2\_OTavgNa\_adjP, logFC2\_OTavgVe\_adjP, by.x = "Row.names", by.y = "Row.names")

logFC2\_TE = merge(logFC2\_TEavgNa\_adjP, logFC2\_TEavgVe\_adjP, by.x = "Row.names", by.y = "Row.names")

```

```{r}

## names of genes with logFC >2 and only DE in one brain part in one species

BS\_avgNaFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Na`== 1 & OnlySignGenesFC2$`CE-avg\_Na` == 0 & OnlySignGenesFC2$`DI-avg\_Na` == 0 & OnlySignGenesFC2$`OB-avg\_Na` == 0 & OnlySignGenesFC2$`OT-avg\_Na` == 0 & OnlySignGenesFC2$`TE-avg\_Na`== 0)))

CE\_avgNaFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Na`== 0 & OnlySignGenesFC2$`CE-avg\_Na` == 1 & OnlySignGenesFC2$`DI-avg\_Na` == 0 & OnlySignGenesFC2$`OB-avg\_Na` == 0 & OnlySignGenesFC2$`OT-avg\_Na` == 0 & OnlySignGenesFC2$`TE-avg\_Na`== 0)))

DI\_avgNaFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Na`== 0 & OnlySignGenesFC2$`CE-avg\_Na` == 0 & OnlySignGenesFC2$`DI-avg\_Na` == 1 & OnlySignGenesFC2$`OB-avg\_Na` == 0 & OnlySignGenesFC2$`OT-avg\_Na` == 0 & OnlySignGenesFC2$`TE-avg\_Na`== 0)))

OB\_avgNaFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Na`== 0 & OnlySignGenesFC2$`CE-avg\_Na` == 0 & OnlySignGenesFC2$`DI-avg\_Na` == 0 & OnlySignGenesFC2$`OB-avg\_Na` == 1 & OnlySignGenesFC2$`OT-avg\_Na` == 0 & OnlySignGenesFC2$`TE-avg\_Na`== 0)))

OT\_avgNaFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Na`== 0 & OnlySignGenesFC2$`CE-avg\_Na` == 0 & OnlySignGenesFC2$`DI-avg\_Na` == 0 & OnlySignGenesFC2$`OB-avg\_Na` == 0 & OnlySignGenesFC2$`OT-avg\_Na` == 1 & OnlySignGenesFC2$`TE-avg\_Na`== 0)))

TE\_avgNaFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Na`== 0 & OnlySignGenesFC2$`CE-avg\_Na` == 0 & OnlySignGenesFC2$`DI-avg\_Na` == 0 & OnlySignGenesFC2$`OB-avg\_Na` == 0 & OnlySignGenesFC2$`OT-avg\_Na` == 0 & OnlySignGenesFC2$`TE-avg\_Na`== 1)))

BS\_avgVeFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Ve`== 1 & OnlySignGenesFC2$`CE-avg\_Ve` == 0 & OnlySignGenesFC2$`DI-avg\_Ve` == 0 & OnlySignGenesFC2$`OB-avg\_Ve` == 0 & OnlySignGenesFC2$`OT-avg\_Ve` == 0 & OnlySignGenesFC2$`TE-avg\_Ve`== 0)))

CE\_avgVeFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Ve`== 0 & OnlySignGenesFC2$`CE-avg\_Ve` == 1 & OnlySignGenesFC2$`DI-avg\_Ve` == 0 & OnlySignGenesFC2$`OB-avg\_Ve` == 0 & OnlySignGenesFC2$`OT-avg\_Ve` == 0 & OnlySignGenesFC2$`TE-avg\_Ve`== 0)))

DI\_avgVeFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Ve`== 0 & OnlySignGenesFC2$`CE-avg\_Ve` == 0 & OnlySignGenesFC2$`DI-avg\_Ve` == 1 & OnlySignGenesFC2$`OB-avg\_Ve` == 0 & OnlySignGenesFC2$`OT-avg\_Ve` == 0 & OnlySignGenesFC2$`TE-avg\_Ve`== 0)))

OB\_avgVeFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Ve`== 0 & OnlySignGenesFC2$`CE-avg\_Ve` == 0 & OnlySignGenesFC2$`DI-avg\_Ve` == 0 & OnlySignGenesFC2$`OB-avg\_Ve` == 1 & OnlySignGenesFC2$`OT-avg\_Ve` == 0 & OnlySignGenesFC2$`TE-avg\_Ve`== 0)))

OT\_avgVeFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Ve`== 0 & OnlySignGenesFC2$`CE-avg\_Ve` == 0 & OnlySignGenesFC2$`DI-avg\_Ve` == 0 & OnlySignGenesFC2$`OB-avg\_Ve` == 0 & OnlySignGenesFC2$`OT-avg\_Ve` == 1 & OnlySignGenesFC2$`TE-avg\_Ve`== 0)))

TE\_avgVeFC2 = c(rownames (subset (OnlySignGenesFC2, OnlySignGenesFC2$`BS-avg\_Ve`== 0 & OnlySignGenesFC2$`CE-avg\_Ve` == 0 & OnlySignGenesFC2$`DI-avg\_Ve` == 0 & OnlySignGenesFC2$`OB-avg\_Ve` == 0 & OnlySignGenesFC2$`OT-avg\_Ve` == 0 & OnlySignGenesFC2$`TE-avg\_Ve`== 1)))

## III.4.3. Hypothesis 2: There is no differential expression between a brain part and another brain part between species (i.e. the interaction brain part x species)

## III.4.3.1. Screening stage

```{r}

## contrasts between species.

LBetween=matrix(0,nrow=ncol(fit\_2way$coefficients),ncol=15)

rownames(LBetween)=colnames(fit\_2way$coefficients)

colnames(LBetween)=c("BSvCE\_Na-BSvCE\_Ve","BSvDI\_Na-BSvDI\_Ve","BSvOB\_Na-BSvOB\_Ve","BSvOT\_Na-BSvOT\_Ve","BSvTE\_Na-BSvTE\_Ve",

 "CEvDI\_Na-CEvDI\_Ve","CEvOB\_Na-CEvOB\_Ve","CEvOT\_Na-CEvOT\_Ve","CEvTE\_Na-CEvTE\_Ve",

 "DIvOB\_Na-DIvOB\_Ve","DIvOT\_Na-DIvOT\_Ve","DIvTE\_Na-DIvTE\_Ve",

 "OBvOT\_Na-OBvOT\_Ve","OBvTE\_Na-OBvTE\_Ve",

 "OTvTE\_Na-OTvTE\_Ve")

LBetween[11:12,1]=c(-1,1)

LBetween[13:14,2]=c(-1,1)

LBetween[15:16,3]=c(-1,1)

LBetween[17:18,4]=c(-1,1)

LBetween[19:20,5]=c(-1,1)

LBetween[c("CE\_Na", "DI\_Na", "CE\_Ve", "DI\_Ve"),"CEvDI\_Na-CEvDI\_Ve"]= c(1, -1, -1, 1)

LBetween[c("CE\_Na", "OB\_Na", "CE\_Ve", "OB\_Ve"),"CEvOB\_Na-CEvOB\_Ve"]= c(1, -1, -1, 1)

LBetween[c("CE\_Na", "OT\_Na", "CE\_Ve", "OT\_Ve"),"CEvOT\_Na-CEvOT\_Ve"]= c(1, -1, -1, 1)

LBetween[c("CE\_Na", "TE\_Na", "CE\_Ve", "TE\_Ve"),"CEvTE\_Na-CEvTE\_Ve"]= c(1, -1, -1, 1)

LBetween[c("DI\_Na", "OB\_Na", "DI\_Ve", "OB\_Ve"),"DIvOB\_Na-DIvOB\_Ve"]= c(1, -1, -1, 1)

LBetween[c("DI\_Na", "OT\_Na", "DI\_Ve", "OT\_Ve"),"DIvOT\_Na-DIvOT\_Ve"]= c(1, -1, -1, 1)

LBetween[c("DI\_Na", "TE\_Na", "DI\_Ve", "TE\_Ve"),"DIvTE\_Na-DIvTE\_Ve"]= c(1, -1, -1, 1)

LBetween[c("OB\_Na", "OT\_Na", "OB\_Ve", "OT\_Ve"),"OBvOT\_Na-OBvOT\_Ve"]= c(1, -1, -1, 1)

LBetween[c("OB\_Na", "TE\_Na", "OB\_Ve", "TE\_Ve"),"OBvTE\_Na-OBvTE\_Ve"]= c(1, -1, -1, 1)

LBetween[c("OT\_Na", "TE\_Na", "OT\_Ve", "TE\_Ve"),"OTvTE\_Na-OTvTE\_Ve"]= c(1, -1, -1, 1)

alpha=0.05

screenTestBetween = glmQLFTest(fit\_2way, contrast=LBetween)

pScreenBetween <- screenTestBetween$table$PValue

names(pScreenBetween) <- rownames(screenTestBetween$table)

sum(p.adjust(pScreenBetween, method="BH")<=0.05) #32

```

## III.4.3.2. Confirmation stage

```{r}

confirmationResultsHyp2 <- sapply(1:ncol(LBetween),function(i) glmQLFTest(fit\_2way, contrast = LBetween[,i]), simplify=FALSE)

confirmationPListBetween <- lapply(confirmationResultsHyp2, function(x) x$table$PValue)

confirmationPBetween <- as.matrix(Reduce(f=cbind,confirmationPListBetween))

rownames(confirmationPBetween) <- rownames(confirmationResultsHyp2[[1]]$table)

colnames(confirmationPBetween) <- colnames(LBetween)

stageRObjBetween <- stageR(pScreen=pScreenBetween, pConfirmation=confirmationPBetween)

stageRAdjBetween <- stageWiseAdjustment(object=stageRObjBetween, method="holm", alpha=0.05)

resBetween <- getResults(stageRAdjBetween)

SignGenesBetween = colSums(resBetween) #number of DE genes in every contrast.

SignGenesBetween

adjusted\_p\_Between = getAdjustedPValues(stageRAdjBetween) #p-values to report in manuscript.pAdjScreen = pvalue of the global test, other columns are then the posthoc tests

resBetween\_df = as.data.frame(resBetween) #resBrain is a matrix, to be able to subset, change it to a dataframe

OnlySignGenesBetween = resBetween\_df[resBetween\_df$padjScreen == 1,] #removes rows for which global test was non significant

### get fold changes for the significant gene(s)

Hyp2SignGenesBSvsCE=names(which(resBetween[,2]==1))

logFC\_BSvsCE = confirmationResultsHyp2[[1]]$table[Hyp2SignGenesBSvsCE,] #adjust column to select correct contrast

BSvsCE\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesBSvsCE)

logFC\_BSvsCE\_adjP = merge(logFC\_BSvsCE,BSvsCE\_adjustedP, by = 0, all = TRUE)

logFC\_BSvsCE\_adjP$contrast = "BSvsCE" #add column to be able to differentiate after merging

logFC\_BSvsCE\_adjP = logFC\_BSvsCE\_adjP[,c(1,2,7,22)]

colnames(logFC\_BSvsCE\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesBSvsDI=names(which(resBetween[,3]==1))

logFC\_BSvsDI = confirmationResultsHyp2[[2]]$table[Hyp2SignGenesBSvsDI,] #adjust column to select correct contrast

BSvsDI\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesBSvsDI)

logFC\_BSvsDI\_adjP = merge(logFC\_BSvsDI,BSvsDI\_adjustedP, by = 0, all = TRUE)

logFC\_BSvsDI\_adjP$contrast = "BSvsDI"

logFC\_BSvsDI\_adjP = logFC\_BSvsDI\_adjP[,c(1,2,8,22)]

colnames(logFC\_BSvsDI\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesBSvsOB=names(which(resBetween[,4]==1))

logFC\_BSvsOB = confirmationResultsHyp2[[3]]$table[Hyp2SignGenesBSvsOB,] #adjust column to select correct contrast

BSvsOB\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesBSvsOB)

logFC\_BSvsOB\_adjP = merge(logFC\_BSvsOB,BSvsOB\_adjustedP, by = 0, all = TRUE)

logFC\_BSvsOB\_adjP$contrast = "BSvsOB"

logFC\_BSvsOB\_adjP = logFC\_BSvsOB\_adjP[,c(1,2,9,22)]

colnames(logFC\_BSvsOB\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesBSvsTE=names(which(resBetween[,6]==1))

logFC\_BSvsTE = confirmationResultsHyp2[[5]]$table[Hyp2SignGenesBSvsTE,] #adjust column to select correct contrast

BSvsTE\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesBSvsTE)

logFC\_BSvsTE\_adjP = merge(logFC\_BSvsTE,BSvsTE\_adjustedP, by = 0, all = TRUE)

logFC\_BSvsTE\_adjP$contrast = "BSvsTE"

logFC\_BSvsTE\_adjP = logFC\_BSvsTE\_adjP[,c(1,2,11,22)]

colnames(logFC\_BSvsTE\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesCEvsDI=names(which(resBetween[,7]==1))

logFC\_CEvsDI = confirmationResultsHyp2[[6]]$table[Hyp2SignGenesCEvsDI,] #adjust column to select correct contrast

CEvsDI\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesCEvsDI)

logFC\_CEvsDI\_adjP = merge(logFC\_CEvsDI,CEvsDI\_adjustedP, by = 0, all = TRUE)

logFC\_CEvsDI\_adjP$contrast = "CEvsDI"

logFC\_CEvsDI\_adjP = logFC\_CEvsDI\_adjP[,c(1,2,12,22)]

colnames(logFC\_CEvsDI\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesCEvsOB=names(which(resBetween[,8]==1))

logFC\_CEvsOB = confirmationResultsHyp2[[7]]$table[Hyp2SignGenesCEvsOB,] #adjust column to select correct contrast

CEvsOB\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesCEvsOB)

logFC\_CEvsOB\_adjP = merge(logFC\_CEvsOB,CEvsOB\_adjustedP, by = 0, all = TRUE)

logFC\_CEvsOB\_adjP$contrast = "CEvsOB"

logFC\_CEvsOB\_adjP = logFC\_CEvsOB\_adjP[,c(1,2,13,22)]

colnames(logFC\_CEvsOB\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesCEvsOT=names(which(resBetween[,9]==1))

logFC\_CEvsOT = confirmationResultsHyp2[[8]]$table[Hyp2SignGenesCEvsOT,] #adjust column to select correct contrast

CEvsOT\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesCEvsOT)

logFC\_CEvsOT\_adjP = merge(logFC\_CEvsOT,CEvsOT\_adjustedP, by = 0, all = TRUE)

logFC\_CEvsOT\_adjP$contrast = "CEvsOT"

logFC\_CEvsOT\_adjP = logFC\_CEvsOT\_adjP[,c(1,2,14,22)]

colnames(logFC\_CEvsOT\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesCEvsTE=names(which(resBetween[,10]==1))

logFC\_CEvsTE = confirmationResultsHyp2[[9]]$table[Hyp2SignGenesCEvsTE,] #adjust column to select correct contrast

CEvsTE\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesCEvsTE)

logFC\_CEvsTE\_adjP = merge(logFC\_CEvsTE,CEvsTE\_adjustedP, by = 0, all = TRUE)

logFC\_CEvsTE\_adjP$contrast = "CEvsTE"

logFC\_CEvsTE\_adjP = logFC\_CEvsTE\_adjP[,c(1,2,15,22)]

colnames(logFC\_CEvsTE\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesDIvsOB=names(which(resBetween[,11]==1))

logFC\_DIvsOB = confirmationResultsHyp2[[10]]$table[Hyp2SignGenesDIvsOB,] #adjust column to select correct contrast

DIvsOB\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesDIvsOB)

logFC\_DIvsOB\_adjP = merge(logFC\_DIvsOB,DIvsOB\_adjustedP, by = 0, all = TRUE)

logFC\_DIvsOB\_adjP$contrast = "DIvsOB"

logFC\_DIvsOB\_adjP = logFC\_DIvsOB\_adjP[,c(1,2,16,22)]

colnames(logFC\_DIvsOB\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesDIvsOT=names(which(resBetween[,12]==1))

logFC\_DIvsOT = confirmationResultsHyp2[[11]]$table[Hyp2SignGenesDIvsOT,] #adjust column to select correct contrast

DIvsOT\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesDIvsOT)

logFC\_DIvsOT\_adjP = merge(logFC\_DIvsOT,DIvsOT\_adjustedP, by = 0, all = TRUE)

logFC\_DIvsOT\_adjP$contrast = "DIvsOT"

logFC\_DIvsOT\_adjP = logFC\_DIvsOT\_adjP[,c(1,2,17,22)]

colnames(logFC\_DIvsOT\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesDIvsTE=names(which(resBetween[,13]==1))

logFC\_DIvsTE = confirmationResultsHyp2[[12]]$table[Hyp2SignGenesDIvsTE,] #adjust column to select correct contrast

DIvsTE\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesDIvsTE)

logFC\_DIvsTE\_adjP = merge(logFC\_DIvsTE,DIvsTE\_adjustedP, by = 0, all = TRUE)

logFC\_DIvsTE\_adjP$contrast = "DIvsTE"

logFC\_DIvsTE\_adjP = logFC\_DIvsTE\_adjP[,c(1,2,18,22)]

colnames(logFC\_DIvsTE\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesOBvsOT=names(which(resBetween[,14]==1))

logFC\_OBvsOT = confirmationResultsHyp2[[13]]$table[Hyp2SignGenesOBvsOT,] #adjust column to select correct contrast

OBvsOT\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesOBvsOT)

logFC\_OBvsOT\_adjP = merge(logFC\_OBvsOT,OBvsOT\_adjustedP, by = 0, all = TRUE)

logFC\_OBvsOT\_adjP$contrast = "OBvsOT"

logFC\_OBvsOT\_adjP = logFC\_OBvsOT\_adjP[,c(1,2,19,22)]

colnames(logFC\_OBvsOT\_adjP)[3] = "adj Pvalue"

Hyp2SignGenesOTvsTE=names(which(resBetween[,16]==1))

logFC\_OTvsTE = confirmationResultsHyp2[[15]]$table[Hyp2SignGenesOTvsTE,] #adjust column to select correct contrast

OTvsTE\_adjustedP = subset(adjusted\_p\_Between, rownames(adjusted\_p\_Between)%in%Hyp2SignGenesOTvsTE)

logFC\_OTvsTE\_adjP = merge(logFC\_OTvsTE,OTvsTE\_adjustedP, by = 0, all = TRUE)

logFC\_OTvsTE\_adjP$contrast = "OTvsTE"

logFC\_OTvsTE\_adjP = logFC\_OTvsTE\_adjP[,c(1,2,21,22)]

colnames(logFC\_OTvsTE\_adjP)[3] = "adj Pvalue"

logFC\_adjP\_interactiongenes = rbind(logFC\_OTvsTE\_adjP,logFC\_OBvsOT\_adjP, logFC\_DIvsTE\_adjP, logFC\_DIvsOT\_adjP, logFC\_DIvsOB\_adjP, logFC\_CEvsTE\_adjP, logFC\_CEvsOT\_adjP, logFC\_CEvsOB\_adjP, logFC\_CEvsDI\_adjP, logFC\_BSvsTE\_adjP, logFC\_BSvsOB\_adjP, logFC\_BSvsDI\_adjP, logFC\_BSvsCE\_adjP)

```

## III.4.3.2. Equivalence testing

```{r}

# credit to Aaron Lun for the equivalence test, at https://support.bioconductor.org/p/66283/

### reparametrizing the design matrix

contrastList = list()

for(i in 1:5) contrastList[[i]] = c(1:5)

for(i in 6:9) contrastList[[i]] = c(1:5)+(i-4)-1

contrastList[[10]] = c(1,4,7,9,10)

contrastList[[11]] = c(1,4,7,9,11)

contrastList[[12]] = c(1,4,7,9,12)

contrastList[[13]] = c(1,4,6,9,13)

contrastList[[14]] = c(1,4,6,9,14)

contrastList[[15]] = c(1,4,6,7,15)

fit=fit\_2way

pEquivalence=matrix(NA,nrow=nrow(d),ncol=15)

for(i in 1:15){

 cat(i)

 # prepare design matrix

 hlp=contrastAsCoef(design\_2way, contrast=LBetween[,contrastList[[i]]])

 design=hlp$design

 y=d

orig.offs = log(y$samples$lib.size\*y$samples$norm.factors)

if(i %in% 1:5){

 y$offset <- as.vector(orig.offs + log(4)\*design[,i])

 fit.up <- glmFit(y, design)

 lrt.up <- glmLRT(fit.up, coef=i)

 y$offset <- as.vector(orig.offs - log(4)\*design[,i])

 fit.down <- glmFit(y, design)

 lrt.down <- glmLRT(fit.down, coef=i)

 p.up <- ifelse(lrt.up$table$logFC < 0, lrt.up$table$PValue/2, 1)

 p.down <- ifelse(lrt.down$table$logFC > 0, lrt.down$table$PValue/2, 1)

 p.tost <- pmax(p.up, p.down)

} else if(i %in% 6:15){

 #for these contrasts, always the last of the rank 5 contrast matrix is the contrast of interest.

 y$offset <- as.vector(orig.offs + log(4)\*design[,5])

 fit.up <- glmFit(y, design)

 lrt.up <- glmLRT(fit.up, coef=5)

 y$offset <- as.vector(orig.offs - log(4)\*design[,5])

 fit.down <- glmFit(y, design)

 lrt.down <- glmLRT(fit.down, coef=5)

 p.up <- ifelse(lrt.up$table$logFC < 0, lrt.up$table$PValue/2, 1)

 p.down <- ifelse(lrt.down$table$logFC > 0, lrt.down$table$PValue/2, 1)

 p.tost <- pmax(p.up, p.down)

}

pEquivalence[,i] = p.tost

}

## how many are equivalent in all contrasts

maxPEquivalence = apply(pEquivalence,1,max)

fdrAdjEquivalence = p.adjust(maxPEquivalence,"fdr")

mean(fdrAdjEquivalence<=0.05) #71% genes equivalent across all contrasts

#within sample significance

apply(pEquivalence,2,function(x) mean(p.adjust(x,"fdr")<=0.05))

```

#IV. Plots

##IV.2. Stage-wise analysis

## IV.2. Hypothesis 1: there is no differential expression between a brain part and the average of all other brain parts in O. nasuta and O. ventralis.

#the heatmap function requires the data to be a numerical matrix; we extract the DE genes from the DGE list, calculate cpm and return log2values

```{r}

nGenesToPlot=200

topGenes = rownames(getAdjustedPValues(stageRAdj,order=TRUE))[1:nGenesToPlot]

cpmBrain=cpm(d[topGenes,],log = TRUE) #subset the DGElist, calculate cpm, return log2 values (this makes comparison across orders of magnitude of expression possible)

hmcol <- colorRampPalette(brewer.pal(9, "GnBu"))(100)

heatmap.2(cpmBrain,dendrogram = "both",trace="none",col=hmcol,main="top 200 DE genes", cexRow = 0.2, cexCol = 0.5)

```

## top interaction genes

```{r}

topGenesInteraction = rownames(getAdjustedPValues(stageRAdjBetween,order=TRUE))[1:nGenesToPlot]

cpmBrain=cpm(d[topGenesInteraction,],log=TRUE)

hmcol <- colorRampPalette(brewer.pal(9, "GnBu"))(100)

heatmap.2(cpmBrain,trace="none",col=hmcol,main= "interaction genes")

nGenesToPlot=colSums(resBetween)[1] #nr of screening stage significant genes

topGenesInteraction = rownames(getAdjustedPValues(stageRAdjBetween))[1:nGenesToPlot]

cpmInter=cpm(d[topGenesInteraction,],log=TRUE)

plotHlp=cbind(cpmInter[,1:nGenesToPlot],NA,cpmInter[,-(1:nGenesToPlot)])

pdf("InteractionGenes.pdf", width = 7, height = 5)

par(mfrow=c(3,3),mar=c(2,5,3,1))

for(i in 1:nGenesToPlot)

{

plotHlp2=matrix(plotHlp[i,],nrow=5,ncol=12)

matplot(t(plotHlp2[,1:6]),type="b",ylim=range(plotHlp,na.rm=TRUE),xaxt="n",col=1,lty=1,pch=1,main=topGenesInteraction[i], ylab="log CPM")

matplot(t(plotHlp2[,-(1:6)]),type="b",ylim=range(plotHlp,na.rm=TRUE),xaxt="n",col=2,add=TRUE,lty=1,pch=2)

axis(at=1:6,labels=levels(brain),side=1)

}

dev.off()

plotHlp2

?gplots

```

#3. GO annotation

the count files contain the gene names, we obtained the GeneIDs from the genes using the genome annotation table from genbank (https://www.ncbi.nlm.nih.gov/genome/proteins/197?genome\_assembly\_id=293496).

```{r}

oreochromis = read.table ("Oniloticus\_ProteinTable197\_293496.txt", sep = "", comment.char="", quote = "\"", header = TRUE, fill = TRUE, stringsAsFactors = FALSE)

namesOnlySignGenes = rownames(OnlySignGenes) # all locus names of the significant genes

geneIDOnlySignGenesTable = oreochromis[oreochromis$Locus %in% namesOnlySignGenes,] #get only those that are DE

geneIDOnlySignGenesTableUnique = subset(geneIDOnlySignGenesTable, !duplicated(geneIDOnlySignGenesTable$GeneID)) # remove duplicated gene IDs

geneIDOnlySignGenes = c(geneIDOnlySignGenesTableUnique$GeneID) # put geneIDs in a vector to be used in biomart

library(biomaRt)

ensembl = useMart("ensembl",dataset="oniloticus\_gene\_ensembl")

filters = listFilters(ensembl) #to see which filters can be used

attributes = listAttributes(ensembl) #to see which parameters can be in the output table

go\_all = getBM(attributes=c("refseq\_peptide","refseq\_peptide\_predicted", "entrezgene", "go\_id", "name\_1006", "definition\_1006", "namespace\_1003"), filters = "entrezgene", values = geneIDOnlySignGenes, mart=ensembl)

go\_all\_goID = go\_all[!go\_all$go\_id == "",] #remove rows with no go\_id

go\_all\_goID\_biologicalprocess = go\_all\_goID[go\_all\_goID$namespace\_1003 == 'biological\_process',] #get the biological function

write.table (go\_all\_goID\_biologicalprocess, "GO\_brainparts.txt", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = ";")

go\_all\_goID\_biologicalprocess\_unique = go\_all\_goID\_biologicalprocess[!duplicated(go\_all\_goID\_biologicalprocess[,c(3:4)]),]

#get the GeneIDs from the loci DE only in each brainpart for O. nasuta:

geneID\_BS\_avgNa = oreochromis[oreochromis$Locus %in% BS\_avgNa,]

geneID\_BS\_avgNa\_unique = subset(geneID\_BS\_avgNa, !duplicated(geneID\_BS\_avgNa$GeneID))

geneID\_CE\_avgNa = oreochromis[oreochromis$Locus %in% CE\_avgNa,]

geneID\_CE\_avgNa\_unique = subset(geneID\_CE\_avgNa, !duplicated(geneID\_CE\_avgNa$GeneID))

geneID\_TE\_avgNa = oreochromis[oreochromis$Locus %in% TE\_avgNa,]

geneID\_TE\_avgNa\_unique = subset(geneID\_TE\_avgNa, !duplicated(geneID\_TE\_avgNa$GeneID))

geneID\_OB\_avgNa = oreochromis[oreochromis$Locus %in% OB\_avgNa,]

geneID\_OB\_avgNa\_unique = subset(geneID\_OB\_avgNa, !duplicated(geneID\_OB\_avgNa$GeneID))

geneID\_OT\_avgNa = oreochromis[oreochromis$Locus %in% OT\_avgNa,]

geneID\_OT\_avgNa\_unique = subset(geneID\_OT\_avgNa, !duplicated(geneID\_OT\_avgNa$GeneID))

geneID\_DI\_avgNa = oreochromis[oreochromis$Locus %in% DI\_avgNa,]

geneID\_DI\_avgNa\_unique = subset(geneID\_DI\_avgNa, !duplicated(geneID\_DI\_avgNa$GeneID))

#get the GeneIDs from the loci DE in each brainpart for O. ventralis:

geneID\_BS\_avgVe = oreochromis[oreochromis$Locus %in% BS\_avgVe,]

geneID\_BS\_avgVe\_unique = subset(geneID\_BS\_avgVe, !duplicated(geneID\_BS\_avgVe$GeneID))

geneID\_CE\_avgVe = oreochromis[oreochromis$Locus %in% CE\_avgVe,]

geneID\_CE\_avgVe\_unique = subset(geneID\_CE\_avgVe, !duplicated(geneID\_CE\_avgVe$GeneID))

geneID\_TE\_avgVe = oreochromis[oreochromis$Locus %in% TE\_avgVe,]

geneID\_TE\_avgVe\_unique = subset(geneID\_TE\_avgVe, !duplicated(geneID\_TE\_avgVe$GeneID))

geneID\_OB\_avgVe = oreochromis[oreochromis$Locus %in% OB\_avgVe,]

geneID\_OB\_avgVe\_unique = subset(geneID\_OB\_avgVe, !duplicated(geneID\_OB\_avgVe$GeneID))

geneID\_OT\_avgVe = oreochromis[oreochromis$Locus %in% OT\_avgVe,]

geneID\_OT\_avgVe\_unique = subset(geneID\_OT\_avgVe, !duplicated(geneID\_OT\_avgVe$GeneID))

geneID\_DI\_avgVe\_unique = subset(geneID\_DI\_avgVe, !duplicated(geneID\_DI\_avgVe$GeneID))

geneID\_DI\_avgVe = oreochromis[oreochromis$Locus %in% DI\_avgVe,]

#get the Gene ID for the DE genes with logFC > 2 in both species

gene\_ID\_logFC2\_BS = oreochromis[oreochromis$Locus %in% logFC2\_BS$Row.names,]

gene\_ID\_logFC2\_BS\_unique = subset(gene\_ID\_logFC2\_BS, !duplicated(gene\_ID\_logFC2\_BS$GeneID))

go\_logFC2\_BS = subset(go\_all\_goID\_biologicalprocess\_unique,go\_all\_goID\_biologicalprocess\_unique$entrezgene %in% gene\_ID\_logFC2\_BS\_unique$GeneID)

go\_logFC2\_BS = go\_logFC2\_BS[,3:5]

go\_logFC2\_BS = merge(gene\_ID\_logFC2\_BS\_unique, go\_logFC2\_BS, by.x = "GeneID", by.y = "entrezgene")

go\_logFC2\_BS = go\_logFC2\_BS[,c(1,6,9,10)]

go\_logFC2\_BS = merge(go\_logFC2\_BS,logFC2\_BS, by.x = "Locus", by.y = "Row.names")

colnames(go\_logFC2\_BS) = c("Locus", "GeneID", "GoID", "Biological Process", "LogFC\_Na", "AdjustedPvalue\_Na", "LogFC\_Ve", "AdjustedPvalue\_Ve")

gene\_ID\_logFC2\_CE = oreochromis[oreochromis$Locus %in% logFC2\_CE$Row.names,]

gene\_ID\_logFC2\_CE\_unique = subset(gene\_ID\_logFC2\_CE, !duplicated(gene\_ID\_logFC2\_CE$GeneID))

go\_logFC2\_CE = subset(go\_all\_goID\_biologicalprocess\_unique,go\_all\_goID\_biologicalprocess\_unique$entrezgene %in% gene\_ID\_logFC2\_CE\_unique$GeneID)

go\_logFC2\_CE = go\_logFC2\_CE[,3:5]

go\_logFC2\_CE = merge(gene\_ID\_logFC2\_CE\_unique, go\_logFC2\_CE, by.x = "GeneID", by.y = "entrezgene")

go\_logFC2\_CE = go\_logFC2\_CE[,c(1,6,9,10)]

go\_logFC2\_CE = merge(go\_logFC2\_CE,logFC2\_CE, by.x = "Locus", by.y = "Row.names")

colnames(go\_logFC2\_CE) = c("Locus", "GeneID", "GoID", "Biological Process", "LogFC\_Na", "AdjustedPvalue\_Na", "LogFC\_Ve", "AdjustedPvalue\_Ve")

gene\_ID\_logFC2\_DI = oreochromis[oreochromis$Locus %in% logFC2\_DI$Row.names,]

gene\_ID\_logFC2\_DI\_unique = subset(gene\_ID\_logFC2\_DI, !duplicated(gene\_ID\_logFC2\_DI$GeneID))

go\_logFC2\_DI = subset(go\_all\_goID\_biologicalprocess\_unique,go\_all\_goID\_biologicalprocess\_unique$entrezgene %in% gene\_ID\_logFC2\_DI\_unique$GeneID)

go\_logFC2\_DI = go\_logFC2\_DI[,3:5]

go\_logFC2\_DI = merge(gene\_ID\_logFC2\_DI\_unique, go\_logFC2\_DI, by.x = "GeneID", by.y = "entrezgene")

go\_logFC2\_DI = go\_logFC2\_DI[,c(1,6,9,10)]

go\_logFC2\_DI = merge(go\_logFC2\_DI,logFC2\_DI, by.x = "Locus", by.y = "Row.names")

colnames(go\_logFC2\_DI) = c("Locus", "GeneID", "GoID", "Biological Process", "LogFC\_Na", "AdjustedPvalue\_Na", "LogFC\_Ve", "AdjustedPvalue\_Ve")

gene\_ID\_logFC2\_OB = oreochromis[oreochromis$Locus %in% logFC2\_OB$Row.names,]

gene\_ID\_logFC2\_OB\_unique = subset(gene\_ID\_logFC2\_OB, !duplicated(gene\_ID\_logFC2\_OB$GeneID))

go\_logFC2\_OB = subset(go\_all\_goID\_biologicalprocess\_unique,go\_all\_goID\_biologicalprocess\_unique$entrezgene %in% gene\_ID\_logFC2\_OB\_unique$GeneID)

go\_logFC2\_OB = go\_logFC2\_OB[,3:5]

go\_logFC2\_OB = merge(gene\_ID\_logFC2\_OB\_unique, go\_logFC2\_OB, by.x = "GeneID", by.y = "entrezgene")

go\_logFC2\_OB = go\_logFC2\_OB[,c(1,6,9,10)]

go\_logFC2\_OB = merge(go\_logFC2\_OB,logFC2\_OB, by.x = "Locus", by.y = "Row.names")

colnames(go\_logFC2\_OB) = c("Locus", "GeneID", "GoID", "Biological Process", "LogFC\_Na", "AdjustedPvalue\_Na", "LogFC\_Ve", "AdjustedPvalue\_Ve")

gene\_ID\_logFC2\_OT = oreochromis[oreochromis$Locus %in% logFC2\_OT$Row.names,]

gene\_ID\_logFC2\_OT\_unique = subset(gene\_ID\_logFC2\_OT, !duplicated(gene\_ID\_logFC2\_OT$GeneID))

go\_logFC2\_OT = subset(go\_all\_goID\_biologicalprocess\_unique,go\_all\_goID\_biologicalprocess\_unique$entrezgene %in% gene\_ID\_logFC2\_OT\_unique$GeneID)

go\_logFC2\_OT = go\_logFC2\_OT[,3:5]

go\_logFC2\_OT = merge(gene\_ID\_logFC2\_OT\_unique, go\_logFC2\_OT, by.x = "GeneID", by.y = "entrezgene")

go\_logFC2\_OT = go\_logFC2\_OT[,c(1,6,9,10)]

go\_logFC2\_OT = merge(go\_logFC2\_OT,logFC2\_OT, by.x = "Locus", by.y = "Row.names")

colnames(go\_logFC2\_OT) = c("Locus", "GeneID", "GoID", "Biological Process", "LogFC\_Na", "AdjustedPvalue\_Na", "LogFC\_Ve", "AdjustedPvalue\_Ve")

gene\_ID\_logFC2\_TE = oreochromis[oreochromis$Locus %in% logFC2\_TE$Row.names,]

gene\_ID\_logFC2\_TE\_unique = subset(gene\_ID\_logFC2\_TE, !duplicated(gene\_ID\_logFC2\_TE$GeneID))

go\_logFC2\_TE = subset(go\_all\_goID\_biologicalprocess\_unique,go\_all\_goID\_biologicalprocess\_unique$entrezgene %in% gene\_ID\_logFC2\_TE\_unique$GeneID)

go\_logFC2\_TE = go\_logFC2\_TE[,3:5]

go\_logFC2\_TE = merge(gene\_ID\_logFC2\_TE\_unique, go\_logFC2\_TE, by.x = "GeneID", by.y = "entrezgene")

go\_logFC2\_TE = go\_logFC2\_TE[,c(1,6,9,10)]

go\_logFC2\_TE = merge(go\_logFC2\_TE,logFC2\_TE, by.x = "Locus", by.y = "Row.names")

colnames(go\_logFC2\_TE) = c("Locus", "GeneID", "GoID", "Biological Process", "LogFC\_Na", "AdjustedPvalue\_Na", "LogFC\_Ve", "AdjustedPvalue\_Ve")

write.table (go\_logFC2\_BS, "logFC2\_BS\_all.txt", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = ";")

write.table (go\_logFC2\_CE, "logFC2\_CE\_all.txt", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = ";")

write.table (go\_logFC2\_DI, "logFC2\_DI\_all.txt", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = ";")

write.table (go\_logFC2\_OB, "logFC2\_OB\_all.txt", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = ";")

write.table (go\_logFC2\_OT, "logFC2\_OT\_all.txt", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = ";")

write.table (go\_logFC2\_TE, "logFC2\_TE\_all.txt", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = ";")

##geneID and GO annotation interaction genes

geneID\_interactiongenes = oreochromis[oreochromis$Locus %in% logFC\_adjP\_interactiongenes$Row.names,]

geneID\_interactiongenes\_unique = subset(geneID\_interactiongenes, !duplicated(geneID\_interactiongenes$GeneID))

go\_interactiongenes = subset(go\_all\_goID\_biologicalprocess\_unique,go\_all\_goID\_biologicalprocess\_unique$entrezgene %in% geneID\_interactiongenes\_unique$GeneID)

go\_interactiongenes = go\_interactiongenes[,3:5]

go\_geneID\_interactiongenes = merge(geneID\_interactiongenes\_unique, go\_interactiongenes, by.x = "GeneID", by.y = "entrezgene", all=TRUE)

go\_logFC\_interactiongenes = merge(logFC\_adjP\_interactiongenes, go\_geneID\_interactiongenes, by.x = "Row.names", by.y = "Locus", all=TRUE)

go\_logFC\_interactiongenes = go\_logFC\_interactiongenes[,-(6:11)]

write.table (go\_logFC\_interactiongenes, "go\_interactiongenes.txt", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = ";")

```

# GO enrichment analysis

Make a set of DE genes for every contrast and compare with all DE genen in all contrasts. We use CAMERA to perform the enrichment analysis.

```{r}

#enrichment analysis for O. nasuta

d\_allDE\_nasuta = d[allSignGenes\_Na,] #subset the DGElist to only the significant genes #6847

geneID\_allDE\_nasuta = oreochromis[oreochromis$Locus %in% allSignGenes\_Na,]

geneID\_allDE\_nasuta\_unique = subset(geneID\_allDE\_nasuta, !duplicated(geneID\_allDE\_nasuta$GeneID)) #6749

go\_all\_biologicalprocess = go\_all[go\_all$namespace\_1003 == 'biological\_process',]

go\_allDE\_nasuta = subset(go\_all\_goID\_biologicalprocess\_unique,go\_all\_goID\_biologicalprocess\_unique$entrezgene %in% geneID\_allDE\_nasuta\_unique$GeneID)

#to add locus names to geneID and GO, merge two dataframes

LocusID\_allDE\_nasuta = geneID\_allDE\_nasuta\_unique[,5:6] # get the colmuns with locus and geneID so that only the locus column is added to the GO dataframe

Locus\_geneID\_allDE\_nasuta = merge(LocusID\_allDE\_nasuta, go\_allDE\_nasuta, by.x = "GeneID", by.y = "entrezgene")

write.table (Locus\_geneID\_allDE\_nasuta, "Locus\_geneID\_allDE\_nasuta.txt", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = ";")

#create gene sets based on GO terms

UniqueGeneSets\_nasuta=unique(Locus\_geneID\_allDE\_nasuta$go\_id)

length(UniqueGeneSets\_nasuta)#1720

geneSetList\_nasuta=list()

for(i in 1:length(UniqueGeneSets\_nasuta)){

geneSetList\_nasuta[[i]] = Locus\_geneID\_allDE\_nasuta$Locus.x[Locus\_geneID\_allDE\_nasuta$go\_id==UniqueGeneSets\_nasuta[i]]

}

names(geneSetList\_nasuta)=UniqueGeneSets\_nasuta

#CEREBELLUM (not subsetted)

c2.ind\_nasuta <- ids2indices(geneSetList\_nasuta, rownames(d))

resCAMERA\_CENa = camera(d,index=c2.ind\_nasuta,design=design\_2way,contrast=LBrain[,3], inter.gene.cor = 0.05)

table(resCAMERA\_CENa$FDR < 0.05) #13

write.table (resCAMERA\_CENa, "resCAMERA\_CENa.txt", col.names = TRUE, row.names = TRUE, quote = FALSE, sep = ";")

#BRAINSTEM (not subsetted)

resCAMERA\_BSNa = camera(d,index=c2.ind\_nasuta,design=design\_2way,contrast=LBrain[,1], inter.gene.cor = 0.05)

table(resCAMERA\_BSNa$FDR < 0.05) #75

#DIENCEPHALON (not subsetted)

resCAMERA\_DINa = camera(d,index=c2.ind\_nasuta,design=design\_2way,contrast=LBrain[,5], inter.gene.cor = 0.05)

table(resCAMERA\_DINa$FDR < 0.05) #46

#OlFACTORY BULBS (not subsetted)

resCAMERA\_OBNa = camera(d,index=c2.ind\_nasuta,design=design\_2way,contrast=LBrain[,7], inter.gene.cor = 0.05)

table(resCAMERA\_OBNa$FDR < 0.05) #15

#OPTIC TECTUM (not subsetted)

resCAMERA\_OTNa = camera(d,index=c2.ind\_nasuta,design=design\_2way,contrast=LBrain[,9], inter.gene.cor = 0.05)

table(resCAMERA\_OTNa$FDR < 0.05) #69

#TELENCEPHALON (not subsetted)

resCAMERA\_TENa = camera(d,index=c2.ind\_nasuta,design=design\_2way,contrast=LBrain[,11], inter.gene.cor = 0.05)

table(resCAMERA\_TENa$FDR < 0.05) #16

#enrichment analysis for 0. ventralis

d\_allDE\_ventralis = d[allSignGenes\_Ve,] #subset the DGElist to only the significant genes #7098

geneID\_allDE\_ventralis = oreochromis[oreochromis$Locus %in% allSignGenes\_Ve,]

geneID\_allDE\_ventralis\_unique = subset(geneID\_allDE\_ventralis, !duplicated(geneID\_allDE\_ventralis$GeneID)) #6967

go\_all\_biologicalprocess = go\_all[go\_all$namespace\_1003 == 'biological\_process',]

go\_allDE\_ventralis = subset(go\_all\_goID\_biologicalprocess\_unique,go\_all\_goID\_biologicalprocess\_unique$entrezgene %in% geneID\_allDE\_ventralis\_unique$GeneID)

#to add locus names to geneID and GO, merge two dataframes

LocusID\_allDE\_ventralis = geneID\_allDE\_ventralis\_unique[,5:6] # get the colmuns with locus and geneID so that only the locus column is added to the GO dataframe

Locus\_geneID\_allDE\_ventralis = merge(LocusID\_allDE\_ventralis, go\_allDE\_ventralis, by.x = "GeneID", by.y = "entrezgene")

write.table (Locus\_geneID\_allDE\_ventralis, "Locus\_geneID\_allDE\_ventralis.txt", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = ";")

#create gene sets based on GO terms

UniqueGeneSets\_ventralis=unique(Locus\_geneID\_allDE\_ventralis$go\_id)

length(UniqueGeneSets\_ventralis) #1734

geneSetList\_ventralis=list()

for(i in 1:length(UniqueGeneSets\_ventralis)){

geneSetList\_ventralis[[i]] = Locus\_geneID\_allDE\_ventralis$Locus[Locus\_geneID\_allDE\_ventralis$go\_id==UniqueGeneSets\_ventralis[i]]

}

names(geneSetList\_ventralis)=UniqueGeneSets\_ventralis

#CEREBELLUM (not subsetted)

c2.ind\_ventralis <- ids2indices(geneSetList\_ventralis, rownames(d))

resCAMERA\_CEVe = camera(d,index=c2.ind\_ventralis,design=design\_2way,contrast=LBrain[,4], inter.gene.cor = 0.05)

table(resCAMERA\_CEVe$FDR < 0.05) #8

write.table (resCAMERA\_CEVe, "resCAMERA\_CEVe.txt", col.names = TRUE, row.names = TRUE, quote = FALSE, sep = ";")

#BRAINSTEM (not subsetted)

resCAMERA\_BSVe = camera(d,index=c2.ind\_ventralis,design=design\_2way,contrast=LBrain[,2], inter.gene.cor = 0.05)

table(resCAMERA\_BSVe$FDR < 0.05) #64

#DIENCEPHALON (not subsetted)

resCAMERA\_DIVe = camera(d,index=c2.ind\_ventralis,design=design\_2way,contrast=LBrain[,6], inter.gene.cor = 0.05)

table(resCAMERA\_DIVe$FDR < 0.05) #41

#OlFACTORY BULBS (not subsetted)

resCAMERA\_OBVe = camera(d,index=c2.ind\_ventralis,design=design\_2way,contrast=LBrain[,8], inter.gene.cor = 0.05)

table(resCAMERA\_OBVe$FDR < 0.05) #21

#OPTIC TECTUM (not subsetted)

resCAMERA\_OTVe = camera(d,index=c2.ind\_ventralis,design=design\_2way,contrast=LBrain[,10], inter.gene.cor = 0.05)

table(resCAMERA\_OTVe$FDR < 0.05) #72

#TELENCEPHALON (not subsetted)

resCAMERA\_TEVe = camera(d,index=c2.ind\_ventralis,design=design\_2way,contrast=LBrain[,12], inter.gene.cor = 0.05)

table(resCAMERA\_TEVe$FDR < 0.05) #20

```

#GOplot: visualisation of the enrichment analysis

https://cran.r-project.org/web/packages/GOplot/vignettes/GOplot\_vignette.html

```{r}

##for O. nasuta

geneSetList\_nasuta\_df = data.frame(GO\_ID=names(geneSetList\_nasuta),genes=unlist(lapply(geneSetList\_nasuta,paste,collapse=","))) #convert the genesetlist two two columns

geneSetList\_nasuta\_df = geneSetList\_nasuta\_df[order(geneSetList\_nasuta\_df$GO\_ID),]#contains all significant genes of all brain parts per GO term, but does not contain the GO term itself. It does contain GO ID

geneSetList\_GO\_nasuta = subset(go\_all\_goID\_biologicalprocess\_unique, go\_all\_goID\_biologicalprocess\_unique$go\_id %in% geneSetList\_nasuta\_df$GO\_ID)

geneSetList\_GO\_nasuta\_unique = subset(geneSetList\_GO\_nasuta,!duplicated(geneSetList\_GO\_nasuta$go\_id))

geneSetList\_GO\_nasuta\_unique = geneSetList\_GO\_nasuta\_unique[order(geneSetList\_GO\_nasuta\_unique$go\_id),]#sort based on GO\_ID

#make input file for GOplot for TENa

resCAMERA\_TENa$GO\_ID = rownames(resCAMERA\_TENa)

resCAMERA\_TENa = resCAMERA\_TENa[order(resCAMERA\_TENa$GO\_ID),] # sort based on GO\_ID

goplot\_TENa = cbind(resCAMERA\_TENa,geneSetList\_nasuta\_df, geneSetList\_GO\_nasuta\_unique$name\_1006)#merges results from CAMERA for TE, genes in the GO lists and the GO terms (required for GO plot)

goplot\_TENa$Category = "BP"

goplot\_TENa = goplot\_TENa[,c(5,7:10)]

goplot\_TENa = goplot\_TENa[,c(5,2,4,3,1)]

colnames(goplot\_TENa) = c("category", "ID", "term", "genes", "adj\_pval")

#get FC for all genes in TE

Hyp1AllGenes=rownames(resBrain)

colnames(LBrain)

logFC\_All\_TEavgNa = confirmationResultsHyp1[[11]]$table[Hyp1AllGenes,]

logFC\_All\_TEavgNa$ID = rownames(logFC\_All\_TEavgNa)

#make input file for GOplot for DINa

resCAMERA\_DINa$GO\_ID = rownames(resCAMERA\_DINa)

resCAMERA\_DINa = resCAMERA\_DINa[order(resCAMERA\_DINa$GO\_ID),]

goplot\_DINa = cbind(resCAMERA\_DINa,geneSetList\_nasuta\_df, geneSetList\_GO\_nasuta\_unique$name\_1006)

goplot\_DINa$Category = "BP"

goplot\_DINa = goplot\_DINa[,c(5,7:10)]

goplot\_DINa = goplot\_DINa[,c(5,2,4,3,1)]

colnames(goplot\_DINa) = c("category", "ID", "term", "genes", "adj\_pval")

colnames(LBrain)

logFC\_All\_DIavgNa = confirmationResultsHyp1[[5]]$table[Hyp1AllGenes,]

logFC\_All\_DIavgNa$ID = rownames(logFC\_All\_DIavgNa)

#make input file for GOplot for CENa

resCAMERA\_CENa$GO\_ID = rownames(resCAMERA\_CENa)

resCAMERA\_CENa = resCAMERA\_CENa[order(resCAMERA\_CENa$GO\_ID),]

goplot\_CENa = cbind(resCAMERA\_CENa,geneSetList\_nasuta\_df, geneSetList\_GO\_nasuta\_unique$name\_1006)

goplot\_CENa$Category = "BP"

goplot\_CENa = goplot\_CENa[,c(5,7:10)]

goplot\_CENa = goplot\_CENa[,c(5,2,4,3,1)]

colnames(goplot\_CENa) = c("category", "ID", "term", "genes", "adj\_pval")

colnames(LBrain)

logFC\_All\_CEavgNa = confirmationResultsHyp1[[3]]$table[Hyp1AllGenes,]

logFC\_All\_CEavgNa$ID = rownames(logFC\_All\_CEavgNa)

#make input file for GOplot for OBNa

resCAMERA\_OBNa$GO\_ID = rownames(resCAMERA\_OBNa)

resCAMERA\_OBNa = resCAMERA\_OBNa[order(resCAMERA\_OBNa$GO\_ID),]

goplot\_OBNa = cbind(resCAMERA\_OBNa,geneSetList\_nasuta\_df, geneSetList\_GO\_nasuta\_unique$name\_1006)

goplot\_OBNa$Category = "BP"

goplot\_OBNa = goplot\_OBNa[,c(5,7:10)]

goplot\_OBNa = goplot\_OBNa[,c(5,2,4,3,1)]

colnames(goplot\_OBNa) = c("category", "ID", "term", "genes", "adj\_pval")

colnames(LBrain)

logFC\_All\_OBavgNa = confirmationResultsHyp1[[7]]$table[Hyp1AllGenes,]

logFC\_All\_OBavgNa$ID = rownames(logFC\_All\_OBavgNa)

#make input file for GOplot for OTNa

resCAMERA\_OTNa$GO\_ID = rownames(resCAMERA\_OTNa)

resCAMERA\_OTNa = resCAMERA\_OTNa[order(resCAMERA\_OTNa$GO\_ID),]

goplot\_OTNa = cbind(resCAMERA\_OTNa,geneSetList\_nasuta\_df, geneSetList\_GO\_nasuta\_unique$name\_1006)

goplot\_OTNa$Category = "BP"

goplot\_OTNa = goplot\_OTNa[,c(5,7:10)]

goplot\_OTNa = goplot\_OTNa[,c(5,2,4,3,1)]

colnames(goplot\_OTNa) = c("category", "ID", "term", "genes", "adj\_pval")

colnames(LBrain)

logFC\_All\_OTavgNa = confirmationResultsHyp1[[9]]$table[Hyp1AllGenes,]

logFC\_All\_OTavgNa$ID = rownames(logFC\_All\_OTavgNa)

#make input file for GOplot for BSNa

resCAMERA\_BSNa$GO\_ID = rownames(resCAMERA\_BSNa)

resCAMERA\_BSNa = resCAMERA\_BSNa[order(resCAMERA\_BSNa$GO\_ID),]

goplot\_BSNa = cbind(resCAMERA\_BSNa,geneSetList\_nasuta\_df, geneSetList\_GO\_nasuta\_unique$name\_1006)

goplot\_BSNa$Category = "BP"

goplot\_BSNa = goplot\_BSNa[,c(5,7:10)]

goplot\_BSNa = goplot\_BSNa[,c(5,2,4,3,1)]

colnames(goplot\_BSNa) = c("category", "ID", "term", "genes", "adj\_pval")

colnames(LBrain)

logFC\_All\_BSavgNa = confirmationResultsHyp1[[1]]$table[Hyp1AllGenes,]

logFC\_All\_BSavgNa$ID = rownames(logFC\_All\_BSavgNa)

##for O. ventralis

geneSetList\_ventralis\_df = data.frame(GO\_ID=names(geneSetList\_ventralis),genes=unlist(lapply(geneSetList\_ventralis,paste,collapse=","))) #convert the genesetlist two two columns

geneSetList\_ventralis\_df = geneSetList\_ventralis\_df[order(geneSetList\_ventralis\_df$GO\_ID),]

geneSetList\_GO\_ventralis = subset(go\_all\_goID\_biologicalprocess\_unique, go\_all\_goID\_biologicalprocess\_unique$go\_id %in% geneSetList\_ventralis\_df$GO\_ID)

geneSetList\_GO\_ventralis\_unique = subset(geneSetList\_GO\_ventralis,!duplicated(geneSetList\_GO\_ventralis$go\_id))

geneSetList\_GO\_ventralis\_unique = geneSetList\_GO\_ventralis\_unique[order(geneSetList\_GO\_ventralis\_unique$go\_id),]#sort based on GO\_ID

#make input file for GOplot for TENa

resCAMERA\_TEVe$GO\_ID = rownames(resCAMERA\_TEVe)

resCAMERA\_TEVe = resCAMERA\_TEVe[order(resCAMERA\_TEVe$GO\_ID),] # sort based on GO\_ID

goplot\_TEVe = cbind(resCAMERA\_TEVe,geneSetList\_ventralis\_df, geneSetList\_GO\_ventralis\_unique$name\_1006)

goplot\_TEVe$Category = "BP"

goplot\_TEVe = goplot\_TEVe[,c(5,7:10)]

goplot\_TEVe = goplot\_TEVe[,c(5,2,4,3,1)]

colnames(goplot\_TEVe) = c("category", "ID", "term", "genes", "adj\_pval")

#get FC for all genes in TE

Hyp1AllGenes=rownames(resBrain)

colnames(LBrain)#om te zien welke lijst te selecteren

logFC\_All\_TEavgVe = confirmationResultsHyp1[[12]]$table[Hyp1AllGenes,]

logFC\_All\_TEavgVe$ID = rownames(logFC\_All\_TEavgVe)

#make input file for GOplot for DIVe

resCAMERA\_DIVe$GO\_ID = rownames(resCAMERA\_DIVe)

resCAMERA\_DIVe = resCAMERA\_DIVe[order(resCAMERA\_DIVe$GO\_ID),]

goplot\_DIVe = cbind(resCAMERA\_DIVe,geneSetList\_ventralis\_df, geneSetList\_GO\_ventralis\_unique$name\_1006)

goplot\_DIVe$Category = "BP"

goplot\_DIVe = goplot\_DIVe[,c(5,7:10)]

goplot\_DIVe = goplot\_DIVe[,c(5,2,4,3,1)]

colnames(goplot\_DIVe) = c("category", "ID", "term", "genes", "adj\_pval")

colnames(LBrain)

logFC\_All\_DIavgVe = confirmationResultsHyp1[[6]]$table[Hyp1AllGenes,]

logFC\_All\_DIavgVe$ID = rownames(logFC\_All\_DIavgVe)

#make input file for GOplot for CEVe

resCAMERA\_CEVe$GO\_ID = rownames(resCAMERA\_CEVe)

resCAMERA\_CEVe = resCAMERA\_CEVe[order(resCAMERA\_CEVe$GO\_ID),]

goplot\_CEVe = cbind(resCAMERA\_CEVe,geneSetList\_ventralis\_df, geneSetList\_GO\_ventralis\_unique$name\_1006)

goplot\_CEVe$Category = "BP"

goplot\_CEVe = goplot\_CEVe[,c(5,7:10)]

goplot\_CEVe = goplot\_CEVe[,c(5,2,4,3,1)]

colnames(goplot\_CEVe) = c("category", "ID", "term", "genes", "adj\_pval")

colnames(LBrain)

logFC\_All\_CEavgVe = confirmationResultsHyp1[[4]]$table[Hyp1AllGenes,]

logFC\_All\_CEavgVe$ID = rownames(logFC\_All\_CEavgVe)

#make input file for GOplot for OBVe

resCAMERA\_OBVe$GO\_ID = rownames(resCAMERA\_OBVe)

resCAMERA\_OBVe = resCAMERA\_OBVe[order(resCAMERA\_OBVe$GO\_ID),]

goplot\_OBVe = cbind(resCAMERA\_OBVe,geneSetList\_ventralis\_df, geneSetList\_GO\_ventralis\_unique$name\_1006)

goplot\_OBVe$Category = "BP"

goplot\_OBVe = goplot\_OBVe[,c(5,7:10)]

goplot\_OBVe = goplot\_OBVe[,c(5,2,4,3,1)]

colnames(goplot\_OBVe) = c("category", "ID", "term", "genes", "adj\_pval")

colnames(LBrain)

logFC\_All\_OBavgVe = confirmationResultsHyp1[[8]]$table[Hyp1AllGenes,]

logFC\_All\_OBavgVe$ID = rownames(logFC\_All\_OBavgVe)

#make input file for GOplot for OTVe

resCAMERA\_OTVe$GO\_ID = rownames(resCAMERA\_OTVe)

resCAMERA\_OTVe = resCAMERA\_OTVe[order(resCAMERA\_OTVe$GO\_ID),]

goplot\_OTVe = cbind(resCAMERA\_OTVe,geneSetList\_ventralis\_df, geneSetList\_GO\_ventralis\_unique$name\_1006)

goplot\_OTVe$Category = "BP"

goplot\_OTVe = goplot\_OTVe[,c(5,7:10)]

goplot\_OTVe = goplot\_OTVe[,c(5,2,4,3,1)]

colnames(goplot\_OTVe) = c("category", "ID", "term", "genes", "adj\_pval")

colnames(LBrain)

logFC\_All\_OTavgVe = confirmationResultsHyp1[[10]]$table[Hyp1AllGenes,]

logFC\_All\_OTavgVe$ID = rownames(logFC\_All\_OTavgVe)

#make input file for GOplot for BSVe

resCAMERA\_BSVe$GO\_ID = rownames(resCAMERA\_BSVe)

resCAMERA\_BSVe = resCAMERA\_BSVe[order(resCAMERA\_BSVe$GO\_ID),]

goplot\_BSVe = cbind(resCAMERA\_BSVe,geneSetList\_ventralis\_df, geneSetList\_GO\_ventralis\_unique$name\_1006)

goplot\_BSVe$Category = "BP"

goplot\_BSVe = goplot\_BSVe[,c(5,7:10)]

goplot\_BSVe = goplot\_BSVe[,c(5,2,4,3,1)]

colnames(goplot\_BSVe) = c("category", "ID", "term", "genes", "adj\_pval")

colnames(LBrain)

logFC\_All\_BSavgVe = confirmationResultsHyp1[[2]]$table[Hyp1AllGenes,]

logFC\_All\_BSavgVe$ID = rownames(logFC\_All\_BSavgVe)

library(GOplot)

circ\_TENa = circle\_dat(goplot\_TENa, logFC\_All\_TEavgNa) #column names need to be exactly as defined above - no capital letter

circ\_DINa = circle\_dat(goplot\_DINa, logFC\_All\_DIavgNa)

circ\_CENa = circle\_dat(goplot\_CENa, logFC\_All\_CEavgNa)

circ\_OBNa = circle\_dat(goplot\_OBNa, logFC\_All\_OBavgNa)

circ\_OTNa = circle\_dat(goplot\_OTNa, logFC\_All\_OTavgNa)

circ\_BSNa = circle\_dat(goplot\_BSNa, logFC\_All\_BSavgNa)

circ\_TEVe = circle\_dat(goplot\_TEVe, logFC\_All\_TEavgVe)

circ\_DIVe = circle\_dat(goplot\_DIVe, logFC\_All\_DIavgVe)

circ\_CEVe = circle\_dat(goplot\_CEVe, logFC\_All\_CEavgVe)

circ\_OBVe = circle\_dat(goplot\_OBVe, logFC\_All\_OBavgVe)

circ\_OTVe = circle\_dat(goplot\_OTVe, logFC\_All\_OTavgVe)

circ\_BSVe = circle\_dat(goplot\_BSVe, logFC\_All\_BSavgVe)

library(ggrepel)

pdf("GObubble\_TENa.pdf", width = 7, height = 5)

GOBubble(circ\_TENa, labels = 1.4, title = "TE\_Na", table.legend = FALSE)

dev.off()

pdf("GObubble\_DINa.pdf", width = 7, height = 5)

GOBubble(circ\_DINa, labels = 1.4, title = "DI\_Na", table.legend = FALSE)

dev.off()

pdf("GObubble\_CENa.pdf", width = 7, height = 5)

GOBubble(circ\_CENa, labels = 1.4, title = "CE\_Na", table.legend = FALSE)

dev.off()

pdf("GObubble\_OBNa.pdf", width = 7, height = 5)

GOBubble(circ\_OBNa, labels = 1.4, title = "OB\_Na", table.legend = FALSE)

dev.off()

pdf("GObubble\_OTNa.pdf", width = 7, height = 5)

GOBubble(circ\_OTNa, labels = 1.4, title = "OT\_Na", table.legend = FALSE)

dev.off()

pdf("GObubble\_BSVe.pdf", width = 7, height = 5)

GOBubble(circ\_BSVe, labels = 1.4, title = "BS\_Ve", table.legend = FALSE)

dev.off()

pdf("GObubble\_TEVe.pdf", width = 7, height = 5)

GOBubble(circ\_TEVe, labels = 1.4, title = "TE\_Ve", table.legend = FALSE)

dev.off()

pdf("GObubble\_DIVe.pdf", width = 7, height = 5)

GOBubble(circ\_DIVe, labels = 1.4, title = "DI\_Ve", table.legend = FALSE)

dev.off()

pdf("GObubble\_CEVe.pdf", width = 7, height = 5)

GOBubble(circ\_CEVe, labels = 1.4, title = "CE\_Ve", table.legend = FALSE)

dev.off()

pdf("GObubble\_OBVe.pdf", width = 7, height = 5)

GOBubble(circ\_OBVe, labels = 1.4, title = "OB\_Ve", table.legend = FALSE)

dev.off()

pdf("GObubble\_OTVe.pdf", width = 7, height = 5)

GOBubble(circ\_OTVe, labels = 1.4, title = "OT\_Ve", table.legend = FALSE)

dev.off()

pdf("GObubble\_BSVe.pdf", width = 7, height = 5)

GOBubble(circ\_BSVe, labels = 1.4, title = "BS\_Ve", table.legend = FALSE)

dev.off()

#to sort the circ files to get top upregulated go terms:

circ\_BSNa\_ordered = subset(circ\_BSNa, circ\_BSNa$adj\_pval < 0.05)

circ\_OTNa\_ordered = subset(circ\_OTNa, circ\_OTNa$adj\_pval < 0.05)

circ\_DINa\_ordered = subset(circ\_DINa, circ\_DINa$adj\_pval < 0.05)

circ\_CENa\_ordered = subset(circ\_CENa, circ\_CENa$adj\_pval < 0.05)

circ\_TENa\_ordered = subset(circ\_TENa, circ\_TENa$adj\_pval < 0.05)

circ\_OBNa\_ordered = subset(circ\_OBNa, circ\_OBNa$adj\_pval < 0.05)

circ\_BSVe\_ordered = subset(circ\_BSVe, circ\_BSVe$adj\_pval < 0.05)

circ\_OTVe\_ordered = subset(circ\_OTVe, circ\_OTVe$adj\_pval < 0.05)

circ\_DIVe\_ordered = subset(circ\_DIVe, circ\_DIVe$adj\_pval < 0.05)

circ\_CEVe\_ordered = subset(circ\_CEVe, circ\_CEVe$adj\_pval < 0.05)

circ\_TEVe\_ordered = subset(circ\_TEVe, circ\_TEVe$adj\_pval < 0.05)

circ\_OBVe\_ordered = subset(circ\_OBVe, circ\_OBVe$adj\_pval < 0.05)

```

#Summary table CAMERA

```{r}

summary\_camera\_nasuta = data.frame(geneSetList\_GO\_nasuta\_unique$name\_1006, geneSetList\_GO\_nasuta\_unique$go\_id, resCAMERA\_BSNa$GO\_ID, resCAMERA\_BSNa$FDR, resCAMERA\_TENa$FDR, resCAMERA\_OTNa$FDR, resCAMERA\_OBNa$FDR, resCAMERA\_DINa$FDR, resCAMERA\_CENa$FDR)

colnames(summary\_camera\_nasuta) = c("Term", "go\_ID", "GO\_ID", "BSNa", "TENa", "OTNa", "OBNa", "DINa"," CENa")

write.table (summary\_nasuta\_camera, "Summary\_CAMERA\_Nasuta.txt", col.names = TRUE,row.names = FALSE, quote = FALSE, sep = ";")

summary\_camera\_ventralis = data.frame(geneSetList\_GO\_ventralis\_unique$name\_1006, geneSetList\_GO\_ventralis\_unique$go\_id, resCAMERA\_BSVe$GO\_ID, resCAMERA\_BSVe$FDR, resCAMERA\_TEVe$FDR, resCAMERA\_OTVe$FDR, resCAMERA\_OBVe$FDR, resCAMERA\_DIVe$FDR, resCAMERA\_CEVe$FDR)

colnames(summary\_camera\_ventralis) = c("Term", "go\_ID", "GO\_ID", "BSVe", "TEVe", "OTVe", "OBVe", "DIVe"," CEVe")

write.table (summary\_camera\_ventralis, "Summary\_CAMERA\_Ventralis.txt", col.names = TRUE,row.names = FALSE, quote = FALSE, sep = ";")

summary\_camera = merge(summary\_camera\_nasuta, summary\_camera\_ventralis, by.x = "GO\_ID", by.y = "GO\_ID", all.x = TRUE, all.y = TRUE)

write.table (summary\_camera, "Summary\_CAMERA.txt", col.names = TRUE,row.names = FALSE, quote = FALSE, sep = ";")

summary\_camera\_sign = subset(summary\_camera, summary\_camera$BSNa < 0.05 | summary\_camera$TENa < 0.05 | summary\_camera$OTNa < 0.05 | summary\_camera$OBNa < 0.05 | summary\_camera$DINa < 0.05 | summary\_camera$CENa < 0.05 | summary\_camera$BSVe < 0.05 | summary\_camera$TEVe < 0.05 | summary\_camera$OTVe < 0.05 | summary\_camera$OBVe < 0.05 | summary\_camera$DIVe < 0.05 | summary\_camera$CEVe < 0.05, na.rm = TRUE)

```