Co-expression analysis tool

Introduction

A basic tool to check co-expression between two specific genes in the brainspan dataset.

```
#loading packages and data
library("ggplot2")
library("ggpubr")

expression_data <- read.csv("expression_matrix.csv", header = FALSE)
column_data <- read.csv("columns_metadata.csv")
row_data <- read.csv("rows_metadata.csv")
```

Settings

```
#settings. age in pcw, gene in ensembl gene id (and structure(s) not yet implemented)

from <- "19 pcw"
to <- "26 pcw"

gene1 <- "ENSG00000103512"
gene2 <- "ENSG00000149929"

iterations <- 10000

structures <- c("DFC","VFC","MFC","OFC","M1C-S1C","S1C","M1C","A1C","IPC","STC","ITC","V1C")
```

Data scrubbing

```
#filter out useless samples/genes based on entrez ID & assign sample number
expression_data <- expression_data[-1];
rownames(expression_data) <- row_data[,3]
expression_data$entrez_id <- row_data[,5]
expression_data <- expression_data[!is.na(expression_data$entrez_id), ]
row_data <- row_data[!is.na(row_data$entrez_id), ]
expression_data<- expression_data[ , -which(names(expression_data) %in% c("entrez_id"))]
expression_data.t <- t(expression_data)
expression_data.t <- as.data.frame(expression_data.t)
rownames(expression_data.t) <- paste0("S", 1:nrow(expression_data.t))
rownames(column_data) <- paste0("S", 1:nrow(column_data))
```

Filter for specified period

Start consecutive analyses here.

```
samples.num.from <- expression_data.t[column_data$age %in% c(from),]
samples.num.to <- expression_data.t[column_data$age %in% c(to),]
```

```
num.from <- rownames(samples.num.from[c(1),])
num.to <- rownames(samples.num.to[c(nrow(samples.num.to)),])

expression_data.age <- expression_data.t[c(which(rownames(expression_data.t) == num.from):which(rowname

column_data.age <- column_data[c(which(rownames(expression_data.t) == num.from):which(rownames(expressi
```

Filter on specified structures

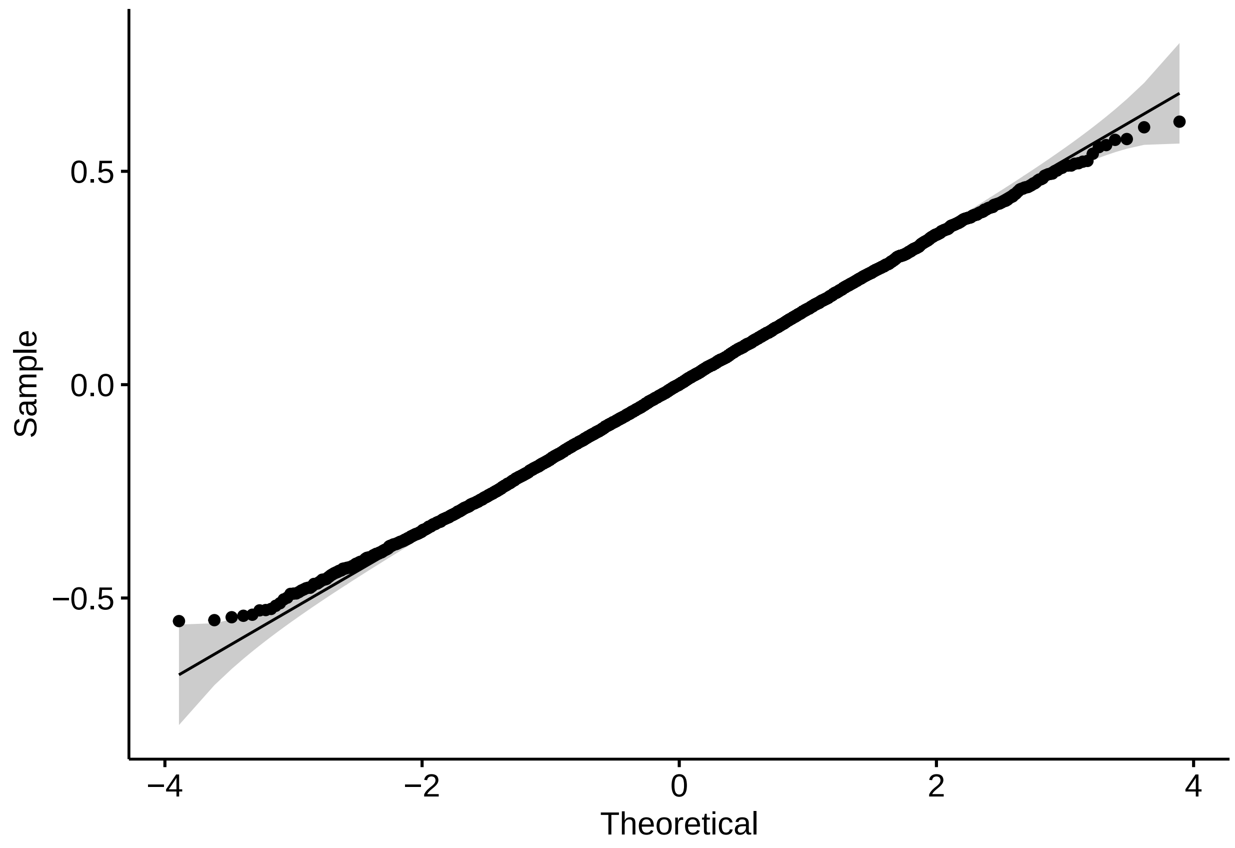
```
expression_data.age.structure <- expression_data.age[column_data.age$structure_acronym %in% structures,]
```

Isolate specified genes

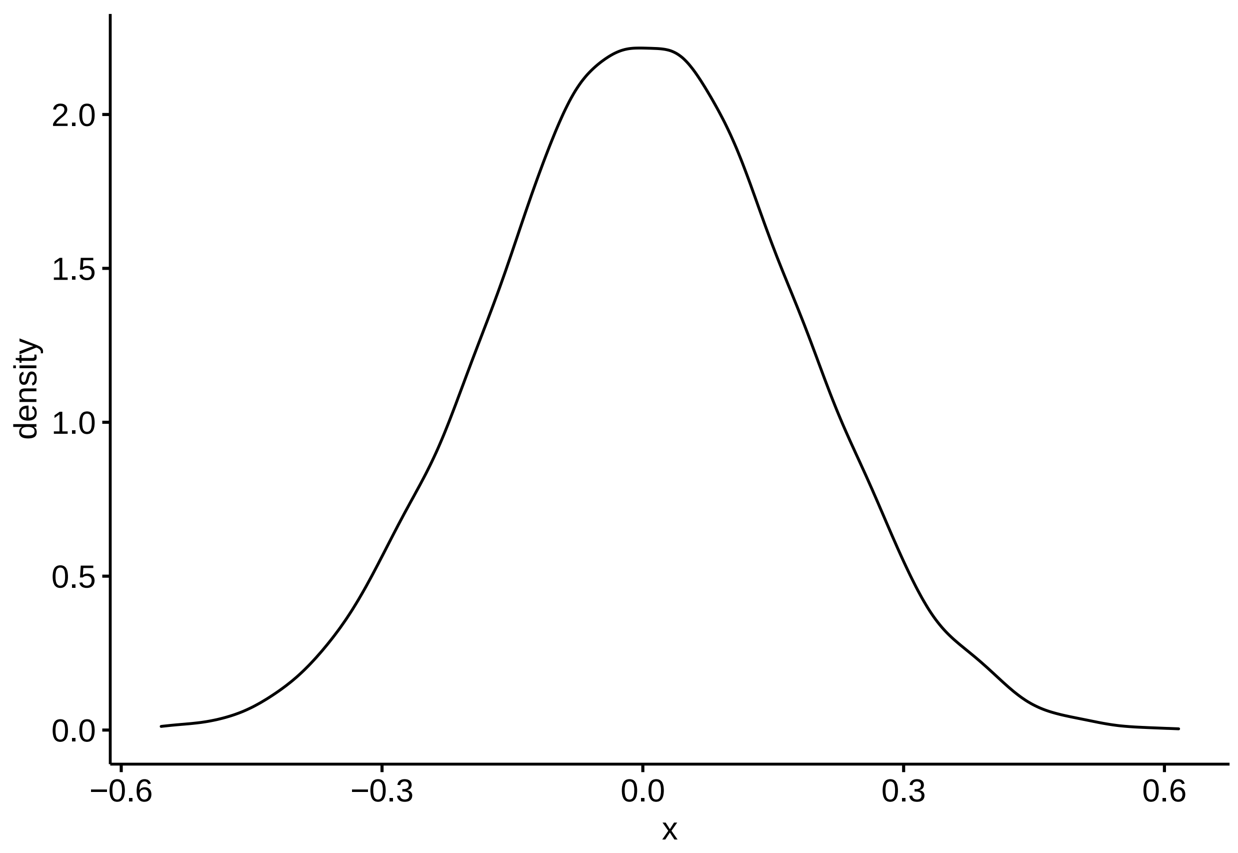
```
#function to filter genes of interest based on ensembl gene id and log transformation

expression_data.goi1 <- as.data.frame(t(log(expression_data.age.structure[row_data$ensembl_gene_id %in%
expression_data.goi2 <- as.data.frame(t(log(expression_data.age.structure[row_data$ensembl_gene_id %in%
```

Correlation calculation / permutation stipulation

```
#calculate pearson correlation between genes of interest

pearson.cor <- cor(t(expression_data.goi1), t(expression_data.goi2),
method = "pearson")

#permutation analysis between genes of interest

permutation_data.goi1.goi2 <- c();

for (i in 1: iterations) {
columns <- sample(ncol(expression_data.goi1));
expression_data.goi1.ran <- expression_data.goi1[,columns];
columns <- sample(ncol(expression_data.goi2));
expression_data.goi2.ran <- expression_data.goi2[,columns];
cormat_goi1_goi2.ran <- cor(t(expression_data.goi1.ran), t(expression_data.goi2.ran), method = "pearson
permutation_data.goi1.goi2 <- append(permutation_data.goi1.goi2, cormat_goi1_goi2.ran[1]) ;
}

#quickly check distribution

ggqqplot(permutation_data.goi1.goi2)
```



```r
ggdensity(permutation_data.goi1.goi2)
```



```r
#calculate p from z

z <- (pearson.cor[1]-mean(permutation_data.goi1.goi2)) / (sd(permutation_data.goi1.goi2))
p.goi1.goi2 = 2*pnorm(-abs(z))

#print correlation and associated p-value

print(pearson.cor[1])
```

```
## [1] -0.7975252
```

```r
print(p.goi1.goi2)
```

```
## [1] 4.247827e-06
```