ThesisTesting_21.01

Version info: R 3.2.3, Biobase 2.30.0, GEOquery 2.40.0, limma 3.26.8

0. Setup

Special GEOquery package

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("GEOquery")
```

Other Packages

```
install.packages('limma')
install.packages('umap')
install.packages("reticulate")
install.packages("homologene")
install.packages('ggplot2')
install.packages('dplyr')
install.packages('svglite')
```

Open Libraries

Install Python Packages

```
py_install("pandas")
py_install("regex")
py_install("seaborn")
```

Global Options (not in use)

Python Packages

```
import pandas as pd
import regex as re
import seaborn as sns
import matplotlib.pyplot as plt
```

1. Dataset Management (Python)

NCBI GEO was searched for expression datasets on 03/02/2021, filtered to mammalian species with age sets, with 221 results. Results were exported as a detailed text file, then parsed into a dataframe using Python below.

```
pattern_organism = re.compile(r"\bOrganism\w*\b")
pattern_GSE = re.compile(r"\bPlatform\w*\b")

list_organism = []
list_GSE = []
list_platform = []
with open("gds_result.txt","rt") as myfile:
    for line in myfile:
        if pattern_organism.search(line) != None:
            betterline = line.replace("Organism:\t","")
            bestline = betterline.replace("\n","")
            list_organism.append(bestline)
        if pattern_GSE.search(line) != None:
            GSE = line.split("Series: ")[1].rsplit()[0]
            list_GSE.append(GSE)
            Platform = line.split("Platform: ")[1].rsplit()[0]
            list_platform.append(Platform)
list_of_lists = [list_GSE, list_organism, list_platform]
df_of_lists = pd.DataFrame(list_of_lists).transpose()
df_of_lists.columns = ["GSE","Organism","Platform"]
print(df_of_lists)
```

```
##           GSE      Organism Platform
## 0    GSE71868  Mus musculus  GPL6885
## 1    GSE65927  Mus musculus  GPL1261
## 2    GSE62173  Mus musculus  GPL1261
## 3    GSE65063  Mus musculus  GPL7202
## 4    GSE52509  Mus musculus  GPL6885
## ..        ...           ...      ...
## 216    GSE459  Mus musculus    GPL76
## 217    GSE459  Mus musculus    GPL75
## 218    GSE362  Homo sapiens    GPL97
## 219    GSE362  Homo sapiens    GPL96
## 220     GSE80  Homo sapiens    GPL91
##
## [221 rows x 3 columns]
```

2. Dataset Management (Manual)

Each of the 221 candidate datasets are manually reviewed. [IN PROGRESS]

Datasets were excluded if two adult age-sets are not available, such as if the study is investigating a developmental process in the embryonic, postnatal, or juvenile period. Specifically, humans under 18yo, rats under 6mo, and mice under 3mo were considered juvenile and avoided in this analysis. Exceptions: Study GSE5666 compared 4mo and 28mo old rats in rarely obtainable tissue types (heart and white adipose tissue), so this study was included.

When more than two age-sets were available – such as "young," "middle," and "old" – comparisons between "middle" and "old" were prioritized. When a continuous set of ages were available, buffer regions were implemented, e.g. human age 30yo-60yo vs 70yo-90yo with 61-69yo excluded as a buffer region.

Other, less common reasons were exclusion included datasets where control/untreated/wild-type groups were not available or when "age-sets" were actually time-points in short time-course experiments.

For all remaining datasets, the samples were categorized as young (0), old (1), or excluded (X) for the purposes of further processing. In addition, the tissue type under study was manually noted.

3. Calculate DEGs in each Dataset (R)

3.1 Set Functions

GEO2R Function

```
GEO2R <- function(GSE_number,platform_code,group_binaries,outputs){
  # load series and platform data from GEO
  gset <- getGEO(GSE_number, GSEMatrix =TRUE, AnnotGPL=TRUE)
  if (length(gset) > 1) idx <- grep(platform_code, attr(gset, "names")) else idx <- 1
  gset <- gset[[idx]]

  # make proper column names to match toptable
  fvarLabels(gset) <- make.names(fvarLabels(gset))

  # group membership for all samples
  gsms <- group_binaries
  sml <- strsplit(gsms, split="")[[1]]

  # filter out excluded samples (marked as "X")
  sel <- which(sml != "X")
  sml <- sml[sel]
  gset <- gset[ ,sel]

  # log2 transformation
  ex <- exprs(gset)
  qx <- as.numeric(quantile(ex, c(0., 0.25, 0.5, 0.75, 0.99, 1.0), na.rm=T))
  LogC <- (qx[5] > 100) ||
            (qx[6]-qx[1] > 50 && qx[2] > 0)
  if (LogC) { ex[which(ex <= 0)] <- NaN
    exprs(gset) <- log2(ex) }

  # assign samples to groups and set up design matrix
  gs <- factor(sml)
  groups <- make.names(c("young","old"))
  levels(gs) <- groups
  gset$group <- gs
  design <- model.matrix(~group + 0, gset)
  colnames(design) <- levels(gs)

  fit <- lmFit(gset, design)  # fit linear model

  # set up contrasts of interest and recalculate model coefficients
```

```
  cts <- paste(groups[1], groups[2], sep="-")
  cont.matrix <- makeContrasts(contrasts=cts, levels=design)
  fit2 <- contrasts.fit(fit, cont.matrix)

  # compute statistics and table of top significant genes
  fit2 <- eBayes(fit2, 0.01)
  tT <- topTable(fit2, adjust="fdr", sort.by="B", number=outputs)

  tT <- subset(tT, select=c("ID","adj.P.Val","P.Value","t","B","logFC","Gene.symbol","Gene.title"))
  return(tT)
}
```

Function compounded to filter results by specified p value (change p value here)

```
GEO2R_pFiltered <- function(GSE_number,platform_code,group_binaries,outputs){
  tempTable <- GEO2R(GSE_number,platform_code,group_binaries,outputs)
  filtTable <- tempTable[tempTable$adj.P.Val<0.25,]
  dupTable <- filtTable[!duplicated(filtTable$Gene.symbol),]
  return(dupTable)
}
```

Set global number of outputs here

```
num_outputs = 5000
```

To prevent a vroom error, run this line before analyzing the data:

```
Sys.setenv(VROOM_CONNECTION_SIZE = 500000)
```

3.2 Import Datasets

DATASET CONTROL ZONE 1 OF 6 *(These zones must be updated if adding datasets.)*

Execute functions on human datasets

```
tT007 <- GEO2R_pFiltered("GSE58015","GPL571","000011111",num_outputs)
tT009 <- GEO2R_pFiltered("GSE53890","GPL570","XXXXXXXX00000000000XX11111111111XXXXXXXXX",num_outputs)
tT023 <- GEO2R_pFiltered("GSE38718","GPL570","00000000000000011111111",num_outputs)
tT036 <- GEO2R_pFiltered("GSE28422","GPL570","00000000XXXXXXXXXXXXXXXXXXXXXXX111111XXXXXXXXXXXXXXXXXX0
tT037 <- GEO2R_pFiltered("GSE28392","GPL570","00000XXXXXXXXXXXXXXXX1111XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
tT038 <- GEO2R_pFiltered("GSE25941","GPL570","00000000111111111110000001111111111",num_outputs)
tT044 <- GEO2R_pFiltered("GSE32719","GPL570","XXXXXXXXXXXXXX00000011111111",num_outputs)
```

Execute function on mouse datasets

```
tT000M <- GEO2R_pFiltered("GSE71868","GPL6885","11110000",num_outputs)
tT008M <- GEO2R_pFiltered("GSE50821","GPL1261","00000XXXX11111",num_outputs)
tT011M <- GEO2R_pFiltered("GSE55162","GPL1261","11110000",num_outputs)
tT014M <- GEO2R_pFiltered("GSE46646","GPL1261","XXXXXX000111",num_outputs)
tT020M <- GEO2R_pFiltered("GSE49543","GPL339","XXXXXXXXX00000000000000000011111111111111",num_outputs)
```

More mouse datasets

```
tT043aM <- GEO2R_pFiltered("GSE25905","GPL6246","000XXXXXXXXX111XXX",num_outputs)
tT043bM <- GEO2R_pFiltered("GSE25905","GPL6246","XXX000XXXXXXXXX111",num_outputs)
tT057M <- GEO2R_pFiltered("GSE27686","GPL1261","111XXXXXX000XXXX",num_outputs)
tT081M <- GEO2R_pFiltered("GSE19677","GPL6333","0000XXXXX1111XXXXX",num_outputs)
tT082M <- GEO2R_pFiltered("GSE19677","GPL1261","XXXX00001X1XX1XXX1",num_outputs)
tT133M <- GEO2R_pFiltered("GSE6323","GPL339","0000011111XXXXX",num_outputs)
```

More mouse datasets:

```
tT097M <- GEO2R_pFiltered("GSE11667","GPL1261","11110000XXXXXXXX",num_outputs)
tT112M <- GEO2R_pFiltered("GSE8150","GPL1261","0000011111XXXXXXXXXX",num_outputs)
tT113M <- GEO2R_pFiltered("GSE8146","GPL81","0000011111XXXXXXXXXX",num_outputs)
```

Non-mouse, non-human datasets:

```
tT058X <- GEO2R_pFiltered("GSE24515","GPL1355","000000111111",num_outputs)
tT088X <- GEO2R_pFiltered("GSE9990","GPL341","XXXXXXXXXXXXXXXXXXXXXXXXXXX000000000111111111111",num_ou
tT092X <- GEO2R_pFiltered("GSE12502","GPL3979","111111000000",num_outputs)

tT108aX <- GEO2R_pFiltered("GSE6718","GPL1355","1111111000000XXXXXXXXXXXXXXXXXXXXXXXXX",num_outputs)
tT108bX <- GEO2R_pFiltered("GSE6718","GPL1355","XXXXXXXXXXXXXXXXXXXXXX11000X00XXX11100X",num_outputs)
tT132X <- GEO2R_pFiltered("GSE4270","GPL890","XXXXXXXXXXXXXXX000000000011111111111111",num_outputs)
```

3.3 Non-human gene conversions

Convert non-human gene symbols to human gene symbols

Function for mouse:

```
M2H <- function(mouse_tT){
  MouseSymbols = mouse_tT[,7]
  M2H_Symbols = mouse2human(MouseSymbols)
  tTMerge = merge(mouse_tT,M2H_Symbols,by.x = "Gene.symbol",by.y = "mouseGene")
  tTReordered.dup = tTMerge[,c(2,3,4,5,6,7,9,8,1)]
  tTReordered <- tTReordered.dup[!duplicated(tTReordered.dup$humanGene),]
  return(tTReordered)
}
```

Function for non-mouse:

```
X2H <- function(animal_tT,TaxID){
  AnimalSymbols = animal_tT[,7]
  X2H_Symbols = homologene(AnimalSymbols, inTax = TaxID, outTax = 9606)
  taxonomy = toString(TaxID)
  tTMerge = merge(animal_tT,X2H_Symbols,by.x = "Gene.symbol",by.y = taxonomy)
  tTReordered.dup = tTMerge[,c(2,3,4,5,6,7,9,8,1)]
  tTReordered <- tTReordered.dup[!duplicated(tTReordered.dup$'9606'),]
  return(tTReordered)
}
```

DATASET CONTROL ZONE 2 OF 6 (NONHUMAN ONLY) Execution for mouse:

```
tT000 <- M2H(tT000M)
tT008 <- M2H(tT008M)
tT011 <- M2H(tT011M)
tT014 <- M2H(tT014M)
tT020 <- M2H(tT020M)
tT043a <- M2H(tT043aM)
tT043b <- M2H(tT043bM)
tT057 <- M2H(tT057M)
tT081 <- M2H(tT081M)
tT082 <- M2H(tT082M)
tT097 <- M2H(tT097M)
tT112 <- M2H(tT112M)
tT113 <- M2H(tT113M)
tT133 <- M2H(tT133M)
```

Execution for non-mouse:

```
tT058 <- X2H(tT058X,10116)
tT088 <- X2H(tT088X,10116)
tT092 <- X2H(tT092X,9615)

tT108a <- X2H(tT108aX,10116)
tT108b <- X2H(tT108bX,10116)
tT132 <- X2H(tT132X,10116)
```

Note: Some gene IDs appear in the results as GeneName1///GeneName2///GeneName3///. I noticed these were not successfully converted to human genes. I performed a test on one such gene where I ran each GeneName individually, and no results were found either, so I conclude homologous genes are not available for those entries regardless of any formatting issue. Record of this Test:

```
M1Test = c("LOC100503923","Gm15433","LOC100041903","Gm2666","Gm7609","Csprs")
M1Test
dfM1Test = mouse2human(M1Test)
dfM1Test #0 Results
```

3.4 Export Results

DATASET CONTROL ZONE 3 OF 6 Write to files

```
#Human:
write.table(tT007, file="tT007.txt", row.names=F, sep="\t")
write.table(tT009, file="tT009.txt", row.names=F, sep="\t")
write.table(tT023, file="tT023.txt", row.names=F, sep="\t")
write.table(tT036, file="tT036.txt", row.names=F, sep="\t")
write.table(tT037, file="tT037.txt", row.names=F, sep="\t")
write.table(tT038, file="tT038.txt", row.names=F, sep="\t")
write.table(tT044, file="tT044.txt", row.names=F, sep="\t")
#Mouse:
write.table(tT000, file="tT000.txt", row.names=F, sep="\t")
write.table(tT008, file="tT008.txt", row.names=F, sep="\t")
```

```
write.table(tT011, file="tT011.txt", row.names=F, sep="\t")
write.table(tT014, file="tT014.txt", row.names=F, sep="\t")
write.table(tT020, file="tT020.txt", row.names=F, sep="\t")
write.table(tT043a, file="tT043a.txt", row.names=F, sep="\t")
write.table(tT043b, file="tT043b.txt", row.names=F, sep="\t")
write.table(tT057, file="tT057.txt", row.names=F, sep="\t")
write.table(tT081, file="tT081.txt", row.names=F, sep="\t")
write.table(tT082, file="tT082.txt", row.names=F, sep="\t")
write.table(tT097, file="tT097.txt", row.names=F, sep="\t")
write.table(tT112, file="tT112.txt", row.names=F, sep="\t")
write.table(tT113, file="tT113.txt", row.names=F, sep="\t")
write.table(tT133, file="tT133.txt", row.names=F, sep="\t")
#Other:
write.table(tT058, file="tT058.txt", row.names=F, sep="\t")
write.table(tT088, file="tT088.txt", row.names=F, sep="\t")
write.table(tT092, file="tT092.txt", row.names=F, sep="\t")
write.table(tT108a, file="tT108a.txt", row.names=F, sep="\t")
write.table(tT108b, file="tT108b.txt", row.names=F, sep="\t")
write.table(tT132, file="tT132.txt", row.names=F, sep="\t")
```

4. Calculate Scores Across Datasets (Python)

4A. Compute Scores

Define Reader Function

```
def file_to_dict(file_name):
    file = open(file_name)
    header = file.readline()
    for line in file:
      row = line.strip().replace('"', '').split('\t')
      logFC = float(row[5])
      geneID = row[6]
      if logFC > 0:
        if geneID in young_dict:
          young_dict[geneID] += 1
          total_dict[geneID] += 1
        elif geneID in total_dict:
          total_dict[geneID] += 1
          young_dict[geneID] = 1
        else:
          young_dict[geneID] = 1
          total_dict[geneID] = 1
      if logFC < 0:
        if geneID in old_dict:
          old_dict[geneID] += 1
          total_dict[geneID] += (-1)
        elif geneID in total_dict:
          total_dict[geneID] += (-1)
          old_dict[geneID] = 1
        else:
```

```
            old_dict[geneID] = 1
            total_dict[geneID] = -1
    file.close()
```

DATASET CONTROL ZONE 4 OF 6 *(Exclude datasets from counts by commenting out here.)* Execute Reader Function

```
young_dict = {}
old_dict = {}
total_dict = {}

#Human
file_to_dict('tT007.txt')
file_to_dict('tT009.txt')
file_to_dict('tT023.txt')
file_to_dict('tT036.txt')
file_to_dict('tT037.txt')
file_to_dict('tT038.txt')
file_to_dict('tT044.txt')

#Mouse
file_to_dict('tT000.txt')
file_to_dict('tT008.txt')
file_to_dict('tT011.txt')
file_to_dict('tT014.txt')
file_to_dict('tT020.txt')
file_to_dict('tT043a.txt')
file_to_dict('tT043b.txt')
file_to_dict('tT057.txt')
#file_to_dict('tT081.txt') excluded as a repeat
file_to_dict('tT082.txt')
file_to_dict('tT097.txt')
file_to_dict('tT112.txt')
file_to_dict('tT113.txt')
file_to_dict('tT133.txt')

#Other:
file_to_dict('tT058.txt')
file_to_dict('tT088.txt')
file_to_dict('tT092.txt')
file_to_dict('tT108a.txt')
file_to_dict('tT108b.txt')
file_to_dict('tT132.txt')
```

RESULTS Convert count dictionary to ordered dataframe

```
total_df = pd.DataFrame.from_dict(total_dict, orient='index')
ordered_df = total_df.sort_values(by=0, ascending=False)
ordered_df
```

```
##            0
## UQCR10     8
```

```
## SIAH2      8
## CA4        8
## SPARC      8
## BRWD1      7
## ...       ..
## PRNP      -8
## HLA-A     -9
## CP        -9
## TMEM176A -11
## EFEMP1   -11
##
## [15495 rows x 1 columns]
```

Record dataframes for separate young and old counts too

```
young_df = pd.DataFrame.from_dict(young_dict, orient='index')
young_counts = young_df.sort_values(by=0, ascending=False)
young_counts
```

```
##           0
## NREP      9
## SPARC     9
## EIF4EBP1  8
## GOT2      8
## CA4       8
## ...      ..
## SGTA      1
## SFTPD     1
## SFTPC     1
## SFTPB     1
## ZMPSTE24  1
##
## [11499 rows x 1 columns]
```

```
old_df = pd.DataFrame.from_dict(old_dict, orient='index')
old_counts = old_df.sort_values(by=0, ascending=False)
old_counts
```

```
##            0
## EFEMP1    11
## TMEM176A  11
## HLA-A     10
## CP        10
## GCNT2     10
## ...       ..
## PPP1R15A   1
## PPM1N      1
## PP2D1      1
## POLR3H     1
## ZFR        1
##
## [11873 rows x 1 columns]
```

Export all counts to CSV files

```
ordered_df.to_csv("Total_Counts.csv")
young_counts.to_csv("Young_Counts.csv")
old_counts.to_csv("Old_Counts.csv")
```

Note: Can skip down to section 5 from here if wanting to fast forward to histogram.

4B. Show dataset characteristics for each high-scoring gene

Show genes with high scores *Note: adjust HERE to change the score limit for section 4B*

```
subset_counts = ordered_df[(ordered_df[0]>6) | (ordered_df[0]<-6)]
subset_counts
```

```
##            0
## UQCR10     8
## SIAH2      8
## CA4        8
## SPARC      8
## BRWD1      7
## BRD3       7
## SMYD1      7
## EIF4EBP1   7
## CDC20      7
## AR         7
## VLDLR      7
## DIRC2      7
## KIFAP3     7
## EIF2D      7
## NREP       7
## RTN4IP1    7
## CASP1     -7
## CTSS      -7
## RSRC1     -7
## FST       -7
## SLC44A1   -7
## CTNNA1    -7
## C1QC      -7
## PJA2      -7
## LYST      -7
## FYB       -7
## SKAP2     -7
## RNF145    -7
## EHD4      -7
## CFLAR     -7
## TMEM123   -7
## SERPINB6  -7
## MGST1     -8
## NPC2      -8
## SFRP1     -8
## APOE      -8
```

```
## ARAP2     -8
## PTPRC     -8
## ANXA4     -8
## GCNT2     -8
## PRNP      -8
## HLA-A     -9
## CP        -9
## TMEM176A -11
## EFEMP1   -11
```

Create gene index

```
gene_index = subset_counts.index
gene_index = list(gene_index)
```

Create indices for dataset characteristics

```
file = open("221_Datasets.csv")
header = file.readline()
indices = [] #0
animals = [] #2
tissues = [] #6
for line in file:
  row = line.strip().split(',')
  index = row[0]
  animal = row[2]
  tissue = row[6]
  indices.append(index)
  animals.append(animal)
  tissues.append(tissue)
file.close()
```

Define function to list all datasets (and their characteristics) contributing to each high-scoring gene:

```
def tissues_for_hits(file_name):
  file_index = file_name.strip("tT").strip(".tx")

  if file_index == '043a':
    file_index = 43
  elif file_index == '043b':
    file_index = 44
  elif file_index == '108a':
    file_index = 109
  elif file_index == '108b':
    file_index = 110
  else:
    file_index = int(file_index)
    if file_index > 108:
      file_index = file_index + 2
    elif file_index > 43 and file_index < 108:
      file_index = file_index + 1

  file = open(file_name)
```

```
  header = file.readline()
  geneIDs = []
  for line in file:
    row = line.strip().replace('"','').split('\t')
    geneID = row[6]
    geneIDs.append(geneID)
  for gene in gene_index:
    if gene in geneIDs and gene not in gene_complex_dictionary:
      gene_complex_dictionary[gene] = {'index':[indices[file_index]],'tissue':[tissues[file_index]],'an

    elif gene in geneIDs and gene in gene_complex_dictionary:
      gene_complex_dictionary[gene]['index'] += [indices[file_index]]
      gene_complex_dictionary[gene]['tissue'] += [tissues[file_index]]
      gene_complex_dictionary[gene]['animal'] += [animals[file_index]]
  file.close()
```

DATASET CONTROL ZONE 5 OF 6 Execute function and print results:

```
gene_complex_dictionary = {}

#Human
tissues_for_hits('tT007.txt')
tissues_for_hits('tT009.txt')
tissues_for_hits('tT023.txt')
tissues_for_hits('tT036.txt')
tissues_for_hits('tT037.txt')
tissues_for_hits('tT038.txt')
tissues_for_hits('tT044.txt')

#Mouse
tissues_for_hits('tT000.txt')
tissues_for_hits('tT008.txt')
tissues_for_hits('tT011.txt')
tissues_for_hits('tT014.txt')
tissues_for_hits('tT020.txt')
tissues_for_hits('tT043a.txt')
tissues_for_hits('tT043b.txt')
tissues_for_hits('tT057.txt')
#tissues_for_hits('tT081.txt') excluded as a repeat
tissues_for_hits('tT082.txt')
tissues_for_hits('tT097.txt')
tissues_for_hits('tT112.txt')
tissues_for_hits('tT113.txt')
tissues_for_hits('tT133.txt')

#Other:
tissues_for_hits('tT058.txt')
tissues_for_hits('tT088.txt')
tissues_for_hits('tT092.txt')
tissues_for_hits('tT108a.txt')
tissues_for_hits('tT108b.txt')
tissues_for_hits('tT132.txt')
```

Visualizing gene by tissue heatmap preparation

ALERT If brand new tissue type is added, make sure to add to dictionary below.

```
simpleGCD = {}
for key in gene_complex_dictionary:
  simpleGCD[key] = gene_complex_dictionary[key]['tissue']

simpleGCD2 = {}
for key in simpleGCD:
  new_dict = {
  'Trachea':0,'Reproduction':0,'Muscle':0,'Liver':0,'Immune':0,'Heart':0,'Fat':0,'Cochlea':0,'Brain':0
  }
  for item in simpleGCD[key]:
    new_dict[item] += 1
  simpleGCD2[key] = new_dict

simpleGCD_df = pd.DataFrame.from_dict(simpleGCD2,orient='index')
simpleGCD_df
```

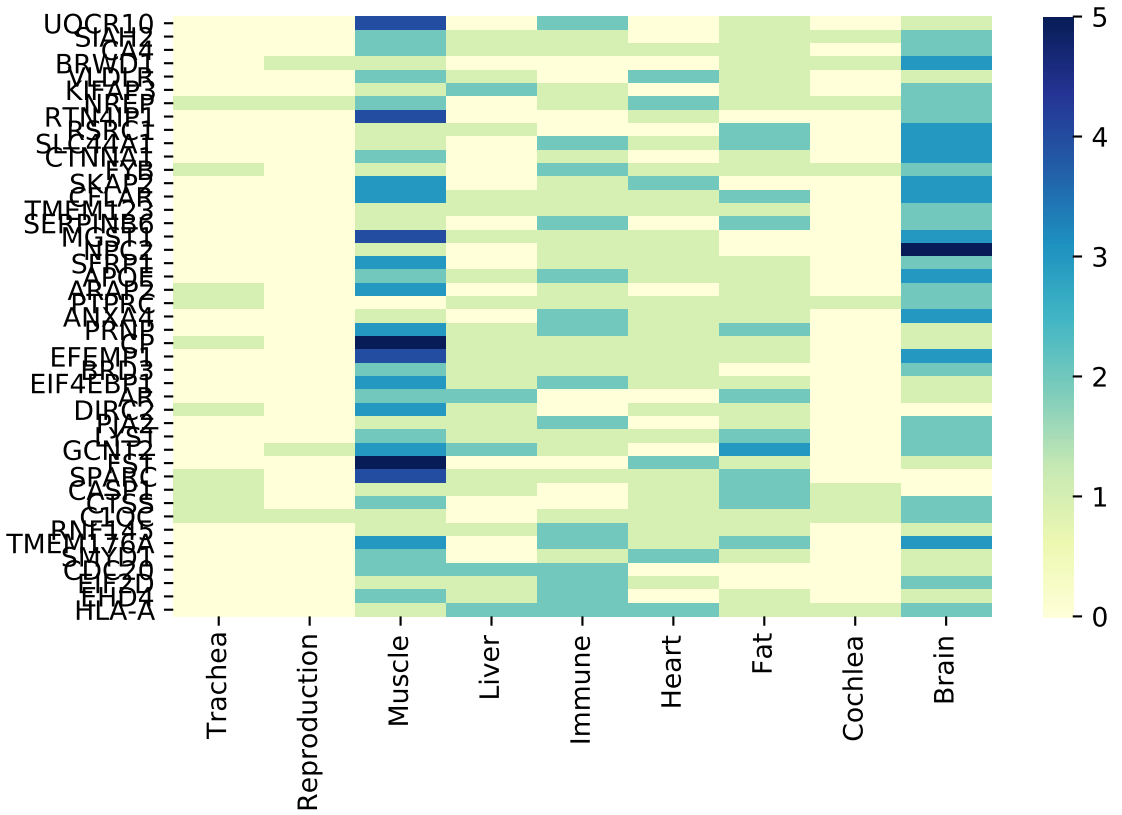
```
##           Trachea  Reproduction  Muscle  Liver  ...  Heart  Fat  Cochlea  Brain
## UQCR10          0             0       4      0  ...      0    1        0      1
## SIAH2           0             0       2      1  ...      0    1        1      2
## CA4             0             0       2      1  ...      1    1        0      2
## BRWD1           0             1       1      0  ...      0    1        1      3
## VLDLR           0             0       2      1  ...      2    1        0      1
## KIFAP3          0             0       1      2  ...      0    1        0      2
## NREP            1             1       2      0  ...      2    1        1      2
## RTN4IP1         0             0       4      0  ...      1    0        0      2
## RSRC1           0             0       1      1  ...      0    2        0      3
## SLC44A1         0             0       1      0  ...      1    2        0      3
## CTNNA1          0             0       2      0  ...      0    1        0      3
## FYB             1             0       1      0  ...      1    1        1      2
## SKAP2           0             0       3      0  ...      2    0        0      3
## CFLAR           0             0       3      1  ...      1    2        0      3
## TMEM123         0             0       1      1  ...      1    1        0      2
## SERPINB6        0             0       1      0  ...      0    2        0      2
## MGST1           0             0       4      1  ...      1    0        0      3
## NPC2            0             0       1      0  ...      1    0        0      5
## SFRP1           0             0       3      0  ...      1    1        0      2
## APOE            0             0       2      1  ...      1    1        0      3
## ARAP2           1             0       3      0  ...      0    1        0      2
## PTPRC           1             0       0      1  ...      1    1        1      2
## ANXA4           0             0       1      0  ...      1    1        0      3
## PRNP            0             0       3      1  ...      1    2        0      1
## CP              1             0       5      1  ...      1    1        0      1
## EFEMP1          0             0       4      1  ...      1    1        0      3
## BRD3            0             0       2      1  ...      1    0        0      2
## EIF4EBP1        0             0       3      1  ...      1    1        0      1
## AR              0             0       2      2  ...      0    2        0      1
## DIRC2           1             0       3      1  ...      1    1        0      0
## PJA2            0             0       1      1  ...      0    1        0      2
## LYST            0             0       2      1  ...      1    2        0      2
## GCNT2           0             1       3      2  ...      0    3        0      2
## FST             0             0       5      0  ...      2    1        0      1
## SPARC           1             0       4      1  ...      1    2        0      0
```

```
## CASP1           1           0       1       1  ...      1     2        1       0
## CTSS            1           0       2       0  ...      1     2        1       2
## C1QC            1           1       1       0  ...      1     1        1       2
## RNF145          0           0       1       1  ...      1     1        0       1
## TMEM176A        0           0       3       0  ...      1     2        0       3
## SMYD1           0           0       2       0  ...      2     1        0       1
## CDC20           0           0       2       2  ...      0     0        0       1
## EIF2D           0           0       1       1  ...      1     0        0       2
## EHD4            0           0       2       1  ...      0     1        0       1
## HLA-A           0           0       1       2  ...      2     1        1       2
##
## [45 rows x 9 columns]
```

```
pd.DataFrame.to_csv(simpleGCD_df,"simpleGCD.csv")
```

Visualizing gene by tissue heatmap

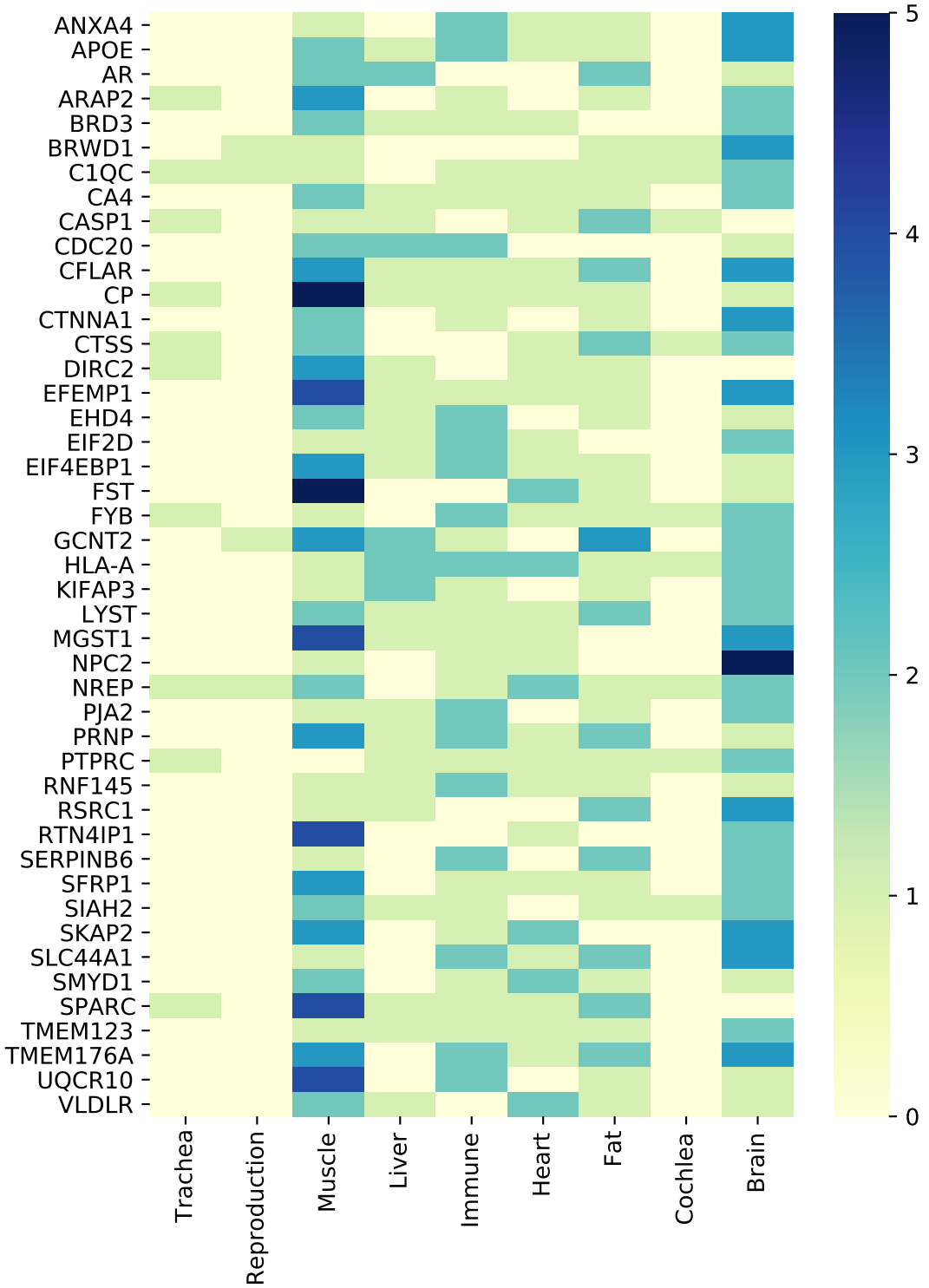
```
ax = sns.heatmap(simpleGCD_df, cmap="YlGnBu", xticklabels=True, yticklabels=True)
ax.figure.tight_layout()
ax
```



Visualizing gene by tissue heatmap - listed alphabetically and high-res?

```
simpleGCD_df_alpha = pd.DataFrame.sort_index(simpleGCD_df)
plt.figure(figsize = (6,8),dpi=600)
ax = sns.heatmap(simpleGCD_df_alpha, cmap="YlGnBu", xticklabels=True, yticklabels=True)
```

```
ax.figure.tight_layout()
plt.savefig('heatmaptest.svg')
ax
```



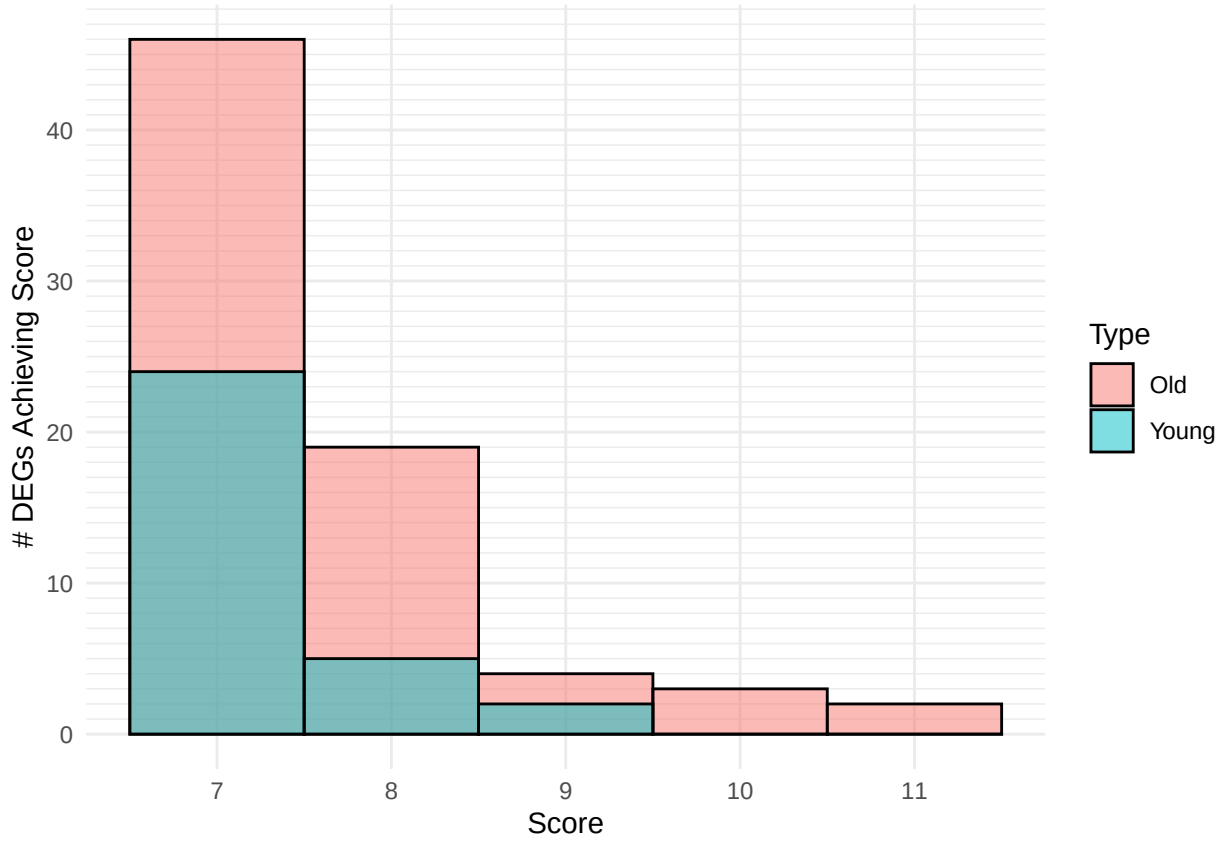
5. Score Analysis (R)

Import counts data and arrange in various dataframes and lists

```
TotalCounts_byGene <- read.csv("Total_Counts.csv")
colnames(TotalCounts_byGene)<-c("Gene","Total_Count")
TotalCounts_numbers <- TotalCounts_byGene[,2]

#adjust HERE to change the score limit moving forward
score_limit = 6

TC2 <- TotalCounts_byGene[which(TotalCounts_byGene$Total_Count > score_limit | TotalCounts_byGene$Total

YoungCounts_byGene <- read.csv("Young_Counts.csv")
YoungCounts_numbers <- YoungCounts_byGene[,2]
colnames(YoungCounts_byGene)<-c("Gene","Young_Count")

#adjust HERE to change histogram left limit (1 of 2):
YC2 <- YoungCounts_byGene[which(YoungCounts_byGene$Young_Count > score_limit),]
YC2L <- YC2[,2]

OldCounts_byGene <- read.csv("Old_Counts.csv")
OldCounts_numbers <- OldCounts_byGene[,2]
colnames(OldCounts_byGene)<-c("Gene","Old_Count")

#adjust HERE to change histogram left limit (2 of 2):
OC2 <- OldCounts_byGene[which(OldCounts_byGene$Old_Count > score_limit),]
OC2L <- OC2[,2]

YC2$Type <- "Young"
OC2$Type <- "Old"
colnames(YC2) <- c("Gene", "Count", "Type")
colnames(OC2) <- c("Gene", "Count", "Type")
HistoCounts <- rbind(YC2, OC2)
```

Histogram of counts

```
ggplot(HistoCounts, aes(Count, fill = Type)) +
  geom_histogram(alpha = 0.5, color = "black", binwidth = 1, position = "identity") +
  theme_minimal() +
  scale_x_continuous(name = "Score", breaks = 3:15, minor_breaks = NULL) +
  scale_y_continuous(name = "# DEGs Achieving Score", minor_breaks = 1:100)
```




```
ggsave(file="histoscorestest.svg")
```

```
## Saving 6.5 x 4.5 in image
```

```
ggplot
```

```
## function (data = NULL, mapping = aes(), ..., environment = parent.frame())
## {
##     UseMethod("ggplot")
## }
## <bytecode: 0x00000000218a1c38>
## <environment: namespace:ggplot2>
```

6. Dataset Characteristics (R)

NOTE Before running these next sessions, it was necessary to manually fill in the # DEGs column in the datasets csv file in excel. Here I am adding two new chunks to update the #DEGs column internally to a dataframe called DS.

(Exclude datasets from counts by commenting out here.) First create the DS dataframe by reading in the 221 datasets file, and modify the index numbers to clear up issues where the same dataset was used for two different analyses because it contained data from two different tissue types.

```
DS <- read.csv("221_Datasets.csv")

DS[44,1]<-"43a"
DS[45,1]<-"43b"
DS[110,1]<-"108a"
DS[111,1]<-"108b"
```

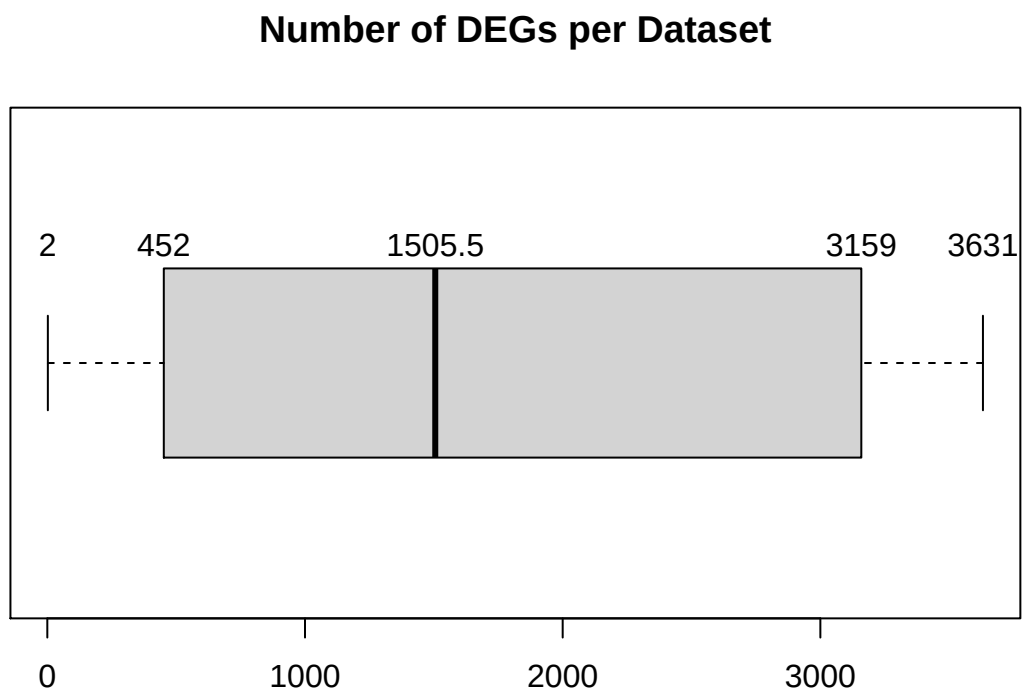
Second, run a function to replace the DEG counts in the DS dataframe with counts corresponding to the length of each DEG dataset (all the tT objects). **DATASET CONTROL ZONE 6 OF 6**

```
DSS_Function <- function(dataset_df,index_char){
  new_DEG_count<-length(dataset_df[,1])
  DS[which(DS$Index==index_char),9]<-new_DEG_count
  DSS_Output <- DS
  return(DSS_Output)
}

#Human:
DS<-DSS_Function(tT007,"7")
DS<-DSS_Function(tT009,"9")
DS<-DSS_Function(tT023,"23")
DS<-DSS_Function(tT036,"36")
DS<-DSS_Function(tT037,"37")
DS<-DSS_Function(tT038,"38")
DS<-DSS_Function(tT044,"44")
#Mouse:
DS<-DSS_Function(tT000,"0")
DS<-DSS_Function(tT008,"8")
DS<-DSS_Function(tT011,"11")
DS<-DSS_Function(tT014,"14")
DS<-DSS_Function(tT020,"20")
DS<-DSS_Function(tT043a,"43a")
DS<-DSS_Function(tT043b,"43b")
DS<-DSS_Function(tT057,"57")
#DS<-DSS_Function(tT081,"81") excluded as repeat
DS<-DSS_Function(tT082,"82")
DS<-DSS_Function(tT097,"97")
DS<-DSS_Function(tT112,"112")
DS<-DSS_Function(tT113,"113")
DS<-DSS_Function(tT133,"133")
#Other:
DS<-DSS_Function(tT058,"58")
DS<-DSS_Function(tT088,"88")
DS<-DSS_Function(tT092,"92")
DS<-DSS_Function(tT108a,"108a")
DS<-DSS_Function(tT108b,"108b")
DS<-DSS_Function(tT132,"132")
```

Now the same analyses should be able to proceed as before, but with automatically updated DEG counts:

Number of DEGs per dataset (R)

```
boxplot(DS$DEGs, horizontal=TRUE, main="Number of DEGs per Dataset")
text(x=fivenum(DS$DEGs), labels =fivenum(DS$DEGs), y=1.25)
```




```
#Note the middle value is median (50th percentile), not mean.
#Lower hinge is 25th percentile, upper is 75th percentile
```

Number of datasets evaluated (R)

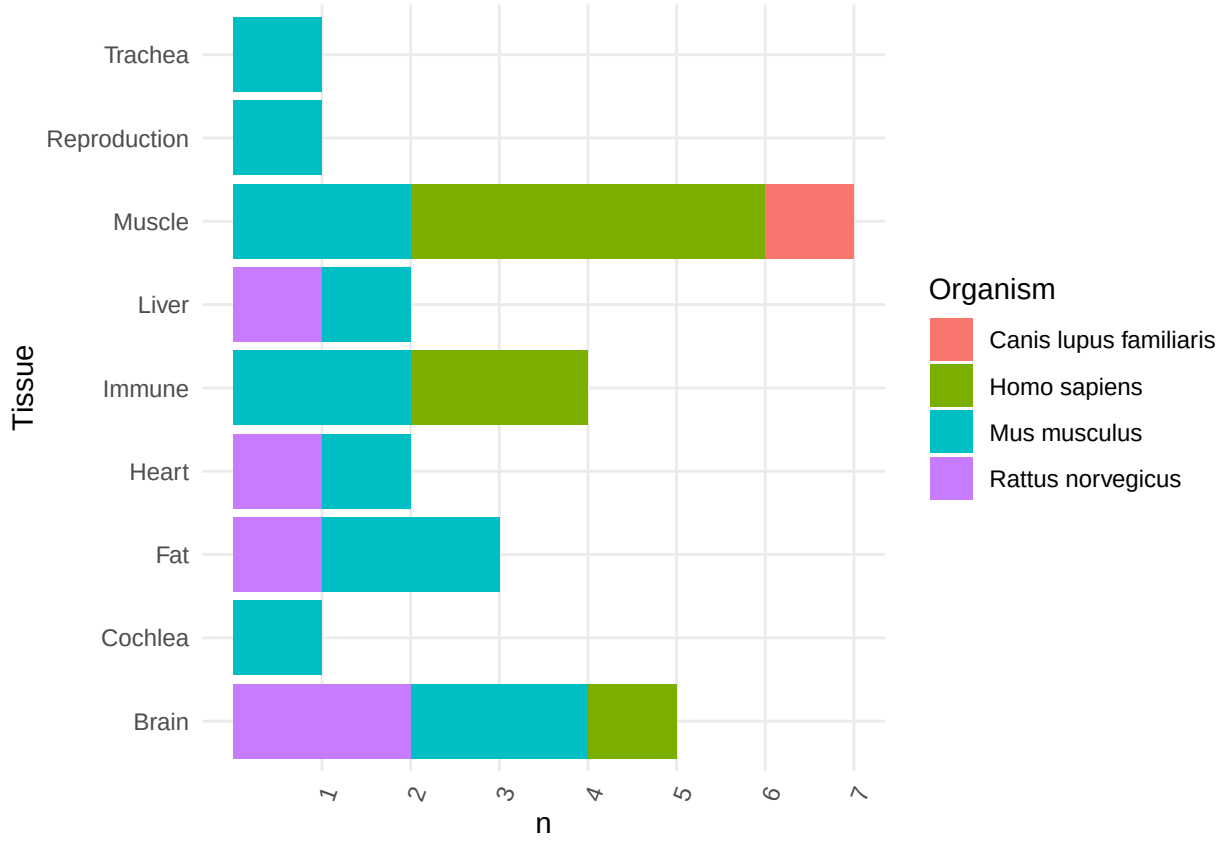
```
DS_Progress = DS[which(DS$DEGs >0),]
length(DS_Progress$DEGs)
```

```
## [1] 26
```

```
write.csv(DS_Progress,"DatasetsToDate.csv")
```

Tissue counts (R)

```
agg <- count(DS_Progress,Tissue,Organism)
ggplot(agg) +
  geom_col(aes(x = Tissue, y = n, fill = Organism)) +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 70, hjust=1))+
  scale_y_continuous(breaks = 1:10, minor_breaks = NULL)+
  coord_flip()
```




```
ggsave(file="datasetstats.svg")
```

```
## Saving 6.5 x 4.5 in image
```

DEG counts (R)

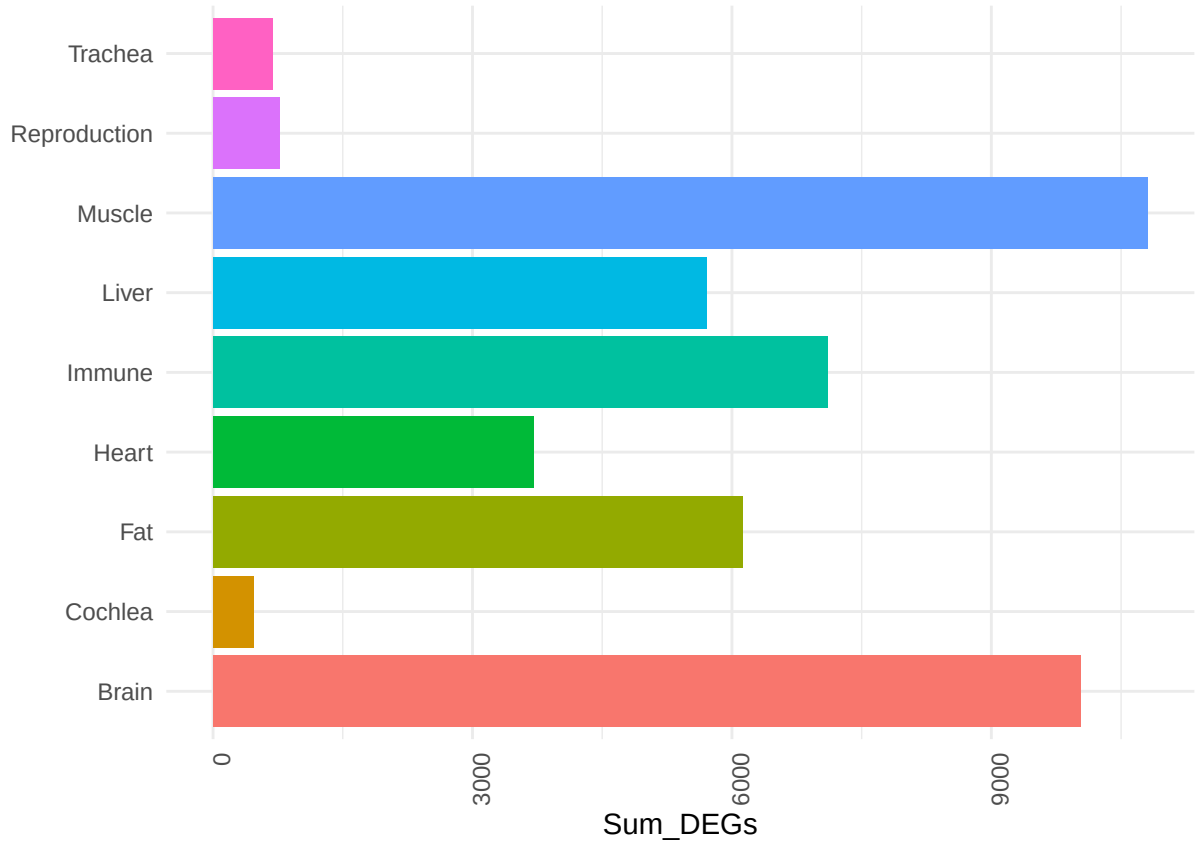
```
DS_Progress %>%
  group_by(Tissue) %>%
  summarise(sum_DEGs = sum(DEGs), mean_DEGs = mean(DEGs))
```

```
## # A tibble: 9 x 3
##   Tissue       sum_DEGs mean_DEGs
##   <chr>           <int>     <dbl>
## 1 Brain           10028     2006.
## 2 Cochlea           466      466
## 3 Fat              6125     2042.
## 4 Heart            3702     1851
## 5 Immune           7107     1777.
## 6 Liver            5712     2856
## 7 Muscle          10807     1544.
## 8 Reproduction      772      772
## 9 Trachea           688      688
```

Graph of DEG counts (R)

```
DS_Progress %>%
  group_by(Tissue) %>%
  summarise(Sum_DEGs = sum(DEGs), mean_DEGs = mean(DEGs)) %>%
  ggplot(aes(x=Tissue, y = Sum_DEGs, fill = Tissue)) +
  geom_bar(stat="identity")+
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 90, hjust=1))+
  xlab("")+
  guides(fill=FALSE)+
  coord_flip()
```

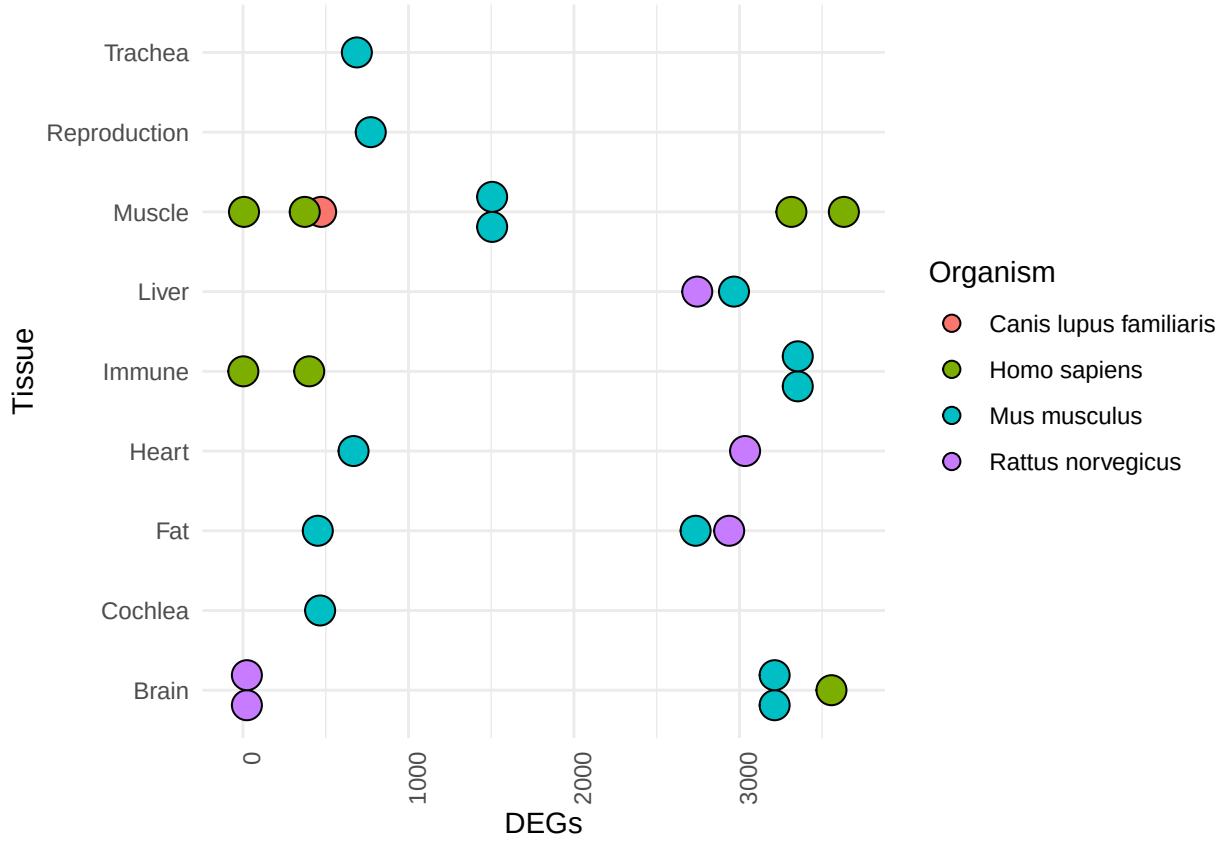
```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



Graph of DEG counts (R)

```
ggplot(DS_Progress, aes(x=Tissue,y=DEGs, fill=Organism))+
  geom_dotplot(binaxis = 'y',stackdir = 'center',dotsize=1.5)+
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 90, hjust=1))+
  coord_flip()
```

```
## Bin width defaults to 1/30 of the range of the data. Pick better value with `binwidth`.
```



7. Worm Orthologs

Note: score_limit variable carried over from histogram section. Can replace it with a number here if needed (on line 3 below)

```
OrthoList <- read.csv("ortholist_master.csv")
OrthoDEGs <- merge(TotalCounts_byGene,OrthoList,by.x = "Gene", by.y = "HGNC.Symbol", all.x = TRUE)
subset_by_counts <- OrthoDEGs[which(OrthoDEGs$Total_Count > score_limit | OrthoDEGs$Total_Count < -score
DEG_Orthologs <- subset_by_counts[,c("Gene","Total_Count","Locus.ID","Common.Name","No..of.Programs","Al
colnames(DEG_Orthologs) <- c("Human.Gene","DE.Score","Worm.Gene","Name","Programs","RNAi")
DEG_Orthologs
```

```
##       Human.Gene DE.Score  Worm.Gene    Name Programs                  RNAi
## 1867       ANXA4       -8    ZC155.1   nex-1        1               III-2P15
## 1868       ANXA4       -8    T07C4.9   nex-2        3      III-5F15|III-8O24
## 1965        APOE       -8       <NA>    <NA>       NA                  <NA>
## 2008          AR        7       <NA>    <NA>       NA                  <NA>
## 2011       ARAP2       -8  F23H11.4                3               III-1G04
## 3086        BRD3        7    F59F5.7                1                 X-5K11
## 3087        BRD3        7  Y119C1B.8   bet-1        5                 I-9E23
## 3088        BRD3        7    F57C7.1   bet-2        2                 X-8B17
## 3121       BRWD1        7       <NA>    <NA>       NA                  <NA>
## 3338        C1QC       -7       <NA>    <NA>       NA                  <NA>
## 3517         CA4        8     R173.1   cah-5        1                 X-8I08
```

```
## 3518        CA4        8    K05G3.3   cah-3        1                  X-7M16
## 3713      CASP1       -7  Y73B6BL.7   csp-2        1
## 3714      CASP1       -7    C48D1.2   ced-3        1                IV-9F04
## 3715      CASP1       -7   Y47H9C.6   csp-3        1                 I-5P02
## 3716      CASP1       -7  Y48E1B.13   csp-1        2               II-10A10
## 4089      CDC20        7    ZK177.6   fzy-1        1        II-4O16|II-4O18
## 4090      CDC20        7   ZK1307.6   fzr-1        1               II-10A17
## 4378      CFLAR       -7       <NA>    <NA>       NA                   <NA>
## 5195         CP       -9       <NA>    <NA>       NA                   <NA>
## 5622     CTNNA1       -7    R13H4.4   hmp-1        5
## 5701       CTSS       -7    T03E6.7   cpl-1        2                V-11I07
## 5702       CTSS       -7   R09F10.1                1                 X-4A20
## 5703       CTSS       -7    F41E6.6 tag-196        1                 V-6K19
## 6744      DIRC2        7    C42C1.8                1                IV-6D23
## 6745      DIRC2        7    C05G5.1                1                 X-6H18
## 6746      DIRC2        7    B0416.5                1                 X-4N07
## 7265     EFEMP1      -11   F56H11.1   fbl-1        1        IV-4N18|IV-4P14
## 7314        EHD4       -7    W06H8.1   rme-1        4               V-14F01
## 7350       EIF2D        7    C25H3.4                6       II-10E20|II-4H08
## 7383    EIF4EBP1        7       <NA>    <NA>       NA                   <NA>
## 8593         FST       -7       <NA>    <NA>       NA                   <NA>
## 8653         FYB       -7       <NA>    <NA>       NA                   <NA>
## 8967       GCNT2       -8    T15D6.2  gly-16        4                 I-6A08
## 8968       GCNT2       -8   F22D6.11  gly-18        4                 I-3B11
## 8969       GCNT2       -8   F22D6.12  gly-19        4                 I-3B13
## 8970       GCNT2       -8    F44F4.6   gly-1        4                II-7F21
## 8971       GCNT2       -8   C54C8.11  gly-15        4                 I-6E22
## 8972       GCNT2       -8    T14B4.9                1                II-5O03
## 8973       GCNT2       -8    T15D6.3  gly-17        4                 I-6A10
## 8974       GCNT2       -8   T09E11.9                1           I-6O23|I-9F14
## 8975       GCNT2       -8    T27F6.1                1                 I-6E24
## 8976       GCNT2       -8    T28F3.9                1                IV-8C02
## 8977       GCNT2       -8  Y51H4A.25                1 IV-8A09|IV-8A13|IV-8A15
## 8978       GCNT2       -8   ZK1225.2                1                 I-6N23
## 8979       GCNT2       -8   H41C03.3                1        II-10G20|II-4P06
## 8980       GCNT2       -8    R07C3.3                1                II-1F15
## 8981       GCNT2       -8    R07B7.6                1                 V-8I05
## 8982       GCNT2       -8    F26D2.3                1                V-10P22
## 8983       GCNT2       -8   F30A10.4                1                 I-4N21
## 8984       GCNT2       -8    F35H8.2                1
## 8985       GCNT2       -8   T09E11.6                1                 I-9C07
## 10813      HLA-A       -9       <NA>    <NA>       NA                   <NA>
## 12394     KIFAP3        7    F08F8.3   kap-1        6               III-3N14
## 13751       LYST       -7   T01H10.8  lyst-1        1                 X-5F04
## 14440      MGST1       -8       <NA>    <NA>       NA                   <NA>
## 15838       NPC2       -8     R148.6   heh-1        1               III-1P05
## 15947       NREP        7       <NA>    <NA>       NA                   <NA>
## 17402       PJA2       -7 Y54E10BR.3                1                 I-7N18
## 18146       PRNP       -8       <NA>    <NA>       NA                   <NA>
## 18585      PTPRC       -8    F56D1.4   clr-1        1                II-4M08
## 19399     RNF145       -7  Y119C1B.5                5                 I-9E22
## 19676      RSRC1       -7    C04G2.8  spch-1        1                IV-5E06
## 19677      RSRC1       -7   C10G11.9  spch-2        1                 I-3C03
## 19678      RSRC1       -7  Y57G11C.9                0                IV-9C17
```

```
## 19679      RSRC1       -7    T27A3.4  spch-3        1                 I-2J16
## 19692    RTN4IP1        7    F56H1.6   rad-8        6                 I-9F05
## 20202   SERPINB6       -7    F20D6.4   srp-7        3                V-14F18
## 20203   SERPINB6       -7    C05E4.1   srp-2        2                 V-1M06
## 20204   SERPINB6       -7   Y32G9A.4   srp-3        1                V-14F17
## 20205   SERPINB6       -7    F20D6.3   srp-8        1                V-14F19
## 20206   SERPINB6       -7    C05E4.3   srp-1        2                 V-1M10
## 20207   SERPINB6       -7   C03G6.19   srp-6        3                 V-5G14
## 20318      SFRP1       -8 Y73B6BL.21  sfrp-1        4
## 20551      SIAH2        8 Y37E11AR.2  siah-1        4                IV-8K18
## 20612      SKAP2       -7       <NA>    <NA>       NA                   <NA>
## 21261    SLC44A1       -7       <NA>    <NA>       NA                   <NA>
## 21636      SMYD1        7    F33H2.7  set-10        1                 I-7M22
## 21637      SMYD1        7    T22A3.4  set-18        3                 I-5G16
## 21638      SMYD1        7      ZC8.3  set-30        2          X-2L08|X-2L10
## 21639      SMYD1        7    R06F6.4  set-14        3                II-7D01
## 21851      SPARC        8   C44B12.2   ost-1        5                IV-9H03
## 23082    TMEM123       -7       <NA>    <NA>       NA                   <NA>
## 23136   TMEM176A      -11       <NA>    <NA>       NA                   <NA>
## 24949     UQCR10        8       <NA>    <NA>       NA                   <NA>
## 25145      VLDLR        7    T13C2.6                 6                II-5A06
```

```
write.csv(DEG_Orthologs,"DEG_Orthologs.csv")
```

Percent of mammalian gene hits for which worm orthologs are available:

```
no.hum<-length(TC2$Gene)
cat("Number of human genes queried:",no.hum)
```

```
## Number of human genes queried: 45
```

```
no.notworm <- sum(is.na(DEG_Orthologs$Worm.Gene))
cat("\nNumber without orthologs:",no.notworm)
```

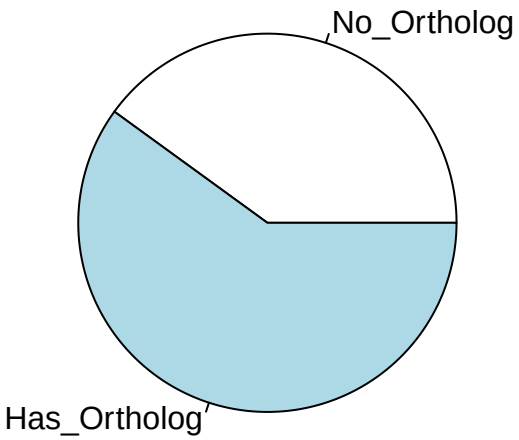
```
##
## Number without orthologs: 18
```

```
no.worm <- no.hum - no.notworm
percent.worm <- no.worm/no.hum
cat("\nFraction with orthologs:",percent.worm)
```

```
##
## Fraction with orthologs: 0.6
```

Pie Chart

```
pie_slices <- c(no.notworm,no.worm)
pie_labels <- c("No_Ortholog","Has_Ortholog")
pie(pie_slices,labels=pie_labels)
```



Print human gene names for online searches:

```
paste(DEG_Orthologs$Human.Gene, collapse = ", ")
```

```
## [1] "ANXA4, ANXA4, APOE, AR, ARAP2, BRD3, BRD3, BRD3, BRWD1, C1QC, CA4, CA4, CASP1, CASP1, CASP1, CA
```

Print c. elegans gene names for online searches:

```
paste(DEG_Orthologs$Name, collapse = ", ")
```

```
## [1] "nex-1, nex-2, NA, NA, , , bet-1, bet-2, NA, NA, cah-5, cah-3, csp-2, ced-3, csp-3, csp-1, fzy-1
```