go_term_analysis

Scott McIsaac

12/30/2018

```
## R Markdown

#functions
######################################################
######################################################

AddSystematic <- function(data){
library('org.Sc.sgd.db')
xx= as.list(org.Sc.sgdCOMMON2ORF)
yy = sapply(xx, function(x){x[1]})
common=names(xx)
orf = as.vector(yy)
names(common)=orf

#add systematic names to dataset
data$systematic <- as.vector(yy[as.vector(data$gene)])
sysmissing_positions <- is.na(data$systematic)
data$systematic[sysmissing_positions] = as.vector(data$gene[sysmissing_positions])
rmlist <- data$systematic[duplicated(data$systematic)] #get dups
data <-data[!(data$systematic %in% rmlist),] #remove dups
return(data)
}

#function for comparing data before and after adding systematic name
RowTest <- function(d,d_ref,tests){ #d: barseq2, d_ref = barseq, test = 50

d_ref <- barseq
d <- barseq2

X = dim(d)[1]
Y = dim(d_ref)[2]
row_pick <- sample(X,tests,replace=F)

d_selected <- d[row_pick,] #from data that includes systematic names

counter = 0
for(i in 1:tests){
  V <- d_ref[d_ref$gene %in% d_selected[i,]$gene,] #data for gene i from reference dataset

  #get rid of NAs and replace with 9000
  V[is.na(V)] <- 9000
  d_selected[i,][is.na(d_selected[i,])] <- 9000

  #test for matching
  Score <- sum(d_selected[i,1:Y] == V[1:Y])
  if(Score == Y){
    counter = counter + 1
  }
}

if(counter == tests){
  print("PASSES")}


if(counter != tests){
  print("FAILS")}
}


#end of functions
######################################################
######################################################


library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------ tidyverse 1.2.1 --

## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0

## -- Conflicts --------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
######################################################
######################################################

#load GO annotation
GO <- read.delim(file="/Volumes/GoogleDrive/My Drive/genome_data/Yeast/Datasets/go_slim_mapping.tab",se
colnames(GO)[1] = "systematic"
colnames(GO)[2] = "gene"
colnames(GO)[5] = "goterm"

#load data
barseq <- read.delim(file="/Volumes/GoogleDrive/My Drive/Projects/BARSeq/GOtermAnalysis/allBarSeq_20181

#annotate samples
samples <- c(1,1,1,2,3,4,5) #label replicates here

#add systematic gene names to data
barseq2 <- AddSystematic(barseq)
```

```
## Loading required package: AnnotationDbi

## Loading required package: stats4

## Loading required package: BiocGenerics

## Loading required package: parallel

##
```

```
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: IRanges

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:tidyr':
##
##     expand

## The following object is masked from 'package:base':
##
##     expand.grid

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice
## The following object is masked from 'package:purrr':
##
##     reduce

##
## Attaching package: 'AnnotationDbi'

## The following object is masked from 'package:dplyr':
##
##     select

##
```

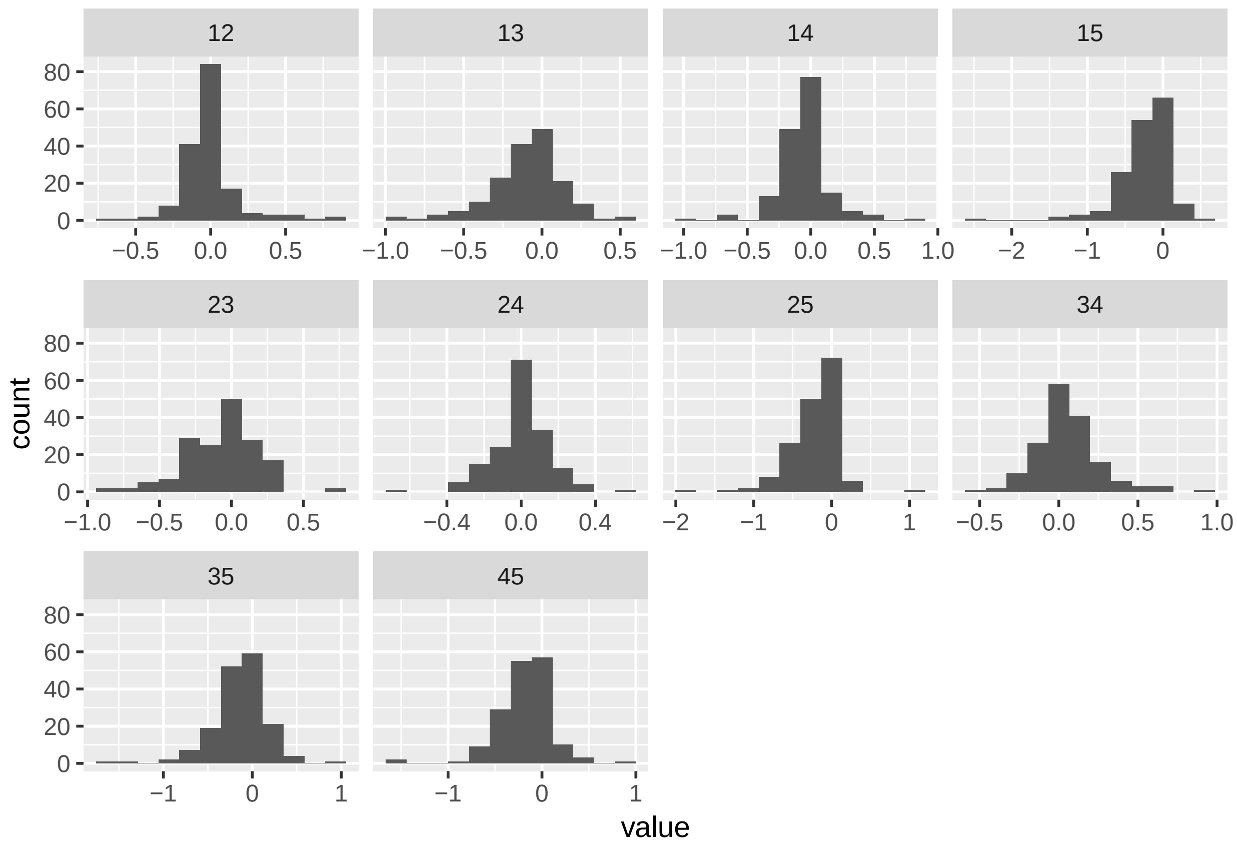
```
#Check that data matches after adding systematic names
RowTest(barseq,barseq2,150)
```

```
## [1] "PASSES"
```

```
#list of GO terms
GOtib <- as.tibble(GO)
GOlist <- as.vector(GOtib %>% group_by(goterm) %>% summarise())
GOlist <- as.vector(GOlist$goterm)


#Atlas:
#MyMeans: For each sample, the average score for each GO term
#avg_data: The sample average of each gene in a specific GO term
#MyMeans_Pairwise: take differences of genes for each GO term and then calculate average coherence of t


#dataframe for putting scores for each sample
MyMeans <-data.frame(GO = GOlist) #this is where final data will go
uniquesamples <- unique(samples)
for(i in 1:length(uniquesamples)){
  MyMeans$placeholder_name = ""
  names(MyMeans)[names(MyMeans) == "placeholder_name"] <- toString(uniquesamples[i])
}

################################################
################################################
#store_pairs: dataframe for putting pairwise scores
#note: if there are N samples, then there are N*(N-1) pairs to go through

store_pairs <- c()
MyMeans_Pairwise <- data.frame(GO = GOlist)

for(i in 1:(length(uniquesamples)-1)){
  temp <- paste0(uniquesamples[i],uniquesamples[i+1:(length(uniquesamples)-i)])
  store_pairs <- c(store_pairs,temp)
}


for(i in 1:length(store_pairs)){
  MyMeans_Pairwise$placeholder_name = ""
  names(MyMeans_Pairwise)[names(MyMeans_Pairwise) == "placeholder_name"] <- store_pairs[i]
}
```

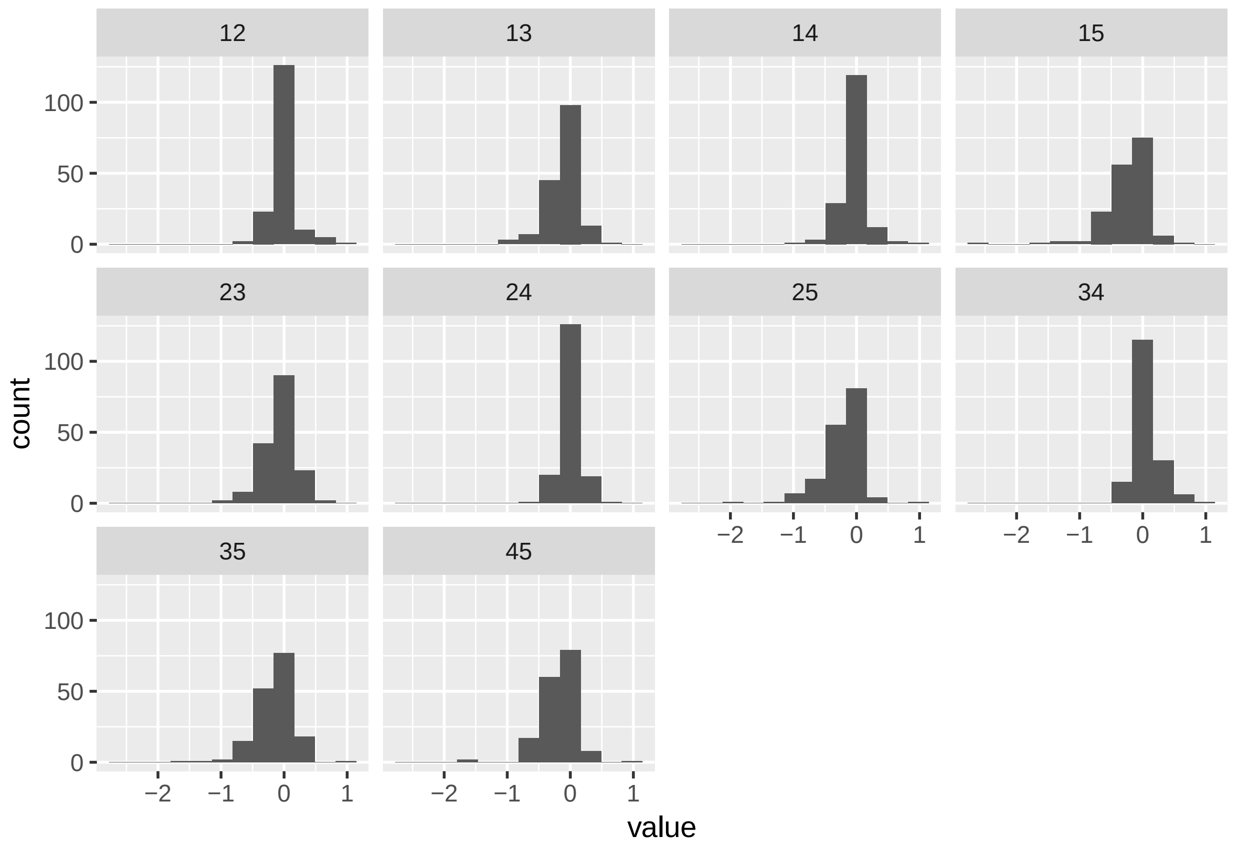
```
TotalPairs = length(store_pairs)

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

for(m in 1:length(GOlist)){

######get scoring for each GO term in GOlist
  GOgenes <- as.vector(GO[GO$goterm %in% GOlist[m],]$systematic) #genes in the m'th GOterm
  barseq2_GOgenes <- barseq2[barseq2$systematic %in% GOgenes,] #get data associated with GOterm
  numgenes <- dim(barseq2_GOgenes) #number of genes in dataset and in Goterm

  if(numgenes[1] == 0) next

#avg_data: dataframe for putting log2 means of data for each gene


  ################################################################
  ################do each sample 1 at a time
  ################################################################

  #create avg_data dataframe
  avg_data <- data.frame(gene = barseq2_GOgenes$gene)
  for(i in 1:length(uniquesamples)){
    avg_data$placeholder_name = ""
    names(avg_data)[names(avg_data) == "placeholder_name"] <- toString(uniquesamples[i])
  }

  #populate avg_data dataframe for this particular GO term
  for(i in 1:dim(avg_data)[1]){
    genedata <- as.vector(as.matrix(barseq2_GOgenes[,1:length(samples)+1][i,])) #data from i'th gene
    for(j in 1:length(uniquesamples)){ #compute mean for each gene from each sample
      grab <- samples == uniquesamples[j]
      avg_data[i,names(avg_data)==uniquesamples[j]] = mean(genedata[grab],na.rm=TRUE)
    }
  }

  #calculate average coherence for each sample --> store in MyMeans
  for(k in 1:length(uniquesamples)){
    MyMeans[m,toString(uniquesamples[k])] = mean(as.numeric(avg_data[,toString(uniquesamples[k])]),na.r
  }

  ################################################################
  ################do pairwise comparisons of datasets
  ################################################################

#create avg_data_pairwise dataframe
  avg_data_pairwise <- data.frame(gene = barseq2_GOgenes$gene)
  for(i in 1:length(store_pairs)){
    avg_data_pairwise$placeholder_name = ""
    names(avg_data_pairwise)[names(avg_data_pairwise) == "placeholder_name"] <- store_pairs[i]
  }
  counter = 1

#populate avg_data_pairwise dataframe
  for(i in 1:(length(uniquesamples)-1)){k
    for(j in (i+1):(length(uniquesamples))){
      avg_data_pairwise[,paste0(i,j)] = as.numeric(avg_data[,toString(i)]) - as.numeric(avg_data[,toStr
      counter = counter + 1
    }
    #
  }

##calculate average coherence for the DIFFERENCE of each sample --> store in MyMeans_Pairwise
#remember: TotalPairs = number of samples to compare
  for(k in 1:TotalPairs){
    MyMeans_Pairwise[m,colnames(avg_data_pairwise)[k+1]] = mean(as.numeric(avg_data_pairwise[,colnames(a
  }



}


write.table(MyMeans,file="/Volumes/GoogleDrive/My Drive/Projects/BARSeq/GOtermAnalysis/GOscores.txt",col
write.table(MyMeans_Pairwise,file="/Volumes/GoogleDrive/My Drive/Projects/BARSeq/GOtermAnalysis/GOscore




#Make histograms
#The changes with respect to
a <- MyMeans_Pairwise[,-1]
a <- gather(a)
a$value <- as.numeric(a$value)
ggplot(gather(a),aes(value)) +
  geom_histogram(bins=12) +
  facet_wrap(~key,scales="free_x")
```

```
## Warning: Removed 20 rows containing non-finite values (stat_bin).
```



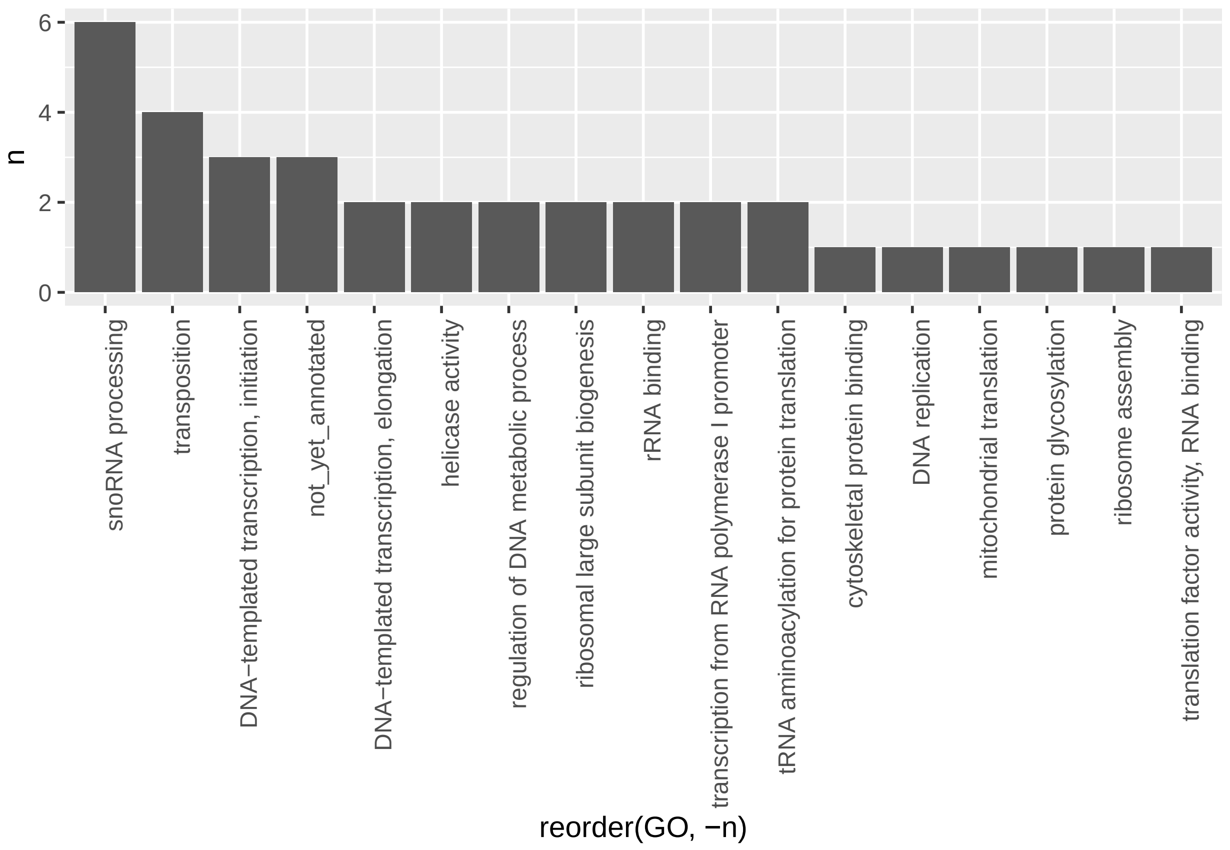
```
ggplot(gather(a),aes(value)) +
  geom_histogram(bins=12) +
  facet_wrap(~key)
```

```
## Warning: Removed 20 rows containing non-finite values (stat_bin).
```



```
##get genes from particular GO category
View(barseq2[barseq2$systematic %in% as.vector(GO[GO$goterm %in% "transposition",]$systematic),])
```

```
#get longformat MyMeans_Pairwise and convert to tidy format
pairwise_tidy <- gather(MyMeans_Pairwise,sample,value,-GO)
pairwise_tidy$value <- as.vector(sapply(pairwise_tidy$value, as.numeric))
pairwise_tidy <- as.tibble(pairwise_tidy)

pairwise_tidyfilter <- pairwise_tidy %>% filter(abs(value) > 0.8)

d <- pairwise_tidyfilter %>% count(GO) %>%
  arrange(desc(n))

p <- ggplot(d,aes(x=reorder(GO,-n),y=n)) + geom_col() + theme(axis.text.x = element_text(angle = 90, hj
p
```



```
#confirm MyMeans values by hand for a chosen GO-term

#GO-term = "organelle fission"
#of: get genes from "organelle fission"
GO_tib <- as.tibble(GO)
of <- GO_tib %>%
  dplyr::filter(goterm == "organelle fission") %>%
  dplyr::select(systematic) %>%
  .$systematic %>%
  as.vector()

#of_dat: get bar-seq data for gene genes in of
#samples <- c(1,1,1,2,3,4,5) #label replicates here

of_dat <- barseq2[barseq2$systematic %in% of,]

of_dat_means <- colMeans(of_dat[,2:8],na.rm=TRUE)

#COMPARE!

of_dat_means
```

```
##  YNB_NH4_GLU_20180713  YNB_NH4_GLU_20180731  YNB_NH4_GLU_20180807
##            -0.3468662            -0.4039856            -0.4207511
## YNB_UREA_GLU_20180724   SC_NH4_GLU_20180827  SC_UREA_GLU_20180911
##            -0.3412937            -0.2127937            -0.2374767
##   SC_NH4_GAL_20181023
##              0.1252596
```

```
MyMeans %>%
  filter(GO == "organelle fission")
```

```
##                  GO                  1          2                  3
## 1 organelle fission -0.398423565868263 -0.34129374 -0.212793657342657
##                    4                  5
## 1 -0.237476692307692 0.125259591194969
```

```
#Looks correct!
```