ComplexAnalysis

Barseq analysis

Bar-seq is used to find potentially long-lived strains.

In this analysis, we use protein complexes to look at the coherence of enrichment patterns across envrionmen-talconditions.

```
setwd("/Volumes/GoogleDrive/My Drive/Projects/BARSeq/")
#libraries
library('org.Sc.sgd.db')
```

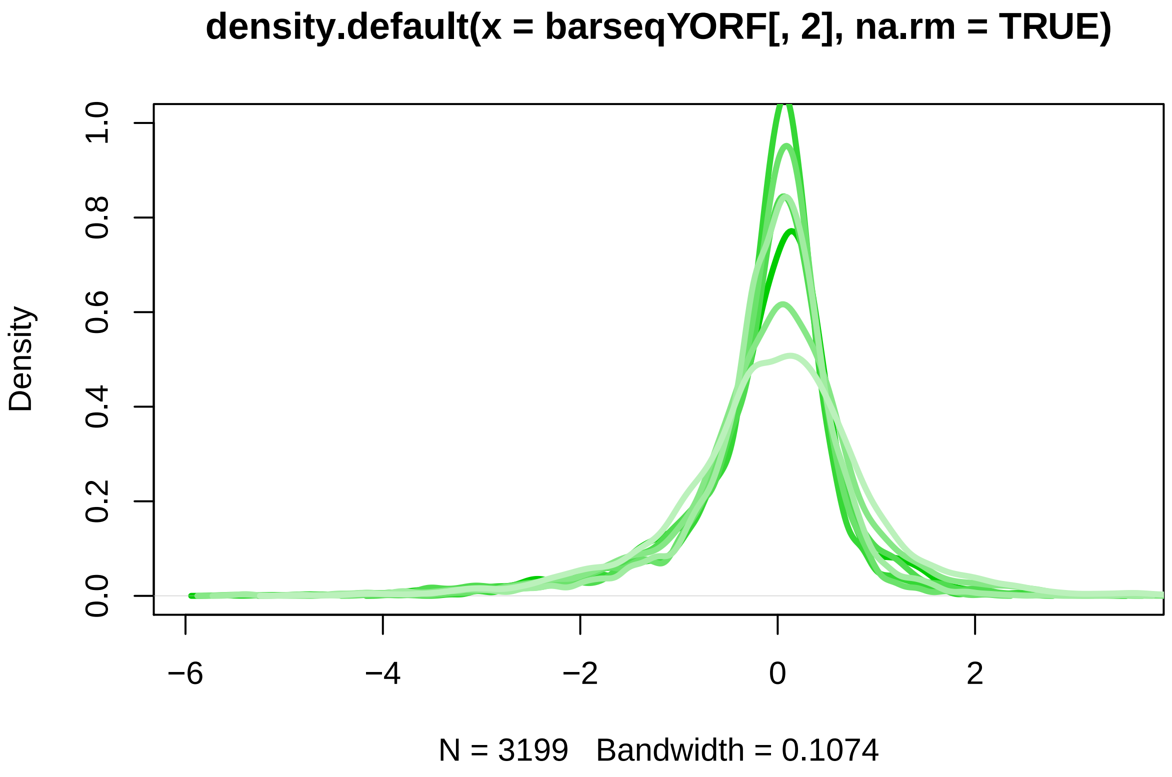
```
## Loading required package: AnnotationDbi

## 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: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")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:BiocGenerics':
##
##     dims
## Loading required package: IRanges

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

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

## Warning: replacing previous import 'BiocGenerics::dims' by 'Biobase::dims'
## when loading 'AnnotationDbi'

##
```

```
xx= as.list(org.Sc.sgdCOMMON2ORF)
yy = sapply(xx, function(x){x[1]})
common=names(xx)
orf = as.vector(yy)
names(common)=orf

library(devtools)
library(Biobase)
library(preprocessCore)
```

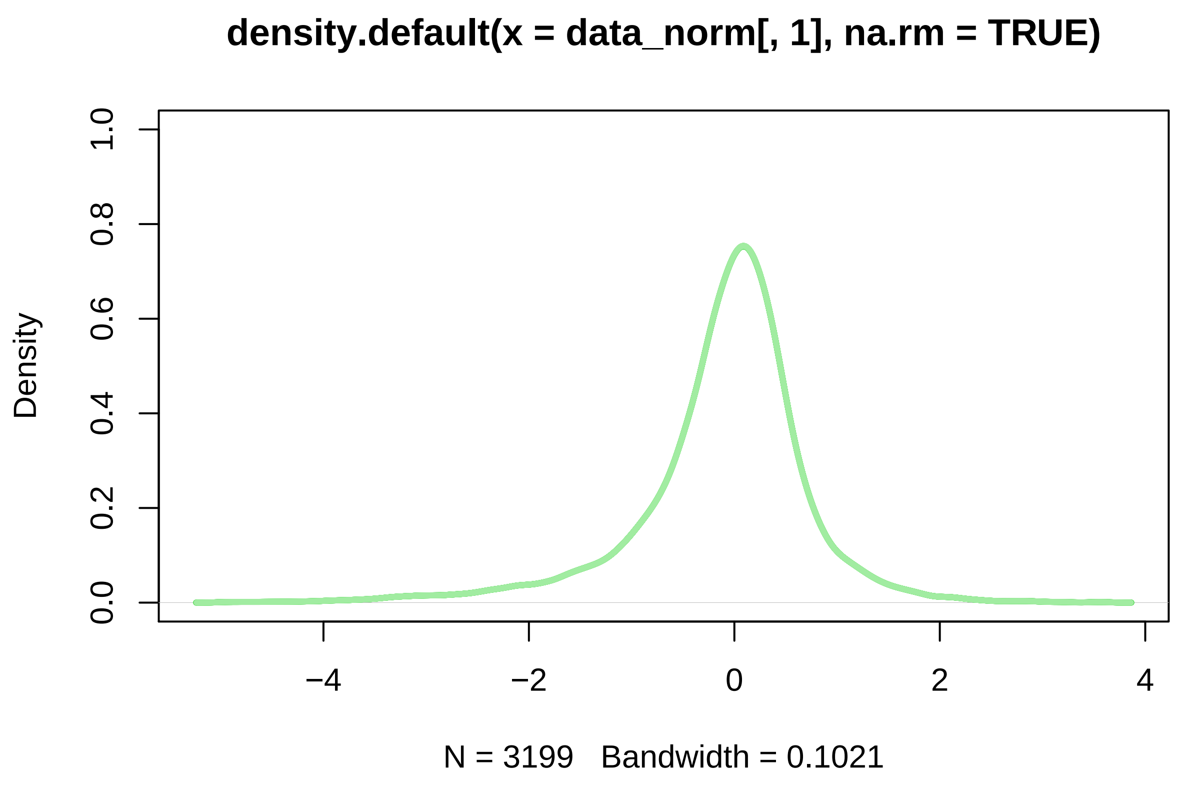
Grab data, quantile normalize, and plot

```
datagrab <- function(x,complex,mydata){
  dat <- complex[complex$V5==x,]$V3
  genes <- gsub(" ","",strsplit(as.vector(dat),";")[[1]])

  myclust <- mydata[mydata$YORF %in% genes,]
  #convert name to gene name
  row.names(myclust) <- common[as.vector(myclust$YORF)]
  View(myclust)
  return(myclust)
}

#datasets
barseq <- read.delim(file="/Volumes/GoogleDrive/My Drive/Projects/BARSeq/allBarSeq_20181112.txt",sep="\
complexes <- read.delim(file="/Volumes/GoogleDrive/My Drive/genome_data/Yeast/Datasets/yeastcomplexes.t
complexes$V5 <- paste0("complex",seq(1:dim(complexes)[1]))
complexes <- complexes[,c(5,1,2,3,4)]

#in bar-seq data, convert names to systematic
barseq$systematic <- as.vector(yy[as.vector(barseq$gene)])
sysmissing_positions <- is.na(barseq$systematic)
barseq$systematic[sysmissing_positions] = as.vector(barseq$gene[sysmissing_positions])
rmlist <- barseq$systematic[duplicated(barseq$systematic)] #get dups
barseq <-barseq[!(barseq$systematic %in% rmlist),] #remove dups

barseqYORF <- data.frame(YORF = barseq$systematic,barseq[,2:8])
```

```
#####quantile normalize
colramp = colorRampPalette(c(3,"white",2))(20)
plot(density(barseqYORF[,2],na.rm=TRUE),col=colramp[1],lwd=3,ylim=c(0,1))
for(i in 3:8){lines(density(barseqYORF[,i],na.rm=TRUE),lwd=3,col=colramp[i])}
```



```
data_norm <- normalize.quantiles(as.matrix(barseqYORF[,2:8]))
colnames(data_norm) <- colnames(barseqYORF)[2:8]

plot(density(data_norm[,1],na.rm=TRUE),col=colramp[1],lwd=3,ylim=c(0,1))
for(i in 2:7){lines(density(data_norm[,i],na.rm=TRUE),lwd=3,col=colramp[i])}
```



```
data_norm <- data.frame(data_norm)
data_norm <- cbind(barseqYORF[,1],data_norm)
colnames(data_norm)[1] ="YORF"
barseqYORF = data_norm
```

Calculate scores for complexes, as well as the fraction of complex components present in dataset

```
#get protein complex size vector
Total <- dim(complexes)[1]
ComplexesOnly <- complexes$V3

ComplexesSize <- rep(NA,Total)
for(i in 1:Total){
  ComplexesSize[i] <- length(gsub(" ","",strsplit(as.vector(ComplexesOnly[i]),";")[[1]]))
}

complexes$Size <- ComplexesSize

BigComplexes <- complexes[complexes$Size > 4,]
BigComplexes[colnames(barseqYORF[2:8])] = ""
BigComplexes["InData"] = as.numeric("")
BigComplexes["FracInData"] = as.numeric("")
BigComplexesOnly <- BigComplexes$V3
TotalBig <- dim(BigComplexes)[1]
##


for(i in 1:TotalBig){
  genes <- gsub(" ","",strsplit(as.vector(BigComplexesOnly[i]),";")[[1]])
  a <- barseqYORF[barseqYORF$YORF %in% genes,]
  a <- a[,-1]
  dat <- as.vector(apply(data.matrix(a),2,mean,na.rm=TRUE))
  names(dat) <- colnames(barseqYORF[2:8])
  BigComplexes[,names(dat)][i,] <- as.vector(dat)
  BigComplexes[,"InData"][i] = dim(a)[1]
  BigComplexes[,"FracInData"][i] = BigComplexes[,"InData"][i]/BigComplexes[,"Size"][i]
}

BigComplexes <- BigComplexes[,c(1,3,4,5,13,7:12,2,6,14,15)]


#write.table(BigComplexes,file="barseq_scores_quantilnormalize.txt",sep="\t",col.names=NA)
#########Get data for specific complex
datagrab("complex131",complexes,barseqYORF)
```

```
##          YORF YNB_NH4_GLU_20180713 YNB_NH4_GLU_20180731
## SKI7 YOR076C           -0.3841364           0.15259283
## RRP6 YOR001W           -1.4506807          -1.53658637
## MPP6 YNR024W           -0.2186391          -0.09280128
##      YNB_NH4_GLU_20180807 YNB_UREA_GLU_20180724 SC_NH4_GLU_20180827
## SKI7            0.5786852           -0.78922973           0.5082254
## RRP6           -2.5975150           -0.09250907           0.3863515
## MPP6           -0.2325251            0.14900242          -0.5335409
##      SC_UREA_GLU_20180911 SC_NH4_GAL_20181023
## SKI7           0.42149048        -0.007709257
## RRP6          -0.07004855         1.367002762
## MPP6           0.15425305         0.435788387
```