Exploratory analysis of RNA-seq dataset

gg

2016-01-18

To run this file: Rscript -e "rmarkdown::render('biplots.Rmd')"

Setup

```
library(compositions)
library(zCompositions)
library(ALDEx2)
# you can get this from:
# https://github.com/DavidRLovell/propr
source("~/git/proprBayes/R/propr-functions.R")

library(igraph)

################ incorporating phi into ALDEx
# calculate phi on Dirichlet log-ratio distributions
# returns dataframe of the lower-triangle of symmetrical phi metric
# requires David Lovell's propr.phisym function
# this depends in turn on the compositons R package
# turtles all the way down!

# we ignore all the other measures that are used for trouble-shooting phi
# the sma.df function in particular is very time and memory intensive

aldex.phi <- function(aldex.clr){

    # calculate expected value of phi
    # a single high phi value will push the component out of consideration
    # a median is right out for memory considerations

    # get first value
    sym.phi <- propr.phisym(t(sapply(getMonteCarloInstances(aldex.clr),
        function(y){y[,1]})))

    # sum the rest of the values as we proceed through the DIR MC instances
    for(i in 2:numMCInstances(aldex.clr)){
        #print(i)
        sym.phi <- sym.phi + propr.phisym(t(sapply(getMonteCarloInstances(aldex.clr),
            function(y){y[,i]})))
    }

    ##### Done ALDEx2 stuff

    # make indices of the correct size
    lt <- which(col(sym.phi)<row(sym.phi), arr.ind=FALSE)
    lt.ind <- which(col(sym.phi)<row(sym.phi), arr.ind=TRUE)

    # dataframe to hold the info,
```

```
    # data is a set of vectors where only the lower triangle is kept, because the matrix
    #    is symmetrical
    # this is needed so the subset function works properly
    sma.df <- data.frame(row=factor(rownames(sym.phi)[lt.ind[,"row"]]),
        col=factor(colnames(sym.phi)[lt.ind[,"col"]]))

    #save the lower triangle as an expected value
    sma.df$phi <- sym.phi[lt] /  numMCInstances(aldex.clr)

    return(sma.df)

}
############ end functions
```

First let us explore the data. The idea here is to identify those samples that are closely related

Filter the refseq to a mean count across samples of greater than 5

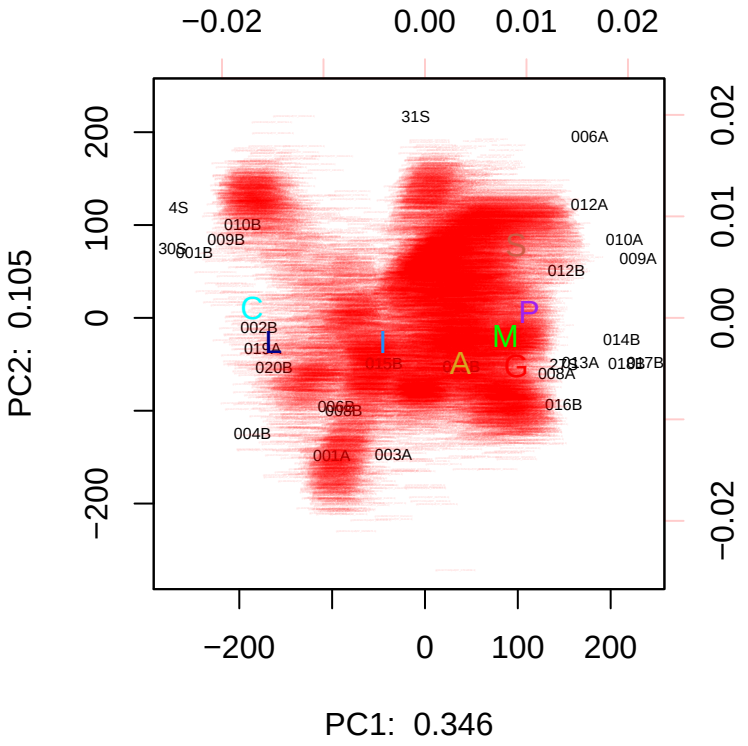
This is a form biplot that represents distances between samples with variances of components mapped onto the plot.

```
# this is the taxon table
o <- read.table(
   "~/git/twntyfr/data/merged_readcounts_taxonomy_qiime_L6_16Sorder_color_sum.txt", header=T, row.names=
)


# this is the refseq set
d <- read.table("data/merged_readcounts_subsys4.txt", header=T,
   row.names=1, check.names=F, sep="\t", comment.char="", quote="")

sub4 <- d$subsys4
d$subsys4 <- NULL

refseq <- d$refseqID
d$refseqID <- NULL

len <- d$length
d$length <- NULL

d.min <- d[which(apply(d,1,mean)> 5),]

# we need to replace 0 values with a best estimate
# use zCompositions CZM by default
# but samples must be by row, so use t()

d.n0.CZM <- cmultRepl(t(d.min), label=0, method="CZM")
```

```
## No. corrected values:  20044
```

```
# turn this into a centered log-ratio transform
# samples are by row
# remember that apply by row rotates the data
# R is terrible, so we need to use t() again
```

```
d.clr <- t( apply(d.n0.CZM, 1, function(x){log(x) - mean(log(x))}) )

d.mvar.clr <- mvar(d.clr)

d.pcx <- prcomp(d.clr)

# TODO color samples by most abundant organism
biplot(d.pcx, cex=c(0.5,0.05), col=c("black",rgb(1,0,0,0.2)), var.axes=F,
    xlab=paste("PC1: ", round(sum(d.pcx$sdev[1]^2)/d.mvar.clr, 3)),
    ylab=paste("PC2: ", round(sum(d.pcx$sdev[2]^2)/d.mvar.clr, 3)),
    scale=0
)

# filter the taxon table by what is in e.min
o.d <- o[,colnames(d)]
o.d.CZM <- cmultRepl(t(o.d), label=0, method="CZM")
o.d.clr <- t( apply(o.d.CZM, 1, function(x){log(x) - mean(log(x))}) )

# overplot taxon correlations with location of samples in the first two components

points(cor(d.pcx$x[,1], o.d.clr[,1], method="kendall") /50,
    cor(d.pcx$x[,2], o.d.clr[,1], method="kendall")/50, pch="I", col="DodgerBlue")
points(cor(d.pcx$x[,1], o.d.clr[,6], method="kendall") /50,
    cor(d.pcx$x[,2], o.d.clr[,6], method="kendall")/50, pch="L", col="DarkBlue")

points(cor(d.pcx$x[,1], o.d.clr[,2], method="kendall") /50,
    cor(d.pcx$x[,2], o.d.clr[,2], method="kendall")/50, pch="C", col="cyan")

points(cor(d.pcx$x[,1], o.d.clr[,10], method="kendall") /50,
    cor(d.pcx$x[,2], o.d.clr[,10], method="kendall")/50, pch="M", col="green")

points(cor(d.pcx$x[,1], o.d.clr[,8], method="kendall") /50,
    cor(d.pcx$x[,2], o.d.clr[,8], method="kendall")/50, pch="P", col="purple")

points(cor(d.pcx$x[,1], o.d.clr[,7], method="kendall") /50,
    cor(d.pcx$x[,2], o.d.clr[,7], method="kendall")/50, pch="G", col="red")

points(cor(d.pcx$x[,1], o.d.clr[,9], method="kendall") /50, cor(d.pcx$x[,2], o.d.clr[,9],
    method="kendall")/50, pch="A", col="Goldenrod")

points(cor(d.pcx$x[,1], o.d.clr[,12], method="kendall") /50, cor(d.pcx$x[,2],
    o.d.clr[,12], method="kendall")/50, pch="S", col="coral3")
```



This is a good representation of the data explaining 0.346 proportion of the variance in the data on the first component, and a much smaller amount on the second and later components. This reduced dataset contains 19496 genes. We can see that the samples cluster into a few fairly discrete groups. These groups contain fairly discrete sets of genes, and we can infer that the sample set distributions are driven by gene occurrence in different genomes.

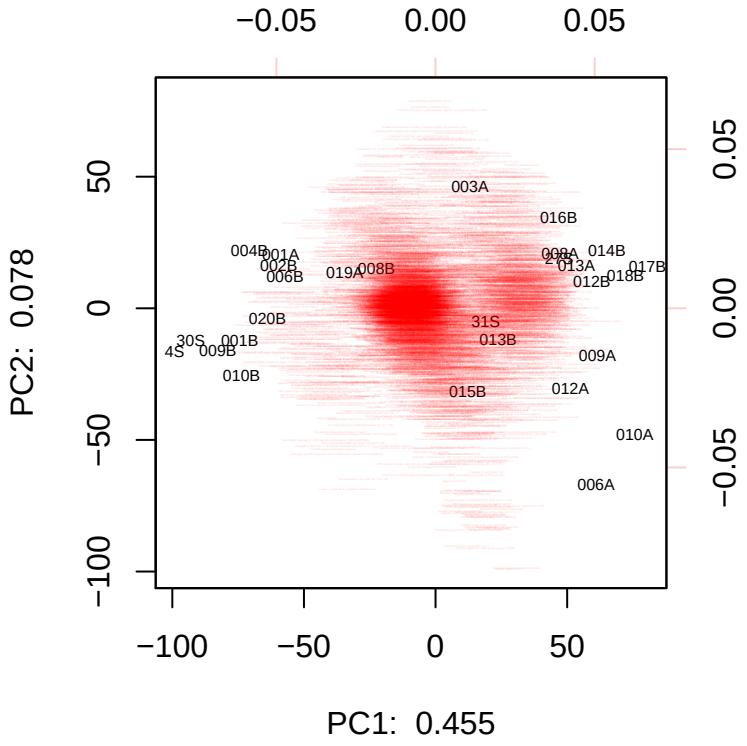
To get around the gene occurrence problem, we aggregate the data to the SEED subsystem 4 level.

```
e <- read.table("data/AitchisonTransformedDataForALDExInput.txt", header=T,
   row.names=1, check.names=F, sep="\t", comment.char="", quote="")

e.n0.CZM <- cmultRepl(t(e), label=0, method="CZM")
```

```
## No. corrected values:  2869
```

```
# turn this into a centered log-ratio transform
# samples are by row
# remember that apply by row rotates the data
# R is terrible, so we need to use t() again

e.clr <- t( apply(e.n0.CZM, 1, function(x){log(x) - mean(log(x))}) )

e.mvar.clr <- mvar(e.clr)

e.pcx <- prcomp(e.clr)

# TODO color samples by most abundant organism
biplot(e.pcx, cex=c(0.5,0.05), col=c("black",rgb(1,0,0,0.2)), var.axes=F,
    xlab=paste("PC1: ", round(sum(e.pcx$sdev[1]^2)/e.mvar.clr, 3)),
    ylab=paste("PC2: ", round(sum(e.pcx$sdev[2]^2)/e.mvar.clr, 3)),
```

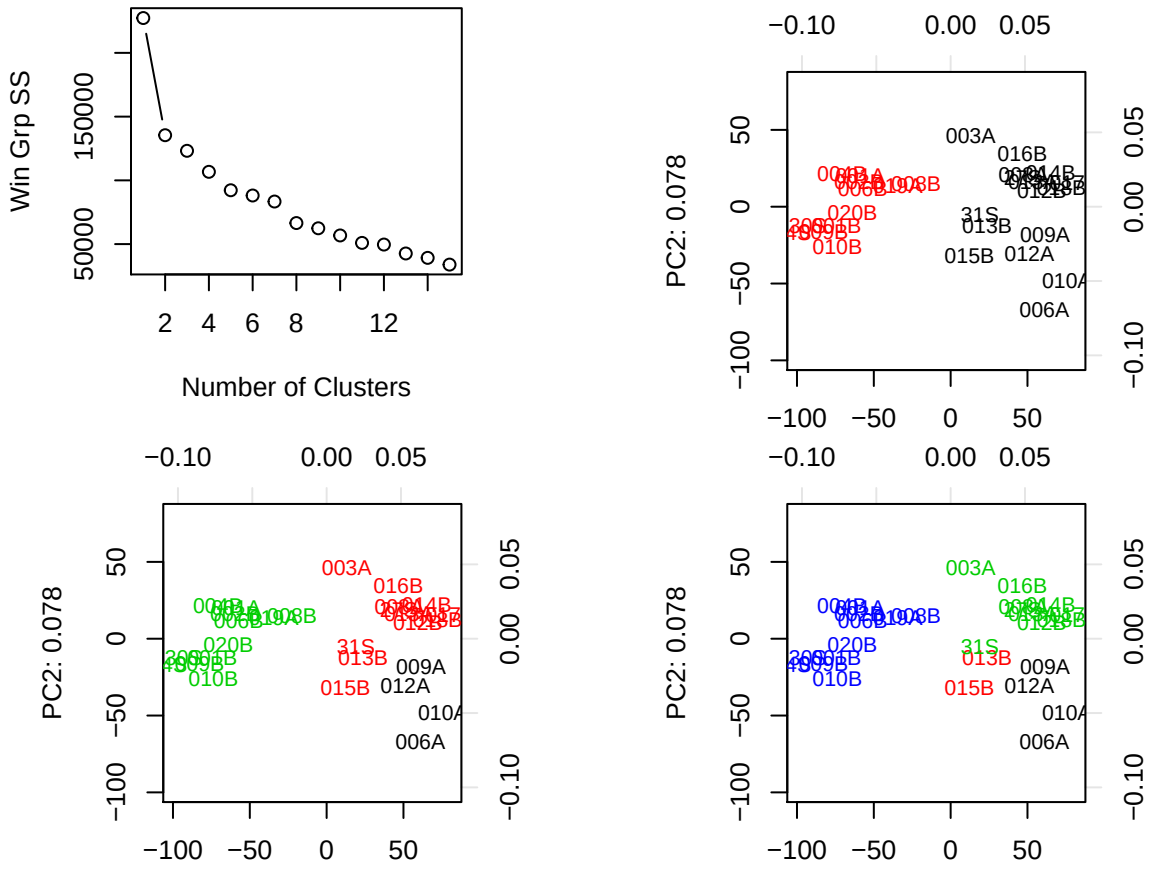
```
    scale=0
)
```



This is a much better representation of the data. We did not bother filtering by abundance since there is a much smaller number of features (3874). Now we can see a fairly clear split in the data that is less driven by gene occurrence and more by functions within the microbiome (cite Beiko review regarding ecosystems of genes rather than organisms here!).

```
mydata <- as.data.frame(e.clr)

# calculate the within group sum of squares
wss <- (nrow(mydata) -1) * sum(apply(mydata,2,var))

# summarize the fit for the first 15 groups
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
        centers=i)$withinss)

# the plot suggests a reasonable fit can be had with about 2-3 clusters
# and after that there are only small improvements with additional clusters
layout( matrix(c(1,2,3,4),2,2, byrow=T), widths=c(10,10), height=c(10,10))

par(mar=c(5,4,0,5)+0.1)

# make the sum of squares distance plot
plot(1:15, wss[1:15], type="b", xlab="Number of Clusters",
    ylab="Win Grp SS")

par(mar=c(2,0,2,0)+0.1)

# plot the data for cluster sizes 2-4
```

```
for(i in 2:4){
    fit <- kmeans(mydata,i)
    mydata$fit.cluster <- NULL # clear any previous data
    mydata <- data.frame(mydata, fit$cluster) # add the cluster data

    coloredBiplot(e.pcx,col=rgb(0,0,0,0.1),cex=c(0.8,0.05),
     xlab=paste("PC1: ", round(sum(e.pcx$sdev[1]^2)/e.mvar.clr, 3), sep=""),
     ylab=paste("PC2: ", round(sum(e.pcx$sdev[2]^2)/e.mvar.clr, 3), sep=""),
     xlabs.col=mydata$fit.cluster, arrow.len=0.05, var.axes=F, expand=0.8,  scale=0)
}
```



K-means clustering supports only 2 groups. In fact, only the first plot needs to be shown to support this: maybe include the barplot?

Samples 3A, 31S, 13B and 15B are near the midpoint on the right, and 19A and 8B are near the midpoint on the left. Five of these six microbiomes are atypical, and the other 19A is exclusively L. iners, but phenotypically is BV. We next exclude these six samples and re-examine the partitioning.

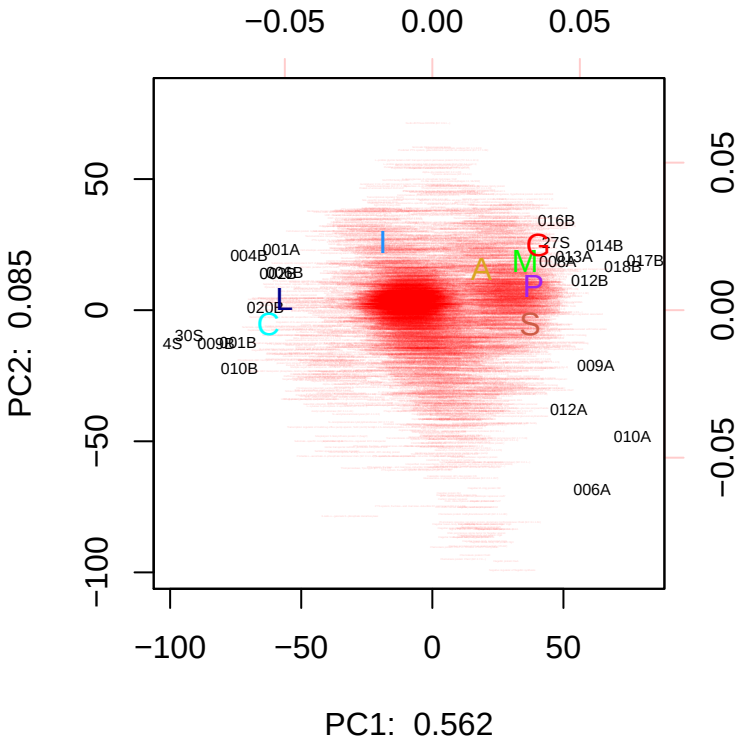
```
e.min <- e

e.min[,"003A"] <- NULL
e.min[,"31S"] <- NULL
e.min[,"013B"] <- NULL
e.min[,"015B"] <- NULL
e.min[,"019A"] <- NULL
e.min[,"008B"] <- NULL
```

```
e.min.n0.CZM <- cmultRepl(t(e.min), label=0, method="CZM")
```

```
## No. corrected values:  545
```

```
# turn this into a centered log-ratio transform
# samples are by row
# remember that apply by row rotates the data
# R is terrible, so we need to use t() again

e.min.clr <- t( apply(e.min.n0.CZM, 1, function(x){log(x) - mean(log(x))}) )

e.min.mvar.clr <- mvar(e.min.clr)

e.min.pcx <- prcomp(e.min.clr)

# TODO color samples by most abundant organism
biplot(e.min.pcx, cex=c(0.5,0.05), col=c("black",rgb(1,0,0,0.2)), var.axes=F,
    xlab=paste("PC1: ", round(sum(e.min.pcx$sdev[1]^2)/e.min.mvar.clr, 3)),
    ylab=paste("PC2: ", round(sum(e.min.pcx$sdev[2]^2)/e.min.mvar.clr, 3)),
    scale=0
)

# filter the taxon table by what is in e.min
o.min <- o[,colnames(e.min)]
o.CZM <- cmultRepl(t(o.min), label=0, method="CZM")
o.clr <- t( apply(o.CZM, 1, function(x){log(x) - mean(log(x))}) )

# overplot taxon correlations with first two components
cor.scale.0 <-  max(abs(1/range(e.min.pcx$rotation[,1:2])))

points(cor(e.min.pcx$x[,1], o.clr[,1], method="kendall") /cor.scale.0,
    cor(e.min.pcx$x[,2], o.clr[,1], method="kendall")/cor.scale.0, pch="I",
    col="DodgerBlue")
points(cor(e.min.pcx$x[,1], o.clr[,6], method="kendall") /cor.scale.0,
    cor(e.min.pcx$x[,2], o.clr[,6], method="kendall")/cor.scale.0, pch="L",
    col="DarkBlue")

points(cor(e.min.pcx$x[,1], o.clr[,2], method="kendall") /cor.scale.0,
    cor(e.min.pcx$x[,2], o.clr[,2], method="kendall")/cor.scale.0, pch="C", col="cyan")

points(cor(e.min.pcx$x[,1], o.clr[,10], method="kendall") /cor.scale.0,
    cor(e.min.pcx$x[,2], o.clr[,10], method="kendall")/cor.scale.0, pch="M", col="green")

points(cor(e.min.pcx$x[,1], o.clr[,8], method="kendall") /cor.scale.0,
    cor(e.min.pcx$x[,2], o.clr[,8], method="kendall")/cor.scale.0, pch="P", col="purple")

points(cor(e.min.pcx$x[,1], o.clr[,7], method="kendall") /cor.scale.0,
    cor(e.min.pcx$x[,2], o.clr[,7], method="kendall")/cor.scale.0, pch="G", col="red")

points(cor(e.min.pcx$x[,1], o.clr[,9], method="kendall") /cor.scale.0,
    cor(e.min.pcx$x[,2], o.clr[,9], method="kendall")/cor.scale.0, pch="A",
    col="Goldenrod")
```

```
points(cor(e.min.pcx$x[,1], o.clr[,12], method="kendall") /cor.scale.0,
    cor(e.min.pcx$x[,2], o.clr[,12], method="kendall")/cor.scale.0, pch="S", col="coral3")
```

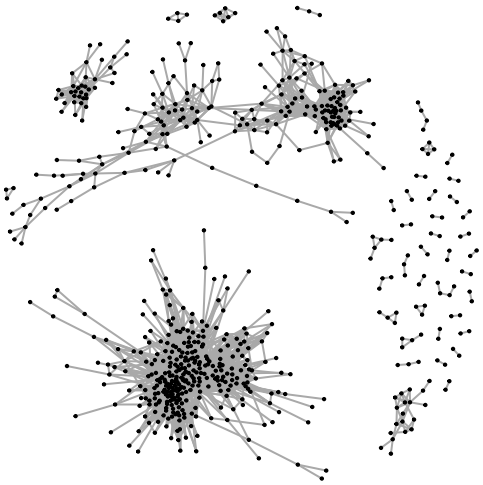



```
###### phi with Bayesian estimation
# the problem is that a low phi can easily arise between functions
# that have the same distribution of 0 values across samples
# so we need to estimate the value of 0 (and other low-count functions)

# generate random DIR clr instances with ALDEx2
e.x <- aldex.clr(e.min)
```

```
## [1] "operating in serial mode"
```

```
# calculate phi divided by number of random instances
e.min.sma.df <- aldex.phi(e.x)

# find the set of connections with phi less than some value
# we choose an arbitrary cutoff, but it is higher after Bayesian estimation
# obviously
 phi.cutoff <- 0.03

e.min.sma.lo.phi <- subset(e.min.sma.df, phi < phi.cutoff)

# generate a graphical object
g <- graph.data.frame(e.min.sma.lo.phi, directed=FALSE)

# overview of all the proportional relationships
# this can take a long time!!!
```

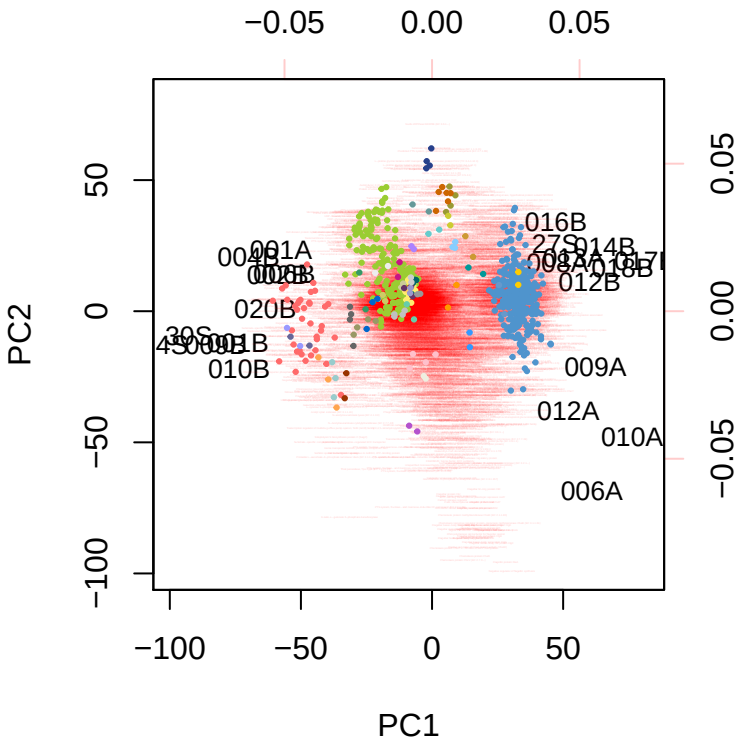
```
plot(g, layout=layout.fruchterman.reingold.grid(g, weight=0.05/E(g)$phi), vertex.size=1,
    vertex.color="black", vertex.label=NA)
```



```
# # get the clusters from the graph object
 g.clust <- clusters(g)

# write them to a file

# # data frame containing the names and group memberships of each cluster
g.df <- data.frame(Systematic.name=V(g)$name, cluster=g.clust$membership,
    cluster.size=g.clust$csize[g.clust$membership])
write.table(g.df, file="g.df.txt", sep="\t", quote=F, col.names=NA)

max.clust <- induced.subgraph(g, which(g.clust$membership %in% which(g.clust$csize ==
    max(g.clust$csize))))
max.clust.names <- V(max.clust)$name
```

```
par(mfrow=c(1,1))

biplot(e.min.pcx,col=c("black", rgb(1,0,0,0.2)),cex=c(0.8,0.05),
var.axes=F, scale=0)

#nms <- rownames(g.df)[g.df$cluster==max.clust.names]

#points(e.min.pcx$rotation[max.clust.names,][,1],e.min.pcx$rotation[max.clust.names,][,2],
#    col=rgb(0,0,1,0.2),pch=19,cex=0.2)

colours <- c("steelblue3", "skyblue1", "indianred1", "mediumpurple1", "olivedrab3",
   "pink", "#FFED6F", "mediumorchid3", "ivory2", "tan1", "aquamarine3", "#C0C0C0",
   "royalblue4", "mediumvioletred", "#999933", "#666699", "#CC9933", "#006666", "#3399FF",
   "#993300", "#CCCC99", "#666666", "#FFCC66", "#6699CC", "#663366", "#9999CC", "#CCCCCC",
   "#669999", "#CCCC66", "#CC6600", "#9999FF", "#0066CC", "#99CCCC", "#999999", "#FFCC00",
   "#009999", "#FF9900", "#999966", "#66CCCC", "#339966", "#CCCC33", "#EDEDED"
)

big <- g.df[which(g.df$cluster.size >= 2),]
colnames(big) <- colnames(g.df)
```

```
lev <- factor(big$cluster)
for(i in as.numeric(levels(lev))){
nms <- rownames(big)[big$cluster==i]
#print(rownames(big)[big$cluster==i])
#print("")
points(e.min.pcx$rotation[nms,][,1],e.min.pcx$rotation[nms,][,2], col=colours[i],
    pch=19, cex=0.3)
}
```



Here the vast majority of low phi values come from core functions that are constant ratios in all samples, but not differential. A few smaller groups are both differential and compositionally associated.

Incorporating the Bayesian estimate of 0 gives a much clearer picture, than using point estimates. When point estimates are used, then the majority of the clusters are near 0,0 - simply because of sets of functions that share the same distribution of 0 across samples.

The relationship between phi and effect size.

```
    B <- match(rownames(e.min.pcx$x)[e.min.pcx$x[,1] > 0], rownames(e.min.pcx$x))
    H <- match(rownames(e.min.pcx$x)[e.min.pcx$x[,1] < 0], rownames(e.min.pcx$x))

    conds <- vector()
    conds[B] <- "B"
    conds[H] <- "H"

    x.e <- aldex.effect(e.x, conds, verbose=FALSE)
```

```
## [1] "operating in serial mode"
```

```
    plot(x.e$diff.win, x.e$diff.btw, pch=19,cex=0.4, col=rgb(0,0,0,0.2))

    abline(0,2, lty=2, col="grey")
    abline(0,-2, lty=2, col="grey")
    abline(0,1, lty=3, col="grey")
    abline(0,-1, lty=3, col="grey")

    for(i in as.numeric(levels(lev))){
    nms <- rownames(big)[big$cluster==i]
    #print(rownames(big)[big$cluster==i])
    #print("")
    points(x.e[nms,"diff.win"],x.e[nms,"diff.btw"], col=colours[i], pch=19, cex=0.5)
}
```