Notes are in black --
code to cut and paste into R is in purple
Notes to be ignored

When there are multiple lines of code, you can cut and paste all of them at once - if code is separated with a blank line - cut/paste the blank line separated one at a time - for example

cut/paste_these_together
cut/paste_these_together

cut/paste_these_sperately

cut/paste_these_sperately

NOTE:
this has to be run first to change the default server:
mconfig$server("http://test.metagenomics.anl.gov/api.cgi/")
[do this AFTER:
> library (matR)
--dtb]
##################################
Begin Tutorial >

We'll work through a typical data analysis
We'll be using annotation abundance data generated via MG-RAST for a series
of different animal guts

We start by looking through public data available to KBase -- in this case, metagenomic sequence and annotation
data available as public data in MG-RAST

Specifically, we'll use the client to download annotation abundance profiles for a number of samples through MG-RAST,
we'll perform some preliminary analyses to determine if samples can be used in a comparative analysis, and then we'll work through
a typical analysis.

We start with a collection of 14 samples that represent 4 distinct biomes, the guts of chicken, mouse, cow, and fish

First, we'll download a single data set, and see how we can use matR to explore it.
The data are in the form of functional annotation based abundance profiles for a series of gut samples:

4 samples from Chicken cecum	“chic”
3 from cow rumen	“cow”
2 from mouse gut	“mus”
2 from fish gut	“fish”

We create a list that has the sample name and mgrast id for each of our samples:

```
gut_samples <- c (chic.1= "4440283.3",
chic.2 = "4440284.3",
chic.3 = "4440285.3",
chic.4 = "4440286.3",
cow.1 = "4441679.3",
cow.2 = "4441680.3",
cow.3 = "4441682.3",
mus.1 = "4440463.3",
mus.2 = "4440464.3",
fish.1 = "4441695.3",
fish.2 = "4441696.3"
)
```

Now we download the data for these samples into an R object called my_gut_data

```
my_gut_data <- collection(gut_samples)
```

It could take a couple minutes for the data to load
By default, you will download the following data for each of the samples
- raw abundance counts
`my_gut_data$count`
- abundance counts that have been normalized and standardized per sample
`my_gut_data$normed`
- average e value for each annotation
`my_gut_data$evalue` [something is messed up with this call. the data is fubar'd]
- average read lengths for each annotation
`my_gut_data$length`
- average percent identity for similarities calculated for each annotations
`my_gut_data$percentid`

We can look at the metadata for each metagenome
First we extract all of the metadata for the group of samples in our collection

```
my_metadata <- metadata(my_gut_data)
```

Then we can look at the "names" property to see the mgrast id's for each of the samples

```
names(my_metadata)
```

We take the first id, and pull out the metadata that corresponds to just that sample

```
single_sample_metadata <- my_metadata[["4440283.3"]]
```

We can look at all of the metadata for the sample

```
single_sample_metadata
```

Or drill down to look at individual metadata values - examples below pull out individual metadata values:

[for copying & pasting it's handy to have purple text on separate lines from non-command text]
single_sample_metadata$name user given name

single_sample_metadata$metadata$library$data$seq_center sequencing center

single_sample_metadata$metadata$library$data$seq_meth sequencing method/technology

single_sample_metadata$metadata$sample$data$biome biome

Let's say we want to look at the same metadata value for all of the samples, we can do this:

```
for (i in 1:dim(as.matrix(names(my_metadata)))[1]){
        print(
                paste( names(my_metadata)[i], "   ",
                                (
        my_metadata[[names(my_metadata)[i]]]$metadata$library$data$metagenome_name
                                )
                        )
                )
}
```

Here we see the mgrastid and the user given sample name
use this as the middle line and you can see the biome

my_metadata[[names(my_metadata)[i]]]$metadata$sample$data$biome

This to see the sequencing technology

my_metadata[[names(my_metadata)[i]]]$metadata$library$data$seq_meth

This is a little clunky now, but in the very near future this will allow us to easily sort and group samples by their metadata.

Before we perform comparative analyses, we want to take a look at the data distributions. In this case, the distributions of the raw abundance counts for each of the samples:

render (my_gut_data, views = "count")
[use parameter main = "alternate title" if you want]

Here we can clearly see bias in the distributions of the raw counts -- we can eliminate the majority of this bias if we normalized and standardize the abundance data from each sample. Now we look at the distribution of the raw (top) and normalized (bottom) counts:

render (my_gut_data, views = c ("count", "normed"))

Normalization/standardization of the abundance profiles has made them more comparable

We can perform a preliminary comparative analysis with a PCoA based analysis of the data:

```
my_PCoA <- mpco (my_gut_data$normed, method = "euclidean")
```

We can look at the raw output of the analysis like this:

```
my_PCoA
```

or, we can produce a visualization of the results like this:

```
render (my_PCoA, main = "PCoA of gut data", labels = names(gut_samples))
```

Interesting; it's pretty clear that the abundance profiles cluster by host. We can make this a little clearer if we color the data by organism. First, we create a list that we can use to color the sample in the PCoA, and to establish groupings for subsequent statistical tests:

```
my_color_groupings <- c ( rep("red", 4), rep("blue", 3), rep("green", 2), rep("hotpink",2))
```

Now we can create a PCoA with each of the groups colored:

```
render (my_PCoA, main = "Colored PCoA of gut data", col = my_color_groupings, labels =
names(gut_samples))
```

Another way to visualize these data is with a heatmap dendrogram, like this:

```
mheatmap (my_gut_data$normed, image_out = "my_gut_data.heatmap_dendrogram.jpg",
labRow = NA, labCol = names(gut_samples), col_lab_mult = 1.2, margins = c (9,1), image_title
= "my_gut_data.heatmap_dendrogram")
```

This may take a minute or two - when it's finished, we can open it from R like this:

```
system("open my_gut_data.heatmap_dendrogram.jpg")
```

Each line represents the abundance of a single subsystem (red = low to green = high). Once again, we clearly see that functional content of gut microbes varies as a function of host -- but we could have guessed that before sequencing. What we really want to get at is more specific information: what functions are responsible for the differences among the hosts --- in other words, which subsystems exhibit statistically significant differences in abundance among the 4 types of host (chicken, cow, mouse, and fish). We can use the groupings from before to perform a statistical test -- remember the boxplots? -- they indicate that the data are normally (well almost) distributed, justifying the use of a parametric test, in this case, ANOVA:

```
my_gut_ANOVA_stats <- doStats(my_gut_data$normed, my_color_groupings, "ANOVA-one-
way")
```

We can look at the raw results this way:

```
my_gut_ANOVA_stats
```

That's not too much use -- what we want to do is use the statistical output to subselect the data. i.e. use the results of the statistical analysis to select the subsystems/functions that exhibit the most significant differences among the 4 examined hosts. First we pull the p values out of the statistical analysis results:

```
my_p_values <- my_gut_ANOVA_stats[ ,6]
```

Now we subselect based on relatively stringent criteria to help control for the FDR - we'll use a p value of 0.05 as the threshold, but then apply the Bonfferoni adjustment (0.05/num_tests = 0.05/1040 = 5E-07) - first, lets see if any of the data exhibit this level of significance:

```
min(my_p_values)
```

dough, this is too stringent. We need to try something else -- here we can use another R package to easily use an alternative means to apply FDR control. We'll use q value analysis. Install and load the qvalue package:

```
source("http://bioconductor.org/biocLite.R")
biocLite("qvalue")
library(qvalue)
```

[an annoying thing that can happen in the middle of this sequence of commands (specifically, after the second) is that you'll be prompted to update packages used by qvalue for which a new version is available]

[also this crashed hard for me just now, I think due to version mismatches between R and various packages, although it ran fine the 2nd time. something to be careful about. R stuff is very finicky about versions.]

Perform a conventional q value analysis of the p values -- we'll require a relatively stringent FDR level

```
my_q_value_analysis <- qvalue(my_p_values, fdr.level = 0.001)
```

Now we can subselect just those subsystems that exhibit an FDR of 0.001 or better:

```
my_FDR_controlled_gut_data <- my_gut_data$normed [names
(my_q_value_analysis$significant) [my_q_value_analysis$significant==TRUE], ]
```

and we can generate visualizations and or simple table outputs for the data that pass FDR control:

We can generate a PCoA like this:

```
my_FDR_controlled_PCoA <- mpco (matrix(my_FDR_controlled_gut_data@x,ncol=11), method
= "euclidean")

render (my_FDR_controlled_PCoA, main = "(FDR <= 0.001) Colored PCoA of gut data", col =
my_color_groupings, labels = names(gut_samples))
```

or a heatmap like this:

```
mheatmap (matrix(my_FDR_controlled_gut_data@x,ncol=11), image_out
= "my_subselected_gut_data.heatmap_dendrogram.jpg", labRow = NA, labCol
= names(gut_samples), col_lab_mult = 1.2, margins = c (9,1), image_title
= "my_subselected_gut_data.heatmap_dendrogram")
```

```
system("open my_subselected_gut_data.heatmap_dendrogram.jpg")
```

Note that for these last two options, we're using a lot more R code -- this will be replaced with more user friendly functions for C and D users, but A and B users will be able to access the R code to customize analyses and their visualizations.

If we compare the original PCoA and heatmap-dendrograms to those created from the statistically subselected data, we can see that the differences among the 4 groups of hosts become much more distinct

Finally - if we want to write the analysis outputs to files (for Excel, Matlab, etc.) we can do this:

First - pick out the data we want to write to file - here we will output the abundance profiles and statistical results for all data considered in our analysis:

```
my_output_data <- cbind(as.matrix(my_gut_data$normed), my_gut_ANOVA_stats,
matrix(my_q_value_analysis$qvalues, ncol=1), matrix(my_q_value_analysis$significant, ncol=1)
)
dimnames(my_output_data)[[2]][18] <- "qvalue"
dimnames(my_output_data)[[2]][19] <- "pass FDR <= 0.001"

write.table(my_output_data, file = "my_output_analysis.txt", col.names=NA, row.names =
TRUE, sep="\t", quote=FALSE)
```

DONE - Just notes from here on.

```
my_metadata[[names(my_metadata)[i]]]$metadata$library$data$metagenome_name

my_metadata[[test]]$metadata$sample$data$biome
my_metadata[[names(my_metadata)[1]]]$metadata$sample$data$biome

for (i in 1:dim(as.matrix(names(my_metadata)))[1]){print(names(my_metadata)[i])} # print ids
my_metadata[["4440283.3"]]$metadata$library$data$seq_center # value from 1 clctn sample
```

pvals <- results [, 4]
subselection

- raw subselection data

- images created from subselection

subW <- my_gut_data$normed [names (*my_q_value_analysis$significant)*
*[*my_q_value_analysis$significant==TRUE*],]*

pvals <- results [, 4]
subW <- W$normed [names (pvals) [pvals < 0.0000005],]

comparing dimensions of the original and subselection matrices shows what
proportion of the original functional annotations are retained:

dim (W$normed)
dim (subW)

finally, a heatmap of the subselected matrix highlights rows of interest more clearly than before:

mheatmap (subW, image_out = "waters_sub_HD.jpg", labRow = NA, labCol = names (waters), col_lab_mult = 1.2, margins = c (8,1), image_title = "subselection")
system ("open waters_sub_HD.jpg")

this analysis could continue by setting more restrictive p-value thresholds, like this:

subW01 <- W$normed [names (pvals) [pvals < 0.01],]
subW003 <- W$normed [names (pvals) [pvals < 0.003],]

Looks as though fish exhbit the most unique aundance profiles, lets analyze again with just two groups, fish, and everything else:

my_color_groupings.2 <- c (rep("blue", 9), rep("hotpink",2)) ...

write p value analyses to a tab delimited text you can open in excel

write.table(my_stats, file = "my_ANOVA_P_values.txt", col.names=NA, row.names =TRUE, sep="\t", quote=FALSE)

or get a little fancier and produce a table that has the normalized abundance values followed by the ANOVA analyses

my_output_data <- cbind(as.matrix(my_data$normed_function_counts),my_stats)
write.table(my_output_data, file = "my_normed_abundances_and_ANOVA_P_values.txt", col.names=NA, row.names = TRUE, sep="\t", quote=FALSE)

QUESTIONS FOR DAN:

if I want to get the metadata for a single sample in a collection, what is the syntax?

I know metadata(collection) gives me the whole bag
mm <- metadata (<ids>)

or alternatively, if I want to see just a certain class of metadata for the collections -- say all of the $id in the collection
to access a given element use mm$<id>$field$subfield$...$subfield
or:
j <- c (“<id>”, “field”, “subfield”, …)
mm [[j]]
but there is not presently a convenient way to get the same field from every one of multiple ids. (there’s a slight problem with the very idea, since different metagenomes are not guaranteed to _have_ any of the same metadata).

for the boxplot functions - what is the syntax to change names from the default (mgrast ids)
pass a parameter names = c (...) to “render”

if I want to see all of the counts, what is the syntax -- if I do my_collection$counts I get NULL
use count not counts

does the current version have q values?
no

We need to change the stat output to include the group averages
We also need to add (probably as first line) the list of sample that are in each group of any statistical analysis
ok

NOTES FROM HERE ON
NOTES FROM HERE ON
NOTES FROM HERE ON
NOTES FROM HERE ON
NOTES FROM HERE ON

Here we’ve used the defaults - subsystem level 3 -- but we can download any data type at any relevant level, for example -- if we wanted to download the level 2 subsystem data we would do the following:

my_gut_data.subsystem_lvl2 <- collection (gut_samples, function_counts = view(of = "count", annotation="function", level="level2"))

Note that because this is not a default data call, it will only collect the data requested - the raw count abundances.

```
standardViews <- list (
   count = view (of = "count"),
   normed = view (of = "normed"),
   evalue = view (of = "evalue"),
   length = view (of = "length"),
   percentid = view (of = "percentid"))
```
[8/7/12 3:27:50 PM] Kevin Keegan: ... for the selected mgm's?

```
<-
scrubIds("4440283.3;4440284.3;4440285.3;4440286.3;4440463.3;4440464.3;4441679.3;44416
80.3;4441681.3;4441682.3;4440055.3;4440056.3;4440062.3;4440063.3")
```

```
c("Chicken.1","Chicken.2","Chicken.3","Chicken.4","LM","OM","CR1","CR2","CR3","CR4","FG1",
"FG2","FG3","FG4")
```

CC = Chicken cecum
LM = Lean mouse
OM = Obese mouse
CR = Cow rumen
FG = Fish gut

Download data -- if you need to get data other than subsystem level 3, just let me know and # I or Dan can give you the right syntax -- documentation for the package is a bit spotty at the moment

```
my_data <- collection (my_data_mgids, function_counts = view(of = "count",
annotation="function", level="level3"), normed_function_counts = view(of = "normed",
annotation="function", level="level3"))
```

We have a single data object (my_data) that contains the raw (function_counts) and normalized (normed_function_counts) counts for your data

Let's start by look at boxplot of the abundance distributions in the raw and normalized data

```
render (my_data, views = c ("function_counts", "normed_function_counts"))
```

I'd stick with the normed counts -- distributions look a whole lot better (more similar and normally distributed -- with the exception of the last two)

now let’s take a peek at how similar the abundance profiles are for the data

we'll produce a PCoA of the normalized values using Euclidean distance

```
my_PCoA <- mpco (my_data$normed_function_counts, method = "euclidean")
```

create a plot from the first two components

```
render (my_PCoA, main = "This is a test PCoA", labels = my_data_names)
```

Notice that the last two samples look extremely different from the rest -- these are the two with the wacky distributions -- lets do that again, excluding those two samples:

```
my_data_mgids <-
scrubIds("4440283.3;4440284.3;4440285.3;4440286.3;4440463.3;4440464.3;4441679.3;4441680.3;4441681.3;4441682.3;4440055.3;4440056.3")

my_data_names <-
 c("CC1","CC2","CC3","CC4","LM","OM","CR1","CR2","CR3","CR4","FG1","FG2")

my_data <- collection (my_data_mgids, function_counts = view(of = "count", annotation="function", level="level3"), normed_function_counts = view(of = "normed", annotation="function", level="level3"))

my_PCoA <- mpco (my_data$normed_function_counts, method = "euclidean")

render (my_PCoA, main = "This is a test PCoA", labels = my_data_names)
```

We see some grouping that makes sense -- lets color the points to make this a little clearer. We'll create a vector that has the sample colors, and can also be used to group the samples for the statistical tests below.

```
my_groupings <- c ( rep("red", 4), rep("blue", 2), rep("green", 4), rep("hotpink",2))

render (my_PCoA, main = "This is a colored test PCoA", labels = my_data_names,  col = my_groupings)
```

CC = Chicken cecum (red)
LM = Lean mouse (blue)
OM = Obese mouse (blue)
CR = Cow rumen (green)
FG = Fish gut (hotpink)

Another way to visualize the data is with a heatmap dendrogram

```
mheatmap (my_data$normed_function_counts, image_out = "my_data_HD.jpg", labRow = NA, labCol = my_data_names, col_lab_mult = 1.2, margins = c (9,1), image_title = "my_data_heatmap_dendrogram")
```

now lets see which subsystems exhibit the most significant differences among these four groups - we'll use ANOVA on the normalized data (parametric tests are justified by the normality we observed in the (normalized data) boxplots)

```
my_stats <- doStats(my_data$normed_function_counts, my_groupings, "ANOVA-one-way")
```

write p value analyses to a tab delimited text you can open in excel

```
write.table(my_stats, file = "my_ANOVA_P_values.txt", col.names=NA, row.names =TRUE, sep="\t", quote=FALSE)
```

or get a little fancier and produce a table that has the normalized abundance values followed by the ANOVA analyses

```
my_output_data <- cbind(as.matrix(my_data$normed_function_counts),my_stats)
write.table(my_output_data, file = "my_normed_abundances_and_ANOVA_P_values.txt", col.names=NA, row.names = TRUE, sep="\t", quote=FALSE)
```

It looks as though the fish gut samples are very different from the rest, we can perform another analysis where we compare the data as two groups:

```
my_groupings <- c ( rep("red", 10), rep("hotpink",2))

my_stats <- doStats(my_data$normed_function_counts, my_groupings, "ANOVA-one-way")
```

ANOVA will be the same as a t-test (just two samples)

```
my_output_data <- cbind(as.matrix(my_data$normed_function_counts),my_stats)
write.table(my_output_data, file = "my_normed_abundances_and_ANOVA_P_values.txt", col.names=NA, row.names = TRUE, sep="\t", quote=FALSE)
```

CC = Chicken cecum (red)
LM = Lean mouse (red)
OM = Obese mouse (red)
CR = Cow rumen (red)
FG = Fish gut (hotpink)
In addition to this demo, the package comes with another demo pre-installed -- it repeats some of the functionality we've covered here as well as additional features - retrieval of metadata, subselecting data sets based on statistical analysis outputs etc. Note that the entire demo will take a few minutes to run through --

```
demo2()
```

Press enter to proceed through each step in this demo

Notes from here on - please ignore

dutilh@cmbi.ru.nl, itai.sharon@gmail.com, k6logc@gmail.com, nathaniel.hubert@gmail.com, wltrimbl@gmail.com, mesude@gmail.com, tgaw@msu.edu, wtang222@gmail.com, anadimpalli1@gmail.com, weigandenator@gmail.com, adina.chuang@gmail.com, googled@d-dub.org.uk, kmhandley@googlemail.com, folker.meyer@gmail.com, dantonop@gmail.com, xzhu1@uchicago.edu,

Notes - ignore

> pvals <- results [, 4]

```
> subW <- W$normed [names (pvals) [pvals < 0.05], ]
```