Exercise 4: Metagenomics - Oilspill Analysis

Code

Submit until Friday 22 December, 2017

Notice

This exercise is an RMarkdown (http://rmarkdown.rstudio.com/) notebook composed of text and code which you can load into RStudio, run the code, record the output and fill in missing parts. This is a form of literate programming (https://en.wikipedia.org/wiki/Literate_programming) which has gained much attention in science lately. Markdown syntax is simple text with additional syntax and the RStudio website has a good reference card for RMarkdown (https://www.rstudio.com/resources/cheatsheets/). Before you submit your solution, make sure that your entire notebook runs through and generated corresponding output. You can then turn in your report as a generated HTML file together with your modified Rmd file. However, if you want, you can also follow the exercise manually and create your report in another program. Make sure to fill in your names below.

```
**Name: FIRST NAME HERE**
**Name: SECOND NAME HERE**
```

Background

In this exercise we will analyze environmental metagenome data. The data comes from the study of Mason et al. (2014) (https://doi.org/10.1038/ismej.2013.254) and derives from marine sediments taken to investigate in the Deepwater horizon oilspill in the Mexican Gulf (https://en.wikipedia.org/wiki/Deepwater_Horizon_oil_spill) which had a severe impact on the ecosystem. Here is a summary of the samples is presented in the table below. Three samples were taken close to the oilspill well head, and three samples taken further away are used as control. We will use 16S sequences obtained by amplicon sequencing (https://en.wikipedia.org/wiki/Amplicon) in the first part and and shotgun metagenomics sequences in the second part of this exercise sheet.

The samples where taken at different places around the wellhead and the contamination of the samples was measured in terms of polycyclic aromatic hydrocarbons (PAHs) and set as low or high contamination level.

Sample ID	Group	Distance from wellhead (km)	Total PAH (g/kg)
BP278	Highly contaminated	2.7	6041.22
BP315	Highly contaminated	0.5	5248.74
BP331	Highly contaminated	0.3	2580.95
BP101	Less contaminated	10.1	2.64
BP155	Less contaminated	15.1	88.70
BP186	Less contaminated	59.5	26.31

Aims

By analyzing the data we want to address the following questions:

- Which species/OTUs are present in the samples? (taxonomic profiling using the 16S gene)
- How does the species composition and diversity compare between samples?
- Which protein functions can be found in the different samples? (shotgun metagenomics)
- Which functions differ between polluted and clean environment?

General

Here, we will run the entire analysis in R. Typically, for larger datasets command line program packages such as QIIME (http://qiime.org/) and mothur (https://www.mothur.org/) are used. The logic and workflow is comparable.

Part 1 - 16S sequence data diversity and taxonomic composition

Sequence variant clustering

We will use the R package dada2 (https://benjjneb.github.io/dada2/index.html) to denoise the 16S sequences. Denoising means to eliminate the sequence variation which is due to sequencing errors while trying to keep natural sequence variation representing for instance inter-species diversity.

You need to download the sequence data and unpack it in the working directory.

To start, we load the package `dada2` and create some auxilary vectors

Hide

```
library(dada2)
sample_names <- c("BP101","BP155","BP186","BP278","BP315","BP331")
sample_files <- paste0(sample_names, "_16s.fq.gz")
```

The read sequences are already quality filtered and PCR primers, adapters and barcodes have been removed. Still, we will have a look at the quality values of the sequenced positions because these are used in the denoising procedure.

Hide

```
plotQualityProfile(sample_files)
```

Exercise 1: Why do you think does the quality of the Illumina reads drop towards the end (consult the help of the plotting function to understand the colors of the lines)?

WRITE YOUR ANSWER HERE

We start by collapsing identical sequences for each sample to save computation time and memory.

Hide

```
reads_derep <- derepFastq(sample_files, verbose=TRUE)
```

Exercise 2: How many sequences are in each sample and how redundant are these? Extract the number of unique sequences as a vector and report the them as fractions of the original read number

Hide

```
# WRITE YOUR CODE HERE
```

Next, we will cluster (denoise) the 16S sequences so that each cluster represents a species variant or Operational Taxonomic Unit (OTU). Dada2 incorporates a sequence error model to distinguish noise from real sequence variation. The first step is to learn the model parameters from the read dataset and the second is the actual clustering. Finally, we construct a sequence count table after removing chimeras which are artifical hybrid sequences created during DNA amplification using PCR.

Hide

```
error_reads <- learnErrors(reads_derep, multithread=TRUE)
cluster_reads <- dada(reads_derep, err=error_reads, multithread=TRUE)
seqtab_reads <- removeBimeraDenovo(makeSequenceTable(cluster_reads), multithread=TRUE, verbose=TRUE)
```

Exercise 3: Inspect the content of `seqtab_reads` and extract the number of inferred sequence variants per sample as a vector. Calculate the fraction of variant sequences both among unique sequences and the original sample size for each sample.

Hide

```
# WRITE YOUR CODE HERE
```

Taxonomic annotation

Basically, we have reduced the heterogenous sequences and their qualities to a smaller set of 16S sequence variants and corresponding counts in `seqtab_reads`. In order to annotate assign these variants to taxa, we match them against a set of reference sequences with annotation in the file 'greengenes_13.8_97.fna.gz'.

Hide

```
taxa_reads <- assignTaxonomy(seqtab_reads, "greengenes_13.8_97.fna.gz", tryRC=TRUE)
```

Plotting

Now we have everything read to start our analysis. We will use the phyloseq R package (https://joey711.github.io/phyloseq/) for this.

Hide

```
library(phyloseq)
library(ggplot2)
```

We need to load the metadata table which groups the samples into low and high contamination samples. We will also make sure that the samples are called the same in the otu and taxa tables. Finally, we convert bundle the information in a 'phyloseq' object.

Hide

```
metadata <- read.delim("metadata.tsv")
rownames(metadata) <- metadata$ID
rownames(seqtab_reads) <- metadata$ID
ps_reads <- phyloseq(otu_table(seqtab_reads, taxa_are_rows=FALSE), sample_data(metadata), tax_table(taxa_reads))
```

First, we want to show the sample diversity for each of the samples. This so-called alpha-diversity (https://en.wikipedia.org/wiki/Alpha_diversity) tells us how many distinct OTUs inhabit an ecosystem.

Hide

```
plot_richness(ps_reads, x="SampleType", measures=c("Shannon", "Simpson"), col="ID")
```

Exercise 4: Explain what you see and what Shannon and Simpson stand for.

WRITE YOUR ANSWER HERE

Let's have a look at the taxon distributions. We create a bar plot using the phyloseq-provided function. We normalize the OTUs counts in each table and select the 100 most abundant entries for visualization. Then we decide to plot one bar for each class and color by family.

Hide

```
ps_plot <- transform_sample_counts(ps_reads, function(x) x/sum(x))
ps_plot <- prune_taxa(names(sort(taxa_sums(ps_reads), decreasing=TRUE)[1:200]), ps_plot)
plot_bar(ps_plot, x="Class", facet_grid=SampleType~., fill="Family") + geom_bar(aes(color=Family, fill=Family), s
tat="identity", position="stack") + theme(legend.position="none")
```

Exercise 5: Can you spot a difference? Formulate a hypothesis and try to validate it by plotting the same barplot but once per sample. Elaborate on whether you can confirm your hypothesis.

WRITE YOUR ANSWER HERE

Hide

```
# WRITE YOUR CODE HERE
```

WRITE YOUR ANSWER HERE

Normalization of count data is a critical point in the analyis of changes of taxa and OTUs. To avoid inappropriate conclusions, it is better to use statistical models designed for count data so that normalization and subsampling is not needed. Here, we try do identify the OTUs that vary significantly in frequency between the two groups. We will use the the DESeq2 R package (https://doi.org/10.18129/B9.bioc.DESeq2) for this task, which is also often used in the analysis of RNA and gene expression data. We run a Wald test for the count data in both groups which calculates the log-fold-change and significance (p-values), among other measures.

Hide

```
library(DESeq2)
test_reads <- DESeq(phyloseq_to_deseq2(ps_reads, ~SampleType), test="Wald", fitType="parametric", parallel=TRUE)
head(results(test_reads))
```

We retrieve the results of the test and create an object for plotting. We add all the information needed to layout the plot. The plotting code looks complicated but is needed to format and sort the axes and to color the plot.

Hide

```
results_reads <- results(test_reads, contrast=list("SampleType_less_contaminated_vs_highly_contaminated"))
sigtab_reads <- cbind(as(results_reads, "data.frame"), as(tax_table(ps_reads)[rownames(results_reads),], "matrix"
))
class_ordering <- sort(tapply(sigtab_reads$log2FoldChange, sigtab_reads$Class, function(x) max(x)), decreasing=TR
UE)  # sort class by max fold change
sigtab_reads$Class <- factor(as.character(sigtab_reads$Class), levels=names(class_ordering))  # order the x-axis
genus_ordering <- sort(tapply(sigtab_reads$log2FoldChange, sigtab_reads$Genus, function(x) max(x)), decreasing=TR
UE)  # sort genus by max fold change
sigtab_reads$Genus <- factor(as.character(sigtab_reads$Genus), levels=names(genus_ordering))  # order ???
size <- apply(otu_table(ps_reads), 2, mean)
sigtab_reads$Size <- 50*(size/sum(size))^0.4
```

We subset the OTUs by selecting only those which have an adjusted p-value less than 1% and plot the results of the test. Note that by default, 'DESeq2` performs a conservative outlier filtering to reduce false positives.

Hide

```
sigtab_reads_plot <- sigtab_reads[which(sigtab_reads$padj < 0.01),]
ggplot(sigtab_reads_plot, aes(x=Class, y=log2FoldChange, color=Class)) + geom_point(size=sigtab_reads_plot$Size,
alpha=0.5) + theme(axis.text.x=element_text(angle=-90, hjust=0, vjust=0.5), legend.position="none")
```

Exercise 6: What conclusions do you draw from this plot and the previous abundance and diversity plots. Do they agree? (Note: the log foldchanges is positive if the OTU is abundant in the less contaminated samples and negative, if it is abundant in the contaminated samples). Characterize the OTUs which are significantly enrichted in the contaminated samples, for instance what are their lowest taxonomic annotations?

Part 2 - Shotgun metagenomics

In the second part of this exercise sheet, the data we analyze are whole genome shotgun sequences instead of partial 16S gene sequences. This means, that they derive from any part of any genome in the corresponding environments. The sequenced reads are paired end Illumina sequences. We will perform the following data analysis:

1. Visual analysis of an alignment of the sequences to an assembled metagenome
2. Annotation with taxon and gene GFF files
3. Calculation of gene abundance values
4. Gene differential abundance analysis

In the last step, we will use the same methods as for the 16S differential OTU count analysis but at the gene level to find genes which are significantly enriched or depleted under the two different contamination levels.

Loading data files

We start working with a small subset of the data. The reads were assembled into contigs which can be found in the FASTA file `contigs.fna`. The file `contigs.tsv` contains the name and the length of each contig. For each of the six samples, the reads where aligned to the contigs using bowtie2 and the alignments can be found the the corresponding files `BPXXX.bam`. Finally, we compared the contigs to reference collections to derive gene annotation using TIGRFAM and taxonomic annotation using the best BLAST hit. These annotations can be found in the files `contigs_genes.gff3` and `contigs_taxa.gff3`.

Coverage annotation plot

We need to import some special packages. If they are not installed, you need to install them using the RStudio GUI or 'install.packages()`.

Hide

```
library(GenomicRanges)
library(GenomicAlignments)
library(rtracklayer)
library(ggbio)
```

We start by reading the data files into R.

Hide

```
contigs.meta <- read.delim("contigs.tsv", header=FALSE)
contigs.seqinfo <- Seqinfo(seqnames=as.vector(contigs.meta[,1]), seqlengths=contigs.meta[,2], genome="contigs")
contigs.ranges <- GRanges(seqnames=seqnames(contigs.seqinfo), strand="*", ranges = IRanges(start=1, width=seqleng
ths(contigs.seqinfo)), seqinfo=contigs.seqinfo)
genes.ranges <- GRanges(readGFF("contigs_genes.gff3", version=3), seqinfo=contigs.seqinfo)
taxa.ranges <- GRanges(readGFF("contigs_taxa.gff3", version=3), seqinfo=contigs.seqinfo)
sample1 <- coverage(readGAlignments("BP101.bam"))
sample2 <- coverage(readGAlignments("BP155.bam"))
sample3 <- coverage(readGAlignments("BP186.bam"))
sample4 <- coverage(readGAlignments("BP278.bam"))
sample5 <- coverage(readGAlignments("BP315.bam"))
sample6 <- coverage(readGAlignments("BP331.bam"))
```

And we create a circular plot with different tracks using the package `ggbio`. The tracks represent the following different sequence features (from inner to outer):

1. The positional read coverage over in each of the contigs in each of the six samples
2. The taxonomic annotation of subsequences based on the closest hit in a reference sequence collection
3. The gene annotation using TIGRFAM gene families (https://en.wikipedia.org/wiki/TIGRFAMs).

Hide

```
colors <- c("#89C5DA", "#DA5724", "#74D944", "#CE50CA", "#3F4921", "#C0717C", "#CBD588", "#5F7FC7",  "#673770", "
#D3D93E", "#38333E", "#508578", "#D7C1B1", "#689030", "#AD6F3B", "#CD9BCD", "#D14285", "#6DDE88", "#652926", "#7F
DCC0", "#C84248", "#8569D5", "#5E738F", "#D1A33D", "#8A7C64", "#599861")  # custom color palette

ggbio() +
circle(GRanges(sample1), aes(y=score), geom="bar", fill="blue", col="blue", grid=TRUE, grid.background="gray90",
radius=10) +
circle(GRanges(sample2), aes(y=score), geom="bar", fill="blue", col="blue", grid=TRUE, grid.background="gray90",
radius=15) +
circle(GRanges(sample3), aes(y=score), geom="bar", fill="blue", col="blue", grid=TRUE, grid.background="gray90",
radius=20) +
circle(GRanges(sample4), aes(y=score), geom="bar", fill="red", col="red", grid=TRUE, grid.background="gray90", ra
dius=25) +
circle(GRanges(sample5), aes(y=score), geom="bar", fill="red", col="red", grid=TRUE, grid.background="gray90", ra
dius=30) +
circle(GRanges(sample6), aes(y=score), geom="bar", fill="red", col="red", grid=TRUE, grid.background="gray90", ra
dius=35) +
circle(contigs.ranges, fill="gray70", geom="rect", col=NA, radius=40) +
circle(taxa.ranges, aes(fill=rtaxname), geom="rect", col=NA, radius=40) +
circle(contigs.ranges, fill="gray70", geom="rect", col=NA, radius=45) +
circle(genes.ranges, aes(fill=ID), geom="rect", col=NA, radius=45) +
circle(contigs.ranges, geom="scale", scale.unit=100, col=NA, trackWidth=2, radius=50) +
scale_fill_manual(values=colors, name=NULL) + guides(fill=guide_legend(ncol=1)) + theme(legend.position="right")
```

Exercise 7: Try to understand the plot. How long are the longest and the shortest contigs in the set? Based on what you see in the plot, what can you say about the possible origin of the longest contig?

WRITE YOUR ANSWER HERE

Coverage calculation

We extract the gene positions and names and put them into a dataframe. We also define a function to extract the read coverage of a contig as a plain vector.

Hide

```
genes.positions <- cbind(as.vector(seqnames(genes.ranges)), as.data.frame(ranges(genes.ranges)))  # extract data
colnames(genes.positions)[1] <- "id"
vectorize.coverage <- function(x) as.vector(x[[1]])  # turns contig into coverage vector
```

Exercise 8: Calculate the mean read coverage of the first (longest) six contigs in the sixth sample using the `vectorize.coverage` function (compare with the circular plot to validate).

Hide

```
# WRITE YOUR CODE HERE
```

Now, here is a function to calculate the mean coverage of a gene region over all samples.

Hide

```
samples <- list(sample1, sample2, sample3, sample4, sample5, sample6)  # list of samples
gene.abundances <- function(name, start, stop) sapply(samples, function(x) mean(vectorize.coverage(x[name])[start
:stop]))  # gene abundace in each sample
```

Exercise 9: Write a function to calculate the mean read coverage of every gene in every sample using the dataframe `genes.positions` and the function `gene.abundances` . Each row should correspond to a gene and each column should correspond to a sample. Remember, that you can use `apply()` to achieve this. Don't forget to output the table below.

Hide

```
# WRITE YOUR CODE HERE
```

Differential gene abundance analysis

Here, we will start with an abundance matrix which is much larger than we have created but the steps are the same.

To quickly explore the data, we do a PCA and plot the variance in the principal components.

Hide

```
genes.full <- read.delim('gene_abundance.tsv', row.names=1)
genes.full.pca <- prcomp(t(genes.full), center=TRUE, scale=TRUE)
summary(genes.full.pca)
plot(genes.full.pca)
```

Exercise 10: What proportion of the variability in the data can be explained by using only the first two principal components (PC1 and PC2)?

WRITE YOUR ANSWER HERE

Exercise 11: Plot the samples as points in 2d using only the first two principal components. Note that you can find the coordinates in `genes.full.pca$x` . Can we see a general seperation in the data between highly contaminated samples and less contaminated samples?

WRITE YOUR ANSWER HERE

Exercise 12: Finally, perform a statistical analysis to determine differentially abundant genes using DESeq2 as we did for the taxa in part 1. Report the three most significant genes and their adjusted p-values. Look up their meaning in the file `TIGRFAM_info.tsv` (either manually or using R) or online (http://www.jcvi.org/cgi-bin/tigrfams/Listing.cgi) and report.

Hide

```
# first we convert the data into a DESeq2 object
genes.full.deseq <- DESeqDataSetFromMatrix(countData=genes.full, colData=metadata, design=~SampleType)

# WRITE YOUR CODE HERE
```

WRITE YOUR ANSWER HERE

Exercise 13: List possible reasons for zero counts in the gene abundance matrix.