WY Lake Sediment Prokaryotic Community Data Cleaning

Jordan Von Eggers

2024-01-19

This markdown file contains the code used by Genome Technologies Laboratory at UW to convert raw fastq files to OTU/ESV tables. The rest of this code is assigning taxonomy, and filtering/cleaning the taxonomy table and ESV table to be made into a final Phyloseq object for further data analysis in a separate markdown file. If you have any questions, please contact me at jordanvoneggers@gmail.com.

Load packages

```
Sys.Date()
```

```
## [1] "2024-06-26"
```

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
require(vegan)
```

```
## Loading required package: vegan
## Loading required package: permute
## Loading required package: lattice
## This is vegan 2.6-6.1
```

```
require(phyloseq)
```

```
## Loading required package: phyloseq
```

```
require(decontam)
```

```
## Loading required package: decontam
```

```
require(Biostrings)
```

```
## Loading required package: Biostrings
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
##
## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union
##
## 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, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, table,
##     tapply, union, unique, unsplit, which.max, which.min
##
## Loading required package: S4Vectors
## Loading required package: stats4
##
## Attaching package: 'S4Vectors'
##
## The following objects are masked from 'package:lubridate':
##
##     second, second<-
##
## 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:utils':
##
##     findMatches
```

```
##
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
##
## Loading required package: IRanges
##
## Attaching package: 'IRanges'
##
## The following object is masked from 'package:phyloseq':
##
##     distance
##
## The following object is masked from 'package:lubridate':
##
##     %within%
##
## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice
##
## The following object is masked from 'package:purrr':
##
##     reduce
##
## Loading required package: XVector
##
## Attaching package: 'XVector'
##
## The following object is masked from 'package:purrr':
##
##     compact
##
## Loading required package: GenomeInfoDb
##
## Attaching package: 'Biostrings'
##
## The following object is masked from 'package:base':
##
##     strsplit
```

```
sessionInfo()
```

```
## R version 4.4.0 (2024-04-24)
## Platform: x86_64-apple-darwin20
## Running under: macOS Monterey 12.7.5
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-x86_64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
```

```
## time zone: America/Denver
## tzcode source: internal
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] Biostrings_2.72.1   GenomeInfoDb_1.40.1 XVector_0.44.0
##  [4] IRanges_2.38.0      S4Vectors_0.42.0    BiocGenerics_0.50.0
##  [7] decontam_1.24.0     phyloseq_1.48.0     vegan_2.6-6.1
## [10] lattice_0.22-6      permute_0.9-7       lubridate_1.9.3
## [13] forcats_1.0.0       stringr_1.5.1       dplyr_1.1.4
## [16] purrr_1.0.2         readr_2.1.5         tidyr_1.3.1
## [19] tibble_3.2.1        ggplot2_3.5.1       tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] ade4_1.7-22             tidyselect_1.2.1        fastmap_1.2.0
##  [4] digest_0.6.35           timechange_0.3.0        lifecycle_1.0.4
##  [7] cluster_2.1.6           survival_3.7-0          magrittr_2.0.3
## [10] compiler_4.4.0          rlang_1.1.4             tools_4.4.0
## [13] igraph_2.0.3            utf8_1.2.4              yaml_2.3.8
## [16] data.table_1.15.4       knitr_1.47              plyr_1.8.9
## [19] withr_3.0.0             grid_4.4.0              fansi_1.0.6
## [22] multtest_2.60.0         biomformat_1.32.0       colorspace_2.1-0
## [25] Rhdf5lib_1.26.0         scales_1.3.0            iterators_1.0.14
## [28] MASS_7.3-60.2           cli_3.6.2               rmarkdown_2.27
## [31] crayon_1.5.2            generics_0.1.3          rstudioapi_0.16.0
## [34] httr_1.4.7              reshape2_1.4.4          tzdb_0.4.0
## [37] ape_5.8                 rhdf5_2.48.0            zlibbioc_1.50.0
## [40] splines_4.4.0           parallel_4.4.0          vctrs_0.6.5
## [43] Matrix_1.7-0            jsonlite_1.8.8          hms_1.1.3
## [46] foreach_1.5.2           glue_1.7.0              codetools_0.2-20
## [49] stringi_1.8.4           gtable_0.3.5            UCSC.utils_1.0.0
## [52] munsell_0.5.1           pillar_1.9.0            htmltools_0.5.8.1
## [55] rhdf5filters_1.16.0     GenomeInfoDbData_1.2.12 R6_2.5.1
## [58] evaluate_0.23           Biobase_2.64.0          Rcpp_1.0.12
## [61] nlme_3.1-165            mgcv_1.9-1              xfun_0.44
## [64] pkgconfig_2.0.3
```

Part I: Extract files and assign taxonomy

2. Assigned taxonomy

Using the maintained datasets for SILVA training dataset v 138.1 from DADA2 hosted on Zenodo

McLaren, Michael R., & Callahan, Benjamin J. (2021). Silva 138.1 prokaryotic SSU taxonomic training data formatted for DADA2 [Data set]. Zenodo. https://doi.org/10.5281/zenodo.4587955

Merge the ZOTU/OTU id with the assigned taxonomy, based on the sequence in column one of the tax_table

```
# read in fasta file
fasta <- readDNAStringSet("../1_RawSequenceProcessing/output/ZOTU_no_chimeras_denovo.fa")
```

```
# read in assigned taxonomy
tax_table<-read.csv("../1_RawSequenceProcessing/output/ZOTU_taxonomy_80_DADA2.csv", header=T)
colnames(tax_table)[1]<-"seq"
ESVseq<-data.frame(seq=as.character(fasta), names=names(fasta))
ESVseq$ESV<-str_split(ESVseq$names,pattern = ";", 3, simplify = T)[,1]
ESVseq$names<-NULL

tax_table<-merge(tax_table,ESVseq, by="seq")
tax_table$seq<-NULL
rownames(tax_table)<-tax_table$ESV
tax_table$ESV<-NULL
write.csv(tax_table,"Taxonomy/ZOTU_Silva138_assignedTaxonomy_toSpecies.csv")
rm(ESVseq);rm(fasta)
tax_tab<-tax_table;rm(tax_table)
```

Put these into an "AssignTaxonomy" folder and then the final one in the "FinalTaxFile"

Part 2: Data Cleaning

1. Remove unassigned and eukaryotic ESVs

a. Read in taxonomy table

```
tax_tab<-read.csv("Taxonomy/ZOTU_Silva138_assignedTaxonomy_toSpecies.csv", header=T,row.names=1)
table(tax_tab$Kingdom) # showing NO Eukaryotes
```

```
##
##  Archaea Bacteria
##    15195   207060
```

```
unique(tax_tab$Kingdom) # when the cell is empty there is an NA
```

```
## [1] NA         "Bacteria" "Archaea"
```

b. Remove unassigned kingdom, and taxa assigned to chloroplast, and mitochondria

```
# count unassigned taxa at the kingdom level (column 1)
table(is.na(tax_tab[,1]))
```

```
##
##  FALSE   TRUE
## 222255    909
```

```
tax_tab<-tax_tab[-which(is.na(tax_tab$Kingdom)),] # remove unassigned kingdom

table(tax_tab[,]=="Chloroplast")
```

```
##
##  FALSE   TRUE
## 896631   1068
```

```
table(tax_tab$Kingdom=="Chloroplast")
```

```
##
##  FALSE
## 222255
```

```
table(tax_tab$Phylum=="Chloroplast")
```

```
##
##  FALSE
## 193708
```

```
table(tax_tab$Class=="Chloroplast")
```

```
##
##  FALSE
## 180702
```

```
table(tax_tab$Order=="Chloroplast") # only here
```

```
##
##  FALSE   TRUE
## 152359   1068
```

```
table(tax_tab$Family=="Chloroplast")
```

```
##
##  FALSE
## 104343
```

```
table(tax_tab$Genus=="Chloroplast")
```

```
##
## FALSE
## 42419
```

```
table(tax_tab$Species=="Chloroplast")
```

```
##
## FALSE
##   845
```

```
tax_tab<-tax_tab[-which(tax_tab$Order=="Chloroplast"),] # remove chloroplasts

table(tax_tab[,]=="Mitochondria")
```

```
##
##  FALSE   TRUE
## 891901   1526
```

```
table(tax_tab$Kingdom=="Mitochondria")
```

```
##
##  FALSE
## 221187
```

```
table(tax_tab$Phylum=="Mitochondria")
```

```
##
##  FALSE
## 192640
```

```
table(tax_tab$Class=="Mitochondria")
```

```
##
##  FALSE
## 179634
```

```
table(tax_tab$Order=="Mitochondria")
```

```
##
##  FALSE
## 152359
```

```
table(tax_tab$Family=="Mitochondria") #only here
```

```
##
##  FALSE   TRUE
## 102817   1526
```

```
table(tax_tab$Genus=="Mitochondria")
```

```
##
## FALSE
## 42419
```

```
table(tax_tab$Species=="Mitochondria")
```

```
##
## FALSE
##   845
```

```
tax_tab<-tax_tab[-which(tax_tab$Family=="Mitochondria"),]
```

@count_track (taxa)

```
nrow(tax_tab)
```

```
## [1] 219661
```

2. Read in ESV table

```
esv_table<-read.delim("../1_RawSequenceProcessing/output/ZOTU_table_exact", header = TRUE, stringsAsFac
#change the ESV name to be a row name
rownames(esv_table)<-esv_table$X.OTU.ID
esv_table$X.OTU.ID<-NULL
```

a. Remove samples not intended for this study

1. remove all water samples that are not included in this study
2. remove samples that should not have been included in this study
3. remove any ESVs without reads

```
ESV_names<-names(esv_table)

keep<-read.csv("wanted_samples.csv",header=F) %>% pull(.,var=V1)

positions_keep<-NULL
for(i in 1:length(keep)){
   positions_keep<-c(positions_keep, grep(keep[i], ESV_names))
}
# this matched sometimes to multiple columns so make sure to unique this.
positions_keep<-unique(positions_keep)

# look at which samples are removed
remove_df<-esv_table[,-positions_keep]
(str_split(names(remove_df),pattern = ".16S.", 2, simplify = T)[,1])
```

```
##   [1] "Calder.33_1_0_DNA"     "Calder.33_1_0_DNA"     "Calder.33_1_10_DNA"
##   [4] "Calder.33_1_10_DNA"    "Calder.33_1_6_DNA"     "Calder.33_1_6_DNA"
##   [7] "Calder.33_1_8_DNA"     "Calder.33_1_8_DNA"     "Calder.34_1_10_DNA"
##  [10] "Calder.34_1_10_DNA"    "Calder.34_1_12_DNA"    "Calder.34_1_12_DNA"
##  [13] "Calder.34_1_22_DNA"    "Calder.34_1_22_DNA"    "Calder.34_1_24_DNA"
##  [16] "Calder.34_1_24_DNA"    "Calder.34_1_26_DNA"    "Calder.34_1_26_DNA"
##  [19] "Calder.34_1_28_DNA"    "Calder.34_1_28_DNA"    "Calder.34_1_6_DNA"
##  [22] "Calder.34_1_6_DNA"     "Calder.34_1_8_DNA"     "Calder.34_1_8_DNA"
##  [25] "Calder.35_1_10_DNA"    "Calder.35_1_10_DNA"    "Calder.35_1_12_DNA"
##  [28] "Calder.35_1_12_DNA"    "Calder.35_1_14_DNA"    "Calder.35_1_14_DNA"
##  [31] "Calder.35_1_6_DNA"     "Calder.35_1_6_DNA"     "Calder.35_1_8_DNA"
```

```
##  [34] "Calder.35_1_8_DNA"     "Calder.35_2_6_DNA"     "Calder.35_2_6_DNA"
##  [37] "Calder.36_1_10_DNA"    "Calder.36_1_10_DNA"    "Calder.36_1_12_DNA"
##  [40] "Calder.36_1_12_DNA"    "Calder.36_1_14_DNA"    "Calder.36_1_14_DNA"
##  [43] "Calder.36_1_8_DNA"     "Calder.36_1_8_DNA"     "Calder.37_1_10_DNA"
##  [46] "Calder.37_1_10_DNA"    "Calder.37_1_2_DNA"     "Calder.37_1_2_DNA"
##  [49] "Calder.37_1_6_DNA"     "Calder.37_1_6_DNA"     "Calder.37_1_8_DNA"
##  [52] "Calder.37_1_8_DNA"     "Calder.39_1_10_DNA"    "Calder.39_1_10_DNA"
##  [55] "Calder.39_1_12_DNA"    "Calder.39_1_12_DNA"    "Calder.39_1_14_DNA"
##  [58] "Calder.39_1_14_DNA"    "Calder.39_1_18_DNA"    "Calder.39_1_18_DNA"
##  [61] "Calder.39_1_22_DNA"    "Calder.39_1_22_DNA"    "Calder.39_1_24_DNA"
##  [64] "Calder.39_1_24_DNA"    "Calder.39_1_26_DNA"    "Calder.39_1_26_DNA"
##  [67] "Calder.39_1_28_DNA"    "Calder.39_1_28_DNA"    "Calder.39_1_30_DNA"
##  [70] "Calder.39_1_30_DNA"    "Calder.39_1_6_DNA"     "Calder.39_1_6_DNA"
##  [73] "Calder.39_1_8_DNA"     "Calder.39_1_8_DNA"     "Calder.40_1_10_DNA"
##  [76] "Calder.40_1_10_DNA"    "Calder.40_1_12_DNA"    "Calder.40_1_12_DNA"
##  [79] "Calder.40_1_14_DNA"    "Calder.40_1_14_DNA"    "Calder.40_1_16_DNA"
##  [82] "Calder.40_1_16_DNA"    "Calder.40_1_18_DNA"    "Calder.40_1_18_DNA"
##  [85] "Calder.40_1_22_DNA"    "Calder.40_1_22_DNA"    "Calder.40_1_24_DNA"
##  [88] "Calder.40_1_24_DNA"    "Calder.40_1_26_DNA"    "Calder.40_1_26_DNA"
##  [91] "Calder.40_1_6_DNA"     "Calder.40_1_6_DNA"     "Calder.40_1_8_DNA"
##  [94] "Calder.40_1_8_DNA"     "Calder.41_1_10_DNA"    "Calder.41_1_10_DNA"
##  [97] "Calder.41_1_12_DNA"    "Calder.41_1_12_DNA"    "Calder.41_1_14_DNA"
## [100] "Calder.41_1_14_DNA"    "Calder.41_1_22_DNA"    "Calder.41_1_22_DNA"
## [103] "Calder.41_1_24_DNA"    "Calder.41_1_24_DNA"    "Calder.41_1_6_DNA"
## [106] "Calder.41_1_6_DNA"     "Calder.41_1_8_DNA"     "Calder.41_1_8_DNA"
## [109] "Calder.42_1_10_DNA"    "Calder.42_1_10_DNA"    "Calder.42_1_12_DNA"
## [112] "Calder.42_1_12_DNA"    "Calder.42_1_22_DNA"    "Calder.42_1_22_DNA"
## [115] "Calder.42_1_24_DNA"    "Calder.42_1_24_DNA"    "Calder.42_1_26_DNA"
## [118] "Calder.42_1_26_DNA"    "Calder.42_1_28_DNA"    "Calder.42_1_28_DNA"
## [121] "Calder.42_1_6_DNA"     "Calder.42_1_6_DNA"     "Calder.42_1_8_DNA"
## [124] "Calder.42_1_8_DNA"     "Calder.43_1_10_DNA"    "Calder.43_1_10_DNA"
## [127] "Calder.43_1_8_DNA"     "Calder.43_1_8_DNA"     "Calder.44_1_10_DNA"
## [130] "Calder.44_1_10_DNA"    "Calder.44_1_12_DNA"    "Calder.44_1_12_DNA"
## [133] "Calder.44_1_14_DNA"    "Calder.44_1_14_DNA"    "Calder.44_1_22_DNA"
## [136] "Calder.44_1_22_DNA"    "Calder.44_1_24_DNA"    "Calder.44_1_24_DNA"
## [139] "Calder.44_1_26_DNA"    "Calder.44_1_26_DNA"    "Calder.44_1_28_DNA"
## [142] "Calder.44_1_28_DNA"    "Calder.44_1_30_DNA"    "Calder.44_1_30_DNA"
## [145] "Calder.44_1_32_DNA"    "Calder.44_1_32_DNA"    "Calder.44_1_34_DNA"
## [148] "Calder.44_1_34_DNA"    "Calder.44_1_8_DNA"     "Calder.44_1_8_DNA"
## [151] "Calder.45_1_10_DNA"    "Calder.45_1_10_DNA"    "Calder.45_1_12_DNA"
## [154] "Calder.45_1_12_DNA"    "Calder.45_1_14_DNA"    "Calder.45_1_14_DNA"
## [157] "Calder.45_1_22_DNA"    "Calder.45_1_22_DNA"    "Calder.45_1_24_DNA"
## [160] "Calder.45_1_24_DNA"    "Calder.45_1_26_DNA"    "Calder.45_1_26_DNA"
## [163] "Calder.45_1_28_DNA"    "Calder.45_1_28_DNA"    "Calder.45_1_30_DNA"
## [166] "Calder.45_1_30_DNA"    "Calder.45_1_32_DNA"    "Calder.45_1_32_DNA"
## [169] "Calder.45_1_34_DNA"    "Calder.45_1_34_DNA"    "Calder.45_1_36_DNA"
## [172] "Calder.45_1_36_DNA"    "Calder.45_1_6_DNA"     "Calder.45_1_6_DNA"
## [175] "Calder.45_1_8_DNA"     "Calder.45_1_8_DNA"     "Calder.46_1_10_DNA"
## [178] "Calder.46_1_10_DNA"    "Calder.46_1_12_DNA"    "Calder.46_1_12_DNA"
## [181] "Calder.46_1_14_DNA"    "Calder.46_1_14_DNA"    "Calder.46_1_6_DNA"
## [184] "Calder.46_1_6_DNA"     "Calder.46_1_8_DNA"     "Calder.46_1_8_DNA"
## [187] "Calder.47_1_0_DNA"     "Calder.47_1_0_DNA"     "Calder.47_1_10_DNA"
## [190] "Calder.47_1_10_DNA"    "Calder.47_1_12_DNA"    "Calder.47_1_12_DNA"
## [193] "Calder.47_1_14_DNA"    "Calder.47_1_14_DNA"    "Calder.47_1_22_DNA"
```

```
## [196] "Calder.47_1_22_DNA"    "Calder.47_1_24_DNA"    "Calder.47_1_24_DNA"
## [199] "Calder.47_1_26_DNA"    "Calder.47_1_26_DNA"    "Calder.47_1_28_DNA"
## [202] "Calder.47_1_28_DNA"    "Calder.47_1_6_DNA"     "Calder.47_1_6_DNA"
## [205] "Calder.47_1_8_DNA"     "Calder.47_1_8_DNA"     "Calder.48_1_10_DNA"
## [208] "Calder.48_1_10_DNA"    "Calder.48_1_12_DNA"    "Calder.48_1_12_DNA"
## [211] "Calder.48_1_14_DNA"    "Calder.48_1_14_DNA"    "Calder.48_1_6_DNA"
## [214] "Calder.48_1_6_DNA"     "Calder.48_1_8_DNA"     "Calder.48_1_8_DNA"
## [217] "Calder.49_1_10_DNA"    "Calder.49_1_10_DNA"    "Calder.49_1_12_DNA"
## [220] "Calder.49_1_12_DNA"    "Calder.49_1_14_DNA"    "Calder.49_1_14_DNA"
## [223] "Calder.49_1_22_DNA"    "Calder.49_1_22_DNA"    "Calder.49_1_24_DNA"
## [226] "Calder.49_1_24_DNA"    "Calder.49_1_26_DNA"    "Calder.49_1_26_DNA"
## [229] "Calder.49_1_28_DNA"    "Calder.49_1_28_DNA"    "Calder.49_1_30_DNA"
## [232] "Calder.49_1_30_DNA"    "Calder.49_1_6_DNA"     "Calder.49_1_6_DNA"
## [235] "Calder.49_1_8_DNA"     "Calder.49_1_8_DNA"     "Calder.50_1_10_DNA"
## [238] "Calder.50_1_10_DNA"    "Calder.50_1_12_DNA"    "Calder.50_1_12_DNA"
## [241] "Calder.50_1_2_DNA"     "Calder.50_1_2_DNA"     "Calder.50_1_6_DNA"
## [244] "Calder.50_1_6_DNA"     "Calder.51_1_10_DNA"    "Calder.51_1_10_DNA"
## [247] "Calder.51_1_12_DNA"    "Calder.51_1_12_DNA"    "Calder.51_1_6_DNA"
## [250] "Calder.51_1_6_DNA"     "Calder.51_1_8_DNA"     "Calder.51_1_8_DNA"
## [253] "Calder.52_1_0_DNA"     "Calder.52_1_0_DNA"     "Calder.52_1_10_DNA"
## [256] "Calder.52_1_10_DNA"    "Calder.52_1_14_DNA"    "Calder.52_1_14_DNA"
## [259] "Calder.52_1_6_DNA"     "Calder.52_1_6_DNA"     "Calder.52_1_8_DNA"
## [262] "Calder.52_1_8_DNA"     "Calder.BL0106L1"       "Calder.BL0106L1"
## [265] "Calder.BL0106L2"       "Calder.BL0106L2"       "Calder.BL0108L"
## [268] "Calder.BL0108L"        "Calder.BL0110L"        "Calder.BL0110L"
## [271] "Calder.BL0112L"        "Calder.BL0112L"        "Calder.BL0114L"
## [274] "Calder.BL0114L"        "Calder.BL0118L"        "Calder.BL0118L"
## [277] "Calder.BL0122L"        "Calder.BL0122L"        "Calder.BL0124L"
## [280] "Calder.BL0124L"        "Calder.BL0126L"        "Calder.BL0126L"
## [283] "Calder.BL0128L"        "Calder.BL0128L"        "Calder.BL0130L"
## [286] "Calder.BL0130L"        "Calder.BL0132L"        "Calder.BL0132L"
## [289] "Calder.BL0200L"        "Calder.BL0200L"        "Calder.BL0202L"
## [292] "Calder.BL0202L"        "Calder.BL0204L"        "Calder.BL0204L"
## [295] "Calder.BL0206L"        "Calder.BL0206L"        "Calder.BL0208L"
## [298] "Calder.BL0208L"        "Calder.BL0210L"        "Calder.BL0210L"
## [301] "Calder.BL0212L"        "Calder.BL0212L"        "Calder.BL0214L"
## [304] "Calder.BL0214L"        "Calder.BL0220L"        "Calder.BL0220L"
## [307] "Calder.BL0222L"        "Calder.BL0222L"        "Calder.BL0224L"
## [310] "Calder.BL0224L"        "Calder.BL0226L"        "Calder.BL0226L"
## [313] "Calder.BL0228L"        "Calder.BL0228L"        "Calder.BL0230L"
## [316] "Calder.BL0230L"        "Calder.BL0232L"        "Calder.BL0232L"
## [319] "Calder.BN0100"         "Calder.BN0100"         "Calder.CL0104L"
## [322] "Calder.CL0104L"        "Calder.CL0106L"        "Calder.CL0106L"
## [325] "Calder.CL0108L"        "Calder.CL0108L"        "Calder.CL0110L"
## [328] "Calder.CL0110L"        "Calder.CL0112L"        "Calder.CL0112L"
## [331] "Calder.CL0114L"        "Calder.CL0114L"        "Calder.CL0122L"
## [334] "Calder.CL0122L"        "Calder.CL0124L"        "Calder.CL0124L"
## [337] "Calder.CL0126L"        "Calder.CL0126L"        "Calder.CL0128L"
## [340] "Calder.CL0128L"        "Calder.CL0130L"        "Calder.CL0130L"
## [343] "Calder.CL0206L"        "Calder.CL0206L"        "Calder.CL0208L"
## [346] "Calder.CL0208L"        "Calder.CL0210L"        "Calder.CL0210L"
## [349] "Calder.CL0212L"        "Calder.CL0212L"        "Calder.CL0214L"
## [352] "Calder.CL0214L"        "Calder.CL0216L"        "Calder.CL0216L"
## [355] "Calder.CL0220L"        "Calder.CL0220L"        "Calder.CL0222L"
```

```
## [358] "Calder.CL0222L"       "Calder.CL0224L"       "Calder.CL0224L"
## [361] "Calder.CL0226L"       "Calder.CL0226L"       "Calder.CL0228L"
## [364] "Calder.CL0228L"       "Calder.EG0106L"       "Calder.EG0106L"
## [367] "Calder.EG0108L"       "Calder.EG0108L"       "Calder.EG0110L"
## [370] "Calder.EG0110L"       "Calder.EG0112L"       "Calder.EG0112L"
## [373] "Calder.EG0114L"       "Calder.EG0114L"       "Calder.EG0122L"
## [376] "Calder.EG0122L"       "Calder.EG0124L"       "Calder.EG0124L"
## [379] "Calder.EG0128L"       "Calder.EG0128L"       "Calder.EG0130L"
## [382] "Calder.EG0130L"       "Calder.EG0200L"       "Calder.EG0200L"
## [385] "Calder.EG0204L2"      "Calder.EG0204L2"      "Calder.EG0206L"
## [388] "Calder.EG0206L"       "Calder.EG0208L"       "Calder.EG0208L"
## [391] "Calder.EG0210L"       "Calder.EG0210L"       "Calder.EG0212L"
## [394] "Calder.EG0212L"       "Calder.EG0214L"       "Calder.EG0214L"
## [397] "Calder.EG0222L"       "Calder.EG0222L"       "Calder.EG0226L"
## [400] "Calder.EG0226L"       "Calder.EG0228L"       "Calder.EG0228L"
## [403] "Calder.EG0306L"       "Calder.EG0306L"       "Calder.EG0308L1"
## [406] "Calder.EG0308L1"      "Calder.EG0310L"       "Calder.EG0310L"
## [409] "Calder.EG0312L"       "Calder.EG0312L"       "Calder.EG0314L"
## [412] "Calder.EG0314L"       "Calder.EG0322L"       "Calder.EG0322L"
## [415] "Calder.EG0324L"       "Calder.EG0324L"       "Calder.EG0326L"
## [418] "Calder.EG0326L"       "Calder.EG0328L"       "Calder.EG0328L"
## [421] "Calder.EG0330L"       "Calder.EG0330L"       "Calder.FB0104L"
## [424] "Calder.FB0104L"       "Calder.FB0106L"       "Calder.FB0106L"
## [427] "Calder.FB0108L"       "Calder.FB0108L"       "Calder.FB0110L"
## [430] "Calder.FB0110L"       "Calder.FB0112L"       "Calder.FB0112L"
## [433] "Calder.H10HW00_1"     "Calder.H10HW00_1"     "Calder.H11EG03_3"
## [436] "Calder.H11EG03_3"     "Calder.H12EG01_2"     "Calder.H12EG01_2"
## [439] "Calder.H13EG00_1"     "Calder.H13EG00_1"     "Calder.H14EG06_2"
## [442] "Calder.H14EG06_2"     "Calder.H15EG00_3"     "Calder.H15EG00_3"
## [445] "Calder.H16EG01_3"     "Calder.H16EG01_3"     "Calder.H17EG01_1"
## [448] "Calder.H17EG01_1"     "Calder.H18EG03_2"     "Calder.H18EG03_2"
## [451] "Calder.H19EG06_3"     "Calder.H19EG06_3"     "Calder.H1RL00_1"
## [454] "Calder.H1RL00_1"      "Calder.H20EG00_2"     "Calder.H20EG00_2"
## [457] "Calder.H21EG03_1"     "Calder.H21EG03_1"     "Calder.H22EG06_1"
## [460] "Calder.H22EG06_1"     "Calder.H23SG10_1"     "Calder.H23SG10_1"
## [463] "Calder.H24blank"      "Calder.H24blank"      "Calder.H25SV06_3"
## [466] "Calder.H25SV06_3"     "Calder.H26SR00_1"     "Calder.H26SR00_1"
## [469] "Calder.H27SG00_1"     "Calder.H27SG00_1"     "Calder.H28SV04_2"
## [472] "Calder.H28SV04_2"     "Calder.H29CL00_1"     "Calder.H29CL00_1"
## [475] "Calder.H2LS00_1"      "Calder.H2LS00_1"      "Calder.H30SV01_3"
## [478] "Calder.H30SV01_3"     "Calder.H31CL01_2"     "Calder.H31CL01_2"
## [481] "Calder.H32SV00_1"     "Calder.H32SV00_1"     "Calder.H33SV04_1"
## [484] "Calder.H33SV04_1"     "Calder.H34CL01_1"     "Calder.H34CL01_1"
## [487] "Calder.H35CL07_3"     "Calder.H35CL07_3"     "Calder.H36CL07_2"
## [490] "Calder.H36CL07_2"     "Calder.H37SG01_1"     "Calder.H37SG01_1"
## [493] "Calder.H38LO08_3"     "Calder.H38LO08_3"     "Calder.H39LO08_1"
## [496] "Calder.H39LO08_1"     "Calder.H3BL00_3"      "Calder.H3BL00_3"
## [499] "Calder.H40LO01_3"     "Calder.H40LO01_3"     "Calder.H41SV04_3"
## [502] "Calder.H41SV04_3"     "Calder.H42SG15_1"     "Calder.H42SG15_1"
## [505] "Calder.H43CL00_3"     "Calder.H43CL00_3"     "Calder.H44LO08_2"
## [508] "Calder.H44LO08_2"     "Calder.H45LO01_3"     "Calder.H45LO01_3"
## [511] "Calder.H46CL16_1"     "Calder.H46CL16_1"     "Calder.H47SV00_3"
## [514] "Calder.H47SV00_3"     "Calder.H48CL00_2"     "Calder.H48CL00_2"
## [517] "Calder.H49SV01_1"     "Calder.H49SV01_1"     "Calder.H4LW00_1"
```

```
## [520] "Calder.H4LW00_1"      "Calder.H50LO05_1"      "Calder.H50LO05_1"
## [523] "Calder.H51CL16_3"     "Calder.H51CL16_3"      "Calder.H52LO00_3"
## [526] "Calder.H52LO00_3"     "Calder.H53LO05_3"      "Calder.H53LO05_3"
## [529] "Calder.H54LO05_2"     "Calder.H54LO05_2"      "Calder.H55CL16_2"
## [532] "Calder.H55CL16_2"     "Calder.H56LO00_1"      "Calder.H56LO00_1"
## [535] "Calder.H57LO01_1"     "Calder.H57LO01_1"      "Calder.H58ML00_2"
## [538] "Calder.H58ML00_2"     "Calder.H59BL01_1"      "Calder.H59BL01_1"
## [541] "Calder.H5HL00_1"      "Calder.H5HL00_1"       "Calder.H60SV01_2"
## [544] "Calder.H60SV01_2"     "Calder.H61CL01_3"      "Calder.H61CL01_3"
## [547] "Calder.H62SV06_1"     "Calder.H62SV06_1"      "Calder.H63SV00_2"
## [550] "Calder.H63SV00_2"     "Calder.H64BL00_1"      "Calder.H64BL00_1"
## [553] "Calder.H65CL07_1"     "Calder.H65CL07_1"      "Calder.H66BL00_2"
## [556] "Calder.H66BL00_2"     "Calder.H67BL07_2"      "Calder.H67BL07_2"
## [559] "Calder.H68LL00_2"     "Calder.H68LL00_2"      "Calder.H69LO00_2"
## [562] "Calder.H69LO00_2"     "Calder.H6BL01_3"       "Calder.H6BL01_3"
## [565] "Calder.H70BL10_2"     "Calder.H70BL10_2"      "Calder.H71SV06_2"
## [568] "Calder.H71SV06_2"     "Calder.H72BL10_1"      "Calder.H72BL10_1"
## [571] "Calder.H73BL07_1"     "Calder.H73BL07_1"      "Calder.H74BL01_2"
## [574] "Calder.H74BL01_2"     "Calder.H7BL01_3"       "Calder.H7BL01_3"
## [577] "Calder.H9BL10_3"      "Calder.H9BL10_3"       "Calder.HL0106L"
## [580] "Calder.HL0106L"       "Calder.HL0108L"        "Calder.HL0108L"
## [583] "Calder.HL0110L"       "Calder.HL0110L"        "Calder.HL0112L"
## [586] "Calder.HL0112L"       "Calder.HL0114L"        "Calder.HL0114L"
## [589] "Calder.HW0106L"       "Calder.HW0106L"        "Calder.HW0108L"
## [592] "Calder.HW0108L"       "Calder.HW0110L"        "Calder.HW0110L"
## [595] "Calder.HW0112L"       "Calder.HW0112L"        "Calder.HW0114L"
## [598] "Calder.HW0114L"       "Calder.HW0122L"        "Calder.HW0122L"
## [601] "Calder.LB0200L"       "Calder.LB0200L"        "Calder.LB0206L"
## [604] "Calder.LB0206L"       "Calder.LB0208L1"       "Calder.LB0208L1"
## [607] "Calder.LB0208L2"      "Calder.LB0208L2"       "Calder.LB0210L"
## [610] "Calder.LB0210L"       "Calder.LB0212L"        "Calder.LB0212L"
## [613] "Calder.LB0214L"       "Calder.LB0214L"        "Calder.LB0222L"
## [616] "Calder.LB0222L"       "Calder.LB0224L"        "Calder.LB0224L"
## [619] "Calder.LL0102L"       "Calder.LL0102L"        "Calder.LL0104L"
## [622] "Calder.LL0104L"       "Calder.LL0106L"        "Calder.LL0106L"
## [625] "Calder.LL0108L"       "Calder.LL0108L"        "Calder.LL0110L"
## [628] "Calder.LL0110L"       "Calder.LL0112L"        "Calder.LL0112L"
## [631] "Calder.LL0114L"       "Calder.LL0114L"        "Calder.LL0116L"
## [634] "Calder.LL0116L"       "Calder.LL0120L"        "Calder.LL0120L"
## [637] "Calder.LL0122L"       "Calder.LL0122L"        "Calder.LL0124L"
## [640] "Calder.LL0124L"       "Calder.LL0126L"        "Calder.LL0126L"
## [643] "Calder.LL0128L"       "Calder.LL0128L"        "Calder.LL0130L"
## [646] "Calder.LL0130L"       "Calder.LL0132L"        "Calder.LL0132L"
## [649] "Calder.LL0134L"       "Calder.LL0134L"        "Calder.LL0206L"
## [652] "Calder.LL0206L"       "Calder.LL0208L"        "Calder.LL0208L"
## [655] "Calder.LL0210L"       "Calder.LL0210L"        "Calder.LL0212L"
## [658] "Calder.LL0212L"       "Calder.LL0214L"        "Calder.LL0214L"
## [661] "Calder.LL0222L"       "Calder.LL0222L"        "Calder.LL0224L"
## [664] "Calder.LL0224L"       "Calder.LL0226L"        "Calder.LL0226L"
## [667] "Calder.LL0228L"       "Calder.LL0228L"        "Calder.LL0306L"
## [670] "Calder.LL0306L"       "Calder.LL0308L"        "Calder.LL0308L"
## [673] "Calder.LL0310L"       "Calder.LL0310L"        "Calder.LL0312L"
## [676] "Calder.LL0312L"       "Calder.LL0314L"        "Calder.LL0314L"
## [679] "Calder.LL0322L"       "Calder.LL0322L"        "Calder.LL0324L"
```

```
## [682] "Calder.LL0324L"       "Calder.LL0326L"       "Calder.LL0326L"
## [685] "Calder.LL0328L"       "Calder.LL0328L"       "Calder.LL0330L"
## [688] "Calder.LL0330L"       "Calder.LL0404L"       "Calder.LL0404L"
## [691] "Calder.LL04102L"      "Calder.LL04102L"      "Calder.LL04106L"
## [694] "Calder.LL04106L"      "Calder.LL04110L"      "Calder.LL04110L"
## [697] "Calder.LL04114L"      "Calder.LL04114L"      "Calder.LL0438L"
## [700] "Calder.LL0438L"       "Calder.LL0450L"       "Calder.LL0450L"
## [703] "Calder.LL0454L"       "Calder.LL0454L"       "Calder.LL0458L"
## [706] "Calder.LL0458L"       "Calder.LL0462L"       "Calder.LL0462L"
## [709] "Calder.LL0466L"       "Calder.LL0466L"       "Calder.LL0470L"
## [712] "Calder.LL0470L"       "Calder.LL0474L"       "Calder.LL0474L"
## [715] "Calder.LL0478L"       "Calder.LL0478L"       "Calder.LL0482L"
## [718] "Calder.LL0482L"       "Calder.LL0486L"       "Calder.LL0486L"
## [721] "Calder.LL0490L"       "Calder.LL0490L"       "Calder.LL0494L"
## [724] "Calder.LL0494L"       "Calder.LL0498L"       "Calder.LL0498L"
## [727] "Calder.LS0104L"       "Calder.LS0104L"       "Calder.LS0110L"
## [730] "Calder.LS0110L"       "Calder.LS0112L"       "Calder.LS0112L"
## [733] "Calder.LS0114L"       "Calder.LS0114L"       "Calder.LS0116L"
## [736] "Calder.LS0116L"       "Calder.LS0118L"       "Calder.LS0118L"
## [739] "Calder.LS0120L"       "Calder.LS0120L"       "Calder.LS0122L"
## [742] "Calder.LS0122L"       "Calder.LS0124L"       "Calder.LS0124L"
## [745] "Calder.LS0126L"       "Calder.LS0126L"       "Calder.LS0128L"
## [748] "Calder.LS0128L"       "Calder.LS0130L"       "Calder.LS0130L"
## [751] "Calder.LW0100"        "Calder.LW0100"        "Calder.LW0106L"
## [754] "Calder.LW0106L"       "Calder.LW0108L"       "Calder.LW0108L"
## [757] "Calder.LW0110L"       "Calder.LW0110L"       "Calder.LW0112L"
## [760] "Calder.LW0112L"       "Calder.LW0114L"       "Calder.LW0114L"
## [763] "Calder.LW0122L"       "Calder.LW0122L"       "Calder.LW0124L"
## [766] "Calder.LW0124L"       "Calder.LW0126L"       "Calder.LW0126L"
## [769] "Calder.LW0128L"       "Calder.LW0128L"       "Calder.LWL0104L"
## [772] "Calder.LWL0104L"      "Calder.LWL100L"       "Calder.LWL100L"
## [775] "Calder.ML0100L"       "Calder.ML0100L"       "Calder.ML0206L"
## [778] "Calder.ML0206L"       "Calder.ML0208L"       "Calder.ML0208L"
## [781] "Calder.ML0210L"       "Calder.ML0210L"       "Calder.ML0212L"
## [784] "Calder.ML0212L"       "Calder.ML0214L"       "Calder.ML0214L"
## [787] "Calder.ML0220L"       "Calder.ML0220L"       "Calder.RL0100L2"
## [790] "Calder.RL0100L2"      "Calder.RL0100L2"      "Calder.RL0100L2"
## [793] "Calder.RL0104L"       "Calder.RL0104L"       "Calder.RL0204L"
## [796] "Calder.RL0204L"       "Calder.RL0206L"       "Calder.RL0206L"
## [799] "Calder.SG0104L"       "Calder.SG0104L"       "Calder.SG0106L"
## [802] "Calder.SG0106L"       "Calder.SG0108L"       "Calder.SG0108L"
## [805] "Calder.SG0110L"       "Calder.SG0110L"       "Calder.SG0112L"
## [808] "Calder.SG0112L"       "Calder.SM0100L"       "Calder.SM0100L"
## [811] "Calder.SM0104L"       "Calder.SM0104L"       "Calder.SR0102L"
## [814] "Calder.SR0102L"       "Calder.SR0104L"       "Calder.SR0104L"
## [817] "Calder.SR0106L"       "Calder.SR0106L"       "Calder.SR0108L"
## [820] "Calder.SR0108L"       "Calder.SR0110L"       "Calder.SR0110L"
## [823] "Calder.SR0112L"       "Calder.SR0112L"       "Calder.SR0114L"
## [826] "Calder.SR0114L"       "Calder.SR0122L"       "Calder.SR0122L"
## [829] "Calder.SR0126L"       "Calder.SR0126L"       "Calder.SR0128L"
## [832] "Calder.SR0128L"       "Calder.SR0130L"       "Calder.SR0130L"
## [835] "Calder.SR0132L"       "Calder.SR0132L"       "Calder.SV0100L"
## [838] "Calder.SV0100L"       "Calder.SV0104L"       "Calder.SV0104L"
## [841] "Calder.SV0206L"       "Calder.SV0206L"       "Calder.SV0208L"
```

```
## [844] "Calder.SV0208L"        "Calder.SV0210L"        "Calder.SV0210L"
## [847] "Calder.SV0212L"        "Calder.SV0212L"        "Calder.SV0214L"
## [850] "Calder.SV0214L"        "Calder.SV0222L"        "Calder.SV0222L"
## [853] "Calder.SV0224L"        "Calder.SV0224L"        "Calder.SV0226L"
## [856] "Calder.SV0226L"        "Calder.SV0306L"        "Calder.SV0306L"
## [859] "Calder.SV0308L"        "Calder.SV0308L"        "Calder.SV0310L"
## [862] "Calder.SV0310L"        "Calder.SV0312L"        "Calder.SV0312L"
## [865] "Calder.SV0314L"        "Calder.SV0314L"        "Calder.SV0322L"
## [868] "Calder.SV0322L"        "Calder.SV0324L"        "Calder.SV0324L"
## [871] "Calder.SV0326L"        "Calder.SV0326L"        "Calder.SV0404L"
## [874] "Calder.SV0404L"        "Calder.SV0406L"        "Calder.SV0406L"
## [877] "Calder.SV0408L"        "Calder.SV0408L"        "Calder.SV0410L"
## [880] "Calder.SV0410L"        "Calder.SV0412L"        "Calder.SV0412L"
## [883] "Calder.SV0414L"        "Calder.SV0414L"        "Calder.SV0418L"
## [886] "Calder.SV0418L"        "Calder.SV0422L2"       "Calder.SV0422L2"
## [889] "Calder.SV0424L2"       "Calder.SV0424L2"       "Calder.SV0426L"
## [892] "Calder.SV0426L"        "Calder.TL0100L2"       "Calder.TL0100L2"
## [895] "Calder.TL0100L2"       "Calder.TL0100L2"       "Calder.TL0104L"
## [898] "Calder.TL0104L"        "MAWC.PossBlank_5DGR4A" "MAWC.PossBlank_5DGR4A"
## [901] "MAWC.PossBlank_5DGR4B" "MAWC.PossBlank_5DGR4B" "MAWC.PossBlank_5DGr3A"
## [904] "MAWC.PossBlank_5DGr3A" "MAWC.PossBlank_5DGr3B" "MAWC.PossBlank_5DGr3B"
## [907] "MAWC.PossBlank_5DGr3C" "MAWC.PossBlank_5DGr3C" "MAWC.PossBlank_5DGr3D"
## [910] "MAWC.PossBlank_5DGr3D" "MAWC.PossBlank_5DGr3E" "MAWC.PossBlank_5DGr3E"
```

```
rm(remove_df)

#overwrite esv_table with columns (samples) removed
esv_table<-esv_table[,positions_keep]

#remove ESVs without reads
reads_per_esv<-as.data.frame(rowSums(esv_table))
names(reads_per_esv)<-"reads"
reads_per_esv$esv<-rownames(reads_per_esv)
rownames(reads_per_esv)<-NULL

# count ESVS with 0 reads after removing samples
table(reads_per_esv$reads==0)
```

```
##
##  FALSE   TRUE
## 219080   4084
```

```
# remove ESVs with 0 reads (if there are any)
if(length(which(reads_per_esv$reads<1))>0){
     esv_table<-esv_table[which(reads_per_esv$reads>0),]
}

rm(reads_per_esv); rm(i); rm(ESV_names);rm(keep);rm(positions_keep)
```

@count_track (samples and blanks)

So far we have: - removed samples that should not be included in this study - removed ESVs without reads in the ESV table - blanks are included

```r
#starting number of reads in the ESV table including blanks
sum(colSums(esv_table))
```

```
## [1] 70372389
```

```r
#average number of reads per sample + blanks before any further filtering
sum(colSums(esv_table))/(length(esv_table)/2) #divided by two here because we have not summed technical
```

```
## [1] 106463.5
```

```r
#number of ESVS
nrow(esv_table)
```

```
## [1] 219080
```

@count_track (blanks)

These are the number of reads in each blank sample after: - removed ESVs without reads in the ESV table

```r
blanks<-c("Calder.Blank3.16S.GGCGCCGAA.GTAACCTAA",
 "Calder.Blank3.16S.TAGAACGAA.GTAACCTAA"    ,
 "Calder.JC7_Blank1.16S.GAGCCGCAA.GGTTAACCA"        ,
  "Calder.JC7_Blank1.16S.TACTTGCAA.GGTTAACCA"    ,
  "Calder.JC7_Blank2.16S.GGTACTCAA.AGACGACCA"     ,
  "Calder.JC7_Blank2.16S.TACTTGCAA.AGACGACCA"     ,
  "Calder.JC7_Blank3.16S.CTCAGCAA.TTGGACGGCA"     ,
 "Calder.JC7_Blank3.16S.GGTACTCAA.TTGGACGGCA"    ,
 "Calder.JC7_Blank4.16S.CTCAGCAA.CATAATTGCA"     ,
 "Calder.JC7_Blank4.16S.GTTGGCCAA.CATAATTGCA"     ,
 "Calder.JC7_Blank5.16S.GTTGGCCAA.CAATTAATCA"      ,
 "Calder.JC7_Blank5.16S.TCTCTCCAA.CAATTAATCA"       ,
 "Calder.JC7_Blank6.16S.CGTCAGCCAA.TATTCTATCA"       ,
 "Calder.JC7_Blank6.16S.TCTCTCCAA.TATTCTATCA"        ,
 "Calder.JC7_Blank7.16S.CGTCAGCCAA.AGCATGCTCA"        ,
 "Calder.JC7_Blank7.16S.GTATAGCCAA.AGCATGCTCA"        ,
 "Calder.JC_Blank1.16S.CAGCAGAA.AGCATGCTCA"           ,
 "Calder.JC_Blank1.16S.GTCTCGAA.AGCATGCTCA"           ,
"Calder.JC_Blank2.16S.ATGGATAA.ATTCTGGAA"            ,
 "Calder.JC_Blank2.16S.TTACCTAA.ATTCTGGAA"            ,
 "Calder.JC_Blank3.16S.AGGTAACCAA.CTGGATGAA"          ,
 "Calder.JC_Blank3.16S.TTACCTAA.CTGGATGAA"            ,
 "MAWC.BLANK.16S.CCTTATCAA.TAACCAGCAA"                ,
 "MAWC.BLANK.16S.CGGTCCAA.TAACCAGCAA"                 ,
 "MAWC.BLANK1_DG1.16S.GCGCAAGA.AGCTAAGA"              ,
 "MAWC.BLANK1_DG1.16S.TTACCTCAA.AGCTAAGA"             ,
 "MAWC.BLANK2_DG2.16S.AGCCTGGCAA.AGCTAAGA"            ,
 "MAWC.BLANK2_DG2.16S.CGCCTCCA.AGCTAAGA"              ,
 "MAWC.BLANK3_DG3.16S.AGACGCAA.TCTCTCCAA"             ,
 "MAWC.BLANK3_DG3.16S.CCAGAACCAA.TCTCTCCAA"           ,
 "MAWC.BLANK4_DG4_BLANK4_DG4.16S.GTTGGCCAA.TCCGCTCA"  ,
 "MAWC.BLANK4_DG4_BLANK4_DG4.16S.TCTCTCCAA.TCCGCTCA"  ,
```

```
 "SNOTEL.IOANA_D12.16S.CATTGACCAA.GACGCAAGAA"          ,
 "SNOTEL.IOANA_D12.16S.TCGCGACCAA.GACGCAAGAA"          ,
 "SNOTEL.IOANA_E12.16S.CAGCAGAA.TCAAGAAGAA"            ,
 "SNOTEL.IOANA_E12.16S.CATTGACCAA.TCAAGAAGAA"          ,
 "SNOTEL.IOANA_F12.16S.CAGCAGAA.GGTTGAAGAA"            ,
 "SNOTEL.IOANA_F12.16S.GTCTCGAA.GGTTGAAGAA"            ,
 "SNOTEL.IOANA_H12.16S.ATGGATAA.AGCTAAGA"              ,
 "SNOTEL.IOANA_H12.16S.CCATGGAA.AGCTAAGA")


# subset blanks from esv_table & calculate reads
blank_data<-esv_table[, names(esv_table)%in%blanks]
blank_reads<-as.data.frame(colSums(blank_data))
names(blank_reads)<-"reads"
blank_reads$sample<-rownames(blank_reads)
rownames(blank_reads)<-NULL
blank_reads
```

```
##    reads                                                sample
## 1    120              Calder.Blank3.16S.GGCGCCGAA.GTAACCTAA
## 2    123              Calder.Blank3.16S.TAGAACGAA.GTAACCTAA
## 3     29          Calder.JC7_Blank1.16S.GAGCCGCAA.GGTTAACCA
## 4     46          Calder.JC7_Blank1.16S.TACTTGCAA.GGTTAACCA
## 5    207          Calder.JC7_Blank2.16S.GGTACTCAA.AGACGACCA
## 6     69          Calder.JC7_Blank2.16S.TACTTGCAA.AGACGACCA
## 7     52          Calder.JC7_Blank3.16S.CTCAGCAA.TTGGACGGCA
## 8    238         Calder.JC7_Blank3.16S.GGTACTCAA.TTGGACGGCA
## 9     44          Calder.JC7_Blank4.16S.CTCAGCAA.CATAATTGCA
## 10    55         Calder.JC7_Blank4.16S.GTTGGCCAA.CATAATTGCA
## 11    81         Calder.JC7_Blank5.16S.GTTGGCCAA.CAATTAATCA
## 12    49         Calder.JC7_Blank5.16S.TCTCTCCAA.CAATTAATCA
## 13    51        Calder.JC7_Blank6.16S.CGTCAGCCAA.TATTCTATCA
## 14    59         Calder.JC7_Blank6.16S.TCTCTCCAA.TATTCTATCA
## 15    30        Calder.JC7_Blank7.16S.CGTCAGCCAA.AGCATGCTCA
## 16    51        Calder.JC7_Blank7.16S.GTATAGCCAA.AGCATGCTCA
## 17    31           Calder.JC_Blank1.16S.CAGCAGAA.AGCATGCTCA
## 18    31           Calder.JC_Blank1.16S.GTCTCGAA.AGCATGCTCA
## 19 32662            Calder.JC_Blank2.16S.ATGGATAA.ATTCTGGAA
## 20 14318            Calder.JC_Blank2.16S.TTACCTAA.ATTCTGGAA
## 21    37          Calder.JC_Blank3.16S.AGGTAACCAA.CTGGATGAA
## 22    32            Calder.JC_Blank3.16S.TTACCTAA.CTGGATGAA
## 23   418                MAWC.BLANK.16S.CCTTATCAA.TAACCAGCAA
## 24   714                 MAWC.BLANK.16S.CGGTCCAA.TAACCAGCAA
## 25 36438              MAWC.BLANK1_DG1.16S.GCGCAAGA.AGCTAAGA
## 26 59524             MAWC.BLANK1_DG1.16S.TTACCTCAA.AGCTAAGA
## 27    35            MAWC.BLANK2_DG2.16S.AGCCTGGCAA.AGCTAAGA
## 28    40              MAWC.BLANK2_DG2.16S.CGCCTCCA.AGCTAAGA
## 29   500             MAWC.BLANK3_DG3.16S.AGACGCAA.TCTCTCCAA
## 30   516           MAWC.BLANK3_DG3.16S.CCAGAACCAA.TCTCTCCAA
## 31   295 MAWC.BLANK4_DG4_BLANK4_DG4.16S.GTTGGCCAA.TCCGCTCA
## 32   395 MAWC.BLANK4_DG4_BLANK4_DG4.16S.TCTCTCCAA.TCCGCTCA
## 33   296         SNOTEL.IOANA_D12.16S.CATTGACCAA.GACGCAAGAA
## 34   302         SNOTEL.IOANA_D12.16S.TCGCGACCAA.GACGCAAGAA
```

```
## 35   472          SNOTEL.IOANA_E12.16S.CAGCAGAA.TCAAGAAGAA
## 36   312        SNOTEL.IOANA_E12.16S.CATTGACCAA.TCAAGAAGAA
## 37   245          SNOTEL.IOANA_F12.16S.CAGCAGAA.GGTTGAAGAA
## 38   182          SNOTEL.IOANA_F12.16S.GTCTCGAA.GGTTGAAGAA
## 39     5            SNOTEL.IOANA_H12.16S.ATGGATAA.AGCTAAGA
## 40    11            SNOTEL.IOANA_H12.16S.CCATGGAA.AGCTAAGA
```

```
# number of total reads
sum(colSums(blank_data))
```

```
## [1] 149115
```

```
# number of blanks
length(blank_data)
```

```
## [1] 40
```

```
# summary of blank reads
summary(blank_reads$reads)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     5.0    43.0   100.5  3727.9   332.8 59524.0
```

```
# number of ESVs in the blank samples
tmp<-as.data.frame(rowSums(blank_data))
tmp<-tmp[tmp$`rowSums(blank_data)`>0,]
length(tmp)
```

```
## [1] 18684
```

```
rm(blank_reads)
rm(blank_data)
rm(tmp)
```

@count_track (samples)

So far we have: - removed samples that should not be included in this study - removed ESVs without reads in the ESV table - excluded 11 blanks

```
# remove blanks from esv_table
esv_table_no_blanks<-esv_table[, names(esv_table)%in%setdiff(names(esv_table),blanks)]

# total reads
sum(colSums(esv_table_no_blanks))
```

```
## [1] 70223274
```

```
#average number of reads per sample
sum(colSums(esv_table_no_blanks))/(length(esv_table_no_blanks)/2)
```

```
## [1] 109552.7
```

```
# number of samples
ncol(esv_table_no_blanks)
```

```
## [1] 1282
```

```
rm(blanks)
rm(esv_table_no_blanks)
```

b. Remove ESVs that were removed from the taxonomy table

esv_table and tax_tab have a different number of ESVs because we deleted ESVs in the taxonomy table by removing chloroplast, mitochondria, and kingdom == NA

```
keep<-rownames(tax_tab)
esvs<-rownames(esv_table)
removefromesvtab<-setdiff(esvs,keep)
table(esvs%in%keep)
```

```
##
##  FALSE   TRUE
##   3235 215845
```

```
table(row.names(esv_table) %in% removefromesvtab)
```

```
##
##  FALSE   TRUE
## 215845   3235
```

```
# remove esvs that are not in the taxonomy table
esv_table<-esv_table[!(row.names(esv_table) %in% removefromesvtab),]


rm(keep)
rm(esvs)
rm(removefromesvtab)
```

@count_track (samples and blanks)

So far we have: - removed samples that should not be included in this study - removed ESVs without reads in the ESV table - 11 blanks are included - removed ESVs assigned to chloroplast, mitochondria, and kingdom NA

```
#total number of reads
sum(colSums(esv_table))
```

```
## [1] 68540950
```

```
#number of ESVS
nrow(esv_table)
```

```
## [1] 215845
```

```
#avg. number of reads per sample
sum(colSums(esv_table))/(length(esv_table)/2) #divided by two because we still haven't summed technical
```

```
## [1] 103692.8
```

```
#number of technically replicated samples with blanks included
ncol(esv_table)
```

```
## [1] 1322
```

@count_track (blanks)

These are the number of reads in each blank sample after: - removed ESVs without reads in the ESV table - removed ESVs assigned to chloroplast, mitochondria, and kingdom NA

```
blanks<-c("Calder.Blank3.16S.GGCGCCGAA.GTAACCTAA",
 "Calder.Blank3.16S.TAGAACGAA.GTAACCTAA"    ,
 "Calder.JC7_Blank1.16S.GAGCCGCAA.GGTTAACCA"        ,
  "Calder.JC7_Blank1.16S.TACTTGCAA.GGTTAACCA"    ,
  "Calder.JC7_Blank2.16S.GGTACTCAA.AGACGACCA"     ,
  "Calder.JC7_Blank2.16S.TACTTGCAA.AGACGACCA"     ,
  "Calder.JC7_Blank3.16S.CTCAGCAA.TTGGACGGCA"     ,
 "Calder.JC7_Blank3.16S.GGTACTCAA.TTGGACGGCA"    ,
 "Calder.JC7_Blank4.16S.CTCAGCAA.CATAATTGCA"     ,
 "Calder.JC7_Blank4.16S.GTTGGCCAA.CATAATTGCA"     ,
 "Calder.JC7_Blank5.16S.GTTGGCCAA.CAATTAATCA"      ,
 "Calder.JC7_Blank5.16S.TCTCTCCAA.CAATTAATCA"       ,
 "Calder.JC7_Blank6.16S.CGTCAGCCAA.TATTCTATCA"       ,
 "Calder.JC7_Blank6.16S.TCTCTCCAA.TATTCTATCA"         ,
 "Calder.JC7_Blank7.16S.CGTCAGCCAA.AGCATGCTCA"         ,
 "Calder.JC7_Blank7.16S.GTATAGCCAA.AGCATGCTCA"         ,
 "Calder.JC_Blank1.16S.CAGCAGAA.AGCATGCTCA"            ,
 "Calder.JC_Blank1.16S.GTCTCGAA.AGCATGCTCA"            ,
"Calder.JC_Blank2.16S.ATGGATAA.ATTCTGGAA"             ,
 "Calder.JC_Blank2.16S.TTACCTAA.ATTCTGGAA"             ,
 "Calder.JC_Blank3.16S.AGGTAACCAA.CTGGATGAA"           ,
 "Calder.JC_Blank3.16S.TTACCTAA.CTGGATGAA"             ,
 "MAWC.BLANK.16S.CCTTATCAA.TAACCAGCAA"                 ,
 "MAWC.BLANK.16S.CGGTCCAA.TAACCAGCAA"                  ,
 "MAWC.BLANK1_DG1.16S.GCGCAAGA.AGCTAAGA"               ,
```

```
 "MAWC.BLANK1_DG1.16S.TTACCTCAA.AGCTAAGA"              ,
 "MAWC.BLANK2_DG2.16S.AGCCTGGCAA.AGCTAAGA"             ,
 "MAWC.BLANK2_DG2.16S.CGCCTCCA.AGCTAAGA"               ,
 "MAWC.BLANK3_DG3.16S.AGACGCAA.TCTCTCCAA"              ,
 "MAWC.BLANK3_DG3.16S.CCAGAACCAA.TCTCTCCAA"            ,
 "MAWC.BLANK4_DG4_BLANK4_DG4.16S.GTTGGCCAA.TCCGCTCA"   ,
 "MAWC.BLANK4_DG4_BLANK4_DG4.16S.TCTCTCCAA.TCCGCTCA"   ,
 "SNOTEL.IOANA_D12.16S.CATTGACCAA.GACGCAAGAA"          ,
 "SNOTEL.IOANA_D12.16S.TCGCGACCAA.GACGCAAGAA"          ,
 "SNOTEL.IOANA_E12.16S.CAGCAGAA.TCAAGAAGAA"            ,
 "SNOTEL.IOANA_E12.16S.CATTGACCAA.TCAAGAAGAA"          ,
 "SNOTEL.IOANA_F12.16S.CAGCAGAA.GGTTGAAGAA"            ,
 "SNOTEL.IOANA_F12.16S.GTCTCGAA.GGTTGAAGAA"            ,
 "SNOTEL.IOANA_H12.16S.ATGGATAA.AGCTAAGA"              ,
 "SNOTEL.IOANA_H12.16S.CCATGGAA.AGCTAAGA")


# subset blanks from esv_table & calculate reads
blank_data<-esv_table[, names(esv_table)%in%blanks]
blank_reads<-as.data.frame(colSums(blank_data))
names(blank_reads)<-"reads"
blank_reads$sample<-rownames(blank_reads)
rownames(blank_reads)<-NULL
blank_reads
```

```
##    reads                                              sample
## 1     52              Calder.Blank3.16S.GGCGCCGAA.GTAACCTAA
## 2     46              Calder.Blank3.16S.TAGAACGAA.GTAACCTAA
## 3      3         Calder.JC7_Blank1.16S.GAGCCGCAA.GGTTAACCA
## 4     15         Calder.JC7_Blank1.16S.TACTTGCAA.GGTTAACCA
## 5    142         Calder.JC7_Blank2.16S.GGTACTCAA.AGACGACCA
## 6      4         Calder.JC7_Blank2.16S.TACTTGCAA.AGACGACCA
## 7      5         Calder.JC7_Blank3.16S.CTCAGCAA.TTGGACGGCA
## 8     15        Calder.JC7_Blank3.16S.GGTACTCAA.TTGGACGGCA
## 9      5         Calder.JC7_Blank4.16S.CTCAGCAA.CATAATTGCA
## 10     5        Calder.JC7_Blank4.16S.GTTGGCCAA.CATAATTGCA
## 11     3        Calder.JC7_Blank5.16S.GTTGGCCAA.CAATTAATCA
## 12     2        Calder.JC7_Blank5.16S.TCTCTCCAA.CAATTAATCA
## 13     5       Calder.JC7_Blank6.16S.CGTCAGCCAA.TATTCTATCA
## 14     7        Calder.JC7_Blank6.16S.TCTCTCCAA.TATTCTATCA
## 15     2       Calder.JC7_Blank7.16S.CGTCAGCCAA.AGCATGCTCA
## 16    15       Calder.JC7_Blank7.16S.GTATAGCCAA.AGCATGCTCA
## 17     3          Calder.JC_Blank1.16S.CAGCAGAA.AGCATGCTCA
## 18     2          Calder.JC_Blank1.16S.GTCTCGAA.AGCATGCTCA
## 19 31387           Calder.JC_Blank2.16S.ATGGATAA.ATTCTGGAA
## 20 13712           Calder.JC_Blank2.16S.TTACCTAA.ATTCTGGAA
## 21     5         Calder.JC_Blank3.16S.AGGTAACCAA.CTGGATGAA
## 22     4           Calder.JC_Blank3.16S.TTACCTAA.CTGGATGAA
## 23   326               MAWC.BLANK.16S.CCTTATCAA.TAACCAGCAA
## 24   576                MAWC.BLANK.16S.CGGTCCAA.TAACCAGCAA
## 25 35903            MAWC.BLANK1_DG1.16S.GCGCAAGA.AGCTAAGA
## 26 58683           MAWC.BLANK1_DG1.16S.TTACCTCAA.AGCTAAGA
## 27    35           MAWC.BLANK2_DG2.16S.AGCCTGGCAA.AGCTAAGA
```

```
## 28    40              MAWC.BLANK2_DG2.16S.CGCCTCCA.AGCTAAGA
## 29   497            MAWC.BLANK3_DG3.16S.AGACGCAA.TCTCTCCAA
## 30   511          MAWC.BLANK3_DG3.16S.CCAGAACCAA.TCTCTCCAA
## 31   293 MAWC.BLANK4_DG4_BLANK4_DG4.16S.GTTGGCCAA.TCCGCTCA
## 32   394 MAWC.BLANK4_DG4_BLANK4_DG4.16S.TCTCTCCAA.TCCGCTCA
## 33   295        SNOTEL.IOANA_D12.16S.CATTGACCAA.GACGCAAGAA
## 34   300        SNOTEL.IOANA_D12.16S.TCGCGACCAA.GACGCAAGAA
## 35   470          SNOTEL.IOANA_E12.16S.CAGCAGAA.TCAAGAAGAA
## 36   310        SNOTEL.IOANA_E12.16S.CATTGACCAA.TCAAGAAGAA
## 37   245          SNOTEL.IOANA_F12.16S.CAGCAGAA.GGTTGAAGAA
## 38   182          SNOTEL.IOANA_F12.16S.GTCTCGAA.GGTTGAAGAA
## 39     5            SNOTEL.IOANA_H12.16S.ATGGATAA.AGCTAAGA
## 40    11            SNOTEL.IOANA_H12.16S.CCATGGAA.AGCTAAGA
```

```
# number of total reads
sum(colSums(blank_data))
```

```
## [1] 144515
```

```
# number of blanks
length(blank_data)
```

```
## [1] 40
```

```
rm(blank_reads)
rm(blanks)
rm(blank_data)
```

3. Sum technical replicates

a. Test for correlation between technical replicates

###QUESTION

Is this all one sample? meaning four technical replicates instead of two? MAWC.GG1A.16S.AGACGCAA.GAAGGTAA MAWC.GG1A.16S.CATACTAA.TAACCAGCAA MAWC.GG1A.16S.CCAGAACCAA.GAAGGTAA MAWC.GG1A.16S.CCAGAACCAA.TAACCAGCAA

```
esv_tab_derep <- esv_table

# pull out the sample name from longer sample ID string
# split by "_" 4 times and save the forth column with the sample name
samp_names<-str_split(names(esv_tab_derep),pattern = ".16S.", 2, simplify = T)[,1]

names(esv_tab_derep)<-samp_names

cor_df<-list()
for(i in 1:length(samp_names)){
        for(j in 1:length(samp_names)){
                if(samp_names[i] == samp_names[j] & i!=j & i>j){
                        tmp<-cbind(esv_tab_derep[i], esv_tab_derep[j])
                        cor_stat <-data.frame(cor(tmp[1], tmp[2]))
```

```
                        cor_df<-c(cor_df, cor_stat)
                }
        }
}

# unlist to a dataframe containing sample ID and the correlation among technical replicates.
cor_df <- data.frame(SampleID = rep(names(cor_df), sapply(cor_df, length)), correlation = unlist(cor_df)

# Look at samples less than 70% correlated
low_cor_samp<-cor_df[cor_df$correlation<0.7,]
#number of samples with low correlation
nrow(low_cor_samp)
```

```
## [1] 42
```

```
rm(cor_df)
rm(cor_stat)
#rm(low_cor_samp)
rm(i)
rm(j)
rm(esv_tab_derep)
```

b. Assess read numbers of technical replicates with low correlations

```
reads<-as.data.frame(colSums(esv_table))
names(reads)<-"reads"
reads$sample<-rownames(reads)
rownames(reads)<-NULL
reads$SampleID<-str_split(reads$sample,pattern = ".16S.", 2, simplify = T)[,1]
reads$sample<-NULL
head(reads)
```

```
##   reads          SampleID
## 1    52     Calder.Blank3
## 2    46     Calder.Blank3
## 3     3 Calder.JC7_Blank1
## 4    15 Calder.JC7_Blank1
## 5   142 Calder.JC7_Blank2
## 6     4 Calder.JC7_Blank2
```

```
cor70<-reads[which(reads$SampleID%in%low_cor_samp$SampleID),]

cor70 <- cor70 %>%
  group_by(SampleID) %>%
  summarise(
    reads_low = min(reads),
    reads_high = max(reads)
  ) %>%
  select(reads_low, reads_high, SampleID) %>%
        inner_join(.,low_cor_samp, by=join_by(SampleID)) %>%
```

```
        mutate(percent=reads_low/reads_high,
               total_reads=reads_low+reads_high,
               keep=case_when(percent < 0.25 | total_reads < 2000 ~"keep"))

# samples to be removed in next step that do not meet the criteria listed below
remove<-cor70 %>% filter(is.na(keep)==T)%>%pull(SampleID)

rm(reads)
rm(cor70)
rm(low_cor_samp)
rm(samp_names)
```

For technical replicates that were correlated less than 70%, I kept samples if the technical replicate with the smaller number of reads was less than 25% of the total reads in the larger technical replicate. If samples that were less than 70% correlated and the technical replicate with the smaller number of reads was over 25% of the technical replicate, I retained the sample if the total reads between both technical replicates was less than 2000. Technical replicates in the samples explained in the last sentence are likely uncorrelated because of the overall low number of reads. These samples end up getting filtered out during the normalizing step where we remove samples with less than 10,000 reads in Step 7.

c. Sum technical replicates

```
esv_tab_derep <- esv_table

samp_names<-str_split(names(esv_tab_derep),pattern = ".16S.", 2, simplify = T)[,1]

# starting number of unique samples
length(unique(samp_names))
```

```
## [1] 660
```

```
# rename columns in esv_tab_derep with the extracted info in samp_names
names(esv_tab_derep)<-samp_names
table(names(esv_tab_derep)==str_split(names(esv_table),pattern = ".16S.", 2, simplify = T)[,1]) # all T
```

```
##
## TRUE
## 1322
```

```
summed_esv_table <- rowsum(t(esv_tab_derep), group = colnames(esv_tab_derep), na.rm = T)
summed_esv_table<-as.data.frame(t(summed_esv_table))

# check this worked
original<-esv_table
names(original)<-samp_names
original<-original[,order(names(original))]

original[1:10,1:5]
```

```
##             Calder.33_1_4_DNA Calder.33_1_4_DNA.1 Calder.34_1_0_DNA
## ZOTU_10                   491                 202                 2
## ZOTU_100                    0                   0                 4
## ZOTU_1000                   0                   0                 0
## ZOTU_10000                  0                   0                 0
## ZOTU_100000                 0                   0                 0
## ZOTU_100001                 0                   0                 0
## ZOTU_100002                 0                   0                 0
## ZOTU_100003                 0                   0                 0
## ZOTU_100004                 0                   0                 0
## ZOTU_100005                 0                   0                 0
##             Calder.34_1_0_DNA.1 Calder.34_1_16_DNA
## ZOTU_10                       2                28
## ZOTU_100                      4                 6
## ZOTU_1000                     0                 0
## ZOTU_10000                    0                 0
## ZOTU_100000                   0                 0
## ZOTU_100001                   0                 0
## ZOTU_100002                   0                 0
## ZOTU_100003                   0                 0
## ZOTU_100004                   0                 0
## ZOTU_100005                   0                 0
```

```
summed_esv_table[1:10,1:6]
```

```
##             Calder.33_1_4_DNA Calder.34_1_0_DNA Calder.34_1_16_DNA
## ZOTU_10                   693                 4                102
## ZOTU_100                    0                 8                 23
## ZOTU_1000                   0                 0                  0
## ZOTU_10000                  0                 0                  0
## ZOTU_100000                 0                 0                  0
## ZOTU_100001                 0                 0                  0
## ZOTU_100002                 0                 0                  0
## ZOTU_100003                 0                 0                  0
## ZOTU_100004                 0                 0                  0
## ZOTU_100005                 0                 0                  0
##             Calder.34_1_18_DNA Calder.34_1_2_DNA Calder.34_1_20_DNA
## ZOTU_10                    240                 5                305
## ZOTU_100                    11                26                 22
## ZOTU_1000                    0                 0                  0
## ZOTU_10000                   0                 0                  0
## ZOTU_100000                  0                 0                  0
## ZOTU_100001                  0                 0                  0
## ZOTU_100002                  0                 0                  0
## ZOTU_100003                  0                 0                  0
## ZOTU_100004                  0                 0                  0
## ZOTU_100005                  0                 0                  0
```

```
# remove extra dataframes
rm(original)
rm(samp_names)
rm(esv_tab_derep)
```

```r
# midway count track
sum(colSums(summed_esv_table))
```

```
## [1] 68540950
```

```r
#remove samples with low reads
where<-which(names(summed_esv_table) %in% remove)
table(names(summed_esv_table)[where] %in% remove)
```

```
##
## TRUE
##    6
```

```r
summed_esv_table_goodcorronly<-summed_esv_table[,-where]
table(names(summed_esv_table_goodcorronly)%in% remove) #all FALSE
```

```
##
## FALSE
##   654
```

```r
# rewrite esv_table
esv_table<-summed_esv_table_goodcorronly
rm(summed_esv_table_goodcorronly)

# second midway count track
sum(colSums(esv_table))
```

```
## [1] 68504654
```

```r
# check for ESVs with zeros reads again
reads_per_esv<-as.data.frame(rowSums(esv_table))
names(reads_per_esv)<-"reads"
reads_per_esv$esv<-rownames(reads_per_esv)
rownames(reads_per_esv)<-NULL

# remove ESVs with 0 reads (if there are any)
if(length(which(reads_per_esv$reads<1))>0){
     esv_table<-esv_table[which(reads_per_esv$reads>0),]
}


rm(reads_per_esv)
rm(where)
rm(remove)
rm(summed_esv_table)
```

@count_track (samples and blanks)

So far we have: - removed samples that should not be included in this study - removed ESVs without reads in the ESV table - 11 blanks are included - removed ESVs assigned to chloroplast, mitochondria, and kingdom NA - summed technical replicates and removed one sample that was uncorrelated with high reads

```
# total number of reads
sum(colSums(esv_table))
```

```
## [1] 68504654
```

```
# number of ESVS
nrow(esv_table)
```

```
## [1] 215806
```

```
# avg number of reads per sample
sum(colSums(esv_table))/(length(esv_table))
```

```
## [1] 104747.2
```

```
# number of samples including blanks
ncol(esv_table)
```

```
## [1] 654
```

@count_track (blanks)

These are the number of reads in each blank sample after: - removed ESVs without reads in the ESV table - removed ESVs assigned to chloroplast, mitochondria, and kingdom NA - technical replicates summed

```
blanks<-c("Calder.Blank3.16S.GGCGCCGAA.GTAACCTAA",
 "Calder.Blank3.16S.TAGAACGAA.GTAACCTAA"    ,
 "Calder.JC7_Blank1.16S.GAGCCGCAA.GGTTAACCA"        ,
  "Calder.JC7_Blank1.16S.TACTTGCAA.GGTTAACCA"    ,
  "Calder.JC7_Blank2.16S.GGTACTCAA.AGACGACCA"     ,
  "Calder.JC7_Blank2.16S.TACTTGCAA.AGACGACCA"     ,
  "Calder.JC7_Blank3.16S.CTCAGCAA.TTGGACGGCA"     ,
 "Calder.JC7_Blank3.16S.GGTACTCAA.TTGGACGGCA"    ,
 "Calder.JC7_Blank4.16S.CTCAGCAA.CATAATTGCA"     ,
 "Calder.JC7_Blank4.16S.GTTGGCCAA.CATAATTGCA"     ,
 "Calder.JC7_Blank5.16S.GTTGGCCAA.CAATTAATCA"      ,
 "Calder.JC7_Blank5.16S.TCTCTCCAA.CAATTAATCA"       ,
 "Calder.JC7_Blank6.16S.CGTCAGCCAA.TATTCTATCA"       ,
 "Calder.JC7_Blank6.16S.TCTCTCCAA.TATTCTATCA"         ,
 "Calder.JC7_Blank7.16S.CGTCAGCCAA.AGCATGCTCA"         ,
 "Calder.JC7_Blank7.16S.GTATAGCCAA.AGCATGCTCA"         ,
 "Calder.JC_Blank1.16S.CAGCAGAA.AGCATGCTCA"            ,
 "Calder.JC_Blank1.16S.GTCTCGAA.AGCATGCTCA"            ,
"Calder.JC_Blank2.16S.ATGGATAA.ATTCTGGAA"             ,
 "Calder.JC_Blank2.16S.TTACCTAA.ATTCTGGAA"             ,
 "Calder.JC_Blank3.16S.AGGTAACCAA.CTGGATGAA"           ,
 "Calder.JC_Blank3.16S.TTACCTAA.CTGGATGAA"             ,
 "MAWC.BLANK.16S.CCTTATCAA.TAACCAGCAA"                 ,
 "MAWC.BLANK.16S.CGGTCCAA.TAACCAGCAA"                  ,
 "MAWC.BLANK1_DG1.16S.GCGCAAGA.AGCTAAGA"               ,
```

```
 "MAWC.BLANK1_DG1.16S.TTACCTCAA.AGCTAAGA"              ,
 "MAWC.BLANK2_DG2.16S.AGCCTGGCAA.AGCTAAGA"             ,
 "MAWC.BLANK2_DG2.16S.CGCCTCCA.AGCTAAGA"               ,
 "MAWC.BLANK3_DG3.16S.AGACGCAA.TCTCTCCAA"              ,
 "MAWC.BLANK3_DG3.16S.CCAGAACCAA.TCTCTCCAA"            ,
 "MAWC.BLANK4_DG4_BLANK4_DG4.16S.GTTGGCCAA.TCCGCTCA"   ,
 "MAWC.BLANK4_DG4_BLANK4_DG4.16S.TCTCTCCAA.TCCGCTCA"   ,
 "SNOTEL.IOANA_D12.16S.CATTGACCAA.GACGCAAGAA"          ,
 "SNOTEL.IOANA_D12.16S.TCGCGACCAA.GACGCAAGAA"          ,
 "SNOTEL.IOANA_E12.16S.CAGCAGAA.TCAAGAAGAA"            ,
 "SNOTEL.IOANA_E12.16S.CATTGACCAA.TCAAGAAGAA"          ,
 "SNOTEL.IOANA_F12.16S.CAGCAGAA.GGTTGAAGAA"            ,
 "SNOTEL.IOANA_F12.16S.GTCTCGAA.GGTTGAAGAA"            ,
 "SNOTEL.IOANA_H12.16S.ATGGATAA.AGCTAAGA"              ,
 "SNOTEL.IOANA_H12.16S.CCATGGAA.AGCTAAGA")

blanks<-str_split(blanks, pattern = ".16S.", simplify = TRUE)[, 1]


# subset blanks from esv_table & calculate reads
blank_data<-esv_table[, names(esv_table)%in%blanks]
blank_reads<-as.data.frame(colSums(blank_data))
names(blank_reads)<-"reads"
blank_reads$sample<-rownames(blank_reads)
rownames(blank_reads)<-NULL
blank_reads
```

```
##    reads                    sample
## 1     98              Calder.Blank3
## 2      5           Calder.JC_Blank1
## 3  45099           Calder.JC_Blank2
## 4      9           Calder.JC_Blank3
## 5     18          Calder.JC7_Blank1
## 6    146          Calder.JC7_Blank2
## 7     20          Calder.JC7_Blank3
## 8     10          Calder.JC7_Blank4
## 9      5          Calder.JC7_Blank5
## 10    12          Calder.JC7_Blank6
## 11    17          Calder.JC7_Blank7
## 12   902                 MAWC.BLANK
## 13 94586            MAWC.BLANK1_DG1
## 14    75            MAWC.BLANK2_DG2
## 15  1008            MAWC.BLANK3_DG3
## 16   687 MAWC.BLANK4_DG4_BLANK4_DG4
## 17   595           SNOTEL.IOANA_D12
## 18   780           SNOTEL.IOANA_E12
## 19   427           SNOTEL.IOANA_F12
## 20    16           SNOTEL.IOANA_H12
```

```
# number of total reads
sum(colSums(blank_data))
```

```
## [1] 144515
```

```
# number of blanks
length(blank_data)
```

```
## [1] 20
```

```
# summary of blank reads
summary(blank_reads$reads)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     5.0    15.0    86.5  7225.8   710.2 94586.0
```

```
rm(blank_reads)
rm(blank_data)
```

@count_track (samples)

So far we have: - removed samples that should not be included in this study - removed ESVs without reads in the ESV table - excluded 11 blanks - removed ESVs assigned to chloroplast, mitochondria, and kingdom NA - summed technical replicates and removed one sample that was uncorrelated with high reads

```
# remove blanks from esv_table
esv_table_no_blanks<-esv_table[, names(esv_table)%in%setdiff(names(esv_table),blanks)]

# total reads
sum(colSums(esv_table_no_blanks))
```

```
## [1] 68360139
```

```
#average number of reads per sample
sum(colSums(esv_table_no_blanks))/length(esv_table_no_blanks)
```

```
## [1] 107823.6
```

```
#number of ESVs with ESVs with zero reads removed
tmp<-as.data.frame(rowSums(esv_table_no_blanks))
tmp<-tmp[tmp$`rowSums(esv_table_no_blanks)`>0,]
length(tmp)
```

```
## [1] 215778
```

```
# same number as:
length(tmp)==nrow(esv_table_no_blanks) # all TRUE
```

```
## [1] FALSE
```

```
rm(tmp)

# number of samples
ncol(esv_table_no_blanks)
```

```
## [1] 634
```

```
rm(esv_table_no_blanks)
```

4. Decontaminate samples

Identify contaminants using the "decontam" package and the prevalence function

Reference: Davis, N. M., Proctor, D. M., Holmes, S. P., Relman, D. A., & Callahan, B. J. (2018). Simple statistical identification and removal of contaminant sequences in marker-gene and metagenomics data. Microbiome, 6, 1-14.

a. Set up input files and create phyloseq object

```
#make simple metadata table saying which samples are blanks verses true samples
meta_dat<-data.frame(sample_name=colnames(esv_table),Sample_or_Control=rep("True Sample", ncol(esv_table

meta_dat[which(meta_dat$sample_name%in%blanks),2]<-"Control Sample" # specificy which ones are controls
rownames(meta_dat)<-meta_dat$sample_name


# create objects to input into phyloseq
tax_tab <-as.matrix(tax_tab)
taxa_tab <- tax_table(tax_tab)

# make sure there there are no blanks in the taxonomy table
table(taxa_tab@.Data=="") # all false
```

```
##
##  FALSE
## 885797
```

```
#convert metadata and ESV table to phyloseq sub-objects
samp_dat <- sample_data(meta_dat)
esv_tab <- otu_table(esv_table, taxa_are_rows = T)
```

b. Identify contaminants

```
ps <- phyloseq(esv_tab, samp_dat, taxa_tab)
sample_data(ps)$is.neg <- sample_data(ps)$Sample_or_Control == "Control Sample"
contamdf.prev <- isContaminant(ps, method="prevalence", neg="is.neg")

table(contamdf.prev$contaminant)
```

```
##
##  FALSE   TRUE
## 214512   1294
```

```
#what proportion of the total reads are the contaminants?
contamdf.prev$esv<-rownames(contamdf.prev)
contaminants<-contamdf.prev[which(contamdf.prev$contaminant==TRUE),]$esv

# what how many reads of these contaminants are in the sample dataset?
sum(colSums(esv_table[rownames(esv_table)%in%contaminants,]))
```

```
## [1] 47740
```

```
 # proportion of the total reads will be removed
sum(colSums(esv_table[rownames(esv_table)%in%contaminants,]))/sum(colSums(esv_table))
```

```
## [1] 0.000696887
```

```
rm(meta_dat)
rm(ps)
rm(esv_tab)
rm(taxa_tab)
rm(samp_dat)
```

c. Remove contaminants and blanks

```
# remove contaminants from ESV table
esv_table<-esv_table[!rownames(esv_table)%in%contaminants,]

# remove blanks from esv_table
esv_table<-esv_table[, !names(esv_table)%in%blanks] # DOUBLE CHECK THIS!!

rm(blanks)
rm(contaminants)
rm(contamdf.prev)
```

@count_track (samples)

Here I: - removed samples that should not be included in this study - removed ESVs without reads in the ESV table - removed ESVs assigned to chloroplast, mitochondria, and kingdom NA - summed technical replicates and removed one sample that was uncorrelated with high reads - removed contaminant ESVs - removed the blanks

```
# total number of reads
sum(colSums(esv_table))
```

```
## [1] 68315978
```

```
# number of ESVS
nrow(esv_table)
```

```
## [1] 214512
```

```r
#avg. number of reads per sample
sum(colSums(esv_table))/(length(esv_table))
```

```
## [1] 107753.9
```

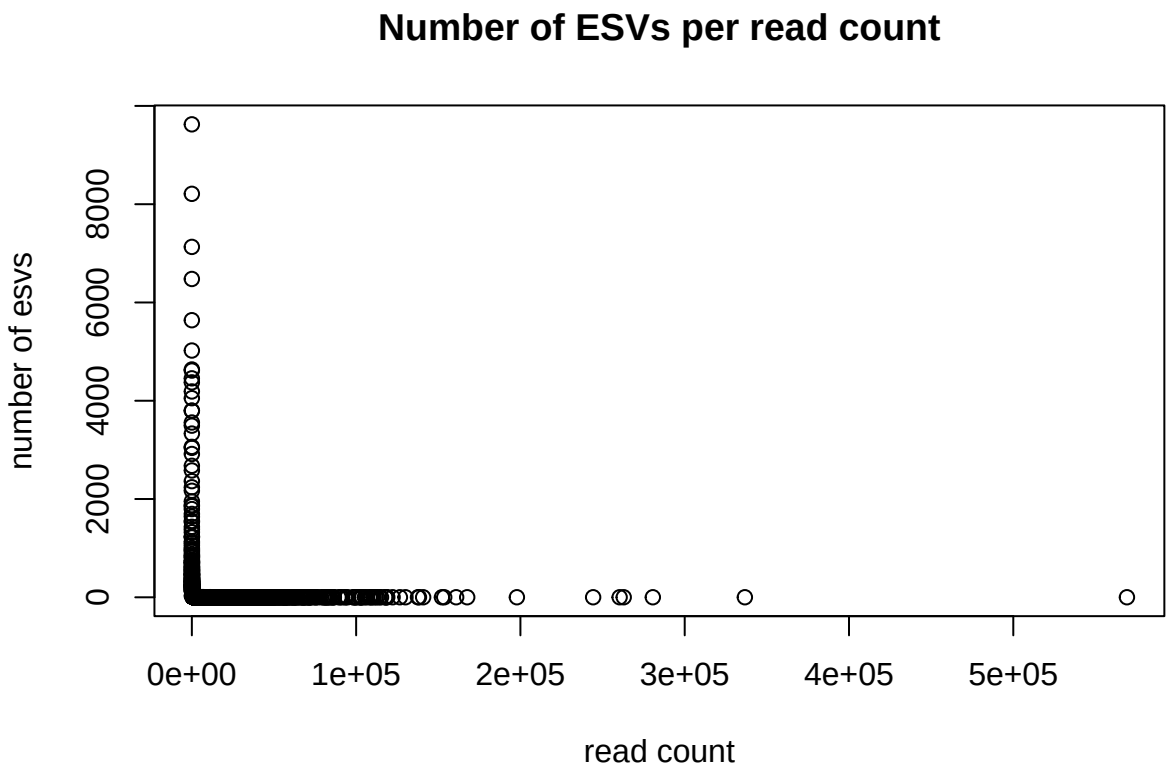
```r
# number of samples, blanks excluded
ncol(esv_table)
```

```
## [1] 634
```

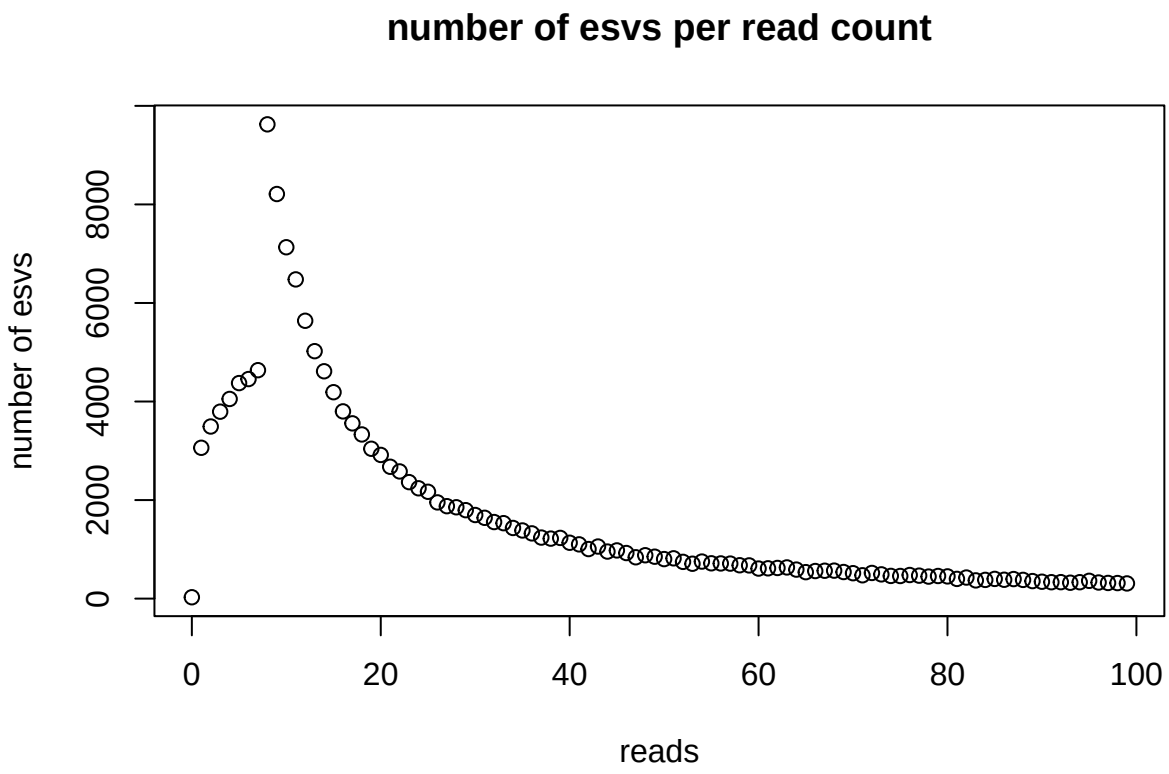
5. Remove ESVs with low reads

a. Plot esvs per read count

```r
reads_per_esv<-as.data.frame(rowSums(esv_table))
names(reads_per_esv)<-"reads"
reads_per_esv$esv<-rownames(reads_per_esv)
rownames(reads_per_esv)<-NULL

#make a table to see the frequency of the reads per ESV
table_reads_per_esv<-as.data.frame(table(reads_per_esv$reads))
colnames(table_reads_per_esv)<-c("reads","freqESVs")
table_reads_per_esv$reads<-as.numeric(as.character(table_reads_per_esv$reads))

#plot all to see that this is just happening within the first 100 reads
plot(table_reads_per_esv$reads,table_reads_per_esv$freqESVs,main="Number of ESVs per read count", xlab=
```




```
#plot the subset of 0-20 reads - but this really starts at 8 because in raw sequence processing we made
plot(table_reads_per_esv$reads[1:100],table_reads_per_esv$freqESVs[1:100],main="number of esvs per read
```




```
table_reads_per_esv[1:25,]
```

```
##    reads freqESVs
## 1      0       28
## 2      1     3061
## 3      2     3493
## 4      3     3794
## 5      4     4052
## 6      5     4374
## 7      6     4456
## 8      7     4636
## 9      8     9625
## 10     9     8211
## 11    10     7131
## 12    11     6478
## 13    12     5639
## 14    13     5022
## 15    14     4614
## 16    15     4189
## 17    16     3800
## 18    17     3557
## 19    18     3332
## 20    19     3041
## 21    20     2916
## 22    21     2676
```

```
## 23    22     2581
## 24    23     2364
## 25    24     2239
```

```
rm(table_reads_per_esv)
rm(reads_per_esv)
```

Citation for removing OTUs with less than reads 10: Urrutia-Cordero, P., Langenheder, S., Striebel, M., Eklöv, P., Angeler, D. G., Bertilsson, S., ... Hillebrand, H. (2021). Functionally reversible impacts of disturbances on lake food webs linked to spatial and seasonal dependencies. Ecology, 102(4). https://doi.org/10.1002/ecy.3283

b. Remove ESVs with less than X reads

```
# output values for unpruned esv table without blanks
reads_per_esv<-as.data.frame(rowSums(esv_table))
names(reads_per_esv)<-"reads"
reads_per_esv$esv<-rownames(reads_per_esv)
rownames(reads_per_esv)<-NULL

# how many reads and ESVs removed if you remove ESVs with less than X reads total
lessthanX<-which(reads_per_esv$reads<10) # JUST CHANGE ONCE HERE
esv_table_lessthanXremoved<-esv_table[lessthanX,]
sum(colSums(esv_table_lessthanXremoved))
```

```
## [1] 269594
```

```
nrow(esv_table_lessthanXremoved)
```

```
## [1] 45730
```

```
# overwrite the table after removing ESVs with less than X reads across the dataset
esv_table<-esv_table[-lessthanX,]
rm(esv_table_lessthanXremoved)
rm(reads_per_esv)
rm(lessthanX)
```

@count_track (samples)

Here I: - removed samples that should not be included in this study - removed ESVs without reads in the ESV table - removed ESVs assigned to chloroplast, mitochondria, and kingdom NA - summed technical replicates and removed one sample that was uncorrelated with high reads - removed contaminant ESVs - removed the blanks - removed ESVs with less than 10 reads across all samples

```
# total number of reads
sum(colSums(esv_table))
```

```
## [1] 68046384
```

```r
# number of ESVS
nrow(esv_table)
```

```
## [1] 168782
```

```r
#avg. number of reads per sample
sum(colSums(esv_table))/(length(esv_table))
```

```
## [1] 107328.7
```

```r
# number of samples, blanks excluded
ncol(esv_table)
```

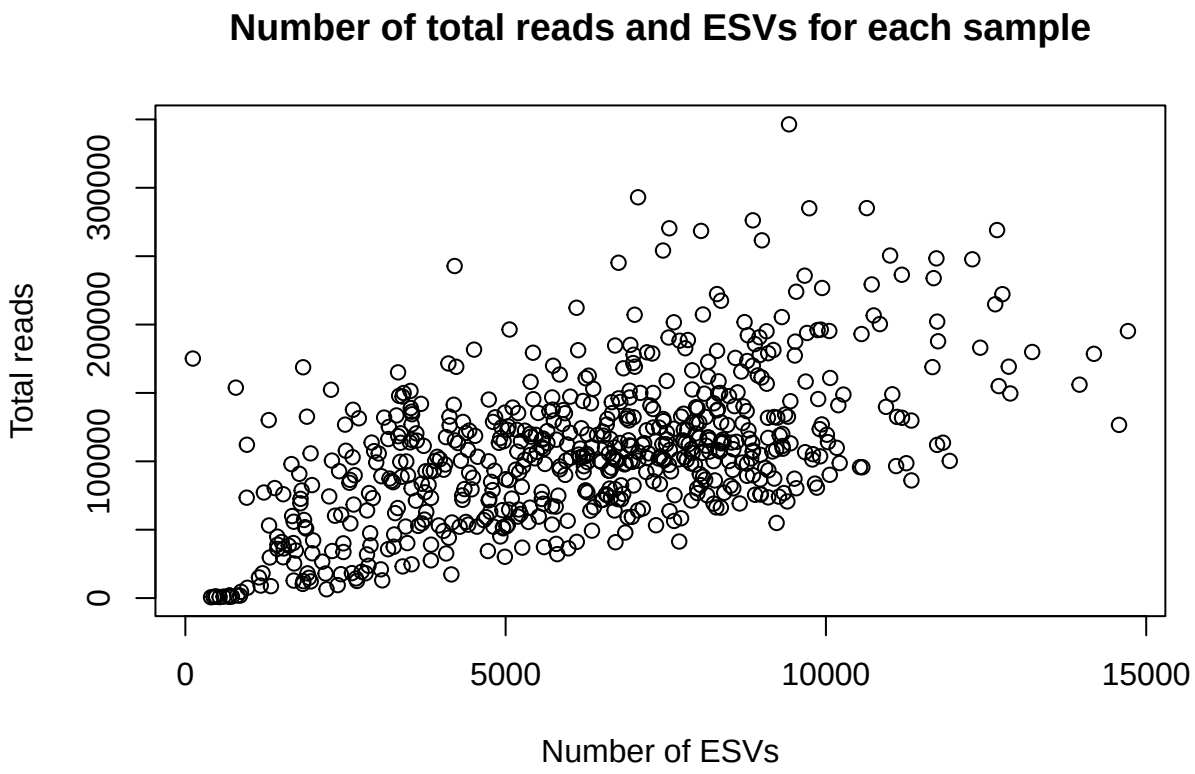
```
## [1] 634
```

c. Plot reads by ESVs for each sample

```r
#calculate number of reads per sample
reads<-as.data.frame(colSums(esv_table))
names(reads)<-"reads"

#calculate number of esvs per sample
samples<-names(esv_table)
table(rownames(reads)==samples) # all TRUE
```

```
##
## TRUE
##  634
```

```r
ESVs<-NULL
for(i in 1:ncol(esv_table)){
        tmp<-as.data.frame(esv_table[,i])
        tmp2<-tmp[tmp>0,]
        ESVs<-c(ESVs, length(tmp2))
}
rm(i)
rm(tmp)
rm(tmp2)
rm(samples)


tmp3<-as.data.frame(cbind(reads$reads,ESVs))
names(tmp3)<-c("reads3","esvs3")
tmp3$sample<-rownames(reads)


plot(ESVs,reads$reads, main="Number of total reads and ESVs for each sample", ylab="Total reads", xlab=
```




```
rm(ESVs)
rm(reads)
rm(tmp3)
```

This figure suggest that I should normalize because the number of ESVs increases with then number of reads in a sample (rather than having equally distributed points with no positive relationship)

7. Write ESV and taxa CSVs before normalization

```
#write.csv(esv_table,file=paste0("ZOTU/",Sys.Date(),"_ZOTUexact_table_notnorm.csv"),row.names = T)
write.csv(tax_tab,file=paste0("ZOTU/",Sys.Date(),"_ZOTUexact_tax_tab_notnorm.csv"),row.names = T)
```

8. Normalize

a. Remove samples with less than 10,000 reads

```
esv_to_normalize<-esv_table
num_read<-as.data.frame(colSums(esv_to_normalize))
names(num_read)<-"reads"
num_read$sample<-rownames(num_read)
rownames(num_read)<-NULL
```

```
# look at how many samples would be removed if the threshold is 10,000
length(which(num_read$reads<10000))
```

```
## [1] 18
```

```
# which lakes are these samples from
table(str_sub(num_read[num_read$reads<10000,]$sample,start = 1, end = 4))
```

```
##
## MAWC SNOT
##   10    8
```

```
# look at how many samples would be removed if the threshold is 5000
length(which(num_read$reads<5000))
```

```
## [1] 13
```

```
esv_to_normalize_10k<-esv_to_normalize[,-which(num_read$reads<10000)]
ncol(esv_to_normalize_10k)
```

```
## [1] 616
```

```
rm(esv_to_normalize)

# check the columns were removed visually
range(num_read[which(num_read$reads<10000),]$reads) # all of these are under 9965
```

```
## [1]  615 9478
```

```
range(as.data.frame(colSums(esv_to_normalize_10k))) # all over 10,000 reads
```

```
## [1]  10378 346383
```

```
rm(num_read)


# check for ESVs with zeros reads again
reads_per_esv<-as.data.frame(rowSums(esv_to_normalize_10k))
names(reads_per_esv)<-"reads"
reads_per_esv$esv<-rownames(reads_per_esv)
rownames(reads_per_esv)<-NULL

# remove ESVs with 0 reads (if there are any)
if(length(which(reads_per_esv$reads<1))>0){
     esv_to_normalize_10k<-esv_to_normalize_10k[which(reads_per_esv$reads>0),]
}

rm(reads_per_esv)
```

b. Normalize

Notes on the rrarefy() function from the help page: Function rrarefy generates one randomly rarefied community data frame or vector of given sample size. The sample can be a vector giving the sample sizes for each row. If the sample size is equal to or smaller than the observed number of individuals, the non-rarefied community will be returned. The random rarefaction is made without replacement so that the variance of rarefied communities is rather related to rarefaction proportion than to the size of the sample.

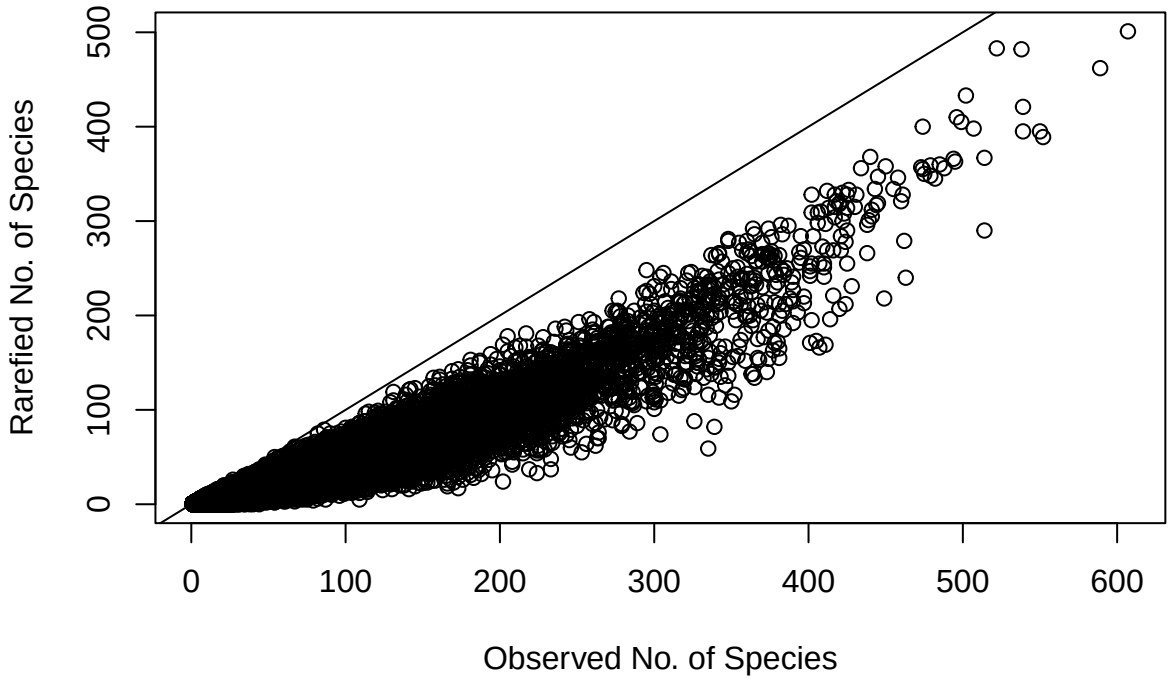
```
S <- specnumber(esv_to_normalize_10k)
#observed number of species
length(S)
```

```
## [1] 168777
```

```
#minimum reads to normalize to
(raremax <- min(rowSums(t(esv_to_normalize_10k))))
```

```
## [1] 10378
```

```
set.seed(061319)
rrare<-rrarefy(t(esv_to_normalize_10k), sample = raremax)
rrare<-as.data.frame(t(rrare))

plot(S, specnumber(rrare), xlab = "Observed No. of Species", ylab = "Rarefied No. of Species")
abline(0, 1)
```



```
num_read<-as.data.frame(colSums(rrare))
names(num_read)<-"reads"
num_read$sample<-rownames(num_read)
rownames(num_read)<-NULL

# count the number of species after rarefaction
esv_counts<-as.data.frame(rowSums(rrare))
names(esv_counts)<-"reads"
esv_counts$ESV<-rownames(esv_counts)
rownames(esv_counts)<-NULL
zeroESV<-which(esv_counts$reads<1)
#ESVS that are zero after rarefying
length(zeroESV)
```

```
## [1] 16832
```

```
# look at the range of the number of reads
range(as.data.frame(rowSums(rrare)))
```

```
## [1]     0 37642
```

```
esv_tab_10knorm<-rrare[-zeroESV,]

#check you removed ESVs with 0 reads.
range(as.data.frame(rowSums(esv_tab_10knorm)))
```

```
## [1]     1 37642
```

```
rm(num_read)
rm(esv_counts)
rm(raremax)
rm(S)
rm(zeroESV)
rm(rrare)
rm(esv_table)

# keep the non-normalized ESV table that is cut off at 10k just in case
#write.csv(esv_to_normalize_10k, paste0("ZOTU/",Sys.Date(),"_ZOTUexact_ESVs_to_normalize_10K_reads.csv"
write.csv(esv_tab_10knorm, paste0("ZOTU/",Sys.Date(),"_ZOTUexact_ESV_tab_10K_reads_normalized.csv"), ro
rm(esv_to_normalize_10k)
```

@ count_track (samples)

For the normalized dataframe just created: - removed samples that should not be included in this study - removed ESVs without reads in the ESV table - removed ESVs assigned to chloroplast, mitochondria, and kingdom NA - summed technical replicates and removed one sample that was uncorrelated with high reads - removed contaminant ESVs - removed the blanks - removed ESVs with less than 10 reads across all samples - normalized reads to ~10K reads

```r
# total number of reads
sum(colSums(esv_tab_10knorm))
```

```
## [1] 6392848
```

```r
# number of ESVS
nrow(esv_tab_10knorm)
```

```
## [1] 151945
```

```r
#avg. number of reads per sample
sum(colSums(esv_tab_10knorm))/(length(esv_tab_10knorm)) # these are all what we rarefied to.
```

```
## [1] 10378
```

```r
# number of samples
ncol(esv_tab_10knorm)
```

```
## [1] 616
```