

R Notebook

Analysis of the Mock Data

Preprocessing Data

```
# rubric for aligning ASV numbers and sequences across datasets  
library("DECIPHER")
```

```
## Loading required package: Biostrings  
## Loading required package: BiocGenerics  
## Loading required package: parallel  
##  
## Attaching package: 'BiocGenerics'  
## The following objects are masked from 'package:parallel':  
##  
##   clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,  
##   clusterExport, clusterMap, parApply, parCapply, parLapply,  
##   parLapplyLB, parRapply, parSapply, parSapplyLB  
## The following objects are masked from 'package:stats':  
##  
##   IQR, mad, sd, var, xtabs  
## The following objects are masked from 'package:base':  
##  
##   anyDuplicated, append, as.data.frame, basename, cbind, 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, sort, table, tapply,  
##   union, unique, unsplit, which, which.max, which.min  
## Loading required package: S4Vectors  
## Loading required package: stats4  
##  
## Attaching package: 'S4Vectors'  
## The following object is masked from 'package:base':  
##  
##   expand.grid  
## Loading required package: IRanges  
## Loading required package: XVector  
##  
## Attaching package: 'Biostrings'  
## The following object is masked from 'package:base':  
##
```

```

##      strsplit
## Loading required package: RSQLite
rubber <- readDNASTringSet("~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/mock_analysis/mock_d

raw_LCA_out_mock_v9_pr2_May20.txt
# svN and taxonomies separated by one comma
v9.mock.LCA.pr2 <- read.csv("~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/mock_analysis/mock_

source("~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/taxonomy_pipeline/helper_fcns/LCA2df.R")

lca.mock <- LCA2df(v9.mock.LCA.pr2, rubber)

v9_mock_bothPrimers_idtax_pr2_0boot
v9.mock.idtax.pr2 <- readRDS("~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/mock_analysis/mock_

source("~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/taxonomy_pipeline/helper_fcns/idtax2df_p

idtax.mock <- idtax2df_pr2(v9.mock.idtax.pr2, boot=50, rubric=rubber)

v9_mock_bothPrimers_bayes_pr2_0boot
v9.mock.bayes.pr2 <- readRDS("~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/mock_analysis/mock_

source("~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/taxonomy_pipeline/helper_fcns/bayestax2d

bayes.mock <- bayestax2df(v9.mock.bayes.pr2, boot=60, rubric=rubber)

Setting table names for standardization
# standardized table names for the pr2
table.names <- c("svN", "ASV", "kingdom", "supergroup", "division", "class", "order", "family", "genus"

colnames(bayes.mock) <- table.names
colnames(idtax.mock) <- table.names

Map the LCA data sets to pr2 using taxmapper.
# getting the taxmapper function
source("~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/taxonomy_pipeline/tax_table_mapping/taxm

# setting up the parameters
synonym.filepath <- "~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/taxonomy_pipeline/tax_table

# Bacteria and Archaea doesn't exist in pr2
nonexistent <- c('Bacteria', 'Archaea')

pr2 <- read.csv("~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/taxonomy_pipeline/tax_table_map
pr2 <- pr2[,-1]

v9.mock.LCA.pr2 <- taxmapper(taxin=lca.mock, tax2map2=pr2,
                             exceptions=nonexistent,
                             synonym.file=synonym.filepath,
                             ignore.format=TRUE)

```

```
lca.mock.mapped <- v9.mock.LCA.pr2[[3]]

lca.mock.sort <- lca.mock.mapped[order(lca.mock.mapped$svN), ]
```

Reading in the Expected

```
exp.mock <- readRDS('~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/mock_analysis/mock_data/v9_r
colnames(exp.mock) <- table.names

make.NA <- function(x) if(is.character(x)||is.factor(x)){
  is.na(x) <- x=="NaN"; x} else {
  x}

exp.mock[] <- lapply(exp.mock, make.NA)
```

Comparisons - how many ASVs are perfect matches to expected, other names agree - mis-classified -> different names at any point - over-classified -> name in tax table, NA in expected - under-classified -> NA in tax table, other names agree

```
all.equal(nrow(bayes.mock), nrow(idtax.mock), nrow(lca.mock.mapped))
```

```
## [1] TRUE
```

```
all.equal(ncol(bayes.mock), ncol(idtax.mock), ncol(lca.mock.mapped))
```

```
## [1] TRUE
```

Implement a function for this comparison This keeps track of the numbers for each row of the ASV

```
compare_results <- function(exp, ref, compare.cols) {
  # create a new data frame with the svN and ASV with the results
  results <- data.frame(matrix(ncol=6,nrow=0, dimnames=list(NULL, c('svN', 'ASV', 'exact', 'mis', 'over

  # iterate through each row
  # iterate from most general column to specific
  nrows <- nrow(exp)
  for (row in 1:nrows) {
    n.same <- 0
    n.mis <- 0
    n.over <- 0
    n.under <- 0
    for (col in compare.cols) {
      exp.tax <- exp[row, col]
      ref.tax <- ref[row, col]
      # over classified
      if (is.na(exp.tax) && !is.na(ref.tax)) {
        n.over <- n.over + 1
      }
      # under classified
      else if (!is.na(exp.tax) && is.na(ref.tax)) {
        n.under <- n.under + 1
      }
      # exact match
      else if ((is.na(exp.tax) && is.na(ref.tax)) || (exp.tax == ref.tax)) {
        n.same <- n.same + 1
      }
      # mis classified
```

```

    else {
      n.mis <- n.mis + 1
    }
  }
  r.row <- data.frame(matrix(rep(NA, 6), ncol=6, nrow = 1, dimnames=list(NULL, names(results))))
  r.row[, 'svN'] <- ref[row, 'svN']
  r.row[, 'ASV'] <- ref[row, 'ASV']
  r.row[, 'exact'] <- n.same
  r.row[, 'mis'] <- n.mis
  r.row[, 'over'] <- n.over
  r.row[, 'under'] <- n.under
  results <- rbind(results, r.row)
}
return(results)
}

```

```

bayes.mock.exp.c1 <- compare_results(exp.mock, bayes.mock, table.names[-c(1:2)])
idtax.mock.exp.c1 <- compare_results(exp.mock, idtax.mock, table.names[-c(1:2)])
lca.mock.exp.c1 <- compare_results(exp.mock, lca.mock.sort, table.names[-c(1:2)])

```

```

plot_results <- function(exp, ref, compare.cols) {
  n.same <- 0
  n.mis <- 0
  n.over <- 0
  n.under <- 0

  # iterate through each row
  # iterate from most general column to specific
  nrows <- nrow(exp)
  for (row in 1:nrows) {
    curr.col <- 0
    for (col in compare.cols) {
      exp.tax <- exp[row, col]
      ref.tax <- ref[row, col]
      # over classified
      if (is.na(exp.tax) && !is.na(ref.tax)) {
        n.over <- n.over + 1
        break
      }
      # under classified
      else if (!is.na(exp.tax) && is.na(ref.tax)) {
        n.under <- n.under + 1
        break
      }
      # mis classified
      else if (!is.na(exp.tax) && !is.na(ref.tax) && (exp.tax != ref.tax)) {
        n.mis <- n.mis + 1
        break
      }
      # same
      else {
        curr.col <- curr.col + 1
      }
    }
  }
}

```

```

    if (curr.col == length(compare.cols)) {
      n.same <- n.same + 1
    }
  }
  return(c(n.same, n.mis, n.over, n.under))
}

bayes.mock.exp.c2 <- plot_results(exp.mock, bayes.mock, table.names[-c(1:2)])
idtax.mock.exp.c2 <- plot_results(exp.mock, idtax.mock, table.names[-c(1:2)])
lca.mock.exp.c2 <- plot_results(exp.mock, lca.mock.sort, table.names[-c(1:2)])

# compute number of exact matches
nrow(bayes.mock.exp.c1[which(bayes.mock.exp.c1$exact == 8), ]) == bayes.mock.exp.c2[1]

## [1] TRUE
nrow(idtax.mock.exp.c1[which(idtax.mock.exp.c1$exact == 8), ]) == idtax.mock.exp.c2[1]

## [1] TRUE
nrow(lca.mock.exp.c1[which(lca.mock.exp.c1$exact == 8), ]) == lca.mock.exp.c2[1]

## [1] TRUE
# compute number of mismatches
nrow(bayes.mock.exp.c1[which(bayes.mock.exp.c1$mis > 0), ]) == bayes.mock.exp.c2[2]

## [1] TRUE
nrow(idtax.mock.exp.c1[which(idtax.mock.exp.c1$mis > 0), ]) == idtax.mock.exp.c2[2]

## [1] TRUE
nrow(lca.mock.exp.c1[which(lca.mock.exp.c1$mis > 0), ]) == lca.mock.exp.c2[2]

## [1] TRUE
# compute number of over
nrow(bayes.mock.exp.c1[which(bayes.mock.exp.c1$over > 0 & bayes.mock.exp.c1$mis == 0), ]) == bayes.mock.exp.c2[3]

## [1] TRUE
nrow(idtax.mock.exp.c1[which(idtax.mock.exp.c1$over > 0 & idtax.mock.exp.c1$mis == 0), ]) == idtax.mock.exp.c2[3]

## [1] TRUE
nrow(lca.mock.exp.c1[which(lca.mock.exp.c1$over > 0 & lca.mock.exp.c1$mis == 0), ]) == lca.mock.exp.c2[3]

## [1] TRUE
# compute number of under
nrow(bayes.mock.exp.c1[which(bayes.mock.exp.c1$under > 0 & bayes.mock.exp.c1$mis == 0), ]) == bayes.mock.exp.c2[4]

## [1] TRUE
nrow(idtax.mock.exp.c1[which(idtax.mock.exp.c1$under > 0 & idtax.mock.exp.c1$mis == 0), ]) == idtax.mock.exp.c2[4]

## [1] TRUE
nrow(lca.mock.exp.c1[which(lca.mock.exp.c1$under > 0 & lca.mock.exp.c1$mis == 0), ]) == lca.mock.exp.c2[4]

## [1] TRUE

```

Consensus Tax

```
source("~/Desktop/Taxonomic Sequencing/amplicon_bioinformatics/taxonomy_pipeline/consensus_taxonomies/c

tblnam <- c("bayes-pr2", "idtax-pr2", "lca-pr2")

all.c <- consensus_tax_mostCom2(bayes.mock, idtax.mock, lca.mock.mapped,
                                tablenames=tblnam, ranknamez=table.names,
                                tiebreakz="none", count.na=TRUE, weights=c(1,1,1))

all.c.compare <- plot_results(exp.mock, all.c, table.names[-c(1:2)])
```

Create a quad bar plot

```
# create a data frame
vals <- rbind(bayes.mock.exp.c2, idtax.mock.exp.c2, lca.mock.exp.c2, all.c.compare)
vals
```

```
##           [,1] [,2] [,3] [,4]
## bayes.mock.exp.c2 5856  865 1247 2197
## idtax.mock.exp.c2 8580    5  551 1029
## lca.mock.exp.c2   37 8137  160 1831
## all.c.compare    6456    1  311 3397
```

```
data.frame(vals, row.names=c('exact', 'mis', 'over', 'under'))
```

```
##           X1  X2  X3  X4
## exact 5856  865 1247 2197
## mis  8580    5  551 1029
## over   37 8137  160 1831
## under 6456    1  311 3397
```

```
df <- data.frame(matrix(vals, ncol=4, nrow = 4, dimnames=list(c('bayes', 'idtax', 'lca', 'consensus'), 
df
```

```
##           exact  mis over under
## bayes      5856  865 1247 2197
## idtax      8580    5  551 1029
## lca         37 8137  160 1831
## consensus 6456    1  311 3397
```

```
table <- c(rep('bayes', 4), rep('idtax', 4), rep('lca', 4), rep('consensus', 4))
class <- c(rep(c('exact', 'mis', 'over', 'under'), 4))
val <- c(bayes.mock.exp.c2, idtax.mock.exp.c2, lca.mock.exp.c2, all.c.compare)
```

```
data <- data.frame(table, class, val)
data
```

```
##           table class  val
## 1         bayes exact 5856
## 2         bayes  mis  865
## 3         bayes over 1247
## 4         bayes under 2197
## 5         idtax exact 8580
## 6         idtax  mis    5
## 7         idtax over  551
## 8         idtax under 1029
## 9          lca exact   37
```

```
## 10      lca    mis 8137
## 11      lca  over  160
## 12      lca under 1831
## 13 consensus exact 6456
## 14 consensus  mis    1
## 15 consensus  over  311
## 16 consensus under 3397
```

```
library(ggplot2)
ggplot(data, aes(fill=class, y=val, x=table)) +
  geom_bar(position=position_dodge(width=0.8), stat='identity') +
  labs(x="taxonomy table", y='count') +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 12), axis.title.x = element_text(size = 12),
        axis.text.y = element_text(size = 12), axis.title.y = element_text(size = 12, face="bold"),
        panel.background = element_rect(fill = "white",
                                          colour = "white",
                                          linetype = "solid"),
        panel.grid.major = element_line(size = 0.5, linetype = 'solid',
                                          colour = "white"),
        panel.grid.minor = element_line(size = 0.25, linetype = 'solid',
                                          colour = "white"),
        axis.line = element_line(size = 0.5, linetype = "solid", colour = "black")) +
  scale_fill_discrete(name = "Rank")
```

