```
#install.packages('rhmmer')
#install.packages("pheatmap")
#install.packages("RColorBrewer")
#install.packages("seqinr")
#install.packages("stringr")

# Load rhmmer for parsing HMMER results.
library(rhmmer)

# Import pheatmap library for making nice heatmaps.
library("pheatmap")
library("RColorBrewer")

# Import seqinr for editing fasta files.
library(seqinr)

# Import stringr for matching strings.
library(stringr)

# Remove all objects.
rm(list = ls())

# For getting all the google colours.
load("/home/kim_lab_kvm/Documents/Species_Phylogeny/google_colours.RData"
)

# Set the working directory.
setwd("/home/kim_lab_kvm/Documents/Species_Phylogeny/bacterial/")

# Set location of programs to run.
hmmsearch <- "/usr/local/bin/hmmsearch"
esl.sfetch <- "/usr/local/bin/esl-sfetch"


# Experiment - always use refseq rather than genbank data. The latter
contains unfinished genomes, which are more complicated to analyse.
# Read in experiment from command line arguments
args = commandArgs(trailingOnly=TRUE)
if (length(args)==0) {
  stop("At least one argument must be supplied (experiment).n",
call.=FALSE)
} else if (length(args)==1) {
  experiment <- args[1]
}

# For getting all the google colours.
load(paste("/home/kim_lab_kvm/Documents/Species_Phylogeny/bacterial/data/
", experiment, "/genomes.RData", sep = ""))


## Load housekeeping gene hidden markov models for HMMER.

# Get the file names of the hmms from the bac120 set.
```

```
files.bac120.hmm <- list.files(path = "databases/bac120/", pattern =
".hmm")

# Store number of hmms.
num.bac120.hmm <- length(files.bac120.hmm)

# Create an empty data.frame for the hmm information.
data.bac120.hmm.info <- data.frame(hmmer3.f = character(),
                                   name = character(),
                                   accession = character(),
                                   description = numeric(),
                                   length = character(),
                                   stringsAsFactors = FALSE)


# Extract the bac120 hmm information.
for(i in 1:num.bac120.hmm){

  # Read in the header of the assembly statistics file.
  file.bac120.hmm.head <- readLines(paste("databases/bac120/",
files.bac120.hmm[i], sep = ""))[1:5]

  # Input the data into data.assembly.info.
  data.bac120.hmm.info[i, ]$hmmer3.f <- trimws(gsub(pattern = "HMMER3/f",
"", paste("", file.bac120.hmm.head[grep(pattern = "HMMER3/f", x =
file.bac120.hmm.head)], sep = "")))
  data.bac120.hmm.info[i, ]$name <- trimws(gsub(pattern = "NAME", "",
paste("", file.bac120.hmm.head[grep(pattern = "NAME", x =
file.bac120.hmm.head)], sep = "")))
  data.bac120.hmm.info[i, ]$accession <- trimws(gsub(pattern = "ACC", "",
paste("", file.bac120.hmm.head[grep(pattern = "ACC", x =
file.bac120.hmm.head)], sep = "")))
  data.bac120.hmm.info[i, ]$description <- trimws(gsub(pattern = "DESC",
"", paste("", file.bac120.hmm.head[grep(pattern = "DESC", x =
file.bac120.hmm.head)], sep = "")))
  data.bac120.hmm.info[i, ]$length <- trimws(gsub(pattern = "LENG", "",
paste("", file.bac120.hmm.head[grep(pattern = "LENG", x =
file.bac120.hmm.head)], sep = "")))

}

# Number the hmms.
rownames(data.bac120.hmm.info) <- 1:num.bac120.hmm

# Write to tsv file.
write.table(data.bac120.hmm.info, file = paste("data/", experiment,
"/data_bac120_hmm_info.tsv", sep = ""), quote = FALSE, sep = "\t",
col.names = NA)

# Create data frame to store hits.
data.bac120.hmm.hits <- data.frame(assembly.name = character(),
                                   domain_name = character(),
                                   domain_accession = character(),
                                   query_name = character(),
```

```
                                    query_accession = character(),
                                    sequence_evalue = numeric(),
                                    sequence_score = numeric(),
                                    sequence_bias = numeric(),
                                    best_domain_evalue = numeric(),
                                    best_domain_score = numeric(),
                                    best_domain_bis = numeric(),
                                    domain_number_exp = numeric(),
                                    domain_number_reg = integer(),
                                    domain_number_clu = integer(),
                                    domain_number_ov = integer(),
                                    domain_number_env = integer(),
                                    domain_number_dom = integer(),
                                    domain_number_rep = integer(),
                                    domain_number_inc = character(),
                                    description = character(),
                                    stringsAsFactors = FALSE)

# Create data frame to summarise hits.
data.bac120.hmm.hits.summary <- matrix(0, nrow = num.bac120.hmm, ncol =
num.genomes)
rownames(data.bac120.hmm.hits.summary) <-
paste(data.bac120.hmm.info$accession, " ",
data.bac120.hmm.info$description)
colnames(data.bac120.hmm.hits.summary) <-
paste(data.assembly.info$chromosome.sequence.refseq.accession, " ",
data.assembly.info$species.strain.isolate, sep = "")

# Remove old folder and create new folder for hmmsearch against bac120
sequences.
command <- paste("rm -r data/", experiment,
"/hmmsearch_bac120_output_from_translated_cds/ 2> /dev/null", sep = "")
system(command, intern = FALSE)
command <- paste("mkdir -p data/", experiment,
"/hmmsearch_bac120_output_from_translated_cds/", sep = "")
system(command, intern = FALSE)

# Use hmmer to get all the fasta sequences.
for(i in 1:num.genomes){ # 1:num.genomes

  # Print which genome is being processed.
  cat(paste("Processing genome ", i, ": ",
data.assembly.info$refseq.genome.id[i], "\n", sep = ""))

  for(j in 1:num.bac120.hmm){ # 1:num.bac120.hmm

    # Create new folder for hmmsearch against bac120 sequences.
    command <- paste("mkdir -p data/", experiment,
"/hmmsearch_bac120_output_from_translated_cds/",
data.bac120.hmm.info$accession[j], sep = "")
    system(command, intern = TRUE)

    # Run hmmsearch to match the bac120 genes.
```

```
    file.tbl <- paste("data/", experiment,
"/hmmsearch_bac120_output_from_translated_cds/",
data.bac120.hmm.info$accession[j], "/",
data.assembly.info$refseq.genome.id[i], "_hmmsearch_bac120_output_from_",
data.bac120.hmm.info$accession[j], ".tbl", sep = "")

    if(!file.exists(file.tbl)){

      command <- paste(hmmsearch, " -E 1e-10 --incE 1e-10 --tblout ",
file.tbl, " databases/bac120/", files.bac120.hmm[j], " data/genomes/",
data.assembly.info$refseq.genome.id[i], "/",
data.assembly.info$file.translated.cds[i], sep = "")
      system(command, intern = TRUE)

      # Add information to data frame.
      tbl <- read_tblout(file.tbl)
      num.hits <- dim(tbl)[1]
      tbl <- cbind(rep(data.assembly.info$assembly.name[i], times =
num.hits), tbl)
      names(tbl)[1] <- "assembly.name"
      data.bac120.hmm.hits <- rbind(data.bac120.hmm.hits, tbl)

      # Store the number of hits to each gene.
      data.bac120.hmm.hits.summary[j, i] <- dim(tbl)[1]

    }

    # Cds fasta file.
    file.fasta.cds <- paste("data/", experiment,
"/hmmsearch_bac120_output_from_translated_cds/",
data.bac120.hmm.info$accession[j], "/",
data.assembly.info$refseq.genome.id[i], "_hmmsearch_bac120_output_from_",
data.bac120.hmm.info$accession[j], ".fna", sep = "")

    # Translated cds fasta file.
    file.fasta.translated.cds <- paste("data/", experiment,
"/hmmsearch_bac120_output_from_translated_cds/",
data.bac120.hmm.info$accession[j], "/",
data.assembly.info$refseq.genome.id[i], "_hmmsearch_bac120_output_from_",
data.bac120.hmm.info$accession[j], ".faa", sep = "")

    if(!file.exists(file.fasta.cds) ||
!file.exists(file.fasta.translated.cds)){

      if(num.hits >= 1){ # Save the top hit fasta.

        # Find the top hmmer hit.
        command <- paste("grep -v '^#' ", file.tbl, " | awk 'NR==1{print
$1}'", sep = "")
        top.translated.cds.hit <- system(command = command, intern =
TRUE)
        top.cds.hit <- gsub("prot", "cds", top.translated.cds.hit)

        # Get using esl.sfetch index file.
```

```
        command <- paste(esl.sfetch, " data/genomes/",
data.assembly.info$refseq.genome.id[i], "/",
data.assembly.info$file.cds[i], " '", top.cds.hit, "' > ",
file.fasta.cds, sep = "") # Approach for getting the top hit (reccomended
by HMMER).
        system(command, intern = FALSE)

        # Get using esl.sfetch index file.
        command <- paste(esl.sfetch, " data/genomes/",
data.assembly.info$refseq.genome.id[i], "/",
data.assembly.info$file.translated.cds[i], " '", top.translated.cds.hit,
"' > ", file.fasta.translated.cds, sep = "") # Approach for getting the
top hit (reccomended by HMMER).
        system(command, intern = FALSE)

      } else{ # Create a fasta with an empty string - this helps with
concatenation later.

          # Make empty cds fasta.
          fasta.header <- paste("lcl|", data.assembly.info[i,
]$chromosome.sequence.refseq.accession, "_cds_NA [gene=NA] [gbkey=CDS]",
sep = "")
          fasta.string <- paste(c("ATG", rep("-", times = 99 * 3)),
collapse = "")
          write.fasta(sequences = as.list(fasta.string), names =
fasta.header, file.out = file.fasta.cds, as.string = TRUE)

          # Make empty translated cds fasta.
          fasta.header <- paste("lcl|", data.assembly.info[i,
]$chromosome.sequence.refseq.accession, "_translated.cds_NA [gene=NA]
[gbkey=CDS]", sep = "")
          fasta.string <- paste(c("M", rep("-", times = 99)), collapse =
"")
          write.fasta(sequences = as.list(fasta.string), names =
fasta.header, file.out = file.fasta.translated.cds, as.string = TRUE)

      }
    }

  }

}

# Write to tsv file.
file.bac120.hmm.hits <- paste("data/", experiment,
"/data_bac120_hmm_hits.tsv", sep = "")
write.table(data.bac120.hmm.hits, file = file.bac120.hmm.hits, quote =
FALSE, sep = "\t", col.names = NA)

## Heatmap of housekeeping copy number.

# Remove old folder and create new folder for hmmsearch against bac120
sequences.
command <- paste("rm -r figures/", experiment, " 2> /dev/null", sep = "")
```

```
system(command, intern = FALSE)
command <- paste("mkdir -p figures/", experiment, "", sep = "")
system(command, intern = FALSE)

# Specify the number of colours to use and the pallette they are taken
from.
colour.ge <- c(google.white, google.blue.500, google.red.500)

# List of breaks for colours
breaks <- c(0, 0.99, 1.99, max(data.bac120.hmm.hits.summary))

# Draw heatmaps
f.heatmap <- pheatmap(data.bac120.hmm.hits.summary,
                      breaks = breaks,
                      color = colour.ge,
                      border_color = "white",
                      treeheight_row = 50,
                      treeheight_col = 50,
                      cellwidth = 4,
                      cellheight = 4,
                      fontsize = 4,
                      fontsize_row = 4,
                      fontsize_col = 4,
                      clustering_method = "ward.D", # From: "ward.D",
"ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (=
WPGMA), "median" (= WPGMC), "centroid" (= UPGMC)
                      cluster_rows = TRUE,
                      clustering_distance_rows = "euclidean", # From:
'correlation', 'euclidean', 'maximum', 'manhattan', 'canberra', 'binary',
'minkowski'
                      cutree_rows = TRUE,
                      # gaps_rows = c(length(idx.gcb), length(idx.gcb) +
length(idx.unc)),
                      cluster_cols = TRUE,
                      clustering_distance_cols = "euclidean", # From:
'correlation', 'euclidean', 'maximum', 'manhattan', 'canberra', 'binary',
'minkowski'
                      legend = FALSE,
                      filename = file.path(paste("figures/", experiment,
"/heatmap_housekeeping_hits.pdf", sep = "")))

# Save the workspace so it can be loaded from start without having to
repeat all these steps.
file.rdata <- paste("data/", experiment,
"/housekeeping_fasta_from_hmm.RData", sep = "")
save(list = ls(all.names = TRUE), file = file.rdata)
```