```
# Load seqinr to manipulate fasta files.
library(seqinr)

# 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.
esl.sfetch <- "/usr/local/bin/esl-sfetch"
makeblastdb <- "/home/kim_lab_kvm/miniconda3/envs/py3/bin/makeblastdb"

# 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]
}

#experiment <- experiment_name

## Annotation information.

# Get the file names of the genomes to analyse.
files.assembly.stats <- list.files(path = paste("data/", experiment,
"/assembly_stats/", sep = ""), pattern = "assembly_stats.txt$")

# Store number of genomes.
num.genomes <- length(files.assembly.stats)

# Create an empty data.frame for the assembly information.
data.assembly.info <- data.frame(refseq.genome.id = character(),
                                 assembly.name = character(),
                                 organism.name = character(),
                                 infraspecific.name = character(),
                                 taxid = numeric(),
                                 biosample = character(),
                                 bioproject = character(),
                                 submitter = character(),
                                 date = as.Date(character()),
                                 assembly.type = character(),
                                 release.type = character(),
                                 assembly.level = character(),
                                 genome.representation = character(),
```

```
                                relation.to.type.material = character(),
                                refseq.category = character(),
                                refseq.assembly.accession = character(),
                                species = character(),
                                strain = character(),
                                isolate = character(),
                                species.strain.isolate = character(),
                                group = character(),
                                file.assembly.report = character(),
                                file.feature.table = character(),
                                file.cds = character(),
                                file.translated.cds = character(),
                                file.rna = character(),
                                file.genomic.gff = character(),
                                num.cds = numeric(),
                                num.translated.cds = numeric(),
                                num.rna = numeric(),
                                num.cds.codon.errors = numeric(),
                                num.cds.translation.errors = numeric(),
                                chromosome.sequence.name = character(),
                                chromosome.sequence.role = character(),
                                chromosome.assigned.molecule =
character(),
                                chromosome.sequence.location.type =
character(),
                                chromosome.sequence.genbank.accession =
character(),
                                chromosome.sequence.relationship =
character(),
                                chromosome.sequence.refseq.accession =
character(),
                                chromosome.sequence.assembly.unit =
character(),
                                chromosome.sequence.length = numeric(),
                                chromosome.sequence.ucsc.style.name =
character(),
                                num.plasmids = numeric(),
                                plasmid.sequence.name = character(),
                                plasmid.sequence.role = character(),
                                plasmid.assigned.molecule = character(),
                                plasmid.sequence.location.type =
character(),
                                plasmid.sequence.genbank.accession =
character(),
                                plasmid.sequence.relationship =
character(),
                                plasmid.sequence.refseq.accession =
character(),
                                plasmid.sequence.assembly.unit =
character(),
                                plasmid.sequence.length = numeric(),
                                plasmid.sequence.ucsc.style.name =
character(),
                                stringsAsFactors = FALSE)
```

```
# A list of hard coded groups the NCBI uses to classify genotypes.
groups <- c("\\(bacteria\\)",
            "\\(a-proteobacteria\\)",
            "\\(b-proteobacteria\\)",
            "\\(g-proteobacteria\\)",
            "\\(d-proteobacteria\\)",
            "\\(e-proteobacteria\\)",
            "\\(high GC Gram\\+\\)",
            "\\(enterobacteria\\)",
            "\\(E. coli\\)",
            "\\(planctomycetes\\)",
            "\\(firmicutes\\)",
            "\\(GNS bacteria\\)",
            "\\(cyanobacteria\\)",
            "\\(thermotogales\\)",
            "\\(spirochetes\\)",
            "\\(mycoplasmas\\)",
            "\\(fusobacteria\\)",
            "\\(green sulfur bacteria\\)",
            "\\(CFB group bacteria\\)",
            "\\(chlamydias\\)",
            "\\(aquificales\\)")

# Create new folder if it doesn't already exist.
command <- paste("mkdir -p data/genomes/", sep = "")
system(command, intern = TRUE)

# Get the genomes from the NCBI.
for(i in 1:num.genomes){

  # Read in the header of the assembly statistics file.
  file.assembly.stats.head <- readLines(paste("data/", experiment,
"/assembly_stats/", files.assembly.stats[i], sep = ""))[2:40]

  # Get the refseq assembly accession.
  refseq.assembly.accession <- trimws(gsub(pattern = "# RefSeq assembly
accession:", "", paste("", file.assembly.stats.head[grep(pattern = "#
RefSeq assembly accession:", x = file.assembly.stats.head)], sep = "")))

  # Get the refseq numbers needed to ftp genome files.
  refseq.number <- gsub(".*_|\\..*", "", refseq.assembly.accession)
  refseq.number.1 <- substr(refseq.number, 1, 3)
  refseq.number.2 <- substr(refseq.number, 4, 6)
  refseq.number.3 <- substr(refseq.number, 7, 9)

  # Check if a directory already exists for this genome, and if not,
create on.
  command <- paste("ls data/genomes/ 2> /dev/null", sep = "")
  directory.list <- system(command = command, intern = TRUE, wait = TRUE)
  num.matches <- length(grep(pattern = refseq.assembly.accession, x =
directory.list))

  if(num.matches == 1){
```

```
    # Get the refseq assemly id.
    refseq.genome.id <- directory.list[grep(pattern =
refseq.assembly.accession, x = directory.list)]

  } else {

    # Location of genome data.
    ftp.site <- paste("ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/",
refseq.number.1, "/", refseq.number.2, "/", refseq.number.3, "/", sep =
"")

    # Download the assembly report file from NCBI.
    command <- paste("wget -r -np -nd -A '*_assembly_report.txt' '",
ftp.site, "'", sep = "")
    system(command = command, wait = TRUE)

    # Get the refseq assemly id (I know, there should be a better way to
get this, but inconsistencies in NCBI FTP site naming conventions means
this is probably the best way.)
    command <- paste("ls | grep '", refseq.assembly.accession, "'", sep =
"")
    refseq.genome.id <- gsub("_assembly_report.txt", "", system(command =
command, intern = TRUE))

    # Refseq genome identifier.
    assembly.name <- gsub(paste(refseq.assembly.accession, "_", sep =
""), "", refseq.genome.id)

    # Remove the assembly report.
    command <- paste("rm *", refseq.number, "*", sep = "")
    system(command = command, wait = TRUE)

  }

  # Print which genome is being processed.
  cat(paste("Processing genome ", i, ": ", refseq.genome.id, "\n", sep =
""))

  # Location of genome data.
  ftp.site <- paste("ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/",
refseq.number.1, "/", refseq.number.2, "/", refseq.number.3, "/",
refseq.genome.id, sep = "")

  # Get the assembly report file.
  file.assembly.report <- paste("data/genomes/", refseq.genome.id, "/",
refseq.genome.id, "_assembly_report.txt", sep = "")

  if(file.exists(file.assembly.report)){

    # Add file location to assebmly info.
    data.assembly.info[i, ]$file.assembly.report <-
paste(refseq.genome.id, "_assembly_report.txt", sep = "")
```

```
    # Read in the header of the assembly statistics file.
    file.assembly.report.head <- readLines(paste("data/genomes/",
refseq.genome.id, "/", refseq.genome.id, "_assembly_report.txt", sep =
""))

    # Input the data into data.assembly.info.
    data.assembly.info[i, ]$refseq.genome.id <- refseq.genome.id
    data.assembly.info[i, ]$assembly.name <- gsub(" ", "_",
trimws(gsub(pattern = "# Assembly name:", "", paste("",
file.assembly.stats.head[grep(pattern = "# Assembly name:", x =
file.assembly.stats.head)], sep = ""))))
    data.assembly.info[i, ]$organism.name <- trimws(gsub(pattern = "#
Organism name:", "", paste("", file.assembly.report.head[grep(pattern =
"# Organism name:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$infraspecific.name <- trimws(gsub(pattern =
"# Infraspecific name:", "", paste("",
file.assembly.report.head[grep(pattern = "# Infraspecific name:", x =
file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$taxid <- as.numeric(trimws(gsub(pattern = "#
Taxid:", "", paste("", file.assembly.report.head[grep(pattern = "#
Taxid:", x = file.assembly.report.head)], sep = ""))))
    data.assembly.info[i, ]$biosample <- trimws(gsub(pattern = "#
BioSample:", "", paste("", file.assembly.report.head[grep(pattern = "#
BioSample:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$bioproject <- trimws(gsub(pattern = "#
BioProject:", "", paste("", file.assembly.report.head[grep(pattern = "#
BioProject:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$submitter <- trimws(gsub(pattern = "#
Submitter:", "", paste("", file.assembly.report.head[grep(pattern = "#
Submitter:", x = file.assembly.report.head)], sep = "")))

    if(!is.na(match("# Date", file.assembly.report.head))){
      data.assembly.info[i, ]$date <- as.Date(trimws(gsub(pattern = "#
Date:", "", paste("", file.assembly.report.head[grep(pattern = "# Date:",
x = file.assembly.report.head)], sep = ""))))
    }

    data.assembly.info[i, ]$assembly.type <- trimws(gsub(pattern = "#
Assembly type:", "", paste("", file.assembly.report.head[grep(pattern =
"# Assembly type:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$release.type <- trimws(gsub(pattern = "#
Release type:", "", paste("", file.assembly.report.head[grep(pattern = "#
Release type:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$assembly.level <- trimws(gsub(pattern = "#
Assembly level:", "", paste("", file.assembly.report.head[grep(pattern =
"# Assembly level:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$genome.representation <- trimws(gsub(pattern
= "# Genome representation:", "", paste("",
file.assembly.report.head[grep(pattern = "# Genome representation:", x =
file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$relation.to.type.material <-
trimws(gsub(pattern = "# Relation to type material:", "", paste("",
file.assembly.report.head[grep(pattern = "# Relation to type material:",
x = file.assembly.report.head)], sep = "")))
```

```
    data.assembly.info[i, ]$refseq.category <- trimws(gsub(pattern = "#
RefSeq category:", "", paste("", file.assembly.report.head[grep(pattern =
"# RefSeq category:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$refseq.assembly.accession <-
trimws(gsub(pattern = "# RefSeq assembly accession:", "", paste("",
file.assembly.report.head[grep(pattern = "# RefSeq assembly accession:",
x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$species <- trimws(gsub(pattern =
paste(groups, collapse = "|"), "", x = data.assembly.info[i,
]$organism.name)) # First remove the groups.
    data.assembly.info[i, ]$species <- trimws(gsub(pattern = ":", " ", x
= data.assembly.info[i, ]$species)) # Then replace problematic
characters.
    data.assembly.info[i, ]$species <- trimws(gsub(pattern = ",", " ", x
= data.assembly.info[i, ]$species))
    data.assembly.info[i, ]$species <- trimws(gsub(pattern = "\\(",
"\\[", x = data.assembly.info[i, ]$species))
    data.assembly.info[i, ]$species <- trimws(gsub(pattern = "\\)",
"\\]", x = data.assembly.info[i, ]$species))
    data.assembly.info[i, ]$strain <- trimws(gsub(pattern = "strain=",
"", paste("", data.assembly.info[i, ]$infraspecific.name, sep = "")))
    data.assembly.info[i, ]$isolate <- trimws(gsub(pattern = "#
Isolate:", "", paste("", file.assembly.report.head[grep(pattern = "#
Isolate:", x = file.assembly.report.head)], sep = "")))

    data.assembly.info[i, ]$species.strain.isolate <-
trimws(paste(trimws(gsub(data.assembly.info[i, ]$strain, """,
data.assembly.info[i, ]$species)), " ", trimws(data.assembly.info[i,
]$strain), sep = ""))

    # If the strain isn't defined, use the isolate name.
    if(nchar(data.assembly.info[i, ]$strain) == 0){
      data.assembly.info[i, ]$species.strain.isolate <-
trimws(paste(trimws(gsub(data.assembly.info[i, ]$isolate, "",
data.assembly.info[i, ]$species)), " ", trimws(data.assembly.info[i,
]$isolate), sep = ""))
    }

    data.assembly.info[i, ]$group <- trimws(gsub("\\)", "",
unlist(strsplit(data.assembly.info[i, ]$organism.name,
"\\("))[length(unlist(strsplit(data.assembly.info[i, ]$organism.name,
"\\(")))]))

    # Get information about structure of assembly in terms of chrosomoes
and plasmids.
    structure.start <- grep("# Sequence-Name", file.assembly.report.head)
+ 1
    structure.end <- length(file.assembly.report.head)
    data.structure <- read.table(text =
file.assembly.report.head[structure.start:structure.end], sep = "\t",
stringsAsFactors = FALSE)
    colnames(data.structure) <- c("sequence.name", "sequence.role",
"assigned.molecule", "sequence.location.type",
"sequence.genbank.accession", "sequence.relationship",
```

```
"sequence.refseq.accession", "sequence.assembly.unit", "sequence.length",
"sequence.ucsc.style.name")

    # Store information about primary chromosome.
    data.assembly.info[i, ]$chromosome.sequence.name <- data.structure[1,
]$sequence.name
    data.assembly.info[i, ]$chromosome.sequence.role <- data.structure[1,
]$sequence.role
    data.assembly.info[i, ]$chromosome.assigned.molecule <-
data.structure[1, ]$assigned.molecule
    data.assembly.info[i, ]$chromosome.sequence.location.type <-
data.structure[1, ]$sequence.location.type
    data.assembly.info[i, ]$chromosome.sequence.genbank.accession <-
data.structure[1, ]$sequence.genbank.accession
    data.assembly.info[i, ]$chromosome.sequence.relationship <-
data.structure[1, ]$sequence.relationship
    data.assembly.info[i, ]$chromosome.sequence.refseq.accession <-
data.structure[1, ]$sequence.refseq.accession
    data.assembly.info[i, ]$chromosome.sequence.assembly.unit <-
data.structure[1, ]$sequence.assembly.unit
    data.assembly.info[i, ]$chromosome.sequence.length <-
data.structure[1, ]$sequence.length
    data.assembly.info[i, ]$chromosome.sequence.ucsc.style.name <-
data.structure[1, ]$sequence.ucsc.style.name

    # Count number of plasmids.
    num.plasmids <- dim(data.structure)[1] - 1

    # If there are plasmids the store this information too.
    data.assembly.info[i, ]$num.plasmids <- num.plasmids

    if(num.plasmids > 0){

      data.assembly.info[i, ]$plasmid.sequence.name <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.name, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.role <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.role, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.assigned.molecule <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$assigned.molecule, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.location.type <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.location.type, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.genbank.accession <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.genbank.accession, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.relationship <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.relationship, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.refseq.accession <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.refseq.accession, collapse = "; ", sep = "")
```

```
      data.assembly.info[i, ]$plasmid.sequence.assembly.unit <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.assembly.unit, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.length <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.length, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.ucsc.style.name <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.ucsc.style.name, collapse = "; ", sep = "")

    }

  } else{

    # Download file from NCBI.
    file.ftp <- paste(ftp.site, "/", refseq.genome.id,
"_assembly_report.txt", sep = "")
    message(paste("Currently working on genome ", refseq.genome.id, "
trying to download ftp file ", file.ftp, sep = ""))
    command <- paste("wget --directory-prefix=data/genomes/",
refseq.genome.id, " '", file.ftp, "'", sep = "")
    system(command = command, wait = TRUE)

    # Add file location to assebmly info.
    data.assembly.info[i, ]$file.assembly.report <-
paste(refseq.genome.id, "_assembly_report.txt", sep = "")

    # Read in the header of the assembly statistics file.
    file.assembly.report.head <- readLines(paste("data/genomes/",
refseq.genome.id, "/", refseq.genome.id, "_assembly_report.txt", sep =
""))

    # Input the data into data.assembly.info.
    data.assembly.info[i, ]$refseq.genome.id <- refseq.genome.id
    data.assembly.info[i, ]$assembly.name <- gsub(" ", "_",
trimws(gsub(pattern = "# Assembly name:", "", paste("",
file.assembly.stats.head[grep(pattern = "# Assembly name:", x =
file.assembly.stats.head)], sep = ""))))
    data.assembly.info[i, ]$organism.name <- trimws(gsub(pattern = "#
Organism name:", "", paste("", file.assembly.report.head[grep(pattern =
"# Organism name:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$infraspecific.name <- trimws(gsub(pattern =
"# Infraspecific name:", "", paste("",
file.assembly.report.head[grep(pattern = "# Infraspecific name:", x =
file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$taxid <- as.numeric(trimws(gsub(pattern = "#
Taxid:", "", paste("", file.assembly.report.head[grep(pattern = "#
Taxid:", x = file.assembly.report.head)], sep = ""))))
    data.assembly.info[i, ]$biosample <- trimws(gsub(pattern = "#
BioSample:", "", paste("", file.assembly.report.head[grep(pattern = "#
BioSample:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$bioproject <- trimws(gsub(pattern = "#
BioProject:", "", paste("", file.assembly.report.head[grep(pattern = "#
BioProject:", x = file.assembly.report.head)], sep = "")))
```

```
    data.assembly.info[i, ]$submitter <- trimws(gsub(pattern = "#
Submitter:", "", paste("", file.assembly.report.head[grep(pattern = "#
Submitter:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$date <- as.Date(trimws(gsub(pattern = "#
Date:", "", paste("", file.assembly.report.head[grep(pattern = "# Date:",
x = file.assembly.report.head)], sep = ""))))
    data.assembly.info[i, ]$assembly.type <- trimws(gsub(pattern = "#
Assembly type:", "", paste("", file.assembly.report.head[grep(pattern =
"# Assembly type:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$release.type <- trimws(gsub(pattern = "#
Release type:", "", paste("", file.assembly.report.head[grep(pattern = "#
Release type:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$assembly.level <- trimws(gsub(pattern = "#
Assembly level:", "", paste("", file.assembly.report.head[grep(pattern =
"# Assembly level:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$genome.representation <- trimws(gsub(pattern
= "# Genome representation:", "", paste("",
file.assembly.report.head[grep(pattern = "# Genome representation:", x =
file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$relation.to.type.material <-
trimws(gsub(pattern = "# Relation to type material:", "", paste("",
file.assembly.report.head[grep(pattern = "# Relation to type material:",
x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$refseq.category <- trimws(gsub(pattern = "#
RefSeq category:", "", paste("", file.assembly.report.head[grep(pattern =
"# RefSeq category:", x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$refseq.assembly.accession <-
trimws(gsub(pattern = "# RefSeq assembly accession:", "", paste("",
file.assembly.report.head[grep(pattern = "# RefSeq assembly accession:",
x = file.assembly.report.head)], sep = "")))
    data.assembly.info[i, ]$species <- trimws(gsub(pattern =
paste(groups, collapse = "|"), "", x = data.assembly.info[i,
]$organism.name)) # First remove the groups.
    data.assembly.info[i, ]$species <- trimws(gsub(pattern = ":", " ", x
= data.assembly.info[i, ]$species)) # Then replace problematic
characters.
    data.assembly.info[i, ]$species <- trimws(gsub(pattern = ",", " ", x
= data.assembly.info[i, ]$species))
    data.assembly.info[i, ]$species <- trimws(gsub(pattern = "\\(",
"\\[", x = data.assembly.info[i, ]$species))
    data.assembly.info[i, ]$species <- trimws(gsub(pattern = "\\)",
"\\]", x = data.assembly.info[i, ]$species))
    data.assembly.info[i, ]$strain <- trimws(gsub(pattern = "strain=",
"", paste("", data.assembly.info[i, ]$infraspecific.name, sep = "")))
    data.assembly.info[i, ]$isolate <- trimws(gsub(pattern = "#
Isolate:", "", paste("", file.assembly.report.head[grep(pattern = "#
Isolate:", x = file.assembly.report.head)], sep = "")))

    data.assembly.info[i, ]$species.strain.isolate <-
trimws(paste(trimws(gsub(data.assembly.info[i, ]$strain, "",
data.assembly.info[i, ]$species)), " ", trimws(data.assembly.info[i,
]$strain), sep = ""))

    # If the strain isn't defined, use the isolate name.
```

```
    if(nchar(data.assembly.info[i, ]$strain) == 0){
      data.assembly.info[i, ]$species.strain.isolate <-
trimws(paste(trimws(gsub(data.assembly.info[i, ]$isolate, "",
data.assembly.info[i, ]$species)), " ", trimws(data.assembly.info[i,
]$isolate), sep = ""))
    }

    data.assembly.info[i, ]$group <- trimws(gsub("\\)", "",
unlist(strsplit(data.assembly.info[i, ]$organism.name,
"\\("))[length(unlist(strsplit(data.assembly.info[i, ]$organism.name,
"\\(")))]))

    # Get information about structure of assembly in terms of chrosomoes
and plasmids.
    structure.start <- grep("# Sequence-Name", file.assembly.report.head)
+ 1
    structure.end <- length(file.assembly.report.head)
    data.structure <- read.table(text =
file.assembly.report.head[structure.start:structure.end], sep = "\t",
stringsAsFactors = FALSE)
    colnames(data.structure) <- c("sequence.name", "sequence.role",
"assigned.molecule", "sequence.location.type",
"sequence.genbank.accession", "sequence.relationship",
"sequence.refseq.accession", "sequence.assembly.unit", "sequence.length",
"sequence.ucsc.style.name")

    # Store information about primary chromosome.
    data.assembly.info[i, ]$chromosome.sequence.name <- data.structure[1,
]$sequence.name
    data.assembly.info[i, ]$chromosome.sequence.role <- data.structure[1,
]$sequence.role
    data.assembly.info[i, ]$chromosome.assigned.molecule <-
data.structure[1, ]$assigned.molecule
    data.assembly.info[i, ]$chromosome.sequence.location.type <-
data.structure[1, ]$sequence.location.type
    data.assembly.info[i, ]$chromosome.sequence.genbank.accession <-
data.structure[1, ]$sequence.genbank.accession
    data.assembly.info[i, ]$chromosome.sequence.relationship <-
data.structure[1, ]$sequence.relationship
    data.assembly.info[i, ]$chromosome.sequence.refseq.accession <-
data.structure[1, ]$sequence.refseq.accession
    data.assembly.info[i, ]$chromosome.sequence.assembly.unit <-
data.structure[1, ]$sequence.assembly.unit
    data.assembly.info[i, ]$chromosome.sequence.length <-
data.structure[1, ]$sequence.length
    data.assembly.info[i, ]$chromosome.sequence.ucsc.style.name <-
data.structure[1, ]$sequence.ucsc.style.name

    # Count number of plasmids.
    num.plasmids <- dim(data.structure)[1] - 1

    # If there are plasmids the store this information too.
    data.assembly.info[i, ]$num.plasmids <- num.plasmids
```

```
    if(num.plasmids > 0){

      data.assembly.info[i, ]$plasmid.sequence.name <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.name, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.role <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.role, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.assigned.molecule <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$assigned.molecule, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.location.type <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.location.type, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.genbank.accession <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.genbank.accession, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.relationship <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.relationship, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.refseq.accession <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.refseq.accession, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.assembly.unit <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.assembly.unit, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.length <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.length, collapse = "; ", sep = "")
      data.assembly.info[i, ]$plasmid.sequence.ucsc.style.name <-
paste(paste("Plasmid", 1:num.plasmids, "=", sep = ""), data.structure[-1,
]$sequence.ucsc.style.name, collapse = "; ", sep = "")

    }

  }

  # Get the feature table file.
  file.feature.table <- paste("data/genomes/", refseq.genome.id, "/",
refseq.genome.id, "_feature_table.txt", sep = "")
  file.feature.table.gz <- paste(file.feature.table, ".gz", sep = "")

  if(file.exists(file.feature.table)){

    # Add file location to assebmly info.
    data.assembly.info[i, ]$file.feature.table <- paste(refseq.genome.id,
"_feature_table.txt", sep = "")

  } else{

    # Download file from NCBI.
    file.ftp <- paste(ftp.site, "/", refseq.genome.id,
"_feature_table.txt.gz", sep = "")
```

```
    message(paste("Currently working on genome ", refseq.genome.id, "
trying to download ftp file ", file.ftp, sep = ""))
    command <- paste("wget --directory-prefix=data/genomes/",
refseq.genome.id, " '", file.ftp, "'", sep = "")
    system(command = command, wait = TRUE)

    if(file.exists(file.feature.table.gz)){

      # Unzip file.
      command <- paste("gunzip ", file.feature.table.gz, sep = "")
      system(command = command, wait = TRUE)

      # Add file location to assebmly info.
      data.assembly.info[i, ]$file.feature.table <-
paste(refseq.genome.id, "_feature_table.txt", sep = "")

    }

  }

  # # Get the cds from genomic fasta file.
  file.cds <- paste("data/genomes/", refseq.genome.id, "/",
refseq.genome.id, "_cds_from_genomic.fna", sep = "")
  file.cds.gz <- paste(file.cds, ".gz", sep = "")

  if(file.exists(file.cds)){

    # Add file location to assebmly info.
    data.assembly.info[i, ]$file.cds <- paste(refseq.genome.id,
"_cds_from_genomic.fna", sep = "")

    # Get the fasta data.
    fasta.cds <- read.fasta(file = file.cds, seqtype = "DNA", as.string =
TRUE, forceDNAtolower = FALSE)

    # Count the number of sequences.
    data.assembly.info[i, ]$num.cds <- length(fasta.cds)

    # Store number of occurences where cds length is not divisble by 3.
    data.assembly.info[i, ]$num.cds.codon.errors <-
sum(getLength(fasta.cds) %% 3)


  } else{

    # Download file from NCBI.
    file.ftp <- paste(ftp.site, "/", refseq.genome.id,
"_cds_from_genomic.fna.gz", sep = "")
    message(paste("Currently working on genome ", refseq.genome.id, "
trying to download ftp file ", file.ftp, sep = ""))
    command <- paste("wget --directory-prefix=data/genomes/",
refseq.genome.id, " '", file.ftp, "'", sep = "")
    system(command = command, wait = TRUE)
```

```
    if(file.exists(file.cds.gz)){

      # Unzip file.
      command <- paste("gunzip ", file.cds.gz, sep = "")
      system(command = command, wait = TRUE)

      # Remove unwanted characters.
      command <- paste("sed -i '/^[^>]/s/[^ATGCatgc]/N/g' ", file.cds,
sep = "")
      system(command = command, intern = FALSE)

      # Add file location to assebmly info.
      data.assembly.info[i, ]$file.cds <- paste(refseq.genome.id,
"_cds_from_genomic.fna", sep = "")

      # Run esl-sfetch to index the cds for hmmer
      command <- paste(esl.sfetch, " --index ", file.cds, sep = "")
      system(command = command, wait = TRUE)

      # Run makeblastdb to create blast database for cds.
      command <- paste(makeblastdb, " -in ", file.cds, " -dbtype nucl -
title ", refseq.genome.id, "_cds_from_genomic.fna -parse_seqids", sep =
"")
      system(command = command, wait = TRUE)

      # Get the fasta data.
      fasta.cds <- read.fasta(file = file.cds, seqtype = "DNA", as.string
= TRUE, forceDNAtolower = FALSE)

      # Count the number of sequences.
      data.assembly.info[i, ]$num.cds <- length(fasta.cds)

      # Store number of occurences where cds length is not divisble by 3.
      data.assembly.info[i, ]$num.cds.codon.errors <-
sum(getLength(fasta.cds) %% 3)

    }

  }

  # # Get the translated cds fasta file.
  file.translated.cds <- paste("data/genomes/", refseq.genome.id, "/",
refseq.genome.id, "_translated_cds.faa", sep = "")
  file.translated.cds.gz <- paste(file.translated.cds, ".gz", sep = "")

  if(file.exists(file.translated.cds)){

    # Add file location to assebmly info.
    data.assembly.info[i, ]$file.translated.cds <-
paste(refseq.genome.id, "_translated_cds.faa", sep = "")

    # Get the fasta data.
    fasta.translated.cds <- read.fasta(file = file.translated.cds,
seqtype = "AA", as.string = TRUE, forceDNAtolower = FALSE)
```

```
    # Count the number of sequences.
    data.assembly.info[i, ]$num.translated.cds <-
length(fasta.translated.cds)

    if(file.exists(file.cds)){

      # Get the fasta data.
      fasta.cds <- read.fasta(file = file.cds, seqtype = "DNA", as.string
= TRUE, forceDNAtolower = FALSE)

      # Store number of occurences where cds length does not correspond
to translated cds length.
      data.assembly.info[i, ]$num.cds.translation.errors <- sum(3 *
getLength(fasta.cds) == getLength(fasta.translated.cds))

    }

  } else {

    # Download file from NCBI.
    file.ftp <- paste(ftp.site, "/", refseq.genome.id,
"_translated_cds.faa.gz", sep = "")
    message(paste("Currently working on genome ", refseq.genome.id, "
trying to download ftp file ", file.ftp, sep = ""))
    command <- paste("wget --directory-prefix=data/genomes/",
refseq.genome.id, " '", file.ftp, "'", sep = "")
    system(command = command, wait = TRUE)

    if(file.exists(file.translated.cds.gz)){

      # Unzip file.
      command <- paste("gunzip ", file.translated.cds.gz, sep = "")
      system(command = command, wait = TRUE)

      # Remove unwanted characters.
      command <- paste("sed -i
'/^[^>]/s/[^ARNDCQEGHILKMFPSTWYVBZarndcqeghilkmfpstwyv]/X/g' ",
file.translated.cds, sep = "")
      system(command = command, intern = FALSE)

      # Add file location to assebmly info.
      data.assembly.info[i, ]$file.translated.cds <-
paste(refseq.genome.id, "_translated_cds.faa", sep = "")

      # Run esl-sfetch to index the translated cds for hmmer.
      command <- paste(esl.sfetch, " --index ", file.translated.cds, sep
= "")
      system(command = command, wait = TRUE)

      # Run makeblastdb to create blast database for translated cds
      command <- paste(makeblastdb, " -in ", file.translated.cds, " -
dbtype prot -title ", refseq.genome.id, "_translated_cds.faa -
parse_seqids", sep = "")
```

```
      system(command = command, wait = TRUE)

      # Get the fasta data.
      fasta.translated.cds <- read.fasta(file = file.translated.cds,
seqtype = "AA", as.string = TRUE, forceDNAtolower = FALSE)

      # Count the number of sequences.
      data.assembly.info[i, ]$num.translated.cds <-
length(fasta.translated.cds)

      if(file.exists(file.cds)){

        # Get the fasta data.
        fasta.cds <- read.fasta(file = file.cds, seqtype = "DNA",
as.string = TRUE, forceDNAtolower = FALSE)

        # Store number of occurences where cds length does not correspond
to translated cds length.
        data.assembly.info[i, ]$num.cds.translation.errors <- sum(3 *
getLength(fasta.cds) == getLength(fasta.translated.cds))

      }

    }

  }

  # Get the rna from genomic fasta file.
  file.rna <- paste("data/genomes/", refseq.genome.id, "/",
refseq.genome.id, "_rna_from_genomic.fna", sep = "")
  file.rna.gz <- paste(file.rna, ".gz", sep = "")

  if(file.exists(file.rna)){

    # Add file location to assebmly info.
    data.assembly.info[i, ]$file.rna <- paste(refseq.genome.id,
"_rna_from_genomic.fna", sep = "")

    # Get the fasta data.
    fasta.rna <- read.fasta(file = file.rna, seqtype = "DNA", as.string =
TRUE, forceDNAtolower = FALSE)

    # Count the number of sequences.
    data.assembly.info[i, ]$num.rna <- length(fasta.rna)

  } else {

    # Download file from NCBI.
    file.ftp <- paste(ftp.site, "/", refseq.genome.id,
"_rna_from_genomic.fna.gz", sep = "")
    message(paste("Currently working on genome ", refseq.genome.id, "
trying to download ftp file ", file.ftp, sep = ""))
    command <- paste("wget --directory-prefix=data/genomes/",
refseq.genome.id, " '", file.ftp, "'", sep = "")
```

```
    system(command = command, wait = TRUE)

    if(file.exists(file.rna.gz)){

      # Unzip file.
      command <- paste("gunzip ", file.rna.gz, sep = "")
      system(command = command, wait = TRUE)

      # Remove unwanted characters.
      command <- paste("sed -i '/^[^>]/s/[^ATGCUatgcu]/N/g' ", file.rna,
sep = "")
      system(command = command, intern = FALSE)

      # Add file location to assebmly info.
      data.assembly.info[i, ]$file.rna <- paste(refseq.genome.id,
"_rna_from_genomic.fna", sep = "")

      # Run esl-sfetch to index the rna for hmmer.
      command <- paste(esl.sfetch, " --index ", file.rna, sep = "")
      system(command = command, wait = TRUE)

      # Run makeblastdb to create blast database for rna.
      command <- paste(makeblastdb, " -in ", file.rna, " -dbtype nucl -
title ", refseq.genome.id, "rna_from_genomic.fna -parse_seqids", sep =
"")
      system(command = command, wait = TRUE)

      # Get the fasta data.
      fasta.rna <- read.fasta(file = file.rna, seqtype = "DNA", as.string
= TRUE, forceDNAtolower = FALSE)

      # Count the number of sequences.
      data.assembly.info[i, ]$num.rna <- length(fasta.rna)

    }

  }

  # Get the genomic gff file.
  file.genomic.gff <- paste("data/genomes/", refseq.genome.id, "/",
refseq.genome.id, "_genomic.gff", sep = "")
  file.genomic.gff.gz <- paste(file.genomic.gff, ".gz", sep = "")

  if(file.exists(file.genomic.gff)){

    # Add file location to assebmly info.
    data.assembly.info[i, ]$file.genomic.gff <- paste(refseq.genome.id,
"_genomic.gff", sep = "")

  } else{

    # Download file from NCBI.
    file.ftp <- paste(ftp.site, "/", refseq.genome.id, "_genomic.gff.gz",
sep = "")
```

```
    message(paste("Currently working on genome ", refseq.genome.id, "
trying to download ftp file ", file.ftp, sep = ""))
    command <- paste("wget --directory-prefix=data/genomes/",
refseq.genome.id, " '", file.ftp, "'", sep = "")
    system(command = command, wait = TRUE)

    if(file.exists(file.genomic.gff.gz)){

      # Unzip file.
      command <- paste("gunzip ", file.genomic.gff.gz, sep = "")
      system(command = command, wait = TRUE)

      # Add file location to assebmly info.
      data.assembly.info[i, ]$file.genomic.gff <- paste(refseq.genome.id,
"_genomic.gff", sep = "")

    }

  }

}

# Number the genomes
rownames(data.assembly.info) <- 1:dim(data.assembly.info)[1]

# Now remove genomes that were not fully annotated.
idx.not.annotated <- which(is.na(data.assembly.info$file.assembly.report)
| is.na(data.assembly.info$file.feature.table) |
is.na(data.assembly.info$file.cds) |
is.na(data.assembly.info$file.translated.cds) |
is.na(data.assembly.info$file.rna) |
is.na(data.assembly.info$file.genomic.gff))

# Number of excluded genomes.
num.genomes.excluded <- length(idx.not.annotated)

# Update number of genomes.
num.genomes <- num.genomes - num.genomes.excluded

if(num.genomes.excluded > 0){

  # Remove excluded data.
  data.assembly.info.excluded <- data.assembly.info[idx.not.annotated, ]

  # Number the excluded genomes
  rownames(data.assembly.info.excluded) <-
1:dim(data.assembly.info.excluded)[1]

  # Remove excluded data.
  data.assembly.info <- data.assembly.info[-idx.not.annotated, ]

  # Re-number the genomes
  rownames(data.assembly.info) <- 1:dim(data.assembly.info)[1]
```

```
  # Create new folders if they doesn't already exist.
  command <- paste("mkdir -p data/", experiment, "/genomes_excluded/",
sep = "")
  system(command, intern = TRUE)
  command <- paste("mkdir -p data/", experiment,
"/assembly_stats_excluded/", sep = "")
  system(command, intern = TRUE)

  for(i in 1:num.genomes.excluded){

    # Move the excluded assembly stats files.
    command <- paste("mv data/", experiment, "/assembly_stats/",
data.assembly.info.excluded$refseq.assembly.accession[i], "_",
data.assembly.info.excluded$assembly.name[i], "_assembly_stats.txt
data/", experiment, "/assembly_stats_excluded/", sep = "")
    system(command = command, intern = FALSE)

    # Move the excluded genomes.
    command <- paste("mv data/genomes/",
data.assembly.info.excluded$refseq.assembly.accession[i], "_",
data.assembly.info.excluded$assembly.name[i], " data/", experiment,
"/genomes_excluded/", sep = "")
    system(command = command, intern = FALSE)

  }

  # Write excluded to tsv file.
  file.data.assembly.info.excluded <- paste("data/", experiment,
"/data_assembly_info_excluded.tsv", sep = "")
  write.table(data.assembly.info.excluded, file =
file.data.assembly.info.excluded, quote = FALSE, sep = "\t", col.names =
NA)

}

# Write to tsv file.
file.data.assembly.info <- paste("data/", experiment,
"/data_assembly_info.tsv", sep = "")
write.table(data.assembly.info, file = file.data.assembly.info, quote =
FALSE, sep = "\t", col.names = NA)

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