CRISPR-Cas Systems analysis

Thomas Nicholson

2/9/2017

savehistory(file = paste("~/.logs/r_history_", Sys.Date() , ".log", sep = ""))

sink(file = paste("~/.logs/r_output_", Sys.Date() , ".log", sep = ""), split = T, append = T)

Genomes with cas genes are identified with the following code (after previous steps have been carried out in a shell script). Each gene has a score for a match to a cas model and a cdd model (which contains all models of proteins).

A gene is labelled as a cas gene if: * the cas model e value is less than 1e-10 * the cdd model e value is greater than 1e-20 * the cas model e value is less than the cdd model e value.

Genes are then matched with the CRISPR systems in which they are found. A list of these matches can be seen in the table "Cas genes assignment to CRISPR systems"

Each gene is then assigned as a signature gene (able to determine subtypes) or not based on the genes indicated as signatures in Makarova 2015 and further curated data based on genes that have caused problems in analysis (such as DinG). The table presented contains columns unique.y.n and subtypes.y.n Both of these must be true for the gene to act as a signature.

These results are then recorded, along with all of the data used to make the calculations into a data frame with all of the genomes containing cas proteins.

_this part of the script is not currently running as it takes some time and the files are only on the server at the moment.

```
##Import data
setwd('~/Desktop/Project/identifycasproteins/')

##set the names for importing files and variables to use in this section. The files are not currently or
args <- c('archaea_refseq_79.hmm.tab',
          'archaeal_proteins_rps_results_refseq79.nr.tab',
          'fasta_file_location_archaeal_refseq79.txt',
          'cas_genes_and_systems.txt',
          'models_and_systems.txt',
          'Archaea')
cas <- read.table(args[1], header = T, comment.char = '#', as.is = T, fill = T)
cdd <- read.table(args[2], header = T, sep = '\t', comment.char = '', as.is = T, fill = T)
genomes <- read.table(args[3], header = F, sep = '\t', comment.char = '', as.is = T, fill = T)
genes <- read.table(args[4], header = T, sep = '\t', comment.char = '', as.is = T, fill = T)
signatures <- read.table(args[5], header = T, sep = '\t', comment.char = '', as.is = T, fill = T)
genes.dat <- data.frame(Accession = cas$Protein_ID, cas.id = cas$Cas_Model,
                        gene.name = vector(mode='character', length=length(cas$Protein_ID)), putative.sy
                        cas.e.value.num= cas$Cas_E_value, cdd.e.value.num = vector(mode='numeric', lengt
                        genome = vector(mode='character', length=length(cas$Protein_ID)), keep.y.n = vec
                        also.in.genomes = vector(mode='character', length=length(cas$Protein_ID)))
cas <- cas[,c(1,3,5,19,20)]
x1 <- vector(mode = 'character', length = length(unique(genomes$V2)))
for(i in 1:length(unique(genomes$V2))){
  y <- strsplit(unique(genomes[i,2]), '/')[[1]][14]
  x1[i] <- y
}
```

```
x2 <- vector(mode = 'character', length = length(unique(genomes$V2)))
for(i in 1:length(unique(genomes$V2))){
  y <- paste('/mnt/SSD/CD_Tracr_Cas9',paste(strsplit(substr(genomes[i,2], 57, nchar(genomes[i,2])),'/')
  x2[i] <- y
}
x3 <- vector(mode = 'character', length = length(unique(genomes$V2)))
for(i in 1:length(unique(genomes$V2))){
  y <- substr(unique(genomes[i,2]),12,nchar(genomes[i,2]))
  x3[i] <- y
}
genomes.dat <- data.frame(genome =x1,
                          genes = vector(mode = 'character', length=length(unique(genomes$V2))),
                          basic.genes.present.y.n = vector(mode = 'logical', length=length(unique(genome
                          subtypes = vector(mode = 'character', length=length(unique(genomes$V2))),
                          fastafile_path =  x3,
                          CRISPRDetect_path = x2,
                          Classification = vector(mode = 'character', length=length(unique(genomes$V2))
                          cas.e.values.num = vector(mode = 'character', length=length(unique(genomes$V2
                          cdd.e.values.num = vector(mode = 'character', length=length(unique(genomes$V2
                          cas.models = vector(mode = 'character', length=length(unique(genomes$V2))),
                          all.proteins = genomes$V1)
genes[genes[,3]=='CAS-III-C' & genes[,2]=='cas10', 2] <- 'cas10c'
gene.summary.dat <- data.frame(gene = unique(genes$Gene), systems = vector(mode = 'character', length =
gene.summary.dat <- transform.df(gene.summary.dat)
#unique(genes$Gene)[substr(unique(genes$Gene), 1, 5)=='cas10']
for(i in 1:length(unique(genes$Gene))){
  cas.gene <- unique(genes$Gene)[i]
  dat <- genes[genes[,2]==cas.gene,]
  gene.summary.dat[i, 2] <- paste(sort(unique(dat[,3])), collapse = ',')
  gene.summary.dat[i, 5] <- paste(sort(unique(dat[,1])), collapse = ',')
  if(length(unique(dat[,3]))==1){
    gene.summary.dat[i, 3] <- T
  }else{
    gene.summary.dat[i, 3] <- F
  }

}

gene.summary.dat <- gene.summary.dat[gene.summary.dat[,2]!='CAS-I,CAS-III',]
gene.summary.dat$subtypes.y.n <- T
gene.summary.dat[gene.summary.dat[,2]=='CAS-I', 4] <- F
gene.summary.dat[gene.summary.dat[,2]=='CAS-II', 4] <- F
gene.summary.dat[gene.summary.dat[,2]=='CAS-III', 4] <- F
gene.summary.dat[gene.summary.dat[,1]=='csa3', 3] <- F
gene.summary.dat[gene.summary.dat[,1]=='DinG', 3] <- F
gene.summary.dat[gene.summary.dat[,1]=='csf2gr7', 3] <- F
gene.summary.dat[gene.summary.dat[,1]=='csf3gr5', 3] <- F
gene.summary.dat[gene.summary.dat[,1]=='csf4gr11', 3] <- F
gene.summary.dat[gene.summary.dat[,1]=='csf5gr6', 3] <- F

#-----------------------------------commands to run single core----------------------------------------
#genes.dat <- transform.df(genes.dat)
#genomes.dat <- transform.df(genomes.dat)
```

```
#ptm <- proc.time()
#tmp <- sapply(1:length(cas[,1]),
#               function(x) {write.genes.dat(x, cas, cdd, genes.dat, signatures, gene.summary.dat, geno
#proc.time() - ptm
#genes.dat <- post.sapply.funct(tmp, genes.dat)

genes.dat <- transform.df(genes.dat)
genomes.dat <- transform.df(genomes.dat)
gene.summary.dat <- transform.df(gene.summary.dat)

ptm <- proc.time()
for(i in 1:length(cas[,1])){
  #print(i)
  genes.dat[i,] <- write.genes.dat(i, cas, cdd, genes.dat, signatures, gene.summary.dat, genomes, print
}

proc.time() - ptm
genes.dat[genes.dat[,3]=='DinG',3] <- ''
genes.dat[genes.dat[,3]=='casR',3] <- ''
genes.dat[genes.dat[,3]=='cas3HD',3] <- ''
genes.dat[genes.dat[,3]=='cmr4gr7',3] <- ''
genes.dat[genes.dat[,3]=='cmr6gr7',3] <- ''
genes.dat[genes.dat[,3]=='csm3gr7',3] <- ''
genes.dat[genes.dat[,3]=='csm6gr7',3] <- ''
genes.dat[genes.dat[,3]=='csx1gr7',3] <- ''
genes.dat[genes.dat[,3]=='csm5gr7',3] <- ''
genes.dat[genes.dat[,3]=='cmr1gr7',3] <- ''
genes.dat[genes.dat[,3]=='csm4gr5',3] <- ''
#genes.dat[genes.dat] <- ''
genes.dat[genes.dat[,3]=='csm6',3] <- ''
genes.dat[genes.dat[,3]=='csx1',3] <- ''
genes.dat[genes.dat[,3]=='DEDDh',3] <- ''

y <- grep('cas7', genes.dat[,3])
if(length(y)!=0){
  for(i in 1:length(y)){
    genes.dat[y[i],3] <- 'cas7'
  }

}
y <- grep('cas8b', genes.dat[,3])
if(length(y)!=0){
  for(i in 1:length(y)){
    genes.dat[y[i],3] <- 'cas8b'
  }
}
y <- grep('cas5', genes.dat[,3])
if(length(y)!=0){
  for(i in 1:length(y)){
    genes.dat[y[i],3] <- 'cas5'
  }
}
y <- grep('cas3', genes.dat[,3])
```

```
if(length(y)!=0){
  for(i in 1:length(y)){
    genes.dat[y[i],3] <- 'cas3'
  }
}

a <- c()
for(i in 1:length(genomes.dat[,1])){
  print(i)
  genome <- genomes.dat[i,1]
  x <- grep(genome, genes.dat$genome)
  #if(length(x)==0){
  #x <- grep(x, genes.dat$also.in.genomes)
  #}
  #print(paste('Genome ',genome,' (',i, '): ',x, sep=''))
  dat <- genes.dat[x,]

  uniq.uids <- unique(dat[,1])
  uniq.dat <- dat[1:length(uniq.uids),]
  #print(uniq.dat)
  uniq.dat <- transform.df(uniq.dat)
  for(j in 1:length(uniq.uids)){
    x <- uniq.uids[j]
    tmp.dat <- dat[dat[,1]==x,]
    tmp.dat <- tmp.dat[order(tmp.dat[,5]),]
    if(length(unique(tmp.dat[,3]))<2){
      uniq.dat[j,] <- tmp.dat[1,]
    }else{
      uniq.dat[j,] <- tmp.dat[1,]
      a <- c(a, paste(unique(tmp.dat[,3]), collapse = ','))
      #print(uniq.genes)
    }
  }
  genomes.dat[i,8] <- args[6]
  gene.list <- uniq.dat[,3]
  print(gene.list)
  genomes.dat[i,2] <- paste(gene.list, collapse = ',')
  genomes.dat[i,5] <- paste(sort(unique(uniq.dat[,4])), collapse = ',')
  genomes.dat[i,9] <- paste(uniq.dat[,5], collapse = ',')
  genomes.dat[i,10] <- paste(sort(unique(uniq.dat[,5])), collapse = ',')
  genomes.dat[i,11] <- paste(uniq.dat[,2], collapse = ',')
  genomes.dat[i,12] <- paste(uniq.dat[,1], collapse = ',')
  x <-match('cas1', uniq.dat[,3])
  y <-match('cas2', uniq.dat[,3])

  if(!is.na(x) & !is.na(y)){
    genomes.dat[i,3] <- T
  }
  if(genomes.dat[i,5]==''){
    x <- grep('cas9', uniq.dat$gene.name)
    y <- grep('cas4', uniq.dat$gene.name)
    z <- grep('csn2', uniq.dat$gene.name)
    if(length(x)>0){
```

```
      if(length(y)>0){
        genomes.dat[i,5] <- ',CAS-II-B'
      }else if(length(z)==0){
        genomes.dat[i,5] <- ',CAS-II-C'

      }
    }
  }
  if(genomes.dat[i,5]==',CAS-I-E'){
    y <- grep('cas4', uniq.dat$gene.name)
    if(length(x)>0){
      genomes.dat[i,5] <- ',CAS-I-E, OTHER?'
      genomes.dat[i,4] <- F
    }
  }
  if(genomes.dat[i,5]==',CAS-I-F'){
    y <- grep('cas4', uniq.dat$gene.name)
    if(length(x)>0){
      genomes.dat[i,5] <- ',CAS-I-F, OTHER?'
      genomes.dat[i,4] <- F
    }
  }
  x <- strsplit(genomes.dat[i, 5], ',')
  if(length(x[[1]])!=0){
    if(x[[1]][1]==''){
      x[[1]] <- x[[1]][2:length(x[[1]])]
    }
    if(length(x[[1]])==1){
      genomes.dat[i,4] <- T

    }else{
      genomes.dat[i,4] <- F
    }
  }else{
    genomes.dat[i,4] <- F

  }
  if(genomes.dat[i,2]==F){
    print(i)
    #genomes.dat[i,7] <- ''

  }


}

a.uniq <- data.frame(list = unique(a), counts.num = vector(mode = 'numeric', length = length( unique(a)
a.uniq <- transform.df(a.uniq)
for(i in 1:length(a.uniq[,1])){
  x <- a.uniq[i,1]
  #print(x)
  y <- a[a==x]
  #print(length(y))
```

```
  a.uniq[i,2] <- length(y)
}
write.table(a.uniq, 'uids_matched_with_multiple_cas_models_archaea_refseq_79.txt', quote = F, row.names
write.table(genes.dat, 'cas_protein_data_archaea_refseq_79.txt', quote = F, row.names = F, sep = '\t')
write.table(genomes.dat, 'genomes_with_cas_proteins_archaea_refseq_79.txt', quote = F, row.names = F, s
write.table(genomes.dat[genomes.dat[,4]==T, ], 'genomes_with_single_system_archaea_refseq_79.txt', quot
```

```
crispr_models_data <- read.table("~/Desktop/Project/identifycasproteins/genes_and_systems_summary.txt",
crispr_models_data <- crispr_models_data[crispr_models_data[,3]==T,]
tab_7 <- xtable(crispr_models_data[,1:4], caption=('Cas genes assignment to CRISPR systems'))
print(tab_7, type="latex", caption.placement='top', comment=FALSE)
```

Data at this point has been assigned CRISPR systems and matched up with CRISPRDetect files. The script from here is working on producing some summary results.

I will need to take the genomes that look like they have CRISPR systems and are missing CRISPRDetect files and rerun these to see what is going on. I many cases the support for a CRISPR systems is strong based on the genes found.

```
#----------------------------------Setup and import data----------------------------------#
##log history from session

#savehistory(file = paste("~/.logs/r_history_", Sys.Date() , ".log", sep = ""))
#sink(file = paste("~/.logs/r_output_", Sys.Date() , ".log", sep = ""), split = T, append = T)
setwd('~/Desktop/Project/data_analysis_crispr/')
genomes <- read.table("refseq_79.genomes_summary.txt", header = T, sep = '\t', as.is = T, comment.char =

gene.count <- vector(mode = 'numeric', length = length(genomes[,1]))
for(i in 1:length(genomes[,1])){
  x <- strsplit(genomes[i,2], ',')
  y <- length(x[[1]])
  gene.count[i] <- y
}


systems.count <- vector(mode = 'numeric', length = length(genomes[,1]))
for(i in 1:length(genomes[,1])){
  if(!is.na(genomes[i,5])){
  if(substr(genomes[i,5],1,1)==','){
    x <- strsplit(genomes[i,5], ',')
    y <- length(x[[1]])
    systems.count[i] <- y-1
  }else{
    x <- strsplit(genomes[i,5], ',')
    y <- length(x[[1]])
    systems.count[i] <- y
  }
}
}
systems.count.not.zero <- systems.count[systems.count!=0]
y <- c(length(genomes[genomes[,2]==',',1]),length(systems.count.not.zero[systems.count.not.zero==1]),le
x <- c('0', '1','2','3','4','5')
#
```

```
subtype.list <- c('CAS-I-A','CAS-I-B','CAS-I-C','CAS-I-D','CAS-I-E','CAS-I-F','CAS-I-U','CAS-II-A','CAS

subtypes.dat <- subtypes.counts.funct(subtype.list, genomes )#subtype.list can be replaced with any oth
subtypes.dat.arc <- subtypes.counts.funct(subtype.list, genomes[genomes[,8]=='Archaea',] )#subtype.list
subtypes.dat.bac <- subtypes.counts.funct(subtype.list, genomes[genomes[,8]=='Bacteria',] )#subtype.lis
#hist(z, xlim = c(0, max(z)), breaks = max(z/5), main = 'Distribution of number of spacers in arrays', 
arrays.dat <- c(length(genomes[genomes[,14]==1,1]),length(genomes[genomes[,14]==2,1]),length(genomes[ge
arrays.dat.names <- c('1','2','3','4','5','6','7','8','9')

spacer.count <- c()
av.spacers.num <- c()
for(i in 1:length(genomes[,1])){
  a <- strsplit(genomes[i,13], ',')
  spacer.count <- c(spacer.count,as.numeric(a[[1]]))
  av.spacers.num <- c(av.spacers.num, mean(as.numeric(a[[1]])))
}
gene.count.num <- gene.count
genomes$array.num <- genomes$array.num/2
genomes <- cbind(genomes, av.spacers.num, gene.count.num)
genomes <- transform.df(genomes)
arrays.model <- lm(genomes$av.spacers.num~genomes$array.num )
summary.arrays.model <- summary(arrays.model)


ordered.gene.count <- gene.count[order(-gene.count)]
ordered.gene.count[1:10]
```

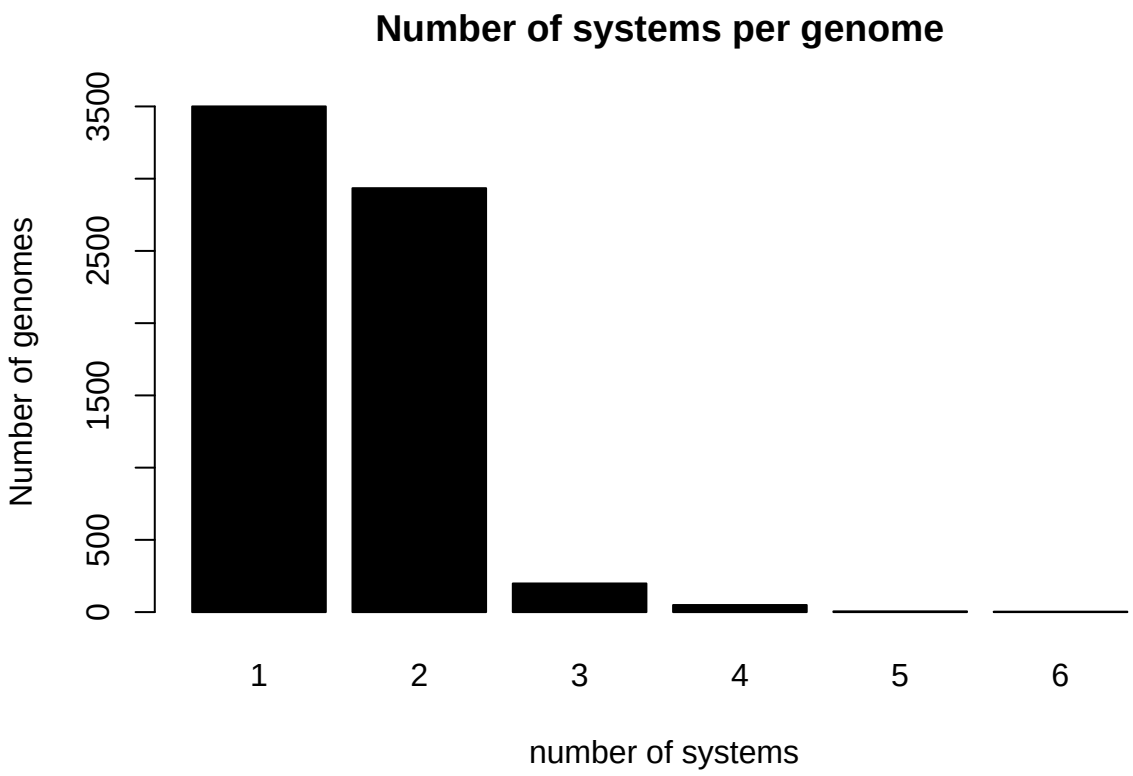
[1] 166 137 121 102 99 95 92 90 87 85

```
gene.frequency <- data.frame(table(ordered.gene.count))
ordered.gene.freq <- gene.frequency[order(-gene.frequency[,2]),]
tab_1 <- xtable(ordered.gene.freq[1:20,], caption = "Gene Frequency")
print(tab_1, type="latex", caption.placement='top', comment=FALSE)
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

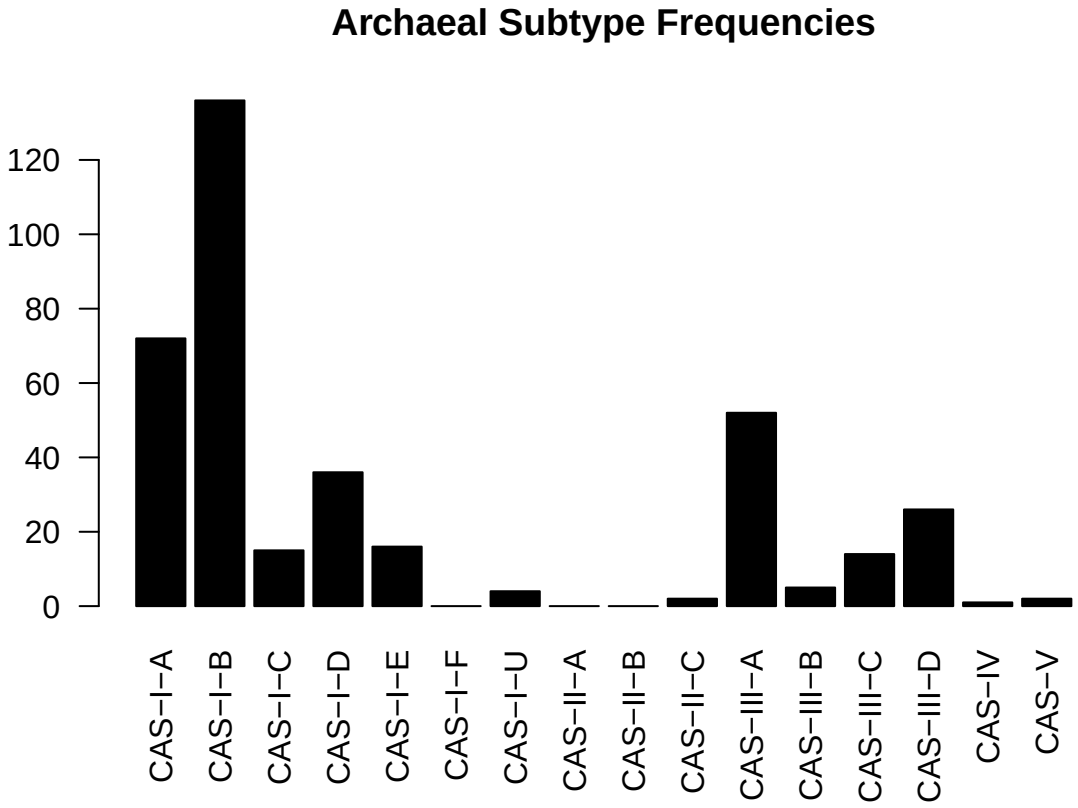
```
system.freq <- data.frame(table(systems.count))
system.freq <- system.freq[system.freq[,1]!="0",]
barplot(system.freq[,2], names.arg = system.freq[,1], col = "Black", xlab = " number of systems", ylab =
```




```
tab_2 <- xtable(system.freq, caption = "Number of systems per genome")
print(tab_2, type="latex", caption.placement='top', comment=FALSE)
```

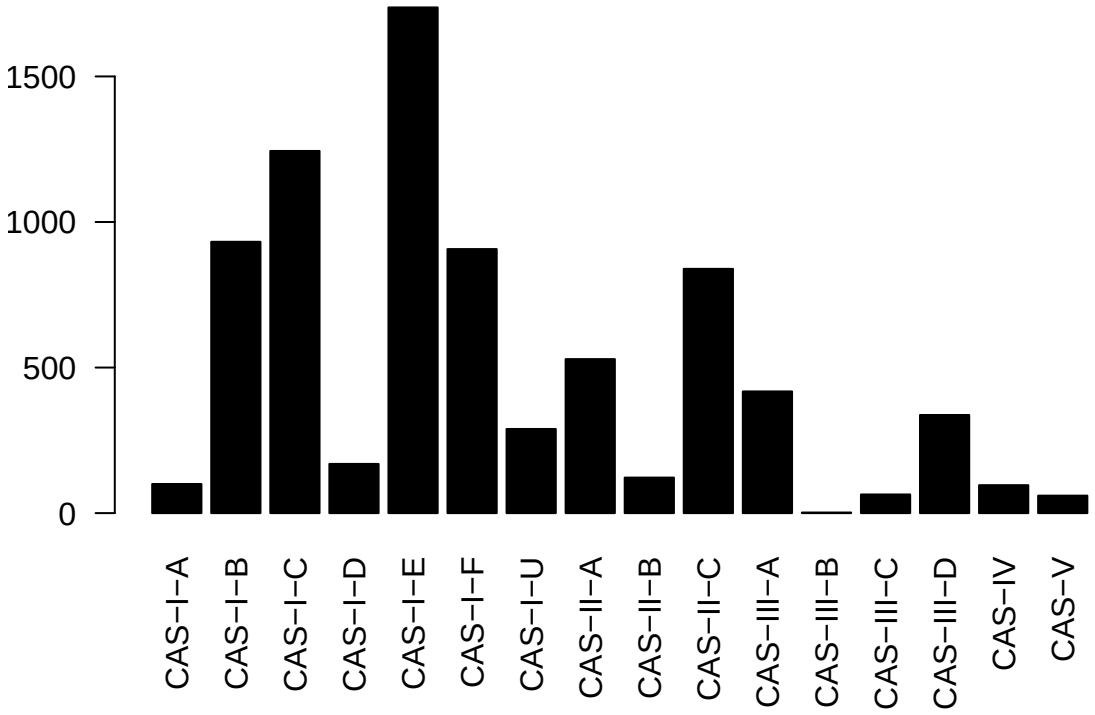
```
tab_3 <- xtable(subtypes.dat.arc, caption = "Number of archaeal genomes with each subtype")
print(tab_3, type="latex", caption.placement='top', comment=FALSE)
```

```
barplot(subtypes.dat.arc$counts, names.arg = subtypes.dat.arc$subtype, cex.names = 1, main = 'Archaeal S
```



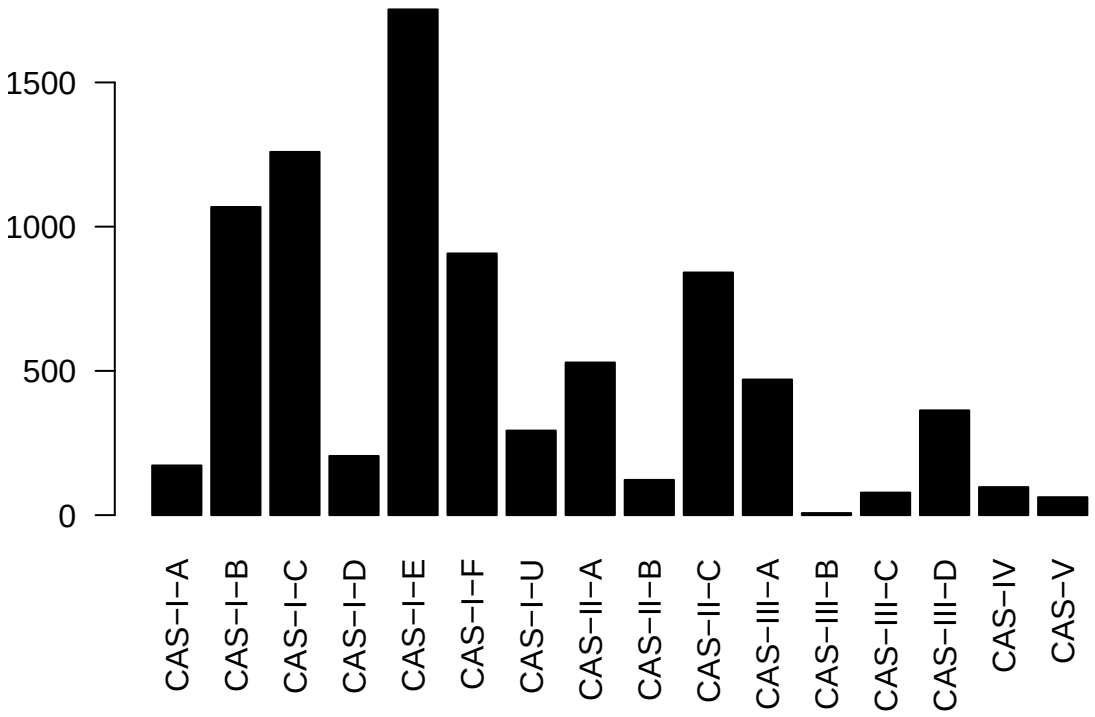

```
tab_4 <- xtable(subtypes.dat.bac, caption = "Number of bacterial genomes with each subtype")
print(tab_4, type="latex", caption.placement='top', comment=FALSE)
```

```
barplot(subtypes.dat.bac$counts, names.arg = subtypes.dat.bac$subtype, cex.names = 1, main = 'Bacterial
```




```
tab_5 <- xtable(subtypes.dat, caption = "Number of genomes with each subtype")
print(tab_5, type="latex", caption.placement='top', comment=FALSE)
```

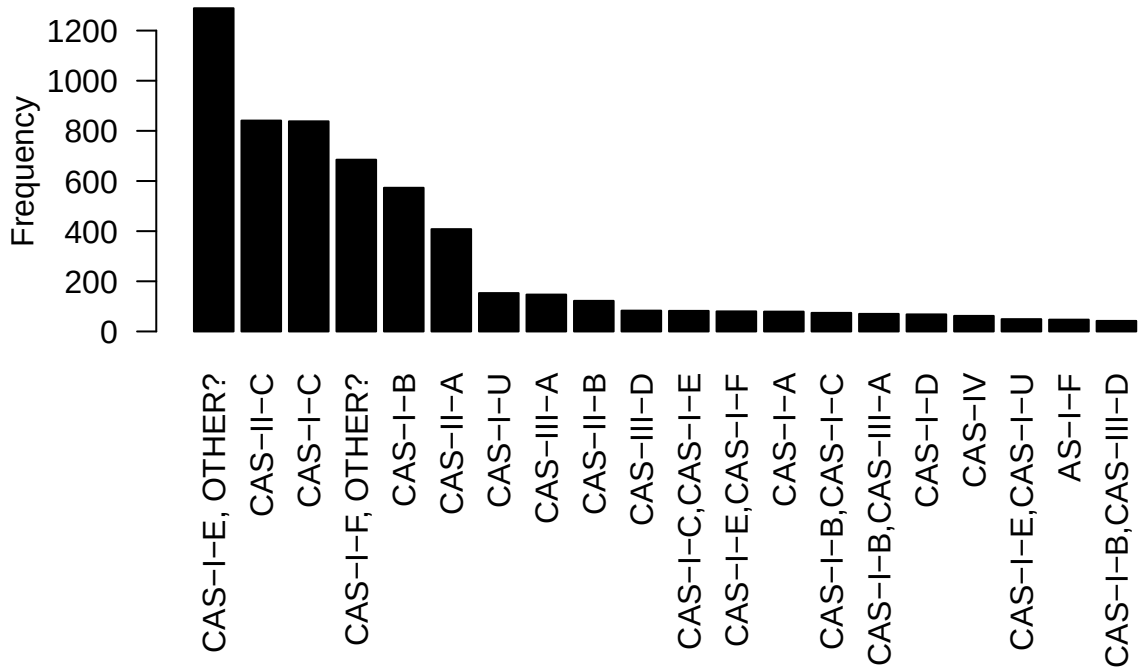
```
barplot(subtypes.dat$counts, names.arg = subtypes.dat$subtype, cex.names = 1, main = 'Subtype Frequenci
```




```
summary.dat <- read.table("~/Desktop/Project/data_analysis_crispr/refseq_79.genomes_summary.txt", sep =
summary.dat <- transform.df(summary.dat)
summary.dat[,5] <- substr(summary.dat[,5], 2, nchar(summary.dat[,5]))

subtype.dat <- as.data.frame(table(as.factor(summary.dat$subtypes)))
colnames(subtype.dat) <- c("Subtypes", "Freq")
subtype.dat <- subtype.dat[subtype.dat[,1]!='',]
subtype.dat.common <- subtype.dat[subtype.dat[,2]>50,]
ordered.subtype.dat <- subtype.dat[order(-subtype.dat[,2]),]
tab1 <- xtable(ordered.subtype.dat[1:50,], caption=('Frequency of different combinations of subtypes'))
print(tab1, type="latex", caption.placement='top', comment=FALSE)
```

```
subtype.dat <- subtype.dat[order(-subtype.dat$Freq),]
par(mar = c(10,4,4,2))
barplot(ordered.subtype.dat[1:20,2], names.arg = ordered.subtype.dat[1:20,1], col = "Black", las=2, ylab
```




```
##Summary of the overall abundance and other data like that

xx <- data.frame(Archaea = c(453, 415, 313, 159, 220), Bacteria = c(31953, 18621, 6651, 3340, 3489))
rownames(xx) <- c("Number of genomes", "Number of genomes with cas", "Number of genomes with cas protei
tab_6 <- xtable(xx, caption=('Genome analysis summary'))
print(tab_6, type="latex", caption.placement='top', comment=FALSE)
```

CRISPRTarget files I current have for refseq_79: * Archaea * Genbank and refseq phage * PHAST * IslandViewer * Bacteria * Still Running full search

Using archaeal data to work ont he filtering, genome lengths and setup of the distributions.

Steps to work through:

- Filter the results
  - combine related host genomes to reduce bias from differences in sampling.
  - some reduction will be needed on the protospacer hits where there is more than one match in a genome.
  - find ways to identify related target genomes.
- Use genome length information to plot the distances from each spacer (needed for statistical analysis and because the genomes will be treated as circular).
- Decision about using unique arrays or unique genomes is needed.
- Assign initial hit for each target genome.
- Assign the subsequent hits to target and non-target strand.
- Use ks-test to compare the two strands.
- Look at the distribution of all spacers across the genome.
  - Looking to see if clustering is occuring
  - Need a distribution for circular genomes that would be expected by chance (maybe similar to last year's naive distribution).

```
##Import CRISPRTarget output
dat <- read.table(file = "~/Desktop/Project/CRISPRClustering/archaeal_refseq_79_CRISPRTarget_text_repor

##change column names so that transform.df works
colnames(dat) <- c("Spacer_ID", "Spacer_index", "Protospacer_seq_id", "Protospacer_start.num", "Protospa
 dat <- transform.df(dat)
##Remove any duplicate rows
duplicate.rows <- unique(dat)
if(nrow(dat)==nrow(duplicate.rows)){
  print("There are no duplicate rows.
        Now checking the protospacers, score, start and stop lines.")
}else{
  print("Duplicate rows removed.
        Now checking the protospacers, score, start and stop lines.")
}
```

```
## [1] "There are no duplicate rows.\n        Now checking the protospacers, score, start and stop lines
```

```
##Looking for any hits that are identical

duplicate.hits <- dat[,c(3,4,5,8)]
duplicate.hits <- unique(duplicate.hits)
nrow(duplicate.hits)
```

```
## [1] 3095
```

```
if(nrow(dat)==nrow(duplicate.hits)){
  print("There are no duplicate hits.")
}else{
  print("Duplicate hits removed.")
}
```

```
## [1] "There are no duplicate hits."
```

```
##get array numbers and spacer numbers for each hit
index.numbers <- rep(NA, nrow(dat))
index.numbers <- sapply(1:nrow(dat), function(i){
  strsplit(dat[i,2], "\\|")
  })
index.numbers <- sapply(1:length(index.numbers), function(i) {
  index.numbers[[i]][1]
})
index.numbers[1:3]
```

```
## [1] "Archaea_III_D_NZ_CP009511_Methanosarcina_mazei_SarPi,_complete_genome._2_31"
## [2] "Archaea_III_D_NZ_CP009511_Methanosarcina_mazei_SarPi,_complete_genome._2_31"
## [3] "Archaea_III_D_NZ_CP009511_Methanosarcina_mazei_SarPi,_complete_genome._2_31"
```

```
index.numbers <- sapply(1:length(index.numbers), function(i){
  strsplit(index.numbers[i], "_")
})
index.numbers[1:3]
```

```
## [[1]]
##  [1] "Archaea"        "III"            "D"              "NZ"
##  [5] "CP009511"       "Methanosarcina" "mazei"          "SarPi,"
##  [9] "complete"       "genome."        "2"              "31"
##
## [[2]]
##  [1] "Archaea"        "III"            "D"              "NZ"
##  [5] "CP009511"       "Methanosarcina" "mazei"          "SarPi,"
##  [9] "complete"       "genome."        "2"              "31"
##
## [[3]]
##  [1] "Archaea"        "III"            "D"              "NZ"
##  [5] "CP009511"       "Methanosarcina" "mazei"          "SarPi,"
##  [9] "complete"       "genome."        "2"              "31"
```

```
spacer.num <- sapply(1:length(index.numbers), function(i){
  index.numbers[[i]][(length(index.numbers[[i]]))]
})
array.num <- sapply(1:length(index.numbers), function(i){
  index.numbers[[i]][(length(index.numbers[[i]]))-1]
})

## add the metadata to new columns
dat <- cbind(dat, array.num, spacer.num)

##Checking to see if there are multiple similar target genomes

host.genomes <- unique(dat[,1])
length(host.genomes)
```

```
## [1] 119
```

```
target.genomes <- unique(dat[,3])
length(target.genomes)
```

```
## [1] 2007
```

```
host.arrays <- unique(dat[,c(1,20)])
nrow(host.arrays)
```

```
## [1] 142
```

Each host genome will be selected. From this, results will be generated for each of the target genomes that hits were found in. I will record the which spacers hit each genome, the order of the hits, the protospacer_start coordinates (taking the end to be more or less the same place), the strand information and the scores.

Every host genome and target genome will be given a unique row in the new data frame.

```
##set up new dataframe

##Get data frame size using the total number of host-targets there are.
x <- 0
```

```
for(i in host.genomes){
  ##select host data
  host.dat <- dat[dat[,1]==i, ]

  ##get list of target genomes
  target.genomes.i <- unique(host.dat[,3])
  x <- x + length(target.genomes.i)
}

##make data frame
genomes.summary <- data.frame(host.genome = vector(mode = "character", length = x),host.domain = vector
genomes.summary <- transform.df(genomes.summary)

## produce data frame of a summary of target genome information for each host genome.
m <- 0
for(i in host.genomes){
  ##select host data
  host.dat <- dat[dat[,1]==i, ]

  ##get information about host from the name
  host.metadata <- strsplit(host.dat[1,1], "_")
  host.domain <- host.metadata[[1]][1]
  Type <- host.metadata[[1]][2]
  Subtype <- paste(host.metadata[[1]][2:3], collapse = "_")
  ##get list of target genomes
  target.genomes.i <- unique(host.dat[,3])
  x <- x + length(target.genomes.i)
  ##loop where each target genome is selected
  for(j in target.genomes.i){
    m <- m + 1
    target.dat <- host.dat[host.dat[,3]==j,]
    target.dat <- transform.df(target.dat)
    target.dat <- target.dat[order(-target.dat[,21]),]
    ##write data to genomes summary
    genomes.summary[m,1] <- i
    genomes.summary[m,2] <- host.domain
    genomes.summary[m,3] <- Type
    genomes.summary[m,4] <- Subtype
    genomes.summary[m,5] <- j
    genomes.summary[m,6] <- paste(target.dat[,20], collapse = ",")
    genomes.summary[m,7] <- paste(target.dat[,21], collapse = ",")
    genomes.summary[m,8] <- paste(target.dat[,4], collapse = ",")
    genomes.summary[m,9] <- paste(target.dat[,9], collapse = ",")
    genomes.summary[m,10] <- paste(target.dat[,8], collapse = ",")
    genomes.summary[m,11] <- nrow(target.dat)
    genomes.summary[m,12] <- nrow(unique(target.dat[,c(20,21)]))
  }
}

##identify duplicates
genomes.summary.dup <- unique(genomes.summary[,c(1,2,3,4,6,7,10,11,12)])
nrow(genomes.summary)
```

```
## [1] 2769
```

```
nrow(genomes.summary.dup)
```

```
## [1] 816
```

Table 1: Cas genes assignment to CRISPR systems

	gene	systems	unique.y.n	subtypes.y.n
4	casR	CAS-I	TRUE	FALSE
5	cpf1	CAS-V	TRUE	TRUE
8	csx1	CAS-III	TRUE	FALSE
10	DEDDh	CAS-I	TRUE	FALSE
13	cas5a	CAS-I-A	TRUE	TRUE
14	cas5f	CAS-I-F	TRUE	TRUE
15	cas5u	CAS-I-U	TRUE	TRUE
17	csb2gr5	CAS-I-U	TRUE	TRUE
18	csc1gr5	CAS-I-D	TRUE	TRUE
21	csx10gr5	CAS-III-D	TRUE	TRUE
24	cas7f	CAS-I-F	TRUE	TRUE
28	cmr8gr7	CAS-III-B	TRUE	TRUE
29	csb1gr7	CAS-I-U	TRUE	TRUE
30	csc2gr7	CAS-I-D	TRUE	TRUE
33	csm5gr7	CAS-III-A	TRUE	TRUE
35	cas10c	CAS-III-C	TRUE	TRUE
36	cas10d	CAS-I-D	TRUE	TRUE
37	cas8a1	CAS-I-A	TRUE	TRUE
38	cas8a2	CAS-I-A	TRUE	TRUE
39	cas8a3	CAS-I-A	TRUE	TRUE
40	cas8a4	CAS-I-A	TRUE	TRUE
41	cas8a5	CAS-I-A	TRUE	TRUE
42	cas8a6	CAS-I-A	TRUE	TRUE
43	cas8a7	CAS-I-A	TRUE	TRUE
44	cas8a8	CAS-I-A	TRUE	TRUE
45	cas8b1	CAS-I-B	TRUE	TRUE
46	cas8b10	CAS-I-B	TRUE	TRUE
47	cas8b2	CAS-I-B	TRUE	TRUE
48	cas8b3	CAS-I-B	TRUE	TRUE
49	cas8b4	CAS-I-B	TRUE	TRUE
50	cas8b5	CAS-I-B	TRUE	TRUE
51	cas8b6	CAS-I-B	TRUE	TRUE
52	cas8b7	CAS-I-B	TRUE	TRUE
53	cas8b8	CAS-I-B	TRUE	TRUE
54	cas8b9	CAS-I-B	TRUE	TRUE
55	cas8c	CAS-I-C	TRUE	TRUE
56	cas8e	CAS-I-E	TRUE	TRUE
57	cas8f	CAS-I-F	TRUE	TRUE
58	cas8u1	CAS-I-U	TRUE	TRUE
59	cas8u2	CAS-I-U	TRUE	TRUE
60	csf1gr8	CAS-IV	TRUE	TRUE
61	cas11b	CAS-I-B	TRUE	TRUE
62	cas11d	CAS-I-D	TRUE	TRUE
64	csa5gr11	CAS-I-A	TRUE	TRUE
65	cse2gr11	CAS-I-E	TRUE	TRUE
68	cmr7	CAS-III-B	TRUE	TRUE
69	csb3	CAS-I-U	TRUE	TRUE
71	csx19	CAS-III-D	TRUE	TRUE
72	csx22	CAS-III-A	TRUE	TRUE
73	csx24	CAS-III-D	TRUE	TRUE
74	csx25	CAS-III-D	TRUE	TRUE
75	csx26	CAS-III-D	TRUE	TRUE
80	cas3f	CAS-I-F	TRUE	TRUE
81	cas3HD	CAS-I	TRUE	FALSE
82	csn2	CAS-II-A	TRUE	TRUE
83	csx23	CAS-III-D	TRUE	TRUE

Table 2: Gene Frequency

	ordered.gene.count	Freq
2	1	2024
3	2	1289
4	3	914
5	4	691
12	11	688
14	13	677
13	12	666
7	6	648
10	9	638
16	15	632
15	14	630
6	5	622
11	10	621
8	7	603
9	8	602
17	16	584
18	17	550
19	18	488
20	19	480
21	20	476

Table 3: Number of systems per genome

	systems.count	Freq
2	1	3500
3	2	2934
4	3	198
5	4	49
6	5	5
7	6	3

Table 4: Number of archaeal genomes with each subtype

	subtype	counts
1	CAS-I-A	72.00
2	CAS-I-B	136.00
3	CAS-I-C	15.00
4	CAS-I-D	36.00
5	CAS-I-E	16.00
6	CAS-I-F	0.00
7	CAS-I-U	4.00
8	CAS-II-A	0.00
9	CAS-II-B	0.00
10	CAS-II-C	2.00
11	CAS-III-A	52.00
12	CAS-III-B	5.00
13	CAS-III-C	14.00
14	CAS-III-D	26.00
15	CAS-IV	1.00
16	CAS-V	2.00

Table 5: Number of bacterial genomes with each subtype

	subtype	counts
1	CAS-I-A	100.00
2	CAS-I-B	932.00
3	CAS-I-C	1244.00
4	CAS-I-D	169.00
5	CAS-I-E	1737.00
6	CAS-I-F	907.00
7	CAS-I-U	289.00
8	CAS-II-A	529.00
9	CAS-II-B	122.00
10	CAS-II-C	839.00
11	CAS-III-A	418.00
12	CAS-III-B	2.00
13	CAS-III-C	64.00
14	CAS-III-D	337.00
15	CAS-IV	96.00
16	CAS-V	60.00

Table 6: Number of genomes with each subtype

	subtype	counts
1	CAS-I-A	172.00
2	CAS-I-B	1068.00
3	CAS-I-C	1259.00
4	CAS-I-D	205.00
5	CAS-I-E	1753.00
6	CAS-I-F	907.00
7	CAS-I-U	293.00
8	CAS-II-A	529.00
9	CAS-II-B	122.00
10	CAS-II-C	841.00
11	CAS-III-A	470.00
12	CAS-III-B	7.00
13	CAS-III-C	78.00
14	CAS-III-D	363.00
15	CAS-IV	97.00
16	CAS-V	62.00

Table 7: Frequency of different combinations of subtypes

	Subtypes	Freq
145	CAS-I-E, OTHER?	1289
183	CAS-II-C	841
102	CAS-I-C	838
162	CAS-I-F, OTHER?	685
46	CAS-I-B	573
176	CAS-II-A	408
171	CAS-I-U	153
184	CAS-III-A	147
182	CAS-II-B	122
190	CAS-III-D	83
108	CAS-I-C,CAS-I-E	82
146	CAS-I-E,CAS-I-F	80
12	CAS-I-A	79
47	CAS-I-B,CAS-I-C	74
89	CAS-I-B,CAS-III-A	70
133	CAS-I-D	68
192	CAS-IV	62
151	CAS-I-E,CAS-I-U	49
8	AS-I-F	47
98	CAS-I-B,CAS-III-D	42
75	CAS-I-B,CAS-I-E	41
127	CAS-I-C,CAS-III-A	33
154	CAS-I-E,CAS-II-A	32
67	CAS-I-B,CAS-I-D	31
118	CAS-I-C,CAS-I-F	31
130	CAS-I-C,CAS-III-D	31
160	CAS-I-E,CAS-III-D	31
124	CAS-I-C,CAS-II-A	29
193	CAS-V	29
144	CAS-I-D,CAS-III-D	25
96	CAS-I-B,CAS-III-C	23
156	CAS-I-E,CAS-III-A	22
177	CAS-II-A,CAS-III-A	20
6	AS-I-E	17
13	CAS-I-A,CAS-I-B	17
90	CAS-I-B,CAS-III-A,CAS-III-C	16
85	CAS-I-B,CAS-II-A	15
101	CAS-I-B,CAS-V	15
92	CAS-I-B,CAS-III-A,CAS-III-D	14
112	CAS-I-C,CAS-I-E,CAS-I-U	14
121	CAS-I-C,CAS-I-U	14
19	CAS-I-A,CAS-I-B,CAS-III-A	13
10	AS-III-A	12
82	CAS-I-B,CAS-I-U	11
168	CAS-I-F,CAS-III-D	11
4	AS-I-C	10
109	CAS-I-C,CAS-I-E,CAS-I-F	10
187	CAS-III-A,CAS-III-D	10
74	CAS-I-B,CAS-I-D,CAS-III-D	9
80	CAS-I-B,CAS-I-E,CAS-III-D	9

Table 8: Genome analysis summary

	Archaea	Bacteria
Number of genomes	453.00	31953.00
Number of genomes with cas	415.00	18621.00
Number of genomes with cas proteins and CRISPR arrays	313.00	6651.00
Number of genomes with a single system	159.00	3340.00
Number of genomes wth cas1 and cas2	220.00	3489.00