Compare GO Terms

Necessary packages:

```
library(dplyr)
library(stringr)
```

GO terms, direct from tools

eggNOG

```
eggnog <- read.delim("eggnogmap_results/diamond_annotations.tabular",
                     stringsAsFactors = FALSE,
                     header = FALSE)
# get all unique go terms from all peptides
eggnog_gos <- unlist(str_split(eggnog$V6, ",")) %>% unique()
eggnog_gos_clean <- eggnog_gos[str_sub(eggnog_gos, 1, 2) == "GO"]
```

We are going to write the lists to file, for later use in Python. First, I define a function `clean_write()` which writes a list without all the quotes and stuff:

```
clean_write <- function(object, file){
    write.table(object, file=file, quote = FALSE, row.names = FALSE, col.names = FALSE)
}
```

Write list to file for later use:

```
eggnog_list_file <- "go_lists/eggnog.tab"
clean_write(eggnog_gos_clean, file=eggnog_list_file)
```

MetaGOmics

```
metagomics_gos <- read.delim("metaGOmics_results/go_compare_149_150.txt",
                             stringsAsFactors = FALSE,
                             comment.char = "#")$GO.acc %>% unique()
metagomics_gos_clean <- metagomics_gos[str_sub(metagomics_gos, 1, 2) == "GO"]
clean_write(metagomics_gos_clean, file="go_lists/metagomics.tab")
```

MEGAN6

```
megan_go_lists <- read.delim("MEGAN_outputs/go_terms.txt",
                         header = TRUE,
                         stringsAsFactors = FALSE)$gos
megan_gos <- unlist(str_split(megan_go_lists, pattern = ",")) %>% unique()
megan_gos_clean <- megan_gos[str_sub(megan_gos, 1, 2) == "GO" & !is.na(megan_gos)]
clean_write(megan_gos_clean, file="go_lists/megan.tab")
```

Unipept

For Unipept, the results are divided into BP, MF, and CC files, so I combine the three ontologies for WS and NS.

```
unipept_results_NS <- paste('unipept_results/',
                            list.files("unipept_results/", pattern = "^737NS.*\\.csv"),
                            sep = "")
unipept_results_WS <- paste('unipept_results/',
                            list.files("unipept_results/", pattern = "^737WS.*\\.csv"),
                            sep = "")
unipeptNS <- lapply(unipept_results_NS, function(i) {
        read.delim(i, sep = ',', as.is = TRUE)}) %>%
    bind_rows() %>%
    select(-X) %>%
    rename(peptides = X.peptides)
unipeptWS <- lapply(unipept_results_WS, function(i) {
    read.delim(i, sep = ',', as.is = TRUE)}) %>%
    bind_rows() %>%
    select(-X) %>%
    rename(peptides = X.peptides)

unipept_all <- inner_join(unipeptNS, unipeptWS, by = c("GO.term", "Name")) %>%
    rename(countNS = peptides.x, countWS = peptides.y) %>%
    mutate(log2ratio = log(countWS/countNS))

unipept_gos <- unipept_all$GO.term %>% unique()
unipept_gos_clean <- unipept_gos[str_sub(unipept_gos, 1, 2) == "GO"]
clean_write(unipept_gos_clean, file="go_lists/unipept.tab")
```

MetaProteomeAnalyzer

```
# get all uniprot ids
ns_files <- list.files('mpa_results/NS', pattern = "_proteins.csv", full.names = TRUE)
protNS <- bind_rows(lapply(ns_files, function(i) read.delim(i, stringsAsFactors = FALSE))) %>%
    select("prot" = Protein.Accession) %>% unique()

ws_files <- list.files('mpa_results/WS', pattern = "_proteins.csv", full.names = TRUE)
protWS <- bind_rows(lapply(ws_files, function(i) read.delim(i, stringsAsFactors = FALSE))) %>%
    select("prot" = Protein.Accession) %>% unique()

protAll <- rbind(protNS, protWS) %>% unique()
# write for use on uniprot
clean_write(protAll, file="mpa_results/all_proteins.tab")

# these are the uniprot results from the above file
mpa_uniprot <- read.delim('mpa_results/uniprot_protein_results.tab', stringsAsFactors = FALSE)
mpa_gos <- str_trim(unlist(str_split(mpa_uniprot$Gene.ontology.IDs, ";")))
mpa_gos_clean <- mpa_gos[str_sub(mpa_gos, 1, 2) == "GO"] %>% unique()

clean_write(mpa_gos_clean, file="go_lists/mpa.tab")
```

Euler Diagram of Results

```
library(VennDiagram)
```

```
## Loading required package: grid
```

```
## Loading required package: futile.logger
```

```
gos <- list('megan' = megan_gos_clean,
            'metagomics' = metagomics_gos_clean,
            'unipept' = unipept_gos_clean,
            'eggnog' = eggnog_gos_clean,
            'mpa' = mpa_gos_clean)
```

GO terms: all parents

Each of the tools produces a list of GO terms, but we don't necessarily know how the tools assign terms - if a protein or peptide matches a single term, some tools might annotate that protein or peptide with the term and all of its ancestors, while some might annotate the tool with only the term itself. Thus, to reduce this kind of bias, we get all of the ancestors of all of the annotated terms, and produce another Venn diagram.

Download the current GO ontology:

```
wget http://purl.obolibrary.org/obo/go/go-basic.obo -O GO_files/go-basic.obo
```

Run python to get the list of all ancestors:

```
from goatools import obo_parser
import sys
go = obo_parser.GODag('GO_files/go-basic.obo')
```

```
## load obo file GO_files/go-basic.obo
## GO_files/go-basic.obo: fmt(1.2) rel(2018-05-03) 47,215 GO Terms
```

```
def set_of_all_ancestors(terms):
    all_ancestors = set(terms)
    for i in set(terms):
        if i in go.keys():
            all_ancestors.update(go[i].get_all_parents())
    return all_ancestors
def get_all_ancestors(infile, outfile):
    f = open(infile, 'r')
    gos = [x.strip() for x in f.readlines()]
    f.close()

    gos_with_ancestors = set_of_all_ancestors(gos)

    with open(outfile, 'w') as outf:
        for x in gos_with_ancestors:
            outf.write(x + '\n')
for i in ['eggnog', 'megan', 'metagomics', 'unipept', 'mpa']:
    infile = 'go_lists/' + i + '.tab'
    outfile = 'go_lists/' + i + '_parents.tab'
    get_all_ancestors(infile, outfile)
```

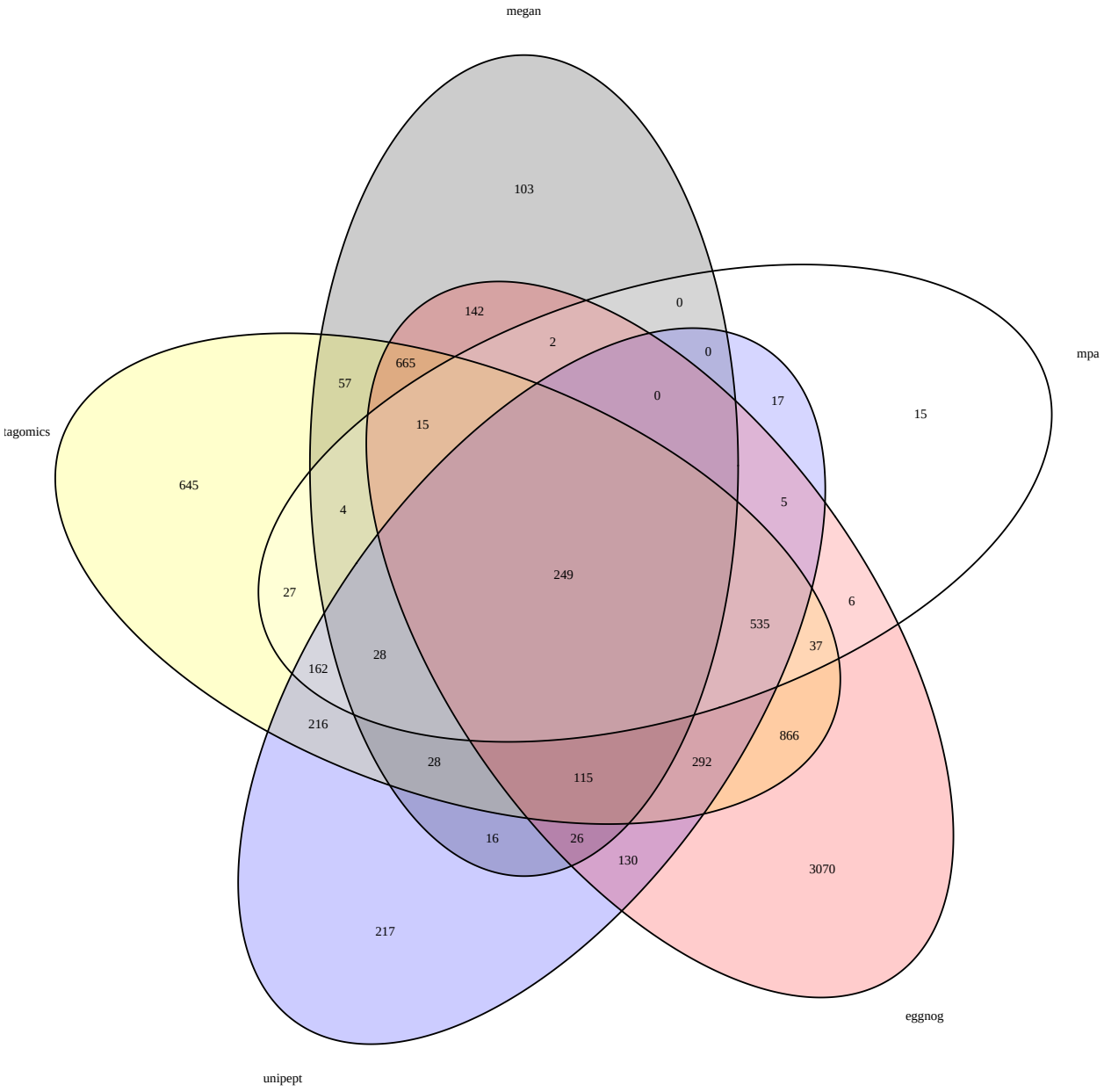
Read into R for Euler diagram:

```
methods <- c("megan", "metagomics", "unipept", "eggnog", 'mpa')
gos_ancestors <- lapply(methods,
    function(f) {
        "go" = read.delim(paste("go_lists/", f, '_parents.tab', sep = ""),
                          header = FALSE,
                          stringsAsFactors=FALSE)$V1
    }
)
names(gos_ancestors) <- methods
```

Results:

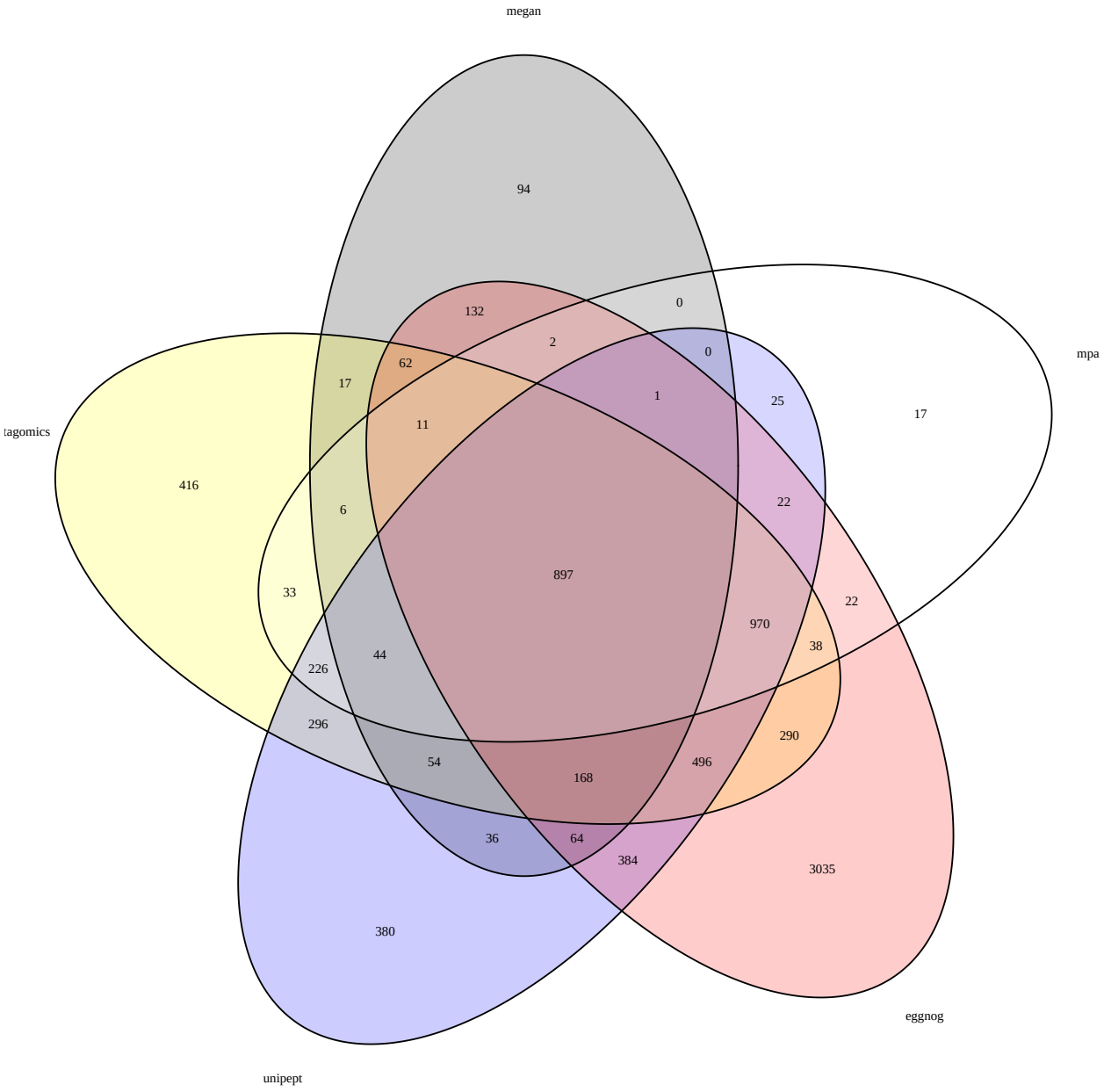
GO terms, direct from tools

```
grid.draw(venn.diagram(gos, NULL,
             fill = c("black", "yellow", "blue", "red", 'white'),
             alpha = 0.2))
```



GO terms, with all ancestors

```
grid.draw(venn.diagram(gos_ancestors, NULL,
                       fill = c("black", "yellow", "blue", "red", 'white'),
                       alpha = 0.2, width = 5000, height = 5000, force.unique = TRUE))
```


megan
94
0
132
2
0
mpa
62
17
1
25
17
11
tagomics
416
6
22
897
33
22
970
38
44
226
296
290
54
496
168
36
64
384
3035
380
eggnog
unipept