

Files for Enrichment Map

This analysis was done using [gProfiler](#). If you have already run gProfiler and have downloaded the file as a csv, this function will convert the gProfiler output to the Enrichment Map format.

```
library(tidyverse)
convert_gprofiler_file <- function(in_file, out_file) {
  enrichments <- read_csv(in_file)
  dplyr::filter(enrichments, grepl("^GO:", term_id)) %>%
    dplyr::mutate(FDR = adjusted_p_value) %>%
    dplyr::select(., term_id, term_name,
                  adjusted_p_value, FDR,
                  negative_log10_of_adjusted_p_value) %>%
    write_tsv(., file = out_file)
}

convert_gprofiler_file('data/go/Amp.gprofiler.csv',
                       'data/go/Amp.gprofiler.EM.tsv')
convert_gprofiler_file('data/go/Oxy.gprofiler.csv',
                       'data/go/Oxy.gprofiler.EM.tsv')
```

If you haven't already run gProfiler you can either run it through the web site or there is a [gProfiler R package](#).

```
# the gprofiler2 package isn't installed on the VMs
# This checks for the package and installs it if
# it isn't already installed
if (!require("gprofiler2", quietly = TRUE)) {
  install.packages("gprofiler2")
}

library(gprofiler2)
```

For the analysis, we need the gene ids for the differentially expressed genes.

```
amp_deseq_results <- read_tsv(file = "data/Amp.counts.tsv")
oxy_deseq_results <- read_tsv(file = "data/Oxy.counts.tsv")

# get differentially expressed genes
amp_sig_genes <- filter(amp_deseq_results, adjp < 0.05) %>%
  pull('Gene')
oxy_sig_genes <- filter(oxy_deseq_results, adjp < 0.05) %>%
  pull('Gene')
```

We also need a background set. For an RNA-seq experiment, we use all the genes that were tested for differential expression using DESeq2's independent filtering. This is all the genes in the DESeq2 results set where the adjusted pvalue is not NA.

```
# get background ids
# this checks if the adjusted p values are not NA
# and gets the gene ids
amp_background_ids <- filter(amp_deseq_results,
                             !is.na(adjp)) %>%
  pull('Gene')
oxy_background_ids <- filter(oxy_deseq_results,
                             !is.na(adjp)) %>%
  pull('Gene')
```

To run the query, we use the `gost` function. Check the [vignette](#) for more details on the arguments to `gost`.

The gProfiler [organism list](#) has a table showing what string to enter as the `organism` argument. For zebrafish, it's "drerio".

```
# enrichment is done using the gost function.
amp_results <- gost(
  query = amp_sig_genes,
  organism = 'drerio',
  significant = TRUE,
  correction_method = 'fdr',
  domain_scope = "custom",
  evcodes = TRUE,
  custom_bg = amp_background_ids,
  sources = c("GO:BP", "GO:CC", "GO:MF")
)

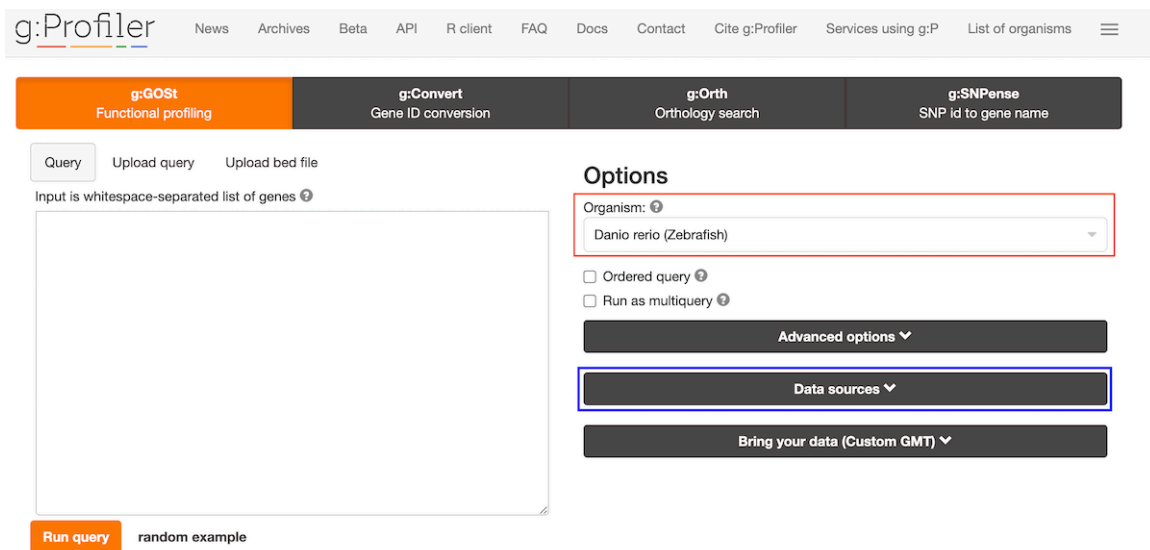
oxy_results <- gost(
  query = oxy_sig_genes,
  organism = 'drerio',
  significant = TRUE,
  correction_method = 'fdr',
  domain_scope = "custom",
  evcodes = TRUE,
  custom_bg = oxy_background_ids,
  sources = c("GO:BP", "GO:CC", "GO:MF")
)
```

To convert the results file to enrichment format, we need to create some new columns and select the correct columns in the right order.

```
convert_to_em <- function(gostres) {  
  gem <- dplyr::mutate(gostres$result,  
                      FDR = p_value,  
                      Phenotype = "+1") %>%  
  dplyr::select(term_id, term_name, p_value,  
                FDR, Phenotype, intersection) %>%  
  magrittr::set_colnames(c("GO.ID", "Description", "p.Val",  
                           "FDR", "Phenotype", "Genes"))  
  return(gem)  
}  
  
# convert to EM format and write to file  
convert_to_em(amp_results) %>%  
  write_tsv(file = 'data/go/Amp.gprofiler2.EM.tsv')  
convert_to_em(oxy_results) %>%  
  write_tsv(file = 'data/go/Oxy.gprofiler2.EM.tsv')
```

You also need a geneset (.gmt) file. These can be downloaded from gProfiler.

If you go to the website and select the correct organism and then click on Data Sources.



Then you can down a zip file of all the gmt files for each domain.

Data sources ^

select all

clear all

Show data versions

Gene Ontology

- ☒ GO molecular function
- ☒ GO cellular component
- ☒ GO biological process
- ☐ No electronic GO annotations ?

biological pathways

- ☐ KEGG
- ☐ Reactome
- ☒ WikiPathways

regulatory motifs in DNA

- ☐ TRANSFAC
- ☐ miRTarBase

protein databases

- ☒ Human Protein Atlas
- ☒ CORUM

Human phenotype ontology

- ☐ HP

name.gmt zip

combined name.gmt

ENSG.gmt zip

combined ENSG.gmt

You'll then need to concatenate the individual files together using the command line to make one .gmt file for Enrichment Map.

e.g.

```
cat drerio.GO:BP.ENSG.gmt drerio.GO:CC.ENSG.gmt drerio.GO:MF.ENSG.gmt  
> drerio.GO.ENSG.gmt
```

We can also use the 'counts.tsv' files to make the expressions and ranks files.

```
filter(amp_deseq_results, adjp < 0.05) %>%  
  dplyr::select(Gene, Name, ends_with('normalised count')) %>%  
  rename_with(.fn = function(x){  
    str_replace(x, " normalised count", "") } ) %>%  
  write_tsv(file = 'data/go/Amp.expressions.txt')  
  
filter(oxy_deseq_results, adjp < 0.05) %>%  
  dplyr::select(Gene, Name, ends_with('normalised count')) %>%  
  rename_with(.fn = function(x){  
    str_replace(x, " normalised count", "") } ) %>%  
  write_tsv(file = 'data/go/Oxy.expressions.txt')
```

```
filter(amp_deseq_results, adjp < 0.05) %>%  
  dplyr::select(Gene, adjp) %>%  
  arrange(adjp) %>%  
  write_tsv(file = 'data/go/Amp.rnk')  
  
filter(oxy_deseq_results, adjp < 0.05) %>%  
  dplyr::select(Gene, adjp) %>%  
  arrange(adjp) %>%  
  write_tsv(file = 'data/go/Oxy.rnk')
```