CBW pathways Workshops - example R notebooks

Ruth Isserlin

2023-04-24

Contents

Chapter 1

Index

Chapter 2

CBW Workshop example R Notebooks

Do you want to run the pathways and network analysis from R instead of doing everything mannually as demonstrated in the workshop?

Everything (almost!) that was discussed in the lectures and practicals can be done computationally through R.

We are using the **bookdown** package (Xie 2023) in this Workshop R Notebooks book, which was built on top of R Markdown and **knitr** (Xie 2015).

Chapter 3

Setup

3.1 Install R and RStudio

As with many open source projects, **R** is a constantly evolving language with regular updates. There is a major release once a year with patch releases through out the year. Often scripts and packages will work from one release to the next (ignoring pesky warnings that a package was compiled on a previous version of R is common) but there are exceptions. Some newer packages will only work on the latest version of **R** so sometimes the choice of upgrading or not using a new package might present themselves. Often, the amount of packages and work that is need to upgrade is not realized until the process has begun. This is where docker demonstrates it most valuable features. You can create a new instance based on the latest release of **R** and all your needed packages without having to change any of your current settings.

In order to use these notebooks supplied here you need to have:

- **R** installed on your computer and
- a list of packages. (including BiocManager, BiomaRt, gprofiler2, GSA)

Each notebook in this set will check for the required packages and install them if they are missing so at the base level you need to just have **R** installed.

There are many different ways you can use and setup **R**.

1. By simply installing **R** you can use it directly but
2. it is highly recommended that you also install and use RStudio which is an Integrate development environment (IDE) for **R**. You cannot just download RStudio and use it. It requires an installation of **R**.

You don't need to install R and RStudio though. You can also use **R** and RStudio through docker. **I highly recommend using docker instead**

3.2 Docker [Optional]

Changing versions and environments are a continuing struggle with bioinformatics pipelines and computational pipelines in general. An analysis written and performed a year ago might not run or produce the same results when it is run today. Recording package and system versions or not updating certain packages rarely work in the long run.

One the best solutions to reproducibility issues is containing your workflow or pipeline in its own coding environment where everything from the operating system, programs and packages are defined and can be built from a set of given instructions. There are many systems that offer this type of control including:

- Docker.
- Singularity

"A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another." ("What Is a Container?" n.d.)

Why are containers great for Bioiformatics?

- allows you to create environments to run bioinformatis pipelines.
- create a consistent environment to use for your pipelines.
- test modifications to the pipeline without disrupting your current set up.
- Coming back to an analysis years later and there is no need to install older versions of packages or programming languages. Simply create a container and re-run.

3.3 Install Docker

1. Download and install docker desktop.
2. Follow slightly different instructions for Windows or MacOS/Linux

3.3.1 Windows

- it might prompt you to install additional updates (for example - https://docs.Microsoft.com/en-us/windows/wsl/install-win10#step-4---download-the-linux-kernel-update-package) and require multiple restarts of your system or docker.
- launch docker desktop app.
- Open windows Power shell

- navigate to directory on your system where you plan on keeping all your code. For example: C:\USERS\risserlin\cbw_workshop_code
- Run the following command: (the only difference with the windows command is the way the current directory is written. ${PWD} instead of "$(pwd)")

```
docker run -e PASSWORD=changeit --rm \
  -v ${PWD}:/home/rstudio/projects -p 8787:8787 \
  risserlin/workshop_base_image
```

- Windows defender firewall might pop up with warning. Click on *Allow access*.
- In docker desktop you see all containers you are running and easily manage them.

3.3.2 MacOS / Linux

- Open Terminal
- navigate to directory on your system where you plan on keeping all your code. For example: /Users/risserlin/bcb420_code
- Run the following command: (the only difference with the windows command is the way the current directory is written. ${PWD} instead of "$(pwd)")

```
docker run -e PASSWORD=changeit --rm \
  -v "$(pwd)":/home/rstudio/projects -p 8787:8787 \
  --add-host "localhost:My.IP.address"
  risserlin/workshop_base_image
```

Chapter 4

Run g:profiler from R

4.1 Initialize variables and libraries

Detailed instructions on how to run g:Profiler programmatically from R

The parameters are set manually here but if you want to run the script from the command line then you can update the notebook to pull the parameters from the command line given arguments by updating each variable below to pull the values from the paramters - for example:

- variable <- params$parameter_name

For more details see - defining and using parameters and Knitting with parameters

```
#where to put all the generated files
working_dir <- "./generated_data/g_profiler"

# where to find the data files needed to run the analysis
data_dir <-  "./data"

# File name containing the list of genes to be used for analysis
genelist_file <- "Supplementary_Table1_Cancer_drivers.txt"

# default max size of the genesets for example -  250.  For this example we
# will be varying this parameter
max_gs_size <- 250

# default min size of the genesets for example -  3
min_gs_size <- 3
```

```
#min intersection between your genelist and the geneset - for example 3
min_intersection <- 3

# organism parameter used for g:profiler.
# First letter of first word in species name followed by
# the second word for example - hsapiens
organism <- "hsapiens"
```

```
#use library
tryCatch(expr = { library("gprofiler2")},
         error = function(e) {
           install.packages("gprofiler2")},
         finally = library("gprofiler2"))

tryCatch(expr = { library("GSA")},
         error = function(e) {
           install.packages("GSA")},
         finally = library("GSA"))
```

Create or set a directory to store all the generatd results

```
if(!dir.exists(params$working_dir)){
  dir.create(params$working_dir)
}
```

4.2 Load in Query set

Load in the set of genes that we will be running g:profiler with

```
 #load in the file
    current_genelist <- read.table(file =
                                     file.path(data_dir, genelist_file),
                                   header = FALSE,
                                   sep = "\t", quote = "",
                                   stringsAsFactors = FALSE)

  query_set <- current_genelist$V1
```

With regards to pathway sets there are two options when using g:Profiler -

- Use the genesets that are supplied by g:Profiler
- Upload your own genesets.

The most common reasons for supplying your own genesets is the ability to use up to date annotations or in-house annotations that might not be available in the public sphere yet. One of the greatest features of g:Profiler is that it is updated on a regular basis and most of the previous versions are available online ont the gprofiler archive.

The gprofielr2 -g:Profiler R implementation is a wrapper for the web version. You require an internet connection to get enrichment results.

4.3 Run g:profiler with supplied genesets

For detailed descriptions of all the parameters that can be specified for the gost g:profiler function see -here

For this query we are specifying -

- query - the set of genes of interest, as loaded in from the Supplementary_Table1_Cancer_drivers.txt file.
- significant - set to FALSE because we want g:Profiler to return all the results not just the ones that it deems significant by its perdetermined threshold.
- ordered_query - set to TRUE because for this set of genes they are ordered in order of their significance
- correction_method - set to fdr. by default g:Profiler uses g:Scs
- organism - set to "hsapiens" for homo sapiens. Organism names are constructed by concatenating the first letter of the name and the family name (according to gprofiler2 documentation)
- source - the geneset source databases to use for the analysis. We recommend using GO biological process (GO:BP), WikiPathways (WP) and Reactome (Reac) but there are additional sources you can add (GO molecular function or cellular component(GO:MF, GO:CC), KEGG, transcription factors (TF), microRNA targets (MIRNA), corum complexes (CORUM), Human protein atlas (HPA),Human phenotype ontology (HP))

```
gprofiler_results <- gost(query = query_set ,
                          significant=FALSE,
                          ordered_query = TRUE,
                          exclude_iea=FALSE,
                          correction_method = "fdr",
                          organism = organism,
                          source = c("REAC","WP","GO:BP"))
```

```
 #get the gprofiler results table
enrichment_results <- gprofiler_results$result
```

```
enrichment_results[1:5,]
```

```
##     query significant      p_value term_size query_size intersection_size
## 1 query_1        TRUE 9.272751e-39      5493        121               103
## 2 query_1        TRUE 1.694623e-36      5836        121               103
## 3 query_1        TRUE 1.276992e-35      5662        121               101
## 4 query_1        TRUE 4.128404e-35      2976        121                79
## 5 query_1        TRUE 1.198524e-34      3131        121                80
##   precision     recall    term_id source
## 1 0.8512397 0.01875114 GO:0031323  GO:BP
## 2 0.8512397 0.01764907 GO:0080090  GO:BP
## 3 0.8347107 0.01783822 GO:0051171  GO:BP
## 4 0.6528926 0.02654570 GO:0031325  GO:BP
## 5 0.6611570 0.02555094 GO:0051173  GO:BP
##                                                    term_name
## 1                  regulation of cellular metabolic process
## 2                   regulation of primary metabolic process
## 3          regulation of nitrogen compound metabolic process
## 4          positive regulation of cellular metabolic process
## 5 positive regulation of nitrogen compound metabolic process
##   effective_domain_size source_order
## 1                 21110         7510
## 2                 21110        18790
## 3                 21110        14316
## 4                 21110         7512
## 5                 21110        14318
##                                          parents
## 1             GO:0019222, GO:0044237, GO:0050794
## 2                         GO:0019222, GO:0044238
## 3                         GO:0006807, GO:0019222
## 4 GO:0009893, GO:0031323, GO:0044237, GO:0048522
## 5             GO:0006807, GO:0009893, GO:0051171
```

4.4 Download and load g:profiler geneset file

In order to create a proper Generic enrichment results file we will need a copy of the gmt file used by g:Profiler. (also to create an Enrichment map).

Download the gmt file used for this analysis from g:profiler

```
#the link to the gmt file is static no matter what version
gprofiler_gmt_url <-
  "https://biit.cs.ut.ee/gprofiler/static/gprofiler_full_hsapiens.name.gmt"
```

```
#get version info gprofiler as the gmt file is always associated with
# a specific version of g:profiler
gprofiler_version <- get_version_info(organism=organism)

gprofiler_gmt_filename <- file.path(working_dir,
                                   paste("gprofiler_full", organism,
                                     gprofiler_version$gprofiler_version,sep="_",
                                     ".name.gmt"))

if(!file.exists(gprofiler_gmt_filename)){
  download.file(url = gprofiler_gmt_url,
                destfile = gprofiler_gmt_filename)
}
```

To create a proper Generic enrichmentMap results file we need to include the list of genes that are associated with each geneset. To do that we need to know what genes are associated with each set and filter them by our query set. Load in the geneset definitions from the gmt file we just downloaded from g:profiler site.

```
#load in the g:profiler geneset file
capt_output <- capture.output(genesets_gprofiler <- GSA.read.gmt(
                                      filename = gprofiler_gmt_filename))

names(genesets_gprofiler$genesets) <- genesets_gprofiler$geneset.names
```

For the next module the name of the gmt file is - gprofiler_full_hsapiens.name.gmt but it is important to preserve the database version so in the future when we revisit these results for publication or results verfication we have the exact version used. Instead of creating a copy of the file (which can be pretty large) create a symbolic link to the file with the generic name.

```
#file.exists does not work for a symbolic link on my computer for some reason
# list the files in the directory and check if the symbolic link is there
#if(file.exists(file.path(working_dir, "gprofiler_full_hsapiens.name.gmt"))){
if(length(grep(x = list.files(file.path(working_dir)),
               pattern = "gprofiler_full_hsapiens.name.gmt",
               fixed = TRUE) > 0 )){

  file.remove(file.path(working_dir, "gprofiler_full_hsapiens.name.gmt"))
}
```

```
## [1] TRUE
```

```
file.symlink( gprofiler_gmt_filename,file.path(working_dir,
                                    "gprofiler_full_hsapiens.name.gmt"))
```

```
## [1] TRUE
```

```
# Given:
# query_genes - genes used for enrichment analysis (or as query)
#
# returns - the genes that overlap with the query set and part of the given
#           genesets
getGenesetGenes <- function(query_genes, subset_genesets){
  genes <- lapply(subset_genesets,FUN=function(x){intersect(x,query_genes)})

  # For each of the genes collapse to the comma separate text
  genes_collapsed <- unlist(lapply(genes,FUN=function(x){
                                                paste(x,collapse = ",")}))

  genes_collapsed_df <- data.frame(term_id = names(genes),
                             genes = genes_collapsed,stringsAsFactors = FALSE)

  return(genes_collapsed_df)
}
```

4.5 Filter results by geneset size

Filter the table to include just the columns that are required for the generic enrichment map file results GEM. Restrict the results to just the ones that have at least min_gs_size and less than max_gs_size terms and min_intersection size include only the term_id, term_name, p_value (and p_value again because the p_value is actually the corrected p-value. The output file does not contain the nominal p_value. For down stream analysis though it is expected to have both a p-value and a q-value so just duplicate the q-value as both p-value and q-value)

Vary the thresholds for max_gs_size just as we did in Module 2 lab -

- min_gs_size = 3
- max_gs_size = 10000
- max_gs_size = 1000
- max_gs_size = 250

```
# filer by params defined above
# by default we have set the max and min gs size to 250 and 3, respectively.
enrichment_results_mxgssize_250_min_3 <-
                          subset(enrichment_results,term_size >= min_gs_size &
                                    term_size <= max_gs_size &
                                    intersection_size >= min_intersection ,
                                  select = c(term_id,term_name,p_value,p_value ))

enrichment_results_mxgssize_1000_min_3 <-
                          subset(enrichment_results,term_size >= min_gs_size &
                                    term_size <= 1000 &
                                    intersection_size >= min_intersection ,
                                  select = c(term_id,term_name,p_value,p_value ))

enrichment_results_mxgssize_10000_min_3 <-
                          subset(enrichment_results,term_size >= min_gs_size &
                                    term_size <= 10000 &
                                    intersection_size >= min_intersection ,
                                  select = c(term_id,term_name,p_value,p_value ))
```

4.6 Create an output file of the results - Generic enrichment Map file from g:profiler gmt

The file requires -

- name
- description
- p-value
- q-value
- phenotyp
- list of genes (overlap of query set and original geneset)

The list of genes needs to be calculated using the gmt file and original query set. For each geneset found in the result find the overlap between the set of genes that are a part of the geneset and the query set.

```
# Given:
# gprofiler_results - results form g_profiler R function (filtered by desired)
# parameters
# gs - genes associated with each geneset, loaded in from a gmt file.
#
# returns - the properly formatted GEM file results
```

```
#
createGEMformat <- function(results, gs, query_genes){

  if(nrow(results) >0){

           #add phenotype to the results
          formatted_results <- cbind(results,1)

          # Add the genes to the genesets
          subset_genesets <- gs$genesets[
            which(gs$geneset.names
                  %in% results$term_id)]

          genes <- getGenesetGenes(query_genes, subset_genesets)

          formatted_results <- merge(formatted_results,genes,by.x=1, by.y=1)

          colnames(formatted_results) <- c("name","description","p-value",
                                           "q-value","phenotype","genes")

  }
  return(formatted_results)
}
```

```
enrichment_results_mxgssize_10000_min_3_GEMfile <- createGEMformat(
  enrichment_results_mxgssize_10000_min_3, genesets_gprofiler, query_set)

enrichment_results_mxgssize_1000_min_3_GEMfile <- createGEMformat(
  enrichment_results_mxgssize_1000_min_3, genesets_gprofiler, query_set)

enrichment_results_mxgssize_250_min_3_GEMfile <- createGEMformat(
  enrichment_results_mxgssize_250_min_3, genesets_gprofiler, query_set)
```

Output each of the above filtered files

```
#output the enrichment map file
write.table(enrichment_results_mxgssize_10000_min_3_GEMfile,
            file = file.path(working_dir,
                "gProfiler_hsapiens_lab2_results_GEM_maxterm10000.txt"),
            row.names = FALSE,
            col.names = TRUE,
            quote = FALSE)

#output the enrichment map file
write.table(enrichment_results_mxgssize_1000_min_3_GEMfile,
```

```
            file = file.path(working_dir,
                "gProfiler_hsapiens_lab2_results_GEM_maxterm1000.txt"),
            row.names = FALSE,
            col.names = TRUE,
            quote = FALSE)

#output the enrichment map file
write.table(enrichment_results_mxgssize_250_min_3_GEMfile,
            file = file.path(working_dir,
                "gProfiler_hsapiens_lab2_results_GEM_maxterm250.txt"),
            row.names = FALSE,
            col.names = TRUE,
            quote = FALSE)
```

4.7 Run g:profiler with your own genesets (example using BaderLab genesets)

4.8 Download and load Bader lab geneset file

Download the latest Bader lab genesets

```
gmt_url = "http://download.baderlab.org/EM_Genesets/current_release/Human/symbol/"

#list all the files on the server
filenames = RCurl::getURL(gmt_url)
tc = textConnection(filenames)
contents = readLines(tc)
close(tc)

#get the gmt that has all the pathways and does not include
# terms inferred from electronic annotations(IEA)
#start with gmt file that has pathways only
rx = gregexpr("(?<=<a href=\")(.*.GOBP_AllPathways_no_GO_iea.*.)(.gmt)(?=\">)",
  contents, perl = TRUE)
gmt_file = unlist(regmatches(contents, rx))

dest_gmt_file <- file.path(working_dir,gmt_file)

if(!file.exists(dest_gmt_file)){
  download.file(
    paste(gmt_url,gmt_file,sep=""),
    destfile=dest_gmt_file
```

```
  )
}
```

In order to use our results down stream in the Enrichment map we need to generate results files that we can pass to Enrichment Map.

Load in the GMT file

4.9 Filter Bader lab geneset file

The g:Profiler interface only allows for filtering genesets by size only after the analysis is complete. After the analysis is complete means the filtering is happening after Multiple hypothesis testing. Filtering prior to the analysis will generate more robust results because we exclude the uninformative large genesets prior to testing changing the sets that multiple hypothesis filtering will get rid of.

Create multiple gmt files with different filtering thresholds - remove * genesets greater than 250 genes * geneset greater than 1000 genes * geneset greater than 10000 genes

```
# Filter geneset GSA object by specified gs size threshold
#
# Given -
# genesets - in GSA object
# gs_sizes - list of all the sizes of the genesets found in the genesets
# filter_threshold - value to filter the geneset by.
#
# returns - filtered genesets in GSA object
filter_genesets <- function(genesets, gs_sizes, filter_threshold) {

  filtered_genesets <- genesets

  filtered_genesets$genesets <- genesets$genesets[
                      which(gs_sizes<filter_threshold)]
  filtered_genesets$geneset.names <- genesets$geneset.names[
                      which(gs_sizes<filter_threshold)]
  filtered_genesets$geneset.descriptions <- genesets$geneset.descriptions[
                      which(gs_sizes<filter_threshold)]

  return(filtered_genesets)
}

# You can not simply write a list of lists to a file in R.  In order
```

```
# to output the new geneset file you need to convert it ot a data.frame
# To do this convert the list of genes to a tab delmiated list in one column
# of the dataframe.
# format to write out to a file.
#
# Given -
# genesets - in GSA object

# returns - formatted genesets as data frame

format_genesets <- function(genesets) {

  collapsed_genesets <- data.frame(name=genesets$geneset.names,
                             description= genesets$geneset.description)
  collapsed_genesets$genes <- unlist(lapply(genesets$genesets,
                                            FUN=function(x){
                                             paste(x,collapse = "\t")
                                           }))

  return(collapsed_genesets)
}
```

The format of the GMT file is described https://software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Da
and consists of rows with the following

- Name
- Description
- tab delimited list of genes a part of this geneset

Write out the gmt file with genenames

```
#get the geneset sizes
gs_sizes_baderlab_sets <- lapply(genesets_baderlab_genesets$genesets,
                                 FUN = function(x){
                                              length(x)
                                                 })


# max 10,000
genesets_baderlab_genesets_max10000 <- filter_genesets(genesets_baderlab_genesets,
                                                       gs_sizes_baderlab_sets,
                                                       10000)

genesets_baderlab_genesets_max10000_filename <- gsub(x =dest_gmt_file,
                                                     pattern = "symbol" ,
```

```
                                                  replacement = "symbol_max10000"
                                                    )

if(!file.exists(genesets_baderlab_genesets_max10000_filename)){

  write.table(x = format_genesets(genesets_baderlab_genesets_max10000),
              file = genesets_baderlab_genesets_max10000_filename,
              quote = FALSE,sep = "\t",row.names = FALSE,
              col.names=TRUE)
}

#max gs size of 1,000
genesets_baderlab_genesets_max1000 <- filter_genesets(genesets_baderlab_genesets,
                                                      gs_sizes_baderlab_sets,
                                                      1000)
genesets_baderlab_genesets_max1000_filename <- gsub(x =dest_gmt_file,
                                                    pattern = "symbol" ,
                                                    replacement = "symbol_max1000"
                                                        )
if(!file.exists(genesets_baderlab_genesets_max1000_filename)){

  write.table(x = format_genesets(genesets_baderlab_genesets_max1000),
              file = genesets_baderlab_genesets_max1000_filename,
              quote = FALSE,sep = "\t",row.names = FALSE,
              col.names=TRUE)
}

#max gs size of 250
genesets_baderlab_genesets_max250 <- filter_genesets(genesets_baderlab_genesets,
                                                      gs_sizes_baderlab_sets,
                                                      250)


genesets_baderlab_genesets_max250_filename <- gsub(x =dest_gmt_file,
                                                   pattern = "symbol" ,
                                                   replacement = "symbol_max250"
                                                     )
if(!file.exists(genesets_baderlab_genesets_max250_filename)){
  write.table(x = format_genesets(genesets_baderlab_genesets_max250),
              file = genesets_baderlab_genesets_max250_filename,
              quote = FALSE,sep = "\t",row.names = FALSE,
              col.names=TRUE)
}
```

4.10 Upload the gmt files to gprofiler

In order to use your own genesets with g:Profiler you need to upload the the file to their server first. The function will return an ID that you need to specify in the organism parameter of the g:Profiler gost function call.

```
custom_gmt_max250 <- upload_GMT_file(
                        gmtfile=genesets_baderlab_genesets_max250_filename)
```

```
## Your custom annotations ID is gp__hEME_6OXR_gXM
## You can use this ID as an 'organism' name in all the related enrichment tests against this cus

## Just use: gost(my_genes, organism = 'gp__hEME_6OXR_gXM')
```

```
custom_gmt_max1000 <- upload_GMT_file(
                        gmtfile=genesets_baderlab_genesets_max1000_filename)
```

```
## Your custom annotations ID is gp__laL2_yq04_aLo
## You can use this ID as an 'organism' name in all the related enrichment tests against this cus

## Just use: gost(my_genes, organism = 'gp__laL2_yq04_aLo')
```

```
custom_gmt_max10000 <- upload_GMT_file(
                        gmtfile=genesets_baderlab_genesets_max10000_filename)
```

```
## Your custom annotations ID is gp__tsE5_EgE1_05A
## You can use this ID as an 'organism' name in all the related enrichment tests against this cus

## Just use: gost(my_genes, organism = 'gp__tsE5_EgE1_05A')
```

For this query we are specifying -

- query - the set of genes of interest, as loaded in from the Supplementary_Table1_Cancer_drivers.txt file.
- significant - set to FALSE because we want g:Profiler to return all the results not just the ones that it deems significant by its perdetermined threshold.
- ordered_query - set to TRUE because for this set of genes they are ordered in order of their significance
- correction_method - set to fdr. by default g:Profiler uses g:Scs
- organism - set to the custom_gmt ID (for this run it is - gp___hEME_6OXR_gXM) that we received when we uploaded our genetset file.

```
gprofiler_results_custom_max250 <- gost(query = query_set ,
                                        significant=FALSE,
                                    ordered_query = TRUE,
                                       exclude_iea=FALSE,
                                        correction_method = "fdr",
                                    organism = custom_gmt_max250
                                        )
```

```
## Detected custom GMT source request
```

```
gprofiler_results_custom_max1000 <- gost(query = query_set ,
                                        significant=FALSE,
                                    ordered_query = TRUE,
                                       exclude_iea=FALSE,
                                        correction_method = "fdr",
                                    organism = custom_gmt_max1000
                                        )
```

```
## Detected custom GMT source request
```

```
gprofiler_results_custom_max10000 <- gost(query = query_set ,
                                        significant=FALSE,
                                    ordered_query = TRUE,
                                       exclude_iea=FALSE,
                                        correction_method = "fdr",
                                    organism = custom_gmt_max10000
                                        )
```

```
## Detected custom GMT source request
```

```
 #get the gprofiler results table
enrichment_results_customgmt_max250 <- gprofiler_results_custom_max250$result
enrichment_results_customgmt_max1000 <- gprofiler_results_custom_max1000$result
enrichment_results_customgmt_max10000 <- gprofiler_results_custom_max10000$result


enrichment_results_customgmt_max250[1:5,]
```

```
##     query significant      p_value term_size query_size intersection_size
## 1 query_1        TRUE 1.385009e-22        64         78                17
## 2 query_1        TRUE 1.988040e-20        68         78                16
## 3 query_1        TRUE 6.846277e-15        54        101                13
## 4 query_1        TRUE 3.849668e-14        97        107                15
```

```
## 5 query_1        TRUE 1.546752e-13       159        108                17
##   precision    recall
## 1 0.2179487 0.2656250
## 2 0.2051282 0.2352941
## 3 0.1287129 0.2407407
## 4 0.1401869 0.1546392
## 5 0.1574074 0.1069182
##                                                                                term_id
## 1         HEAD AND NECK SQUAMOUS CELL CARCINOMA%WIKIPATHWAYS_20220510%WP4674%HOMO SAPIENS
## 2               GLIOBLASTOMA SIGNALING PATHWAYS%WIKIPATHWAYS_20220510%WP2261%HOMO SAPIENS
## 3 PATHWAYS AFFECTED IN ADENOID CYSTIC CARCINOMA%WIKIPATHWAYS_20220510%WP3651%HOMO SAPIENS
## 4                                    CELL CYCLE%WIKIPATHWAYS_20220510%WP179%HOMO SAPIENS
## 5                          REGULATION OF CELL CYCLE G1/S PHASE TRANSITION%GOBP%GO:1902806
##                                                    source
## 1 Human_GOBP_AllPathways_no_GO_iea_April_02_2023_symbol_max250
## 2 Human_GOBP_AllPathways_no_GO_iea_April_02_2023_symbol_max250
## 3 Human_GOBP_AllPathways_no_GO_iea_April_02_2023_symbol_max250
## 4 Human_GOBP_AllPathways_no_GO_iea_April_02_2023_symbol_max250
## 5 Human_GOBP_AllPathways_no_GO_iea_April_02_2023_symbol_max250
##                                      term_name effective_domain_size
## 1          Head and neck squamous cell carcinoma                 17047
## 2               Glioblastoma signaling pathways                  17047
## 3  Pathways affected in adenoid cystic carcinoma                 17047
## 4                                    Cell cycle                  17047
## 5 regulation of cell cycle G1/S phase transition                 17047
##   source_order parents
## 1         5002    NULL
## 2         5554    NULL
## 3         4968    NULL
## 4         4934    NULL
## 5        19009    NULL
```

Filter the table to include just the columns that are required for the generic enrichment map file results GEM. Restrict the results to just the ones that have at least min_gs_size and less than max_gs_size terms and min_intersection size include only the term_id, term_name, p_value (and p_value again because the p_value is actually the corrected p-value. The output file does not contain the nominal p_value. For down stream analysis though it is expected to have both a p-value and a q-value so just duplicate the q-value as both p-value and q-value)

```
# filer by params defined above
enrichment_results_customgmt_max250 <- subset(enrichment_results_customgmt_max250,
                                       term_size >= min_gs_size &
                                   term_size <= max_gs_size &
                                   intersection_size >= min_intersection ,
```

```
                                    select = c(term_id,term_name,p_value,p_value ))

enrichment_results_customgmt_max1000 <- subset(enrichment_results_customgmt_max1000,
                                        term_size >= min_gs_size &
                                      term_size <= max_gs_size &
                                      intersection_size >= min_intersection ,
                                    select = c(term_id,term_name,p_value,p_value ))

enrichment_results_customgmt_max10000 <- subset(enrichment_results_customgmt_max10000,
                                        term_size >= min_gs_size &
                                      term_size <= max_gs_size &
                                      intersection_size >= min_intersection ,
                                    select = c(term_id,term_name,p_value,p_value ))
```

4.11 Create an output file of the results - Generic enrichment Map file from Baderlab gmt

Use the same function defined above but instead of passing the genesets from the g_profiler gmt file pass the geneset defitnions we loaded in from the Baderlab gmt file.

```
enrichment_results_customgmt_GEM_max250 <- createGEMformat(
                                     enrichment_results_customgmt_max250,
                                     genesets_baderlab_genesets_max250,
                                     query_set)

#output the enrichment map file
write.table(enrichment_results_customgmt_GEM_max250,
                  file = file.path(
                    working_dir, "gProfiler_hspaiens_Baderlab_max250.txt"),
                  row.names = FALSE,
                  col.names = TRUE,
                  quote = FALSE)

enrichment_results_customgmt_GEM_max1000 <- createGEMformat(
                                    enrichment_results_customgmt_max1000,
                                    genesets_baderlab_genesets_max1000,
                                    query_set)

#output the enrichment map file
write.table(enrichment_results_customgmt_GEM_max1000,
```

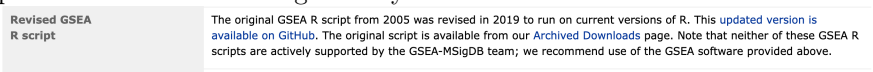
```
                  file = file.path(
                    working_dir, "gProfiler_hspaiens_Baderlab_max1000.txt"),
                  row.names = FALSE,
                  col.names = TRUE,
                  quote = FALSE)

enrichment_results_customgmt_GEM_max10000 <- createGEMformat(
                                    enrichment_results_customgmt_max10000,
                                    genesets_baderlab_genesets_max10000,
                                    query_set)

#output the enrichment map file
write.table(enrichment_results_customgmt_GEM_max10000,
                  file = file.path(
                    working_dir, "gProfiler_hspaiens_Baderlab_max1000.txt"),
                  row.names = FALSE,
                  col.names = TRUE,
                  quote = FALSE)
```

Chapter 5

Run GSEA from within R

This notebook is based largely on the original notebook published with EnrichmentMap Protocol(Reimand et al. 2019)

There is no package to run the original algorithm of GSEA(Subramanian et al. 2005) in R. There are many packages that have been published to imitate the process but none are recognized by The GSEA team.



5.1 Load in required libraries

```
#install required R and bioconductor packages
tryCatch(expr = { library("RCurl")},
         error = function(e) {
           install.packages("RCurl")},
         finally = library("RCurl"))
```

5.2 Configurable Parameters

In order to run GSEA automatically through the notebook you will need to download the gsea jar from here. Specify the exact path to the gsea jar in the parameters in order to automatically compute enrichments using GSEA.

If you are running this notebook using the baderlab workshop docker image then the image comes pre-installed with the gsea jar that you can use to run

gsea directly in the docker. The path to the GSEA jar in the docker is - /home/rstudio/GSEA_4.3.2/gsea-cli.sh

In order to run GSEA automatically you need to speciry the path to the gsea jar file. The gsea_jar needs to be the full path to the GSEA 4.3.2 directory that you downloaded from GSEA. for example /Users/johnsmith/GSEA_4.3.2/gsea-cli.sh

The parameters are set manually here but if you want to run the script from the command line then you can update the notebook to pull the parameters from the command line given arguments by updating each variable below to pull the values from the paramters - for example:

- variable <- params$parameter_name

For more details see - defining and using parameters and Knitting with parameters

```
#path to GSEA jar
gsea_jar <- "/home/rstudio/GSEA_4.3.2/gsea-cli.sh"
```

Set the working directory as the directory to the directory where you downloaded all protocol files. For example /User/JohnSmith/EMProtocolFiles/data

```
#directory where all the data files are found.  For example -   ./data/
working_dir <- "./data/"

#The name to give the analysis in GSEA - for example Mesen_vs_Immuno
analysis_name <- "Mesen_vs_Immuno"

#rank file to use in GSEA analysis.
#For example - MesenchymalvsImmunoreactive_edger_ranks.rnk
rnk_file <- "MesenchymalvsImmunoreactive_edger_ranks.rnk"

#run_gsea - true/false
# This parameter is for the compilation of the notebook.
run_gsea <- FALSE
```

5.3 Download the latest pathway definition file

Only Human, Mouse, Rat, and Woodchuck gene set files are currently available on the baderlab downloads site. If you are working with a species other than human (and it is either rat,mouse or woodchuck) change the gmt_url below to the correct species. Check here to see all available species.

To create your own GMT file using Ensembl see Create GMT file from Ensembl

```
gmt_url = "http://download.baderlab.org/EM_Genesets/current_release/Human/symbol/"

#list all the files on the server
filenames = getURL(gmt_url)
tc = textConnection(filenames)
contents = readLines(tc)
close(tc)

#get the gmt that has all the pathways and does not include terms
# inferred from electronic annotations(IEA)
#start with gmt file that has pathways only and GO Biological Process only.
rx = gregexpr("(?<=<a href=\")(.*.GOBP_AllPathways_no_GO_iea.*.)(.gmt)(?=\">)",
  contents, perl = TRUE)
gmt_file = unlist(regmatches(contents, rx))

dest_gmt_file <- file.path(working_dir,gmt_file )

#check if this gmt file already exists
if(!file.exists(dest_gmt_file)){
  download.file(
    paste(gmt_url,gmt_file,sep=""),
    destfile=dest_gmt_file
  )
}
```

5.4 Run GSEA

(GSEA)[http://software.broadinstitute.org/gsea/index.jsp] is a stand alone java program with many customizable options. It can be easily run through its integrated user interface. To make this a seemless pipeline we can run GSEA from the command line with a set of options. Any of the supplied options can be customized and there are many additional options that can be specified. For more details see (here)[http://software.broadinstitute.org/gsea/doc/GSEAUserGuideTEXT.htm#_Running_GSEA_from]

In the below command the following options have been specified:

- rnk - path to the rank file
- gmx - path to the gene set definition (gmt) file
- collapse - true/false indicates whether the expression/rnk file needs to be collapsed from probes to gene symbols
- nperm - number of permutations

- scoring_scheme -
- rpt_label - name of the directory with output
- rnd_seed - random seed to use
- set_max - maximum size for individual gene sets. In GSEA interface this is set to 500 but we prefer to use a more stringent setting of 200.
- set_min - minimum size for individual gene sets
- zip_report - true/false to zip output directory
- out - directory where to place the result directory.

```
if(run_gsea){
  command <- paste("",gsea_jar,
                   "GSEAPreRanked -gmx", dest_gmt_file,
                   "-rnk" ,file.path(working_dir,rnk_file),
                   "-collapse false -nperm 1000 -scoring_scheme weighted",
                   "-rpt_label ",analysis_name,
                   "  -plot_top_x 20 -rnd_seed 12345  -set_max 200",
                   " -set_min 15 -zip_report false ",
                   " -out" ,working_dir,
                   " > gsea_output.txt",sep=" ")
  system(command)
}
```

Chapter 6

Create GMT file from Ensembl

The Baderlab geneset download site is an updated resource for geneset files from GO, Reactome, WikiPathways, Pathbank, NetPath, HumanCyc, IOB, … many others that can be used in g:Profiler or GSEA and many other enrichment tools that support the gmt format.

Unfortunately genesets are only supplied for:

- Human
- Mouse
- Rat
- Woodchuck

If you are working in a different species you will need to generate your own gmt file. The best way to do this is through ensembl. Ensembl doesn't have annotations for all the pathway databases listed above but it has annotations for most species from GO.

The parameters are set in the params option on this notebook but you can also manually set them here.

```
# for example - working_dir <- "./genereated_data"
working_dir <- params$working_dir

# for example - species <- "horse"
species <- params$species

# for example - ensembl_dataset <- "ecaballus_gene_ensembl"
ensembl_dataset <- params$ensembl_dataset
```

```
#use library
#make sure biocManager is installed
tryCatch(expr = { library("BiocManager")},
         error = function(e) {
           install.packages("BiocManager")},
         finally = library("BiocManager"))


tryCatch(expr = { library("biomaRt")},
         error = function(e) {
           BiocManager::install("biomaRt")},
         finally = library("biomaRt"))
```

6.1 Load Libraries

Create or set a directory to store all the generatd results

```
if(!dir.exists(params$working_dir)){
  dir.create(params$working_dir)
}
```

6.2 Set up Biomart connection

Connect to Biomart

```
ensembl <- useMart("ensembl")
```

Figure out which dataset you want to use - for some species there might be a few datasets to choose from. Not all of the datasets have common namesa associated with them. For example, if you search for 'yeast' nothing will be returned but if you look for Saccharomyces or cerevisiae you will be able to find it.

```
all_datasets <- listDatasets(ensembl)

#get all the datasets that match our species definition
all_datasets[grep(all_datasets$description,
                  pattern=species,
                  ignore.case = TRUE),]
```

```
##                      dataset                                description
```

```
## 60       ecaballus_gene_ensembl                    Horse genes (EquCab3.0)
## 76          hcomes_gene_ensembl  Tiger tail seahorse genes (H_comes_QL1_v1)
## 164 rferrumequinum_gene_ensembl Greater horseshoe bat genes (mRhiFer1_v1.p)
##            version
## 60       EquCab3.0
## 76  H_comes_QL1_v1
## 164  mRhiFer1_v1.p
```

If you know the ensembl dataset that you want to use you can specify it in the parameters above or grab from the above table the dataset of the species that you are interested in.

```
ensembl = useDataset(ensembl_dataset,mart=ensembl)
```

6.3 Get species GO annotations

Get the GO annotations for our species

```
go_annotation <- getBM(attributes = c("external_gene_name",
                                      "ensembl_gene_id",
                                      "ensembl_transcript_id",
                                      "go_id",
                                      "name_1006",
                                      "namespace_1003",
                                      "go_linkage_type"),
                       filters=list(biotype='protein_coding'), mart=ensembl);

#get just the go biological process subset
#####
# Get rid of this line if you want to include all of go and not just biological process
#####
go_annotation_bp <- go_annotation[which(
  go_annotation$namespace_1003 == "biological_process"),]

#compute the unique pathway sets
go_pathway_sets <- aggregate(go_annotation_bp[,1:5],
                             by = list(go_annotation_bp$go_id),
                             FUN = function(x){list(unique(x))})

#unlist the go descriptions
go_pathway_sets$name_1006 <- apply(go_pathway_sets,1,FUN=function(x){
   paste(gsub(unlist(x$name_1006),pattern= "\"",
              replacement = ""),collapse = "")})
```

There are two identifiers that you can choose from in the above table * external_symbols * ensembl_ids

Each of these is stored as a list in the dataframe. In order to convert it to the right format for the gmt file we need to convert the list to string of tab delimited strings. (unfortunately there is no streaightforward way to write out a dataframe's column of lists.)

```
go_pathway_sets[1:3,"external_gene_name"]
```

```
## [[1]]
## [1] "MEF2A"    "SLC25A36" "OPA1"     "MGME1"    "SLC25A33" "TYMP"     "AKT3"
## [8] "PIF1"
##
## [[2]]
## [1] "GNRH1" "GNRH2" "LIN9"
##
## [[3]]
## [1] "ERCC6" "ERCC8" "LIG4"  "APLF"  "APTX"  "XRCC1" "SIRT1" "XNDC1"
```

```
go_pathway_sets[1:3,"ensembl_gene_id"]
```

```
## [[1]]
## [1] "ENSECAG00000011593" "ENSECAG00000010094" "ENSECAG00000024248"
## [4] "ENSECAG00000012675" "ENSECAG00000016862" "ENSECAG00000001072"
## [7] "ENSECAG00000019722" "ENSECAG00000005316"
##
## [[2]]
## [1] "ENSECAG00000010664" "ENSECAG00000039220" "ENSECAG00000014325"
##
## [[3]]
## [1] "ENSECAG00000014160" "ENSECAG00000018335" "ENSECAG00000003257"
## [4] "ENSECAG00000013246" "ENSECAG00000012674" "ENSECAG00000014127"
## [7] "ENSECAG00000013909" "ENSECAG00000042118"
```

6.4 Format results into GMT file

Convert column of lists to a tab delimited string of gene names

```
go_pathway_sets$collapsed_genenames <- apply(go_pathway_sets,1,
                                             FUN=function(x){
   paste(gsub(unlist(x$external_gene_name),pattern= "\"",
              replacement = ""),collapse = "\t")
})
```

Convert column of lists to a tab delimited string of gene names

```
go_pathway_sets$collapsed_ensemblids <- apply(go_pathway_sets,1,
                                              FUN=function(x){
   paste(gsub(unlist(x$ensembl_gene_id),pattern= "\"",
              replacement = ""),collapse = "\t")
})
```

The format of the GMT file is described https://software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Da
and consists of rows with the following

- Name
- Description
- tab delimited list of genes a part of this geneset

Write out the gmt file with genenames

```
gmt_file_genenames <- go_pathway_sets[,c("Group.1","name_1006",
                                         "collapsed_genenames")]
colnames(gmt_file_genenames)[1:2] <- c("name","description")

gmt_genenames_filename <- file.path(params$working_dir, paste(species,ensembl_dataset,"GO_geneset

write.table(x = gmt_file_genenames,file = gmt_genenames_filename,
            quote = FALSE,sep = "\t",row.names = FALSE,
            col.names=TRUE)
```

Write out the gmt file with ensembl ids

```
gmt_file_ensemblids <- go_pathway_sets[,c("Group.1","name_1006",
                                          "collapsed_ensemblids")]
colnames(gmt_file_ensemblids)[1:2] <- c("name","description")

gmt_ensemblids_filename <- file.path(params$working_dir, paste(species,ensembl_dataset,"GO_genese

write.table(x = gmt_file_ensemblids,file = gmt_ensemblids_filename,
            quote = FALSE,sep = "\t",row.names = FALSE,
            col.names=TRUE)
```

Reimand, Jüri, Ruth Isserlin, Veronique Voisin, Mike Kucera, Christian Tannus-Lopes, Asha Rostamianfar, Lina Wadi, et al. 2019. "Pathway Enrichment Analysis and Visualization of Omics Data Using g: Profiler, GSEA, Cytoscape and EnrichmentMap." *Nature Protocols* 14 (2): 482–517.

Subramanian, Aravind, Pablo Tamayo, Vamsi K Mootha, Sayan Mukherjee, Benjamin L Ebert, Michael A Gillette, Amanda Paulovich, et al. 2005.

"Gene Set Enrichment Analysis: A Knowledge-Based Approach for Interpreting Genome-Wide Expression Profiles." *Proceedings of the National Academy of Sciences* 102 (43): 15545–50.

"What Is a Container?" n.d. *Docker*. https://www.docker.com/resources/what-container.

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. http://yihui.name/knitr/.

———. 2023. *Bookdown: Authoring Books and Technical Documents with r Markdown*. https://CRAN.R-project.org/package=bookdown.