CBW pathways Workshops - example R notebooks

Ruth Isserlin

2023-04-03

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

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. 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**. By simply installing **R** you can use it directly but 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**

3.1 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.

3.2 Install Docker

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

3.2.1 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.

3.2.2 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

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

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

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

data_dir <- params$data_dir

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

# max size of the genesets for example -  350
max_gs_size <- params$max_gs_size

# max size of the genesets for example - 3
min_gs_size <- params$min_gs_size

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

```r
#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)
}
```

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.

4.1 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 = "hsapiens",
                          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 1.426353e-37      5653        121               103
## 2 query_1        TRUE 3.391992e-36      5882        121               103
## 3 query_1        TRUE 7.172333e-36      3097        121                81
## 4 query_1        TRUE 2.511953e-35      5724        121               101
## 5 query_1        TRUE 7.298888e-35      3540        121                84
##   precision     recall    term_id source
## 1 0.8512397 0.01822041 GO:0031323  GO:BP
## 2 0.8512397 0.01751105 GO:0080090  GO:BP
## 3 0.6694215 0.02615434 GO:0031325  GO:BP
## 4 0.8347107 0.01764500 GO:0051171  GO:BP
## 5 0.6942149 0.02372881 GO:0010604  GO:BP
##                                                 term_name effective_domain_size
## 1               regulation of cellular metabolic process                 21128
## 2                regulation of primary metabolic process                 21128
## 3      positive regulation of cellular metabolic process                 21128
## 4      regulation of nitrogen compound metabolic process                 21128
## 5 positive regulation of macromolecule metabolic process                 21128
##   source_order                                       parents
## 1         7549             GO:0019222, GO:0044237, GO:0050794
## 2        18924                         GO:0019222, GO:0044238
## 3         7551 GO:0009893, GO:0031323, GO:0044237, GO:0048522
## 4        14399                         GO:0006807, GO:0019222
## 5         4369             GO:0009893, GO:0043170, GO:0060255
```

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 <- 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 ))
```

4.2 Run g:profiler with your own genesets

4.3 Upload the gmt file 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 <- upload_GMT_file(gmtfile=dest_gmt_file)
```

```
## Your custom annotations ID is gp__A9gE_WMmN_u4o
## You can use this ID as an 'organism' name in all the related enrichment tests agains

## Just use: gost(my_genes, organism = 'gp__A9gE_WMmN_u4o')
```

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___A9gE_WMmN_u4o) that we received when we uploaded our genetset file.

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

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

```
## No results to show
## Please make sure that the organism is correct or set significant = FALSE
```

4.4 Create enrichment Results files

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

Chapter 5

Applications

Some *significant* applications are demonstrated in this chapter.

5.1 Example one

5.2 Example two

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
```

```r
#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

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

6.2 Set up Biomart connection

Connect to Biomart

```r
ensembl <- useMart("ensembl",host = "https://asia.ensembl.org")
```

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.

```r
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 streightforward 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)
```

"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.