Introduction to Bulk RNAseq data analysis

Annotation of Differential Expression Results

Last modified: 30 Sep 2024

Contents

Adding annotation to the DESeq2 results

We have a list of significantly deferentially expressed genes, but the only annotation we can see is the Ensembl Gene ID, which is not very informative.

There are a number of ways to add annotation. One method is to do this using a Bioconductor annotation package. These packages which are re-built every periodically with the latest annotations. These packages are listed on the annotation section of the Bioconductor, and are installed in the same way as regular Bioconductor packages.

Another approach is to use `biomaRt`, an interface to the BioMart resource. Using BioMart ensures that you are able to get the latest annotations for the GeneIDs, and can match the version of the gene annotation that was used for read counting.

A third method is to use `AnnotationHub`, this is like the bioconductor packages but in an online database like `bioMaRt`. They keep them slightly more up to date than the standard bioconductor packages and each time you use them the results are cached on your machine.

We could also simply download an annotation table from a suitable repository, for example Gencode provided tables for human and mouse that map Ensembl IDs to Gene Symbol and Entrez IDs. This may be the most straightforward method in some cases, but it is not as flexible as the other methods.

Today we will demonstrate how to use `AnnotationHub`. A workflow for annotation with biomaRt is included in the extended materials section accessible on the course website.

First we will load the differential gene expression results obtained from the interaction model for Infected vs Uninfected at day 11.

```
library(AnnotationHub)
library(AnnotationDbi)
library(ensembldb)
library(DESeq2)
library(tidyverse)
```

```
results.d11 <- readRDS("RObjects/DESeqResults.interaction_d11.rds")
```

Query the database

We need to get the correct database from `AnnotationHub`. We make the instance of the database in our R session (the first time we do this it will create a local cache on your machine so that repeat queries are very quick).

```
ah <- AnnotationHub()
ah
```

```
## AnnotationHub with 71634 records
## # snapshotDate(): 2024-04-30
## # $dataprovider: Ensembl, BroadInstitute, UCSC, ftp://ftp.ncbi.nlm.nih.gov/g...
## # $species: Homo sapiens, Mus musculus, Drosophila melanogaster, Rattus norv...
## # $rdataclass: GRanges, TwoBitFile, BigWigFile, EnsDb, Rle, OrgDb, SQLiteFil...
## # additional mcols(): taxonomyid, genome, description,
## #   coordinate_1_based, maintainer, rdatadateadded, preparerclass, tags,
## #   rdatapath, sourceurl, sourcetype
## # retrieve records with, e.g., 'object[["AH5012"]]'
##
##             title
##   AH5012   | Chromosome Band
##   AH5013   | STS Markers
##   AH5014   | FISH Clones
##   AH5015   | Recomb Rate
##   AH5016   | ENCODE Pilot
##   ...        ...
##   AH117051 | Ensembl 112 EnsDb for Xiphophorus maculatus
##   AH117052 | Ensembl 112 EnsDb for Xenopus tropicalis
##   AH117053 | Ensembl 112 EnsDb for Zonotrichia albicollis
##   AH117054 | Ensembl 112 EnsDb for Zalophus californianus
##   AH117055 | Ensembl 112 EnsDb for Zosterops lateralis melanops
```

As you can see `ah` contains huge amounts of information and it is constantly changing. This is why it gives us the snapshot date so we know when our cached version is from. The `ah` object actually online contains pointers to where all the information is online and we don't want to download all of them as it would take a very long time and we don't need all of it.

The object is a vector and you can get information about a single resource by indexing it with a single bracket `[` or actually download a resource with a double bracket `[[`.

```
ah[1]
```

```
## AnnotationHub with 1 record
## # snapshotDate(): 2024-04-30
## # names(): AH5012
## # $dataprovider: UCSC
## # $species: Homo sapiens
## # $rdataclass: GRanges
## # $rdatadateadded: 2013-03-26
## # $title: Chromosome Band
## # $description: GRanges object from UCSC track 'Chromosome Band'
## # $taxonomyid: 9606
## # $genome: hg19
## # $sourcetype: UCSC track
## # $sourceurl: rtracklayer://hgdownload.cse.ucsc.edu/goldenpath/hg19/database...
## # $sourcesize: NA
## # $tags: c("cytoBand", "UCSC", "track", "Gene", "Transcript",
## #   "Annotation")
## # retrieve record with 'object[["AH5012"]]'
```

Now we can query the database for the annotation we need. We can use the `query` function to search for the database we need by providing it with a vector of search terms. In our case we will look for:

- The Ensembl database - from which we obtained our reference:`EnsDb`
- The species - `Mus musculus`
- The specific Ensembl release version - `102`

Genomic and transcriptomic databases undergo a lot of changes and updates, so it is important to note which version of the database you are using when you download your reference. The current course materials use Ensembl release 102 (a little old, but we will update soon).

```
dbResult <- query(ah, c("EnsDb", "Mus musculus", "102"))
dbResult[1]
```

```
## AnnotationHub with 1 record
## # snapshotDate(): 2024-04-30
## # names(): AH89211
## # $dataprovider: Ensembl
## # $species: Mus musculus
## # $rdataclass: EnsDb
## # $rdatadateadded: 2020-10-27
## # $title: Ensembl 102 EnsDb for Mus musculus
## # $description: Gene and protein annotations for Mus musculus based on Ensem...
## # $taxonomyid: 10090
## # $genome: GRCm38
## # $sourcetype: ensembl
## # $sourceurl: http://www.ensembl.org
## # $sourcesize: NA
## # $tags: c("102", "AHEnsDbs", "Annotation", "EnsDb", "Ensembl", "Gene",
## #   "Protein", "Transcript")
## # retrieve record with 'object[["AH89211"]]'
```

This returns one record of a database in Annotation Hub that matches our requirements. We can download it using the `[[` operator.

Download the relevant resource

```
MouseEnsDb <- ah[[dbResult$ah_id]]
```

This database contains the entire gene annotation from Ensembl release 102 for mouse. It includes descriptions of genes, transcripts, exons, UTRs etc.

We can turn the whole gene-level annotation table into a data frame so we can work with it using the tidyverse suite of tools.

```
annotations <- genes(MouseEnsDb, return.type = "data.frame")

t(annotations[1, ])
```

```
##                          1
## gene_id                  "ENSMUSG00000102693"
## gene_name           "4933401J01Rik"
## gene_biotype        "TEC"
## gene_seq_start      3073253
## gene_seq_end        3074322
## seq_name            "1"
## seq_strand          1
## seq_coord_system    "chromosome"
## description         "RIKEN cDNA 4933401J01 gene [Source:MGI Symbol;Acc:MGI:1918292]"
## gene_id_version     "ENSMUSG00000102693.1"
## canonical_transcript "ENSMUST00000193812"
## symbol              "4933401J01Rik"
## entrezid            NA
```

Subset the annotation to the genes and columns of interest

We are just going to keep the gene symbol, the description, and also the Entrez ID (which we will need later on for gene set testing).

```
annot <- annotations %>%
    select(gene_id, gene_name, description, entrezid) %>%
    filter(gene_id %in% rownames(results.d11))
```

NOTE: You may get an error with this command that looks like:

```
Error in (function (classes, fdef, mtable)  :
  unable to find an inherited method for function 'select' for signature '"data.frame"'
```

This is due to the `select` function from **dplyr** (part of **tidyverse**) being masked by the `select` function from one of the annotation packages. This will have happened because the annotation package was loaded after the **tidyverse**. You can either restart your R session and reload the required packages, this time being sure to load **tidyverse** last, or just use `dplyr::select` to explicitly use the `select` function from **dplyr**:

```
annot <- annotations %>%
    dplyr::select(gene_id, gene_name, description, entrezid) %>%
    dplyr::filter(gene_id %in% rownames(results.d11))
```

Missing annotations

Let's inspect the annotation.

```
length(annot$entrezid)
```

```
## [1] 20091
```

```
length(unique(annot$entrezid))
```

```
## [1] 17278
```

```
sum(is.na(annot$entrezid))
```

```
## [1] 2782
```

There nearly 3000 genes with missing Entrez IDs. Gene/transcript/protein IDs mapping between different databases not always perfect.

Although majority of IDs map between databases, small subset may not have matching ID or may have more than one match. This is because feature identification algorithms, naming methodologies and versions may differ among databases. For instance NCBI and HGNC give same ID for different gene versions, whereas Ensembl assigned separate IDs for gene versions. Read interesting discussion on biostars.

There are some Ensembl IDs with no EntrezID. These gene ids has no corresponding Entrez ID in the `EnsDb` database package. The Ensembl and Entrez databases don't match on a 1:1 level although they have started taking steps towards consolidating in recent years.

Duplicated annotations

In addition to the missing Entrez, there are number of cases in which multiple Ensembl IDs map to the same Entrez ID.

```
dupEntrez <- annot %>%
  filter(!is.na(entrezid)) %>%
  add_count(entrezid) %>%
  arrange(entrezid) %>%
  filter(n > 1)
head(dupEntrez)
```

```
##               gene_id gene_name
## 1 ENSMUSG00000026064    Ptp4a1
## 2 ENSMUSG00000117310    Ptp4a1
## 3 ENSMUSG00000115958   Gm28040
## 4 ENSMUSG00000116158     Kiss1
## 5 ENSMUSG00000102976   Zc3h11a
## 6 ENSMUSG00000116275   Zc3h11a
##                                                                description
## 1     protein tyrosine phosphatase 4a1 [Source:MGI Symbol;Acc:MGI:1277096]
## 2     protein tyrosine phosphatase 4a1 [Source:MGI Symbol;Acc:MGI:1277096]
## 3                predicted gene, 28040 [Source:MGI Symbol;Acc:MGI:5547776]
## 4         KiSS-1 metastasis-suppressor [Source:MGI Symbol;Acc:MGI:2663985]
## 5 zinc finger CCCH type containing 11A [Source:MGI Symbol;Acc:MGI:1917829]
## 6 zinc finger CCCH type containing 11A [Source:MGI Symbol;Acc:MGI:1917829]
##   entrezid n
## 1    19243 2
## 2    19243 2
## 3   280287 2
## 4   280287 2
## 5    70579 2
## 6    70579 2
```

In this case many of these genes also have the same gene name and description, in these cases they lie on patches or haplotypes, for more information see this video:

https://www.youtube.com/watch?v=sPE9j_Hw9HU

These duplicates could cause problems with downstream analysis and should be resolved. Often there are not many of these and they can be manually checked and resolved by looking at the gene annotations online, based on which you can make a decision about keeping the annotation or removing one or all of the duplicates. Another option for resolving duplicates is to keep the one with the highest count in the DESeq2 results. There is no ideal solution here and it is up to the researcher to decide what to do.

For now we will leave the duplicates in place, remembering that we will have to deal with them later on in any downstream analysis that requires Entrez IDs.

Annotation our DE results

```
annot <- annot %>%
    rename(GeneID = gene_id,
           Symbol = gene_name,
           Description = description,
           Entrez = entrezid)
results.d11 <- as.data.frame(results.d11) %>%
    rownames_to_column("GeneID") %>%
    left_join(annot, "GeneID")
```

NOTE: If you had the issue earlier with `select` being masked and you used the `dplyr::select` solution, you will also have an issue `rename` and will need to use `dplyr::rename`.

Finally we can output the annotation DE results using `write_tsv`.

```
write_tsv(results.d11, "results.d11_Results_Annotated.txt")
```