How to use MeSH-related Packages

Koki Tsuyuzaki[1,4], Gota Morota[2], Takeru Nakazato[3] and Itoshi Nikaido[4].

May 19, 2021

[1]Department of Medical and Life Science, Tokyo University of Science.
[2]Department of Animal Science, University of Nebraska-Lincoln
[3]Database Center for Life Science, Research Organization of Information and Systems.
[4]Bioinformatics Research Unit, RIKEN Advanced Center for Computing and Communication.

k.t.the-answer@hotmail.co.jp, dritoshi@gmail.com

Contents

1 Introduction

This document provides the way to use MeSH-related packages; *MeSH.db*, *MeSH.AOR.db*, *MeSH.PCR.db*, *MeSH.XXX.eg.db*-type packages, *MeSHDbi*, and *meshr* packages. MeSH (Medical Subject Headings) is the NLM (U. S. National Library of Medicine) controlled vocabulary used to manually index articles for MEDLINE/PubMed [1] and is a collection of a comprehensive life science vocabulary. MeSH contains more than 25,000 clinical and

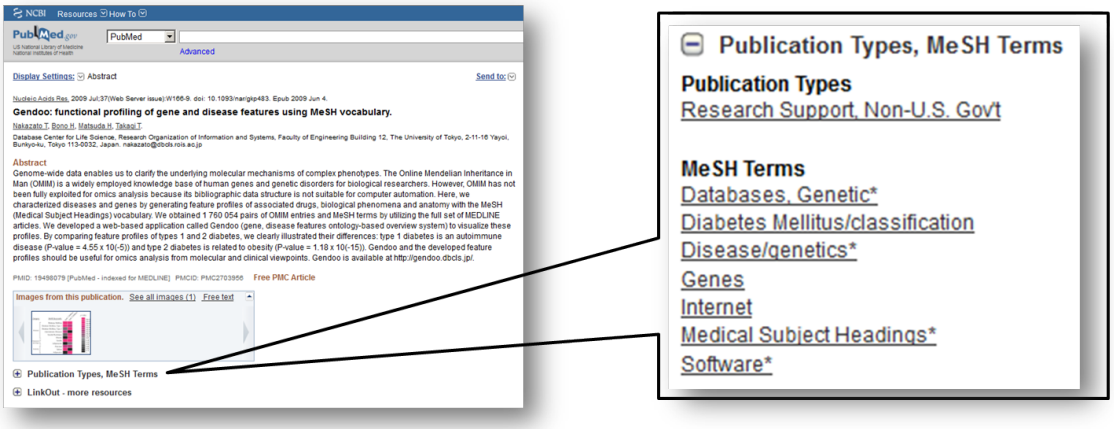

Figure 1: MeSH Term

biological terms. The amount of MeSH term is about twice as large as that of GO (Gene Ontology)[2] and its categories are also wider. MeSH in 2014 proposed its 19 categories and *MeSH.db* provides 16 of them, which are actually assigned to some MeSH terms. Each category is expressed as single capital alphabet as abbreviation defined by NLM. Therefore MeSH is an expected to be much detailed and exhaustive gene annotation tool. Some software or databases using MeSH are now proposed [3, 4, 5, 6].

This vignette introduces R/Bioconductor packages for handling MeSH in R. Original MeSH data is accessible by NLM FTP site (`http://www.nlm.nih.gov/mesh/filelist.html`). The data are downloadable as plain-text format (ASCII MeSH; d2015.bin / q2015.bin). These files were pre-processed by our data-processing pipeline (figure 2) and corresponding information is summarized as a table in SQLite3 file and packed into *MeSH.db*, *MeSH.AOR.db*, and *MeSH.PCR.db*.

Abbreviation	Category
A	Anatomy
B	Organisms
C	Diseases
D	Chemicals and Drugs
E	Analytical, Diagnostic and Therapeutic Techniques and Equipment
F	Psychiatry and Psychology
G	Phenomena and Processes
H	Disciplines and Occupations
I	Anthropology, Education, Sociology and Social Phenomena
J	Technology and Food and Beverages
K	Humanities
L	Information Science
M	Persons
N	Health Care
V	Publication Type
Z	Geographical Locations

1.1 About MeSH

MeSH.db provides the corresponding table which contains MeSH ID, MeSH term, MeSH category, synonym, qualifier ID, and qualifier term. Qualifier term means more rough annotation (subheadings) than MeSH. MeSH has hierarchical structure like GO. Such structure is provided as *MeSH.AOR.db* (AOR: ancestor-offspring Relationships) and *MeSH.PCR.db* (PCR: parent-child Relationships as corresponding table.

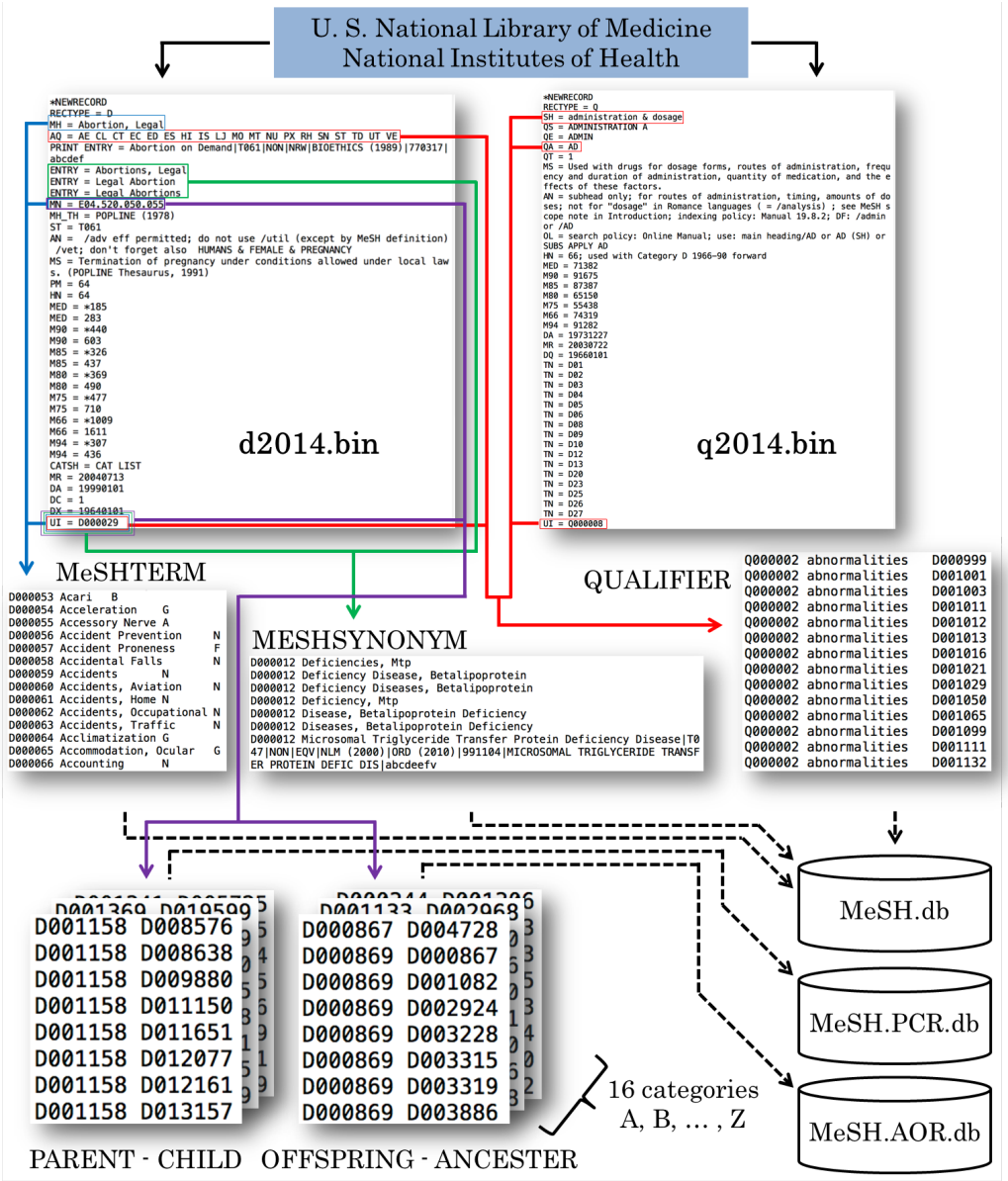

Figure 2: Data pre-process for MeSH.db

1.2 The correspondence between MeSH ID and NCBI Entrez Gene ID

MeSH.XXX.eg.db (XXX is an abbreviation of species name such as Hsa: Homo sapiens) packages provide the correspondence between Entrez Gene IDs and NLM MeSH IDs. Such correspondence in wide variety of organisms are summarized as each MeSH.XXX.eg.db by three way of methods, Gendoo[4], gene2pubmed, and RBBH (reciprocal BLAST best Hit).

Gendoo is the web-application based on text-mining of PubMed. Co-occurrence relations in PubMed document are exhaustively retrieved and much relevant correspondence are filtered by some information science techniques.

gene2pubmed is the correspondence between Entrez Gene IDs and NLM PubMed IDs. These relationship is manually assigned by NCBI curator teams. We also summarized the relationship between MeSH Terms and PubMed IDs from licensed-PubMed, then merged as Gene IDs - MeSH IDs correspondence.

For some minor species including non-model organisms, which have no sufficient databases for annotation, we defined 15 well-annotated organisms and 100 minor-organisms, then conducted RBBH between all possible combinations using BLASTP search.

Method	Way of corresponding Entrez Gene IDs and MeSH IDs
Gendoo	Text-mining
gene2pubmed	Manual curation by NCBI teams
RBBH	sequence homology with BLASTP search (E-value $< 10^{-50}$)

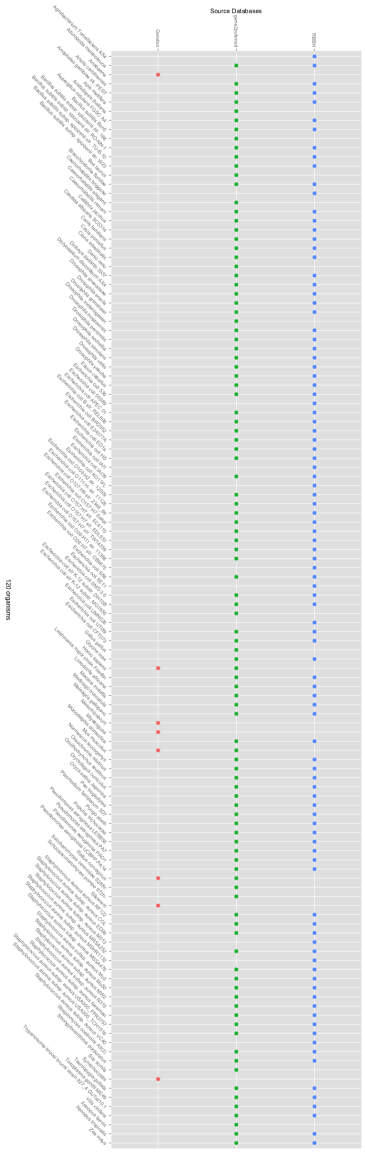

Figure 3: 120 organisms for MeSH.XXX.eg.db and those source databases

1.3 Database interface package for MeSH-related packages

We also implemented a database interface (DBI) package named *MeSHDbi*. This package is important because of two reasons. First reason is a unification of DBI functions for MeSH-related packages. *MeSH.db*, *MeSH.AOR.db*, *MeSH.PCR.db*, and *MeSH.XXX.eg.db* packages inherit the MeSHDb-class defined by *MeSHDbi* and behavior of these packages is uniformly designed. Second reason is supporting construction of user's original MeSH.XXX.eg.db package. Due to the rapid development of DNA sequence technology, wide variety of genome sequences are more and more determined and the correspondence of Gene IDs and MeSH IDs may be designed by many databases [3, 4, 5, 6]. Therefore, we prepared the function to create MeSH.XXX.eg.db package for a situation in which users can retrieved the relationship between Gene IDs and MeSH IDs by some means.

1.4 MeSH term enrichment analysis

To analyze MeSH-related packages with omics data, we implement *meshr* package, which is for conducting enrichment analysis using MeSH data. This package internally imports *MeSH.db*, *MeSH.AOR.db*, *MeSH.PCR.db* and *MeSH.XXX.eg.db*, then conducts enrichment analysis to detect highlly enriched MeSH terms in gene sets of interesting species.

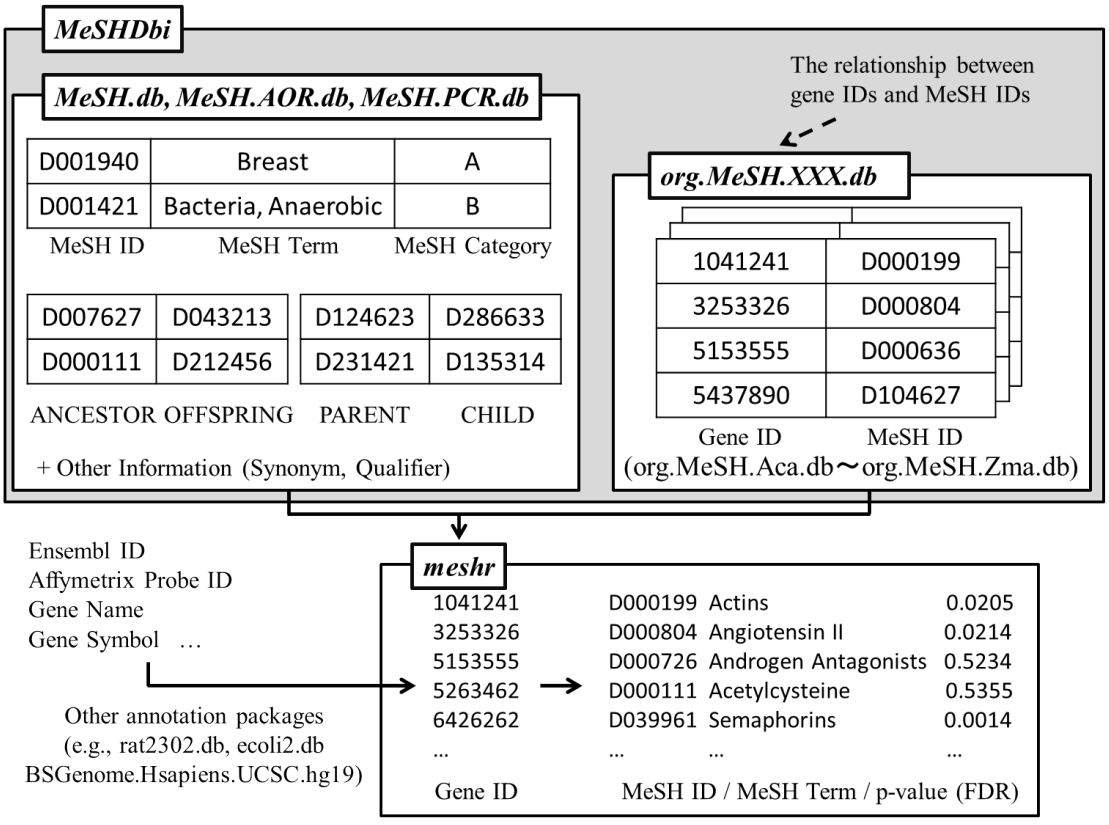

Figure 4: The relationship of meshr and other MeSH-related packages

2 Exercise

2.1 Access MeSH Term

2.1.1 columns, keytypes, keys, and select

In our packages, all data are extracted by only 4 functions defined by *AnnotationDbi*; **keytypes**, **columns**, **keys** and **select**. In this section, we demonstrate how to use these functions by using *MeSH.db*.

At first, install and load the *MeSH.db*.

```
> library(MeSH.db)
```

ls function shows all objects in this package. MeSH.db object is generated. This is also package's name and all MeSH-related packages provide the object named as package name (e.g., *MeSH.db*, *MeSH.Mmu.eg.db*).

```
> ls("package:MeSH.db")
> MeSH.db
```

Here, we use **columns**, **keytypes**, **keys** and **select** against MeSH.db.

columns returns the rows which we can retrieve in MeSH.db.

```
> columns(MeSH.db)
```

keytypes returns the rows which can be used as the optional parameter in **keys** and **select** functions against MeSH.db.

```
> keytypes(MeSH.db)
```

keys function returns the value of keytype.

```
> k <- keys(MeSH.db, keytype = "MESHID")
> length(k)
> head(k)
```

select function returns rows in particular columns, which are having user-specified keys. This function provides the data as a dataframe. Now, we will retrieve the rows in which MESHID is equivalent to MESHTERM.

```
> select(MeSH.db, keys = k[1:10], columns = c("MESHID", "MESHTERM"),
+     keytype = "MESHID")
```

2.1.2 Annotation of *Leukemia*

Next, we will retrieve some information about *Leukemia* by our packages.

select function retrieves rows in which MESHTERM is "*Leukemia*" in the MeSH.db table.

```
> LEU <- select(MeSH.db, keys = "Leukemia", columns = c("MESHID",
+     "MESHTERM", "CATEGORY", "SYNONYM"), keytype = "MESHTERM")
> LEU
```

select function shows that MESHID of *Leukemia* is D007938 and *Leukemia* is categorized as C (Diseases). *Leukemia* has some synonyms like *Leucocythaemias*, *Leucocythaemia*, *Leucocythemias* and *Leukemias*.

As mentioned above, MeSH has hierarchical structures. *MeSH.AOR.db* and *MeSH.PCR.db* packages provide such hierarchical information of MeSH. For example, *MeSH.AOR.db* enable us to examine the top terms of *Leukemia*.

```
> library("MeSH.AOR.db")
> ANC <- select(MeSH.AOR.db, keys = "D007938", columns = c("ANCESTOR",
+     "OFFSPRING"), keytype = "OFFSPRING")
> ANC
```

D009370 has found above *Leukemia*.

These MeSH IDs can be translated to MeSH Term.

```
> select(MeSH.db, keys = ANC[1, 1], columns = c("MESHTERM"), keytype = "MESHID")
```

In this way, we can specify that *Leukemia* is categorized as one of $NeoplasmsbyHistologicType$.

Once keytype-parameter set to opposite direction (OFFSPRING to ANCESTOR), other MeSH IDs in lower hierarchies also can be retrieved.

```
> OFF <- select(MeSH.AOR.db, keys = "D007938", columns = c("ANCESTOR",
+     "OFFSPRING"), keytype = "ANCESTOR")
> OFF
> select(MeSH.db, keys = OFF[, 2], columns = c("MESHTERM"), keytype = "MESHID")
```

There are a lot of MeSH terms, which means *Leukemia* has many lower hierarchies.

MeSH.PCR.db provides the directly lower (or upper) terms.

```
> library("MeSH.PCR.db")
> CHI <- select(MeSH.PCR.db, keys = LEU[1, 1], columns = c("PARENT",
+     "CHILD"), keytype = "PARENT")
> head(CHI)
> head(select(MeSH.db, keys = CHI[, 2], columns = c("MESHTERM"),
+     keytype = "MESHID"))
```

Leukemia has a lot of subtypes like $AvianLeukosis$, $BlastCrisis$, $Leukemia$, $Erythroblastic$, $Acute$ and so on.

2.1.3 Other functions

Some optional functions for much complex data acquisition are also provided. In this section, users may need some basic *SQL* knowledge (see also *RSQLite*).

dbInfo returns the information of the package. *dbfile* returns the directory where sqlite file is stored. *dbschema* returns the schema of database. *dbconn* returns the connection constructed by *RSQLite*.

```
> dbInfo(MeSH.db)
> dbfile(MeSH.db)
> dbschema(MeSH.db)
> dbconn(MeSH.db)
```

dbschema shows the data is stored as a table named "DATA" in the sqlite database and the table has six columns; MESHID, MESHTERM, CATEGORY, SYNONYM, QUALIFIERID, and QUALIFIER. Therefore, we can retrieve data by much complex SQL query like below;

```
> library("RSQLite")
> SQL1 <- paste("SELECT MESHTERM, QUALIFIERID, QUALIFIER FROM DATA",
+     "WHERE MESHID = 'D000001'", "AND QUALIFIERID = 'Q000494'")
> dbGetQuery(dbconn(MeSH.db), SQL1)
> SQL2 <- paste("SELECT ANCESTOR, OFFSPRING FROM DATA", "WHERE OFFSPRING = 'D000002'",
+     "OR OFFSPRING = 'D000003'", "OR OFFSPRING = 'D000004'", "OR ANCESTOR = 'D009275'")
> dbGetQuery(dbconn(MeSH.AOR.db), SQL2)
> SQL3 <- paste("SELECT PARENT, CHILD FROM DATA", "WHERE PARENT = 'D000005'",
+     "AND NOT CHILD = 'D004312'")
> dbGetQuery(dbconn(MeSH.PCR.db), SQL3)
```

2.2 MeSH.XXX.eg.db-type packages

2.2.1 Annotation of 120 organisms

As well as *MeSH.db*, *MeSH.AOR.db*, and *MeSH.PCR.db*, *MeSH.XXX.eg.db*-type packages also use 4 functions (**keytypes**, **columns**, **keys** and **select**) to extract data.

```
> library("MeSH.Hsa.eg.db")
> columns(MeSH.Hsa.eg.db)
> keytypes(MeSH.Hsa.eg.db)
> key_HSA <- keys(MeSH.Hsa.eg.db, keytype = "MESHID")
> select(MeSH.db, keys = key_HSA[1:10], columns = c("MESHID", "MESHTERM"),
+     keytype = "MESHID")
```

Moreover, these packages have other additional functions like **species**, **nomenclature**, **listDatabases**. In each *MeSH.XXX.eg.db*, **species** function returns the common name and **nomenclature** returns the scientific name.

```
> library("MeSH.Aca.eg.db")
> library("MeSH.Bsu.168.eg.db")
> library("MeSH.Syn.eg.db")
> species(MeSH.Hsa.eg.db)
> species(MeSH.Aca.eg.db)
> species(MeSH.Bsu.168.eg.db)
> species(MeSH.Syn.eg.db)
> nomenclature(MeSH.Hsa.eg.db)
> nomenclature(MeSH.Aca.eg.db)
> nomenclature(MeSH.Bsu.168.eg.db)
> nomenclature(MeSH.Syn.eg.db)
```

listDatabases function returns the source of data (figure 3). In regard to RBBH, name of organisms is returned. These values are important when users specify the database for MeSH Term enrichment analysis (see the section 2.4).

```
> listDatabases(MeSH.Hsa.eg.db)
> listDatabases(MeSH.Aca.eg.db)
> listDatabases(MeSH.Bsu.168.eg.db)
> listDatabases(MeSH.Syn.eg.db)
```

2.3 MeSHDbi

2.3.1 User's custom MeSH.XXX.eg.db package

Although most of users may not be conscious of this package, *MeSHDbi* is important for our MeSH packages. This package regulates class definition of MeSH object (MeSHDb-class). Besides, this package constructs user's original $MeSH.XXX.eg.db$ package. **makeGeneMeSHPackage** easily constructs such package.

```
> library("MeSHDbi")
> example("makeGeneMeSHPackage")
```

```
mGMSHP> ## makeGeneMeSHPackage enable users to construct
mGMSHP> ## user's own custom MeSH package
mGMSHP>
mGMSHP> ## this is test data which means the relationship between
mGMSHP> ## Entrez gene IDs of Pseudomonas aeruginosa PAO1
mGMSHP> ## and its MeSH IDs.
mGMSHP> data(PAO1)

mGMSHP> head(PAO1)
  GENEID  MESHID MESHCATEGORY SOURCEID
1 877657 D000265            D   937805
2 877657 D000265            D   937805
3 877657 D001412            B   937805
4 877657 D001412            B   937805
5 877657 D001426            D   937805
6 877657 D001483            G   937805
                                     SOURCEDB
1 Bacillus subtilis subsp. spizizenii str. 168
2 Bacillus subtilis subsp. spizizenii str. 168
3 Bacillus subtilis subsp. spizizenii str. 168
4 Bacillus subtilis subsp. spizizenii str. 168
5 Bacillus subtilis subsp. spizizenii str. 168
6 Bacillus subtilis subsp. spizizenii str. 168

mGMSHP> # We are also needed to prepare meta data as follows.
mGMSHP> data(metaPAO1)

mGMSHP> metaPAO1
              NAME                                  VALUE
1       SOURCEDATE                            31-July-2013
2       SOURCENAME                                 BLASTP
3        SOURCEURL ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/
4         DBSCHEMA                    org.Pae.PAO1.MeSH.db
5  DBSCHEMAVERSION                                   1.0
6         ORGANISM             Pseudomonas aeruginosa PAO1
```

```
7          SPECIES
8          package                          AnnotationDbi
9          Db type                                BLASTDb
10     MESHVERSION                                   2015

mGMSHP> ## sets up a temporary directory for this example
mGMSHP> ## (users won't need to do this step)
mGMSHP> destination <- tempfile()

mGMSHP> dir.create(destination)

mGMSHP> ## makes an Organism package for human called Homo.sapiens
mGMSHP> makeGeneMeSHPackage(pkgname = "MeSH.Pae.eg.db",
mGMSHP+                                           data = PAO1,
mGMSHP+           metadata = metaPAO1,
mGMSHP+                                           organism = "Pseudomonas aeruginosa PAO1",
mGMSHP+                                           version = "1.0.0",
mGMSHP+                                           maintainer = "Koki Tsuyuzaki <k.t.the-answe
mGMSHP+                                           author = "Koki Tsuyuzaki",
mGMSHP+                                           destDir = destination,
mGMSHP+                                           license="Artistic-2.0")
Creating package in /tmp/Rtmpqz5KOH/file1417044c5354e/MeSH.Pae.eg.db
```

2.4 meshr

2.4.1 MeSH enrichment analysis

The *meshr* package is designed to conduct an enrichment analysis for MeSH. The idea behind this package is analogous to GO enrichment analysis, where sets of genes is analyzed to extract common annotated biological properties. The usage of *meshr* closely follows that of the Bioconductor *GOstats* package. Thus, users who are familiar with *GOstats* may easily handle *meshr*.

The *meshr* package accepts selected and universal genes as input, and returns significantly overrepresented MeSH terms. It is used in conjunction with *MeSH.db* package and one of the annotation packages, (e.g., *MeSH.Hsa.eg.db*). This section serves as a quick guide to the *meshr*, while illustrating entire process to perform a MeSH enrichment analysis.

Here, we use the example data set taken from the Bioconductor package *cummeRbund*. The example data are located in library/cummeRbund/extdata/. This RNA-Seq data were taken from three samples, "iPS", "hESC", and "Fibroblasts". We first created two objects of gene sets, i.e., selected and universal genes, by comparing significantly regulated genes between iPS and hESC under the significance level of 0.05, then mapped the Gene Symbols to Entrez Gene IDs through the *org.Hs.eg.db* package. Pre-processed Gene IDs are easily accessible by **data** function.

```
> library("meshr")
> data(geneid.cummeRbund)
> data(sig.geneid.cummeRbund)
```

Finally 303 universal genes and 104 selected genes are detected and subsequently used for the MeSH enrichment analysis.

```
> dim(geneid.cummeRbund)[1]
> dim(sig.geneid.cummeRbund)[1]
```

We proceed to uncover a characteristic of MeSH terms that the set of identified genes share each other via the *meshr* package. We first load the required packages.

```
> library("fdrtool")
> library("MeSH.Hsa.eg.db")
```

We create a parameter instance by specifying the objects of selected and universal genes, the name of the annotation package, the category of MeSH, the database of correspondence between Gene IDs and MeSH IDs (see also **listDatabases** in the section 2.2.1), p-value cutoff, and the choice of a multiple-testing correction method. In this first example, we use *MeSH.Hsa.eg.db* because the above RNA-seq data is extracted from human cells. We choose C (Diseases) category, gendoo database, p-value cutoff 0.05, and no multiple-testing adjustment. For more details on description of all the arguments, readers are referred to the **MeSHHyperGParams-class** help page.

```
> meshParams <- new("MeSHHyperGParams", geneIds = sig.geneid.cummeRbund[,
+     2], universeGeneIds = geneid.cummeRbund[, 2], annotation = "MeSH.Hsa.eg.db",
+     category = "C", database = "gendoo", pvalueCutoff = 0.05,
+     pAdjust = "none")
```

The **meshHyperGTest** function carries out a hypergeometric test and returns an instance of class **MeSHHyperGResult**.

```
> meshR <- meshHyperGTest(meshParams)
```

Simply typing the **MeSHHyperGResult** class gives a brief description of the analysis, including the choice of a MeSH category, the annotation data used, and a total number of identified overrepresented MeSH terms.

```
> meshR
```

Full details of the result is obtained by calling the **summary** function on the **MeSHHyperGResult** instance. This presents significantly enriched MeSH ID, MeSH term, and their associated p-values.

```
> head(summary(meshR))
```

Switching to test another MeSH category and another database can be easily done. For example, to choose the category as G (Phenomena and Processes) and the database as gene2pubmed, we can do the following.

```
> category(meshParams) <- "G"
> database(meshParams) <- "gene2pubmed"
> meshR <- meshHyperGTest(meshParams)
> meshR
```

3 Setup

This vignette was built on:

```
> sessionInfo()

R version 4.1.0 RC (2021-05-10 r80283)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Mojave 10.14.6

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] MeSHDbi_1.28.0      BiocGenerics_0.38.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.6             rstudioapi_0.13        AnnotationDbi_1.54.0
 [4] XVector_0.32.0         zlibbioc_1.38.0        IRanges_2.26.0
 [7] bit_4.0.4              R6_2.5.0               rlang_0.4.11
[10] fastmap_1.1.0          blob_1.2.1             httr_1.4.2
[13] GenomeInfoDb_1.28.0    tools_4.1.0            Biobase_2.52.0
[16] png_0.1-7              DBI_1.1.1              bit64_4.0.5
[19] crayon_1.4.1           GenomeInfoDbData_1.2.6 vctrs_0.3.8
[22] S4Vectors_0.30.0       bitops_1.0-7           KEGGREST_1.32.0
[25] RCurl_1.98-1.3         memoise_2.0.0          cachem_1.0.5
[28] RSQLite_2.2.7          compiler_4.1.0         Biostrings_2.60.0
[31] stats4_4.1.0           pkgconfig_2.0.3
```

References

[1] S. J. Nelson and et al. The MeSH translation maintenance system: structure, interface design, and implementation. *Stud. Health Technol. Inform.*, 107: 67-69, 2004.

[2] M. Ashburner and et al. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genet.*, 25(1): 25-29, 2000.

[3] T. Nakazato and et al. BioCompass: a novel functional inference tool that utilizes MeSH hierarchy to analyze groups of genes. *In Silico Biol.*, 8(1): 53-61, 2007.

[4] T. Nakazato and et al. Nucleic Acids Res. *Gendoo: functional profiling of gene and disease features using MeSH vocabulary.*, 37: W166-W169, 2009.

[5] D. J. Saurin and et al. GeneMeSH: a web-based microarray analysis tool for relating differentially expressed genes to MeSH terms. *BMC Bioinformatics*, 11: 166, 2010.

[6] M. A. Sartor and et al. Metab2MeSH: annotating compounds with medical subject headings. *Bioinformatics*, 28(10): 1408-1410, 2012.