How to use MeSH-related Packages

Koki Tsuyuzaki\textsuperscript{1,4}, Gota Morota\textsuperscript{2}, Takeru Nakazato\textsuperscript{3} and Itoshi Nikaido\textsuperscript{4}.

April 24, 2017

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

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

Contents

1 Introduction 2
  1.1 About MeSH 4
  1.2 The correspondence between MeSH ID and NCBI Entrez Gene ID 5
  1.3 Database interface package for MeSH-related packages 7
  1.4 MeSH term enrichment analysis 7

2 Exercise 8
  2.1 Access MeSH Term 8
    2.1.1 columns, keytypes, keys, and select 8
    2.1.2 Annotation of Leukemia 10
    2.1.3 Other functions 11
  2.2 MeSH.XXX.db-type packages 12
    2.2.1 Annotation of 120 organisms 12
  2.3 MeSHDBs 13
    2.3.1 User’s custom MeSH.XXX.db package 13
  2.4 meshr 15
    2.4.1 MeSH enrichment analysis 15

3 Setup 17
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 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.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Anatomy</td>
</tr>
<tr>
<td>B</td>
<td>Organisms</td>
</tr>
<tr>
<td>C</td>
<td>Diseases</td>
</tr>
<tr>
<td>D</td>
<td>Chemicals and Drugs</td>
</tr>
<tr>
<td>E</td>
<td>Analytical, Diagnostic and Therapeutic Techniques and Equipment</td>
</tr>
<tr>
<td>F</td>
<td>Psychiatry and Psychology</td>
</tr>
<tr>
<td>G</td>
<td>Phenomena and Processes</td>
</tr>
<tr>
<td>H</td>
<td>Disciplines and Occupations</td>
</tr>
<tr>
<td>I</td>
<td>Anthropology, Education, Sociology and Social Phenomena</td>
</tr>
<tr>
<td>J</td>
<td>Technology and Food and Beverages</td>
</tr>
<tr>
<td>K</td>
<td>Humanities</td>
</tr>
<tr>
<td>L</td>
<td>Information Science</td>
</tr>
<tr>
<td>M</td>
<td>Persons</td>
</tr>
<tr>
<td>N</td>
<td>Health Care</td>
</tr>
<tr>
<td>V</td>
<td>Publication Type</td>
</tr>
<tr>
<td>Z</td>
<td>Geographical Locations</td>
</tr>
</tbody>
</table>
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.)
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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Way of corresponding Entrez Gene IDs and MeSH IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gendoo</td>
<td>Text-mining</td>
</tr>
<tr>
<td>gene2pubmed</td>
<td>Manual curation by NCBI teams</td>
</tr>
<tr>
<td>RBBH</td>
<td>sequence homology with BLASTP search (E-value &lt; $10^{-50}$)</td>
</tr>
</tbody>
</table>
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 highly 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.

```r
> 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).

```r
> 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.

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

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

```r
> keytypes(MeSH.db)
```
keys function returns the value of keytype.

```r
> 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.

```r
> 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.

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

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

The `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*.

```r
> 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.

```r
> 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 Neoplasms by Histologic Type.

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

```r
> 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.

```r
> 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 *Avian Leukosis*, *Blast Crisis*, *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.

```r
> 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.

```r
> 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).

```r
> 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.

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

### makeGeneMeSHPackage enable users to construct user's own custom MeSH package

### this is test data which means the relationship between Entrez gene IDs of Pseudomonas aeruginosa PA01 and its MeSH IDs.

```r
> data(PAO1)
> head(PAO1)
```

<table>
<thead>
<tr>
<th>GENEID</th>
<th>MESHID</th>
<th>MESHCATEGORY</th>
<th>SOURCEID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>877657</td>
<td>D000265</td>
<td>D</td>
</tr>
<tr>
<td>2</td>
<td>877657</td>
<td>D000265</td>
<td>D</td>
</tr>
<tr>
<td>3</td>
<td>877657</td>
<td>D001412</td>
<td>B</td>
</tr>
<tr>
<td>4</td>
<td>877657</td>
<td>D001412</td>
<td>B</td>
</tr>
<tr>
<td>5</td>
<td>877657</td>
<td>D001426</td>
<td>D</td>
</tr>
<tr>
<td>6</td>
<td>877657</td>
<td>D001483</td>
<td>G</td>
</tr>
</tbody>
</table>

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

```r
> metaPAO1
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCEDATE</td>
<td>31-July-2013</td>
</tr>
<tr>
<td>SOURCENAME</td>
<td>BLASTP</td>
</tr>
<tr>
<td>DBSCHEMA</td>
<td>org.Pae.PAO1.MeSH.db</td>
</tr>
<tr>
<td>DBSCHEMAVERSION</td>
<td>1.0</td>
</tr>
<tr>
<td>ORGANISM</td>
<td>Pseudomonas aeruginosa PA01</td>
</tr>
</tbody>
</table>
```r
destination <- tempfile()
dir.create(destination)
makeGeneMeSHPackage(pkgrname = "MeSH.Pae.eg.db",
data = PAO1,
metadata = metaPAO1,
organism = "Pseudomonas aeruginosa PAO1",
version = "1.0.0",
maintainer = "Koki Tsuyuzaki <k.t.the-answer@hotmail.co.jp>",
author = "Koki Tsuyuzaki",
destDir = destination,
license="Artistic-2.0")
```

Creating package in /tmp/RtmpYJ6CC0/file37a3605a317b/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.

```r
> 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.

```r
> 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.

```r
> 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:

```r
> sessionInfo()
```

R version 3.4.0 (2017-04-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.2 LTS

Matrix products: default
BLAS: /home/biocbuild/bbs-3.5-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.5-bioc/R/lib/libRlapack.so

locale:

```
[1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C
[3] LC_TIME=en_US.UTF-8 LC_COLLATE=C
[5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8 LC_NAME=C
[9] LC_ADDRESS=C LC_TELEPHONE=C
```

attached base packages:

```
[1] parallel  stats    graphics  grDevices utils    datasets  methods
[8] base
```

other attached packages:

```
[1] MeSHDbi_1.12.0    BiocGenerics_0.22.0
```

loaded via a namespace (and not attached):

```
[1] compiler_3.4.0    IRanges_2.10.0    DBI_0.6-1
[4] tools_3.4.0       memoise_1.1.0     Rcpp_0.12.10
[7] Biobase_2.36.0    AnnotationDbi_1.38.0 RSQLite_1.1-2
[10] S4Vectors_0.14.0  digest_0.6.12    stats4_3.4.0
```
References


