

PANTHER.db: An annotation package to access the PANTHER Classification System

Julius Müller

September 25, 2017

1 Introduction to PANTHER.db

The *PANTHER.db* package provides a select interface to the compiled PANTHER ontology residing within a SQLite database.

```
library(PANTHER.db,quietly=T)

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##   clusterApply, clusterApplyLB, clusterCall, clusterEvalQ, clusterExport,
##   clusterMap, parApply, parCapply, parLapply, parLapplyLB, parRapply,
##   parSapply, parSapplyLB

## The following objects are masked from 'package:stats':
##
##   IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##   Filter, Find, Map, Position, Reduce, anyDuplicated, append, as.data.frame,
##   cbind, colMeans, colSums, colnames, do.call, duplicated, eval, evalq, get,
##   grep, grepl, intersect, is.unsorted, lapply, lengths, mapply, match, mget,
##   order, paste, pmax, pmax.int, pmin, pmin.int, rank, rbind, rowMeans,
##   rowSums, rownames, sapply, setdiff, sort, table, tapply, union, unique,
##   unsplit, which, which.max, which.min

## Welcome to Bioconductor
##
## Vignettes contain introductory material; view with 'browseVignettes()'. To
## cite Bioconductor, see 'citation("Biobase)", and for packages
## 'citation("pkgname)".

##
## Attaching package: 'S4Vectors'
```

```
## The following object is masked from 'package:base':
##
##   expand.grid
## PANTHER.db version 1.0.4
```

If you already know about the select interface, you can immediately learn about the various methods for this object by just looking at the help page.

```
help("PANTHER.db")
```

When you load the *PANTHER.db* package, it creates a *PANTHER.db* object. If you look at the object you will see some helpful information about it.

```
PANTHER.db
## PANTHER.db object:
## | ORGANISMS: ANOCA|ANOPHELES|AQUAE|ARABIDOPSIS|ASHGO|BACCR|BACSU|BACTN|BATDJ|BOVINE|BRADI|BRAD
## | PANTHERVERSION: 12.0
## | PANTHERSOURCEURL: ftp.pantherdb.org
## | PANTHERSOURCEDATE: 2017-Sep15
## | package: AnnotationDbi
## | Db type: PANTHER.db
## | DBSCHEMA: PANTHER_DB
## | DBSCHEMAVERSION: 2.1
## | UNIPROT to ENTREZ mapping: 2017-Sep15
```

By default, you can see that the *PANTHER.db* object is set to retrieve records from the various organisms supported by <http://pantherdb.org>. Methods are provided to restrict all queries to a specific organism. In order to change it, you first need to look up the appropriate organism identifier for the organism that you are interested in. The PANTHER gene ontology is based on the Uniprot reference proteome set. In order to display the choices, we have provided the helper function `availablePthOrganisms` which will list all the supported organisms along with their Uniprot organism name and taxonomy ids:

```
availablePthOrganisms(PANTHER.db)[1:5,]
##   AnnotationDbi Species PANTHER Species Genome Source Genome Date UNIPROT Species ID
## 1              HUMAN      HUMAN      HGNC      2016-7      HUMAN
## 2              MOUSE      MOUSE      MGI      2016-7      MOUSE
## 3              RAT        RAT        Ensembl  2016-7      RAT
## 4              CHICKEN    CHICK     Ensembl  2016-7      CHICK
## 5              ZEBRAFISH  DANRE     ZFIN     2016-7      DANRE
##   UNIPROT Species Name UNIPROT Taxon ID
## 1              Homo sapiens          9606
## 2              Mus musculus          10090
## 3              Rattus norvegicus      10116
## 4              Gallus gallus          9031
## 5              Danio rerio           7955
```

Once you have learned the PANTHER organism name for the organism of interest, you can then change the organism for the *PANTHER.db* object:

```

pthOrganisms(PANTHER.db) <- "HUMAN"
PANTHER.db

## PANTHER.db object:
## | ORGANISMS: HUMAN
## | PANTHERVERSION: 12.0
## | PANTHERSOURCEURL: ftp.pantherdb.org
## | PANTHERSOURCEDATE: 2017-Sep15
## | package: AnnotationDbi
## | Db type: PANTHER.db
## | DBSCHEMA: PANTHER_DB
## | DBSCHEMAVERSION: 2.1
## | UNIPROT to ENTREZ mapping: 2017-Sep15

resetPthOrganisms(PANTHER.db)
PANTHER.db

## PANTHER.db object:
## | ORGANISMS: ANOCA|ANOPHELES|AQUAE|ARABIDOPSIS|ASHGO|BACCR|BACSU|BACTN|BATDJ|BOVINE|BRADI|BRAD
## | PANTHERVERSION: 12.0
## | PANTHERSOURCEURL: ftp.pantherdb.org
## | PANTHERSOURCEDATE: 2017-Sep15
## | package: AnnotationDbi
## | Db type: PANTHER.db
## | DBSCHEMA: PANTHER_DB
## | DBSCHEMAVERSION: 2.1
## | UNIPROT to ENTREZ mapping: 2017-Sep15

```

As you can see, organisms are now restricted to Homo sapiens. To display all data which can be returned from a select query, the columns method can be used:

```

columns(PANTHER.db)

## [1] "CLASS_ID"          "CLASS_TERM"        "COMPONENT_ID"     "COMPONENT_TERM"
## [5] "CONFIDENCE_CODE"  "ENTREZ"            "EVIDENCE"         "EVIDENCE_TYPE"
## [9] "FAMILY_ID"        "FAMILY_TERM"      "GOSLIM_ID"       "GOSLIM_TERM"
## [13] "PATHWAY_ID"       "PATHWAY_TERM"     "SPECIES"          "SUBFAMILY_TERM"
## [17] "UNIPROT"

```

Some of these fields can also be used as keytypes:

```

keytypes(PANTHER.db)

## [1] "CLASS_ID"          "COMPONENT_ID"     "ENTREZ"           "FAMILY_ID"       "GOSLIM_ID"
## [6] "PATHWAY_ID"       "SPECIES"          "UNIPROT"

```

It is also possible to display all possible keys of a table for any keytype. If keytype is unspecified, the FAMILY_ID will be returned.

```

go_ids <- head(keys(PANTHER.db, keytype="GOSLIM_ID"))
go_ids

## [1] "GO:0000003" "GO:0000165" "GO:0000166" "GO:0000228" "GO:0000375" "GO:0000398"

```

Finally, you can loop up whatever combinations of columns, keytypes and keys that you need when using `select`.

```
cols <- c("FAMILY_ID", "CLASS_ID")
res <- select(PANTHER.db, keys=go_ids, columns=cols, keytype="GOSLIM_ID")
head(res)

##      GOSLIM_ID      FAMILY_ID CLASS_ID
## 1 GO:0000003 PTHR10117:SF46  PC00133
## 2 GO:0000003 PTHR10117:SF46  PC00227
## 3 GO:0000003 PTHR10117:SF50  PC00133
## 4 GO:0000003 PTHR10117:SF50  PC00227
## 5 GO:0000003 PTHR10117:SF6   PC00133
## 6 GO:0000003 PTHR10117:SF6   PC00227
```

To access the PANTHER Protein Class ontology tree structure, the method `traverseClassTree` can be used:

```
term <- "PC00209"
select(PANTHER.db, term, "CLASS_TERM", "CLASS_ID")

##  CLASS_ID      CLASS_TERM
## 1  PC00209 sodium channel

ancestors <- traverseClassTree(PANTHER.db, term, scope="ANCESTOR")
select(PANTHER.db, ancestors, "CLASS_TERM", "CLASS_ID")

##      CLASS_ID  CLASS_TERM
## 1      PC00133 ion channel
## 703    PC00227 transporter

parents <- traverseClassTree(PANTHER.db, term, scope="PARENT")
select(PANTHER.db, parents, "CLASS_TERM", "CLASS_ID")

##  CLASS_ID  CLASS_TERM
## 1  PC00133 ion channel

children <- traverseClassTree(PANTHER.db, term, scope="CHILD")
select(PANTHER.db, children, "CLASS_TERM", "CLASS_ID")

##  CLASS_ID      CLASS_TERM
## 1  PC00243 voltage-gated sodium channel

offspring <- traverseClassTree(PANTHER.db, term, scope="OFFSPRING")
select(PANTHER.db, offspring, "CLASS_TERM", "CLASS_ID")

##  CLASS_ID      CLASS_TERM
## 1  PC00243 voltage-gated sodium channel
```