Annotation mapping functions

Stefan McKinnon Edwards
<stefan.hoj-edwards@@agrsci.dk>
Faculty of Agricultural Sciences, Aarhus University

October 17, 2016

Contents

1 Introduction

The Bioconductor[3] contains over 500 packages for annotational data (as of release 2.7), containing all gene data, chip data for microarrays or homology data. These packages can be used to map between different types of annotations, such as RefSeq[9], Entrez or Ensembl[4], but also pathways such as KEGG[7][5][6] og Gene Ontology[2]. This package contains functions for mapping between different annotations using the Bioconductors annotations packages and can in most cases be done with a single line of code.

1.1 Types of Annotation Packages

Bioconductors website on the annotation packages[1] lists five different types of annotation packages. Here, we list some of them:

- org.“XX”.“yy”.db - organism annotation packages containing all the gene data for en entire organism, where “XX” is the abbreviation for Genus and species. “yy” is the *central ID* that binds the data together, such as Entrez gene (eg).
- KEGG.db and GO.db containing systems biology data / pathways.
- hom.“XX”.inp.db - homology packages mapping gene between an organism and 35 other.
- Chip annotation packages for accessing data for e.g. Affymetrix chips.

[1] http://www.bioconductor.org/help/workflows/annotation-data/

1.2 Why this package?

Querying the annotation packages for data is easy, as demonstrated below:

```
> library(org.Bt.eg.db)
> genes <- c('280705', '280706', '100327208')
> symbols <- org.Bt.egSYMBOL[genes]
> toTable(symbols)

    gene_id symbol
1 100327208   BCSE
2    280705     TF
3    280706     TG
```

However, if we were to query for a non-existing gene,

```
>  org.Bt.egSYMBOL[c("280705",  "280706",  "100327208",  "123")]

Error  in  .checkKeys(value,  Lkeys(x),  x@ifnotfound)  :
  value  for  "123"  not  found
```

or try to map from a symbol to e.g. RefSeq, things become a bit more difficult:

```
> symbols <- c('BCSE','TF','TG')
> entrez <- org.Bt.egSYMBOL2EG[symbols]
> entrez <- toTable(entrez)[,'gene_id']
> entrez

[1] "280705"    "280706"    "100327208" "100327231"

> refseq <- org.Bt.egREFSEQ[entrez]
> toTable(refseq)

  gene_id accession
1  280705 NM_177484
2  280705 NP_803450
3  280706 NM_173883
4  280706 NP_776308
```

Note here, that the central ID in `org.Bt.eg.db` is *Entrez Gene ID* as indicated by the "eg" in the package name. This means that all annotational data is linked together by an Entrez Gene ID. Therefore to map from e.g. symbols to RefSeq, it is neccesary to map to Entrez first and then to RefSeq. The consequence of this is that elements can be lost in the mapping.

With this package, it can be done with one-liners:

```
> library(AnnotationFuncs)
> translate(c(280705, 280706, 100327208, 123), org.Bt.egSYMBOL)

$`280705`
[1] "TF"

$`280706`
[1] "TG"

$`100327208`
[1] "BCSE"
```

```
> translate(c('BCSE','TF','TG'), from=org.Bt.egSYMBOL2EG, to=org.Bt.egREFSEQ)

$TF
[1] "NM_177484" "NP_803450"

$TG
[1] "NM_173883" "NP_776308"
```

Note that the elements that could not be mapped to the end product are removed (by option).

2 Usage guide

First things first, we will start by describing the most usefull of functions: `translate`. After this we will cover the other functions that are intended for Gene Ontology pathways (GO) and RefSeq.

Please note, that throughout this chapter, we use `org.Bt.eg.db`, but the annotation package is more or less interchangable with the other annotation packages from Bioconductor. However, when we refer to the annotation types (e.g. `org.Bt.egCHR` or `org.Bt.egREFSEQ`), the package cannot be directly replaced by another, so you must check which types are available for the package.

2.1 translate

To map between different types of annotation data, you must first choose an annotation package for the appropiate organism. For cows there is the `org.Bt.eg.db` that maps between different identifers or `bovine.db` that maps the probes on an Affymatrix bovine chip. For humans there is also `org.Hs.eg.db`, `hgu95av2.db` for Affymetrix Human Genome U95 Set annotation data or `hom.Hs.inp.db` that maps homologues genes from human to 35 other organims.

For `org.Bt.eg.db` there is a data object for each set of data, such as mapping from Entrez Gene IDs to chromosome, `org.Bt.egCHR`. Some of the objects comes in pairs, such as Entrez and RefSeq there is `org.Bt.egREFSEQ` and `org.Bt.egREFSEQ2EG`. The latter can also be obtained with `revmap(org.Bt.egREFSEQ)`. Note that the mapping is generally from the central ID (Entrez Gene ID in this case) to the end product.

We start of by getting some basic information about some Entrez Gene IDs:

```
> library(AnnotationFuncs)
> library(org.Bt.eg.db)
> genes <- c(280705, 280706, 100327208)
> translate(genes, org.Bt.egGENENAME)

$`280705`
[1] "transferrin"

$`280706`
[1] "thyroglobulin"

$`100327208`
[1] "Bilateral convergent strabismus with exophthalmus"

> translate(genes, org.Bt.egCHR)

$`280705`
[1] "1"
```

```
$`280706`
[1] "14"

$`100327208`
[1] "5"
```

```
> translate(genes, org.Bt.egENSEMBL)
```

```
$`280705`
[1] "ENSBTAG00000007273"

$`280706`
[1] "ENSBTAG00000007823"
```

```
> translate(genes, org.Bt.egREFSEQ, remove.missing=FALSE)
```

```
$`280705`
[1] "NM_177484" "NP_803450"

$`280706`
[1] "NM_173883" "NP_776308"

$`100327208`
[1] NA
```

This is however trivial, but not the last example how the non-mapped was included in the result. The first argument is coerced into a character vector of unique entries, so providing a list will do no good (unless you use the function `unlist`).

Now we would like to map from something else than the central ID to the central ID and to a third annotation:

```
> symbols <- c("SERPINA1","KERA","CD5")
> translate(symbols, org.Bt.egSYMBOL2EG)

$SERPINA1
[1] "280699"

$KERA
[1] "280785"

$CD5
[1] "280745"
```

```
> translate(symbols, revmap(org.Bt.egSYMBOL))
```

```
$SERPINA1
[1] "280699"

$KERA
[1] "280785"

$CD5
[1] "280745"
```

The two results should be exactly the same.

To map to a third annotation, we specify a object for the mapping to the central ID and another object for the mapping from the central ID to the end product:

```
> translate(symbols, from=org.Bt.egSYMBOL2EG, to=org.Bt.egGENENAME)

$SERPINA1
[1] "serpin family A member 1"

$KERA
[1] "keratocan"

$CD5
[1] "CD5 molecule"

> # As a curiosity, if you specify another organism, you are warned:
> library(org.Hs.eg.db)
> translate(symbols, from=org.Bt.egSYMBOL2EG, to=org.Hs.egGENENAME)

named list()

> warnings()

NULL
```

If you specify an annotation type that can map to multiple entries, you can use the argument `reduce` to specify how the result is reduced. The options are `all` (default), that does not reduce, `first` that only selects the arbitrarily first element and `last` that does the same on the last element in the set.

```
> symbols <- c("SERPINA1","KERA","CD5")
> translate(symbols, org.Bt.egSYMBOL2EG, org.Bt.egREFSEQ)

$SERPINA1
[1] "NM_173882"    "NP_776307"    "XM_005222107" "XP_005222164"

$KERA
[1] "NM_173910"    "NP_776335"    "XM_010804758" "XM_015470885" "XP_010803060"
[6] "XP_015326371"

$CD5
[1] "NM_173899" "NP_776324"

> translate(symbols, org.Bt.egSYMBOL2EG, org.Bt.egREFSEQ, reduce='first')

$SERPINA1
[1] "NM_173882"

$KERA
[1] "NM_173910"

$CD5
[1] "NM_173899"

> translate(symbols, org.Bt.egSYMBOL2EG, org.Bt.egREFSEQ, reduce='last')

$SERPINA1
[1] "XP_005222164"
```

```
$KERA
[1] "XP_015326371"

$CD5
[1] "NP_776324"
```

And finally, if you for some reason needed the result on matrix form, there is the argument `return.list`:

```
> translate(symbols, org.Bt.egSYMBOL2EG, org.Bt.egREFSEQ, return.list=FALSE)

             to     from
1     NM_173882 SERPINA1
2     NP_776307 SERPINA1
3  XM_005222107 SERPINA1
4  XP_005222164 SERPINA1
5     NM_173910     KERA
6     NP_776335     KERA
7  XM_010804758     KERA
8  XM_015470885     KERA
9  XP_010803060     KERA
10 XP_015326371     KERA
11    NM_173899      CD5
12    NP_776324      CD5
```

2.1.1 Combining with other lists and using lapply and sapply

Sometimes your input might not be a vector of values, but a list object, mapping a grouping of genes:

```
> groups <- list('a'=c('ACR','ASM','S','KERA'), 'IL'=c('IL1','IL2','IL3','IL10'), 'bwahh'=c('AC
```

For these three groups, you want the entrez gene id for each gene. For this we have two options: `lapply/sapply` and `mapLists`. When using `lapply` or `sapply`, we have added an argument, `simplify` which reduces the result from translate to a simple character vector:

```
> lapply(groups, translate, from=org.Bt.egSYMBOL2EG, simplify=TRUE)

$a
[1] "280711" "280724" "280673" "280785"

$IL
[1] "280822" "280823" "281246"

$bwahh
[1] "280711" "280699" "281246" "280745"
```

Of cause, if there are many recurring IDs, we do not want redundant look ups for the genes, so we do it a bit smarter and just start by asking for a translation of all the genes. This just leaves us with the task of mapping the groups to the result.

```
> symbols <- unlist(groups, use.names=FALSE)
> ent <- translate(symbols, org.Bt.egSYMBOL2EG)
> ent
```

```
$ACR
[1] "280711"

$ASM
[1] "280724"

$S
[1] "280673"

$KERA
[1] "280785"

$IL2
[1] "280822"

$IL3
[1] "280823"

$IL10
[1] "281246"

$SERPINA1
[1] "280699"

$CD5
[1] "280745"

> mapLists(groups, ent)

$a
[1] "280711" "280724" "280673" "280785"

$IL
[1] "280822" "280823" "281246"

$bwahh
[1] "280711" "280699" "281246" "280745"
```

Tip: When unlisting a list (`unlist`), set the argument `use.names` to `FALSE`, as this will greatly speed up the process. But, naturally, only if you do not need the names.

Tip 2: `unlist` concatenate the names with a number for each entry, with might make a mess, if the names already end with a number. There is an alternative, `unlist2` in the package `AnnotationDbi` which preserves the names.

2.2 RefSeq

As noted in the previous section, a single gene can map to several RefSeq identifiers. The format of the RefSeq identifier differs depending on the type, but they can be recognized on their prefix. For instance, RefSeq identifiers starting with 'NM_' are mature mRNA transcripts and 'XP_' are model proteins. A thorough explanation can be found by executing `?org.Bt.egREFSEQ` or on NCBIs website for RefSeq[2].

[2] http://www.ncbi.nlm.nih.gov/projects/RefSeq/key.html

To be able to sort the result from `translate` (or a character vector of RefSeq identifiers in general), the function `pickRefSeq`. It uses string comparisons to select the correct identifiers, so options are not limited to those shown here.

```
> symbols <- c("SERPINA1","KERA","CD5")
> refseq <- translate(symbols, from=org.Bt.egSYMBOL2EG, to=org.Bt.egREFSEQ)
> mRNA <- pickRefSeq(refseq, priorities=c('NM','XM'))        # Equals pickRefSeq.mRNA
> mRNA

$SERPINA1
[1] "NM_173882"

$KERA
[1] "NM_173910"

$CD5
[1] "NM_173899"

> proteins <- pickRefSeq(refseq, priorities=c('NP','XP')) # Equals pickRefSeq.Protein
> proteins

$SERPINA1
[1] "NP_776307"

$KERA
[1] "NP_776335"

$CD5
[1] "NP_776324"
```

If used in a context where each element must be mapped to exactly one, the argument `reduce` can be supplied here as in `translate`.

2.3 Gene Ontology

The annotation packages can map to pathways; either Gene Ontology (GO) or KEGG. Using org.Bt.egGO makes things a bit more complicated, as each GO identifier is accompanied by an evidence and category code. We therefore supply the function `pickGO` to pick the correct pathway based on category (biological process, molecular function and/or cellular component) and evidence (typically how they were inferred).

```
> symbols <- c("SERPINA1","KERA","CD5")
> GOs <- translate(symbols, from=org.Bt.egSYMBOL2EG, to=org.Bt.egGO)
> # Pick biological process:
> pickGO(GOs, category='BP')
> # Pick only those biological processes for Entrez Gene ID 280730
> # which have been inferred from sequence similarity or electronic annotation:
> pickGO(translate(280730, org.Bt.egGO), category='BP', evidence=c('ISS','IEA'))
```

If a GO object is used as the `from` argument in `translate` for further mapping, the arguments `evidence` and `category` can be applied to `translate` to filter the intermediate product before mapping to the end product.

The evidence codes can be found by executing `?org.Bt.egGO` or `getEvidenceCodes()`.

2.4 Finding orthologs / using the INPARANOID data packages

The Stockholm Bioinformatics Centre has compiled some data packages containing orthologs and paralogs (collectively called homologs) [1, 8, 10]. These are very useful if you need to map ids from one species to another. At the current time, there are about 38 species in the data packages. These data packages can be downloaded from BioConductor and are named `hom.Xx.inp.db`, where Xx is the central species, such as At, Ce, Dm, Dr, Hs, Mm, Rn and Sc (guess what the abbreviation stands for). The central species are of the same concept as for the organism annotation packages mentioned earlier. For the human package, `hom.Hs.inp.db`, this means that the ortholog mappings are between Humans and cattle, humans and mice and so on.

The builtin method of performing a ortholog mapping is as such:

```
> library(hom.Hs.inp.db)
> submap <- hom.Hs.inpBOSTA[c('ENSP00000356224','ENSP00000384582','ENSP00000364403')]
> toTable(submap)

           inp_id             inp_id
1 ENSP00000356224 ENSBTAP00000012328
2 ENSP00000384582 ENSBTAP00000007926
3 ENSP00000364403 ENSBTAP00000020277
```

This one was quick. But if you query for lots of ids (e.g. the entire set), it takes a long time, and even longer if you are querying on the reversed map. Furthermore, if you included values that could not be mapped, an error is returned:

```
>  hom.Hs.inpBOSTA['bwahh']

Error in .checkKeys(value, Lkeys(x), x@ifnotfound) :
  value for "bwahh" not found
```

Note: This can be resolved by using `AnnotationDbi::mget(values, hom.Hs.inpBOSTA, ifnotfound=NA)` which also returns a list object.

We therefore present the function `getOrthologs`. It is faster and more powerful. Faster, as retrieving all mapped ids from bovine to human took approx. 11 minutes with the method shown above, while `getOrthologs` did it in just 1.17 seconds. More powerful, as it can translate the input values *before* mapping *and* the mapped ids *after* mapping (but not automatically - you have to tell it which translations to use). To reproduce the above (in list form):

```
> getOrthologs(c('ENSP00000356224','ENSP00000384582','ENSP00000364403'), hom.Hs.inpBOSTA, 'BOST

$ENSP00000356224
[1] "ENSBTAP00000012328"

$ENSP00000384582
[1] "ENSBTAP00000007926"

$ENSP00000364403
[1] "ENSBTAP00000020277"
```

Unfortunately we require the INPARANOID style genus names ('BOSTA') to be provided as a character string. These are the same as the five last characters in the mapping object, if you use the original names from the package.

However, the INPARANOID data packages works primarily with Ensembl protein ids. If for instance your ids are entrez, refseq, etc. you are required to translate them into Ensembl before the mapping. This can be done in the same call to the function:

```
> symbols <- c("SERPINA1","KERA","CD5")
> getOrthologs(symbols, hom.Hs.inpBOSTA, 'BOSTA', pre.from=org.Hs.egSYMBOL2EG, pre.to=org.Hs.eg

$KERA
[1] "KERA"
```

Are you surprised about the result? The four arguments `pre.from`, `pre.to`, `post.from` and `post.to` corresponds directly to the `from` and `to` arguments in `translate`, with "pre" for the translation *before* mapping and "post" for the translation *after* mapping. In this example we used the `org.Hs.eg.db` to map the input from the symbol to Ensembl. As `org.Hs.eg.db` is centred around Entrez gene ids, it requires the two-step translation from symbol to Entrez to Ensembl, and likewise for the post mapping translation. If your input was Entrez gene ids, you would only require the `pre.from` argument.

Note! Some of the species in the INPARANOID homology data packages are represented with other ids than Ensembl. For instance, *Arabidopsis thaliana* (`ARATH`) has ids such as `At1g80070.1`, while *Trichoplax adhaerens* (`TRIAD`) as ids such as `24890` which should not be confused with Entrez gene IDs.

3 What was used to create this document

The version number of R and the packages and their versions that were used to generate this document are listed below.

```
> sessionInfo()

R version 3.3.1 (2016-06-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.1 LTS

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
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] parallel  stats4    stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
[1] hom.Hs.inp.db_3.1.2    org.Hs.eg.db_3.4.0     AnnotationFuncs_1.24.0
[4] org.Bt.eg.db_3.4.0     AnnotationDbi_1.36.0   IRanges_2.8.0
[7] S4Vectors_0.12.0       Biobase_2.34.0         BiocGenerics_0.20.0

loaded via a namespace (and not attached):
[1] DBI_0.5-1     tools_3.3.1   RSQLite_1.0.0
```

4 References

References

[1] AC Berglund, E Sjolund, G Ostlund, and Sonnhammer ELL. InParanoid 6: eukaryotic ortholog clusters with inparalogs. *Nucleic Acids Research*, 36:D263–266, 2008.

[2] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25–29, 2000.

[3] Robert C Gentleman, Vincent J. Carey, Douglas M. Bates, et al. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004.

[4] T. J. P. Hubbard, B. L. Aken, S. Ayling, B. Ballester, K. Beal, E. Bragin, S. Brent, Y. Chen, P. Clapham, L. Clarke, G. Coates, S. Fairley, S. Fitzgerald, J. Fernandez-Banet, L. Gordon, S. Graf, S. Haider, M. Hammond, R. Holland, K. Howe, A. Jenkinson, N. Johnson, A. Kahari, D. Keefe, S. Keenan, R. Kinsella, F. Kokocinski, E. Kulesha, D. Lawson, I. Longden, K. Megy, P. Meidl, B. Overduin, A. Parker, B. Pritchard, D. Rios, M. Schuster, G. Slater, D. Smedley, W. Spooner, G. Spudich, S. Trevanion, A. Vilella, J. Vogel, S. White, S. Wilder, A. Zadissa, E. Birney, F. Cunningham, V. Curwen, R. Durbin, X. M. Fernandez-Suarez, J. Herrero, A. Kasprzyk, G. Proctor, J. Smith, S. Searle, and P. Flicek. Ensembl 2009. *Nucleic Acids Research*, 37(suppl 1):D690–D697, 2009.

[5] Minoru Kanehisa and Susumu Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.

[6] Minoru Kanehisa, Susumu Goto, Miho Furumichi, Mao Tanabe, and Mika Hirakawa. KEGG for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Research*, 38(suppl 1):D355–D360, 2010.

[7] Minoru Kanehisa, Susumu Goto, Masahiro Hattori, Kiyoko F. Aoki-Kinoshita, Masumi Itoh, Shuichi Kawashima, Toshiaki Katayama, Michihiro Araki, and Mika Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Research*, 34(suppl 1):D354–D357.

[8] Kevin P O'Brien, Maido Remm, and Erik L.L Sonnhammer. Inparanoid: A Comprehensive Database of Eukaryotic Orthologs. *NAR*, 33:D476–D480, 2005.

[9] Kim D Pruitt, Tatiana Tatusova, and Donna R Maglott. Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res*, 35 (Database issue(:D61–65, 2007.

[10] Maido Remm, Christian E. V. Storm, and Erik L. L. Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J. Mol. Biol.*, 314:1041–1052, 2001.