Using Bioconductor's Annotation Libraries

Marc Carlson, Jianhua Zhang

Overview

The Bioconductor project maintains a rich body of annotation data assembled into R libraries. The purpose of this vignette is to discuss the structure, contents, and usage of these annotation data libraries. Executable code is provided as examples.

Contents

Bioconductor's annotation data libraries are constructed by assembling data collected from various public data repositories using Bioconductor's *AnnotationDbi* package and distributed as regular R libraries that can be installed and loaded in the same way an R library is installed/loaded. Each annotation library is an independent unit that can be used alone or in conjunction with other annotation libraries. Platform specific libraries are a group annotation libraries assembled specifically for given platforms (e. g. Affymetrix HG_U95Av2). `org.XX.eg.db` are libraries containing data assembled at genome level for specific organisms such as human, mouse, fly, or rat. *KEGG.db* and *GO.db* are source specific libraries containing generic data for various genomes.

Each annotation library, when installed, contains a sqlite database contained within the `extdata` along with a `man` subdirectory filled with documentation about the data. The data can be accessed using the standard methods that would work for the classic environment objects (hash table with key-value pairs) and act as if they were simple associations of annotation values to a set of keys. For each of these emulated environment objects (which we will refer to as mappings), there is a corresponding help file in the `man` directory with detail descriptions of the data file and usage. In addition to the traditional access to these data, these databases can also be accessed directly by using DBI interfaces which allow for powerful new combinations of these data.

Each platform specific library creates a series of these mapping objects named by following the convention of package name plus mapping name. The package name is in lower case letters and the mapping names are in capital letters. When a given mapping maps platform specific keys to annotation data, only the name of the annotation data is used for the name of the mapping. Otherwise, the mapping names have a pattern of key name and value name joined by a "2" in between. For example, `hgu95av2ENTREZID` maps probe

ids on an Affymetrix human genome U95Av2 chip to EntrezGene IDs while `hgu95av2GO2PROBE` maps Gene Ontology IDs to probe IDs. Names of the mappings available in a platform specific data package are not listed here to save space but are easily accessible as shown later in the section for usage.

Genome level annotation libraries are named in the form of org.Xx.yy.db where Xx represents an abbreviation of the genus and species. Each of the organism wide genome anotation packages is based upon some type of widely used gene based identifier (such as an Entrez Gene id) that is mapped onto all the other features in the package. The yy part of the name corresponds to this designation, where eg means a package is an entrez gene package and sgd is a package based on the sgd database etc. In many cases the org packages will contain more different kinds of information that the platform based ones, since not all types of information are as widely sought after.

The *KEGG.db* library contains mappings between ids such as Entrez Gene IDs and *GO* to *KEGG* pathway ids and thus also to pathway names. The *GO.db* library maintain the directed acyclic graph structure of the original data from Gene Ontology Consortium by providing mappings of GO ids to their direct parents or children for each of the three categories (molecular function, cellular component, and biological process). Mappings between Entrez Gene and *GO* ids are also available to complement the the GO.db package. These mappings are found within the organism wide packages mentioned above. These mappings are provided with evidence code that specifies the type of evidence that supports the annotation of a gene to a particular *GO* term.

Usage

All the annotation libraries can be obtained from the Bioconductor web site (`http://www.bioconductor.org`). To illustrate their usages, we use the library for Affymetrix HG_U95Av2 chip (*hgu95av2.db*) as an example for platform specific data packages and the *GO.db* library for non-platform specific data packages. We assume that R (`www.r-project.org`) and Bioconductor's Biobase and annotation libraries have already been installed.

Package installation

Download libraries *hgu95av2.db* and *GO.db* with BiocManager::install(). If the *reposTools* library has already been installed/loaded, typing `install.packages2(library name)` installs the library for both Unix and Windows.

Typing library(library name) in an R session will load the library into R. For example,

```
> library("annotate")
> library("hgu95av2.db")
> library("GO.db")
```

Documentations

Each library contains documentation for the library in general and each of the individual mapping objects contained by the library. Two documents at the library level can be accessed by typing a library basename proceeded by a question mark (e. g. ?hgu95av2) and the library basename followed by a pair of brackets (e. g. hgu95av2()), respectively. The former explains what the package is and details how a user can get more information, while the latter lists all the mappings contained by a library and provides information on the total number of keys within each of the maps contained by the library and how many of these keys are annotated. In addition, the latter will indicate the sources for the information provided by the package as well as the date that these sources claim to have last been updated.

The documentation for a given mapping object can be accessed by typing the name of an mapping object proceeded by a question mark (e. g. `hgu95av2GO`). The resulting documentation provides detail explanations to the mapping object, data source used to build the object, and example code for accessing annotation data.

Accessing annotation data within a library

Annotation data of a given library are stored as mapping objects in the form of key (items to be annotated) and value (annotation for an key item) pairs. Each mapping object provides annotation for keys for a particular subject reflected by the name of the object. For example, `hgu95av2GO` annotates probes on the HGU95Av2 chip with ids of the Gene Ontology terms the probes correspond to.

The name of an mapping object consists of package basename (*hgu95av2.db*) and mapping name (*GO*) to avoid confusion when multiple libraries are loaded to the system at the same time. Data contained by an mapping can be accessed easily using Bioconductor's existing functions. For example, the following code stores all the keys contained by the `hgu95av2GO` mapping object to variable *temp* and displays the first five keys on the screen:

```
> as.list(hgu95av2GO[5])

$`1004_at`
$`1004_at`$`GO:0006955`
$`1004_at`$`GO:0006955`$GOID
[1] "GO:0006955"

$`1004_at`$`GO:0006955`$Evidence
[1] "IBA"

$`1004_at`$`GO:0006955`$Ontology
[1] "BP"
```

```
$`1004_at`$`GO:0007186`
$`1004_at`$`GO:0007186`$GOID
[1] "GO:0007186"

$`1004_at`$`GO:0007186`$Evidence
[1] "TAS"

$`1004_at`$`GO:0007186`$Ontology
[1] "BP"


$`1004_at`$`GO:0007204`
$`1004_at`$`GO:0007204`$GOID
[1] "GO:0007204"

$`1004_at`$`GO:0007204`$Evidence
[1] "IBA"

$`1004_at`$`GO:0007204`$Ontology
[1] "BP"


$`1004_at`$`GO:0019722`
$`1004_at`$`GO:0019722`$GOID
[1] "GO:0019722"

$`1004_at`$`GO:0019722`$Evidence
[1] "IBA"

$`1004_at`$`GO:0019722`$Ontology
[1] "BP"


$`1004_at`$`GO:0030595`
$`1004_at`$`GO:0030595`$GOID
[1] "GO:0030595"

$`1004_at`$`GO:0030595`$Evidence
[1] "IEA"

$`1004_at`$`GO:0030595`$Ontology
[1] "BP"


$`1004_at`$`GO:0032467`
$`1004_at`$`GO:0032467`$GOID
```

```
[1] "GO:0032467"

$`1004_at`$`GO:0032467`$Evidence
[1] "IMP"

$`1004_at`$`GO:0032467`$Ontology
[1] "BP"


$`1004_at`$`GO:0042113`
$`1004_at`$`GO:0042113`$GOID
[1] "GO:0042113"

$`1004_at`$`GO:0042113`$Evidence
[1] "IEA"

$`1004_at`$`GO:0042113`$Ontology
[1] "BP"


$`1004_at`$`GO:0048535`
$`1004_at`$`GO:0048535`$GOID
[1] "GO:0048535"

$`1004_at`$`GO:0048535`$Evidence
[1] "IEA"

$`1004_at`$`GO:0048535`$Ontology
[1] "BP"


$`1004_at`$`GO:0060326`
$`1004_at`$`GO:0060326`$GOID
[1] "GO:0060326"

$`1004_at`$`GO:0060326`$Evidence
[1] "IBA"

$`1004_at`$`GO:0060326`$Ontology
[1] "BP"


$`1004_at`$`GO:0070098`
$`1004_at`$`GO:0070098`$GOID
[1] "GO:0070098"
```

```
$`1004_at`$`GO:0070098`$Evidence
[1] "IEA"

$`1004_at`$`GO:0070098`$Ontology
[1] "BP"


$`1004_at`$`GO:0005886`
$`1004_at`$`GO:0005886`$GOID
[1] "GO:0005886"

$`1004_at`$`GO:0005886`$Evidence
[1] "TAS"

$`1004_at`$`GO:0005886`$Ontology
[1] "CC"


$`1004_at`$`GO:0005887`
$`1004_at`$`GO:0005887`$GOID
[1] "GO:0005887"

$`1004_at`$`GO:0005887`$Evidence
[1] "TAS"

$`1004_at`$`GO:0005887`$Ontology
[1] "CC"


$`1004_at`$`GO:0009897`
$`1004_at`$`GO:0009897`$GOID
[1] "GO:0009897"

$`1004_at`$`GO:0009897`$Evidence
[1] "IBA"

$`1004_at`$`GO:0009897`$Ontology
[1] "CC"


$`1004_at`$`GO:0004930`
$`1004_at`$`GO:0004930`$GOID
[1] "GO:0004930"

$`1004_at`$`GO:0004930`$Evidence
[1] "TAS"
```

```
$`1004_at`$`GO:0004930`$Ontology
[1] "MF"


$`1004_at`$`GO:0005515`
$`1004_at`$`GO:0005515`$GOID
[1] "GO:0005515"

$`1004_at`$`GO:0005515`$Evidence
[1] "IPI"

$`1004_at`$`GO:0005515`$Ontology
[1] "MF"


$`1004_at`$`GO:0016493`
$`1004_at`$`GO:0016493`$GOID
[1] "GO:0016493"

$`1004_at`$`GO:0016493`$Evidence
[1] "IBA"

$`1004_at`$`GO:0016493`$Ontology
[1] "MF"


$`1004_at`$`GO:0016494`
$`1004_at`$`GO:0016494`$GOID
[1] "GO:0016494"

$`1004_at`$`GO:0016494`$Evidence
[1] "IEA"

$`1004_at`$`GO:0016494`$Ontology
[1] "MF"


$`1004_at`$`GO:0019957`
$`1004_at`$`GO:0019957`$GOID
[1] "GO:0019957"

$`1004_at`$`GO:0019957`$Evidence
[1] "IBA"

$`1004_at`$`GO:0019957`$Ontology
```

```
[1] "MF"
```

To obtain annotation for a given set of keys, one may use the `mget` function. Suppose we have run an experiment using the HG_U95Av2 chip and found three genes represented by Affymetrix probe ids *738_at, 40840_at*, and *41668_r_at* interesting. To get the names of genes the three probe ids corresponding to, we do:

```
> mget(c("738_at", "40840_at", "41668_r_at"), hgu95av2GENENAME)

$`738_at`
[1] "5'-nucleotidase, cytosolic II"

$`40840_at`
[1] "peptidylprolyl isomerase F"

$`41668_r_at`
[1] "TDP-glucose 4,6-dehydratase"
```

Similarly, identifiers of Gene Ontology terms corrsponding to the three probes can be obtained as shown below:

```
> temp <- mget(c("41561_s_at", "40840_at", "41668_r_at"), hgu95av2GO)
```

In this case, the function `mget` returns a list of pre-defined S4 objects containing data for the ids, ontology, and evidence code of Gene Ontology terms corresponding to the three keys. The following code shows how to access the GO id, evidence code and ontology of the Gene Ontology term corresponding to probe id *40840_at*:

```
> temp <- get("738_at", hgu95av2GO)
> names(temp)

 [1] "GO:0006195" "GO:0016311" "GO:0017144" "GO:0046040" "GO:0046085"
 [6] "GO:0005829" "GO:0000166" "GO:0005515" "GO:0008253" "GO:0046872"

> temp[["GO:0008253"]][["Evidence"]]

[1] "IBA"

> temp[["GO:0008253"]][["Ontology"]]

[1] "MF"
```

As shown above, probe *40840_at* can be annotated by three Gene Ontology terms identified by *GO:0005829*, *GO:0008253*, and *GO:0016787*. The evidence code for *GO:0008253* is *TAS* (traceable author statement) and it belongs to ontology *MF* (molecular function).

Accessing annotation data across libraries

Often, data available in a given data package alone may not be sufficient and need to be sought across packages. Bioconductor's annotation data packages are linked by common public data identifiers to allow traverse between packages (Fig. 1). Using the example above, we know that probe id *738_at* are annotated by three Gene Ontology ids *GO:0005829*, *GO:0008253*, and *GO:0016787*. The Gene Ontology terms for various Gene Ontology ids, however, are stored in another package named *GO.db*. AS package *hgu95av2.db* and *GO.db* are linked by *GO* ids, one can annotate probe id *738_at* with Gene Ontology terms by linking data in the two packages using *GO* id as shown below:

```
> mget(names(get("738_at", hgu95av2GO)), GOTERM)

$`GO:0006195`
GOID: GO:0006195
Term: purine nucleotide catabolic process
Ontology: BP
Definition: The chemical reactions and pathways resulting in
    the breakdown of a purine nucleotide, a compound
    consisting of nucleoside (a purine base linked to a
    deoxyribose or ribose sugar) esterified with a phosphate
    group at either the 3' or 5'-hydroxyl group of the sugar.
Synonym: purine nucleotide breakdown
Synonym: purine nucleotide catabolism
Synonym: purine nucleotide degradation

$`GO:0016311`
GOID: GO:0016311
Term: dephosphorylation
Ontology: BP
Definition: The process of removing one or more phosphoric
    (ester or anhydride) residues from a molecule.

$`GO:0017144`
GOID: GO:0017144
Term: drug metabolic process
Ontology: BP
Definition: The chemical reactions and pathways involving a
    drug, a substance used in the diagnosis, treatment or
    prevention of a disease; as used here antibiotic
    substances (see antibiotic metabolism) are considered to
    be drugs, even if not used in medical or veterinary
    practice.
Synonym: drug metabolism

$`GO:0046040`
```

```
GOID: GO:0046040
Term: IMP metabolic process
Ontology: BP
Definition: The chemical reactions and pathways involving IMP,
    inosine monophosphate.
Synonym: IMP metabolism

$`GO:0046085`
GOID: GO:0046085
Term: adenosine metabolic process
Ontology: BP
Definition: The chemical reactions and pathways involving
    adenosine, adenine riboside, a ribonucleoside found widely
    distributed in cells of every type as the free nucleoside
    and in combination in nucleic acids and various nucleoside
    coenzymes.
Synonym: adenosine metabolism

$`GO:0005829`
GOID: GO:0005829
Term: cytosol
Ontology: CC
Definition: The part of the cytoplasm that does not contain
    organelles but which does contain other particulate
    matter, such as protein complexes.

$`GO:0000166`
GOID: GO:0000166
Term: nucleotide binding
Ontology: MF
Definition: Interacting selectively and non-covalently with a
    nucleotide, any compound consisting of a nucleoside that
    is esterified with (ortho)phosphate or an oligophosphate
    at any hydroxyl group on the ribose or deoxyribose.

$`GO:0005515`
GOID: GO:0005515
Term: protein binding
Ontology: MF
Definition: Interacting selectively and non-covalently with
    any protein or protein complex (a complex of two or more
    proteins that may include other nonprotein molecules).
Synonym: GO:0001948
Synonym: GO:0045308
Synonym: protein amino acid binding
Synonym: glycoprotein binding
```

```
Secondary: GO:0001948
Secondary: GO:0045308

$`GO:0008253`
GOID: GO:0008253
Term: 5'-nucleotidase activity
Ontology: MF
Definition: Catalysis of the reaction: a 5'-ribonucleotide +
    H2O = a ribonucleoside + phosphate.
Synonym: 5' nucleotidase activity
Synonym: 5'-mononucleotidase activity
Synonym: 5'-ribonucleotide phosphohydrolase activity
Synonym: 5'-adenylic phosphatase
Synonym: 5'-AMP nucleotidase
Synonym: 5'-AMPase
Synonym: adenosine 5'-phosphatase
Synonym: adenosine monophosphatase
Synonym: AMP phosphatase
Synonym: AMP phosphohydrolase
Synonym: AMPase
Synonym: snake venom 5'-nucleotidase
Synonym: thimidine monophosphate nucleotidase
Synonym: UMPase
Synonym: uridine 5'-nucleotidase

$`GO:0046872`
GOID: GO:0046872
Term: metal ion binding
Ontology: MF
Definition: Interacting selectively and non-covalently with
    any metal ion.
Synonym: metal binding
Synonym: heavy metal binding
```

It turns out that probe id *738_at* (corresponding to *GO:0008253*, and *GO:0016787*) has molecular function (MF) *5'-nucleotidase activity* and *hydrolase activity*.

1 Session Information

The version number of R and packages loaded for generating the vignette were:

```
R version 4.1.0 beta (2021-05-03 r80259)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.2 LTS

Matrix products: default
```

```
BLAS:   /home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.14-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              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] grid      stats4    parallel  stats     graphics  grDevices
 [7] utils     datasets  methods   base

other attached packages:
 [1] GO.db_3.13.0         hgu95av2.db_3.2.3    org.Hs.eg.db_3.13.0
 [4] Rgraphviz_2.37.0     graph_1.71.0         xtable_1.8-4
 [7] annotate_1.71.0      XML_3.99-0.6         AnnotationDbi_1.55.0
[10] IRanges_2.27.0       S4Vectors_0.31.0     Biobase_2.53.0
[13] BiocGenerics_0.39.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.6             compiler_4.1.0
 [3] GenomeInfoDb_1.29.0    XVector_0.33.0
 [5] bitops_1.0-7           tools_4.1.0
 [7] zlibbioc_1.39.0        bit_4.0.4
 [9] RSQLite_2.2.7          memoise_2.0.0
[11] pkgconfig_2.0.3        png_0.1-7
[13] rlang_0.4.11           DBI_1.1.1
[15] rstudioapi_0.13        fastmap_1.1.0
[17] GenomeInfoDbData_1.2.6 httr_1.4.2
[19] Biostrings_2.61.0      vctrs_0.3.8
[21] bit64_4.0.5            R6_2.5.0
[23] blob_1.2.1             KEGGREST_1.33.0
[25] RCurl_1.98-1.3         cachem_1.0.5
[27] crayon_1.4.1
```