AnnotationDbi Demo

Herve Pages, Nianhua Li, and Seth Falcon

October 17, 2007

1 Introduction

The *AnnotationDbi* package provides an interface to SQLite-based annotation packages. Each SQLite-based annotation package (identified by a ".db" suffix in the package name) contains a number of *AnnDbBimap* objects in place of the *environment* objects found in the old-style environment-based annotation packages. The API provided by *AnnotationDbi* allows you to treat the *AnnDbBimap* objects like *environment* instances. For example, the functions `[[`, `get`, `mget`, and `ls` all behave the same as with the old-style packages. In addition, we are experimenting with new methods like `[`, `toTable`, `subset` and many more that provide some additional flexibility in accessing the annotation data.

```
R> library("hgu95av2.db")
```

The same basic set of objects is provided with the db packages:

```
R> ls(2)
```

```
 [1] "hgu95av2ACCNUM"       "hgu95av2ALIAS2PROBE"
 [3] "hgu95av2CHR"          "hgu95av2CHRLENGTHS"
 [5] "hgu95av2CHRLOC"       "hgu95av2_dbconn"
 [7] "hgu95av2_dbfile"      "hgu95av2_dbInfo"
 [9] "hgu95av2_dbschema"    "hgu95av2ENTREZID"
[11] "hgu95av2ENZYME"       "hgu95av2ENZYME2PROBE"
[13] "hgu95av2GENENAME"     "hgu95av2GO"
[15] "hgu95av2GO2ALLPROBES" "hgu95av2GO2PROBE"
[17] "hgu95av2MAP"          "hgu95av2MAPCOUNTS"
[19] "hgu95av2OMIM"         "hgu95av2ORGANISM"
[21] "hgu95av2PATH"         "hgu95av2PATH2PROBE"
```

```
[23] "hgu95av2PFAM"          "hgu95av2PMID"
[25] "hgu95av2PMID2PROBE"    "hgu95av2PROSITE"
[27] "hgu95av2REFSEQ"        "hgu95av2SYMBOL"
[29] "hgu95av2UNIGENE"
```

To demonstrate the *environment* API, we'll start with a random sample of probe set IDs.

```
R> all_probes <- ls(hgu95av2ENTREZID)
R> length(all_probes)

[1] 12625

R> set.seed(0xa1beef)
R> probes <- sample(all_probes, 5)
R> probes

[1] "31882_at"   "38780_at"   "37033_s_at" "1702_at"    "31610_at"
```

The usual ways of accessing annotation data are available.

```
R> hgu95av2ENTREZID[[probes[1]]]

[1] "9136"

R> hgu95av2ENTREZID$"31882_at"

[1] "9136"

R> syms <- unlist(mget(probes, hgu95av2SYMBOL))
R> syms

  31882_at   38780_at 37033_s_at    1702_at   31610_at
    "RRP9"   "AKR1A1"     "GPX1"    "IL2RA" "PDZK1IP1"
```

Many filtering operations on the annotation *environment* objects require conversion of the *environment* into a *list*. There is an `as.list` method for *AnnDbBimap* objects. In general, converting to lists will not be the most efficient way to filter the annotation data when using a SQLite-based package.

```
R> zz <- as.list(hgu95av2SYMBOL)
```

In an environment-based package, each mapping is its own object. To save disk and memory resources, not all reverse mappings are included in the environment-based packages. Here is the common idiom for creating a list that serves as the reverse map of a given environment.

```
R> library("hgu95av2", warn.conflicts=FALSE)
R> ## we load the environment so as not
R> ## to include the load time in the timing
R> class(hgu95av2SYMBOL)

[1] "environment"

R> system.time({
     p2sym <- as.list(hgu95av2SYMBOL)
     lens <- sapply(p2sym, length)
     nms <- rep(names(p2sym), lens)
     sym2p <- split(unlist(p2sym), nms)
     })

   user  system elapsed
  0.152   0.000   0.153

R> ## in fact, there is a convenience function
R> ## for this operation in Biobase
R> system.time({
     p2sym <- as.list(hgu95av2SYMBOL)
     sym2p <- reverseSplit(p2sym)
     })

   user  system elapsed
  0.140   0.004   0.141

R> detach("package:hgu95av2")
```

The SQLite-based package provide the same reverse maps as objects in the package name space for backwards compatibility, but the reverse mappings of any map is available using `revmap`. Since the data are stored as tables, no extra disk space is needed to provide reverse mappings.

```
R> system.time(sym2p <- revmap(hgu95av2SYMBOL))
```

```
   user  system elapsed
  0.004   0.000   0.002
```

```
R> unlist(mget(syms, revmap(hgu95av2SYMBOL)))
```

```
        RRP9       AKR1A1         GPX1        IL2RA     PDZK1IP1
  "31882_at"   "38780_at" "37033_s_at"    "1702_at"   "31610_at"
```

Another area where the SQLite-based packages provide some advantages is when one wishes to filter the available annotation data in a complex fashion. For example, consider the task of obtaining all gene symbols on which are probed on a chip that have at least one GO BP ID annotation with evidence code IMP, IGI, IPI, or IDA. Here is one way to extract this using the environment-based packages:

```
R> ## Obtain SYMBOLS with at least one GO BP
R> ## annotation with evidence IMP, IGI, IPI, or IDA.
R> probes <- sample(all_probes, 500)
R> library("hgu95av2", warn.conflicts=FALSE)
R> system.time({
 bpids <- eapply(hgu95av2GO, function(x) {
     if (length(x) == 1 && is.na(x))
       NA
     else {
         sapply(x, function(z) {
             if (z$Ontology == "BP")
               z$GOID
             else
               NA
             })
     }
 })
 bpids <- unlist(bpids)
 bpids <- unique(bpids[!is.na(bpids)])
 g2p <- mget(bpids, hgu95av2GO2PROBE)
 wantedp <- lapply(g2p, function(x) {
     x[names(x) %in% c("IMP", "IGI", "IPI", "IDA")]
 })
 wantedp <- wantedp[sapply(wantedp, length) > 0]
 wantedp <- unique(unlist(wantedp))
```

```
 ans <- unlist(mget(wantedp, hgu95av2SYMBOL))
 })

   user  system elapsed
  8.020   0.084   8.106

R> detach("package:hgu95av2")
R> length(ans)

[1] 1499

R> ans[1:10]

 32042_at  32618_at  32805_at  33529_at  33756_at  33899_at  34084_at
  "ENOX2"   "BLVRA"  "AKR1C1"    "ADH7"    "AOC3" "ALDH9A1"  "AKR1D1"
 34826_at  35216_at  36963_at
   "SDHA"     "ME3"     "PGD"
```

All of the above code reduces to a single SQL query with the SQLite-based packages:

```
R> system.time({
 SQL <- "SELECT symbol FROM go_bp INNER JOIN gene_info USING(id)
         WHERE go_bp.evidence in ('IPI', 'IDA', 'IMP', 'IGI')"
 zz <- dbGetQuery(hgu95av2_dbconn(), SQL)
 })

   user  system elapsed
  0.080   0.000   0.081
```

A *Bimap* interface is available to access the data in table (*data.frame*) format using `[` and `toTable`.

```
R> toTable(hgu95av2GO[probes[1:3]])

   probe_id      go_id Evidence Ontology
1  33835_at GO:0006334      IEA       BP
2  38292_at GO:0007216      TAS       BP
3  39297_at GO:0009653      TAS       BP
4  33835_at GO:0005634      IEA       CC
5  38292_at GO:0005737      IEA       CC
6  38292_at GO:0016020      IEA       CC
```

```
7  38292_at GO:0030054      IEA       CC
8  38292_at GO:0045202      IEA       CC
9  38292_at GO:0003779      IEA       MF
10 38292_at GO:0005515      IEA       MF
11 38292_at GO:0030160      IEA       MF
```

```
R> as.list(revmap(hgu95av2PATH)["00300"])
```

```
$`00300`
[1] "34336_at" "35870_at" "35761_at"
```

In the case of the PATH map, we don't need to use revmap(x) because hgu95av2.db already provides the PATH2PROBE map:

```
R> x <- hgu95av2PATH
R> ## except for the name, this is exactly revmap(x)
R> revx <- hgu95av2PATH2PROBE
R> revx2 <- revmap(x, objName="PATH2PROBE")
R> revx2
```

```
PATH2PROBE map for chip hgu95av2 (object of class "AnnDbBimap")
```

```
R> identical(revx, revx2)
```

```
[1] TRUE
```

```
R> as.list(revx["00300"])
```

```
$`00300`
[1] "34336_at" "35870_at" "35761_at"
```

Note that the order of the cols returned by `toTable` does not depend on the direction of the map ("undirected method"):

```
R> toTable(x)[1:6, ]
```

```
  probe_id kegg_id
1 36512_at   00623
2 36512_at   00650
3 36512_at   00960
4 36332_at   00380
5 36185_at   00252
6 36185_at   00970
```

```
R> toTable(revx)[1:6, ]

  probe_id kegg_id
1 36512_at   00623
2 36512_at   00650
3 36512_at   00960
4 36332_at   00380
5 36185_at   00252
6 36185_at   00970
```

NB: the Lkeys are always on the left (1st col), the Rkeys always in the 2nd col

There can be more than 2 columns in the returned data frame:

3 cols:

```
R> toTable(hgu95av2PFAM)[1:6, ]  # the right values are tagged

  probe_id      ipi_id  PfamId
1  1000_at IPI00018195 PF00069
2  1000_at IPI00304111 PF00069
3  1000_at IPI00742900 PF00069
4  1000_at IPI00793141 PF00069
5  1001_at IPI00019530 PF07714
6  1001_at IPI00019530 PF00041

R> as.list(hgu95av2PFAM["1000_at"])

$`1000_at`
IPI00018195 IPI00304111 IPI00742900 IPI00793141
  "PF00069"   "PF00069"   "PF00069"   "PF00069"
```

But the Rkeys are ALWAYS in the 2nd col.

For length() and keys(), the result does depend on the direction ("directed methods"):

```
R> length(x)

[1] 12625

R> length(revx)

[1] 197
```

```
R> allProbeSetIds <- keys(x)
R> allKEGGIds <- keys(revx)
```

There are "undirected" methods related to these methods:

```
R> junk <- Lkeys(x)        # same for all maps in hgu95av2.db (except pseudo-map
R>                         # MAPCOUNTS)
R> Llength(x)              # nb of Lkeys

[1] 12625

R> junk <- Rkeys(x)        # KEGG ids for PATH/PATH2PROBE maps, GO ids for
R>                         # GO/GO2PROBE/GO2ALLPROBES maps, etc...
R> Rlength(x)              # nb of Rkeys

[1] 197
```

NB: they give the same result for x and revmap(x)
Using revmap can be very efficient in some use cases:

```
R> x <- hgu95av2CHR
R> Rkeys(x)

 [1] "8"  "14" "3"  "2"  "17" "16" "9"  "X"  "6"  "1"  "7"  "12" "10"
[14] "11" "22" "19" "15" "20" "21" "5"  "18" "4"  "13" "Y"

R> chroms <- Rkeys(x)[23:24]
R> chroms

[1] "13" "Y"

R> Rkeys(x) <- chroms
R> toTable(x)[1:10, ]

     probe_id chromosome
1    37303_at         13
2    37099_at         13
3  32991_f_at          Y
4    40435_at          Y
5  40436_g_at          Y
6  35447_s_at          Y
7    32482_at         13
8    32439_at         13
9    37930_at         13
10    1503_at         13
```

To get this in the classic named-list format:

```
R> z <- as.list(revmap(x)[chroms])
R> names(z)

[1] "13" "Y"

R> z[["Y"]][1:5]

[1] "32991_f_at" "40435_at"   "40436_g_at" "35447_s_at" "33665_s_at"
```

Compare to what you need to do this with the current envir-based package:

```
R>   library(hgu95av2)
R>   u <- unlist(as.list(hgu95av2CHR))
R>   u <- u[u %in% chroms]
R>   split(names(u), u)
```

A last example with cytogenetic locations:

```
R> x <- hgu95av2MAP
R> toTable(hgu95av2MAP)[1:6, ]

  probe_id cytogenetic_location
1 38187_at         8p23.1-p21.3
2 38912_at                 8p22
3 33825_at              14q32.1
4 36512_at         3q21.3-q25.2
5 38434_at                 2q35
6 36332_at                17q25

R> as.list(revmap(x)["8p22"])

$`8p22`
 [1] "38912_at"   "32372_at"   "41209_at"   "39981_at"   "39982_r_at"
 [6] "36850_at"   "36851_g_at" "36852_at"   "34553_at"   "37363_at"
[11] "37951_at"   "38013_at"
```

Are the probes in 'pbids' mapped to cytogenetic location "6p21.3"?

```
R> pbids <- c("38912_at", "41654_at", "907_at", "2053_at", "2054_g_at",
           "40781_at")
R> x <- subset(x, Lkeys=pbids, Rkeys="18q11.2")
R> toTable(x)

   probe_id cytogenetic_location
1   2053_at              18q11.2
2 2054_g_at              18q11.2
```

To coerce this map to a named vector:

```
R>   pb2cyto <- as.character(x)
R>   pb2cyto[pbids]

     <NA>      <NA>      <NA>   2053_at 2054_g_at      <NA>
       NA        NA        NA "18q11.2" "18q11.2"        NA
```

The coercion of the reverse map works too but issues a warning because of the duplicated names:

```
R>   cyto2pb <- as.character(revmap(x))
```