

Creating a New Annotation Package using SQLForge

Marc Carlson, Herve Pages, Nianhua Li

January 15, 2016

1 Introduction

The *AnnotationForge* package provides a series of functions that can be used to build *AnnotationDbi* packages for supported organisms. This collection of functions is called SQLForge.

In order to use SQLForge you really only need to have one kind of information and that is a list of paired IDs. These IDs are to be stored in a tab delimited file that is formatted in the same way that they used to be for the older AnnBuilder package. For those who are unfamiliar with the AnnBuilder package, this just means that there are two columns separated by a tab where the column on the left contains probe or probeset identifiers and the column on the right contains some sort of widely accepted gene accession. This file should NOT contain a header. SQLForge will then use these IDs along with it's own support databases to make an *AnnotationDbi* package for you. Here is how these IDs should look if you were to read them into R:

```
library(AnnotationForge)
read.table(system.file("extdata", "hcg110_ID",
                      package="AnnotationDbi"),
           sep = "\t", header = FALSE, as.is = TRUE)[1:5,]

##           V1           V2
## 1  1000_at  X60188
## 2  1001_at  X60957
## 3 1002_f_at X65962
## 4 1003_s_at X68149
## 5  1004_at X68149
```

In the example above, Genbank IDs are demonstrated. But it is also possible to use entrez gene IDs, refseq IDs or unigene accessions as the gene identifiers. If refseq IDs are used, it is preferable to strip off the version extensions that can sometimes be added on by some vendors. The version extensions are digits that are sometimes tacked onto the end of a refseq ID and separated from the accession by a dot. As an example consider "NM_000193.2" . The "NM_000193" portion would be the actual accession number and the ".2" would be the version number. These version numbers are not used by these databases and their presence in your input can cause less than desirable results.

Alternatively, if you have an annotation file for an Affymetrix chip, you can use a parameter called affy that will automatically parse such a file and produce a similar mapping from that. It is important to understand however that despite that rather rich contents of an Affymetrix annotation file, almost none of these data are used in the making of an annotation package with SQLForge. Instead, the relevant IDs are stripped out, and then passed along to SQLForge as if you had created a file like is seen above. The option here to use such a

file is offered purely as a convenience because the platform is so popular.

If you have additional information about your probes in the form of other kinds of supported gene IDs, you can pass these in as well by using the `otherSrc` parameter. These IDs must be formatted using the same two column format as described above, and if there are multiple source files, then you can pass them in as a list of strings that correspond to the file paths for these files.

Once you have your IDs ready, SQLForge will read them in, and use the gene IDs to compare to an intermediate database. The data from this database is what is used to make the specialized database that is placed inside of an annotation package.

At the present time, it is possible to make annotation packages for the most common model organisms. For each of these organisms another support package will be maintained and updated biannually which will include all the basic data gathered for this organism from sources such as NCBI, GO, KEGG and Flybase etc. These support packages will each be named after the organism they are intended for and will each include a large sqlite database with all the supporting information for that organism. Please note that support databases are not necessary unless you intend to actually make a new annotation package for one of the supported organisms. In the case where you want to make annotation packages, the support databases are only required for the organism in question. When SQLForge makes a new database, it uses the information supplied by the support database as the data source to make the annotation package. So the relevant support packages needs to be updated to the latest version in order to guarantee that the annotation packages you produce will be made with information from the last biannual update. These support packages are not meant to be annotation packages themselves and they come with no schema of their own. Instead these are merely a way to distribute the data to those who want to make custom annotation packages.

To check if your organism is supported simply look in the metadata packages repository on the bioconductor website for a `.db0` package. Only special organism base packages will end with the `.db0` extension. If you find a package that is named after the organism you are interested in, then your organism is supported, and you can use that database to make custom packages. To list all the supported organism `.db0` packages directly from R you can use `available.db0pkgs()`.

```
available.db0pkgs()

## [1] "anopheles.db0" "arabidopsis.db0" "bovine.db0" "canine.db0"
## [5] "chicken.db0" "chimp.db0" "ecoliK12.db0" "ecoliSakai.db0"
## [9] "fly.db0" "human.db0" "malaria.db0" "mouse.db0"
## [13] "pig.db0" "rat.db0" "rhesus.db0" "worm.db0"
## [17] "xenopus.db0" "yeast.db0" "zebrafish.db0"
```

2 How to use SQLForge

Once you know the name of the package that you need, you can get the latest `.db0` package for your organism by using `biocLite()` like this.

```
source("http://bioconductor.org/biocLite.R")
biocLite("human.db0")
```

```
## Loading required package: human.db0
```

Even if you have installed the `.db0` package before, it's a good idea to run this again to be sure that you have

the latest organism package.

Once you have the appropriate .db0 package, you may also need to install the appropriate org.* package. Most users probably have this already, because using any chip packages for this organism will require it. But if you don't have it yet, you can find a list of these packages it on the web site here:

http://www.bioconductor.org/packages/release/BiocViews.html#___OrgDb

Once you know the name of the package you need, you can install it like before.

```
biocLite("org.Hs.eg.db")
```

All the supported organisms with a .db0 package, will also have a matching org.* package. The org.* packages are named like this: org.jspecies abbreviation.jsource abbreviation.db. The source indicates where the data originates from and also which central ID the data all connects to, while the species abbreviation indicates which organism the data is for. The web site spells out the species a bit more clearly by listing the titles for each package.

Since each organism will have different kinds of data available, the schemas that will be needed for each organism will also change. SQLForge provides support functions for each of the model organisms that will create a sqlite database that complies with a specified database schema. The following helper function will list supported Schemas.

```
available.dbschemas()

## [1] "ARABIDOPSISCHIP_DB" "BOVINECHIP_DB"      "BOVINE_DB"          "CANINECHIP_DB"
## [5] "CANINE_DB"          "CHICKENCHIP_DB"     "CHICKEN_DB"         "ECOLICHIP_DB"
## [9] "ECOLI_DB"           "FLYCHIP_DB"         "FLY_DB"             "GO_DB"
## [13] "HUMANCHIP_DB"       "HUMANCROSSCHIP_DB"  "HUMAN_DB"           "INPARANOIDDROME_DB"
## [17] "INPARANOIDHOMSA_DB" "INPARANOIDMUSMU_DB" "INPARANOIDRATNO_DB" "INPARANOIDSACCE_DB"
## [21] "KEGG_DB"            "MALARIA_DB"         "MOUSECHIP_DB"       "MOUSE_DB"
## [25] "PFAM_DB"            "PIGCHIP_DB"         "PIG_DB"             "RATCHIP_DB"
## [29] "RAT_DB"             "WORMCHIP_DB"        "WORM_DB"            "YEASTCHIP_DB"
## [33] "YEAST_DB"           "ZEBRAFISHCHIP_DB"   "ZEBRAFISH_DB"
```

The following shows an example of how to make a chip package:

```
hcg110_IDs = system.file("extdata",
                          "hcg110_ID",
                          package="AnnotationDbi")

tmpout = tempdir()

makeDBPackage("HUMANCHIP_DB",
              affy=FALSE,
              prefix="hcg110",
              fileName=hcg110_IDs,
              baseMapType="gb",
              outputDir = tmpout,
              version="1.0.0",
              manufacturer = "Affymetrix",
              chipName = "Human Cancer G110 Array",
```

Wrapper functions are provided for making all of the different kinds of chip based package types that are


```
##What is the mapping we just made?
hgu95av2NAMESYMBOL

## NAMESYMBOL map for hgu95av2.db (object of class "AnnDbBimap")

##Display the 1st 4 relationships in this new mapping
as.list(hgu95av2NAMESYMBOL)[1:4]

## `$alpha-1-B glycoprotein`
## [1] "A1BG"
##
## `$alpha-2-macroglobulin`
## [1] "A2M"
##
## `$alpha-2-macroglobulin pseudogene 1`
## [1] "A2MP1"
##
## `$N-acetyltransferase 1 (arylamine N-acetyltransferase)`
## [1] "NAT1"
```

If instead of creating a mapping on an existing example, you wanted to add a new mapping to your customized annotation package, you would need to call this function from zzz.R in your modified annotation package (and also expose it in the namespace). You will then want to be sure that your updated database has replaced the one in the inst/extdata directory that was originally generated by SQLForge. And finally, you will need to also put a man page into your package so that users will know how to make use of this new mapping.

4 Session Information

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

```
## R version 3.2.3 (2015-12-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows Server 2008 R2 x64 (build 7601) Service Pack 1
##
## locale:
## [1] LC_COLLATE=C LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252 LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] parallel stats4 stats graphics grDevices utils datasets methods
## [9] base
##
## other attached packages:
## [1] hgu95av2.db_3.2.2 human.db0_3.2.4
## [3] AnnotationForge_1.12.2 hom.Hs.inp.db_3.1.2
## [5] Biostings_2.38.3 XVector_0.10.0
## [7] XML_3.98-1.3 Homo.sapiens_1.3.1
```

```
## [9] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2 org.Hs.eg.db_3.2.3
## [11] GO.db_3.2.2 RSQlite_1.0.0
## [13] DBI_0.3.1 OrganismDbi_1.12.1
## [15] GenomicFeatures_1.22.8 GenomicRanges_1.22.3
## [17] GenomeInfoDb_1.6.1 AnnotationDbi_1.32.3
## [19] IRanges_2.4.6 S4Vectors_0.8.9
## [21] Biobase_2.30.0 BiocGenerics_0.16.1
## [23] knitr_1.12
##
## loaded via a namespace (and not attached):
## [1] BiocInstaller_1.20.1 formatR_1.2.1 futile.logger_1.4.1
## [4] highr_0.5.1 bitops_1.0-6 futile.options_1.0.0
## [7] tools_3.2.3 zlibbioc_1.16.0 biomaRt_2.26.1
## [10] evaluate_0.8 graph_1.48.0 rtracklayer_1.30.1
## [13] stringr_1.0.0 RBGL_1.46.0 BiocParallel_1.4.3
## [16] lambda.r_1.1.7 magrittr_1.5 Rsamtools_1.22.0
## [19] GenomicAlignments_1.6.3 SummarizedExperiment_1.0.2 BiocStyle_1.8.0
## [22] stringi_1.0-1 RCurl_1.95-4.7 markdown_0.7.7
```