Basic Functions of AnnBuilder

Jianhua Zhang

October 22, 2008

1 Introduction

This vignette is an overview of some of the functions that can be used to build an annotation data package. The purpose of this vignette is to provide guidance for users who are comfortable using the data package building procedures described in `ABPrimer` but would like to have more freedom in building customized data packages. First time users of AnnBuilder are suggested to go through the `ABPrimer` vignette before trying the code here.

Functions contained by *AnnBuilder* include:

```
library(AnnBuilder)
pkgpath <- .find.package("AnnBuilder")
docFiles <- file.path(pkgpath, c("TITLE", "DESCRIPTION", "INDEX"))
headers <- c("", "Description:\n\n", "Index:\n\n")
footers <- c("\n", "\n", "")
for (i in which(file.exists(docFiles))) {
    writeLines(headers[i], sep = "")
    writeLines(readLines(docFiles[i]) )
    writeLines(footers[i], sep = "")
}
```

AnnBuilder relies on these functions to build annotation data packages by extracting data from the following potential public data repositories.

- Entrez Gene (`ftp://ftp.ncbi.nlm.nih.gov/gene/DATA`) to obtain mappings to Gene IDs and annotation data.
- UniGene(`ftp://ftp.ncbi.nih.gov/repository/UniGene/Homo_sapiens/Hs.data.gz`) to obtain mappings to LocusLink ids from ESTs

- GoldenPath(`http://www.genome.ucsc.edu/goldenPath/14nov2002/database`). Two data files (refLink.txt.gz and refGene.txt.gz) are used to obtain chromosomal location and orientation data
- Gene Ontology(`"http://www.godatabase.org/dev/database/archive/latest/go_200506-termdb.rdf-xml.gz"`) to obtain gene ontology terms and relationships among terms.
- KEGG(`ftp://ftp.genome.ad.jp/pub/kegg/pathway/organisms`) to obtain pathway and enzyme data for genes. Several data files may be used depending on the organism of interest.

HomoloGene A data file provided by `ftp://ftp.ncbi.nih.gov/pub/old/HomoloGene/` will be used to extract mappings between LocusLink ids and HomoloGene ids.

The urls with date components may change when the maintainers update the data. However, *AnnBuilder* has the ability to figure out the latest updates and use the corresponding data for annotation as long as the current path structure of the urls remain. Source data will be downloaded from the urls given.

Each of the public data repositories is represented as an object of a S4 class. Common methods for an object include `readData` that reads in data line by line and `parseData` that parses data based on the instructions given in a segment of Perl code. In both cases, data are downloaded from the source url and then processed locally.

As data from the aforementioned data sources are usually large, truncated versions of the corresponding data will be used to ensure reasonable speed. These files have already been stored in Bioconductor web site. Thus, the source urls will be different for a real annotation project.

Getting Source Data

Suppose we are interested in annotating genes on Affymetrix HG_U95av2 gene chip. A file containing a column for Affymetrix probe ids and another for mappings to GenBank accession numbers can be produced based on the data file provided by Affymetrix and then used as the base to extract annotation data from different data sources. The base file has to be saved as a text file with the two columns separated by a delimiter (e. g. a tab - ""). Here we just create a truncated one on the fly and store it in the current working directory.

```
geneNMap <- matrix(c("32468_f_at", "D90278;M16652", "32469_at", "L00693",
                "32481_at", "AL031663", "33825_at", "X68733",
                "35730_at", "X03350", "36512_at", "L32179",
                "38912_at", "D90042", "38936_at", "M16652",
                "39368_at", "AL031668"), ncol = 2, byrow = TRUE)
write.table(geneNMap, file = "geneNMap", sep = "\t", quote = FALSE,
        row.names = FALSE, col.names = FALSE)
```

We can see that the file has two columns for Affymetrix probe ids and the matching GenBank accession numbers:

```
geneNMap
```

The first step to annotating these probe ids in the base file is to map them to Entrez Gene ids and then use mapped gene ids as the point of linkage to other annotation data provided by various data sources. As Affymetrix probe ids (probes for other platform as well) may be mapped to gene ids through Entrez Gene and UniGene (and other sources), each of which can be complementary to each other, we may want to get the mappings from all the available sources and then combine the results to ensure completeness. *Annbuilder* has a unifying mechanism that allows users to unify mapping information from different sources to obtained a combined result that is assumed to be more reliable.

In this vignette, we first would like to map the probes to Entrez Gene ids using data from both Entrez Gene and UniGene. The following code creates objects for Entrez Gene and UniGene with parsers needed to parse the source data file for mapping Affymetrix probe ids in baseF to gene ids.

```
    makeSrcInfo()
    srcObjs <- list()
    egUrl <-
            "http://www.bioconductor.org/datafiles/wwwsources"
    ugUrl <-
            "http://www.bioconductor.org/datafiles/wwwsources/Ths.data.gz"
    eg <- EG(srcUrl = egUrl, parser = file.path(pkgpath, "scripts",
          "gbLLParser"), baseFile = "geneNMap", accession = "Tll_tmpl.gz",
          built = "N/A", fromWeb = TRUE)
    ug <- UG(srcUrl = ugUrl, parser = file.path(pkgpath,
           "scripts", "gbUGParser"), baseFile = "geneNMap",
        organism = "Homo sapiens", built = "N/A", fromWeb = TRUE)
    srcObjs[["eg"]] <- eg
    srcObjs[["ug"]] <- ug
\begin{verbatim}

Again, the urls used in the example are for demonstration purpose
only. eg and ug objects also take a parser as an argument. A parser is
a segment of a Perl script that contains instructions on how the data
source will be parsed and how the output will be generated. Please
refer to the documents for pubRepo for detailed information on parsers
and the objects for various public data repositories. The parser for eg used in the exa

Each object has a function named parserData that can be invoked to obtain the parsed da
should be set to FALSE if the source data has been stored locally. The
```

following code needs human intervention under windows and is therefore turned off. Copying the code chunk and then pasting into an R session under windows should work.

```
\begin{verbatim}
    # This portion only runs interactively under Windows (copy/paste)
    if(.Platform$OS.type != "windows"){
        llMapping <- parseData(eg, eg@accession)
        colnames(llMapping) <- c("PROBE", "EG")
        ugMapping <- parseData(ug)
        colnames(ugMapping) <- c("PROBE", "UG")
    }
```

The parsed data from LocusLink and UniGene are:

```
    # This portion only runs after the previous code has been
    # executed under windows
    if(.Platform$OS.type != "windows"){
        llMapping
        ugMapping
    }
```

Please note the differences between the mappings from the two sources and some of the Affymetrix probe ids can be mapped to multiple Gene ids and ";" is used to separate multiple mappings in such cases.

The mappings obtained from the two sources are then unified to obtain a comprehensive mapping between Affymetrix probe ids and gene ids. The unified mappings are saved in a file show below:

```
    # This portion only runs interactively under Windows (copy/paste)
    base <- matrix(scan("geneNMap", what = "", sep = "\t", quote = "",
                    quiet = TRUE), ncol = 2, byrow = TRUE)
    colnames(base) <- c("PROBE", "ACC")
    merged <- merge(base, llMapping, by = "PROBE", all.x = TRUE)
    merged <- merge(merged, ugMapping, by = "PROBE", all.x = TRUE)
    unified <- AnnBuilder:::resolveMaps(merged, trusted = c("EG", "UG"),
                          srcs = c("EG", "UG"))
    unified
```

In the above code, "EG" has been identified as the trusted source meaning that when the two sources provide conflicting mappings, the one from Entrez Gene will be used. The unified mapping has four columns with the first one for Affymetrix probe ids, second for GenBank accession numbers, third for mappings to gene ids, and forth for the number of sources that agreed with the mappings.

```
read.table(unified, sep = "\t", header = FALSE)
```

The unified mappings can then be used as the base file to parse the data from Entrez Gene to obtain annotation data for each of the Affymetrix probe ids. To do so, we need to assign a new parser that processes the data from Entrez to get annotation data including gene name, chromosomal location, and so on. Again, the parser works for the example only.

```
    # This portion only runs interactively under Windows (copy/paste)
    if(.Platform$OS.type != "windows"){
        parser(eg) <- file.path(.path.package("AnnBuilder"),
                                "scripts", "llParser")
        baseFile(eg) <- unified
        annotation <- parseData(eg, eg@accession, ncol = 14)
        colnames(annotation) <- c("PROBE", "ACCNUM", "LOCUSID", "UNIGENE",
                                  "GENENAME", "SYMBOL","CHR", "MAP",
                                  "PMID", "GRIF", "SUMFUNC", "GO",
                                        "OMIM", "REFSEQ")
    }
```

The annotation data obtained has 12 columns for the elements indicated by the column names. Let us view the chromosomal number of the Affymetrix probe ids.

```
annotation[,c("PROBE", "LOCUSID")]
```

Other annotation data can be obtained from other sources. In this vignette, we try to get data from GoldenPath for chromosomal location and orientation and Gene Ontology for ontology terms and relations among terms. As usual, we create the objects with truncated data from Bioconductor rather than the actual web site. Two source data files (Tlink.txt.gz and TGene.txt.gz) have to be downloaded/unzipped from GoldenPath (`http://www.genome.ucsc.edu/goldenPath/10april2003/database/`) in order to obtain the chromosome location data. We only have to provide the url under unix as the system knows how to get the latest version of the two files.

```
gpUrl <- "http://www.bioconductor.org/datafiles/wwwsources/"
goUrl <- "http://www.bioconductor.org/datafiles/wwwsources/Tgo.xml"
gp <- GP(srcUrl = gpUrl, organism = "Homo sapiens", fromWeb = TRUE)
go <- GO(srcUrl = goUrl, fromWeb = TRUE)
```

To get the chromosomal data from GoldenPath with the actual url, one only needs to call a function called getStrand by typing "strand <- getStrand(gp)" where gp is the object for goldenPath with correct url. In this vignette, however, we take a somewhat different approach to get the data as we are using a truncated set of data from a dummy.

```
strand <- getChroLocation(srcUrl(gp), gpLinkNGene(TRUE))
```

The data processed are then merged with the annotation we previously obtained.

```
annotation <- merge(annotation, strand, by = "LOCUSID", all.x = TRUE)
```

To generate an R data package containing the annotation data, we first create an empty package and then populate the package with data and F fucntions.

```
pkgName <- "test"
pkgPath <- getwd()
createEmptyDPkg("test", getwd(), force = TRUE)
annotation <- as.matrix(annotation)
writeAnnData2Pkg(annotation, pkgName, pkgPath)
revNames <- intersect(colnames(annotation),
                      c("PMID", "PATH", "ENZYME"))
if(length(revNames) != 0){
    writeReverseMap(annotation[, c("PROBE", revNames)],
                    pkgName, pkgPath)
}
```

The data package is stored in the current working directory under the name textttttest.

```
list.files(file.path(getwd(), "test"))
```

As can be seen, the data package contains all the required elements of a normal R package and can be installed in the same way as an R package. The annotation data are all stored as rda files in the data directory.

```
list.files(file.path(getwd(), "test", "data"))
```

Each of the rda files contains key and value pairs with the key being Affymetrix probe ids and value being the annotation element in this case.

The last step is to write the needed documentations and statistic data for quality control purpose. The following code can be used to generate the required documentations for the data package. Some part of the code may fail as the urls used may subject to changes by maintainers of the web sits in the future.

```
repList <- getRepList("all", srcObjs)
repList[["PKGNAME"]] <- pkgName
chrLengths <- getChrLengths("Homo sapiens")
writeOrganism(pkgName, pkgPath, "Homo sapiens")
writeChrLength(pkgName, pkgPath, chrLengths)
writeDocs("geneNMap", pkgName, pkgPath, "1.1.0",
          list(author = "annonymous", maintainer = "annonymous@net.com"),
          repList, "PKGNAME")
```

Now, we can clean up the mess we have left.

```
unlink(c(unified, XMLOut, "geneNMap", "test.xml", "testByNum.xml"))
unlink(file.path(getwd(), "test"), TRUE)
```

2 Session Information

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

```
R version 2.8.0 (2008-10-20)
x86_64-unknown-linux-gnu

locale:
LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=C;LC_MESSAGES=en

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

other attached packages:
[1] AnnBuilder_1.20.0   annotate_1.20.0     xtable_1.5-4
[4] AnnotationDbi_1.4.0 XML_1.98-1          Biobase_2.2.0

loaded via a namespace (and not attached):
[1] DBI_0.2-4     RSQLite_0.7-0
```