Basic Functions of AnnBuilder

Jianhua Zhang

June 23, 2003

©2003 Bioconductor

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 `HowTo` but would like to have more freedom in building customized data packages. First time users of AnnBuilder are suggested to go through the `HowTo` vignette before trying the code here.

Functions contained by *AnnBuilder* include:

```
> library(AnnBuilder)

Welcome to Bioconductor
         Vignettes contain introductory material.  To view,
         simply type: openVignette()
         For details on reading vignettes, see
         the openVignette help page.
Creating a new generic function for "summary" in package
reposTools

> 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 = "")
+ }
```

```
Description:

Package: AnnBuilder
Version: 1.2.12
Date: 2003-5-13
Title: Bioconductor annotation data package builder
Author: J. Zhang <jzhang@jimmy.harvard.edu>
Maintainer: J. Zhang <jzhang@jimmy.harvard.edu>
Depends: R (>= 1.6.0), Biobase
Description: Processing annotation date from public data repositories
        and building annoation data packages or XML data documents
        using the source data.
Keyword: annotation
License: LGPL
Built: R 1.7.1; ; 2003-06-23 12:47:14

Index:

ABPkgBuilder            Functions that support a single API for
                        building data packages
AnnInfo                 A text file containing annotaion data
                        information
GEO-class               Class "GEO" represents a GEO object that
                        reads/downloads data from the GEO web site
GO-class                Class "GO" a class to handle data from Gene
                        Ontology
GOPkgBuilder            A functions to builder a data package using GO
                        data
GOXMLParser             Functions to read/parse the XML document of
                        Gene Ontology data
GP-class                Class "GP" a sub-class of pubRepo to
                        get/process data from GoldenPath
KEGG-class              Class "KEGG" a sub-class of pubRepo to
                        get/process pathway and enzyme information
KEGGPkgbuilder          A function to make the data package for KEGG
LL-class                Class "LL" a sub-class of pubRepo to handle
                        data from LocusLink
UG-class                Class "UG" a sub-class of pubRepo to handle
                        data from UniGene
YG-class                Class "YG" a sub-class of pubRepo that
                        reads/downloads data from yeast genomic
cols2Env                Creates a environment object using data from
```

	two columns of a matrix
fileMuncher	Dynamically create a Perl script to parse a source file base on user specifications
fileToXML	A function to convert a text file to XML.
getChroLocation	Functions to extract data from golden path
getDPStats	A function to read in the statistics about a data package
getKEGGIDNName	Functions to get/process pathway and enzyme data from KEGG
getSrcBuilt	Functions that get the built date or number of the source data used for annotation
getSrcUrl	Functions that find the correct url for downloading annotation data
getYeastData	Functions to get/process yeast genome data
loadFromUrl	Functions to load files from a web site
makeSrcInfo	Functions to make source information available for later use
print.ABQCList	Prints the quality control results for a given data package in a nice format
pubRepo-class	Class "pubRepo" a generic class for downloading/parsing data provided by various public data repositories
queryGEO	Function to extract a data file from the GEO web site
resolveMaps	Functions to obtain unified mappings between ids
sourceURLs	A data file contains urls for data available from various public repositories
unifyMappings	A function to unify mapping result from different sources
writeChrLength	Functions that creates binary files for chromosome length and organism
writeManPage	Functions that write supporting files needed by a data package
writeXMLHeader	A function to write header information to an XML file.
yeastAnn	Functions to annotate yeast genom data
yeastPkgBuilder	Functions to do a data package for yeast genome

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

- LocusLink(`ftp://ftp.ncbi.nih.gov/refseq/LocusLink/LL\protect\T1\textunderscoretmpl.` `gz`) to obtain mappings to LocusLink ids and annotation data.
- UniGene(`ftp://ftp.ncbi.nih.gov/repository/UniGene/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/2003-02-01/go\protect\T1\textunderscore200302-termdb.xml.gz`) to obtain gene ontology terms and relationships among terms.
- KEGG(`ftp://ftp.genome.ad.jp/pub/kegg/pathways`) to obtain pathway and enzyme data for genes. Several data files may be used depending on the organism of interest.

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. For unix users, source data will be downloaded from the urls given. For windows users, however, source data have to be downloaded/unzipped manually and stored locally. Names of the stored files will be used to replace the urls in the code examples of this vignette. This rule applies to all the source data list above except KEGG, for which an url can be used under both unix and windows.

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 save 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", "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
```

```
      [,1]         [,2]
 [1,] "32468_f_at" "D90278"
 [2,] "32469_at"   "L00693"
 [3,] "32481_at"   "AL031663"
 [4,] "33825_at"   " X68733"
 [5,] "35730_at"   "X03350"
 [6,] "36512_at"   "L32179"
 [7,] "38912_at"   "D90042"
 [8,] "38936_at"   "M16652"
 [9,] "39368_at"   "AL031668"
```

The first step to annotating these probe ids in the base file is to map them to LocusLink ids and then use mapped LocusLink 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 LocusLink ids through LocusLink 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 LocusLink ids using data from both LocusLink and UniGene. The following code creates objects for LocusLink and UniGene with parsers needed to parse the source data file for mapping Affymetrix probe ids in baseF to LocusLink ids. For windows users, we have downloaded/unzipped the sample files and stored them in the `data` directory of `AnnBuilder`.

```
> makeSrcInfo()
> if (.Platform$OS.type == "unix") {
+     llUrl <- "http://www.bioconductor.org/datafiles/wwwsources/Tll_tmpl.gz"
+     ugUrl <- "http://www.bioconductor.org/datafiles/wwwsources/Ths.data.gz"
+ } else {
```

```
+     llUrl <- file.path(pkgpath, "data", "Tll_tmpl")
+     ugUrl <- file.path(pkgpath, "data", "Ths.data")
+ }
> ll <- LL(srcUrl = llUrl, parser = file.path(pkgpath, "data",
+     "gbLLParser"), baseFile = "geneNMap")
> ug <- UG(srcUrl = ugUrl, parser = file.path(pkgpath, "data",
+     "gbUGParser"), baseFile = "geneNMap", organism = "human")
```

Again, the urls used in the example are for demonstration purpose only. ll 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. Each object has a function named parserData that can be invoked to obtain the parsed data. `parserData` has a parameter - `fromWeb` that should be set to FALSE if the source data has been stored locally, especially for windows.

```
> if (.Platform$OS.type == "unix") {
+     fromWeb <- TRUE
+ } else {
+     fromWeb <- FALSE
+ }
> if (.Platform$OS.type != "windows") {
+     llMapping <- parseData(ll, fromWeb = fromWeb)
+     colnames(llMapping) <- c("PROBE", "LL")
+     ugMapping <- parseData(ug, fromWeb = fromWeb)
+     colnames(ugMapping) <- c("PROBE", "UG")
+ }
```

The parse data from LocusLink and UniGene are:

```
> if (.Platform$OS.type != "windows") {
+     llMapping
+     ugMapping
+ }
```

```
     PROBE        UG
[1,] "38912_at"   "1084;63036"
[2,] "32469_at"   "10;1084"
[3,] "35730_at"   "7051"
[4,] "32468_f_at" "125"
[5,] "36512_at"   "1084"
[6,] "38936_at"   "10;NA"
[7,] "32481_at"   "63036"
[8,] "39368_at"   "NA"
```

Please note the differences between the mapping from the two sources and some of the Affymetrix probe ids can be mapped to multiple LocusLink 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 LocusLink ids. The unified mappings are saved in a file show below:

```
> base <- matrix(scan("geneNMap", what = "", sep = "\t", quote = "",
+     quiet = TRUE), ncol = 2, byrow = TRUE)
> colnames(base) <- c("PROBE", "ACC")
> if (.Platform$OS.type != "windows") {
+     merged <- merge(base, llMapping, by = "PROBE", all.x = TRUE)
+     merged <- merge(merged, ugMapping, by = "PROBE", all.x = TRUE)
+     unified <- resolveMaps(merged, trusted = c("LL", "UG"), srcs = c("LL",
+         "UG"))
+     unified
+ }
```

```
[1] "/misc/homes/madman/R-1.7.0p/library/AnnBuilder/temp/tempFile4287"
```

In the above code, "LL" has been identified as the trusted source meaning that when the two sources provide conflicting mappings, the one from LocsuLink 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 LocusLink ids, and forth for the number of sources that agreed with the mappings.

```
> if (.Platform$OS.type != "windows") {
+     read.table(unified, sep = "\t", header = FALSE)
+ }
```

```
          V1       V2    V3 V4
1 32468_f_at   D90278   125  1
2   32469_at   L00693    10  1
3   32481_at AL031663 63036  1
4   33825_at   X68733    NA  4
5   35730_at   X03350  7051  1
6   36512_at   L32179    10  1
7   38912_at   D90042  1084  1
8   38936_at   M16652    10  1
9   39368_at AL031668     1 NA
```

The unified mappings can then be used as the base file to parse the data from LocusLink 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 LocusLink to get annotation data including gene name, chromosomal location, and so on.

```
> if (.Platform$OS.type != "windows") {
+     parser(ll) <- file.path(.path.package("AnnBuilder"), "data",
+         "llParser")
+     baseFile(ll) <- unified
+     annotation <- parseData(ll, ncol = 12, fromWeb = fromWeb)
+     colnames(annotation) <- c("PROBE", "ACCNUM", "LOCUSID", "UNIGENE",
+         "GENENAME", "SYMBOL", "CHR", "MAP", "PMID", "GRIF", "SUMFUNC",
+         "GO")
+ }
```

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.

```
> if (.Platform$OS.type != "windows") {
+     annotation[, c("PROBE", "LOCUSID")]
+ }
```

```
      PROBE        LOCUSID
 [1,] "36512_at"   "125"
 [2,] "38936_at"   "10"
 [3,] "32469_at"   "63036"
 [4,] "38912_at"   "NA"
 [5,] "32468_f_at" "7051"
 [6,] "35730_at"   "10"
 [7,] "39368_at"   "1084"
 [8,] "32481_at"   "10"
 [9,] "33825_at"   "1"
```

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. The following code is only applicable to an unix platform. For windows, users have to download/unzip the source data manually and replace the urls with the file names of the downloaded files. 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 loacation data. We only have to provide the url under unix as the system knows how to get the latest version of the two files. Under windows, however, file names the two downloaded files have to be given as a named vector with file names as the values and ”link” and ”gene” as the names for Tlink.txt.gz and TGene.txt.gz, respectively.

```
> if (.Platform$OS.type == "unix") {
+     gpUrl <- "http://www.bioconductor.org/datafiles/wwwsources/"
+     goUrl <- "http://www.bioconductor.org/datafiles/wwwsources/Tgo.xml"
+ } else {
+     gpUrl <- c(link = file.path(.path.package("AnnBuilder"),
+         "data", "TLink"), gene = file.path(.path.package("AnnBuilder"),
+         "data", "TGene"))
+     goUrl <- file.path(.path.package("AnnBuilder"), "data", "TGO.xml")
+ }
> gp <- GP(srcUrl = gpUrl, organism = "human")
> go <- GO(srcUrl = goUrl)
```

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.

```
> if (.Platform$OS.type != "windows") {
+     strand <- getChroLocation(srcUrl(gp), gpLinkNGene(TRUE),
+         fromWeb = fromWeb)
+ }
```

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

```
> if (.Platform$OS.type != "windows") {
+     annotation <- merge(annotation, strand, by = "LOCUSID", all.x = TRUE)
+ }
```

Data from yet some other sources can be processed and merged to the annotation data following the same procedures as described above. When data from all the desired sources have been obtained, an XML data document and an R data package can be produced for easy distribution and usage. The following code generates an XML document containing the annotation data. textttmultC is used to identify elements (columns in data object texttt annotation)that have many to one relations to probe ids. When the data were parsed by the parser, ";" was used to separate multiple entries for these elements (e.g. a probe id may be related to several PubMed ids as shown by 256352;63254;4687264). texttttypeC identifies elements that also have many to one relation to probe ids. Additionally, they also contain another attribute that are appended to the end of the mapped value with "@" as a separator (e.g. GO:0001234@E;GO:0004875@NR). These elements need to be dealt with differently.

```
> if (.Platform$OS.type != "windows") {
+     multC <- c("PMID", "CHR")
+     typeC <- c("GO", "CHRLOC")
+     XMLOut <- file.path(getwd(), "test.xml")
+     fileToXML(targetName = "hgu95a", outName = XMLOut, inName = annotation,
+         colNames = "", idColName = "PROBE", multColNames = multC,
+         typeColNames = typeC, isFile = FALSE, version = "1.0.0")
+ }
```

The contents of the XML document generated are shown below.

```
> if (.Platform$OS.type != "windows") {
+     readLines(XMLOut)
+ }
```

```
  [1] "<?xml version = \"1.0\" encoding = \"UTF-8\" standalone = \"yes\"?>"
  [2] "<!DOCTYPE AnnBuilder: SYSTEM \"http://www.bioconductor.org/datafiles/dtds/annota
  [3] "<AnnBuilder:Annotate xmlns:AnnBuilder = 'http://www.bioconductor.org/AnnBuilder/
  [4] "<AnnBuilder:Attr>"
  [5] "<AnnBuilder:Target value = \"hgu95a\"/>"
  [6] "<AnnBuilder:DateMade value = \"Mon Jun 23 13:29:44 2003\"/>"
  [7] "<AnnBuilder:Version value = \"1.0.0\"/>"
  [8] "<AnnBuilder:SourceFile url = \"ftp://ftp.ncbi.nih.gov/refseq/LocusLink/LL_tmpl.g
  [9] "<AnnBuilder:SourceFile url = \"http://www.godatabase.org/dev/database/archive/20
 [10] "<AnnBuilder:SourceFile url = \"ftp://ftp.genome.ad.jp/pub/kegg/pathways\" built
 [11] "<AnnBuilder:SourceFile url = \"http://www.genome.ucsc.edu/goldenPath/10april2003
 [12] "<AnnBuilder:SourceFile url = \"ftp://ftp.ncbi.nih.gov/repository/UniGene/Hs.data
 [13] "<AnnBuilder:Entryid value = \"LocusLink identifier\"/>"
 [14] "<AnnBuilder:Element value = \"PROBE\" describ = \"Generic identifier\"/>"
 [15] "<AnnBuilder:Element value = \"ACCNUM\" describ = \"GenBank accession number\"/>"
 [16] "<AnnBuilder:Element value = \"UNIGENE\" describ = \"UniGene cluster identifier a
 [17] "<AnnBuilder:Element value = \"GENENAME\" describ = \"Gene Description used for g
 [18] "<AnnBuilder:Element value = \"SYMBOL\" describ = \"Symbol used for gene reports\
 [19] "<AnnBuilder:Element value = \"CHR\" describ = \"Chromosome assignment\"/>"
 [20] "<AnnBuilder:Element value = \"MAP\" describ = \"Cytoband location of gene \"/>"
 [21] "<AnnBuilder:Element value = \"PMID\" describ = \"PubMed unique identifier\"/>"
 [22] "<AnnBuilder:Element value = \"GRIF\" describ = \"PubMed unique identifier\"/>"
 [23] "<AnnBuilder:Element value = \"SUMFUNC\" describ = \"A brief summary of the funct
 [24] "<AnnBuilder:Element value = \"GO\" describ = \"Gene Ontology identifier\"/>"
 [25] "<AnnBuilder:Element value = \"CHRLOC\" describ = \"Chromosomal location \"/>"
 [26] "</AnnBuilder:Attr>"
 [27] "<AnnBuilder:Data>"
 [28] "<AnnBuilder:Entry id=\"33825_at\" describ = \"PROBE\">"
 [29] "\t<AnnBuilder:Item name=\"LOCUSID\" value=\"1\" />"
 [30] "\t<AnnBuilder:Item name=\"ACCNUM\" value=\"AL031668\" />"
 [31] "\t<AnnBuilder:Item name=\"UNIGENE\" value=\"Hs.41997\" />"
 [32] "\t<AnnBuilder:Item name=\"GENENAME\" value=\"alpha-1-B glycoprotein\" />"
```

```
[33] "\t<AnnBuilder:Item name=\"SYMBOL\" value=\"A1BG\" />"
[34] "\t<AnnBuilder:Item name=\"MAP\" value=\"19cen-q13.2\" />"
[35] "\t<AnnBuilder:Item name=\"GRIF\" value=\"NA\" />"
[36] "\t<AnnBuilder:Item name=\"SUMFUNC\" value=\"NA\" />"
[37] "\t<AnnBuilder:Item name=\"PMID\" value=\"8889549\" />"
[38] "\t<AnnBuilder:Item name=\"PMID\" value=\"2591067\" />"
[39] "\t<AnnBuilder:Item name=\"CHR\" value=\"19\" />"
[40] "NULL"
[41] "NULL"
[42] "</AnnBuilder:Entry>"
[43] "<AnnBuilder:Entry id=\"38936_at\" describ = \"PROBE\">"
[44] "\t<AnnBuilder:Item name=\"LOCUSID\" value=\"10\" />"
[45] "\t<AnnBuilder:Item name=\"ACCNUM\" value=\"L00693\" />"
[46] "\t<AnnBuilder:Item name=\"UNIGENE\" value=\"Hs.2\" />"
[47] "\t<AnnBuilder:Item name=\"GENENAME\" value=\"N-acetyltransferase 2 (arylamine N-
[48] "\t<AnnBuilder:Item name=\"SYMBOL\" value=\"NAT2\" />"
[49] "\t<AnnBuilder:Item name=\"MAP\" value=\"NA\" />"
[50] "\t<AnnBuilder:Item name=\"GRIF\" value=\"NA\" />"
[51] "\t<AnnBuilder:Item name=\"SUMFUNC\" value=\"Arylamine N-acetyltransferase 2; N-
[52] "\t<AnnBuilder:Item name=\"PMID\" value=\"8460648\" />"
[53] "\t<AnnBuilder:Item name=\"PMID\" value=\"8102597\" />"
[54] "\t<AnnBuilder:Item name=\"PMID\" value=\"7915226\" />"
[55] "\t<AnnBuilder:Item name=\"PMID\" value=\"7773298\" />"
[56] "\t<AnnBuilder:Item name=\"PMID\" value=\"2734109\" />"
[57] "\t<AnnBuilder:Item name=\"PMID\" value=\"2340091\" />"
[58] "\t<AnnBuilder:Item name=\"PMID\" value=\"1968463\" />"
[59] "\t<AnnBuilder:Item name=\"PMID\" value=\"1676262\" />"
[60] "\t<AnnBuilder:Item name=\"PMID\" value=\"1381364\" />"
[61] "\t<AnnBuilder:Item name=\"PMID\" value=\"13\" />"
[62] "\t<AnnBuilder:Item name=\"CHR\" value=\"8\" />"
[63] "\t<AnnBuilder:Item name=\"GO\" value=\"GO:0004060\" type=\"E\"/>"
[64] "\t<AnnBuilder:Item name=\"CHRLOC\" value=\"-830797\" type=\"10\"/>"
[65] "</AnnBuilder:Entry>"
[66] "<AnnBuilder:Entry id=\"35730_at\" describ = \"PROBE\">"
[67] "\t<AnnBuilder:Item name=\"LOCUSID\" value=\"10\" />"
[68] "\t<AnnBuilder:Item name=\"ACCNUM\" value=\"L32179\" />"
[69] "\t<AnnBuilder:Item name=\"UNIGENE\" value=\"Hs.2\" />"
[70] "\t<AnnBuilder:Item name=\"GENENAME\" value=\"N-acetyltransferase 2 (arylamine N-
[71] "\t<AnnBuilder:Item name=\"SYMBOL\" value=\"NAT2\" />"
[72] "\t<AnnBuilder:Item name=\"MAP\" value=\"NA\" />"
[73] "\t<AnnBuilder:Item name=\"GRIF\" value=\"NA\" />"
[74] "\t<AnnBuilder:Item name=\"SUMFUNC\" value=\"Arylamine N-acetyltransferase 2; N-
```

```
 [75] "\t<AnnBuilder:Item name=\"PMID\" value=\"8460648\" />"
 [76] "\t<AnnBuilder:Item name=\"PMID\" value=\"8102597\" />"
 [77] "\t<AnnBuilder:Item name=\"PMID\" value=\"7915226\" />"
 [78] "\t<AnnBuilder:Item name=\"PMID\" value=\"7773298\" />"
 [79] "\t<AnnBuilder:Item name=\"PMID\" value=\"2734109\" />"
 [80] "\t<AnnBuilder:Item name=\"PMID\" value=\"2340091\" />"
 [81] "\t<AnnBuilder:Item name=\"PMID\" value=\"1968463\" />"
 [82] "\t<AnnBuilder:Item name=\"PMID\" value=\"1676262\" />"
 [83] "\t<AnnBuilder:Item name=\"PMID\" value=\"1381364\" />"
 [84] "\t<AnnBuilder:Item name=\"PMID\" value=\"13\" />"
 [85] "\t<AnnBuilder:Item name=\"CHR\" value=\"8\" />"
 [86] "\t<AnnBuilder:Item name=\"GO\" value=\"GO:0004060\" type=\"E\"/>"
 [87] "\t<AnnBuilder:Item name=\"CHRLOC\" value=\"-830797\" type=\"10\"/>"
 [88] "</AnnBuilder:Entry>"
 [89] "<AnnBuilder:Entry id=\"32481_at\" describ = \"PROBE\">"
 [90] "\t<AnnBuilder:Item name=\"LOCUSID\" value=\"10\" />"
 [91] "\t<AnnBuilder:Item name=\"ACCNUM\" value=\"M16652\" />"
 [92] "\t<AnnBuilder:Item name=\"UNIGENE\" value=\"Hs.2\" />"
 [93] "\t<AnnBuilder:Item name=\"GENENAME\" value=\"N-acetyltransferase 2 (arylamine N-
 [94] "\t<AnnBuilder:Item name=\"SYMBOL\" value=\"NAT2\" />"
 [95] "\t<AnnBuilder:Item name=\"MAP\" value=\"NA\" />"
 [96] "\t<AnnBuilder:Item name=\"GRIF\" value=\"NA\" />"
 [97] "\t<AnnBuilder:Item name=\"SUMFUNC\" value=\"Arylamine N-acetyltransferase 2; N-
 [98] "\t<AnnBuilder:Item name=\"PMID\" value=\"8460648\" />"
 [99] "\t<AnnBuilder:Item name=\"PMID\" value=\"8102597\" />"
[100] "\t<AnnBuilder:Item name=\"PMID\" value=\"7915226\" />"
[101] "\t<AnnBuilder:Item name=\"PMID\" value=\"7773298\" />"
[102] "\t<AnnBuilder:Item name=\"PMID\" value=\"2734109\" />"
[103] "\t<AnnBuilder:Item name=\"PMID\" value=\"2340091\" />"
[104] "\t<AnnBuilder:Item name=\"PMID\" value=\"1968463\" />"
[105] "\t<AnnBuilder:Item name=\"PMID\" value=\"1676262\" />"
[106] "\t<AnnBuilder:Item name=\"PMID\" value=\"1381364\" />"
[107] "\t<AnnBuilder:Item name=\"PMID\" value=\"13\" />"
[108] "\t<AnnBuilder:Item name=\"CHR\" value=\"8\" />"
[109] "\t<AnnBuilder:Item name=\"GO\" value=\"GO:0004060\" type=\"E\"/>"
[110] "\t<AnnBuilder:Item name=\"CHRLOC\" value=\"-830797\" type=\"10\"/>"
[111] "</AnnBuilder:Entry>"
[112] "<AnnBuilder:Entry id=\"39368_at\" describ = \"PROBE\">"
[113] "\t<AnnBuilder:Item name=\"LOCUSID\" value=\"1084\" />"
[114] "\t<AnnBuilder:Item name=\"ACCNUM\" value=\"D90042\" />"
[115] "\t<AnnBuilder:Item name=\"UNIGENE\" value=\"NA\" />"
[116] "\t<AnnBuilder:Item name=\"GENENAME\" value=\"NA\" />"
```

```
[117] "\t<AnnBuilder:Item name=\"SYMBOL\" value=\"NA\" />"
[118] "\t<AnnBuilder:Item name=\"MAP\" value=\"NA\" />"
[119] "\t<AnnBuilder:Item name=\"GRIF\" value=\"NA\" />"
[120] "\t<AnnBuilder:Item name=\"SUMFUNC\" value=\"NA\" />"
[121] "\t<AnnBuilder:Item name=\"PMID\" value=\"NA\" />"
[122] "\t<AnnBuilder:Item name=\"CHR\" value=\"NA\" />"
[123] "NULL"
[124] "\t<AnnBuilder:Item name=\"CHRLOC\" value=\"+845627\" type=\"10\"/>"
[125] "</AnnBuilder:Entry>"
[126] "<AnnBuilder:Entry id=\"36512_at\" describ = \"PROBE\">"
[127] "\t<AnnBuilder:Item name=\"LOCUSID\" value=\"125\" />"
[128] "\t<AnnBuilder:Item name=\"ACCNUM\" value=\"D90278\" />"
[129] "\t<AnnBuilder:Item name=\"UNIGENE\" value=\"NA\" />"
[130] "\t<AnnBuilder:Item name=\"GENENAME\" value=\"NA\" />"
[131] "\t<AnnBuilder:Item name=\"SYMBOL\" value=\"NA\" />"
[132] "\t<AnnBuilder:Item name=\"MAP\" value=\"NA\" />"
[133] "\t<AnnBuilder:Item name=\"GRIF\" value=\"NA\" />"
[134] "\t<AnnBuilder:Item name=\"SUMFUNC\" value=\"NA\" />"
[135] "\t<AnnBuilder:Item name=\"PMID\" value=\"NA\" />"
[136] "\t<AnnBuilder:Item name=\"CHR\" value=\"NA\" />"
[137] "NULL"
[138] "\t<AnnBuilder:Item name=\"CHRLOC\" value=\"+118524\" type=\"10\"/>"
[139] "</AnnBuilder:Entry>"
[140] "<AnnBuilder:Entry id=\"32469_at\" describ = \"PROBE\">"
[141] "\t<AnnBuilder:Item name=\"LOCUSID\" value=\"63036\" />"
[142] "\t<AnnBuilder:Item name=\"ACCNUM\" value=\"AL031663\" />"
[143] "\t<AnnBuilder:Item name=\"UNIGENE\" value=\"NA\" />"
[144] "\t<AnnBuilder:Item name=\"GENENAME\" value=\"NA\" />"
[145] "\t<AnnBuilder:Item name=\"SYMBOL\" value=\"NA\" />"
[146] "\t<AnnBuilder:Item name=\"MAP\" value=\"NA\" />"
[147] "\t<AnnBuilder:Item name=\"GRIF\" value=\"NA\" />"
[148] "\t<AnnBuilder:Item name=\"SUMFUNC\" value=\"NA\" />"
[149] "\t<AnnBuilder:Item name=\"PMID\" value=\"NA\" />"
[150] "\t<AnnBuilder:Item name=\"CHR\" value=\"NA\" />"
[151] "NULL"
[152] "\t<AnnBuilder:Item name=\"CHRLOC\" value=\"+865077\" type=\"10\"/>"
[153] "</AnnBuilder:Entry>"
[154] "<AnnBuilder:Entry id=\"32468_f_at\" describ = \"PROBE\">"
[155] "\t<AnnBuilder:Item name=\"LOCUSID\" value=\"7051\" />"
[156] "\t<AnnBuilder:Item name=\"ACCNUM\" value=\"X03350\" />"
[157] "\t<AnnBuilder:Item name=\"UNIGENE\" value=\"NA\" />"
[158] "\t<AnnBuilder:Item name=\"GENENAME\" value=\"NA\" />"
```

```
[159] "\t<AnnBuilder:Item name=\"SYMBOL\" value=\"NA\" />"
[160] "\t<AnnBuilder:Item name=\"MAP\" value=\"NA\" />"
[161] "\t<AnnBuilder:Item name=\"GRIF\" value=\"NA\" />"
[162] "\t<AnnBuilder:Item name=\"SUMFUNC\" value=\"NA\" />"
[163] "\t<AnnBuilder:Item name=\"PMID\" value=\"NA\" />"
[164] "\t<AnnBuilder:Item name=\"CHR\" value=\"NA\" />"
[165] "NULL"
[166] "NULL"
[167] "</AnnBuilder:Entry>"
[168] "<AnnBuilder:Entry id=\"38912_at\" describ = \"PROBE\">"
[169] "\t<AnnBuilder:Item name=\"LOCUSID\" value=\"NA\" />"
[170] "\t<AnnBuilder:Item name=\"ACCNUM\" value=\" X68733\" />"
[171] "\t<AnnBuilder:Item name=\"UNIGENE\" value=\"NA\" />"
[172] "\t<AnnBuilder:Item name=\"GENENAME\" value=\"NA\" />"
[173] "\t<AnnBuilder:Item name=\"SYMBOL\" value=\"NA\" />"
[174] "\t<AnnBuilder:Item name=\"MAP\" value=\"NA\" />"
[175] "\t<AnnBuilder:Item name=\"GRIF\" value=\"NA\" />"
[176] "\t<AnnBuilder:Item name=\"SUMFUNC\" value=\"NA\" />"
[177] "\t<AnnBuilder:Item name=\"PMID\" value=\"NA\" />"
[178] "\t<AnnBuilder:Item name=\"CHR\" value=\"NA\" />"
[179] "NULL"
[180] "NULL</AnnBuilder:Entry>"
[181] "</AnnBuilder:Data>"
[182] "</AnnBuilder:Annotate>"
[183] ""
```

To generate an R data package containing the annotation data, we first save the data as R environment objects with key-value pairs. Each of the annotation element will be saved as a separate environment with probe ids as keys and the corresponding annotation as values (e. g. gene name) or vice versa. To so, we save each of the environment objects in a common environment and then dump the contents of the common environment to the destination using an existing R function texttttpackage.skeleton. We have to do this in a few steps as the mapping between probe ids and annotation elements may have different structure as previously mentioned.

```
> if (.Platform$OS.type != "windows") {
+     pkgName <- "test"
+     workEnv <- new.env(hash = TRUE, parent = NULL)
+     createEmptyDPkg("test", getwd(), force = TRUE)
+     for (i in getUniColNames()) {
+         env <- new.env(hash = TRUE, parent = NULL)
+         multiassign(as.vector(annotation[, "PROBE"]), sapply(annotation[,
+             i], splitEntry), env)
+         assign(paste(pkgName, i, sep = ""), env, workEnv)
+     }
+     for (i in intersect(colnames(annotation), getMultiColNames())) {
+         env <- new.env(hash = TRUE, parent = NULL)
+         multiassign(as.vector(annotation[, "PROBE"]), sapply(annotation[,
+             i], twoStepSplit), env)
+         assign(paste(pkgName, i, sep = ""), env, workEnv)
+     }
+     for (i in c("GO", "CHRLOC")) {
+         env <- new.env(hash = TRUE, parent = NULL)
+         multiassign(as.vector(annotation[, "PROBE"]), sapply(annotation[,
+             i], splitEntry), env)
+         assign(paste(pkgName, i, sep = ""), env, workEnv)
+     }
+ }
```

When we have all the annotation data we want saved as environment objects in a common environment, we can call textttpackage.skeleton to create a data package with data files stored in correct subdirectories in the package.

```
> if (.Platform$OS.type != "windows") {
+     for (i in ls(workEnv)) {
+         save(list = i, file = file.path(getwd(), "test", "data",
+             paste(i, ".rda", sep = "")), envir = workEnv)
+     }
+ }
```

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

```
> if (.Platform$OS.type != "windows") {
+     list.files(file.path(getwd(), "test"))
+ }
```

```
[1] "R"    "data" "man"
```

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.

```
> if (.Platform$OS.type != "windows") {
+     list.files(file.path(getwd(), "test", "data"))
+ }
```

```
 [1] "testACCNUM.rda"   "testCHR.rda"      "testCHRLOC.rda"   "testGENENAME.rda"
 [5] "testGO.rda"       "testGRIF.rda"     "testLOCUSID.rda"  "testMAP.rda"
 [9] "testPMID.rda"     "testSUMFUNC.rda"  "testSYMBOL.rda"   "testUNIGENE.rda"
```

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.

```
> if (.Platform$OS.type != "windows") {
+     writeAccessory(pkgName = "test", pkgPath = getwd(), organism = "human",
+         version = "1.1.1", author = c(name = "your name", address = "youremail@net.
+     getDPStats(baseF = "geneNMap", pkgName = "test", pkgPath = getwd(),
+         saveList = TRUE)
+     writeMan4QC("test", pkgPath = getwd())
+ }
```

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

```
> if (.Platform$OS.type != "windows") {
+     unlink(c(unified, XMLOut, "geneNMap", "test.xml", "testByNum.xml"))
+     unlink(file.path(getwd(), "test"), TRUE)
+ }
```