

# How to use AnnBuilder

Jianhua Zhang

June 2, 2004

## 1 Overview

AnnBuilder constructs annotation data packages or XML files for given sets of genes with known mappings either to GenBank accession numbers, UniGene identifiers, Image identifiers, or LocusLink identifiers. This vignette describes the process of building an annotation data package based on a sample file that maps probes to GenBank accession numbers. The process involves:

1. Map a given set of probes to LocusLink identifiers. There are two main components:
  - Obtain mappings from different sources. For a given set of genes, especially for genes on an Affymetrix chip, there are quite a few sources of existing mappings available from the web.
  - Unify mappings from different sources. Mapping information from different sources may agree or disagree. *AnnBuilder* resolves conflicts using a voting mechanism to obtain unified mappings between probes and LocusLink ids.
2. Based on the unified mappings, extract data from Locus Link and other sources such as Golden path, GO, KEGG.
3. Combine data into an R data package or XML files.

The capability of *AnnBuilder* is well beyond what has been described above. In theory, any set of genes can be annotated using AnnBuilder as long as they can be mapped to an id used by a public data repository. However, that will require some extend of programming using the existing functions. That part will be covered in another vignette (Advanced). In this vignette, the process of annotating a set of genes that are mapped to GenBank accession numbers using a single function will be discussed.

## 2 Getting Started

### 2.1 Requirements

AnnBuilder requires the support from the following items. The system will fail due to the lack of any of the requirements.

- R package XML is required to support the functions dealing with XML files. The package is available through <http://cran.r-project.org>.
- Perl is required to process the potentially rather large annotation source data files.

### 2.2 Function description

For a set of genes that are mapped to GenBank accession numbers (or UniGene identifiers, Image clone identifiers, LocusLink identifiers) a function named **ABPkgBuilder** can be used for building an annotation package. **ABPkgBuilder** takes the following arguments:

**baseName** A character string for the name of a file to be used as a base file to parse source data. The file should contain two columns with the first one being the target genes to be annotated and the other being the corresponding mappings to GenBank accession numbers, UniGene identifiers, Image clone identifiers, or LocusLink identifiers. The second column should have either a value or "NA".

**srcUrls** A named vector of character strings for the urls where source data files will be obtained. Valid sources are LocusLink, UniGene, Golden Path, Gene Ontology, and KEGG. The names for the character strings should be LL, UG, GP, GO, and KEGG, respectively. LL and UG are required. For windows users, the values should be unzipped files downloaded from the sources. A function call `getSrcUrl("all", "human")` will return the urls needed for building a package for human. Other valid organism names include mouse and rat.

**baseMapType** A character string to indicate whether target genes in **baseName** are mapped to GenBank accession numbers (gb), UniGene identifiers (ug), Image clone identifiers (image), or LocusLink identifiers (ll).

**otherSrc** A named vector of character strings for the names of files that contain mappings between target genes in **baseName** and LocusLink identifiers that will be unified to get more reliable mappings.

**pkgName** A character string for the name of the data package to be built (e. g. hgu95av2, rgu34a).

**pkgPath** A character string for the full path of an existing directory where the package to be built will be stored.

**organism** A character string for the name of the organism of concern (now can only be "human", "mouse", or "rat").

**version** A character string for the version number of the data package or XML files to be built.

**makeXML** A boolean to indicate whether an XML documents containing annotation data will be generated or not.

**author** A list of character strings with an author element for the name of the maintainer of the data package and a maintainer element for the email address of the maintainer.

What we need to do is to assign correct values to the above arguments and then call `ABPkgBuilder` with these arguments.

## 2.3 Datasets

We have placed some mini data sets in the `data` directory of *AnnBuilder* to demonstrate how to use `ABPkgBuilder`. One of them is `thgu95a` that contains Affymetrix probe ids and their mappings to GenBank accession numbers. The file looks like:

```
> library(AnnBuilder)
```

```
Welcome to Bioconductor
```

```
Vignettes contain introductory material. To view,  
simply type: openVignette()  
For details on reading vignettes, see  
the openVignette help page.
```

```
> read.table(file.path(.path.package("AnnBuilder"), "data", "thgu95a"),  
+           sep = "\t", header = FALSE, as.is = TRUE)
```

	V1	V2
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

Now we set the file as the base file (`baseName`) and indicate that the mappings for the base file is GenBank accession numbers.

```
> myBase <- file.path(.path.package("AnnBuilder"), "data", "thgu95a")
> myBaseType <- "gb"
```

The data sources that can be used for annotation are abundant. We focus on following public data sources:

**LocusLink** The data `ftp://ftp.ncbi.nih.gov/refseq/LocusLink/LL\protect\T1\textunderscoretmpl.gz` will be used to map genes to LocusLink identifiers and also to annotate genes after the unified mappings have been obtained.

**UniGene** The data contained at `ftp://ftp.ncbi.nih.gov/repository/UniGene/` will be used to obtain mappings between genes and LocusLink identifiers. The exact data that will be used depend on the organism.

**Golden Path** The data (`refLink.txt` and `refGene.txt`) at `http://www.genome.ucsc.edu/goldenPath/14nov2002/database` will be used to obtain the chromosomal location and orientation data for genes. The part 14nov2002 will be something else for a different organism or when there is new built for the data sets.

**Gene Ontology** The data `http://www.godatabase.org/dev/database/archive/2003-03-01/go_200303-termdb.xml.gz` will be used to obtain gene ontology information. The last part of the url changes with builds.

**KEGG** Some data at `ftp://ftp.genome.ad.jp/pub/kegg/pathways` will be used to extract the pathway and enzyme information. Quite a few individual files will be used and the system has a way of locating them with information available at the site by the url.

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

We may assign the urls to `srcUrls`.

```
> mySrcUrls <- c(LL = "ftp://ftp.ncbi.nih.gov/refseq/LocusLink/LL_tmpl.gz",
+   UG = "ftp://ftp.ncbi.nih.gov/repository/UniGene/Hs.data.gz",
+   GP = "http://www.genome.ucsc.edu/goldenPath/14nov2002/database/",
+   GO = "http://www.godatabase.org/dev/database/archive/2003-03-01/go_200303-termdb.xml.gz",
+   KEGG = "ftp://ftp.genome.ad.jp/pub/kegg/pathways")
```

However, *AnnBuilder* has the url information stored and retrieves them when the function `getSrcUrl` is called.

```

> mySrcUrls <- getSrcUrl("all", organism = "human")
> mySrcUrls

LL
"ftp://ftp.ncbi.nih.gov/refseq/LocusLink/LL_tmpl.gz"
GP
"http://www.genome.ucsc.edu/goldenPath/hg17/database/"
UG
"ftp://ftp.ncbi.nih.gov/repository/UniGene/Hs.data.gz"
GO
"http://www.godatabase.org/dev/database/archive/2004-05-01/go_200405-termdb.xml.gz"
KEGG
"ftp://ftp.genome.ad.jp/pub/kegg/pathways"
YG
"http://www.yeastgenome.org/DownloadContents.shtml"
HG
"ftp://ftp.ncbi.nih.gov/pub/HomoloGene/old/hmlg.ftp"

```

Other valid organism names include mouse and rat.

In the vignette, we will use truncated versions of some of the files to reduce the length of time required to process the source data. The truncated files are stored at the Bioconductor web site.

```

> mySrcUrls <- c(LL = "http://www.bioconductor.org/datafiles/wwwsources/Tll_tmpl.gz",
+   UG = "http://www.bioconductor.org/datafiles/wwwsources/Ths.data.gz",
+   GO = "http://www.bioconductor.org/datafiles/wwwsources/Tgo.xml")

```

If there is not other sources of mappings between the target genes and LocusLink identifiers available, the mappings provided by LocusLink and UniGene will be unified. However, as an example, let us assume that we have the mappings from two other sources and we would like to use them as other sources to obtain the unified mappings. The two sources are also stored in the `data` directory of *AnnBuilder*.

```

> read.table(file.path(.path.package("AnnBuilder"), "data", "srca"),
+   sep = "\t", header = FALSE, as.is = TRUE)

```

	V1	V2
1	32468_f_at	NA
2	32469_at	2
3	32481_at	NA
4	33825_at	9
5	35730_at	1576
6	36512_at	NA

```

7  38912_at    10
8  38936_at    NA
9  39368_at    NA

> read.table(file.path(.path.package("AnnBuilder"), "data", "srcb"),
+           sep = "\t", header = FALSE, as.is = TRUE)

      V1    V2
1 32468_f_at  NA
2  32469_at  NA
3  32481_at 7051
4  33825_at  NA
5  35730_at  NA
6  36512_at 1084
7  38912_at  NA
8  38936_at  NA
9  39368_at   89

```

We assign the two files to `otherSrc`

```

> myOtherSrc <- c(srcone = file.path(.path.package("AnnBuilder"),
+   "data", "srca"), srctwo = file.path(.path.package("AnnBuilder"),
+   "data", "srcb"))

```

The other arguments needed are pretty straight forward and will not be elaborated.

## 2.4 Build annoation

To build an annotation data package, we only have to call `ABPkgBuilder` with correct argument values. However, the code below (and hereafter) is turned off under windows as human intervention is required under the system. Copying the code chunk and pasting into an R session under windows should work.

```

> myDir <- tempdir()
> if (.Platform$OS.type != "windows") {
+   ABPkgBuilder(baseName = myBase, srcUrls = mySrcUrls, baseMapType = myBaseType,
+   otherSrc = myOtherSrc, pkgName = "myPkg", pkgPath = myDir,
+   organism = "human", version = "1.1.0", makeXML = TRUE,
+   author = list(author = "myname", maintainer = "myname@myemail.com"),
+   fromWeb = TRUE)
+ }

```

```
[1] "It may take me a while to process the data. Be patient!"
```

We will have a data package named "myPkg" in the directory defined by myDir. The data package has a data, man, and R subdirectory each with some files.

```
> if (.Platform$OS.type != "windows") {  
+   list.files(file.path(myDir, "myPkg"))  
+   list.files(file.path(myDir, "myPkg", "data"))  
+   list.files(file.path(myDir, "myPkg", "man"))  
+   list.files(file.path(myDir, "myPkg", "R"))  
+ }
```

```
[1] "myPkg.R" "zzz.R"
```

The created data package can be installed the same way as a regular R package. Notice that XML files have also been created as makeXML is set to TRUE.

Now, let us clean up

```
> if (.Platform$OS.type != "windows") {  
+   unlink(file.path(myDir, "myPkg"), TRUE)  
+   unlink(c("geneNMap", "srcone", "srctwo", file.path(myDir,  
+     "myPkg.xml"), file.path(myDir, "myPkgByNum.xml")))  
+ }
```

### 3 Further note

Function `ABPkgBuilder` works only if the data files are of the correct format (e.g. delimiter separated two column text files) and the urls for the source data and information on their builds remain unchanged. When changes to the urls occur, the function will fail and users may not have much power of control because `ABPkgBuilder` makes assumptions and then calls different functions based on the assumptions. Another vignette `AnnBuilder` shows the details of using the functions `ABPkgBuilder` based on but are available in *AnnBuilder* to build data packages. More coding is involved there but users will have much greater control over the building process and avoiding system failures as that may occur when using `ABPkgBuilder`. Users are encouraged to read that vignette when become comfortable with `ABPkgBuilder`.