Creating probe packages

Wolfgang Huber and Robert Gentleman

April 29, 2014

Contents

1 Overview

This document describes how to create a Bioconductor *probe* package from the reporter sequence information of a particular chip. Probe packages are a convenient way for distributing and storing the probe sequences and related information.

First, let us load the *AnnotationForge* package.

```
library("AnnotationForge")
```

```
## Loading required package:  BiocGenerics
## Loading required package:  parallel
##
## Attaching package:  'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ, clusterExport,
##    clusterMap, parApply, parCapply, parLapply, parLapplyLB, parRapply,
##    parSapply, parSapplyLB
##
## The following object is masked from 'package:stats':
##
##    xtabs
##
## The following objects are masked from 'package:base':
##
##    anyDuplicated, append, as.data.frame, as.vector, cbind, colnames, do.call,
##    duplicated, eval, evalq, Filter, Find, get, intersect, is.unsorted, lapply,
##    Map, mapply, match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
```

```
##    Position, rank, rbind, Reduce, rep.int, rownames, sapply, setdiff, sort,
##    table, tapply, union, unique, unlist
##
## Loading required package:  Biobase
## Welcome to Bioconductor
##
##    Vignettes contain introductory material; view with 'browseVignettes()'.  To
##    cite Bioconductor, see 'citation("Biobase")', and for packages
##    'citation("pkgname")'.
##
## Loading required package:  AnnotationDbi
## Loading required package:  GenomeInfoDb
## Loading required package:  org.Hs.eg.db
## Loading required package:  DBI
```

1.1 For Affymetrix genechips

In this section we see how a probe package can be created for Affymetrix genechips from the tabulator-separated sequence files that can be obtained from the vendor (at http://www.affymetrix.com/support). As an example, the file `HG-U95Av2_probe_tab.gz` is provided in the `extdata` subdirectory of the *AnnotationForge* package.

```
filename <- system.file("extdata", "HG-U95Av2_probe_tab.gz", package = "AnnotationForge")
outdir <- tempdir()
me <- "Wolfgang Huber <w.huber@dkfz.de>"
species <- "Homo_sapiens"
makeProbePackage("HG-U95Av2", datafile = gzfile(filename, open = "r"), outdir = outdir, maintaine
    species = species, version = "0.0.1", force = TRUE)
```

```
## Importing the data.
```

```
## Loading required package:  affy
```

```
## Creating package in /tmp/Rtmp7RWT4h/hgu95av2probe
## Writing the data.
## Checking the package.
## *** WARNINGS ***
##  * checking data for ASCII and uncompressed saves ... WARNING WARNING: There was 1 warning.Bui
## [1] "hgu95av2probe"
```

```
dir(outdir)
```

```
## [1] "hgu95av2probe"              "hgu95av2probe_0.0.1.tar.gz"
```

1.2 For other chiptypes

To deal with different file formats and additional types of probe annotation data from public or in-house databases, the function `makeProbePackage` offers a great deal of flexibility. The user can specify her own

import function through the argument `importfun`. By default, its value is `getProbeDataAffy`, a function that reads tabular Affymetrix genechip sequence files. Import functions for other types of arrays can be adapted from this prototype.

The help pages and R code contained in the produced packages are derived from a template directory that obeys the usual R package conventions [1]. A prototype for such a directory is provided within the package *AnnotationForge*. To facilitate the automated production of large numbers of similar packages, we provide a text substitution mechanism similar to the one used in the GNU `configure` system.

The input parameters of an import function are

- a character string naming the array type
- the input data files
- a character string with the directory name of a package template directory, containing at least a file `DESCRIPTION`, a directory `man` with a file @PKGNAME@.Rd, a directory `data`, and possible other directories and files, conforming to the usual R package conventions.
- ... any sort of further parameters, as necessary.

The output of an import function is a named list with elements

- `pkgname`: a character string with the package name.
- `dataEnv`: an environment containing an arbitrary number of data objects; these make the core of the package. Among these, there should be a data frame whose name is the value of `pkgname` with a column `sequence`. This data frame may have other columns such as the x- and y-position of the probes on the array, identifiers linking a probe to genomic databases, or its length and relative position within the gene it represents. The objects in the environment will be saved as individual `.rda` files of the same name into the data directory of the produced package. Documentation needs to be provided for the columns of the data frame, as well as for the other objects.
- `symVals`: a named list, containing the symbol-value substitutions that are used in the text processing. It must at least contain the elements
  - `ARRAYTYPE`: name of the array
  - `DATASOURCE`: a textual description of how the data were obtained. It should contain the URL, or the name of the company / person.

For more details, please refer to the help files for the functions `makeProbePackage` and `getProbeDataAffy`. For an example, refer to the source code of `getProbeDataAffy`.

References

[1] R Foundation (1999). *Writing R Extensions*. http://www.r-project.org.