Handling probe sequence information: the package matchprobes

Wolfgang Huber and Robert Gentleman

October 3, 2006

Contents

1 Overview

This package allows to calculate with the sequences and related information of the probes on a microarray. The sequences themselves are stored in separate packages, one for each type of microarray. For some commonly used types, the packages can be downloaded from http://www.bioconductor.org. You can also create such packages yourself. This is described in section 4.

```
> library("matchprobes")
> library("affy")
> library("hgu95av2cdf")
> library("hgu95av2probe")
> library("hu6800cdf")
> library("hu6800probe")
```

2 Using probe packages

Help for the probe sequence data packages can be accessed through

```
> ? hgu95av2probe
```

2.1 Functions

Complementary sequence

```
> example(complementSeq)
```

```
cmplmS> seq <- c("AAACT", "GGGTT")

cmplmS> complementSeq(seq)
[1] "TTTGA" "CCCAA"

cmplmS> seq <- c("CGACTGAGACCAAGACCTACAACAG", "CCCGCATCATCTTTCCTGTGCTCTT")

cmplmS> complementSeq(seq, start = 13, stop = 13)
[1] "CGACTGAGACCATGACCTACAACAG" "CCCGCATCATCTATCCTGTGCTCTT"
```

Reverse sequence

```
> example(reverseSeq)
```

```
rvrsSq> w <- c("hey there", "you silly fool")

rvrsSq> reverseSeq(w)
[1] "ereht yeh"      "loof yllis uoy"

rvrsSq> w <- "able was I ere I saw Elba"
```

```
rvrsSq> reverseSeq(w)
[1] "ablE was I ere I saw elba"
```

Matching probes

```
> pm <- hgu95av2probe$sequence
> query <- hgu95av2probe$sequence[21:23]
> m <- matchprobes(query, pm)
> unlist(m)
```

```
match1 match2 match3
    21     22     23
```

2.2 Mismatch sequences for Affymetrix GeneChips

The mismatch sequences are not explicitely stored in the probe packages. They are implied by the match sequences, by flipping the middle base. This can be done with the `complementSeq`.

```
> mm <- complementSeq(hgu95av2probe$sequence, start = 13, stop = 13)
> cat(pm[1], mm[1], sep = "\n")
```

```
TCTCCTTTGCTGAGGCCTCCAGCTT
TCTCCTTTGCTGTGGCCTCCAGCTT
```

For Affymetrix GeneChips the length of the probe sequences is 25, and since we start counting at 1, the middle position is 13.

The function `matchprobes` also responds to this type of implied mismatch sequences, by multipying the matching indices with -1:

```
> m <- matchprobes(query, c(pm, mm))
> unlist(m)
```

```
 match1  match2  match3  match4  match5  match6
     21 -199105      22 -199106      23 -199107
```

Note(wh): This may need to be set as an option, rather than as a standard behavior - I don't particularly like that the position 13 is hardcoded into `matchprobes`. Best may in fact be to drop the mismatch matching altogether, since we could simply use `complementSeq(query, start=13, stop=13)` on the query string(s) and then call `matchprobes` with standard exact matching.

2.3 Base content

The base content of the probes on an array can be obtained through the function `basecontent`.

```
> bcpm <- basecontent(hgu95av2probe$sequence)
> as.matrix(bcpm[1:5, ])
```

```
 A T C  G
 2 9 9  5
 6 3 9  7
 5 4 9  7
 5 6 8  6
 5 7 3 10
```

```
> bcmm <- basecontent(complementSeq(hgu95av2probe$sequence, start = 13,
+     stop = 13))
> as.matrix(bcmm[1:5, ])
```

```
 A  T C G
 1 10 9 5
 7  2 9 7
 4  5 9 7
 5  6 7 7
 5  7 4 9
```

3 CombineAffyBatch

Load the data.

```
> f1 <- system.file("extdata", "118T1.cel.gz", package = "matchprobes")
> f2 <- system.file("extdata", "CL2001032020AA.cel.gz", package = "matchprobes")
> pd1 <- new("phenoData", pData = data.frame(id = "pi"), varLabels = list("phenovar"))
> pd2 <- new("phenoData", pData = data.frame(id = "bh"), varLabels = list("phenovar"))
> x1 <- read.affybatch(filenames = f1, compress = TRUE, phenoData = pd1)
> x2 <- read.affybatch(filenames = f2, compress = TRUE, phenoData = pd2)
```

`f1` and `f2` contain the filenames of the CEL files. In practice, they will be vectors with many filenames per array type, here, for demonstration, we only use one single CEL file per array type, which are provided in the subdirectory `extdata` of the package.

Combine the data. For this to work it is required that the packages *hu6800probe* and *hgu95av2probe* are installed.

```
> res <- combineAffyBatch(list(x1, x2), c("hu6800probe", "hgu95av2probe"),
+     newcdf = "comb")
```

```
package:hu6800probe        hu6800probe
package:hgu95av2probe        hgu95av2probe
34428 unique probes in common
```

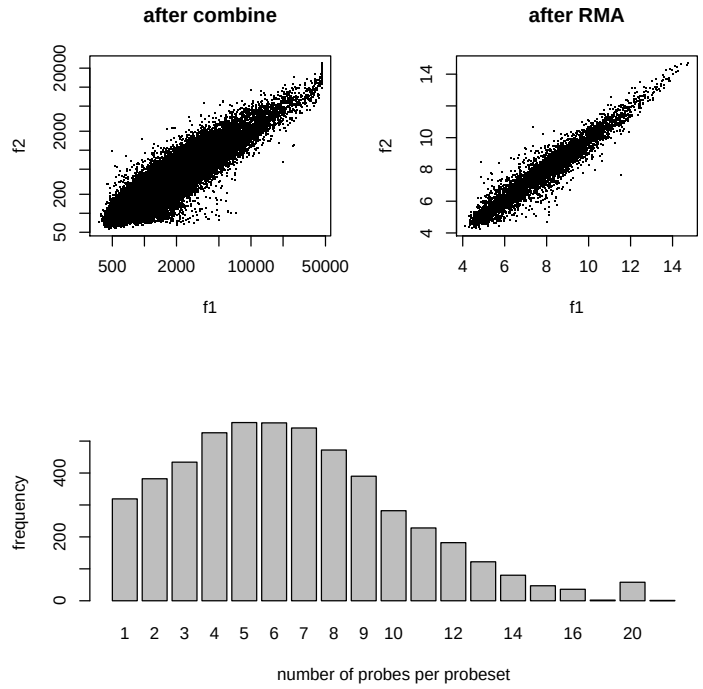
The function returns a list `res` with two elements: an `AffyBatch data` and a `CDF enviroment newcdf`.

```
> comb <- res$cdf
> z <- rma(res$dat)
```

```
Background correcting
Normalizing
Calculating Expression
```

View the results.

```
> layout(rbind(1:2, c(3, 3)))
> plot(exprs(res$dat), main = "after combine", pch = ".", xlab = "f1",
+     ylab = "f2", log = "xy")
> plot(exprs(z), main = "after RMA", pch = ".", xlab = "f1", ylab = "f2")
> prs <- mget(ls(comb), comb, ifnotfound = NA)
> nrprobes <- sapply(prs, function(x) nrow(x))
> barplot(table(nrprobes), xlab = "number of probes per probeset",
+     ylab = "frequency")
```



4 Creating probe packages

Probe packages are a convenient way for distributing and storing the probe sequences and related information.

4.1 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 *matchprobes* package.

```
> filename <- system.file("extdata", "HG-U95Av2_probe_tab.gz",
+     package = "matchprobes")
> outdir <- tempdir()
> me <- "Wolfgang Huber <w.huber@dkfz.de>"
> species <- "Homo_sapiens"
```

```
> makeProbePackage("HG-U95Av2", datafile = gzfile(filename, open = "r"),
+     outdir = outdir, maintainer = me, species = species, version = "0.0.1",
+     force = TRUE)

Importing the data.
Creating package in D:\biocbld\1.9d\tmpdir\RtmpBz0Eqb/hgu95av2probe
Writing the data.
Checking the package.
Building the package.
[1] "hgu95av2probe"

> dir(outdir)

[1] "hgu95av2probe"              "hgu95av2probe_0.0.1.tar.gz"
[3] "Rf4df25e14"
```

4.2 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 *matchprobes*. 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`.