CodelinkSet

Diego Diez

April 26, 2013

1 Introduction

The `CodelinkSet` is an extension of the `ExpressionSet` class, that improves de user experience in terms of integration with other Bioconductor packages. Because this class is derived from `ExpressionSet`, most functions that work on `ExpressionSet` will work in `CodelinkSet` objects. Futhermore, extension of existing methods and functions is easier.

2 Loading of data

There is a new function called `readCodelinkSet` that will load the data in the old format and convert it into the `CodelinkSet` format. This function will replace in the future `readCodelink`. It is possible to include `phenoData` and `featureData` when calling `readCodelinkSet`. In addition, the correct annotation package is guessed from the first file and assigned to the `annotation` slot.

Instead of giving the list of files to read, it is possible to give it the location of a targets file. This will be preferred and the information in the targets file will be used to fill de phenoData slot.

```
> cset <- readCodelinkSet("targets.txt")
>
```

2.1 Feature data

The `CodelinkSet` stores intensity data in the exprs slot, much the same as Affymetrix derived data. It also contains a slot background, to accomodate background intensities. The featureData object will contain further information like ProbeName, ProbeType, meanSNR (computed when loading data), Row and Col locations in the chip, etc. The FeatureID is used to name rows, although the ProbeName is the useful information when accesing annotation data.

2.2 Annotations

The `CodelinkSet` class supports by default new style (*AnnotationDbi* based) annotation packages, i.e. all annotations will contain the .db subfix. To change to the old style you can assign the annotation package by hand:

```
> library(codelink)
> data(codelink.exprset)
> annotation(codelink.exprset)

[1] "rwgcod"

> annotation(codelink.exprset) <- "rwgcod"
> annotation(codelink.exprset)

[1] "rwgcod"

>
```

NOTE: Old style annotation packages will be deprecated for BioC 2.3

3 The User Interface

The user interface has changed to be more consistent and easier to use. All user method for `CodelinkSet` object have the prefix "cod" followed by the corresponding method name.

3.1 Data accession

`getInt` `getBkd` `getSNR`
`featureNames` `probeNames` `sampleNames`

3.2 Preprocessing

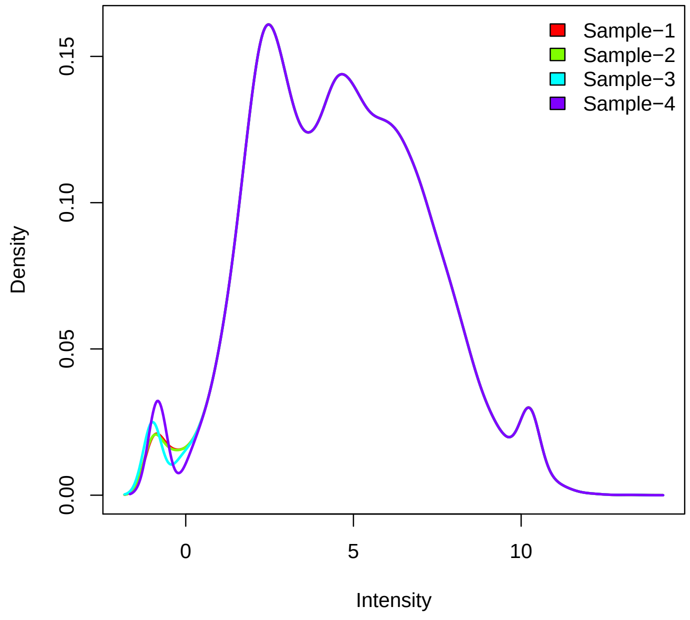
`codCorrect` is used for background correction and `codNormalize` for normalization. A convenience function called `codPreprocess` is also available to combine in one step background correction and normalization. The devault values found in the older functions are preserved, i.e. *half* for `codCorrect` and *quantile* `codNormalize`

```
> cset <- codPreprocess(codelink.exprset)
>
```

3.3 Plots

`codPlotMA`, `codPlotDensities`, `codPlotCorrelation` and `codPlotImage` are the corresponding new functions. A convinience funcion `codPlot` is provided and the type of plot is choose with the parameter *what*.

```
> codPlot(cset, what = "density")
> codPlot(cset, what = "ma")
> codPlot(cset, what = "image")
>
```



4 Analysis with limma

Analysis using *limma* is straightforward. The design matrix can be specified using the `phenoData` information. In the fit step, the `CodelinkSet` object can be passed directly to `lmFit`. Making life easier for users of the Codelink platform.

```
> # assign example experimental conditions:
> cset$Treatment = c("C", "C", "T", "T")
> # create and fit model:
> design <- model.matrix(~ -1 + factor(cset$Treatment))
> fit <- lmFit(cset, design)
>
```

5 Exporting data to a file

You may want to export your normalized data to use it with other analysis tools. The function `writeCodelink` allows to do that. By default the index, probe names, accession number, entrez gene identifiers, intensity and SNR values are output. If flag = TRUE, then the flags are also output.

```
> writeCodelink(codelink.exprset, file = "intensities.txt")
```

6 Creating `CodelinkSetUnique` objects

`CodelinkSet` objects use the feature id of each probe to obtain unique identifiers. This feature id is related to the physical location of the probe in the array. This means that when we use `featureNames` in a `CodelinkSet` object we get these feature ids. To get the probe ids we need to use `probeNames` instead. This can be inconvenient when we want to use our data with other packages that use `featureNames` to obtain the probe names and feed this ids to the annotation package. Because the annotation packages use probe names, the probes wouldn't be found. To solve this problem a `CodelinkSet` with unique probe names as `featureNames` is needed. To construct such an object we can use the function `averageProbes`, that takes a `CodelinkSet` object and computes the `mean` of the intensities for replicated probes. The results are stored in a `CodelinkSetUnique` object.

```
> foo.avg <- averageProbes(codelink.exprset)
```

This computation takes a lot of time. It is possible to use a parallelized version (using the package *multicore*) by making 'parallel=TRUE' in `aver-ageProbes`. See the help for `averageProbes` for more details on this approach. The package *multicore* needs to be loaded in advance by typing `library(multicore)`.

7 Session Info

```
> sessionInfo()

R version 3.0.0 (2013-04-03)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
```

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

other attached packages:
[1] codelink_1.28.1    limma_3.16.2       Biobase_2.20.0     BiocGenerics_0.6.0

loaded via a namespace (and not attached):
[1] AnnotationDbi_1.22.3 DBI_0.2-6            IRanges_1.18.0
[4] RSQLite_0.11.3       XML_3.96-1.1         annotate_1.38.0
[7] stats4_3.0.0         tools_3.0.0          xtable_1.7-1
```