1 Introduction

This package contains an implementation of the CGHnormaliter strategy for normalization of two-channel array Comparative Genomic Hybridization (aCGH) data displaying many copy number imbalances. The key idea of our method is that temporary exclusion of aberrations from the aCGH data allows for a more appropriate calculation of the LOWESS regression curve. As a result, after normalization, the log$_2$ intensity ratios of the normals will generally be closer to zero and better reflect the biological reality. We coined this normalization strategy ‘local-LOWESS’ since only a subset of the log$_2$ ratios is considered in the LOWESS regression.

The strategy can be summarized as follows (see Figure 1). Initially the log$_2$ intensity ratios are segmented using DNAcopy [5]. The segmented data are then given as input to a calling tool named CGHcall [2] to discriminate the normals from gains and losses. These normals are subsequently used for normalization based on LOWESS. These steps are then iterated to refine the normalization. For more detailed information we refer to the publications of the method [4, 3].
2 Data format

The input should be either a data.frame or the file name of a tabseparated text file (text files must contain a header). The first four columns should describe the clone and its position on the genome:

1. ID : The unique identifiers of array elements.
2. Chromosome : Chromosome number of each array element.
3. Start : Chromosomal start position in bp of each array element.
4. End : Chromosomal end position in bp of each array element.

The start and end positions must be numeric. The next columns hold the actual data. For each sample in the experiment, there must be two adjacent columns with the test and reference intensities, respectively. All entries must be delimited by tabs, and missing entries must be denoted with NA or by an empty value. Below, an example is given of a correctly formatted data file or data.frame containing measurements on 7 clones in 2 samples.

<table>
<thead>
<tr>
<th>ID</th>
<th>Chromosome</th>
<th>Start</th>
<th>End</th>
<th>Case1.test</th>
<th>Case1.ref</th>
<th>Case2.test</th>
<th>Case2.ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP11-34P13</td>
<td>1</td>
<td>1</td>
<td>254749</td>
<td>279</td>
<td>294</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>RP11-379K15</td>
<td>1</td>
<td>96421</td>
<td>244136</td>
<td>1815</td>
<td>2269</td>
<td>2793</td>
<td>3996</td>
</tr>
<tr>
<td>RP11-776018</td>
<td>1</td>
<td>357737</td>
<td>465038</td>
<td>367</td>
<td>349</td>
<td>429</td>
<td>362</td>
</tr>
<tr>
<td>RP11-45C18</td>
<td>1</td>
<td>579118</td>
<td>696613</td>
<td>786</td>
<td>734</td>
<td>900</td>
<td>735</td>
</tr>
<tr>
<td>RP11-242B5</td>
<td>1</td>
<td>606617</td>
<td>711982</td>
<td>2955</td>
<td>4158</td>
<td>4478</td>
<td>5229</td>
</tr>
<tr>
<td>RP13-586C17</td>
<td>1</td>
<td>619355</td>
<td>783174</td>
<td>NA</td>
<td>NA</td>
<td>823</td>
<td>841</td>
</tr>
<tr>
<td>RP11-414L23</td>
<td>1</td>
<td>658751</td>
<td>846904</td>
<td>630</td>
<td>937</td>
<td>959</td>
<td>744</td>
</tr>
</tbody>
</table>

3 Example

First, we load the example acute lymphoblastic leukemia dataset [1] which comes with the CGHnormaliter package:

```r
> library(CGHnormaliter)
> data(Leukemia)
```

Next, we run the CGHnormaliter routine on the first four chromosomes of the Leukemia data:

```r
> result <- CGHnormaliter(Leukemia, nchrom=4, cellularity=0.9)
```

CGHnormaliter -- Running an initial segmentation and calling
Start data segmentation ..
Analyzing: Sample.1
Analyzing: Sample.2
Analyzing: Sample.3
Start data calling ..
Mean normalization shift per sample:

- Case1.test_Case1.ref : 0.1053031
- Case2.test_Case2.ref : 0.07000189
- Case3.test_Case3.ref : 0.128865

Start data segmentation..
Analyzing: Sample.1
Analyzing: Sample.2
Analyzing: Sample.3
Start data calling..

Mean normalization shift per sample:

- Case1.test_Case1.ref : 0.00856964
- Case2.test_Case2.ref : 0.01487608
- Case3.test_Case3.ref : 0.01226434

Start data segmentation..
Analyzing: Sample.1
Analyzing: Sample.2
Analyzing: Sample.3
Start data calling..

Mean normalization shift per sample:

- Case1.test_Case1.ref : 0.001326081
- Case2.test_Case2.ref : 0.001739907
- Case3.test_Case3.ref : 0.001817676

CGHnormaliter -- Reached convergence. Running a final segmentation and calling...

Start data segmentation..
Analyzing: Sample.1
Analyzing: Sample.2
Analyzing: Sample.3
Start data calling..

Writing MA-plots to file: MAplot.pdf

CGHnormaliter -- FINISHED

To enable a visual assessment of the bias reduction, MA-plots are (by default) automatically generated before and after normalization of each sample. These plots are stored into a PDF, usually named MAplot.pdf (the exact file name is supplied at the end of each CGHnormaliter run). See Figure 2 for such MA-plots of the second Leukemia sample.

Now, several fields of the result object can be accessed, for example:

```r
> normalized.data <- copynumber(result)  # log2 ratios
> segmented.data <- segmented(result)
> called.data <- calls(result)
```
Figure 2: MA-plot for the second leukemia sample before and after normalization. Note that the normalization is based on the normals only, represented by the black dots.

Plotting the normalized log$_2$ ratios in a density plot provides another means (besides MA plots) to inspect whether or not the centralization has been successful. Figure 3 shows such a density plot for sample 2:

```r
> plot(density(normalized.data[, 2]), col=1, xlab="log2 ratio", + main="Density plot")
> abline(v=0, lty=2)
```

The results, including segments and calls, can be visualized using the `plot` function. In Figure 4 the results of sample 2 are plotted in full resolution:

```r
> plot(result[,2], ylim=c(-2,2), dotres=1)
```

Plotting sample Case2.test_Case2.ref
Figure 3: Density plot after CGHnormalie normalization for the second leukemia sample. The data are adequately centralized around peak at the left, which corresponds to the normals. The peak at the right corresponds to the gains.

Figure 4: Results of the CGHnormalie normalization for the second leukemia sample.
Finally, the package provides the function `CGHnormaliter.write.table` to save the normalized data into a tabdelimited plain text file:

```r
> CGHnormaliter.write.table(result)
```

Saving normalized log2 ratios to file: normalized.txt

The segmented and called data from the `result` object can be saved to file as well using this function:

```r
> CGHnormaliter.write.table(result, data.type="segmented")
```

Saving segmented log2 ratios to file: segmented.txt

```r
> CGHnormaliter.write.table(result, data.type="called")
```

Saving calls to file: called.txt

References


