CGHnormaliter Package (Version 1.30.0)

Bart P.P. van Houte, Thomas W. Binsl, Hannes Hettling

April 24, 2017

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].

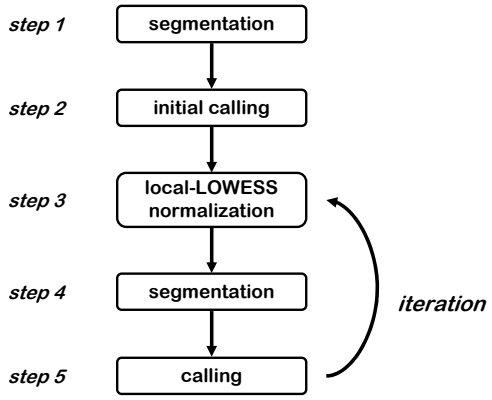

Figure 1: Overview of the CGHnormaliter method.

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.

ID	Chromosome	Start	End	Case1.test	Case1.ref	Case2.test	Case2.ref
RP11-34P13	1	1	254479	279	294	NA	NA
RP11-379K15	1	95421	244136	1815	2269	2793	3996
RP11-776O18	1	357737	465038	387	349	429	362
RP11-45C18	1	579118	696613	786	734	900	735
RP11-242B5	1	606617	711982	2955	4158	4478	5229
RP13-586C17	1	619355	783174	NA	NA	823	841
RP11-414L23	1	658751	846904	630	937	959	744

3 Example

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

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

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

```
> 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 ..
```

```
CGHnormaliter -- Iteration # 1
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 ..
CGHnormaliter -- Iteration # 2
Mean normalization shift per sample:
   Case1.test_Case1.ref : 0.03114221
   Case2.test_Case2.ref : 0.01881228
   Case3.test_Case3.ref : 0.03098722
Start data segmentation ..
Analyzing: Sample.1
Analyzing: Sample.2
Analyzing: Sample.3
Start data calling ..
CGHnormaliter -- Iteration # 3
Mean normalization shift per sample:
   Case1.test_Case1.ref : 0.02481902
   Case2.test_Case2.ref : 0.01685955
   Case3.test_Case3.ref : 0.02337867
Start data segmentation ..
Analyzing: Sample.1
Analyzing: Sample.2
Analyzing: Sample.3
Start data calling ..
CGHnormaliter -- Iteration # 4
Mean normalization shift per sample:
   Case1.test_Case1.ref : 0.01451821
   Case2.test_Case2.ref : 0.01644914
   Case3.test_Case3.ref : 0.01972368
Start data segmentation ..
Analyzing: Sample.1
Analyzing: Sample.2
Analyzing: Sample.3
Start data calling ..
CGHnormaliter -- Iteration # 5
Mean normalization shift per sample:
   Case1.test_Case1.ref : 0.001101871
   Case2.test_Case2.ref : 0.001576463
   Case3.test_Case3.ref : 0.001713899
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.

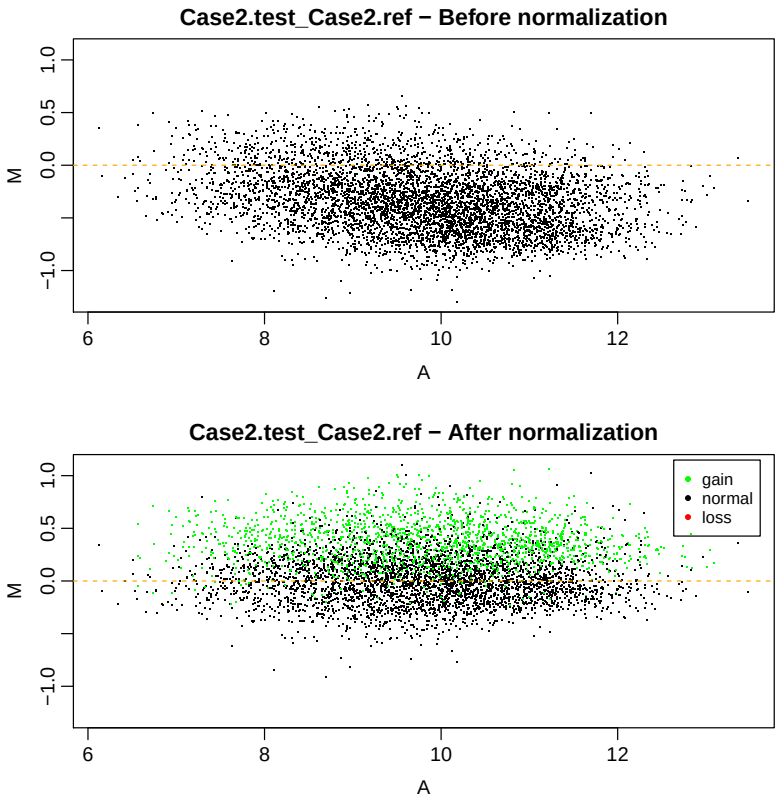

Figure 2: MA-plot for the second leukemia sample before and after normalization. Note that the normalization is based on the normals only, represented by the black dots.

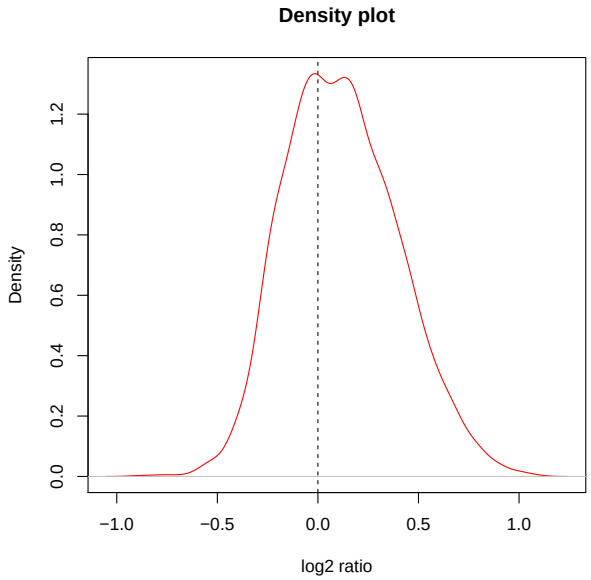

Figure 3: Density plot after CGHnormaliter normalization for the second leukemia sample. The data are adequately centralized around peak at the left, which corresponds to the normals. The peak at the right corresponds to the gains.

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

```
> normalized.data <- copynumber(result)  # log2 ratios
> segmented.data <- segmented(result)
> called.data <- calls(result)
```

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:

```
> 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:

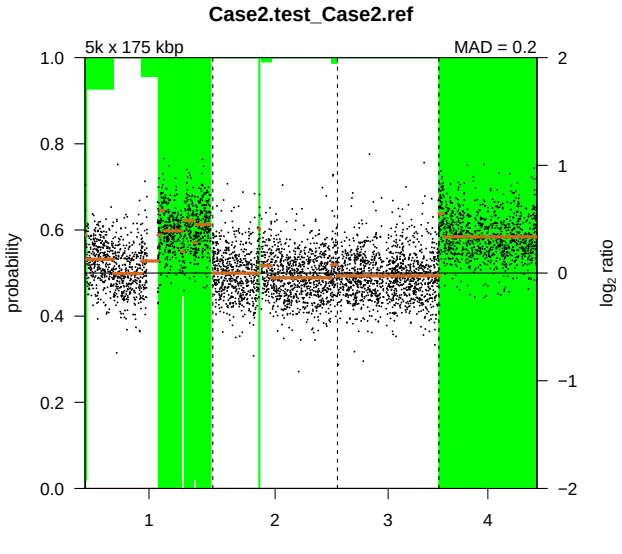

Figure 4: Results of the CGHnormaliter normalization for the second leukemia sample.

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

Plotting sample Case2.test_Case2.ref
```

Finally, the package provides the function `CGHnormaliter.write.table` to save the normalized data into a tabdelimited plain text file:

```
> 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:

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

Saving segmented log2 ratios to file: segmented.txt

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

Saving calls to file: called.txt
```

References

[1] K. Paulsson, M. Heidenblad, H. Mörse, Å. Borg, T. Fioretos, and B. Johansson. Identification of cryptic aberrations and characterization of translocation breakpoints using array CGH in high hyperdiploid childhood acute lymphoblastic leukemia. *Leukemia*, 20:2002–2007, 2006.

[2] M.A. van de Wiel, K.I. Kim, S.J. Vosse, W.N. van Wieringen, S.M. Wilting, and B. Ylstra. CGHcall: calling aberrations for array CGH tumor profiles. *Bioinformatics*, 23:892–894, 2007.

[3] B.P.P. van Houte, T.W. Binsl, H. Hettling, and J. Heringa. CGHnormaliter: a bioconductor package for normalization of array CGH data with many CNAs. *Bioinformatics*, 26(10):1366–1367, 2010.

[4] B.P.P. van Houte, T.W. Binsl, H. Hettling, W. Pirovano, and J. Heringa. CGHnormaliter: an iterative strategy to enhance normalization of array CGH data with imbalanced aberrations. *BMC Genomics*, 10:401, 2009.

[5] E.S. Venkatraman and A.B. Olshen. A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics*, 23(6):657–663, 2007.