CGHcall: Calling aberrations for array CGH tumor profiles.

Sjoerd Vosse and Mark van de Wiel

October 7, 2010

Department of Epidemiology & Biostatistics
VU University Medical Center

`mark.vdwiel@vumc.nl`

Contents

1 Overview

CGHcall allows users to make an objective and effective classification of their aCGH data into copy number states (loss, normal, gain or amplification). This document provides an overview on the usage of the CGHcall package. For more detailed information on the algorithm and assumptions we refer to the article (van de Wiel et al., 2007) and its supplementary material. As example data we attached the first five samples of the Wilting dataset (Wilting et al., 2006). After filtering and selecting only the autosomes 4709 datapoints remained.

2 Example

In this section we will use CGHcall to call and visualize the aberrations in the dataset described above. First, we load the package and the data:

```
> library(CGHcall)
> data(WiltingData)
> Wilting <- cghRaw(WiltingData)
```

Next, we apply the `preprocess` function which:

- removes data with unknown or invalid position information.
- shrinks the data to `nchrom` chromosomes.
- removes data with more than `maxmiss` % missing values.
- imputes missing values using `impute.knn` from the package `impute` (Troyanskaya et al., 2001).

```
> cghdata <- preprocess(Wilting, maxmiss = 30, nchrom = 22)
```

```
Changing impute.knn parameter k from 10 to 4 due to small sample size.
```

To be able to compare profiles they need to be normalized. In this package we provide very basic global median or mode normalization. Of course, other methods can be used outside this package. This function also contains smoothing of outliers as implemented in the DNAcopy package (Venkatraman and Olshen, 2007). Furthermore, when the proportion of tumor cells is not 100% the ratios can be corrected. See the article and the supplementary material for more information on cellularity correction (van de Wiel et al., 2007).

```
> tumor.prop <- c(0.75, 0.9, 0.8, 1, 1)
> norm.cghdata <- normalize(cghdata, method = "median", cellularity = tumor.prop,
+     smoothOutliers = TRUE)
```

```
Applying median normalization ...
Smoothing outliers ...
Adjusting for cellularity ...
Cellularity sample 1 :  0.75
Cellularity sample 2 :  0.9
Cellularity sample 3 :  0.8
Cellularity sample 4 :  1
Cellularity sample 5 :  1
```

The next step is segmentation of the data. This package only provides a simple wrapper function that applies the DNAcopy algorithm (Venkatraman and Olshen, 2007). Again, other segmentation algorithms may be used. To save time we will limit our analysis to the first two samples from here on.

```
> norm.cghdata <- norm.cghdata[, 1:2]
> seg.cghdata <- segmentData(norm.cghdata, method = "DNAcopy")

Start data segmentation ..
Analyzing: Sample.1
Analyzing: Sample.2
```

Post-segmentation normalization allows to better set the zero level after segmentation

```
> postseg.cghdata <- postsegnormalize(seg.cghdata)
```

Now that the data have been normalized and segments have been defined, we need to determine which segments should be classified as losses, normal, gains or amplifications.

```
> result <- CGHcall(postseg.cghdata)

[1] "changed"
EM algorithm started ...
[1] "Total number of segments present in the data: 121"
[1] "Number of segments used for fitting the model: 121"
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 409050   11     741108 19.8   741108 19.8
Vcells 383676    3     786432  6.0   786432  6.0
Calling iteration 1 :
     j        rl       mudl       musl        mun       mug      mudg      mua
[1,] 2 -3784.785 -0.8426813 -0.2957148 0.01181377 0.3261872 0.5576207 1.057502
           sddl     sdsl        sdn       sdg      sddg       sda
[1,] 0.08966041 0.089101 0.08812682 0.1494997 0.1498338 0.1498338
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 409481   11     741108 19.8   741108 19.8
Vcells 384582    3     786432  6.0   786432  6.0
Calling iteration 2 :
     j        rl      mudl       musl         mun       mug      mudg      mua
[1,] 2 -3784.209 -0.848784 -0.2944186 0.01615870 0.3298733 0.5639222 1.063802
```

```
          sddl       sdsl        sdn       sdg      sddg       sda
[1,] 0.08445677 0.08386266 0.08346952 0.1484393 0.1487757 0.1487758
Computing posterior probabilities for all segments ...
Total time: 1 minutes
```

In CGHcall version >=2.9.0 the result of CGHcall needs to be converted to a call object. This can be a large object for large arrays.

```
> result <- ExpandCGHcall(result, postseg.cghdata)
```

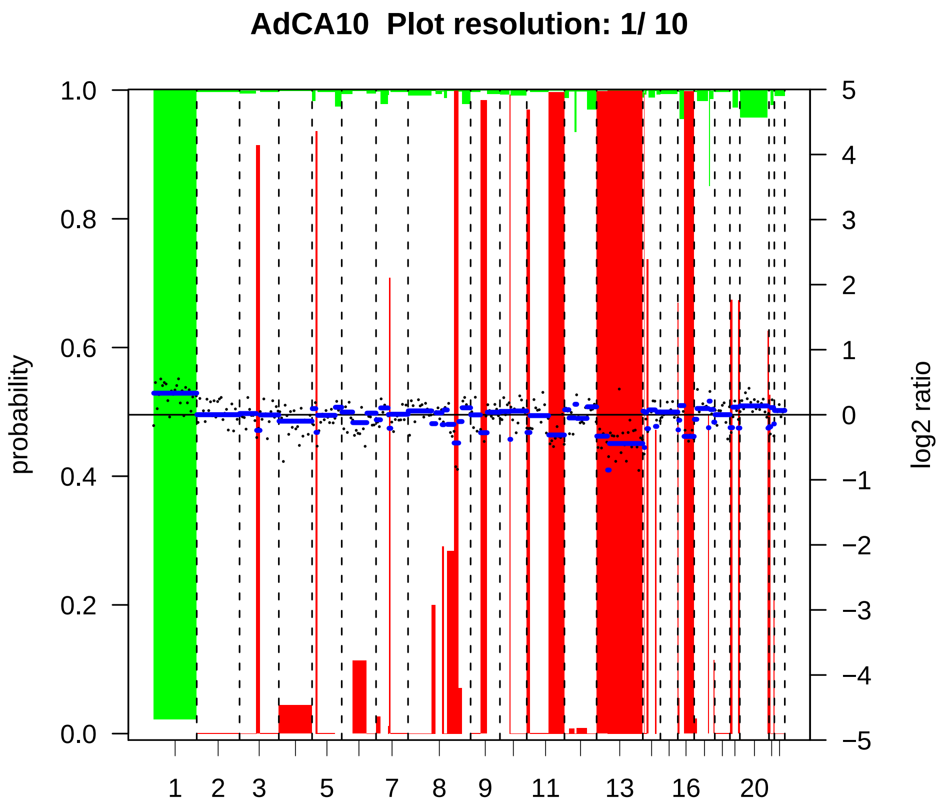
```
[1] 1
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 410804 11.0     741108 19.8   741108 19.8
Vcells 409382  3.2     905753  7.0   786432  6.0
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 410814 11.0     741108 19.8   741108 19.8
Vcells 423593  3.3     905753  7.0   904117  6.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 410813 11.0     741108 19.8   741108 19.8
Vcells 423592  3.3     905753  7.0   904117  6.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 410831 11.0     741108 19.8   741108 19.8
Vcells 444906  3.4     905753  7.0   904117  6.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411145 11.0     741108 19.8   741108 19.8
Vcells 446717  3.5     905753  7.0   904117  6.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411153 11.0     741108 19.8   741108 19.8
Vcells 448496  3.5     905753  7.0   904117  6.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411161 11.0     741108 19.8   741108 19.8
Vcells 450275  3.5     905753  7.0   904117  6.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411169 11.0     741108 19.8   741108 19.8
Vcells 452054  3.5     905753  7.0   904117  6.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411173 11.0     741108 19.8   741108 19.8
Vcells 453832  3.5     905753  7.0   904117  6.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411199 11.0     741108 19.8   741108 19.8
```

```
Vcells 469825  3.6     905753  7.0   904117  6.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411887 11.0     741108 19.8   741108 19.8
Vcells 477232  3.7    1031040  7.9   904117  6.9
[1] 2
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411898 11.0     741108 19.8   741108 19.8
Vcells 491464  3.8    1031040  7.9   904117  6.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411899 11.0     741108 19.8   741108 19.8
Vcells 491465  3.8    1031040  7.9  1027961  7.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411898 11.0     741108 19.8   741108 19.8
Vcells 491464  3.8    1031040  7.9  1027961  7.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411902 11.0     741108 19.8   741108 19.8
Vcells 495017  3.8    1031040  7.9  1027961  7.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411898 11.0     741108 19.8   741108 19.8
Vcells 491464  3.8    1031040  7.9  1027961  7.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411906 11.0     741108 19.8   741108 19.8
Vcells 493243  3.8    1031040  7.9  1027961  7.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411914 11.0     741108 19.8   741108 19.8
Vcells 495022  3.8    1031040  7.9  1027961  7.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411922 11.0     741108 19.8   741108 19.8
Vcells 496801  3.8    1031040  7.9  1027961  7.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411926 11.0     741108 19.8   741108 19.8
Vcells 498579  3.9    1031040  7.9  1027961  7.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 411952 11.1     741108 19.8   741108 19.8
Vcells 514572  4.0    1031040  7.9  1027961  7.9
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 414959 11.1     741108 19.8   741108 19.8
Vcells 502605  3.9    1031040  7.9  1030447  7.9
FINISHED!
Total time: 0 minutes
```

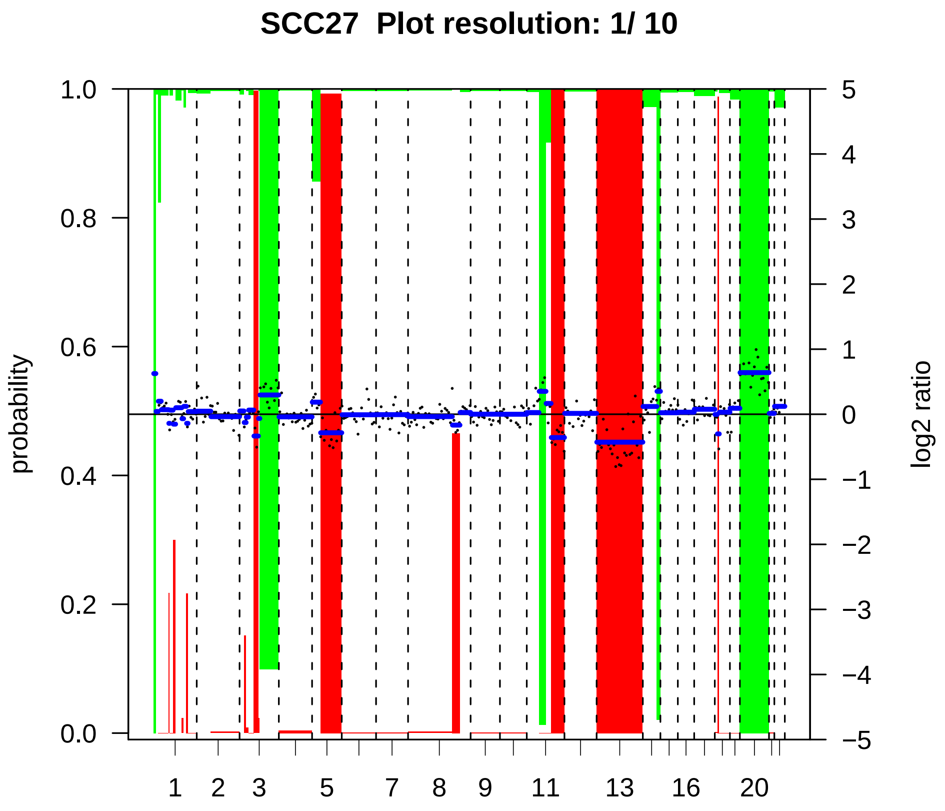
To visualize the results per profile we use the `plotProfile` function:

```
> plot(result[, 1])
```

```
Plotting sample AdCA10
```



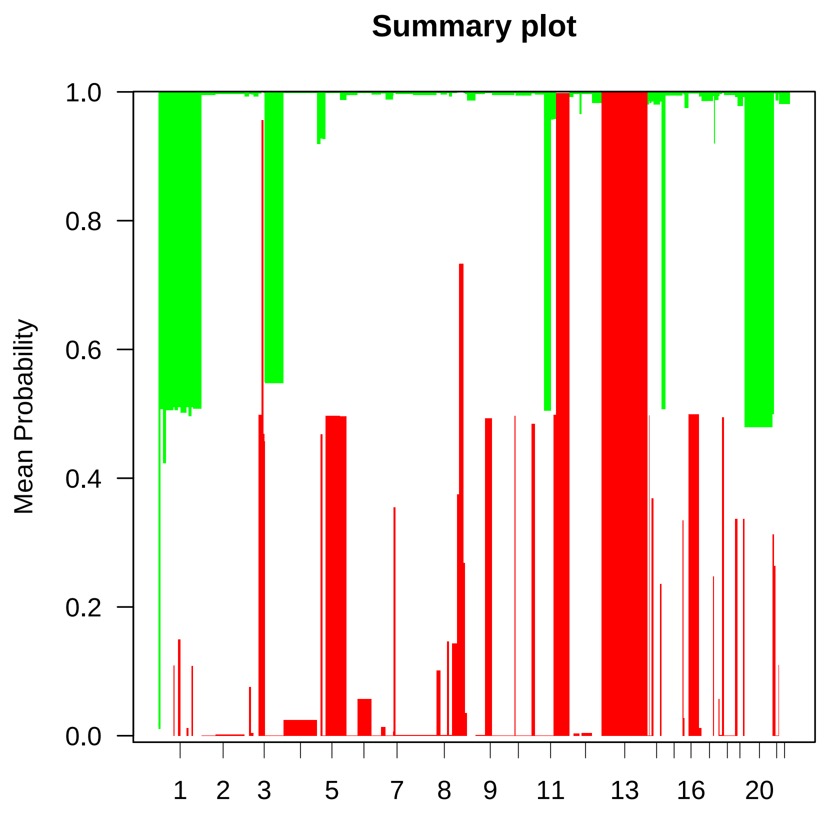
```
> plot(result[, 2])

Plotting sample SCC27
```



Alternatively, we can create a summary plot of all the samples:

```
> plot.summary(result)
```

```
Adding sample AdCA10 to summary plot.
Adding sample SCC27 to summary plot.
```



References

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17:520–525.

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

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

Wilting, S. M., Snijders, P. J. F., Meijer, G. A., Ylstra, B., van den Ijssel, P. R. L. A., Snijders, A. M., Albertson, D. G., Coffa, J., Schouten, J. P., van de Wiel, M. A., Meijer, C. J. L. M., and Steenbergen, R. D. M. (2006). Increased gene copy numbers at chromosome 20q are frequent in both squamous cell carcinomas and adenocarcinomas of the cervix. *J Pathol*, 209:220–230.