

CGHcall: Calling aberrations for array CGH tumor profiles.

Sjoerd Vosse and Mark van de Wiel

October 18, 2010

Department of Epidemiology & Biostatistics
VU University Medical Center

`mark.vdwiel@vumc.nl`

Contents

1 Overview	1
2 Example	1

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: 113"
[1] "Number of segments used for fitting the model: 113"
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 435273 23.3      741108 39.6      741108 39.6
Vcells 499287  3.9      1031040 7.9      1031040 7.9
Calling iteration 1 :
      j      rl      mudl      musl      mun      mug      mudg      mua
[1,] 2 -3770.814 -0.8429234 -0.2959666 0.01151765 0.3355313 0.5735946 1.073453
      sddl      sdsl      sdn      sdg      sddg      sda
[1,] 0.08667158 0.08609276 0.08947486 0.1710695 0.1713615 0.1713616
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 435704 23.3      741108 39.6      741108 39.6
Vcells 500158  3.9      1031040 7.9      1031040 7.9
Calling iteration 2 :
      j      rl      mudl      musl      mun      mug      mudg      mua
[1,] 2 -3769.749 -0.848933 -0.294113 0.01683709 0.3371155 0.5763027 1.076157
```

```

          sddl      sdsl      sdn      sdg      sddg      sda
[1,] 0.08073707 0.08011538 0.08195825 0.170614 0.1709068 0.1709068
Computing posterior probabilities for all segments ...
Total time: 1 minutes

```

In CGHcall version $\geq 2.9.0$ 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 437027 23.4      741108 39.6   741108 39.6
Vcells 532079  4.1     1031040 7.9   1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 437037 23.4      741108 39.6   741108 39.6
Vcells 546288  4.2     1031040 7.9   1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 437036 23.4      741108 39.6   741108 39.6
Vcells 546287  4.2     1031040 7.9   1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 437054 23.4      741108 39.6   741108 39.6
Vcells 567601  4.4     1162592 8.9   1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 437368 23.4      741108 39.6   741108 39.6
Vcells 571189  4.4     1162592 8.9   1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 437376 23.4      741108 39.6   741108 39.6
Vcells 574745  4.4     1162592 8.9   1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 437384 23.4      741108 39.6   741108 39.6
Vcells 578301  4.5     1162592 8.9   1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 437392 23.4      741108 39.6   741108 39.6
Vcells 581857  4.5     1162592 8.9   1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 437396 23.4      741108 39.6   741108 39.6
Vcells 585412  4.5     1162592 8.9   1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 437422 23.4      741108 39.6   741108 39.6

```

```

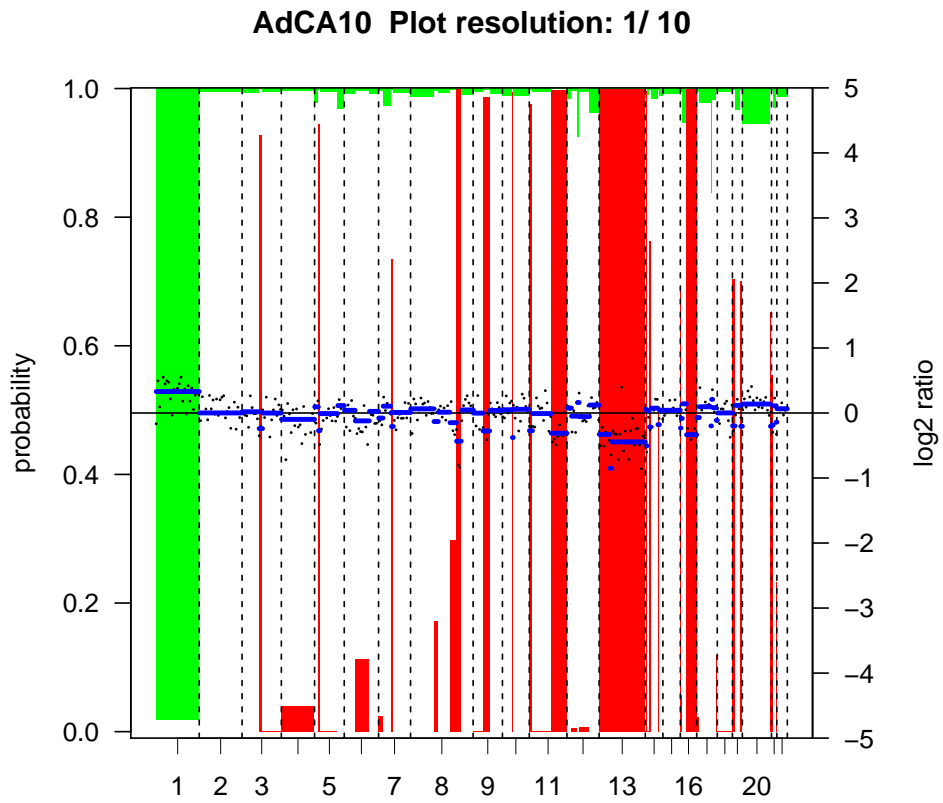
Vcells 606736 4.7 1162592 8.9 1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438110 23.4 741108 39.6 741108 39.6
Vcells 615963 4.7 1162592 8.9 1031040 7.9
[1] 2
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438114 23.4 741108 39.6 741108 39.6
Vcells 630172 4.9 1162592 8.9 1031040 7.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438115 23.4 741108 39.6 741108 39.6
Vcells 630173 4.9 1162592 8.9 1156950 8.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438114 23.4 741108 39.6 741108 39.6
Vcells 630172 4.9 1162592 8.9 1156950 8.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438118 23.4 741108 39.6 741108 39.6
Vcells 633725 4.9 1162592 8.9 1156950 8.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438114 23.4 741108 39.6 741108 39.6
Vcells 630172 4.9 1162592 8.9 1156950 8.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438122 23.4 741108 39.6 741108 39.6
Vcells 633728 4.9 1162592 8.9 1156950 8.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438130 23.4 741108 39.6 741108 39.6
Vcells 637284 4.9 1162592 8.9 1156950 8.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438138 23.4 741108 39.6 741108 39.6
Vcells 640840 4.9 1162592 8.9 1156950 8.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438142 23.4 741108 39.6 741108 39.6
Vcells 644395 5.0 1162592 8.9 1156950 8.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 438168 23.5 741108 39.6 741108 39.6
Vcells 665719 5.1 1300721 10.0 1156950 8.9
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 441175 23.6 741108 39.6 741108 39.6
Vcells 641350 4.9 1300721 10.0 1299655 10.0
FINISHED!
Total time: 0 minutes

```

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

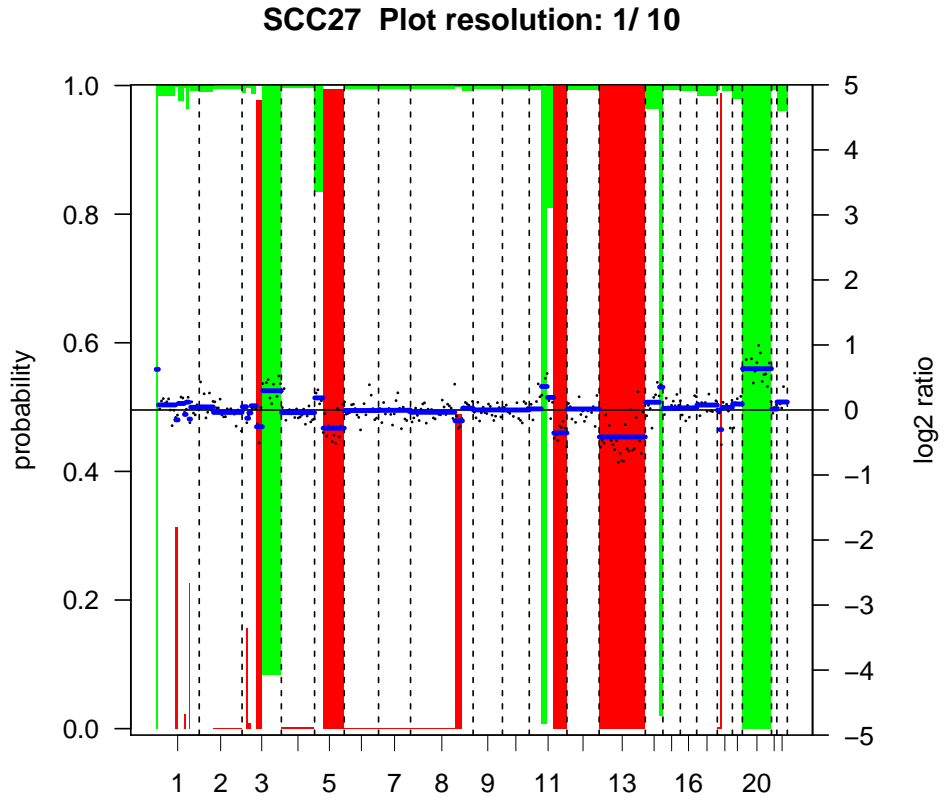
```
> plot(result[, 1])
```

Plotting sample AdCA10



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

Plotting sample SCC27

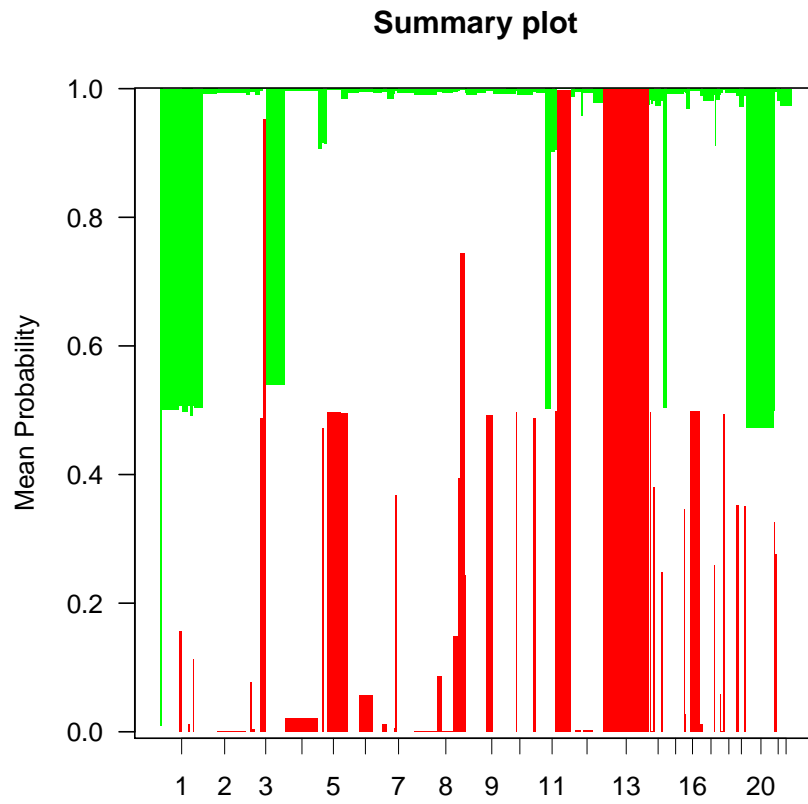


Alternatively, we can create a summary plot of all the samples:

```
> 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.