CNAnorm: A package for normalization of CNA sequencing data

Stefano Berri

October 2, 2012

CNAnorm is a package for the analysis of Copy Number Alteration (CNA) of tumor samples using low coverage (around 0.01 - 0.5X) high throughput sequencing[1]. In particular, CNAnorm aims to perform a meaningful *normalization* of the sample by estimation of the underlying tumor's ploidy. CNAnorm allows both a fully automated as well as an interactive approach to the normalization step. If the user has some external "clues" about the ploidy of the genome, it is possible to manually inform CNAnorm and then perform the normalization. CNAnorm also provides a method to plot the normalised genome.

For more information and the original data, see the authors' website.

`http://www.precancer.leeds.ac.uk/cnanorm/`

1 Input data

You can load the example data

```
> library(CNAnorm)
> data(LS041)
> # show the data
> LS041[1:5,]
```

```
   Chr     Pos Test Norm       GC
1 chr1       1    0    0 41.63200
2 chr1  426912    9    7 44.95821
3 chr1  853823   58   45 61.97635
4 chr1 1280734   34   36 56.40422
5 chr1 1707645   69   46 54.39321
```

The input data can be produced from sam/bam files using the perl script `bam2windows.pl` that you can obtain from the authors' website. There you can also find the bam files used to produce the dataframe LS041.

The first step is to create an object of class CNAnorm with the input data. The input data consists of number of reads in test and control for a variable number of **constant width windows**. Chromosome/contig name and the starting position of each window must be provided as well. GC content for each window is optional. If the data is in a dataframe like LS041, the easiest is to use the function *dataFrame2object* to create a new object.

```
> CN <- dataFrame2object(LS041)
```

Because of difficulty in correctly mapping reads on chromosome Y and M, we can flag them (together with the mitochondrial chromosome) and not to use them for the (optional) GC correction or ploidy detection.

```
> toSkip <- c("chrY", "chrM")
> CN <- gcNorm(CN, exclude = toSkip)
```

It is then strongly recommended to smooth the signal[2] to decrease noise without losing resolution.

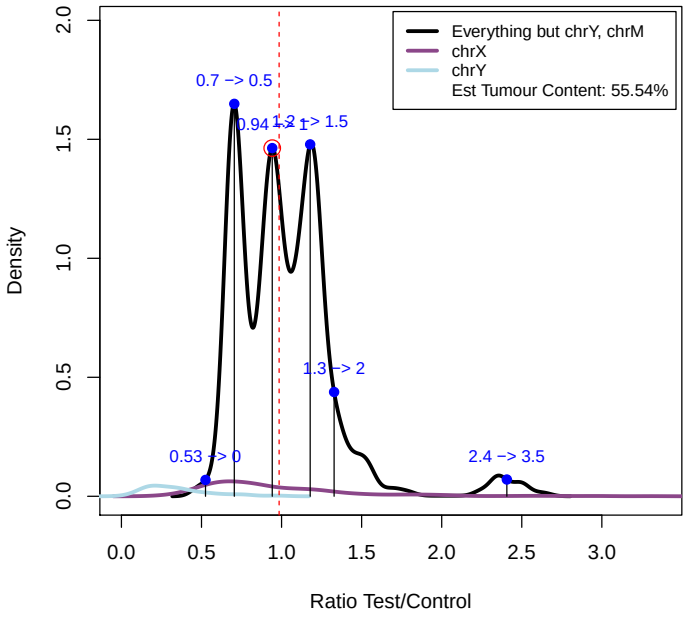
```
> CN <- addSmooth(CN, lambda = 7)
```

It is now possible to estimate peaks and ploidy.

```
> CN <- peakPloidy(CN, exclude = toSkip)
```

We can now visualise the distribution of reads and the suggested ploidy.

```
> plotPeaks(CN, special1 = 'chrX', special2 = 'chrY')
```



At this point, we can accept the suggestion of CNAnorm.

```
> CN.default <- validation(CN)
```

However, if there is a reason - for instance from FISH - to believe that the ploidy is actually one copy more than suggested, we can correct the suggestion.

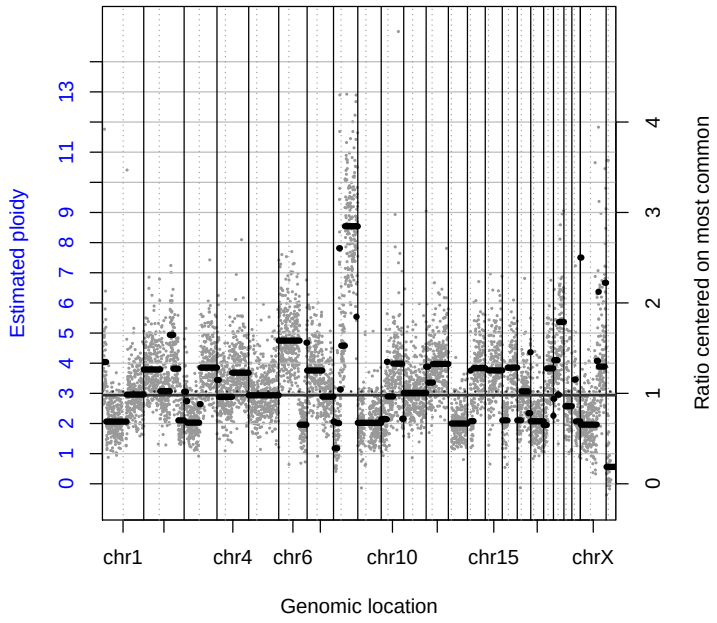
```
> CN <- validation(CN, ploidy = (sugg.ploidy(CN) + 1) )
```

If we want to plot DNAcopy segments and discrete ploidy value, before the normalization we need to add the DNAcopy information.

```
> CN <- addDNACopy(CN)
> CN <- discreteNorm(CN)
```

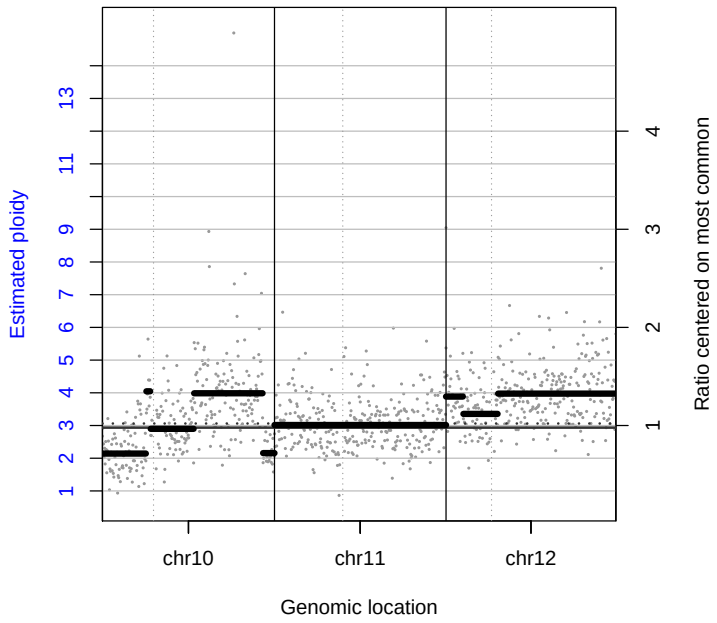
Finally, we can plot the whole genome.

```
> plotGenome(CN, superimpose = 'DNACopy')
```



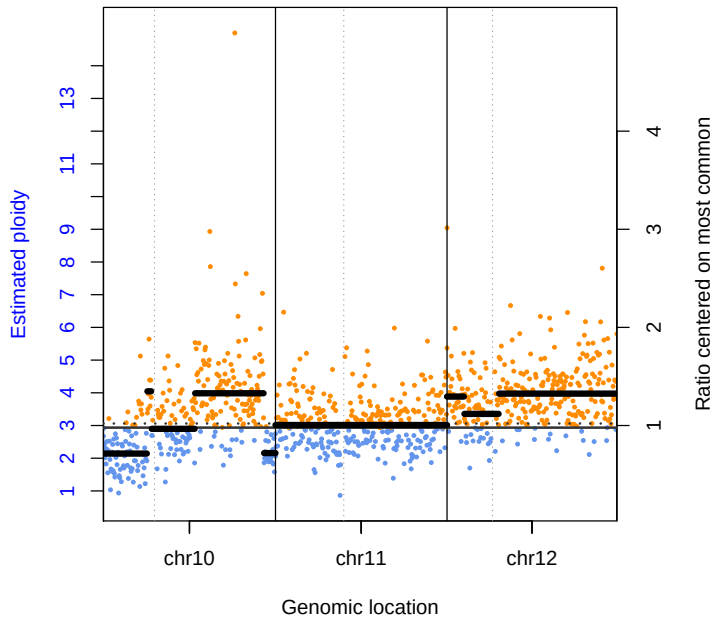
Or just a subset of it.

```
> toPlot <- c('chr10', 'chr11', 'chr12')
> subSet <- chrs(CN) %in% toPlot
> plotGenome(CN[subSet], superimpose = 'DNACopy')
```



Several aspects of the plot can be customized (colors, point and line size and so on) by setting the desired value in object gPar and then passing it to plotGenome

```
> data(gPar)
> gPar$genome$colors$gain.dot <- 'darkorange'
> gPar$genome$colors$grid <- NULL
> gPar$genome$cex$gain.dot <- .4
> gPar$genome$cex$loss.dot <- .4
> plotGenome(CN[subSet], superimpose = 'DNACopy', gPar = gPar, colorful = TRUE)
```



Finally, we can export results in a table-like format.

```
> exportTable(CN, file = "CNAnorm_table.tab", show = 'ploidy')
```

References

[1] Arief Gusnanto, Henry M Wood, Yudi Pawitan, Pamela Rabbitts, and Stefano Berri. Correcting for cancer genome size and tumour cell content enables better estimation of copy number alterations from next generation sequence data. *Bioinformatics*, 2011.

[2] Jian Huang, Arief Gusnanto, Kathleen O'Sullivan, Johan Staaf, Ake Borg, and Yudi Pawitan. Robust smooth segmentation approach for array CGH data analysis. *Bioinformatics*, 23(18):2463–9, Sep 2008.