CNAnorm: A package to detect Copy Number Alterations (CNA) from sequencing data

Stefano Berri

May 19, 2021

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

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 (`http://www.precancer.leeds.ac.uk/cnanorm`). 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 a number of reads in the test (tumour) and control (normal) for a variable number of **constant width windows**. Chromosome/contig name and the starting position of each window must also be provided. GC content for each window is optional. If the data is in a dataframe like LS041, it is easiest to use the function *dataFrame2object* to create a new object.

```
> # set a seed for reproducible comparison
> set.seed(31)
> CN <- dataFrame2object(LS041)
```

2 Basic use

Here, we will demostrate a standard and robust set of instructions for obtaining copy number profile without correcting for tumour content or ploidy.

2.1 GC-correction

GC correction is an optional but recommended step to reduce GC content sequencing bias.

Because of the difficulty in correctly mapping reads to chromosome Y and M, we can flag them for exclusion from the GC correction or ploidy detection.

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

2.2 From read count to copy number profiles

CNAnorm will try to detect "states" of copy number in order to perform normalisation. It is strongly recommended to smooth the signal[2] to decrease noise without losing resolution.

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

It is now possible to search for the most representative peak (state) closest to the median of the ratios.

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

'closest' is a robust normalisation method. It will not attempt to estimate ploidy and tumour content, but it will work with a wide range of samples, windows sizes, total number of reads, and so on. This method can cope with heterogeneous, policlonal tumours. Once the user is more familiar with CNAnorm, methods 'density' or 'mixture' can be used to estimate ploidy and tumour content if the analysed tumour is largely monoclonal. See section "Advanced use"

We can now visualise the distribution of reads.

```
> plotPeaks(CN)
```
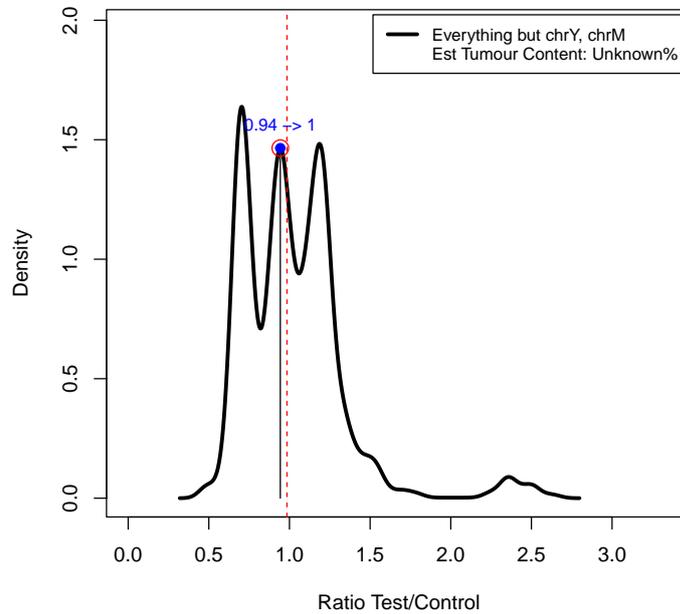
2.2.1 Segmentation

CNAnorm uses an external package, DNAcopy[3], to segment the data. If we want to plot DNAcopy segments, before the normalisation we need to add the DNAcopy information.

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

The above steps can be performed also with function CNAnormWorkflow that exposes most of the parameters of all functions with robust defaults.

```
> # re-set seed for reproducible results
> set.seed(31)
> CNauto <- CNAnormWorkflow(LS041, gc.do=TRUE, gc.exclude=toSkip,
+     peak.exclude=toSkip)
> identical(CN, CNauto)
```

```
[1] TRUE
```

Finally, we can plot the copy number profile for the whole genome. As there is no assumption about the species of the sample, we will use show.centromeres = FALSE

```
> plotGenome(CN, superimpose='DNACopy', show.centromeres=FALSE)
```
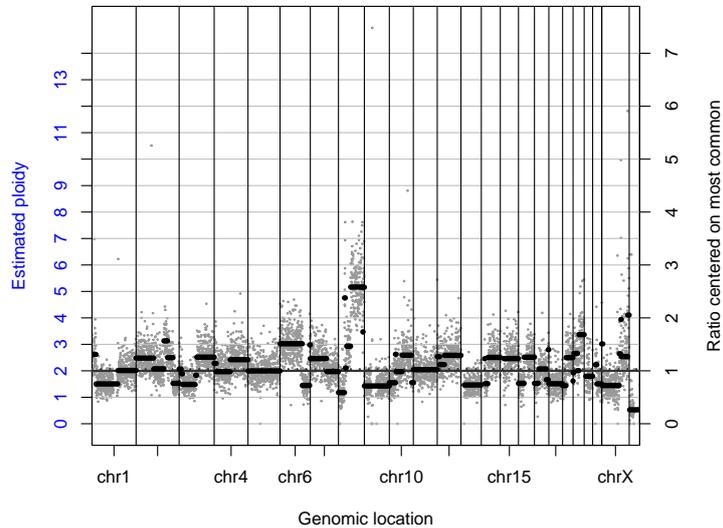
We can also specify a set of chromosomes to show, or a specific range. If the reference genome is the human genome (hg19), we can use built-in information to plot the location of centromeres.

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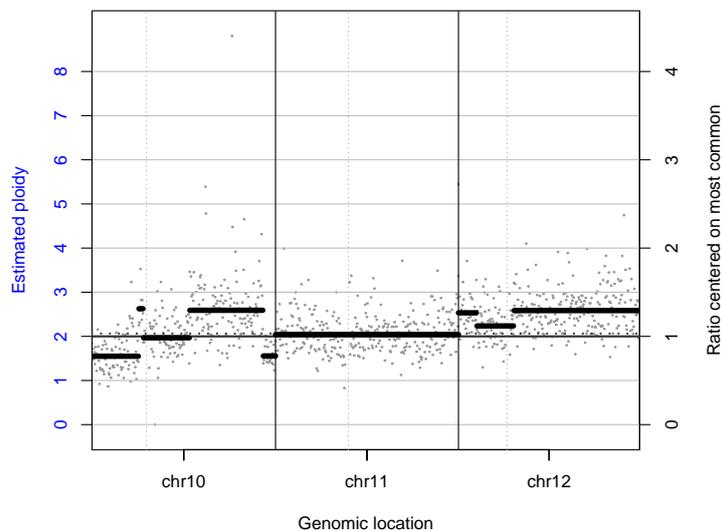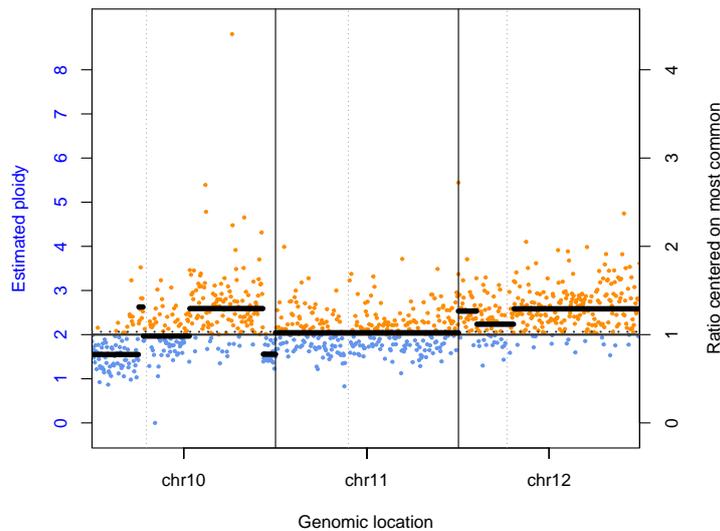

Several aspects of the plot can be customised (colours, point and line size, etc.) by setting the desired value in object gPar and then passing it to plotGenome

```
> 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 = 'center')
```

3   Advanced use

CNAnorm was designed to be able to detect absolute or integer copy number states and detect tumour content. It is important to remember, though, that several solutions could be equally supported by the same dataset[1] and that the process will succeed only if the tumour is largely monoclonal.

We can use a different method to normalise the data, in order to estimate absolute ploidy and tumour content.

```
> CN <- peakPloidy(CN, exclude=toSkip, method='density')
> plotPeaks(CN)
```
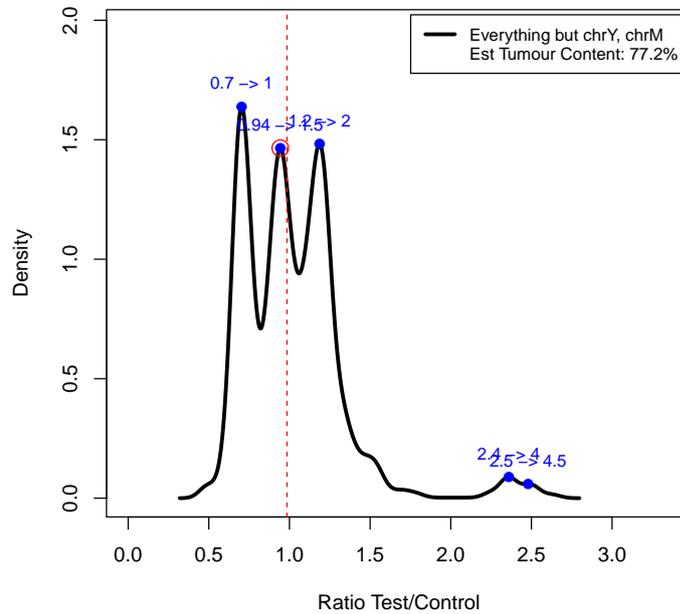
We can now see that each peak has been assigned to an integer state and an estimate of tumour content is given. The vertical dotted red line shows the median value of the ratio. The blue labels (in the form "BN → AN") on top of each peak shows the ratio Before Normalisation and the ratio After Normalisation. Ploidy (plotted on the left axis of the genome plot), will be 2 * AN. The red circle is used to define ratio = 1 on the right axes of the genome plot.

We can also provide extra arguments to the segmentation engine: DNAcopy.weight and DNAcopy.segment improve segmentation. These are particularly important if higher depth data is used in order to minimise false positives (oversegmentation). As the data is from human tumour samples, we can use built-in information about centromere location to perform segmentation on a per chromosome arm basis (independent.arms).

```
> CN <- validation(CN)
> data(hg19_hs_ideogr)
> CN <- addDNACopy(CN, DNAcopy.weight='poisson',
+     DNAcopy.segment=list(alpha=0.001), independent.arms=TRUE,
+     ideograms=hg19_hs_ideogr)
> CN <- discreteNorm(CN)
> plotGenome(CN, superimpose='DNACopy')
```

6
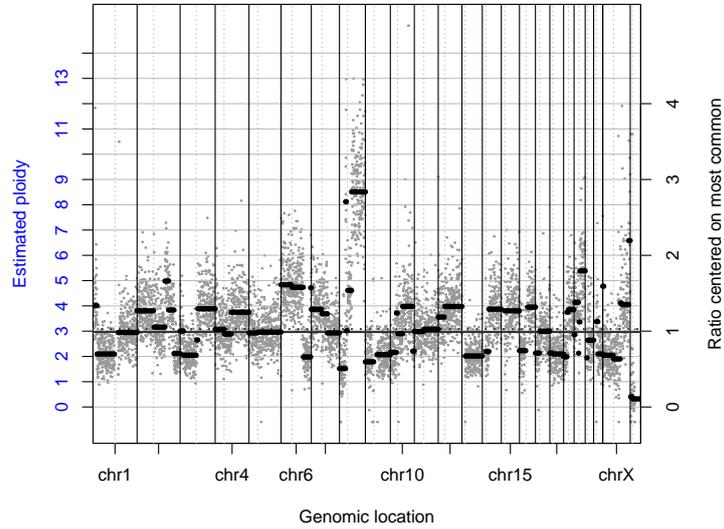
The genome wide copy number profile is now be mostly aligned to integer states.

3.1 Manual correction

The reported normalisation is highly plausible. Let's assume, however, that the user knows that all chromosomes assigned to copy number 2 are in LOH (independent evidence) and/or that there are exactly 2 copies of chr5 (by FISH) and/or that the tumour content is probably lower (pathologist report). The user can thus "shift" the state up or down

```
> CN <- validation(CN, ploidy = (sugg.ploidy(CN) - 1))
> plotPeaks(CN, show='validated')
```
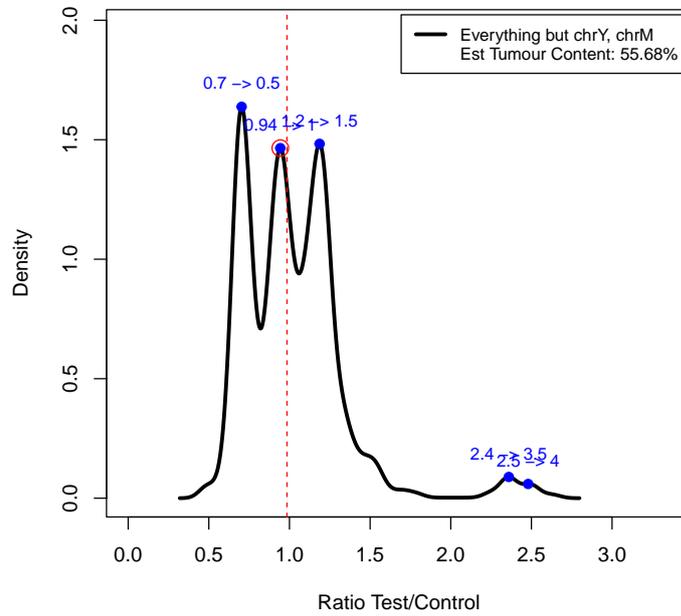
Ploidy has been shifted and a new tumour content estimation provided. We need to perform the normalisation once more and then we can plot the new copy number profile

```
> CN <- discreteNorm(CN)
> plotGenome(CN, superimpose='DNACopy')
```
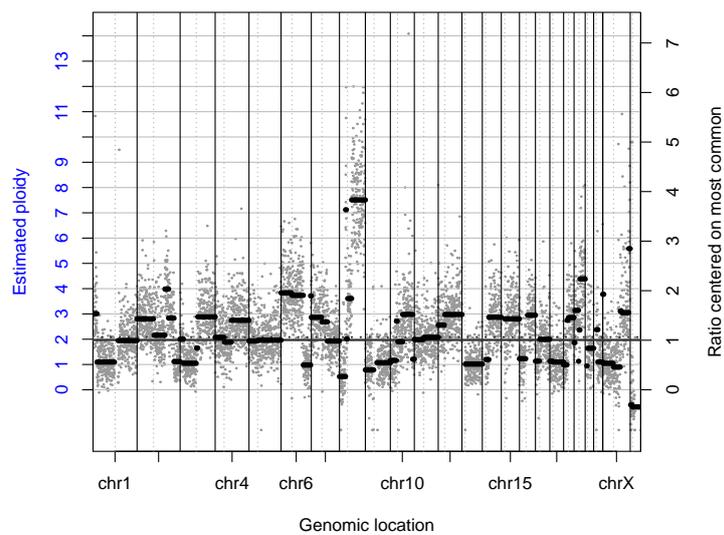
References

[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.

[3] E. S. Venkatraman and A. B. Olshen. A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics*, 2007.