CopywriteR: DNA copy number detection from off-target sequence data.

Thomas Kuilman

October 14, 2015

Department of Molecular Oncology
Netherlands Cancer Institute
The Netherlands

t.kuilman@nki.nl or thomaskuilman@yahoo.com

Contents

1 Overview

To extract copy number information from targeted sequencing data while circumventing problems related to the use of on-target reads, we developed the CopywriteR package. Unlike other currently available tools, CopywriteR is based on off-target reads. We have shown that this approach has several advantages relative to current approaches [1] and it constitutes a viable alternative for copy number detection from targeted sequencing data.

2 CopywriteR workflow

CopywriteR uses .bam files as an input. These files are processed in several steps to allow copy number detection from off-target reads of targeted sequencing efforts. These steps include:

- Removing of low-quality and anomalous reads
- Peak calling (in reference when available, otherwise in sample itself)
- Discarding of reads in peak regions
- Counting reads on bins of pre-defined size

- Compensating for the difference in effective bin size upon discarding peaks in peak regions
- Correcting for GC-content and mappability; applying a blacklist filter for CNV regions

3 Analysis workflow

The full analysis of copy number data with CopywriteR includes three sequential steps, using the `preCopywriteR()`, `CopywriteR()` and `plotCNA()` functions respectively.

In this analysis workflow, we will use CopywriteR to extract copy number information from whole-exome sequening data from a murine small-cell lung cancer. The complete dataset can be downloaded from the European Nucleotide Archive (http://www.ebi.ac.uk/ena/home) using the accession number PRJEB6954. In this vignette, only the sequence reads on chromosome 4 are used. The .bam file, which has been pre-filtered to minimize disk usage, is contained in the *SCLCBam* package. Before starting the analysis, GC-content and mappability helper files need to be created for the desired bin size and reference genome. This is done using the `preCopywriteR()` function.

3.1 `preCopywriteR()`

CopywriteR uses binned read count data as a basis for copy number detection. We have pre-assembled 1 kb bin GC-content and mappability information for the hg18, hg19, hg38, mm9 and mm10 reference genomes in the *CopyhelpeR* package. The `preCopywriteR()` function allows to create the necessary helper files. These files can be created for any custom bin size that is a multiple of 1000 bp, and for the above-mentioned reference genomes. The helper files are saved in a new folder that is names after the relevant reference genome, the bin size, and the prefix (all separated by '_'). The helper files are required to run CopywriteR. If previously created helper files for a specific bin size / reference genome / prefix combination are available, the `preCopywriteR()` step can be omitted.

In this example, CopywriteR will be applied to a bin size of 20 kb. We recommend to use a bin size of 20 kb when trying to analyze whole-exome sequencing data, while we would start analyzing at 50 kb resolution for targeted sequencing on smaller gene panels. As the .bam file contained in the *SCLCBam* package is mapped to the mm10 reference genome, we run `preCopywriteR()` as follows:

```
> library(CopywriteR)
```

```
> data.folder <- tools::file_path_as_absolute(file.path(getwd()))
```

```
> preCopywriteR(output.folder = file.path(data.folder),
+               bin.size = 20000,
+               ref.genome = "mm10_4")
```

```
The output folder âĂŸ/private/tmp/RtmpHWP7PU/Rbuild5590d0d243e/CopywriteR/
  vignettesâĂŹ has been detected
Generated GC-content and mappability data at 20000 bp resolution...
Generated blacklist file...
```

Please not that the "mm10_4" reference genome exclusively contains helper files for chromosome 4, and therefore should only be used when replicating the analysis in this vignette. The custom helper files are placed in a folder that is named after the reference genome and the bin size (in this example: mm10_4_20kb). This folder, on its turn, is placed inside the folder specified by the output.folder argument. The bin.size argument specifies the custom bin size (in bp), and the ref.genome argument can be either "hg18", "hg19", "hg38", "mm9" or "mm10". The default chromosome notation is "1", "2", ..., "X", "Y". The prefix argument is optional and can be used to add a prefix to these names.

If you are interested in getting helper files for other genomes, please contact me by the email addresses supplied at the beginning of this vignette.

```
> list.dirs(path = file.path(data.folder), full.names = FALSE)[2]
```

```
[1] "mm10_4_20kb"
```

```
> list.files(path = file.path(data.folder, "mm10_4_20kb"), full.names = FALSE)
```

```
[1] "GC_mappability.rda" "blacklist.rda"
```

The data are contained in two GRanges objects (from the *GenomicRanges* package) of which one (blacklist.rda) contains regions of copy number variation:

```
> load(file = file.path(data.folder, "mm10_4_20kb", "blacklist.rda"))
> blacklist.grange
```

```
GRanges object with 200 ranges and 0 metadata columns:
        seqnames                ranges strand
           <Rle>             <IRanges>  <Rle>
    [1]        4     [3078753, 3092353]      *
    [2]        4     [3197053, 3198153]      *
    [3]        4     [3227753, 3229153]      *
    [4]        4     [4641353, 4642553]      *
    [5]        4     [5462253, 5465053]      *
    ...      ...                    ...    ...
  [196]        4 [149809791, 149811291]      *
  [197]        4 [152477391, 152478591]      *
  [198]        4 [153152191, 153154091]      *
  [199]        4 [155624191, 155625291]      *
  [200]        4 [155769791, 155770891]      *
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

The other helper file (GC_mappability.rda) contains GC-content and mappability information for a particular bin size:

```
> load(file = file.path(data.folder, "mm10_4_20kb", "GC_mappability.rda"))
> GC.mappa.grange[1001:1011]
```

```
GRanges object with 11 ranges and 3 metadata columns:
       seqnames               ranges strand   | ATcontent GCcontent mappability
          <Rle>            <IRanges>  <Rle>   | <numeric> <numeric>   <numeric>
   [1]        4 [20000001, 20020000]      *   |     0.577     0.423       0.881
   [2]        4 [20020001, 20040000]      *   |     0.605     0.395        0.96
   [3]        4 [20040001, 20060000]      *   |     0.596     0.404       0.973
   [4]        4 [20060001, 20080000]      *   |     0.588     0.412       0.936
   [5]        4 [20080001, 20100000]      *   |     0.603     0.397       0.814
   ...      ...                  ...    ... ...       ...       ...         ...
   [7]        4 [20120001, 20140000]      *   |     0.598     0.402         0.7
   [8]        4 [20140001, 20160000]      *   |      0.61      0.39       0.949
   [9]        4 [20160001, 20180000]      *   |       0.6       0.4       0.597
  [10]        4 [20180001, 20200000]      *   |     0.606     0.394       0.799
  [11]        4 [20200001, 20220000]      *   |     0.601     0.399       0.629
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

3.2 BiocParallel for parallel computing

CopywriteR fully supports paralel computing and is implemented in such a way that every sample is processed on a single core. CopywriteR uses the *BiocParallel* for parallel computing. This package requires the user to specify which parallel environment is used by creating an instance of `BiocParallelParam`. In the example case, we are using the `snow` environment and specify a `SnowParam`:

```
> bp.param <- SnowParam(workers = 1, type = "SOCK")
> bp.param
```

```
class: SnowParam
  bpjobname:BPJOB; bpworkers:1; bptasks:0; bptimeout:Inf; bpRNGseed:; bpisup:FALSE
  bplog:FALSE; bpthreshold:INFO; bplogdir:NA
  bpstopOnError:FALSE; bpprogressbar:FALSE
  bpresultdir:NA
cluster type: SOCK
```

For using other instances of the BiocParallelParam parameter, please refer to the *BiocParallel* package vignette. The bp.param parameter can now be passed on to CopywriteR() to allow parallel computation.

3.3 CopywriteR()

The CopywriteR() function allows to calculate binned read count data based on the helper files created by preCopywriteR(). CopywriteR uses a peak calling algorithm to remove 'on-target' reads. For every sample that is to be analyzed by CopywriteR, the user can specify on which sample the peak regions should be identified. The argument that controls this is sample.control. This matrix or data.frame should contain path names to 'samples' in the first column, and their corresponding 'controls', in the second column. The peaks called in a 'control' are used to remove reads from the corresponding 'sample'. Please note that any sample that is to be used by the downstream plotCNA function (including those that are only to be used as a reference) needs to be analyzed by the CopywriteR function.

In the example used here, no matched germline control is available, and peaks are called on the sample itself. Therefore the paths specified in the samples and controls columns of the sample.control data.frame below are identical.

```
> path <- SCLCBam::getPathBamFolder()
> samples <- list.files(path = path, pattern = ".bam$", full.names = TRUE)
> controls <- samples
> sample.control <- data.frame(samples, controls)
> CopywriteR(sample.control = sample.control,
+            destination.folder = file.path(data.folder),
+            reference.folder = file.path(data.folder, "mm10_4_20kb"),
+            bp.param = bp.param)
```

```
The following samples will be analyzed:
sample: T43_4.bam ;          matching control: T43_4.bam
The bin size for this analysis is 20000
The capture region file is not specified
This analysis will be run on 1 cpus
Plotting to file /private/tmp/RtmpHWP7PU/Rbuild5590d0d243e/CopywriteR/vignettes/CNAprofiles/qc/read.counts
Total calculation time of CopywriteR was:  1.158597
```

The destination.folder argument determines in which folder the output folder is going to be placed, and the reference.folder points to the location of the custom bin size helper files. The keep.intermediary.files is an optional argument that determines whether intermediary .bam, .bai and peak .bed files are kept. The default value is FALSE to limit disk space usage; it can be set to TRUE however for troubleshooting purposes. Finally, the capture.regions.file argument is optional, and can be used to define the location of a capture regions bed file. If this file is specified, statistics of the overlap of these regions with the called peaks is provided.

CopywriteR will create a new folder named CNAprofiles in the directory specified by destination.folder. This folder contains the following files:

```
> cat(list.files(path = file.path(data.folder, "CNAprofiles")), sep = "\n")
```

```
CopywriteR.log
input.Rdata
log2_read_counts.igv
```

```
qc
read_counts.txt
```

The read_counts.txt file contains both uncompensated and compensated read counts for every sample, as well as the corresponding fraction.of.bin values. The 'fraction of bin' indicates what fraction of the bin is not on peaks, and therefore effectively contributes to the read count. In our example, the read_counts.txt has the following content:

```
> read.table(file = file.path(data.folder, "CNAprofiles", "read_counts.txt"),
+            header = TRUE)[1001:1006, ]
```

```
     Chromosome    Start      End             Feature read.counts.compensated.T43_4.bam
1001          4 20000001 20020000 4:20000001-20020000                                  52
1002          4 20020001 20040000 4:20020001-20040000                                  50
1003          4 20040001 20060000 4:20040001-20060000                                  41
1004          4 20060001 20080000 4:20060001-20080000                                  44
1005          4 20080001 20100000 4:20080001-20100000                                  24
1006          4 20100001 20120000 4:20100001-20120000                                  50
     read.counts.T43_4.bam fraction.of.bin.T43_4.bam
1001                    52                         1
1002                    50                         1
1003                    41                         1
1004                    44                         1
1005                    24                         1
1006                    50                         1
```

The log2_read_counts.igv file can be opened in the IGV browser (http://www.broadinstitute.org/igv/), and contains log2-transformed, normalized (ratios of) read counts. These data can be used for further downstream analysis, and are required for `plotCNA()` to allow plotting of the copy number profiles:

```
> read.table(file = file.path(data.folder, "CNAprofiles",
+                             "log2_read_counts.igv"), header = TRUE)[817:822, ]
```

```
    Chromosome    Start      End             Feature log2.T43_4.bam
817          4 20000001 20020000 4:20000001-20020000    -0.43105895
818          4 20020001 20040000 4:20020001-20040000     0.19585525
819          4 20040001 20060000 4:20040001-20060000    -0.53023049
820          4 20060001 20080000 4:20060001-20080000    -0.57540523
821          4 20080001 20100000 4:20080001-20100000    -0.50525699
822          4 20100001 20120000 4:20100001-20120000    -0.01782003
```

The input.Rdata file contains a number of variables that are required to run the last function of the CopywriteR package, `plotCNA()`. The CopywriteR.log file contains log information of the R commands that have been used to perform the various subfunctions, and specifications of the input material. Finally, the qc folder contains two types of quality control plots.

```
> cat(list.files(path = file.path(data.folder, "CNAprofiles", "qc")), sep = "\n")
```

```
fraction.of.bin.T43_4.bam.pdf
read.counts.compensated.T43_4.bam.png
```

The fraction.of.bin files contain the empirical cumulative distribution function for the 'fractions of bins'. The read.counts.compensated files contain the plots and the loesses that are used for GC-content and mappability corrections.

3.4 plotCNA()

The `plotCNA()` function allows segmentation of the copy number data using DNAcopy [2], and subsequent plotting. We run the plotting function as follows:

```
> plotCNA(destination.folder = file.path(data.folder))
```

```
Analyzing: log2.T43_4.bam.vs.log2.T43_4.bam
Analyzing: log2.T43_4.bam.vs.none
Total calculation time of CopywriteR was:  2.076758
```

The plotting function saves the DNAcopy object containing the segmentation values in the segment.Rdata file. In addition, it creates the folder 'plots':

```
> cat(list.files(path = file.path(data.folder, "CNAprofiles")), sep = "\n")
```

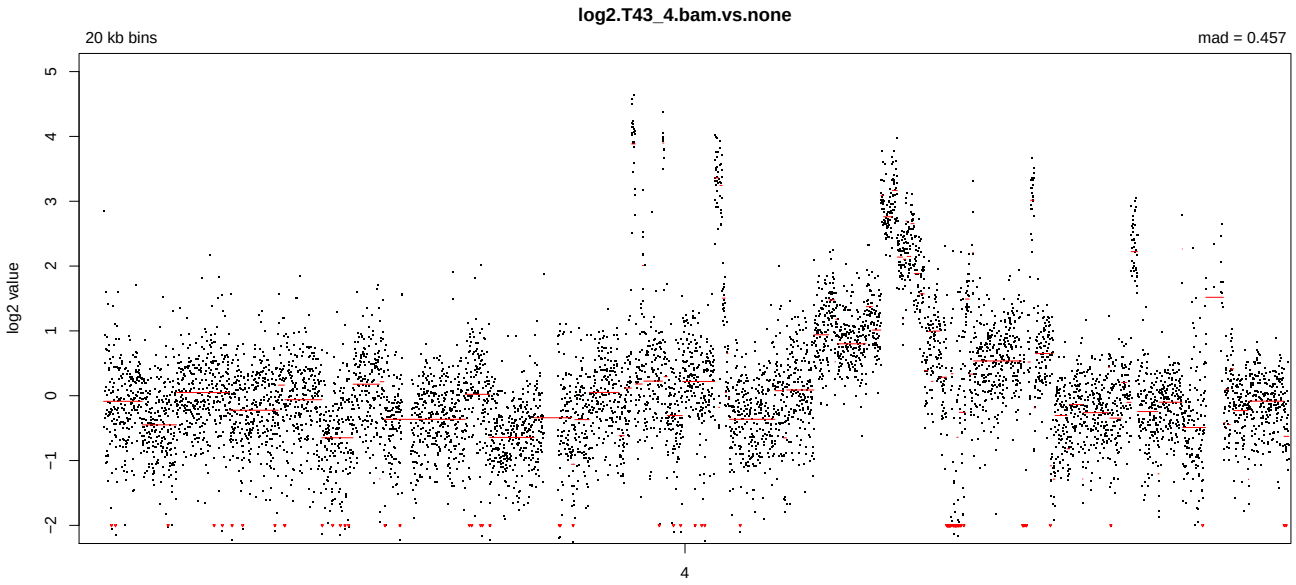
```
CopywriteR.log
input.Rdata
log2_read_counts.igv
plots
qc
read_counts.txt
segment.Rdata
```

For every sample a separate directory is created:

```
> cat(list.files(path = file.path(data.folder, "CNAprofiles", "plots")), sep = "\n")
```

```
log2.T43_4.bam.vs.log2.T43_4.bam
log2.T43_4.bam.vs.none
```

The samples are plotted per chromosome, as well as in a genome-wide fashion. We provide here the result for chromosome 4:



In addition to the plots, the raw segmented values can be obtained from the DNAcopy object 'segment.Rdata'.

We are interested in further improving CopywriteR, so if CopywriteR fails to analyze your samples, please don't hesitate to contact me. In this case, please provide the full command line output and the log-file.

4 Troubleshooting

We maintain a troubleshooting GitHub page where various issues are discussed that have been raised by CopywriteR users. Please follow this link: https://github.com/PeeperLab/CopywriteR#troubleshooting.

5 Session Information

The version number of *R* and packages loaded for generating the vignette were:

```
> toLatex(sessionInfo())
```

- R version 3.2.2 (2015-08-14), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: BiocParallel 1.4.0, CopywriteR 2.2.0
- Loaded via a namespace (and not attached): Biobase 2.30.0, BiocGenerics 0.16.0, BiocStyle 1.8.0, Biostrings 2.38.0, CopyhelpeR 1.1.1, DNAcopy 1.44.0, GenomeInfoDb 1.6.0, GenomicAlignments 1.6.0, GenomicRanges 1.22.0, IRanges 2.4.0, RColorBrewer 1.1-2, Rsamtools 1.22.0, S4Vectors 0.8.0, SCLCBam 1.1.0, ShortRead 1.28.0, SummarizedExperiment 1.0.0, XVector 0.10.0, bitops 1.0-6, chipseq 1.20.0, chron 2.3-47, data.table 1.9.6, futile.logger 1.4.1, futile.options 1.0.0, grid 3.2.2, gtools 3.5.0, hwriter 1.3.2, lambda.r 1.1.7, lattice 0.20-33, latticeExtra 0.6-26, matrixStats 0.14.2, parallel 3.2.2, stats4 3.2.2, tools 3.2.2, zlibbioc 1.16.0

References

[1] Thomas Kuilman, Arno Velds, Kristel Kemper, Marco Ranzani, Lorenzo Bombardelli, Marlous Hoogstraat, Ekaterina Nevedomskaya, Guotai Xu, Julian de Ruiter, Martijn Lolkema, Bauke Ylstra, Jos Jonkers, Sven Rottenberg, Lodewyk Wessels, David Adams, Daniel Peeper, and Oscar Krijgsman. CopywriteR: DNA copy number detection from off-target sequence data. *Genome Biology*, 16(1):49, 2015.

[2] Adam B Olshen, E S Venkatraman, Robert Lucito, and Michael Wigler. Circular binary segmentation for the analysis of array-based DNA copy number data. *Biostatistics*, 5(4):557–572, October 2004.