Bioconductor NGScopy: User's Guide (1.16.0)

A fast and robust algorithm to detect copy number variations by "restriction-imposed windowing" in next generation sequencing

Xiaobei Zhao*[1]

[1]Lineberger Comprehensive Cancer Center, University of North Carolina at Chapel Hill

Modified: 2015-10-07 Compiled: 2018-10-30

You may find the latest version of *NGScopy* and this documentation at,
Latest stable release: http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
Latest devel release: http://www.bioconductor.org/packages/devel/bioc/html/NGScopy.html

Keywords: Sequencing, Copy Number Variation, DNASeq, Targeted Panel Sequencing, Whole Exome Sequencing, Whole Genome Sequencing, Parallel Processing

Contents

*Lineberger Comprehensive Cancer Center, University of North Carolina at Chapel Hill, 450 West Dr, Chapel Hill, NC 27599, USA. xiaobei@binf.ku.dk

1 Introduction

Copy number variation (CNV) is a segment of DNA (> 50 basepair, bp) that has unbalanced number of copies (additions or deletions) with comparison to a control genome [3, 4, 7]. CNVs from 50 bp to 1 kilobase (kb) are also considered as larger indels. CNVs are widely spread throughout the genome, cumulatively accounting for more than one tens of the human genomic DNA and encompassing a large number of our genes [10, 11]. Genomic studies have provided insights into the role of CNVs in health and disease (reviewed in [1, 9, 4]), unveiling the contribution of CNVs in disease pathogenesis.

For over a decade, microarray-based comparative genomic hybridization (arrayCGH or aCGH) and single nucleotide polymorphism microarrays (SNP array) have been *de facto* standard technologies to detect genomic loci subject to CNVs until the emergence of next-generation sequencing (NGS) technologies providing high-resolution DNA sequence data for CNV analysis.

There are three common modes of DNA sequencing: whole genome sequencing (WGS), whole exome sequencing (WES) and targeted panel sequencing (TPS). TPS, as a cost-effective solution, has been widely applied to simultaneously profile cancer-related genes, for instance to find somatic mutations. However, CNV analysis by TPS data has been progressing slowly due to the sparse and inhomogeneous nature of the targeted capture reaction in TPS than in WGS or WES. To address these unique properties, *NGScopy* provides a "restriction-imposed windowing" approach to generate balanced number of reads per window [14], enabling a robust CNV detection in TPS, WES and WGS.

1.1 Requirement and scope

This version of User's Guide introduces the functionality of *NGScopy* by case studies. *NGScopy* requires a pair of samples in *BAM* (`.bam`) files to produce relative copy number ratios (CNRs) between a case and a control samples. In cancer research, the case is typically a tumor sample and the control is usually a matched or pooled normal sample, preferably under the same target enrichment protocol as the case.

Major functionality includes,

- Making windows by the control sample
- Counting reads per window in the control sample
- Counting reads per window in the case sample
- Computing relative CNRs between the case and the control
- Segmentation
- Visualization

The above functions have been intensively tested, and we plan to develop and incorporate more related functionality for CNV analysis. Currently, the *BAM* file parser is integrated by using the *R* (*CRAN*) package `rbamtools` [5] and the segmentation is integrated by using the *R* (*CRAN*) package `changepoint` [6]. Users can also retrieve and modify the produced CNRs and try one of many other available segmentation algorithms, such as *BioHMM* [8].

1.2 Package installation

```
> ## install NGScopy
> if (!requireNamespace("BiocManager", quietly=TRUE))
+     install.packages("BiocManager")
> BiocManager::install("NGScopy")

> ## install NGScopyData for example data sets used in this guide
> BiocManager::install("NGScopyData")
```

1.3 A quick start

A typical *NGScopy* analysis uses a pair of normal/tumor samples to detect the relative CNRs (in log_2). It assumes there are two libraries of DNA sequencing read alignments in *BAM* format, sorted and with index. The total size of each library is also known.

We can either perform the *NGScopy* analysis using a single processor (`pcThreads=1` as below) or parallelize a *NGScopy* call across chromosomes or regions by setting `pcThreads` larger than 1 and up to the appropriate processor/memory limit of the system.

```
> ## Load R libraries
> require("NGScopy")
> require("NGScopyData")
```

```
> ## Create an instance of `NGScopy' class
> obj <- NGScopy$new(
+   outFpre="ngscopy-case1",         # specified directory for output
+   inFpathN=tps_N8.chr6()$bamFpath, # normal sample: tps_90.chr6.sort.bam
+   inFpathT=tps_90.chr6()$bamFpath, # tumor sample:  tps_N8.chr6.sort.bam
+   libsizeN=5777087,                # the library size of the normal sample
+   libsizeT=4624267,                # the library size of the tumor sample
+   mindepth=20,                     # the minimal depth of reads per window
+   minsize=20000,                   # the minimal size of a window
+   regions=NULL,                    # the regions, set to NULL for the entire genome
+   segtype="mean.cusum",            # the type of segmentation
+   dsN=1,                           # the downsampling factor of the normal
+   dsT=1,                           # the downsampling factor of the tumor
+   pcThreads=1                      # the number of processors for computing
+   )

> ## Compute the relative CNRs and save it
> ## A data.frame will be saved to file `ngscopy_cn.txt' in the output directory
> obj$write_cn()

> ## Compute the segmentation and save it
> ## A data.frame will be saved to file `ngscopy_segm.txt' in the output directory
> obj$write_segm()

> ## Plot he relative CNRs with the segmentation
> ## Figure(s) will be saved to file `ngscopy_out.pdf' in the output directory
> obj$plot_out()
```

In the above code, we use the data provided in *Bioconductor NGScopyData* package [13]. We first create an instance of `NGScopy` class, an *R* reference class [2]. Calling the method `write_cn` automatically calls the method `proc_normal`, `proc_tumor`, `calc_cn`, `proc_cn` and `write_cn` in a row to perform window making, read counting in both samples, CNR computing and results processing and saves it in a tab separated file. Calling the method `write_segm` automatically calls the method `calc_segm`, `proc_segm` and `write_segm` in a row to perform segmentation and results processing and saves it in a tab separated file. Finnaly we visualize the results and save it in a pdf file.

This small piece of code provides a compact solution. You may also refer to a functional equivalent step-by-step approach described below in "Case study I".

2 Case study I: copy number detection of a case (tumor) sample compared to a control (normal) sample

This section provides a step-by-step guide to using *NGScopy* for copy number detection of a tumor sample compared to a pooled normal sample, similar steps for a matched normal. A complete source code of this case study is in **Appendix** A.1 on page 26.

We will begin with a case study using the DNA sequencing data, *BAM* files of TPS reads mapped to the human chromossome 6 (hg19), provided in *Bioconductor NGScopyData* package [13]. With limited space in the data package (*NGScopyData*), each of these samples is a 10 percent random subsample drawn from the original sequencing data [14]. A later comparison and discussion of this 10% subsample with the original full data set reveals an ability of *NGScopy* to capture similar chromosome-wide CNV patterns (**Appendix** B on page 38).

We are presenting the CNV detection of a human lung tumor sample (`tps_90,chr6`) against a pooled normal sample (`tps_N8,chr6`). For fast compiling of this vignette, we limit our analysis to a subset region of each *BAM* file: $chr6:41000001-81000000$, $20Mb$ upstream/downstream of the centromere. To do this, we assign an interval to the parameter `regions` which follows the BED format of zero-based, half-open intervals, *i.e.* $(start, end]$.

For normalization purpose, we also need to assign the library sizes (`libsizeN` and `libsizeT`). The library size of one sample is usually the total number reads of all chromosomes throughout the entire genome in the *BAM* file. Because our example *BAM* files only contain reads from $chr6$, the library sizes are computed according to the sequecning reads across all chromosomes of each sample in advance.

There are two windowing parameters `mindepth` and `minsize`, indicating the minimal depth and size required to build a window. We can tune these two parameters according to the coverage characteristics of the samples. Generally, sparser libraries (*e.g.* TPS) require a larger `minsize` while denser libraries (*e.g.* WES and WGS) can have a smaller `minsize`.

We can also adjust the downsampling factor (an integer no less than 1). By setting the downsampling factor to n, we randomly sample $1/n$ of the reads in the sample. For samples with very high depth of coverage, setting a larger downsampling factor would help speed up, however, with possible loss of resolution (**Appendix** B on page 38).

2.1 Create an instance of `NGScopy` class

```
> require(NGScopy)
> require(NGScopyData)
> obj <- NGScopy$new(
+   outFpre="ngscopy-case1",         # specified directory for output
+   inFpathN=tps_N8.chr6()$bamFpath, # normal sample: tps_90.chr6.sort.bam
+   inFpathT=tps_90.chr6()$bamFpath, # tumor sample:  tps_N8.chr6.sort.bam
+   libsizeN=5777087,                # the library size of the normal sample
+   libsizeT=4624267,                # the library size of the tumor sample
+   mindepth=20,                     # the minimal depth of reads per window
+   minsize=20000,                   # the minimal size of a window
+   regions=read_regions("chr6 41000000 81000000"),
+                                    # the regions, set to NULL for the entire genome
+   segtype="mean.norm",             # the type of segmentation
+   dsN=1,                           # the downsampling factor of the normal
+   dsT=1,                           # the downsampling factor of the tumor
+   pcThreads=1                      # the number of processors for computing
+   )
```

```
> obj$show()                          # print the instance

inFpathN: /home/biocbuild/bbs-3.8-bioc/R/library/NGScopyData/extdata/tps_N8.chr6.sort.bam
inFpathT: /home/biocbuild/bbs-3.8-bioc/R/library/NGScopyData/extdata/tps_90.chr6.sort.bam
outFpre: ngscopy-case1
libsizeN: 5777087
libsizeT: 4624267
mindepth: 20
minsize: 20000
regions:
chr6 41000000 81000000
segtype: c("mean.norm")
dsN: 1
dsT: 1
auto.save: FALSE
auto.load: FALSE
```

2.2 Review the input

```
> ## Get the input files
> obj$get_inFpathN()

[1] "/home/biocbuild/bbs-3.8-bioc/R/library/NGScopyData/extdata/tps_N8.chr6.sort.bam"

> obj$get_inFpathT()

[1] "/home/biocbuild/bbs-3.8-bioc/R/library/NGScopyData/extdata/tps_90.chr6.sort.bam"

> ## Get the library sizes
> obj$get_libsizeN()

[1] 5777087

> obj$get_libsizeT()

[1] 4624267

> ## Get the windowing parameters
> obj$get_mindepth()

[1] 20

> obj$get_minsize()

[1] 20000
```

```
> ## Get the regions
> head(obj$get_regions())


   chr   start     end
1 chr6 4.1e+07 8.1e+07

> ## Get the segmentation type(s)
> head(obj$get_segmtype())


[1] "mean.norm"

> ## Get the downsampling factors
> obj$get_dsN()


[1] 1


> obj$get_dsT()


[1] 1

> ## Get the number of processors
> obj$get_pcThreads()


[1] 1

> ## Get the chromosome names of the reference genome
> obj$get_refname()


 [1] "chr1"  "chr2"  "chr3"  "chr4"  "chr5"  "chr6"  "chr7"  "chr8"  "chr9"
[10] "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16" "chr17" "chr18"
[19] "chr19" "chr20" "chr21" "chr22" "chrX"  "chrY"  "chrM"

> ## Get the chromosome lengths of the reference genome
> obj$get_reflength()


     chr1      chr2      chr3      chr4      chr5      chr6      chr7      chr8
249250621 243199373 198022430 191154276 180915260 171115067 159138663 146364022
     chr9     chr10     chr11     chr12     chr13     chr14     chr15     chr16
141213431 135534747 135006516 133851895 115169878 107349540 102531392  90354753
    chr17     chr18     chr19     chr20     chr21     chr22      chrX      chrY
 81195210  78077248  59128983  63025520  48129895  51304566 155270560  59373566
     chrM
    16571
```

2.3 Process reads in the control (normal) sample (Make windows as well)

```
> obj$proc_normal()                 # this may take a while
```

2.4 Process reads in the case (tumor) sample

```
> obj$proc_tumor()                  # this may take a while
```

2.5 Compute/Process the relative copy number ratios and save it

```
> ## A data.frame will be saved to file `ngscopy_cn.txt' in the output directory
> obj$calc_cn()
> obj$proc_cn()
> obj$write_cn()
```

2.6 Compute/Process the segmentation and save it

```
> ## A data.frame will be saved to file `ngscopy_segm.txt' in the output directory
> obj$calc_segm()
> obj$proc_segm()
> obj$write_segm()
```

2.7 Save the output for later reference

```
> ## The NGScopy output is saved as a ngscopy_out.RData file in the output directory
> obj$saveme()
```

2.8 Load and review the output

```
> ## Load the output
> ## (optional if the previous steps have completed in the same R session)
> obj$loadme()

> ## Get the output directory
> obj$get_outFpre()


[1] "ngscopy-case1"

> ## Get the windows
> head(obj$get_windows())


   chr    start      end
1 chr6 41000000 41031350
2 chr6 41031350 41101829
3 chr6 41101829 41144951
4 chr6 41144951 41171264
5 chr6 41171264 41204537
6 chr6 41204537 41238046
```

```
> ## Get the window sizes
> head(obj$get_size())


[1] 31350 70479 43122 26313 33273 33509

> ## Get the window positions (midpoints of the windows)
> head(obj$get_pos())


[1] 41015675 41066590 41123390 41158108 41187901 41221292

> ## Get the number of reads per window in the normal
> head(obj$get_depthN())


[1] 20 20 20 20 20 20

> ## Get the number of reads per window in the tumor
> head(obj$get_depthT())


[1] 23 22 19 20 17 18

> ## Get the data.frame of copy number calling
> data.cn <- obj$get_data.cn()


MoreArgs.cn: List of 2
 $ pseudocount: num 1
 $ logr       : logi TRUE
out$cn:List of 3
 $ cnr        : num [1:567] 0.5138 0.4524 0.2507 0.3211 0.0987 ...
 $ pseudocount: num 1
 $ logr       : logi TRUE


> head(data.cn)


   chr    start      end  size      pos depthN depthT       cnr
1 chr6 41000000 41031350 31350 41015675     20     23 0.5137626
2 chr6 41031350 41101829 70479 41066590     20     22 0.4523621
3 chr6 41101829 41144951 43122 41123390     20     19 0.2507282
4 chr6 41144951 41171264 26313 41158108     20     20 0.3211175
5 chr6 41171264 41204537 33273 41187901     20     17 0.0987251
6 chr6 41204537 41238046 33509 41221292     20     18 0.1767276
```

```
> data.segm <- obj$get_data.segm()


MoreArgs.segm:  list()
out$seg:List of 1
 $ mean.norm:List of 1
  ..$ chr6:Formal class 'cpt' [package "changepoint"] with 12 slots
  .. .. ..@ data.set : Time-Series [1:567] from 1 to 567: 0.5138 0.4524 0.2507 0.3211 0.0987 ...
  .. .. ..@ cpttype  : chr "mean"
  .. .. ..@ method   : chr "PELT"
  .. .. ..@ test.stat: chr "Normal"
  .. .. ..@ pen.type : chr "SIC"
  .. .. ..@ pen.value: num 12.7
  .. .. ..@ minseglen: num 1
  .. .. ..@ cpts     : int [1:2] 303 567
  .. .. ..@ ncpts.max: num Inf
  .. .. ..@ param.est:List of 1
  .. .. .. ..$ mean: num [1:2] 0.0354 -0.4652
  .. .. ..@ date     : chr "Thu Jul 26 18:20:50 2018"
  .. .. ..@ version  : chr "2.2.2"


> head(data.segm)


   chr win.from win.to    start      end        mean   segtype
1 chr6        1    303 41000000 57590917  0.03536811 mean.norm
2 chr6      304    567 57590917 81000000 -0.46519165 mean.norm
```

2.9 Visualize the output

```
> ## A figure will be saved to file `ngscopy_cn.pdf' in the output directory
> obj$plot_out(ylim=c(-3,3))       # reset `ylim' to NULL to allow full-scale display
```

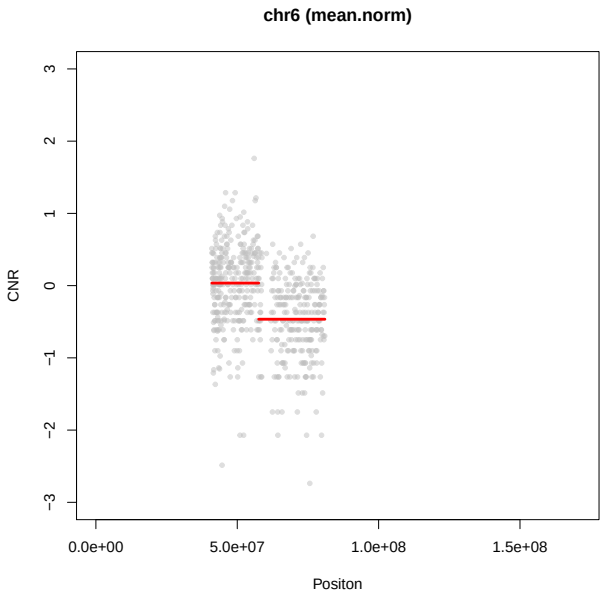


Figure 1: Graphical output of "Case study I", showing CNVs of $chr6 : 41000001 - 81000000$ of sample `tps_90` against `tps_N8`.

3 Case Study I(b): Analyzing the entire chromosome

So far, we have conducted the copy nunber detection on a subregion of a chromosome ($chr6 : 41000001 - 81000000$). Now we would like to apply such analysis on the entrie chromosome by setting the `regions` accordingly, using the same data as in "Case Study I".

```
> obj <- NGScopy$new(
+   outFpre="ngscopy-case1b",        # specified directory for output
+   inFpathN=tps_N8.chr6()$bamFpath, # normal sample: tps_90.chr6.sort.bam
+   inFpathT=tps_90.chr6()$bamFpath, # tumor sample:  tps_N8.chr6.sort.bam
+   libsizeN=5777087,                # the library size of the normal sample
+   libsizeT=4624267,                # the library size of the tumor sample
+   mindepth=20,                     # the minimal depth of reads per window
+   minsize=20000,                   # the minimal size of a window
+   regions=read_regions("chr6 0 171115067"),
+                                    # the regions, set to NULL for the entire genome
+   segtype="mean.norm",             # the type of segmentation
+   dsN=1,                           # the downsampling factor of the normal
+   dsT=1,                           # the downsampling factor of the tumor
+   pcThreads=1                      # the number of processors for computing
+   )

> ## Show the regions
> obj$get_regions()


   chr start       end
1 chr6     0 171115067
```

We keep the rest codes intact and re-run them in the same order as in "Case Study I".

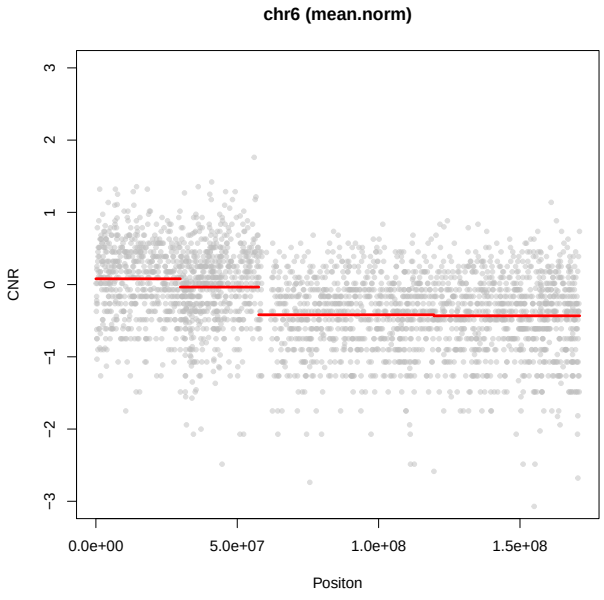


Figure 2: Graphical output of "Case study I(b)", showing CNVs of the entire *chr*6 of the sample `tps_90` against the sample `tps_N8`.

4 Case Study I(c): Choosing types of segmentation

The *NGScopy* package has integrated several external segmentation algorithms ([6]) to facilitate change-point detection.

Currently there are 4 types of segmentation available, which can be obtained by the following code:

```
> NGScopy::parse_segmtype()

[1] "mean.norm"    "meanvar.norm" "mean.cusum"   "var.css"
```

Given a type of segmentation, we can get help of the algorithm by the code below, for instance,

```
> ## NGScopy::help_segmtype("mean.norm")
```

We can set the parameter `segtype` to either one or multiple values (separated by ",") from the above.

```
> require(NGScopy)
> obj <- NGScopy$new()

> ## Set segtype with multiple values
> obj$set_segmtype("mean.norm,meanvar.norm,mean.cusum,var.css")

> ## Get segtype
> obj$get_segmtype()

[1] "mean.norm"    "meanvar.norm" "mean.cusum"   "var.css"
```

Using the same data and the same rest setting as in "Case Study I(b)", we have the chromosome segmented in 4 ways,

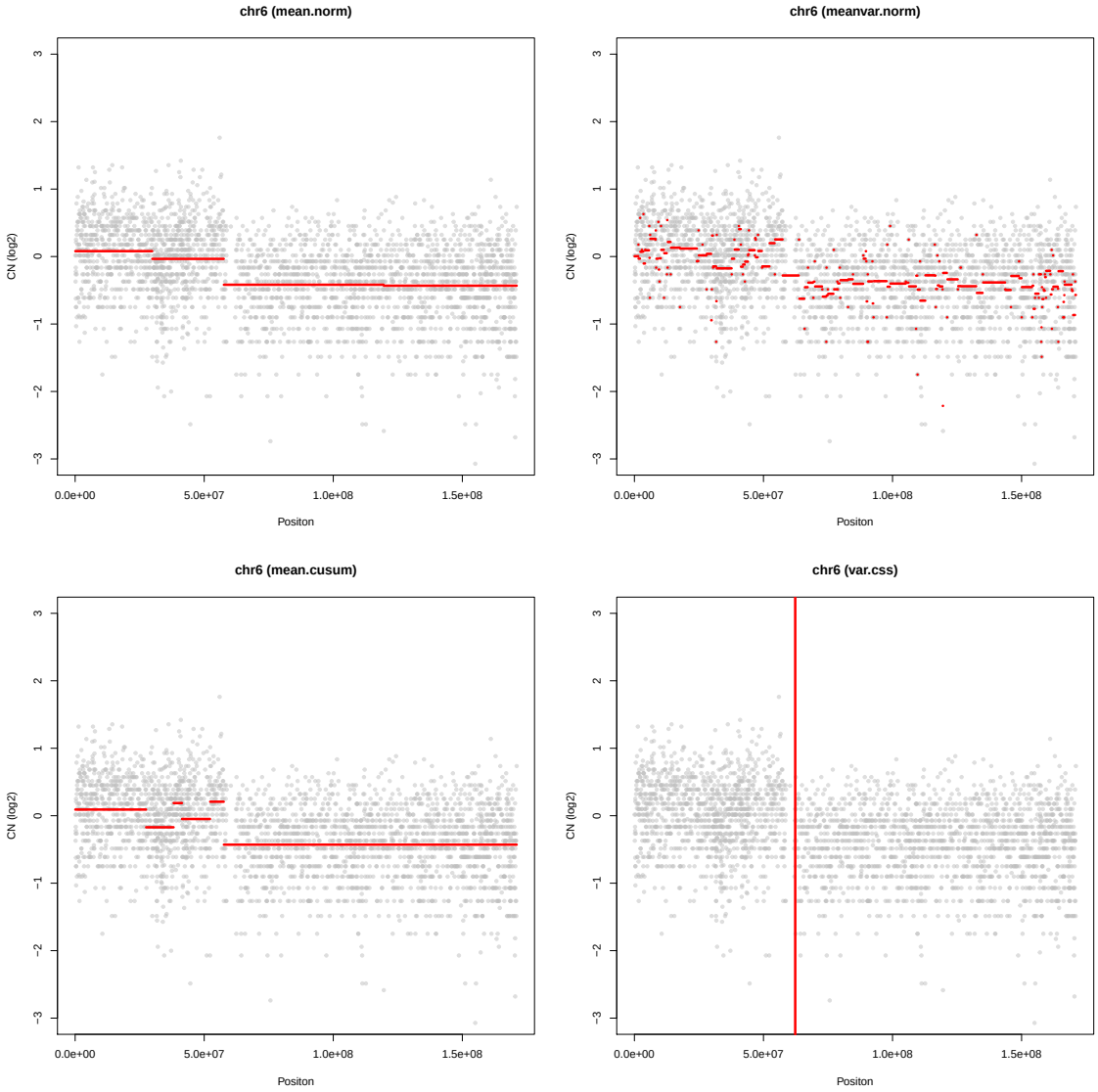


Figure 3: Graphical output of "Case study I(c)", showing CNVs of the entire *chr*6 of the sample `tps_90` against the sample `tps_N8` using 4 types of segmentation algorithms.

5 Case study II: copy number detection of multiple case (tumor) samples compared to a common control (normal) sample

This section provides a step-by-step guide to using *NGScopy* for copy number detection of multiple tumor samples compared to a common normal sample, for instance, a pooled normal sample. We are about analyzing two tumor samples (`tps_90,chr6` and `tps_27,chr6`) against one common pooled normal (`tps_N8,chr6`), provided in *Bioconductor NGScopyData* package [12], as described in "Case study I". A complete source code of this case study is in **Appendix** A.2 on page 29.

5.1 Process the common control (normal) sample

We first analyze the normal sample, make the windows, count the reads and save the output to use with each of the tumor samples in **Section** 5.2 and **Section** 5.3.

```
> require(NGScopy)
> require(NGScopyData)

> ## Create an instance of `NGScopy' class
> obj <- NGScopy$new(pcThreads=1)

> ## Set the normal sample
> obj$set_normal(tps_N8.chr6()$bamFpath)

> ## Set the regions
> regions <- read_regions("chr6 41000000 81000000")
> obj$set_regions(regions)

> ## Set the library size of the normal
> obj$set_libsizeN(5777087)

> ## Specify a directory for output
> ## It will be saved as "ngscopy_normal.RData" in this directory
> obj$set_outFpre("ngscopy-case2/tps_N8")

> ## Show the input
> obj$show()


inFpathN: /home/biocbuild/bbs-3.8-bioc/R/library/NGScopyData/extdata/tps_N8.chr6.sort.bam
outFpre: ngscopy-case2/tps_N8
libsizeN: 5777087
regions:
chr6 41000000 81000000
auto.save: FALSE
auto.load: FALSE

> ## Make windows and count reads in the normal
> obj$proc_normal()

> ## Save the output of the normal for later usage
> obj$save_normal()
```

5.2 Process a case (tumor) sample

Now we create a new *NGScopy* instance, load the result of the normal sample previously saved in **Section** 5.1 and set the parameters for the tumor sample to detect CNVs.

```
> ## Create an instance of `NGScopy' class
> obj1 <- NGScopy$new(pcThreads=1)

> ## Load the previously saved output of the normal
> obj1$load_normal("ngscopy-case2/tps_N8")

> ## Set a tumor sample (ID: tps_90) and specify a directory for output
> obj1$set_tumor(tps_90.chr6()$bamFpath)
> obj1$set_outFpre("ngscopy-case2/tps_90")

> ## Set the library size of the tumor
> obj1$set_libsizeT(4624267)

> ## Show the input
> obj1$show()


inFpathN: /home/biocbuild/bbs-3.8-bioc/R/library/NGScopyData/extdata/tps_N8.chr6.sort.bam
inFpathT: /home/biocbuild/bbs-3.8-bioc/R/library/NGScopyData/extdata/tps_90.chr6.sort.bam
outFpre: ngscopy-case2/tps_90
libsizeN: 5777087
libsizeT: 4624267
mindepth: 20
minsize: 20000
regions:
chr6 41000000 81000000
segtype: c("mean.norm")
dsN: 1
auto.save: FALSE
auto.load: FALSE

> ## Process the tumor
> obj1$proc_tumor()

> ## Process the copy number
> obj1$proc_cn()


MoreArgs.cn: List of 2
 $ pseudocount: num 1
 $ logr       : logi TRUE
out$cn:List of 3
 $ cnr        : num [1:567] 0.5138 0.4524 0.2507 0.3211 0.0987 ...
 $ pseudocount: num 1
 $ logr       : logi TRUE
```

```
> ## Process the segmentation
> obj1$proc_segm()


MoreArgs.segm:  list()
out$seg:List of 1
 $ mean.norm:List of 1
  ..$ chr6:Formal class 'cpt' [package "changepoint"] with 12 slots
  .. .. ..@ data.set : Time-Series [1:567] from 1 to 567: 0.5138 0.4524 0.2507 0.3211 0.0987 ...
  .. .. ..@ cpttype  : chr "mean"
  .. .. ..@ method   : chr "PELT"
  .. .. ..@ test.stat: chr "Normal"
  .. .. ..@ pen.type : chr "SIC"
  .. .. ..@ pen.value: num 12.7
  .. .. ..@ minseglen: num 1
  .. .. ..@ cpts     : int [1:2] 303 567
  .. .. ..@ ncpts.max: num Inf
  .. .. ..@ param.est:List of 1
  .. .. .. ..$ mean: num [1:2] 0.0354 -0.4652
  .. .. ..@ date     : chr "Thu Jul 26 18:20:50 2018"
  .. .. ..@ version  : chr "2.2.2"

> ## Plot
> obj1$plot_out(ylim=c(-3,3))
```

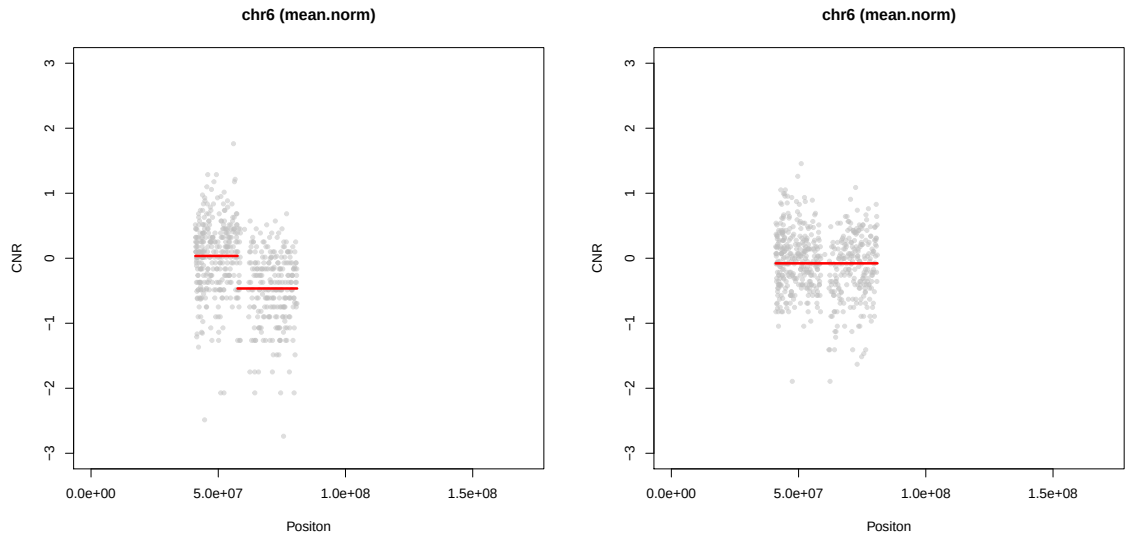
5.3 Process a second case (tumor) sample

Then we process a second tumor sample via similar steps as in **Section** 5.2.

```
> ## Create another instance of `NGScopy' class
> obj2 <- NGScopy$new(pcThreads=1)

> ## Load the previously saved output of the normal
> obj2$load_normal("ngscopy-case2/tps_N8")

> ## Set a tumor sample (ID: tps_27) and specify a directory for output
> obj2$set_tumor(tps_27.chr6()$bamFpath)
> obj2$set_outFpre("ngscopy-case2/tps_27")

> ## Set the library size of the tumor
> obj2$set_libsizeT(10220498)
```

```
> ## Show the input
> obj2$show()


inFpathN: /home/biocbuild/bbs-3.8-bioc/R/library/NGScopyData/extdata/tps_N8.chr6.sort.bam
inFpathT: /home/biocbuild/bbs-3.8-bioc/R/library/NGScopyData/extdata/tps_27.chr6.sort.bam
outFpre: ngscopy-case2/tps_27
libsizeN: 5777087
libsizeT: 10220498
mindepth: 20
minsize: 20000
regions:
chr6 41000000 81000000
segtype: c("mean.norm")
dsN: 1
auto.save: FALSE
auto.load: FALSE

> ## Process the tumor
> obj2$proc_tumor()

> ## Process the copy number
> obj2$proc_cn()


MoreArgs.cn: List of 2
 $ pseudocount: num 1
 $ logr       : logi TRUE
out$cn:List of 3
 $ cnr        : num [1:567] 0.177 0.07 -0.823 0.54 0.37 ...
 $ pseudocount: num 1
 $ logr       : logi TRUE

> ## Process the segmentation
> obj2$proc_segm()


MoreArgs.segm:  list()
out$seg:List of 1
 $ mean.norm:List of 1
  ..$ chr6:Formal class 'cpt' [package "changepoint"] with 12 slots
  .. .. ..@ data.set : Time-Series [1:567] from 1 to 567: 0.177 0.07 -0.823 0.54 0.37 ...
  .. .. ..@ cpttype  : chr "mean"
  .. .. ..@ method   : chr "PELT"
  .. .. ..@ test.stat: chr "Normal"
  .. .. ..@ pen.type : chr "SIC"
  .. .. ..@ pen.value: num 12.7
  .. .. ..@ minseglen: num 1
  .. .. ..@ cpts     : int 567
  .. .. ..@ ncpts.max: num Inf
  .. .. ..@ param.est:List of 1
  .. .. .. ..$ mean: num -0.0783
  .. .. ..@ date     : chr "Thu Jul 26 18:20:50 2018"
  .. .. ..@ version  : chr "2.2.2"

> ## Plot
> obj2$plot_out(ylim=c(-3,3))
```

5.4 Visualize the output



(a) ID: `tps_90` (b) ID: `tps_27`

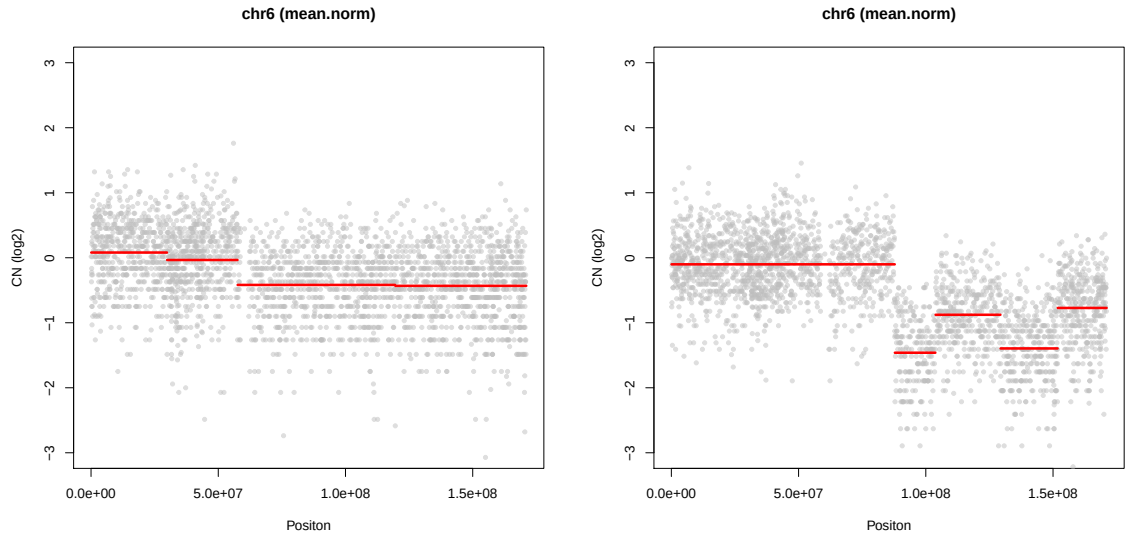
Figure 4: Graphical output of "Case Study II" on a subregion of chromosome 6, showing CNVs of the sample `tps_90` and the sample `tps_27` against a common normal sample `tps_N8`.

6 Case Study II(b): Analyzing the entire chromosome

Similarly as we did in "Case study II", we apply the analysis to the entrie chromosome instead of the subregion ($chr6 : 41000001 - 81000000$) by setting the parameter `regions` appropriately, as described in "Case study I(b)".

```
> ## Create a new instance of `NGScopy' class
> obj <- NGScopy$new(pcThreads=1)

> ## Set the normal sample
> obj$set_normal(tps_N8.chr6()$bamFpath)

> ## Read the regions from an external file
> regions <- Xmisc::get_extdata('NGScopy','hg19_chr6_0_171115067.txt')

> ## ## Or from a text input as we did before
> ## regions <- read_regions("chr6 0 171115067")

> ## Set the regions
> obj$set_regions(regions)

> ## Show the regions
> obj$get_regions()

   chr start       end
1 chr6     0 171115067
```

We keep the rest codes intact and re-run them in the same order as in "Case study II" (**Section** 5.1, 5.2 and 5.3).



(a) ID: `tps_90` (b) ID: `tps_27`

Figure 5: Graphical output of "Case Study II" on entire chromosome 6, showing CNVs of the sample `tps_90` and the sample `tps_27` against a common normal sample `tps_N8`.

7 Run *NGScopy* from command line

An executable R script, `ngscopy`, is placed at the `bin` subdirectory under the top level package directory. A complete source code of this executable is in **Appendix** A.3 on page 31.

In this section, we introduce and run this executable R script from a command line on Unix or a Unix-like operating system.

7.1 Get the path of the executable R script

Given the *NGScopy* package is installed, we can obtain the path of the executable *R* script at the *R* prompt:

```
> system.file('bin', 'ngscopy', package='NGScopy', mustWork=TRUE)

[1] "/tmp/RtmpI56Yln/Rinst66455f2a5a5b/NGScopy/bin/ngscopy"

> ## Or,
> Xmisc::get_executable('NGScopy')

[1] "/tmp/RtmpI56Yln/Rinst66455f2a5a5b/NGScopy/bin/ngscopy"
```

Alternatively, we can extract this path at Unix-like command-line interface (CLI):

```
ngscopyCmd=$(Rscript -e "cat(system.file('bin','ngscopy',package='NGScopy', mustWork=TRUE))")
echo ${ngscopyCmd}

## Or,
ngscopyCmd=$(Rscript -e "cat(Xmisc::get_executable('NGScopy'))")
echo ${ngscopyCmd}
```

7.2 Get help

```
${ngscopyCmd} -h

## Or,
${ngscopyCmd} --help
```

This will print the help page at the console,

```
Usage:
    ngscopy [options]

Description:
    ngscopy scans a pair of input BAM files to detect relative copy numb-
er variants.

Options:
    h          logical   Print the help page. NULL

    help       logical   Print the help page. NULL

    inFpathN   character The file path to the control (normal) sample. NULL

    inFpathT   character The file path to the case (tumor) sample. NULL

    outFpre    character The file path of the directory for output. NULL

    libsizeN   numeric   The library size of the control (normal) sample. NULL

    libsizeT   numeric   The library size of the case (tumor) sample. NULL

    mindepth   numeric   The minimal depth of reads per window. [ 20 ]

    minsize    numeric   The minimal size of a window. [ 20000 ]

    regions    character The regions, a text string or a file path indica-
                         ting the regions with three columns (chr/start/e-
                         nd) and without column names. [ NULL ]

    segtype    character The type of segmentation. One of c("mean.norm","
                         meanvar.norm","mean.cusum","var.css"). Multiple
                         values are allowed and separated by ",". [ mean.norm ]

    dsN        integer   The downsampling factor of the control (normal)
                         sample. [ 1 ]

    dsT        integer   The downsampling factor of the test (tumor) samp-
                         le. [ 1 ]

    pcThreads integer    The number of processors performing the parallel
                          computing. [ 1 ]

    auto.save logical    Whether to save (any completed results) automati-
                         cally. [ FALSE ]

    auto.load logical    Whether to load (any previously completed result-
                         s) automatically. [ FALSE ]
```

7.3 An example

Let's run the command line version of the Case Study I(b).

```
ngscopyCmd=$(Rscript -e "cat(Xmisc::get_executable('NGScopy'))")

## a normal sample, given the NGScopyData package is installed
inFpathN=$(Rscript -e "cat(Xmisc::get_extdata('NGScopyData','tps_N8.chr6.sort.bam'))")

## a tumor sample, given the NGScopyData pack is installed
inFpathT=$(Rscript -e "cat(Xmisc::get_extdata('NGScopyData','tps_90.chr6.sort.bam'))")

## set the regions, given the NGScopy package is installed
regions=$(Rscript -e "cat(Xmisc::get_extdata('NGScopy','hg19_chr6_0_171115067.txt'))")

time ${ngscopyCmd} --inFpathN=${inFpathN} --inFpathT=${inFpathT} --outFpre="ngscopy-case1b-cmdline" \
--libsizeN=5777087 --libsizeT=4624267 --regions=${regions} --dsN=1 --dsT=1 --pcThreads=1
```

A complete source code and the output of this example is in **Appendix** A.4 on page 34.

Acknowledgement

I appreciate Dr. Stergios J Moschos and Dr. D Neil Hayes for giving me advice and full support on developing the *Bioconductor NGScopy* package. I would also like to thank Vonn Walter for valuable discussion, Michele C Hayward and Ashley H Salazar for help in annotation of clinical samples and sample related data, and Xiaoying Yin for conducting DNA sequencing experiments.

I thank the Bioconductor team, particularly Martin T. Morgan for reviewing the package and providing helpful comments.

This project is supported by the University Cancer Research Fund.

A The source code

A.1 The source code for the case study I

ngscopy-case1.R

```
#!/usr/bin/env Rscript

## ************************************************************************
## This is the R script for running `Case Study I' in `Bioconductor NGScopy'.
##
##
##
## (c) Xiaobei Zhao
##
## Mon Aug 11 13:44:26 EDT 2014 -0400 (Week 32)
##
## Reference:
## Bioconductor NGScopy
## http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
## Case study I: copy number detection of a tumor sample compared to
##               a pooled normal sample
##
## Runme:
## Rscript ngscopy-case1.R &> ngscopy-case1.out
##
## ************************************************************************

require(methods)
require(NGScopy)
require(NGScopyData)

## ------------------------------------------------------------------------
## Create an instance of NGScopy class
## ------------------------------------------------------------------------
obj <- NGScopy$new(
  outFpre="ngscopy-case1",         # specified directory for output
  inFpathN=tps_N8.chr6()$bamFpath, # normal sample: tps_90.chr6.sort.bam
  inFpathT=tps_90.chr6()$bamFpath, # tumor sample:  tps_N8.chr6.sort.bam
  libsizeN=5777087,                # the library size of the normal sample
  libsizeT=4624267,                # the library size of the tumor sample
  mindepth=20,                     # the minimal depth of reads per window
  minsize=20000,                   # the minimal size of a window
  regions=read_regions("chr6 41000000 81000000"),
                                   # the regions, set to NULL for the entire genome
  segtype="mean.norm",             # the type of segmentation
  dsN=1,                           # the downsampling factor of the normal
  dsT=1,                           # the downsampling factor of the tumor
  pcThreads=1                      # the number of processors for computing
  )

obj$show()                         # print the instance

## ------------------------------------------------------------------------
## Review the input
## ------------------------------------------------------------------------

## Get the input files
obj$get_inFpathN()
obj$get_inFpathT()

## Get the library sizes
obj$get_libsizeN()
obj$get_libsizeT()

## Get the windowing parameters
obj$get_mindepth()
obj$get_minsize()

## Get the regions
```

```
head(obj$get_regions())

## Get the segmentation type(s)
head(obj$get_segmtype())

## Get the down sampling factors
obj$get_dsN()
obj$get_dsT()

## Get the number of processors
obj$get_pcThreads()

## Get the chromosome names of the reference genome
obj$get_refname()

## Get the chromosome lengths of the reference genome
obj$get_reflength()


## ------------------------------------------------------------------------
## Process reads in the control (normal) sample (Make windows as well)
## ------------------------------------------------------------------------
obj$proc_normal()                 # this may take a while

## ------------------------------------------------------------------------
## Process reads in the case (tumor) sample
## ------------------------------------------------------------------------
obj$proc_tumor()                  # this may take a while

## ------------------------------------------------------------------------
## Compute/Process the relative copy number ratios and save it
## ------------------------------------------------------------------------

## A data.frame will be saved to file `ngscopy_cn.txt' in the output directory
obj$calc_cn()
obj$proc_cn()
obj$write_cn()

## ------------------------------------------------------------------------
## Compute/Process the segmentation and save it
## ------------------------------------------------------------------------

## A data.frame will be saved to file `ngscopy_segm.txt' in the output directory
obj$calc_segm()
obj$proc_segm()
obj$write_segm()

## ------------------------------------------------------------------------
## Save the output for later reference
## ------------------------------------------------------------------------
## The NGScopy output is saved as a ngscopy_out.RData file in the output directory
obj$saveme()

## ------------------------------------------------------------------------
## Load and review the output
## ------------------------------------------------------------------------

## Load the output
## (optional if the previous steps have completed in the same R session)
obj$loadme()

## Get the output directory
obj$get_outFpre()

## Get the windows
head(obj$get_windows())

```

```
## Get the window sizes
head(obj$get_size())

## Get the window positions (midpoints of the windows)
head(obj$get_pos())

## Get the number of reads per window in the normal
head(obj$get_depthN())

## Get the number of reads per window in the tumor
head(obj$get_depthT())

## Get the copy number
str(obj$get_cn())

## Get the segmentation
str(obj$get_segm())

## Get the data.frame of copy number calling
data.cn <- obj$get_data.cn()
head(data.cn)

## Get the data.frame of segmentation calling
data.segm <- obj$get_data.segm()
head(data.segm)


## ------------------------------------------------------------------------
## Visualize the output
## ------------------------------------------------------------------------
## A figure will be saved to file `ngscopy_cn.pdf' in the output directory
obj$plot_out(ylim=c(-3,3))       # reset `ylim' to allow full-scale display
```

A.2 The source code for the case study II

ngscopy-case2.R

```
#!/usr/bin/env Rscript

## ************************************************************************
## This is the R script for running `Case Study II' in `Bioconductor NGScopy'.
##
##
##
## (c) Xiaobei Zhao
##
## Mon Aug 11 13:45:24 EDT 2014 -0400 (Week 32)
##
## Reference:
## Bioconductor NGScopy
## http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
## Case study II: copy number detection of multiple tumor samples compared to
##                a common pooled normal sample
##
## Runme:
## Rscript ngscopy-case2.R &> ngscopy-case2.out
##
## ************************************************************************

require(methods)
require(NGScopy)
require(NGScopyData)

## ------------------------------------------------------------------------
## Process the normal
## ------------------------------------------------------------------------

## Create an instance of `NGScopy' class
obj <- NGScopy$new(pcThreads=1)

## Set the normal sample
obj$set_normal(tps_N8.chr6()$bamFpath)

## Set the regions
regions <- read_regions("chr6 41000000 81000000")
obj$set_regions(regions)

## Set the library size of the normal
obj$set_libsizeN(5777087)

## Specify a directory for output
## It will be saved as "ngscopy_normal.RData" in this directory
obj$set_outFpre("ngscopy-case2/tps_N8")

## Show the input
obj$show()

## Make windows and count reads in the normal
obj$proc_normal()

## Save the output of the normal for later usage
obj$save_normal()

## ------------------------------------------------------------------------
## Process a tumor
## ------------------------------------------------------------------------

## Create an instance of `NGScopy' class
obj1 <- NGScopy$new(pcThreads=1)

## Load the previously saved output of the normal
```

```
obj1$load_normal("ngscopy-case2/tps_N8")

## Set a tumor sample (ID: tps_90) and specify a directory for output
obj1$set_tumor(tps_90.chr6()$bamFpath)
obj1$set_outFpre("ngscopy-case2/tps_90")

## Set the library size of the tumor
obj1$set_libsizeT(4624267)

## Show the input
obj1$show()

## Process the tumor
obj1$proc_tumor()

## Process the copy number
obj1$proc_cn()

## Process the segmentation
obj1$proc_segm()

## Plot
obj1$plot_out(ylim=c(-3,3))

## ------------------------------------------------------------------------
## Process a second tumor
## ------------------------------------------------------------------------

## Create another instance of `NGScopy' class
obj2 <- NGScopy$new(pcThreads=1)

## Load the previously saved output of the normal
obj2$load_normal("ngscopy-case2/tps_N8")

## Set a tumor sample (ID: tps_27) and specify a directory for output
obj2$set_tumor(tps_27.chr6()$bamFpath)
obj2$set_outFpre("ngscopy-case2/tps_27")

## Set the library size of the tumor
obj2$set_libsizeT(10220498)

## Show the input
obj2$show()

## Process the tumor
obj2$proc_tumor()

## Process the copy number
obj2$proc_cn()

## Process the segmentation
obj2$proc_segm()

## Plot
obj2$plot_out(ylim=c(-3,3))
```

A.3 The command-line executable *R* script

ngscopy

```
#!/usr/bin/env Rscript

## ************************************************************************
## This is an executable R script for running `Bioconductor NGScopy'
## at commond line.
##
##
##
## (c) Xiaobei Zhao
##
## Sat Jun 28 21:19:32 EDT 2014 -0400 (Week 25)
##
##
## Reference:
## Bioconductor NGScopy
## http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
##
## Get help
## ./ngscopy -h
## $(Rscript -e "cat(Xmisc::get_executable('NGScopy'))") -h
## ************************************************************************


require(methods)
require(Xmisc)

PARSEME <- function(){
  parser <- ArgumentParser$new()

  parser$add_usage('ngscopy [options]')
  parser$add_description(
    'ngscopy scans a pair of input BAM files to detect relative copy number variants.')

  parser$add_argument(
    '--h',type='logical',
    action='store_true',
    help='Print the help page'
  )

  parser$add_argument(
    '--help',type='logical',
    action='store_true',
    help='Print the help page'
  )

  parser$add_argument(
    '--inFpathN',type='character',
    help='The file path to the control (normal) sample'
  )

  parser$add_argument(
    '--inFpathT',type='character',
    help='The file path to the case (tumor) sample'
  )
  parser$add_argument(
    '--outFpre',type='character',
    help='The file path of the directory for output'
  )
  parser$add_argument(
    '--libsizeN',type='numeric',
    help='The library size of the control (normal) sample'
  )
  parser$add_argument(
    '--libsizeT',type='numeric',
```

```
    help='The library size of the case (tumor) sample'
  )
  parser$add_argument(
    '--mindepth',type='numeric',
    default=20,
    help='The minimal depth of reads per window'
  )
  parser$add_argument(
    '--minsize',type='numeric',
    default=20000,
    help='The minimal size of a window'
  )

  parser$add_argument(
    '--regions',type='character',
    default=NULL,
    help='The regions, a text string or a file path indicating the regions
with three columns (chr/start/end) and without column names.'
  )

  parser$add_argument(
    '--segtype',type='character',
    default="mean.norm",
    help='The type of segmentation. One of c(
"mean.norm","meanvar.norm","mean.cusum","var.css"
). Multiple values are allowed and separated by ",".'
  )

  parser$add_argument(
    '--dsN',type='integer',
    default=1,
    help='The downsampling factor of the control (normal) sample'
  )

  parser$add_argument(
    '--dsT',type='integer',
    default=1,
    help='The downsampling factor of the test (tumor) sample'
  )

  parser$add_argument(
    '--pcThreads',type='integer',
    default=1,
    help='The number of processors performing the parallel computing'
  )

  parser$add_argument(
    '--auto.save',type='logical',
    default=FALSE,
    help='Whether to save (any completed results) automatically'
  )

  parser$add_argument(
    '--auto.load',type='logical',
    default=FALSE,
    help='Whether to load (any previously completed results) automatically'
  )
  return(parser)
}


main <- function(){
  parser <- PARSEME()
  parser$helpme()

  require(NGScopy)

```

```
  if (length(regions)){
    regions <- read_regions(regions)
  }

  obj <- NGScopy$new(
    outFpre=outFpre,
    inFpathN=inFpathN,
    inFpathT=inFpathT,
    libsizeN=libsizeN,
    libsizeT=libsizeT,
    mindepth=mindepth,
    minsize=minsize,
    regions=regions,
    segtype=segtype,
    dsN=dsN,
    dsT=dsT,
    pcThreads=pcThreads,
    auto.save=auto.save,
    auto.load=auto.load
  )

  obj$write_cn()
  obj$write_segm()
  ## obj$plot_out()
  obj$plot_out(ylim=c(-3,3))

  invisible()
}


main()
```

A.4 The command-line example Bash (Unix shell) script and the output

ngscopy-case1b-cmdline.sh

```
#!/usr/bin/env bash


## ************************************************************************
## This is a commond line example to run bin/ngscopy in `Bioconductor NGScopy'.
##
##
## (c) Xiaobei Zhao
##
## v0.99.1
## Mon Aug 11 09:57:53 EDT 2014 -0400 (Week 32)
## [2014-08-11 09:59:52 EDT] add regions, dsN, dsT
##
## v0.99.0
## Sat Jun 28 21:19:32 EDT 2014 -0400 (Week 25)
##
##
## Reference:
## Bioconductor NGScopy
## http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
##
## Runme:
## bash ngscopy-case1b-cmdline.sh &> ngscopy-case1b-cmdline.out
##
## ************************************************************************


## ------------------------------------------------------------------------
## The path of the executable, for instance, ${R_LIBS}/NGScopy/bin/ngscopy
## A portable solution is to extract this path by `system.file' in R
## We call such funciton from the command line below.
## ------------------------------------------------------------------------

## ngscopyCmd=$(Rscript -e "cat(system.file('bin','ngscopy',
## package='NGScopy', mustWork=TRUE))")
ngscopyCmd=$(Rscript -e "cat(Xmisc::get_executable('NGScopy'))")
echo ${ngscopyCmd}


## ------------------------------------------------------------------------
## Get help
## ------------------------------------------------------------------------

## ${ngscopyCmd} -h
## ${ngscopyCmd} --help


## ------------------------------------------------------------------------
## An example
## This is a command line version of Case Study I from NGScopy User's Guide.
## See http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
## ------------------------------------------------------------------------

## a normal sample, given the NGScopyData package is installed
## inFpathN=$(Rscript -e "cat(system.file('extdata','tps_N8.chr6.sort.bam',
## package='NGScopyData', mustWork=TRUE))")
inFpathN=$(Rscript -e "cat(Xmisc::get_extdata('NGScopyData','tps_N8.chr6.sort.bam'))")

## echo ${inFpathN}
## ls -l ${inFpathN}

## a tumor sample, given the NGScopyData package is installed
## inFpathT=$(Rscript -e "cat(system.file('extdata','tps_90.chr6.sort.bam',
## package='NGScopyData', mustWork=TRUE))")
```

```
inFpathT=$(Rscript -e "cat(Xmisc::get_extdata('NGScopyData','tps_90.chr6.sort.bam'))")
## echo ${inFpathT}
## ls -l ${inFpathT}

## set pre-computed libsizes based on the original bam files of all chromosomes
libsizeN=5777087
libsizeT=4624267

## set the regions, given the NGScopy package is installed
## regions=$(Rscript -e "cat(system.file('extdata','hg19_chr6_0_171115067.txt',
## package='NGScopy', mustWork=TRUE))")
regions=$(Rscript -e "cat(Xmisc::get_extdata('NGScopy','hg19_chr6_0_171115067.txt'))")
## echo ${regions}

## set downsampling factor; we do not downsample here by setting them to 1.
dsN=1
dsT=1

## set a directory for output
outFpre="ngscopy-case1b-cmdline"

## Run NGScopy given arguments and time it
time ${ngscopyCmd} --inFpathN=${inFpathN} --inFpathT=${inFpathT} --outFpre="${outFpre}" \
--libsizeN=${libsizeN} --libsizeT=${libsizeT} --regions=${regions} \
--dsN=${dsN} --dsT=${dsT} --pcThreads=1
```

The output,

ngscopy-case1b-cmdline.out

```
/home/xiaobei/supplemental2/usr/local/lib64/R/library/NGScopy/bin/ngscopy
Loading required package: methods
Loading required package: Xmisc
Loading required package: NGScopy
auto.save: FALSE
auto.load: FALSE
20140815 10:02:42 EDT | set_normal | inFpathN
inFpathN: /home/xiaobei/supplemental2/usr/local/lib64/R/library/NGScopyData/extdata/tps_N8.chr6.sort.bam

20140815 10:02:42 EDT | set_tumor | inFpathT
inFpathT: /home/xiaobei/supplemental2/usr/local/lib64/R/library/NGScopyData/extdata/tps_90.chr6.sort.bam

20140815 10:02:42 EDT | set_outFpre
outFpre: ngscopy-case1b-cmdline

20140815 10:02:42 EDT | set_libsize | libsizeN
libsizeN: 5777087

20140815 10:02:42 EDT | set_libsize | libsizeT
libsizeT: 4624267

20140815 10:02:42 EDT | set_normal | mindepth
mindepth: 20

20140815 10:02:42 EDT | set_normal | minsize
minsize: 20000

20140815 10:02:42 EDT | set_regions
20140815 10:02:42 EDT | .proc_ref | process reference genome (in bam header of the normal sample)
20140815 10:02:42 EDT | .trim_regions | trim regsions if exceeding the reference
20140815 10:02:42 EDT | .sort_regions | sort regions by reference
regions:
chr6 0 171115067

20140815 10:02:42 EDT | set_segmtype
segtype: c("mean.norm")
```

```
20140815 10:02:42 EDT | set_ds | dsN
dsN: 1

20140815 10:02:42 EDT | set_ds | dsT
dsT: 1

20140815 10:02:42 EDT | set_pcThreads
pcThreads: 1

20140815 10:02:42 EDT | write_cn
20140815 10:02:42 EDT | proc_cn
20140815 10:02:42 EDT | proc_normal | this may take a while depending on the size of your library.
20140815 10:02:42 EDT | (PID: 54971) Processing coords (refid, start, end): 5, 0, 171115067
20140815 10:04:02 EDT | Processed all 1 regions.
20140815 10:04:02 EDT | proc_tumor | this may take a while depending on the size of your library.
20140815 10:05:12 EDT | Processed all 2808 windows.
20140815 10:05:12 EDT | calc_cn
20140815 10:05:12 EDT | set_MoreArgs.cn
MoreArgs.cn not specified, assuming list(pseudocount=1,logr=TRUE).
MoreArgs.cn:
List of 2
 $ pseudocount: num 1
 $ logr       : logi TRUE


20140815 10:05:12 EDT | calc_cn | done
out$cn:
List of 3
 $ cnr        : num [1:2808] 0.0163 -0.3708 -0.2638 -0.1643 -0.2232 ...
 $ pseudocount: num 1
 $ logr       : logi TRUE



File saved:
outFpath="ngscopy-case1b-cmdline/ngscopy_cn.txt"

20140815 10:05:12 EDT | write_segm
20140815 10:05:12 EDT | proc_segm
20140815 10:05:12 EDT | calc_segm
20140815 10:05:12 EDT | set_MoreArgs.segm
MoreArgs.segm not specified, assuming list().
MoreArgs.segm:
 list()


20140815 10:05:12 EDT | calc_segm | done
out$seg:
List of 1
 $ mean.norm:List of 1
  ..$ chr6:Formal class 'cpt' [package "changepoint"] with 10 slots
  .. .. ..@ data.set : Time-Series [1:2808] from 1 to 2808: 0.0163 -0.3708 -0.2638 -0.1643 -0.2232 ...
  .. .. ..@ cpttype  : chr "mean"
  .. .. ..@ method   : chr "PELT"
  .. .. ..@ test.stat: chr "Normal"
  .. .. ..@ pen.type : chr "SIC"
  .. .. ..@ pen.value: num 7.94
  .. .. ..@ cpts     : int [1:6] 550 551 1141 1951 1953 2808
  .. .. ..@ ncpts.max: num Inf
  .. .. ..@ param.est:List of 1
  .. .. .. ..$ mean: num [1:6] 0.0801 -4.0712 -0.0345 -0.418 -3.3285 ...
  .. .. ..@ date     : chr "Wed Jul 23 19:59:37 2014"


File saved:
```

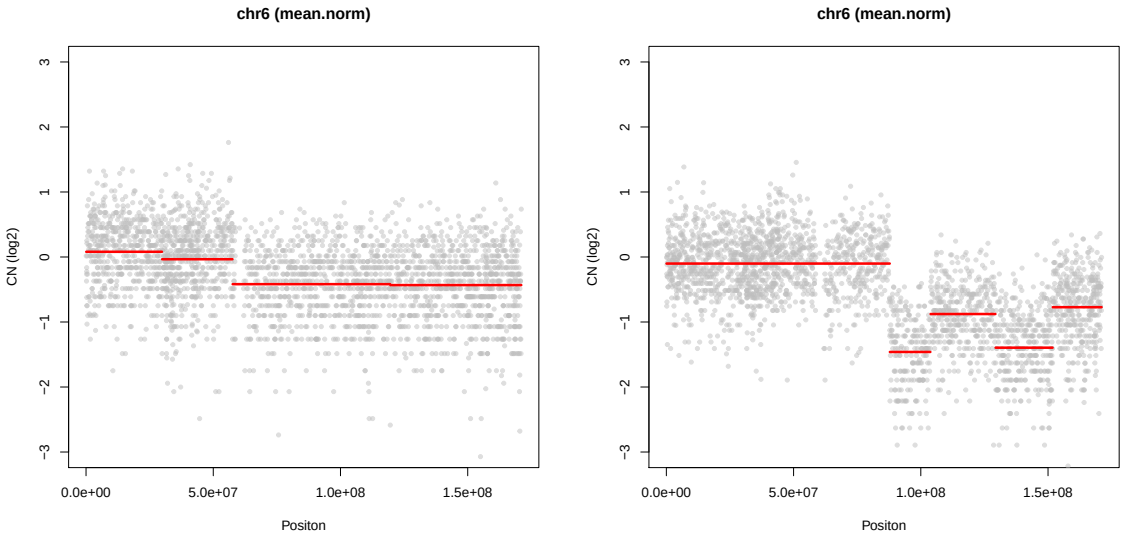
```
outFpath="ngscopy-case1b-cmdline/ngscopy_segm.txt"

20140815 10:05:12 EDT | plot_out
chr=chr6, segtype=mean.norm

File saved:
pdfFpath="ngscopy-case1b-cmdline/ngscopy_out.pdf"


real        2m48.398s
user        2m47.147s
sys        0m0.298s
```

B Comparison with full-scale NGS data

Here we compared the copy number calling in the follow two scenarios:

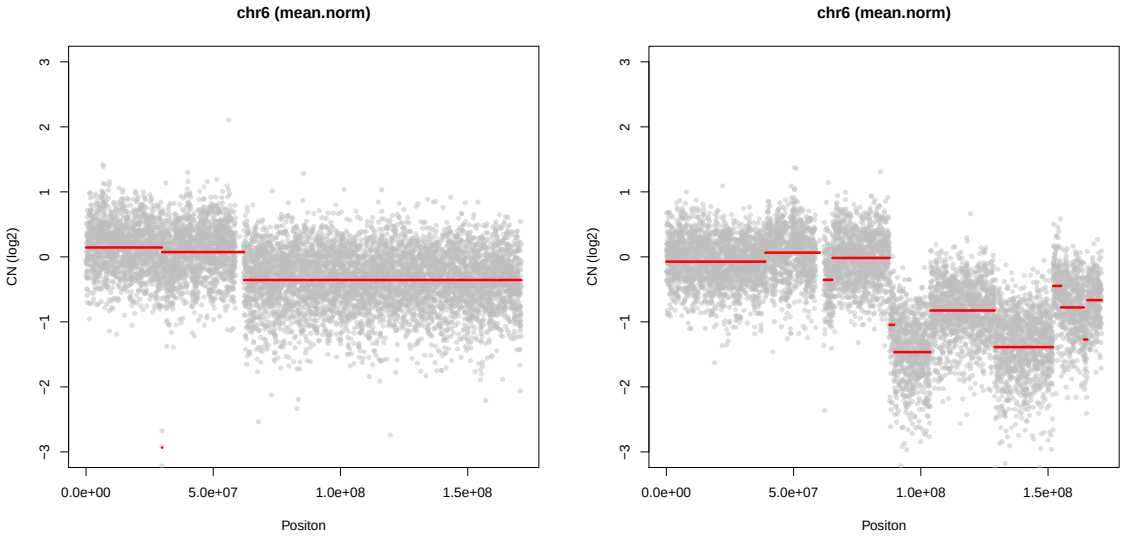
1. Using the 10% subsample (**Figure** 6 (a) and (b)).
2. Using the original data without any downsampling (**Figure** 6 (c) and (d)).

The first scenario identified CNVs with 10% of the original data. It also captured highly similar characteristics in the chromosome-wide view compared with the other scenario. Obviously, downsampling can reduce the computation time though may entail some loss of detail and precision. Whether to down-sample the data depends on the sequencing coverage properties (depth, broadness and distributions of reads) as well as the purpose of the analysis. It is always a good practice to make prelimilary tests with downsampled data.



(a) ID: `tps_90(10%subsample)`

(b) ID: `tps_27(10%subsample)`



(c) ID: `tps_90(originalsample)`

(d) ID: `tps_27(originalsample)`

Figure 6: Graphical output for comparison of NGScopy calling between the 10% subsample [(a) and (b)] and the original sample without downsampling [(c) and (d)].

References

[1] Jacques S. Beckmann, Xavier Estivill, and Stylianos E. Antonarakis. Copy number variants and genetic traits: closer to the resolution of phenotypic to genotypic variability. *Nat Rev Genet*, 8(8):639–646, Aug 2007.

[2] John Chambers. *ReferenceClasses: Objects With Fields Treated by Reference (OOP-style)*. CRAN, 2014. http://stat.ethz.ch/R-manual/R-devel/library/methods/html/refClass.html.

[3] Lars Feuk, Andrew R Carson, and Stephen W Scherer. Structural variation in the human genome. *Nat Rev Genet*, 7(2):85–97, Feb 2006.

[4] Santhosh Girirajan, Catarina D. Campbell, and Evan E. Eichler. Human copy number variation and complex genetic disease. *Annu Rev Genet*, 45:203–226, 2011.

[5] Wolfgang Kaisers. *rbamtools: Reading, manipulation and writing BAM (Binary alignment) files*, 2014. R package version 2.6.0.

[6] Rebecca Killick, Idris Eckley, and Kaylea Haynes. *changepoint: An R package for changepoint analysis*, 2014. R package version 1.1.5.

[7] Jeffrey R. MacDonald, Robert Ziman, Ryan K C. Yuen, Lars Feuk, and Stephen W. Scherer. The database of genomic variants: a curated collection of structural variation in the human genome. *Nucleic Acids Res*, 42(Database issue):D986–D992, Jan 2014.

[8] J. C. Marioni, N. P. Thorne, and S. Tavaré. Biohmm: a heterogeneous hidden markov model for segmenting array cgh data. *Bioinformatics*, 22(9):1144–1146, May 2006.

[9] Steven A. McCarroll and David M. Altshuler. Copy-number variation and association studies of human disease. *Nat Genet*, 39(7 Suppl):S37–S42, Jul 2007.

[10] Richard Redon, Shumpei Ishikawa, Karen R Fitch, Lars Feuk, George H Perry, T Daniel Andrews, Heike Fiegler, Michael H Shapero, Andrew R Carson, Wenwei Chen, et al. Global variation in copy number in the human genome. *nature*, 444(7118):444–454, 2006.

[11] Travis I Zack, Steven E Schumacher, Scott L Carter, Andrew D Cherniack, Gordon Saksena, Barbara Tabak, Michael S Lawrence, Cheng-Zhong Zhang, Jeremiah Wala, Craig H Mermel, et al. Pan-cancer patterns of somatic copy number alteration. *Nature genetics*, 45(10):1134–1140, 2013.

[12] Xiaobei Zhao. *NGScopy: Detection of copy number variations in next generation sequencing*, 2014. Bioconductor/R package.

[13] Xiaobei Zhao. *NGScopyData: Subset of BAM files of human tumor and pooled normal sequencing data (Zhao et al. 2014) for the NGScopy package*, 2014. Bioconductor/R package.

[14] Xiaobei Zhao, Anyou Wang, ..., D Neil Hayes, and Stergios J Moschos. Targeted sequencing in non-small cell lung cancer (nsclc) using the university of north carolina (unc) sequencing assay captures most previously described genetic aberrations in nsclc. *In preparation*, 2014.