

# Bioconductor *NGScopy*: User’s Guide (1.6.0)

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

Xiaobei Zhao<sup>\*1</sup>

<sup>1</sup>Lineberger Comprehensive Cancer Center, University of North Carolina at Chapel Hill

Modified: 2015-10-07    Compiled: 2016-5-3

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

### 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 [?, ?, ?]. 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 [?, ?]. Genomic studies have provided insights into the role of CNVs in health and disease (reviewed in [?, ?, ?]), 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 [?], enabling a robust CNV detection in TPS, WES and WGS.

---

<sup>\*</sup>Lineberger Comprehensive Cancer Center, University of North Carolina at Chapel Hill, 450 West Dr, Chapel Hill, NC 27599, USA. [xiaobei@binf.ku.dk](mailto:xiaobei@binf.ku.dk)

## 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* [?] and the segmentation is integrated by using the *R* (*CRAN*) package *changepoint* [?]. Users can also retrieve and modify the produced CNRs and try one of many other available segmentation algorithms, such as *BioHMM* [?].

## 1.2 Package installation

```
> ## install NGScopy
> source("http://bioconductor.org/biocLite.R")
> biocLite("NGScopy")

> ## install NGScopyData for example data sets used in this guide
> biocLite("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 the 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 [?]. We first create an instance of *NGScopy* class, an *R* reference class [?]. 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. Finally 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 ??** on page ??.

We will begin with a case study using the DNA sequencing data, *BAM* files of TPS reads mapped to the human chromosome 6 (hg19), provided in *Bioconductor NGScopyData* package [?]. With limited space in the data package (*NGScopyData*), each of these samples is a 10 percent random subsample drawn from the original sequencing data [?]. 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 ??** on page ??).

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 sequencing 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 ??** on page ??).

### 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: /Library/Frameworks/R.framework/Versions/3.3/Resources/library/NGScopyData/extdata/tps_N8.chr6.sort.b
inFpathT: /Library/Frameworks/R.framework/Versions/3.3/Resources/library/NGScopyData/extdata/tps_90.chr6.sort.b
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] "/Library/Frameworks/R.framework/Versions/3.3/Resources/library/NGScopyData/extdata/tps_N8.chr6.sort.bam"

> obj$get_inFpathT()

[1] "/Library/Frameworks/R.framework/Versions/3.3/Resources/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_seg.txt' in the output directory
> obj$calc_seg()
> obj$proc_seg()
> obj$write_seg()
```

## 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 "Fri Mar 25 11:09:02 2016"
  .. .. ..@ version   : chr "2.2.1"

> 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

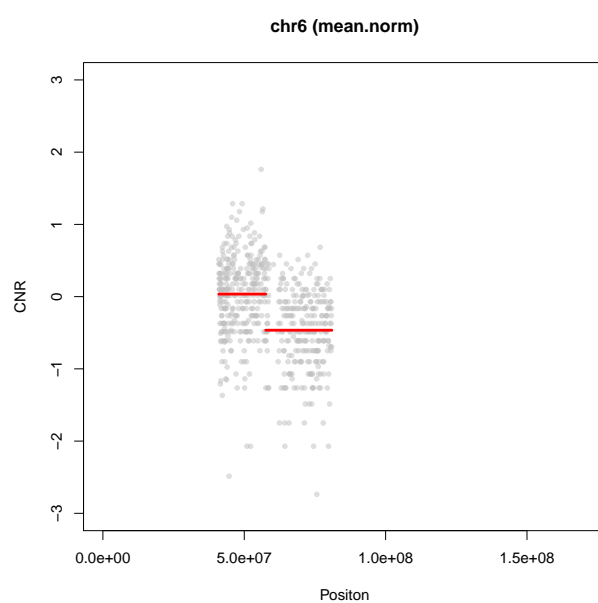
```

## 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

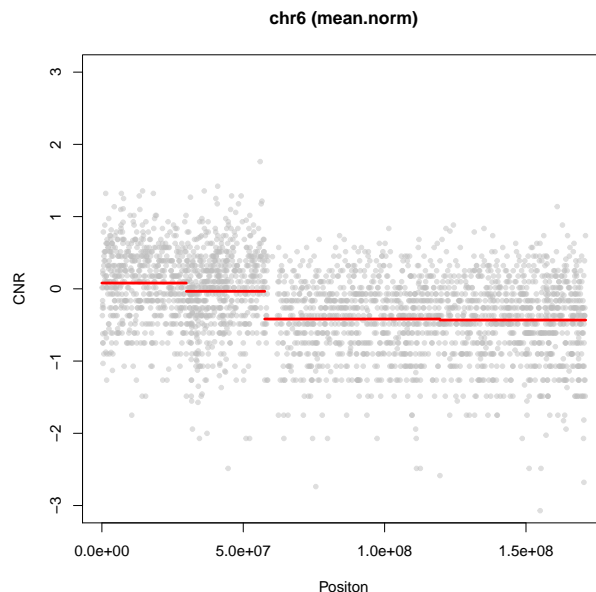
So far, we have conducted the copy number detection on a subregion of a chromosome (*chr6* : 41000001 – 81000000). Now we would like to apply such analysis on the entire chromosome by setting the **regions** accordingly, using the same data as in “Case Study I”.

```
> 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 *chr6* 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 ([?]) to facilitate change-point detection.

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

```
> NGScopy::parse_segmtypes()

[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_segmtypes("mean.norm")
```

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

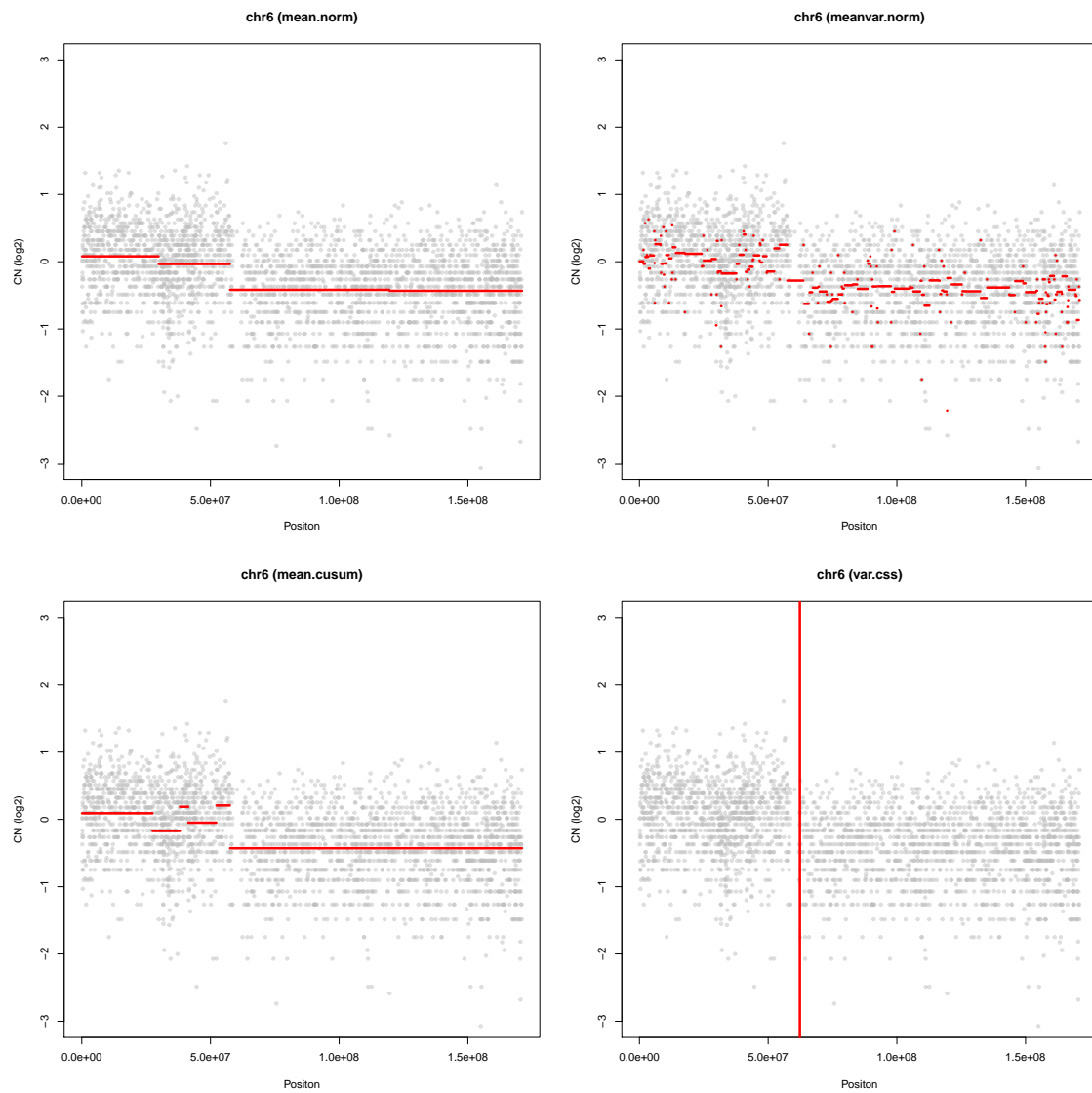
```
> require(NGScopy)
> obj <- NGScopy$new()

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

> ## Get segtype
> obj$get_segmtypes()

[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 *chr6* 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 [?], as described in “Case study I”. A complete source code of this case study is in **Appendix ??** on page ??.

### 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 ??** and **Section ??**.

```
> 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: /Library/Frameworks/R.framework/Versions/3.3/Resources/library/NGScopyData/extdata/tps_N8.chr6.sort.b
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 ??** 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: /Library/Frameworks/R.framework/Versions/3.3/Resources/library/NGScopyData/extdata/tps_N8.chr6.sort.b
inFpathT: /Library/Frameworks/R.framework/Versions/3.3/Resources/library/NGScopyData/extdata/tps_90.chr6.sort.b
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_seg()

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 "Fri Mar 25 11:09:02 2016"
  .. .. ..@ version    : chr "2.2.1"

> ## 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 ??**.

```

> ## 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: /Library/Frameworks/R.framework/Versions/3.3/Resources/library/NGScopyData/extdata/tps_N8.chr6.sort.b
inFpathT: /Library/Frameworks/R.framework/Versions/3.3/Resources/library/NGScopyData/extdata/tps_27.chr6.sort.b
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

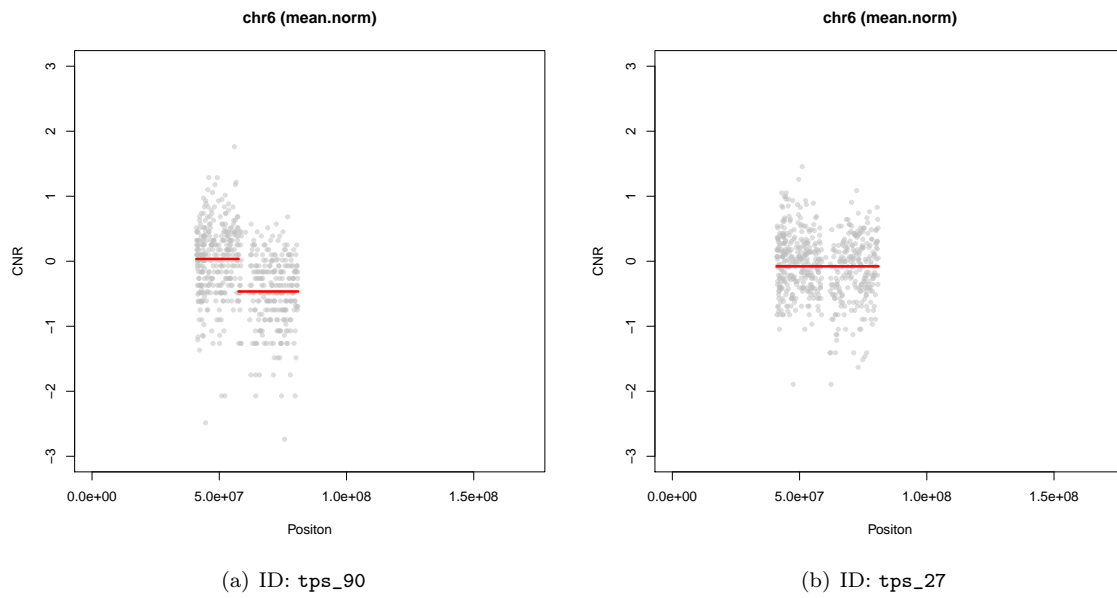
> ## Process the segmentation
> obj2$proc_seg()

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 "Fri Mar 25 11:09:02 2016"
  .. .. ..@ version   : chr "2.2.1"

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

```

## 5.4 Visualize the output



**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 entire 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')

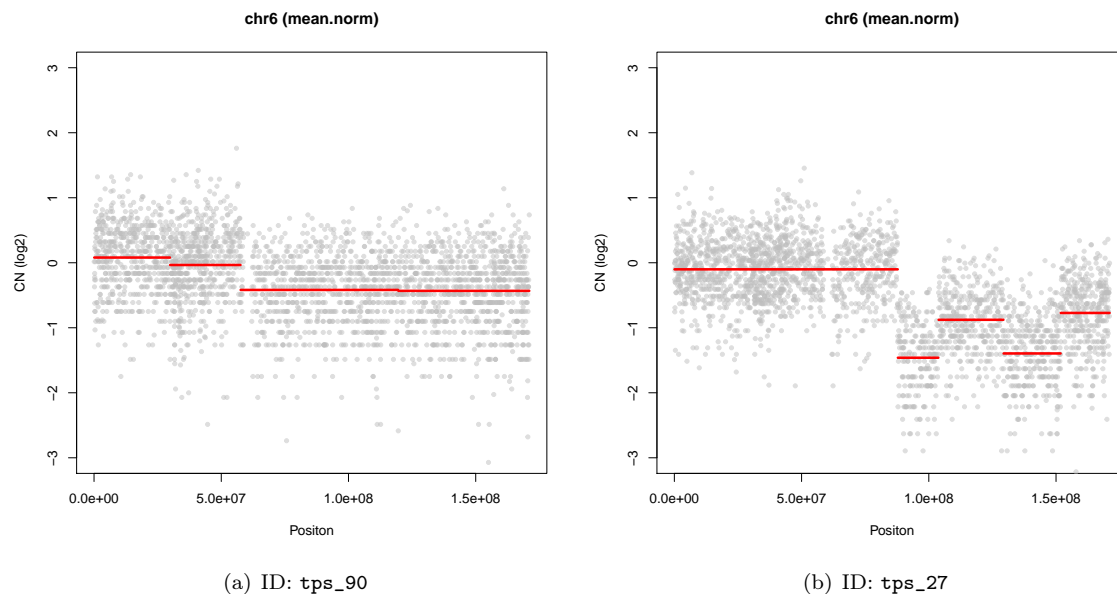
> ## Or from a text input as we did before
> ## regions <- read_regions("chr6 0 171115067")

> ## 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 ??, ?? and ??).



**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 ??** on page ??.

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] "/private/tmp/RtmpHqGPmT/Rinst16c3736beddf7/NGScopy/bin/ngscopy"

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

[1] "/private/tmp/RtmpHqGPmT/Rinst16c3736beddf7/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 number 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 indicating the regions with three columns (chr/start/end) 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) sample. [ 1 ]

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

  auto.save  logical   Whether to save (any completed results) automatically. [ FALSE ]

  auto.load  logical   Whether to load (any previously completed results) 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 ??** on page ??.

## 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

```

1      #!/usr/bin/env Rscript
2
3      ## *****
4      ## This is the R script for running `Case Study I' in `Bioconductor NGScopy'.
5      ##
6      ##
7      ##
8      ## (c) Xiaobei Zhao
9      ##
10     ## Mon Aug 11 13:44:26 EDT 2014 -0400 (Week 32)
11     ##
12     ## Reference:
13     ## Bioconductor NGScopy
14     ## http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
15     ## Case study I: copy number detection of a tumor sample compared to
16     ##               a pooled normal sample
17     ##
18     ## Runme:
19     ## Rscript ngscopy-case1.R &> ngscopy-case1.out
20     ##
21     ## *****
22
23     require(methods)
24     require(NGScopy)
25     require(NGScopyData)
26
27     ## -----
28     ## Create an instance of NGScopy class
29     ## -----
30     obj <- NGScopy$new(
31       outFpre="ngscopy-case1",           # specified directory for output
32       inFpathN=tps_N8.chr6()$bamFpath,   # normal sample: tps_90.chr6.sort.bam
33       inFpathT=tps_90.chr6()$bamFpath,   # tumor sample: tps_N8.chr6.sort.bam
34       libsizeN=5777087,                  # the library size of the normal sample
35       libsizeT=4624267,                  # the library size of the tumor sample
36       mindepth=20,                       # the minimal depth of reads per window
37       minsize=20000,                     # the minimal size of a window
38       regions=read_regions("chr6 41000000 81000000"),
39                                           # the regions, set to NULL for the entire genome
40       segtype="mean.norm",                # the type of segmentation
41       dsN=1,                             # the downsampling factor of the normal
42       dsT=1,                             # the downsampling factor of the tumor
43       pcThreads=1                        # the number of processors for computing
44     )
45
46     obj$show()                          # print the instance
47
48     ## -----
49     ## Review the input
50     ## -----
51
52     ## Get the input files
53     obj$get_inFpathN()
54     obj$get_inFpathT()
55
56     ## Get the library sizes
57     obj$get_libsizeN()
58     obj$get_libsizeT()
59
60     ## Get the windowing parameters
61     obj$get_mindepth()
62     obj$get_minsize()
63
64     ## Get the regions

```

```

65 | head(obj$get_regions())
66 |
67 | ## Get the segmentation type(s)
68 | head(obj$get_segmtype())
69 |
70 | ## Get the down sampling factors
71 | obj$get_dsN()
72 | obj$get_dsT()
73 |
74 | ## Get the number of processors
75 | obj$get_pcThreads()
76 |
77 | ## Get the chromosome names of the reference genome
78 | obj$get_refname()
79 |
80 | ## Get the chromosome lengths of the reference genome
81 | obj$get_reflength()
82 |
83 |
84 | ## -----
85 | ## Process reads in the control (normal) sample (Make windows as well)
86 | ## -----
87 | obj$proc_normal()          # this may take a while
88 |
89 | ## -----
90 | ## Process reads in the case (tumor) sample
91 | ## -----
92 | obj$proc_tumor()          # this may take a while
93 |
94 | ## -----
95 | ## Compute/Process the relative copy number ratios and save it
96 | ## -----
97 |
98 | ## A data.frame will be saved to file `ngscopy_cn.txt' in the output directory
99 | obj$calc_cn()
100 | obj$proc_cn()
101 | obj$write_cn()
102 |
103 | ## -----
104 | ## Compute/Process the segmentation and save it
105 | ## -----
106 |
107 | ## A data.frame will be saved to file `ngscopy_seg.txt' in the output directory
108 | obj$calc_seg()
109 | obj$proc_seg()
110 | obj$write_seg()
111 |
112 | ## -----
113 | ## Save the output for later reference
114 | ## -----
115 | ## The NGScopy output is saved as a ngscopy_out.RData file in the output directory
116 | obj$save()
117 |
118 | ## -----
119 | ## Load and review the output
120 | ## -----
121 |
122 | ## Load the output
123 | ## (optional if the previous steps have completed in the same R session)
124 | obj$loadme()
125 |
126 | ## Get the output directory
127 | obj$get_outFpre()
128 |
129 | ## Get the windows
130 | head(obj$get_windows())
131 |

```

```
132 ## Get the window sizes
133 head(obj$get_size())
134
135 ## Get the window positions (midpoints of the windows)
136 head(obj$get_pos())
137
138 ## Get the number of reads per window in the normal
139 head(obj$get_depthN())
140
141 ## Get the number of reads per window in the tumor
142 head(obj$get_depthT())
143
144 ## Get the copy number
145 str(obj$get_cn())
146
147 ## Get the segmentation
148 str(obj$get_seg())
149
150 ## Get the data.frame of copy number calling
151 data.cn <- obj$get_data.cn()
152 head(data.cn)
153
154 ## Get the data.frame of segmentation calling
155 data.seg <- obj$get_data.seg()
156 head(data.seg)
157
158
159 ## -----
160 ## Visualize the output
161 ## -----
162 ## A figure will be saved to file `ngscopy_cn.pdf' in the output directory
163 obj$plot_out(ylim=c(-3,3))      # reset `ylim' to allow full-scale display
```

## A.2 The source code for the case study II

```

1  #!/usr/bin/env Rscript
2
3  ## *****
4  ## This is the R script for running `Case Study II' in `Bioconductor NGScopy'.
5  ##
6  ##
7  ##
8  ## (c) Xiaobei Zhao
9  ##
10 ## Mon Aug 11 13:45:24 EDT 2014 -0400 (Week 32)
11 ##
12 ## Reference:
13 ## Bioconductor NGScopy
14 ## http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
15 ## Case study II: copy number detection of multiple tumor samples compared to
16 ##               a common pooled normal sample
17 ##
18 ## Runme:
19 ## Rscript ngscopy-case2.R &> ngscopy-case2.out
20 ##
21 ## *****
22
23 require(methods)
24 require(NGScopy)
25 require(NGScopyData)
26
27 ## -----
28 ## Process the normal
29 ## -----
30
31 ## Create an instance of `NGScopy' class
32 obj <- NGScopy$new(pcThreads=1)
33
34 ## Set the normal sample
35 obj$set_normal(tps_N8.chr6())$bamFpath)
36
37 ## Set the regions
38 regions <- read_regions("chr6 41000000 81000000")
39 obj$set_regions(regions)
40
41 ## Set the library size of the normal
42 obj$set_libsizeN(5777087)
43
44 ## Specify a directory for output
45 ## It will be saved as "ngscopy_normal.RData" in this directory
46 obj$set_outFpre("ngscopy-case2/tps_N8")
47
48 ## Show the input
49 obj$show()
50
51 ## Make windows and count reads in the normal
52 obj$proc_normal()
53
54 ## Save the output of the normal for later usage
55 obj$save_normal()
56
57 ## -----
58 ## Process a tumor
59 ## -----
60
61 ## Create an instance of `NGScopy' class
62 obj1 <- NGScopy$new(pcThreads=1)
63
64 ## Load the previously saved output of the normal

```

```
65 obj1$load_normal("ngscopy-case2/tps_N8")
66
67 ## Set a tumor sample (ID: tps_90) and specify a directory for output
68 obj1$set_tumor(tps_90.chr6())$bamFpath)
69 obj1$set_outFpre("ngscopy-case2/tps_90")
70
71 ## Set the library size of the tumor
72 obj1$set_libsizeT(4624267)
73
74 ## Show the input
75 obj1$show()
76
77 ## Process the tumor
78 obj1$proc_tumor()
79
80 ## Process the copy number
81 obj1$proc_cn()
82
83 ## Process the segmentation
84 obj1$proc_segm()
85
86 ## Plot
87 obj1$plot_out(ylim=c(-3,3))
88
89 ## -----
90 ## Process a second tumor
91 ## -----
92
93 ## Create another instance of `NGScopy` class
94 obj2 <- NGScopy$new(pcThreads=1)
95
96 ## Load the previously saved output of the normal
97 obj2$load_normal("ngscopy-case2/tps_N8")
98
99 ## Set a tumor sample (ID: tps_27) and specify a directory for output
100 obj2$set_tumor(tps_27.chr6())$bamFpath)
101 obj2$set_outFpre("ngscopy-case2/tps_27")
102
103 ## Set the library size of the tumor
104 obj2$set_libsizeT(10220498)
105
106 ## Show the input
107 obj2$show()
108
109 ## Process the tumor
110 obj2$proc_tumor()
111
112 ## Process the copy number
113 obj2$proc_cn()
114
115 ## Process the segmentation
116 obj2$proc_segm()
117
118 ## Plot
119 obj2$plot_out(ylim=c(-3,3))
```

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

```

1  #!/usr/bin/env Rscript
2
3  ## *****
4  ## This is an executable R script for running `Bioconductor NGScopy'
5  ## at command line.
6  ##
7  ##
8  ##
9  ## (c) Xiaobei Zhao
10 ##
11 ## Sat Jun 28 21:19:32 EDT 2014 -0400 (Week 25)
12 ##
13 ##
14 ## Reference:
15 ## Bioconductor NGScopy
16 ## http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
17 ##
18 ## Get help
19 ## ./ngscopy -h
20 ## $(Rscript -e "cat(Xmisc::get_executable('NGScopy'))") -h
21 ## *****
22
23
24 require(methods)
25 require(Xmisc)
26
27 PARSEME <- function(){
28   parser <- ArgumentParser$new()
29
30   parser$add_usage('ngscopy [options]')
31   parser$add_description(
32     'ngscopy scans a pair of input BAM files to detect relative copy number variants.')
33
34   parser$add_argument(
35     '--h',type='logical',
36     action='store_true',
37     help='Print the help page'
38   )
39
40   parser$add_argument(
41     '--help',type='logical',
42     action='store_true',
43     help='Print the help page'
44   )
45
46   parser$add_argument(
47     '--inFpathN',type='character',
48     help='The file path to the control (normal) sample'
49   )
50
51   parser$add_argument(
52     '--inFpathT',type='character',
53     help='The file path to the case (tumor) sample'
54   )
55   parser$add_argument(
56     '--outFpre',type='character',
57     help='The file path of the directory for output'
58   )
59   parser$add_argument(
60     '--libsizeN',type='numeric',
61     help='The library size of the control (normal) sample'
62   )
63   parser$add_argument(
64     '--libsizeT',type='numeric',

```

```

65     help='The library size of the case (tumor) sample'
66   )
67   parser$add_argument(
68     '--mindepth',type='numeric',
69     default=20,
70     help='The minimal depth of reads per window'
71   )
72   parser$add_argument(
73     '--minsize',type='numeric',
74     default=20000,
75     help='The minimal size of a window'
76   )
77
78   parser$add_argument(
79     '--regions',type='character',
80     default=NULL,
81     help='The regions, a text string or a file path indicating the regions
82 with three columns (chr/start/end) and without column names.'
83   )
84
85   parser$add_argument(
86     '--segtype',type='character',
87     default="mean.norm",
88     help='The type of segmentation. One of c(
89 "mean.norm","meanvar.norm","mean.cusum","var.css"
90 ). Multiple values are allowed and separated by ",".'
91   )
92
93   parser$add_argument(
94     '--dsN',type='integer',
95     default=1,
96     help='The downsampling factor of the control (normal) sample'
97   )
98
99   parser$add_argument(
100     '--dsT',type='integer',
101     default=1,
102     help='The downsampling factor of the test (tumor) sample'
103   )
104
105   parser$add_argument(
106     '--pcThreads',type='integer',
107     default=1,
108     help='The number of processors performing the parallel computing'
109   )
110
111   parser$add_argument(
112     '--auto.save',type='logical',
113     default=FALSE,
114     help='Whether to save (any completed results) automatically'
115   )
116
117   parser$add_argument(
118     '--auto.load',type='logical',
119     default=FALSE,
120     help='Whether to load (any previously completed results) automatically'
121   )
122   return(parser)
123 }
124
125
126 main <- function(){
127   parser <- PARSEME()
128   parser$helpme()
129
130   require(NGScopy)
131

```

```
132   if (length(regions)){
133     regions <- read_regions(regions)
134   }
135
136   obj <- NGScopy$new(
137     outFpre=outFpre,
138     inFpathN=inFpathN,
139     inFpathT=inFpathT,
140     libsizeN=libsizeN,
141     libsizeT=libsizeT,
142     mindepth=mindepth,
143     minsize=minsize,
144     regions=regions,
145     segtype=segtype,
146     dsN=dsN,
147     dsT=dsT,
148     pcThreads=pcThreads,
149     auto.save=auto.save,
150     auto.load=auto.load
151   )
152
153   obj$write_cn()
154   obj$write_segm()
155   ## obj$plot_out()
156   obj$plot_out(ylim=c(-3,3))
157
158   invisible()
159 }
160
161
162 main()
```



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

```

1  #!/usr/bin/env bash
2
3
4  ## *****
5  ## This is a command line example to run bin/ngscopy in `Bioconductor NGScopy'.
6  ##
7  ##
8  ## (c) Xiaobei Zhao
9  ##
10 ## v0.99.1
11 ## Mon Aug 11 09:57:53 EDT 2014 -0400 (Week 32)
12 ## [2014-08-11 09:59:52 EDT] add regions, dsN, dsT
13 ##
14 ## v0.99.0
15 ## Sat Jun 28 21:19:32 EDT 2014 -0400 (Week 25)
16 ##
17 ##
18 ## Reference:
19 ## Bioconductor NGScopy
20 ## http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
21 ##
22 ## Runme:
23 ## bash ngscopy-case1b-cmdline.sh &> ngscopy-case1b-cmdline.out
24 ##
25 ## *****
26
27
28 ## -----
29 ## The path of the executable, for instance, ${R_LIBS}/NGScopy/bin/ngscopy
30 ## A portable solution is to extract this path by `system.file' in R
31 ## We call such function from the command line below.
32 ## -----
33
34 ## ngscopyCmd=$(Rscript -e "cat(system.file('bin','ngscopy',
35 ## package='NGScopy', mustWork=TRUE))")
36 ngscopyCmd=$(Rscript -e "cat(Xmisc::get_executable('NGScopy'))")
37 echo ${ngscopyCmd}
38
39
40 ## -----
41 ## Get help
42 ## -----
43
44 ## ${ngscopyCmd} -h
45 ## ${ngscopyCmd} --help
46
47
48 ## -----
49 ## An example
50 ## This is a command line version of Case Study I from NGScopy User's Guide.
51 ## See http://www.bioconductor.org/packages/release/bioc/html/NGScopy.html
52 ## -----
53
54 ## a normal sample, given the NGScopyData package is installed
55 ## inFpathN=$(Rscript -e "cat(system.file('extdata','tps_N8.chr6.sort.bam',
56 ## package='NGScopyData', mustWork=TRUE))")
57 inFpathN=$(Rscript -e "cat(Xmisc::get_extdata('NGScopyData','tps_N8.chr6.sort.bam'))")
58
59 ## echo ${inFpathN}
60 ## ls -l ${inFpathN}
61
62 ## a tumor sample, given the NGScopyData package is installed
63 ## inFpathT=$(Rscript -e "cat(system.file('extdata','tps_90.chr6.sort.bam',
64 ## package='NGScopyData', mustWork=TRUE))")

```

```

65 inFpathT=$(Rscript -e "cat(Xmisc::get_extdata('NGScopyData','tps_90.chr6.sort.bam'))")
66 ## echo ${inFpathT}
67 ## ls -l ${inFpathT}
68
69 ## set pre-computed libsizes based on the original bam files of all chromosomes
70 libsizeN=5777087
71 libsizeT=4624267
72
73 ## set the regions, given the NGScopy package is installed
74 ## regions=$(Rscript -e "cat(system.file('extdata','hg19_chr6_0_171115067.txt',
75 ## package='NGScopy', mustWork=TRUE))")
76 regions=$(Rscript -e "cat(Xmisc::get_extdata('NGScopy','hg19_chr6_0_171115067.txt'))")
77 ## echo ${regions}
78
79 ## set downsampling factor; we do not downsample here by setting them to 1.
80 dsN=1
81 dsT=1
82
83 ## set a directory for output
84 outFpre="ngscopy-case1b-cmdline"
85
86 ## Run NGScopy given arguments and time it
87 time ${ngscopyCmd} --inFpathN=${inFpathN} --inFpathT=${inFpathT} --outFpre="${outFpre}" \
88 --libsizeN=${libsizeN} --libsizeT=${libsizeT} --regions=${regions} \
89 --dsN=${dsN} --dsT=${dsT} --pcThreads=1

```

The output,

```

1 /home/xiaobei/supplemental2/usr/local/lib64/R/library/NGScopy/bin/ngscopy
2 Loading required package: methods
3 Loading required package: Xmisc
4 Loading required package: NGScopy
5 auto.save: FALSE
6 auto.load: FALSE
7 20140815 10:02:42 EDT | set_normal | inFpathN
8 inFpathN: /home/xiaobei/supplemental2/usr/local/lib64/R/library/NGScopyData/extdata/tps_N8.chr6.sort.bam
9
10 20140815 10:02:42 EDT | set_tumor | inFpathT
11 inFpathT: /home/xiaobei/supplemental2/usr/local/lib64/R/library/NGScopyData/extdata/tps_90.chr6.sort.bam
12
13 20140815 10:02:42 EDT | set_outFpre
14 outFpre: ngscopy-case1b-cmdline
15
16 20140815 10:02:42 EDT | set_libsize | libsizeN
17 libsizeN: 5777087
18
19 20140815 10:02:42 EDT | set_libsize | libsizeT
20 libsizeT: 4624267
21
22 20140815 10:02:42 EDT | set_normal | mindepth
23 mindepth: 20
24
25 20140815 10:02:42 EDT | set_normal | minsize
26 minsize: 20000
27
28 20140815 10:02:42 EDT | set_regions
29 20140815 10:02:42 EDT | .proc_ref | process reference genome (in bam header of the normal sample)
30 20140815 10:02:42 EDT | .trim_regions | trim regions if exceeding the reference
31 20140815 10:02:42 EDT | .sort_regions | sort regions by reference
32 regions:
33 chr6 0 171115067
34
35 20140815 10:02:42 EDT | set_segmttype
36 segtype: c("mean.norm")

```

```

37 20140815 10:02:42 EDT | set_ds | dsN
38 dsN: 1
39
40 20140815 10:02:42 EDT | set_ds | dsT
41 dsT: 1
42
43 20140815 10:02:42 EDT | set_pcThreads
44 pcThreads: 1
45
46 20140815 10:02:42 EDT | write_cn
47 20140815 10:02:42 EDT | proc_cn
48 20140815 10:02:42 EDT | proc_normal | this may take a while depending on the size of your library.
49 20140815 10:02:42 EDT | (PID: 54971) Processing coords (refid, start, end): 5, 0, 171115067
50 20140815 10:04:02 EDT | Processed all 1 regions.
51 20140815 10:04:02 EDT | proc_tumor | this may take a while depending on the size of your library.
52 20140815 10:05:12 EDT | Processed all 2808 windows.
53 20140815 10:05:12 EDT | calc_cn
54 20140815 10:05:12 EDT | set_MoreArgs.cn
55 MoreArgs.cn not specified, assuming list(pseudocount=1,logr=TRUE).
56 MoreArgs.cn:
57 List of 2
58 $ pseudocount: num 1
59 $ logr      : logi TRUE
60
61 20140815 10:05:12 EDT | calc_cn | done
62 out$cn:
63 List of 3
64 $ cnr      : num [1:2808] 0.0163 -0.3708 -0.2638 -0.1643 -0.2232 ...
65 $ pseudocount: num 1
66 $ logr      : logi TRUE
67
68 20140815 10:05:12 EDT | calc_cn | done
69 out$cn:
70 List of 3
71 $ cnr      : num [1:2808] 0.0163 -0.3708 -0.2638 -0.1643 -0.2232 ...
72 $ pseudocount: num 1
73 $ logr      : logi TRUE
74
75 File saved:
76 outFpath="ngscopy-case1b-cmdline/ngscopy_cn.txt"
77
78 20140815 10:05:12 EDT | write_segm
79 20140815 10:05:12 EDT | proc_segm
80 20140815 10:05:12 EDT | calc_segm
81 20140815 10:05:12 EDT | set_MoreArgs.segm
82 MoreArgs.segm not specified, assuming list().
83 MoreArgs.segm:
84 list()
85
86 20140815 10:05:12 EDT | calc_segm | done
87 out$seg:
88 List of 1
89 $ mean.norm:List of 1
90 .. $ chr6:Formal class 'cpt' [package "changepoint"] with 10 slots
91 .. .. ..@ data.set : Time-Series [1:2808] from 1 to 2808: 0.0163 -0.3708 -0.2638 -0.1643 -0.2232 ...
92 .. .. ..@ cpttype   : chr "mean"
93 .. .. ..@ method    : chr "PELT"
94 .. .. ..@ test.stat : chr "Normal"
95 .. .. ..@ pen.type   : chr "SIC"
96 .. .. ..@ pen.value : num 7.94
97 .. .. ..@ cpts       : int [1:6] 550 551 1141 1951 1953 2808
98 .. .. ..@ ncpts.max : num Inf
99 .. .. ..@ param.est : List of 1
100 .. .. .. ..$ mean: num [1:6] 0.0801 -4.0712 -0.0345 -0.418 -3.3285 ...
101 .. .. .. ..@ date   : chr "Wed Jul 23 19:59:37 2014"
102
103 File saved:

```

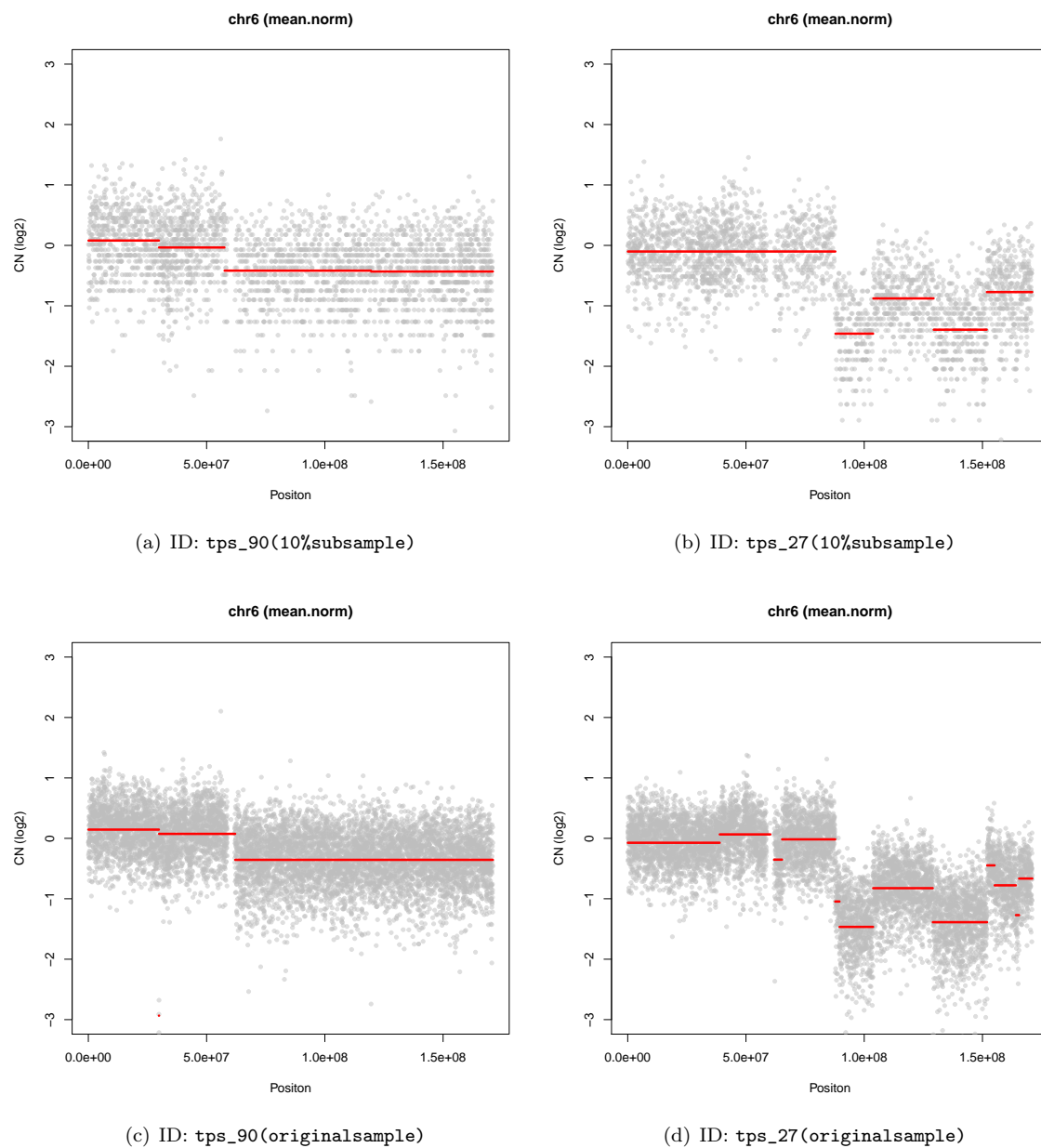
```
104 outFpath="ngscopy-case1b-cmdline/ngscopy_segm.txt"
105
106 20140815 10:05:12 EDT | plot_out
107 chr=chr6, segtype=mean.norm
108
109 File saved:
110 pdfFpath="ngscopy-case1b-cmdline/ngscopy_out.pdf"
111
112
113 real      2m48.398s
114 user      2m47.147s
115 sys       0m0.298s
```

## 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 ?? ?? and ??**).
2. Using the original data without any downsampling (**Figure ?? ?? and ??**).

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 preliminary tests with downsampled data.



**Figure 6:** Graphical output for comparison of NGScopy calling between the 10% subsample [?? and ??] and the original sample without downsampling [?? and ??].