How to use cghMCR

Jianhua Zhang Bin Feng

April 26, 2022

1 Overview

Copy number data (arrayCGH or SNP) can be used to identify genomic regions (Regions Of Interest or ROI) showing gains or losses that are common across samples. Existing algorithms, MCR (Aguirre et. al. 2004), GISTIC(), or GTS(), for the identification of ROI rely on the probe level data, which may be a concern when array density increases (to 1 million for example) or when data generated by arrays of different densities need to be analyzed together. The *cghMCR* initially implemented simplified version of MCR. An alternative approach (Segment Gain Or Loss or SGOL) were then added by applying a modified version of GISTIC algorithm so that the computations can be done based on segmented data. This vignette demonstrate how to use *cghMCR* to identify genomic alterations across samples profiled using copy number platforms using the two approaches (e. g. arrayCGH or SNP array).

Since both approaches use the output of CBS (*DNAcopy*), the first section of the vignette shows how to generate the segmented data using *DNAcopy* based on three raw arrayCGH data files to reduce the length of time required for the calculation.

2 From raw to segmented data

3 From raw data to segment list

The raw data used for segment computation are downloaded from the TCGA web site `http://www.` and stored in the *sampleData* directory of the package. Several *Bioconductor* packages may be used to process the raw data. Here we choose to use *limma* to process and normalize the raw data. The three samples files are:

```
> require("limma")
> arrayFiles  <- list.files(system.file("sampleData", package = "cghMCR"),
+      full.names = TRUE, pattern = "TCGA")
> arrayFiles
```

```
[1] "/tmp/RtmpHvW2NL/Rinst1e4a611124a9e4/cghMCR/sampleData/TCGA-06-0881-01A-02D-0387-02
[2] "/tmp/RtmpHvW2NL/Rinst1e4a611124a9e4/cghMCR/sampleData/TCGA-12-0818-01A-01D-0387-02
[3] "/tmp/RtmpHvW2NL/Rinst1e4a611124a9e4/cghMCR/sampleData/TCGA-12-0827-01A-01D-0387-02
```

`read.maimages` is a generic function of the *limma* package that can be used to read the process the raw data. In the example below we used the default settings of the function. Curious readers may read the man page of `read.maimages` for descriptions of the parameters and their possible settings.

```
> rawData <- read.maimages(arrayFiles, source = "agilent", columns =
+     list(R = "rMedianSignal", G = "gMedianSignal", Rb = "rBGMedianSignal",
+     Gb = "gBGMedianSignal"), annotation = c("Row", "Col", "ControlType",
+     "ProbeName", "GeneName", "SystematicName", "PositionX", "PositionY",
+     "gIsFeatNonUnifOL", "rIsFeatNonUnifOL", "gIsBGNonUnifOL", "rIsBGNonUnifOL",
+     "gIsFeatPopnOL", "rIsFeatPopnOL", "gIsBGPopnOL", "rIsBGPopnOL",
+     "rIsSaturated", "gIsSaturated"), names = basename(arrayFiles))

Read /tmp/RtmpHvW2NL/Rinst1e4a611124a9e4/cghMCR/sampleData/TCGA-06-0881-01A-02D-0387-02
Read /tmp/RtmpHvW2NL/Rinst1e4a611124a9e4/cghMCR/sampleData/TCGA-12-0818-01A-01D-0387-02
Read /tmp/RtmpHvW2NL/Rinst1e4a611124a9e4/cghMCR/sampleData/TCGA-12-0827-01A-01D-0387-02
```

Dye assignment defults to Cy5 = sample and Cy3 for reference in limma (set for expression arrays). However, arrayCGH experiments are usually carried out with sample dyed using Cy3. To take this into account, we define a design vector using 1 (Cy5 = sample) or -1 (Cy3 = sample) to indicate the dye assignment for each sample.

```
> rawData$design <- c(-1, -1, -1)
```

The following code does the background correction and normalization within and then between arrays:

```
> ma <- normalizeWithinArrays(backgroundCorrect(rawData, method = "minimum"), method =
```

Since some of the probes are not mapped to exact positions in the genome, we need to drop them together with the control probes.

```
> chrom <- gsub("chr([0-9XY]+):.*", "\\1", ma$genes[, "SystematicName"])
> dropMe <- c(which(!chrom %in% c(1:22, "X", "Y")), which(ma$genes[, "ControlType"] !=
```

A common approach to analyzing copy number data is to apply the `segment` function of *DNAcopy* to segment the normalized data so that chromosome regions with the same copy number have the same segment mean values.

```
> require(DNAcopy, quietly = TRUE)
> set.seed(25)
> cna <- CNA(ma$M[-dropMe, ],
+     gsub("chr([0-9XY]+):.*", "\\1", ma$genes[-dropMe, "SystematicName"]),
+     as.numeric(gsub(".*:([0-9]+)-.*", "\\1",
+       ma$genes[-dropMe, "SystematicName"])),
+     data.type = "logratio", sampleid = colnames(ma$M))
> segData <- segment(smooth.CNA(cna))

Analyzing: TCGA.06.0881.01A.02D.0387.02.short.txt
Analyzing: TCGA.12.0818.01A.01D.0387.02.short.txt
Analyzing: TCGA.12.0827.01A.01D.0387.02.short.txt
```

The *segData* object contains the segment list that we are going to use in the sections to follow. The segment list can be extracted from the *segData* object by issuing the following command.

```
> mySeglist <- segData[["output"]]
> head(mySeglist)

                                      ID chrom loc.start   loc.end num.mark
1 TCGA.06.0881.01A.02D.0387.02.short.txt     1    554268  66194006      594
2 TCGA.06.0881.01A.02D.0387.02.short.txt     1  66219285  97057519      215
3 TCGA.06.0881.01A.02D.0387.02.short.txt     1  97148420 150775100      248
4 TCGA.06.0881.01A.02D.0387.02.short.txt     1 150844444 150848509        2
5 TCGA.06.0881.01A.02D.0387.02.short.txt     1 150930484 247032049      795
6 TCGA.06.0881.01A.02D.0387.02.short.txt    10    138206 135356671     1066
  seg.mean
1  -0.0173
2   0.0788
3  -0.0001
4  -2.2079
5  -0.0029
6   0.0652
```

4 Identifying Segment Gain Or Loss (SGOL)

In this section or sections to follow, we are not going to used *mySeglist* we created in the previous section as the data set only contains three samples. Instead , we will use a different set of sample data that was created the same way but with more samples to make the results more interesting. The sample data is stored in the *data* subdirectory of the *CNTools* package and can be loaded into R by:

```
> require(CNTools, quietly = TRUE)
> data("sampleData", package = "CNTools")
> head(sampleData)

                   ID chrom loc.start  loc.end num.mark seg.mean
1 TCGA-02-0001-01C-01     1    554267 72533855     6384   0.0883
2 TCGA-02-0001-01C-01     1  72550247 72568008        2   1.2898
3 TCGA-02-0001-01C-01     1  72602596 74674719       93   0.1422
4 TCGA-02-0001-01C-01     1  74693651 74877529       20  -0.3194
5 TCGA-02-0001-01C-01     1  74885003 74952060        7  -0.6418
6 TCGA-02-0001-01C-01     1  74961517 75110250       10  -0.2808
```

The segment list shown above is a data frame but can not be used directly for computation across samples as each row only contains the segment data for a given segment within a sample. For computations on segmented data across samples, the segment list need to be converted into a matrix format with segments as rows and samples as columns. The *CNTools* packages provides the functionalities for data conversion and we are going to take the advantage of the package without detailing the algorithm of the conversion. Curious readers are encouraged to read the vignette of *CNTools* for detailed descriptions of the algorithms. Using the following code, we convert the segment list into a matrix format with by aligning samples based on chromosome segment defined by genes. Alternatively we can align samples based on overlapping chromosomal fragments. The *CNTools* vignette has an example for that. Since the sample data contains over 200 samples, we only take 20 random samples here for the sake of time.

```
> data(geneInfo)
> data(sampleData, package = "CNTools")
> set.seed(1234)
> convertedData <- getRS(CNSeg(sampleData[which(is.element(sampleData[, "ID"], sample(u
+ XY = FALSE, geneMap = geneInfo, what = "median")
```

Once we have the segment data converted into a matrix format, we can try to identify regions showing gains or losses that are common across samples. The *imput* parameter indicates whether cells with missing values will be imputed and the *parameter* indicates whether regions on the X and Y chromosome should be kept. Since our samples are a mixture of male and female DNAs profiled against pooled male human DNA, we choose to drop the data on X and Y chromosomes. Parameter *geneMap* is needed when samples will be aligned by genes. The *CNTools* contains a built human gene information data set (geneInfo) that was used in the code above. Users working on other organisms or their own gene mapping information need to create a gene mapping data set following the same format as *geneInfo* shown below:

Working on the data converted from segment list, we can compute the SGOL scores for genes (or chromosomal fragment if samples are aligned by regions) by calculating the

summations (parameter *method*) for all the positive values over a set threshold and all the negative values below a set threshold (*threshold* below).

```
> require(cghMCR, quietly = TRUE)
> SGOLScores <- SGOL(convertedData, threshold = c(-0.2, 0.2), method = sum)
> plot(SGOLScores)
```

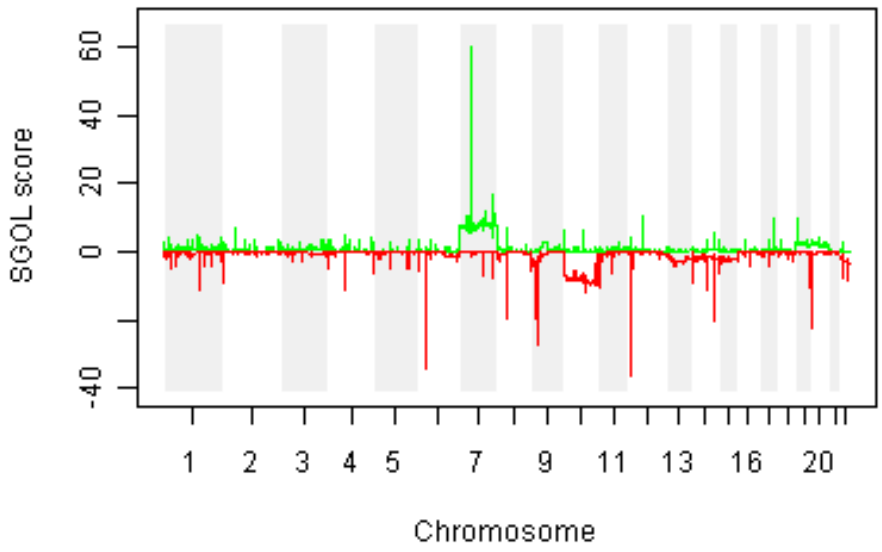

Figure 1:

Based on the SGOL scores, genes in regions of gains or losses can be obtained by a set of thresholds, say -20 and 20.

```
> GOIGains <- SGOLScores[which(as.numeric(unlist(gol(SGOLScores[, "gains"]))) >
+      20), "gains"]
> GOILosses <- SGOLScores[which(as.numeric(unlist(gol(SGOLScores[, "losses"]))) <
+      -20), "losses"]
> head(gol(GOIGains))

         gains
719839 22.0771
719857 38.7827
719869 45.6392
```

```
719916 52.8093
719920 52.0561
720101 48.8043
```

5 Identifying Minimum Common Regions (MCR)

The *MCR* approach was implemented following the heuristics listed below:

- MCRs are identified based on the segments obtained using *DNAcopy*
- Segments above an upper (defined by a parameter `alteredHigh`) and lower (`alteredLow`) threshold values of percentile are identified as altered.
- If two or more altered segments are deparated by less than 500 kb, the entire region spanned by the segments is considered to be an altered span.
- Highly altered segments or spans are retained as informative spans that define descrete locus boundaries.
- Informative spanes are compared acress samples to identify overlapping groups of positive or negative value segments.
- Minimal common regions (MCRs) are defined as contiguous spans having at least a recurrence rate defined by a parameter (`recurrence`) across samples.

We use the *segData* that were generated by running `segements` as the input to the `cghMCR` function. The parameter `gapAllowed` is numeric and indicate how many basepairs should two adjancent segments be apart, below which the segments will be joined to form an altered span. Parameters `alteredLow` and `alteredHigh` are also numerics and specify the lower and upper percential threshold values. Only segements with means less or greater than the lower or upper threshold values will be considered as altered regions and included in the subsequent analysis. `recurrence` is an integer defining the rate of recurrence for a region to show gain/loss across samples before it can be declared as an MCR. Due to the small number of sample size, the parameters are set to values that result in presentable results rather than correctness.

```
> cghmcr <- cghMCR(segData, gapAllowed = 500, alteredLow = 0.9,
+  alteredHigh = 0.9, recurrence = 100)
> mcrs <- MCR(cghmcr)
```

Using the above settings, we get a few MCRs that are common to the samples.

```
> head(cbind(mcrs[, c("chromosome", "status", "mcr.start", "mcr.end",
+                     "samples")]))
```

```
  chromosome status mcr.start  mcr.end
1 "1"        "gain" "55469748" "66010735"
1 "1"        "gain" "66063005" "66219285"
1 "1"        "gain" "66219285" "84504182"
1 "1"        "gain" "84504182" "84591307"
1 "1"        "gain" "84591307" "97057519"
1 "1"        "gain" "97057519" "121013177"
  samples
1 "TCGA.12.0818.01A.01D.0387.02.short.txt,TCGA.12.0827.01A.01D.0387.02.short.txt"
1 "TCGA.12.0818.01A.01D.0387.02.short.txt,TCGA.12.0827.01A.01D.0387.02.short.txt"
1 "TCGA.06.0881.01A.02D.0387.02.short.txt,TCGA.12.0818.01A.01D.0387.02.short.txt,TCGA.1
1 "TCGA.06.0881.01A.02D.0387.02.short.txt,TCGA.12.0818.01A.01D.0387.02.short.txt"
1 "TCGA.06.0881.01A.02D.0387.02.short.txt,TCGA.12.0818.01A.01D.0387.02.short.txt,TCGA.1
1 "TCGA.12.0818.01A.01D.0387.02.short.txt,TCGA.12.0827.01A.01D.0387.02.short.txt"
```

To include probe ids for the MCRs identified, we can call the function `mergeMCRProbes` to have probe ids within each MCR appended. Multiple probes are separated by a ",".

```
> mcrs <- mergeMCRProbes(mcrs[mcrs[, "chromosome"] == "7", ], as.data.frame(segData[["c
> head(cbind(mcrs[, c("chromosome", "status", "mcr.start", "mcr.end",
+                     "probes")]))
```

```
  chromosome status mcr.start  mcr.end    probes
7 "7"        "gain" "48653437" "56089387" NA
7 "7"        "gain" "56089387" "56570930" NA
7 "7"        "gain" "56570930" "63977464" NA
7 "7"        "gain" "68001742" "71432134" NA
7 "7"        "loss" "265449"   "48484884" NA
7 "7"        "loss" "64168901" "67228626" NA
```

6 Session Information

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

```
R version 4.2.0 RC (2022-04-21 r82226)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.4 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.16-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.16-bioc/R/lib/libRlapack.so
```

```
locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] cghMCR_1.55.0     CNTools_1.53.0    genefilter_1.79.0 DNAcopy_1.71.0
[5] limma_3.53.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.8.3           compiler_4.2.0         GenomeInfoDb_1.33.0
 [4] XVector_0.37.0         bitops_1.0-7           zlibbioc_1.43.0
 [7] bit_4.0.4              lattice_0.20-45        annotate_1.75.0
[10] RSQLite_2.2.12         memoise_2.0.1          png_0.1-7
[13] rlang_1.0.2            Matrix_1.4-1           DBI_1.1.2
[16] cli_3.3.0              fastmap_1.1.0          GenomeInfoDbData_1.2.8
[19] httr_1.4.2             Biostrings_2.65.0      S4Vectors_0.35.0
[22] vctrs_0.4.1            IRanges_2.31.0         grid_4.2.0
[25] stats4_4.2.0           bit64_4.0.5            Biobase_2.57.0
[28] R6_2.5.1               AnnotationDbi_1.59.0   XML_3.99-0.9
[31] survival_3.3-1         blob_1.2.3             splines_4.2.0
[34] BiocGenerics_0.43.0    KEGGREST_1.37.0        xtable_1.8-4
[37] RCurl_1.98-1.6         cachem_1.0.6           crayon_1.5.1
```

7 References

Aguirre, AJ, C. Brennan, G. Bailey, R. Sinha, B. Feng, C. Leo, Y. Zhang, J. Zhang, N. Bardeesy, C. Cauwels, C. Cordon-Cardo, MS Redston, RA DePinho and L. Chin. High-resolution Characterization of the Pancreatic Adenocarcinoma Genome. Proc Natl Acad Sci U S A. 2004. 101(24):9067-9072.