How to use CNTools

Jianhua Zhang

October 31, 2011

Overview

Studies have shown that genomic alterations measured as DNA copy number variations invariably occur across chromosomal regions that span over several genes, chromosome arms, or even whole chromosomes (reference). A common approach to analyzing DNA copy number data generated by microarrays, such as arrayCGH or SNP array, is to use the well established Circular Binary Segmentation (CBS) algorithm of the *DNAcopy* package to identify altered chromosome segments within each sample then find aberration accordant across samples. As samples usually differ in the number of altered regions and even for the same region where multiple samples show alterations the margins of the altered region rarely align across samples, the output of CBC can not be arranged as a matrix on which most computational algorithms operate. As the result, useful high level analysis such clustering or prediction can not be applied directly to segmented DNA copy number data.

The *CNTools* package provides the functionalities for converting the segmented DNA copy number data into a matrix format to facilitate further computational analyses. In this vignette, we show an example of clustering analysis using a public data set.

Algorithms

In the CNTools package, we implemented three algorithms of deriving a data set of matrix format that is referred to as reduced segment (RS) from now on for simplicity. First, by aligning chromosome segments across samples and then assigning segment means to overlapping chromosomal fragments (Figure 1) a reduced segment matrix (with the fragments as rows and samples as columns) can be obtained. This approach is straightforward but the drawback is the number of chromosomal fragments (or number of rows of the matrix) varies depending on the samples used for the calculation. Alternatively, by assigning segment means to genes within the segments for each sample a reduced segment matrix (with genes as rows and samples as columns) can be obtained. In this case the number of rows of the matrix does not vary with the samples of concern. Finally, by aligning chromosomal segments across each possible pair of samples a pair-wise reduced

segment matrix can be obtained. The output is a list of reduced segment matrix derived from all the possible sample pairs. The data structure is only suitable for unsupervised classifications but not other computations such as supervised classifications.
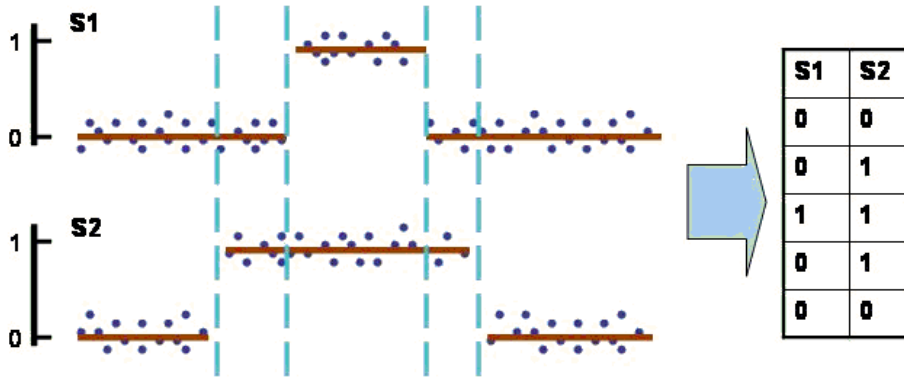

Figure 1: Diagram showing how a reduced segment matrix is derived from segment data for two samples denoted as S1 and S2. Blue dots are probe log 2 ratios and red horizontal lines are the mean values for each identified segment. Vertical dashed lines indicate the edges of overlapping chromosome regions from which the matrix on the right

An example

Here we use a public data set that is available from the TCGA data portal (`http://tcga-data.nci.nih.gov/tcga/homepage.htm`) to demonstrate the usage of the package. The data set contains 277 GBM tumor samples profiled using Agilent 244K density CGH array by the Harvard center. Normalized data were standardized by subtracting the median probe value of a sample from each probe log2 ratio of that sample before subjecting the data to the commonly used CBS algorithm available from the DNAcopy package of the Bioconductor (`http://www.bioconductor.org`) project. The name of the function for CBS is `segment` the output of which is a DNAcopy object containing three elements named *data, output* and *call*. The segment list needed to run `CNseg` of the *CNTools* package is the *output* element that can be extracted following the list operation like *segout[["output"]]* (assuming segout is the name of the object returned by the `segment` function). I have extracted the segment list from a previous run of `segment` and made the output element available as part of the CNTools package. The segment list can be obtained by first loading the package and data set into an R session using the following R code:

The segment list is then used to instantiate a CNSeg object that is associated with methods to process the segment list data. The main method is getRS with the following parameters:

- object: a CNSeg object containing the segment list of concern

- by: a character string that can be either "region", "gene", or "pair" indicating which of the three algorithms to use to convert the data

- imput: a boolean of TRUE or FALSE to indicate whether missing values will be imputed

- XY: a boolean of TRUE or FALSE to indicate whether data on the X and Y chromosomes will be included.

- geneMap: a matrix containing five columns named as chrom - the chromosome a gene is on, start - start location of a gene, end - ending location of a gene, geneid - entrez gene id, and genename - official gene symbol. geneInfo is required when by == "gene". An example of geneInfo can be found by loading the geneInfo object using data("geneInfo"). The object was built for human genes based on build 36.

- what: a character that can only be mean, median, max, or min that set the rule when a region has multiple segment values.

In this example, we elected to generate a reduced segment matrix of overlapping chromosomal regions without inputting missing values and drop data on the X and Y chromosomes (there are samples from both male and female patients in the data set). As the sample data contains more than 200 samples, in the example here, we use 20 of them due to reduce the time to run the examples. For computation by gene, we also truncated geneMap to reduce the time required to run the example. A test run of the whole data set (277 sample) on a MacPro calling `getRS` by region took about 9 minutes. In average, it takes around 20 minutes to run the whole data set.

```
> cnseg <- CNSeg(sampleData[which(is.element(sampleData[, "ID"], sample(unique(sample
> rdseg <- getRS(cnseg, by = "region", imput = FALSE, XY = FALSE, what = "mean")

Processing samples ... Done

> data("geneInfo")
> geneInfo <- geneInfo[sample(1:nrow(geneInfo), 2000), ]
> rdByGene <- getRS(cnseg, by = "gene", imput = FALSE, XY = FALSE, geneMap = geneInfo
```

The rdseg obtained contains the reduced segment matrix and can be extracted by using the rs method if the uses prefer to pursue further computational analyses using their own code.

```
> reducedseg <- rs(rdseg)
```

However, the CNTools package provides functions to further process the data for further analyses. For classifications or differential tests, we wish to filter out features that do not vary much across samples. This can be done by calling the genefilter function of the genefilter package (assuming the filters are applicable to a matrix. An example is provided here to get segments where there are at least five samples with a two copy gain. More filters are available in the *genefilter* package).

```
> f1 <- kOverA(5, 1)
> ffun <- filterfun(f1)
> filteredrs <- genefilter(rdseg, ffun)
```

In the following code we show how to use a filter that is available from this package to retain the top 20

```
> filteredrs <- madFilter(rdseg, 0.8)
```

Now we call the dist function to calculate the similarities between samples using the Euclidian distance and then do a hierarchical clustering analysis using complete linkage for cluster agglomeration.

Starting from raw data

Copy number data may be generated using different platforms (SNP, arrayCGH, ...) and processed using various packages, the steps of processing the data using *CNTools* are listed below without mentioning the detailed process of data normalization.

1. Use your favorite package to process the raw data (e. g. *limma* or *marray* for two color arrays and *oligoClasses* or *SNPchip* for SNP arrays)

2. Use normalized copy number data as input for the `segment` function of *DNAcopy* package to get the segment list. The segment list is the "output" element of the `segment` function.

3. Use the segment list as an input to CNSeg to instantiate a CNSeg object

4. Use the CNSeg object as an input to `getRS` to compute the reduced segment by gene or region

5. Use the `rs` function to extract the reduced segment from the output of `getRS` as a matrix and then operate on the matrix for further analyses or use the functions provided by *genefilter* or *CNTools* for filtering and then other high level analysis such as clustering as shown above
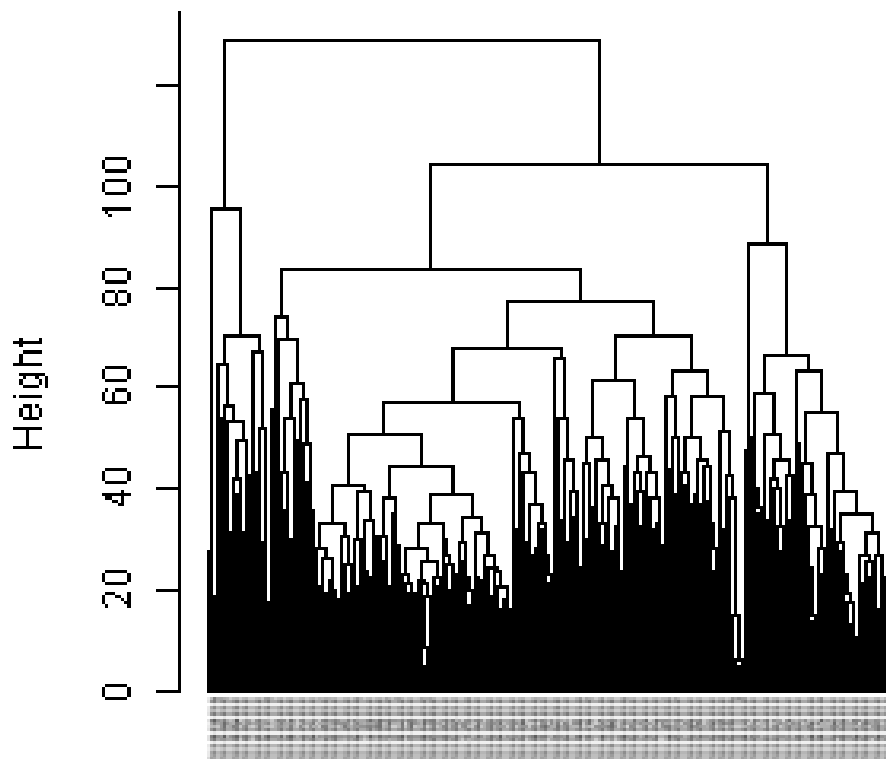
Figure 2: Dendrogram showing the clusters of all the samples contained in the sample data set

1 Session Infor

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

```
R version 2.14.0 (2011-10-31)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 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] CNTools_1.10.0    genefilter_1.36.0

loaded via a namespace (and not attached):
[1] AnnotationDbi_1.16.0 Biobase_2.14.0       DBI_0.2-5
[4] IRanges_1.12.0       RSQLite_0.10.0       annotate_1.32.0
[7] splines_2.14.0       survival_2.36-10     xtable_1.6-0
```