How to use cghMCR

Jianhua Zhang       Bin Feng
April 21, 2009

1 Overview

This vignette demonstrate how to use cghMCR to locate minimum common regions (MCR) across arrayCGH profiles derived from different samples. MCR was initially proposed by Dr. Lynda Chin’s lab (Aguirre et. al. 2004) to identify chromosome regions showing common gains/losses across samples using arrayCGH platform. The cghMCR package implements the algothrim as described 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 separated 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 discrete locus boundaries.
- Informative spans are compared across 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.

2 Getting Started

The example data used in this vignette are artificially constructed following the Agilent arrayCGH format with 5000 probes to maintain a reasonable speed of execution.
2.1 Read the sample data

The sample data are stored in the data subdirectory and can be loaded using data.

```r
> require("cghMCR")
> data("sampleData")
```

The sample data was created by reading three fabricated files using read.Agilent and then normalized using maNorm (norm = "loess") of marray. Readers are referred to marray for more information on how to read in Agilent profile data.

sampleData has three samples with intensity measures for 5000 probes.

```r
> maNsamples(sampleData)
[1] 3

> length(maLabels(maGnames(sampleData)))
[1] 5000
```

2.2 Identify chromosome segments

For each sample, we need to first identify chromosome segments having similar intensity measures. The function getSegments is a wrapper around the main functions provided by DNAcopy that are capable of detecting chromosome regions within which probe intensities remain similar.

```r
> segments <- getSegments(sampleData)
```

Analyzing: X.home.john.lib64.R.library.cghMCR.sampledata.sample1.agi
Analyzing: X.home.john.lib64.R.library.cghMCR.sampledata.sample2.agi
Analyzing: X.home.john.lib64.R.library.cghMCR.sampledata.sample3.agi

Results from the segmentation analysis (segments) is a list with three elements:

```r
> names(segments)
[1] "data" "output" "call"
```

The data element contain the normalized data, the output element contains the chromosome segments identified, and the call elements contains the function call with parameters passed indicated. Now, let’s plot the original data and the segments to see what the segments look like.

```r
> plot(segments)
```
Figure 1:
2.3 Identify MCRs

The element -output of the segments object generated in the previous section contains the segmentation data and will be used to get the MCRs. The parameter gapAllowed is numeric and indicate how many basepairs should two adjacent 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 segments 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 probes in the sample data, the parameters are set to values that result in presentable results rather than correctness.

```r
> cghmcr <- cghMCR(segments, gapAllowed = 500, alteredLow = 0.2, alteredHigh = 0.8, recurrence = 50)
> mcrs <- MCR(cghmcr)
```

Using the above settings, we get four MCRs but only one (on chromosome 7) of them are common to two samples.

```r
> print(cbind(mcrs[, c("chromosome", "status", "mcr.start", "mcr.end", "samples")]))
    [,1]
chromosome "7"
status "loss"
mcr.start "36220646"
mcr.end "39814784"
samples "X.home.john.lib64.R.library.cghMCR.sampledata.sample1.agi,X.home.john.lib64.R.library.cghMCR.sampledata.sample3.agi"
```

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 ",".

```r
> mcrs <- mergeMCRProbes(mcrs, segments[["data"]])
> print(cbind(mcrs[, c("chromosome", "status", "mcr.start", "mcr.end", "probes")]))
    [,1]
chromosome "7"
status "loss"
mcr.start "36220646"
mcr.end "39814784"
probes "A_14_P119613,A_14_P121347,A_14_P101357"
```
3 Session Information

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

R version 2.9.0 (2009-04-17)
x86_64-unknown-linux-gnu

locale:
LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=C;LC_MESSAGES=en_US

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

other attached packages:
[1] cghMCR_1.14.0  arrayQuality_1.22.0  RColorBrewer_1.0-2
[4] gridBase_0.4-3  hexbin_1.18.0  lattice_0.17-22
[7] convert_1.20.0  Biobase_2.4.0  marray_1.22.0
[10] limma_2.18.0  DNAcopy_1.18.0

4 References