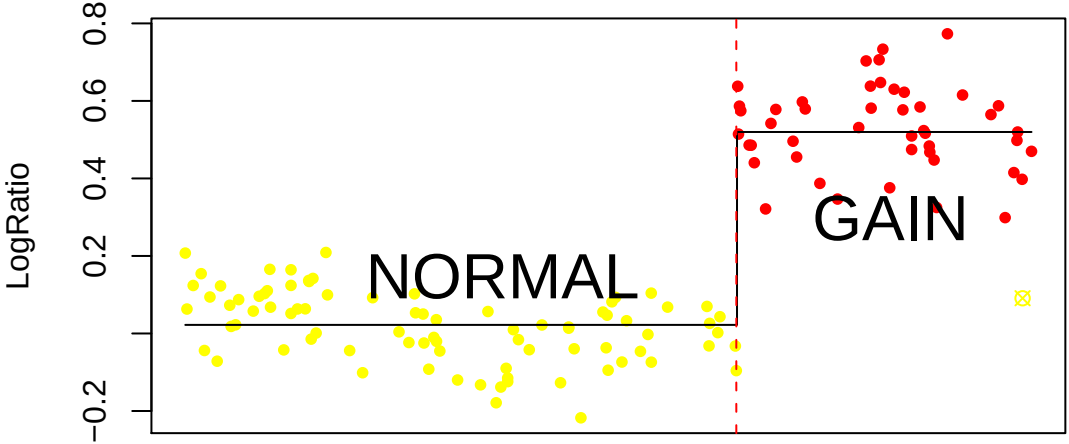
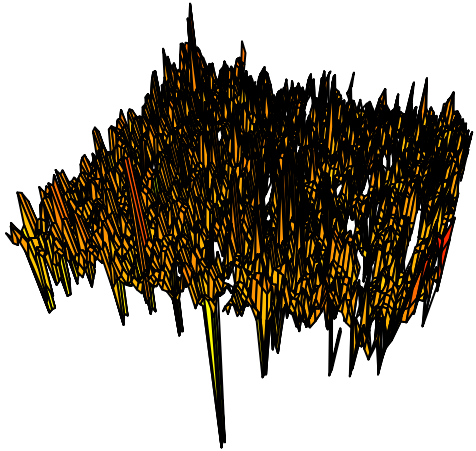
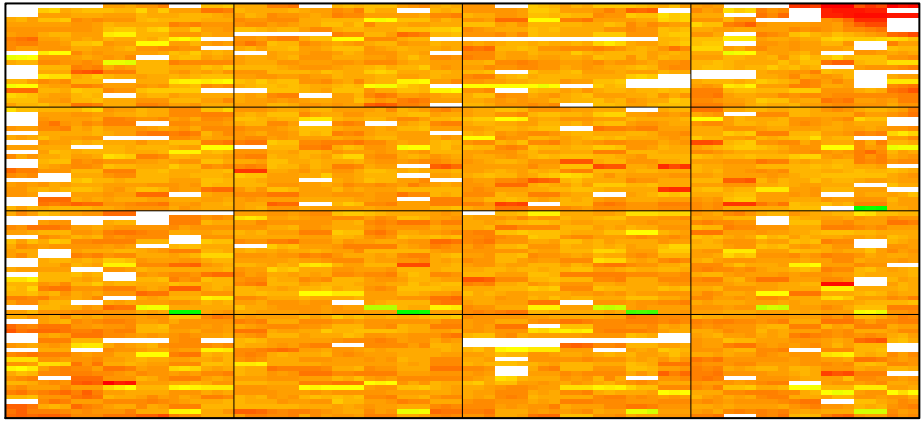
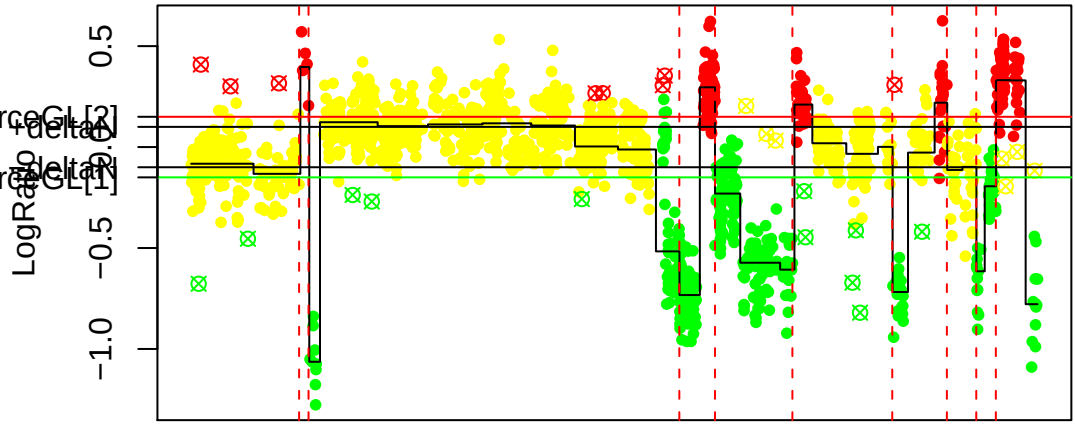
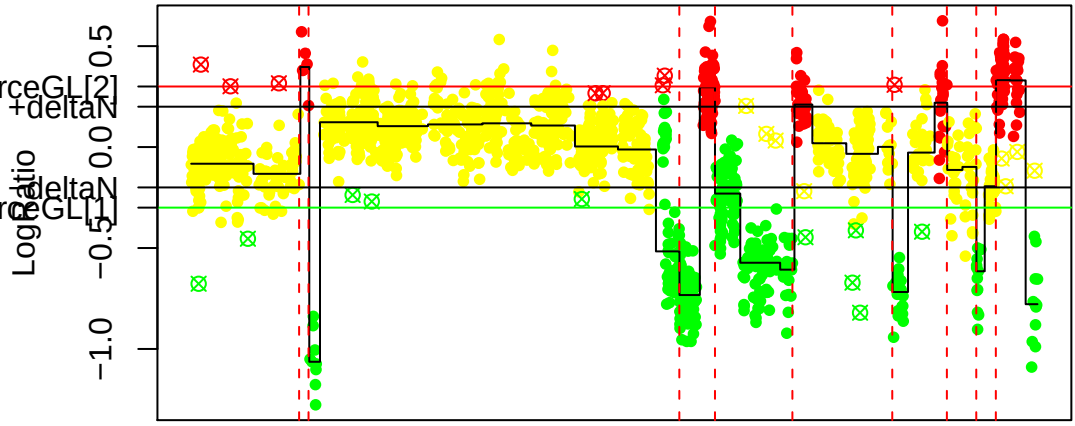
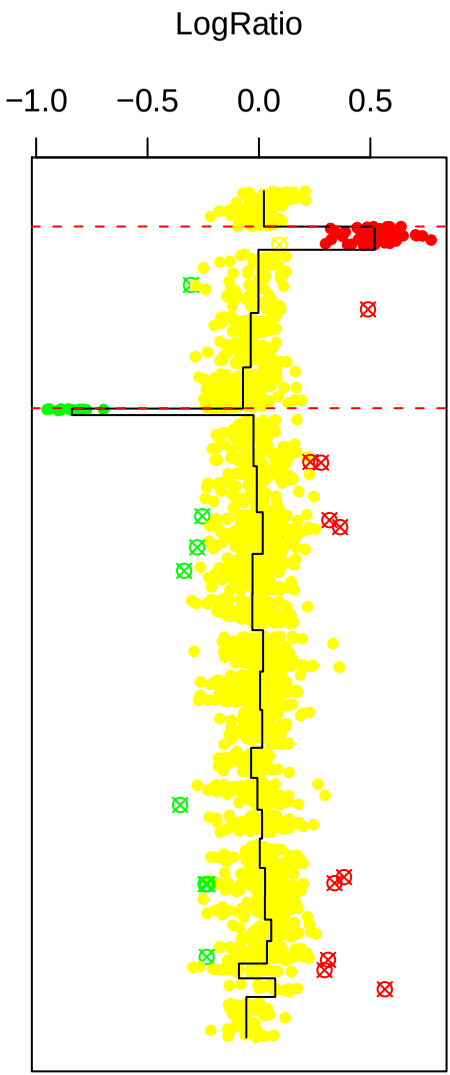

LogRatio
0.8
0.6
0.4
0.2
−0.2
NORMAL
GAIN






0.5
0.0
−0.5
−1.0
LogRatio
+deltaA
-deltaA


rceGL [2]
+deltaN
deltaN
eGL [1]
Log Ratio
0.5
0.0
−0.5
−1.0


LogRatio
−1.0
−0.5
0.0
0.5

GLAD

Jsamanie,Zaidf

November 12, 2009

array CGH I

- CGH (Comparative Genomic Hybridization) is an analysis method of the gain or loss of gene copies in DNA
- It consists on labeling sample and control DNA, hybridizing to probes
- According to the fluorescence patterns, copy loss or gain can be determined
- This method is mostly used in cancer research

GLAD Package I

This package detects anomalies in the number of gene copies, asigning normal, gained or lost status. The maintainer is Philippe Hupe. The package includes datasets:

- The data used is:
- Public Data set: 15 human cell strains with known karyotypes from the NIGMS Human Genetics Cell Repository.Each cell strain has been hybridized on CGH arrays of 2276 BACs, spotted in triplicates.
- Bladder cancer data: data from tumors collected at Henri Mondor Hospital hybridized on CGH arrays composed of 2464 BACs.

Classes

This package usessome of the following classes.

- array CGH: This class stores values from image analysis. It is a list composed of 2 objects, a data.frame (array Values) and a vector (Array design).
- The data.frame should contain the data row and column, the vector should contain Array column, rows, spot column and row.
- Col takes the values in 1:array Row*Spot Row and Row takes the values in 1:array Column *Spot Column

- Profile CGH, profile CHR: These classes stores values related to each clone available on the arrayCGH.
- Those values correspond to data of only one chromosome
- Both are composed of a list with a data frame which contains:
- LogRatio: Test of the log ratios of the data
- PosOrder: Rank of the object on the genome
- PosBase: Position of each clone on the genome
- Chromosome: The name of the chromosome
- Clone: Name of the clone
- These objects can be created with the function the function as.profile.CGH

Functions I

Some of the functions available in this package are glad and daglad, which is an upgraded version of glad. The glad function allows the detection of breakpoints in genomic profiles obtained by array CGH technology and assigns a status. This function segments the data obtained Some of the essential arguments of the daglad function are:

- profileCGH: Object of class profileCGH
- genomestep: If TRUE, a smoothing step over the whole genome is performed and allows to identify a cluster corresponding to the Normal DNA level.
- OnlySmoothing: If true, data segmentation is not optimized.
- OnlyOptimCall: If true, the user can provide segmented data.
- smoothfunc: algorithm used to smooth the data.

Example I

The following examples uses the data set described in Snijders et al. (2001), previously regarded as public data set. We will use data from the cell line gm13330. Load data.

```
> require(GLAD)
```

```
##################################################################
```

```
Have fun with GLAD
```

```
For smoothing it is possible to use either
the AWS algorithm (Polzehl and Spokoiny, 2002)
or the HaarSeg algorithm (Ben-Yaacov and Eldar, Bioinformat
```

```
If you use the package with AWS, please cite:
```

Example II

```
Hupe et al. (Bioinformatics, 2004) and Polzehl and Spokoiny

If you use the package with HaarSeg, please cite:
Hupe et al. (Bioinformatics, 2004) and (Ben-Yaacov and Elda

For fast computation it is recommanded to use
the daglad function with smoothfunc=haarseg

##################################################################

New options are available in daglad: see help for details.
```

```
> data(snijders)
```

Create profile:

Example III

```
> profileCGH <- as.profileCGH(gm13330)
```

glad I

Use the glad function to detect the breakpoints:

```
> res <- glad(profileCGH, mediancenter = FALSE,
+      smoothfunc = "lawsglad", bandwidth = 10,
+      round = 1.5, model = "Gaussian",
+      lkern = "Exponential", qlambda = 0.999,
+      base = FALSE, lambdabreak = 8,
+      lambdacluster = 8, lambdaclusterGen = 40,
+      type = "tricubic", param = c(d = 6),
+      alpha = 0.001, msize = 5, method = "centroid",
+      nmax = 8, verbose = FALSE)
```

```
[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints"
[1] "Results Preparation"
```

Plotting I

Now that the data was processed, we can make a plot to observe gain and loss. From the graph, we can see the gains, losses or matches in copy numbers.

```
> data(cytoband)
> plotProfile(res, unit = 3, Bkp = TRUE,
+     labels = FALSE, plotband = FALSE,
+     Smoothing = "Smoothing", cytoband = cytoband)
```

Plotting II

plots-005.pdf

Another Example I

- Data obtained by Veltman et al (2003) Plot with daglad

```
> data(veltman)
> profileCGH <- as.profileCGH(P9)
> profileCGH <- daglad(profileCGH,
+      mediancenter = FALSE, normalrefcenter = FALSE,
+      genomestep = FALSE, smoothfunc = "lawsglad",
+      lkern = "Exponential", model = "Gaussian",
+      qlambda = 0.999, bandwidth = 10,
+      base = FALSE, round = 1.5,
+      lambdabreak = 8, lambdaclusterGen = 40,
+      param = c(d = 6), alpha = 0.001,
+      msize = 5, method = "centroid",
+      nmin = 1, nmax = 8, amplicon = 1,
```

Another Example II

```
+      deletion = -5, deltaN = 0.2,
+      forceGL = c(-0.3, 0.3), nbsigma = 3,
+      MinBkpWeight = 0.35, CheckBkpPos = TRUE)
[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints and DNA copy numbe
[1] "Check Breakpoints Position"
[1] "Results Preparation"
```

- Plot the data.

```
> plotProfile(profileCGH, Smoothing = "Smoothing",
+     Bkp = TRUE, plotband = FALSE,
+     cytoband = cytoband)
> abline(h = c(-0.3, -0.2, 0.2, 0.3),
+     col = c("green", "black", "black",
+         "red"))
> axis(2, at = c(-0.3, -0.2, 0.2,
+     0.3), labels = c("forceGL[1]",
+     "-deltaN", "+deltaN", "forceGL[2]"),
+     las = 2)
```

plots-007.pdf

- To compare the effect of used parameters

```
> data(veltman)
> profileCGH <- as.profileCGH(P9)
> profileCGH <- daglad(profileCGH,
+      mediancenter = FALSE, normalrefcenter = FALSE,
+      genomestep = FALSE, smoothfunc = "lawsglad",
+      lkern = "Exponential", model = "Gaussian",
+      qlambda = 0.999, bandwidth = 10,
+      base = FALSE, round = 1.5,
+      lambdabreak = 8, lambdaclusterGen = 40,
+      param = c(d = 6), alpha = 0.001,
+      msize = 5, method = "centroid",
+      nmin = 1, nmax = 8, amplicon = 1,
+      deletion = -5, deltaN = 0.1,
```

```
+      forceGL = c(-0.15, 0.15), nbsigma = 3,
+      MinBkpWeight = 0.35, CheckBkpPos = TRUE)

[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints and DNA copy number ca
[1] "Check Breakpoints Position"
[1] "Results Preparation"
```

Comparison I

- Decreasing the parameters leads to a higher number of breakpoints identified

```
> plotProfile(profileCGH, Smoothing = "Smoothing",
+     Bkp = TRUE, plotband = FALSE,
+     cytoband = cytoband)
> abline(h = c(-0.15, -0.1, 0.1,
+     0.15), col = c("green", "black",
+     "black", "red"))
> axis(2, at = c(-0.15, -0.1, 0.1,
+     0.15), labels = c("forceGL[1]",
+     "-deltaN", "+deltaN", "forceGL[2]"),
+     las = 2)
```

Comparison II

plots-009.pdf

Using an arrayCGH object I

- In this example, an arrayCGH object is created.

```
> data(arrayCGH)
> array <- list(arrayValues = array2,
+     arrayDesign = c(4, 4, 21, 22))
> class(array) <- "arrayCGH"
> arrayPlot(array, "Log2Rat", bar = "none")
```

Using an arrayCGH object II

plots-010.pdf

Another View.

```
> arrayPersp(array, "Log2Rat", box = FALSE,
+      theta = 110, phi = 40, bar = FALSE)
```

plots-011.pdf

- Again, we load a data set. Use the glad function.We will see a genomic profile.

```
> data(snijders)
> profileCGH <- as.profileCGH(gm13330)
> res <- glad(profileCGH, mediancenter = FALSE,
+     smoothfunc = "lawsglad", bandwidth = 10,
+     round = 2, model = "Gaussian",
+     lkern = "Exponential", qlambda = 0.999,
+     base = FALSE, lambdabreak = 8,
+     lambdacluster = 8, lambdaclusterGen = 40,
+     type = "tricubic", param = c(d = 6),
+     alpha = 0.001, msize = 5, method = "centroid",
+     nmax = 8, verbose = FALSE)
```

```
[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints"
[1] "Results Preparation"
```

```
> text <- list(x = c(90000, 2e+05),
+     y = c(0.15, 0.3), labels = c("NORMAL",
+         "GAIN"), cex = 2)
> plotProfile(res, unit = 3, Bkp = TRUE,
+     labels = TRUE, Chromosome = 1,
+     Smoothing = "Smoothing", plotband = FALSE,
+     text = text, cytoband = cytoband)
```

plots-013.pdf