Complete process of package

Lori Shepherd and Jonathan Dare

July 20, 2007

Please download and utilize ex1 data

This document will take you through the entire process of loading and making objects. The following are requirements for the package that we have provided:

1 requirements

We have the following requirements for running this loadin example:

- Our flat files from image analysis software (Imagene,Feature Extraction, etc.) are in a directory 'arrays'
- We have an inventory of the samples indicating associated image.name, load.specs file, image.soft, design, default.map, orientation, and User. (see inventory help file/vignette for more details)
- We have a load.specs file located in a subdirectory config.files that will give details on how the sample should be readin. All our example samples came from the same software, however they utilized the mapping by block application. The arrays carry three different block dimensions 12 x 6 (72) , 12 x 4 (48) and 12 x 2 (24), and there are therefore three different load.specs files. (see load.specs help file/vignette for more details)
- We have a mapping file that will be used to create an R object storing mapping information

The ordering of what must take place is:

- build map file
- load in files and make aCGHroster object
- smooth samples
- make aCGH object
- perform circular binary segmentation

- fill in missing values
- have fun visulizing and graphing data

In the current directory which is above the subdirectory containing the flat files from image analysis software, for this example the subdirectory is called arrays, launch a R session at the command line by typing R, and load the aCGHplus package with the following command:

```
> library(aCGHplus)
```

We must first build the mapping file. In this case all the samples were build on the same build. We have provided the mapping file BACTable.txt that will be used to created the R object.

```
> Chrom.labels = c("chr1", "chr2", "chr3", "chr4", "chr5", "chr6",
+     "chr7", "chr8", "chr9", "chr10", "chr11", "chr12", "chr13",
+     "chr14", "chr15", "chr16", "chr17", "chr18", "chr19", "chr20",
+     "chr21", "chr22", "chrX", "chrY")
> buildMap(chrom.band.file = "BACTable.txt", chrom.band.file.sep = ",",
+     map.label = "HB4.5K.HG18", spot.ID.lbl = "spot.ID", loc.start.lbl = "loc.start",
+     loc.center.lbl = "loc.center", loc.stop.lbl = "loc.stop",
+     Chrom.lbl = "Chrom", Fine.Band.lbl = "Fine.Band", Chrom.labels = Chrom.labels,
+     maxnChrom = 1:24, XchromNum = 23, YchromNum = 24)
```

Next we must load in the files. Since there are 200 samples, we will run Batch calls. We will run 4 batch jobs, which means 50 samples will have objects created, be mapped, and have their design information stored per batch job.

```
> RosterBatch(array.dir = "arrays", inventory.file = "example.inv",
+     image.dir = "array.images", inv.sep = "\t", overwrite = T,
+     nbatchjobs = 4, ibatchjob = 1)
> RosterBatch(array.dir = "arrays", inventory.file = "example.inv",
+     image.dir = "array.images", inv.sep = "\t", overwrite = T,
+     nbatchjobs = 4, ibatchjob = 2)
> RosterBatch(array.dir = "arrays", inventory.file = "example.inv",
+     image.dir = "array.images", inv.sep = "\t", overwrite = T,
+     nbatchjobs = 4, ibatchjob = 3)
> RosterBatch(array.dir = "arrays", inventory.file = "example.inv",
+     image.dir = "array.images", inv.sep = "\t", overwrite = T,
+     nbatchjobs = 4, ibatchjob = 4)
```

Now we must assemble the batch roster calls and create an aCGHroster object.

```
> aCGHroster = create.roster.R(array.dir = "arrays", roster.file = "RData/aCGHrosterEx1.
+     inventory.file = "example.inv", image.dir = "array.images",
+     inv.sep = "\t", overwrite = F, returnFlag = T, vrb = T)
```

The next step is to smooth the samples. We again will break this up into 4 Batch calls.

```
> SmoothBatch(aCGHroster, overwrite = T, BatchDX = NA, nbatchjobs = 4,
+     ibatchjob = 1, time.order = F, map.name = NA, thetas = seq(0.5,
+         7.5, length = 101), cvLevel = 10, LossF = "L2", useResid = T,
+     BCoption = "none", lambdas = seq(0, 1, length = 2), DesignFlag = F,
+     lib.loc = NA, cyC.label = "array.images$matrix$cy3$Signal.Mean",
+     cyT.label = "array.images$matrix$cy5$Signal.Mean", cyC.BC.label = "array.images$mat
+     cyT.BC.label = "array.images$matrix$cy5$Background.Mean",
+     cy.grid.label = "array.images$matrix$Grid", exclude.rule = "array.images$matrix$cy3
+     output.label = "smooth2D.noDesign", pname = "auto", plt = T,
+     vrb = T, DesignList = c("Plate", "Pin", "PlateRow", "PlateCol",
+         "Repetition"), noDesignExit = F, weightSex.loess = 1/10,
+     excludeSex.smooth = F, excludeSex.design = F, weightNonMap.loess = 1/10,
+     excludeNonMap.smooth = T, excludeNonMap.design = T, ilambda.default = 1,
+     saveAll = T)
> SmoothBatch(aCGHroster, overwrite = T, BatchDX = NA, nbatchjobs = 4,
+     ibatchjob = 2, time.order = F, map.name = NA, thetas = seq(0.5,
+         7.5, length = 101), cvLevel = 10, LossF = "L2", useResid = T,
+     BCoption = "none", lambdas = seq(0, 1, length = 2), DesignFlag = F,
+     lib.loc = NA, cyC.label = "array.images$matrix$cy3$Signal.Mean",
+     cyT.label = "array.images$matrix$cy5$Signal.Mean", cyC.BC.label = "array.images$mat
+     cyT.BC.label = "array.images$matrix$cy5$Background.Mean",
+     cy.grid.label = "array.images$matrix$Grid", exclude.rule = "array.images$matrix$cy3
+     output.label = "smooth2D.noDesign", pname = "auto", plt = T,
+     vrb = T, DesignList = c("Plate", "Pin", "PlateRow", "PlateCol",
+         "Repetition"), noDesignExit = F, weightSex.loess = 1/10,
+     excludeSex.smooth = F, excludeSex.design = F, weightNonMap.loess = 1/10,
+     excludeNonMap.smooth = T, excludeNonMap.design = T, ilambda.default = 1,
+     saveAll = T)
> SmoothBatch(aCGHroster, overwrite = T, BatchDX = NA, nbatchjobs = 4,
+     ibatchjob = 3, time.order = F, map.name = NA, thetas = seq(0.5,
+         7.5, length = 101), cvLevel = 10, LossF = "L2", useResid = T,
+     BCoption = "none", lambdas = seq(0, 1, length = 2), DesignFlag = F,
+     lib.loc = NA, cyC.label = "array.images$matrix$cy3$Signal.Mean",
+     cyT.label = "array.images$matrix$cy5$Signal.Mean", cyC.BC.label = "array.images$mat
+     cyT.BC.label = "array.images$matrix$cy5$Background.Mean",
+     cy.grid.label = "array.images$matrix$Grid", exclude.rule = "array.images$matrix$cy3
+     output.label = "smooth2D.noDesign", pname = "auto", plt = T,
+     vrb = T, DesignList = c("Plate", "Pin", "PlateRow", "PlateCol",
+         "Repetition"), noDesignExit = F, weightSex.loess = 1/10,
+     excludeSex.smooth = F, excludeSex.design = F, weightNonMap.loess = 1/10,
+     excludeNonMap.smooth = T, excludeNonMap.design = T, ilambda.default = 1,
+     saveAll = T)
> SmoothBatch(aCGHroster, overwrite = T, BatchDX = NA, nbatchjobs = 4,
```

```
+     ibatchjob = 4, time.order = F, map.name = NA, thetas = seq(0.5,
+         7.5, length = 101), cvLevel = 10, LossF = "L2", useResid = T,
+     BCoption = "none", lambdas = seq(0, 1, length = 2), DesignFlag = F,
+     lib.loc = NA, cyC.label = "array.images$matrix$cy3$Signal.Mean",
+     cyT.label = "array.images$matrix$cy5$Signal.Mean", cyC.BC.label = "array.images$ma
+     cyT.BC.label = "array.images$matrix$cy5$Background.Mean",
+     cy.grid.label = "array.images$matrix$Grid", exclude.rule = "array.images$matrix$cy
+     output.label = "smooth2D.noDesign", pname = "auto", plt = T,
+     vrb = T, DesignList = c("Plate", "Pin", "PlateRow", "PlateCol",
+         "Repetition"), noDesignExit = F, weightSex.loess = 1/10,
+     excludeSex.smooth = F, excludeSex.design = F, weightNonMap.loess = 1/10,
+     excludeNonMap.smooth = T, excludeNonMap.design = T, ilambda.default = 1,
+     saveAll = T)
```

After the samples have been smoothed the aCGH object may be created.

```
> aCGH = make.aCGH(aCGHroster, sampleDX = NA, mapping.data = "RData/HB4.5K.HG18.RData",
+     LGR.label = "array.images$smooth2D.noDesign$Mxy.default",
+     Flagmat.label = NA, min.rep = 1, vrb = T, saveFlag = T, saveName = "RData/aCGH.RDa
```

We now need to perform circulary binary segmentation on the data. We again will break the data into four batch jobs.

```
> CBS.Batch(aCGH, overwrite = T, BatchDX = NA, nbatchjobs = 4,
+     ibatchjob = 1, time.order = F, alpha = 0.025, nperm = 10000,
+     outlier.mad.cut = 5, CBS.label = "CBS.nodesign", vrb = T)
> CBS.Batch(aCGH, overwrite = T, BatchDX = NA, nbatchjobs = 4,
+     ibatchjob = 2, time.order = F, alpha = 0.025, nperm = 10000,
+     outlier.mad.cut = 5, CBS.label = "CBS.nodesign", vrb = T)
> CBS.Batch(aCGH, overwrite = T, BatchDX = NA, nbatchjobs = 4,
+     ibatchjob = 3, time.order = F, alpha = 0.025, nperm = 10000,
+     outlier.mad.cut = 5, CBS.label = "CBS.nodesign", vrb = T)
> CBS.Batch(aCGH, overwrite = T, BatchDX = NA, nbatchjobs = 4,
+     ibatchjob = 4, time.order = F, alpha = 0.025, nperm = 10000,
+     outlier.mad.cut = 5, CBS.label = "CBS.nodesign", vrb = T)
```

After the CBS Batch calls finish, we need to assemble the CBS object.

```
> aCGH = Assemble.CBS.Batch(aCGH, CBS.label = "CBS.nodesign", overwrite.log2.ratios = F)
```

The last step before visualization and graphical functions may be called on the data, is to fill in missing values.

```
> aCGH = flankNA.CBS(aCGH, saveFlag = T, file = "RData/aCGH.RData")
```

It is sometimes useful to recenter data. There are three options for recentering data:

1. recenter on autosomes
2. recenter on autosomes removing any missing data (NA)
3. recenter on autosomes removing any missing data and then based on only a percentage of the most aberrant spots.

See R help file on recenter. The recommended, default, method of recentering is the second option.

```
> aCGH = recenter.aCGH(aCGH, useStep = 2)
```

The aCGH object may be used to create plots and graphs. There are also a number of user interactive applications.

The first task is generally to generate heatmaps for samples. We will create files for all the samples' heatmaps. There will be three files: one for before and after correction, one for background images, and one for the mean value chip images.

```
> makeHeatmaps(aCGH, smpls = NA, type = "correction", fileName = "correctionHeatmaps",
+     pdfFlag = T)
> makeHeatmaps(aCGH, smpls = NA, type = "background", fileName = "bgHeatmaps",
+     pdfFlag = T)
> makeHeatmaps(aCGH, smpls = NA, type = "mean", fileName = "meanHeatmaps",
+     pdfFlag = T)
```

The three files created will be in the plots directory.

The next files we will create are genomic plots for all samples. These files will also be created in the plots directory. The genomic plots will include both log2 ratios and smoothed log2 ratios.

```
> makeGenome3Row(aCGH, smpls = NA, fileName = "GenomeImages", pdfFlag = F,
+     fitFlag = T)
```

Each page will be a different sample's log2 ratios over the entire genome. Now that we have genomic profiles for each sample, it might be nice to have an idea of how the genome behaves across all samples. This overview of the genome can be plotted by a mean across sample graph and/or a frequency graph. Although it is possible to generate both these types of graphs seperately, we have included a function that will plot both graphs on a single page. The first page will be the full genomic view and each subsequent page will be of a chromosome.

```
> make.Mean.vs.Freq.plots(aCGH, smplDX = NA, maxNA = 0.25, rmNA = T,
+     fitVls = T, ylims = c(-1, 1), cutoffs = c(-0.15, 0.15), chrmList = 1:23,
+     fileName = "mean.freq.plots", recenter = T, recenterOpt = 2)
```

Now that some static plots have been generated, there are tools within the package to allow user interrogation of the data. This is accomplished through interactive plots. There are few in the package, however we will focus on the principle component viewer and a genomic profile to chip mapping viewer.