Description of gcrma package

Zhijin(Jean) Wu, Rafael Irizarry

April 7, 2020

Contents

1 Introduction

The *gcrma* package is part of the Bioconductor[1] project.

gcrma adjusts for background intensities in Affymetrix array data which include optical noise and non-specific binding (NSB). The main function `gcrma` converts background adjusted probe intensities to expression measures using the same normalization and summarization methods as `rma` (Robust Multiarray Average).

gcrma uses probe sequence information to estimate probe affinity to non-specific binding (NSB). The sequence information is summarized in a more complex way than the simple GC content. Instead, the base types (A,T,G or C) at each position (1-25) along the probe determine the *affinity* of each probe. The parameters of the position-specific base contributions to the probe affinity is estimated in an NSB experiment in which only NSB but no gene-specific bidning is expected. In version 2.0.0 we give options to the users to obtain these parameters from their choice.

[1] `http://www.bioconductor.org/`

With the probe affinities available, we estimate the relationship between the amount of NSB and the probe sequences. Specifically, we estimate the function

$$NSB = h(affinity)$$

by fitting a loess curve through

$$\text{MM probe intensities} \sim \text{MM probe affinities.}$$

In version 2.0.0 we also allow the use of any list of negative control(NC) probes instead of MM.

The background adjusted intensity is computed as the posterior mean of specific binding given the observed intensities and the probe sequences. This is done in function `bg.adjust.gcrma`. The background adjusted data is then converted to expression measures with function `rma` with the option *background=FALSE* to avoid another round of background correction.

The following terms are used throughout this document:

probe oligonucleotides of 25 base pair length used to probe RNA targets.

perfect match probes intended to match perfectly the target sequence.

PM intensity value read from the perfect matches.

mismatch the probes having one base mismatch with the target sequence intended to account for non-specific binding.

MM intensity value read from the mis-matches.

probe pair a unit composed of a perfect match and its mismatch.

affyID an identification for a probe set (which can be a gene or a fraction of a gene) represented on the array.

probe pair set PMs and MMs related to a common *affyID*.

CEL **files** contain measured intensities and locations for an array that has been hybridized.

CDF **file** contain the information relating probe pair sets to locations on the array.

probe **file** contain the information relating probe sequences to locations on the array.

2 What's new in version 2.0.0

We added a function `bg.adjust.gcrma` for the convenience of performing gcrma background adjustment only, without summarizing into gene level data.

In earlier versions, we compute the sequence-determined probe affinities using parameters estimated from data obtained in our non-specific binding (NSB) experiment. In version 2.0.0 we give various options for the users to choose their own sources of such data. Users can choose to

1. compute probe affinities based on their own non-specific biniding experiment (typically all probes in such experiments will contain non-specific bnidng, but the users can choose their list of probes)

2. compute probe affinties using each experimental array, with the user-defined negative control (NC) probes (MMs will be used if NC not specified).

3 Options for `gcrma`

Here we explain the major options in gcrma. Examples will be given in more details in the next section.

1. `affinity.info`

   This is in general an AffyBatch object but you can leave it as *NULL* and let `gcrma` compute it automatically. Instead of storing probe intensities, it contains probe affinities.

   When affinity.info is set to *NULL* (default), it will be computed within the function gcrma. To compute the affinity of each probe, we first obtain base-postion profiles (the contribution of each base type at each postion along the probe) from non-specific binding data. The user can choose to use the developers reference data or use each experimetnal array with indexes of negative control (NC) probes. If the NC probe index is not provided, MM probes will be used as NC probes.

   Some users express the concern that their array type may behave differently from the human hgu95 array, which was used in the developers' non-specific binding experiment. The user can choose to run an independent non-specific binding experiment for her/his own research. The affinity.info can be obtained separately by

   ```
   my.affinity.info <- compute.affinities.local(myNsbData)
   ```

   and this should be passed to function gcrma

   ```
   est<-gcrma(myExprData,affinity.info=my.affinity.info)
   ```

2. *type*

   The options for *type* are

   - *fullmodel*: uses both probe sequence information and observed MM probe intensities
   - *affinities*: uses probe sequence information and ignores MMs
   - *MM*: uses MM probe intensities and ignores sequence information

3. *fast*

   When *fast* is set to *TRUE*, an ad hoc procedure is used to speed up the non-specific binding correction. The default in previous version has been *fast=TRUE*, but is changed to *fast=FALSE* in version 2.0.0.

4 Getting started: the simplest example

You will need the following libraries: *affy*, *MASS*, cdf and probe packages for your chip type(such as *hgu95av2cdf* and *hgu95av2probe*).

The first thing you need to do is **load the package**.

```
R> library(gcrma) ##load the gcrma package
```

If all you want is to go from probe level data (*Cel* files) to expression measures here are some quick ways.

The quickest way of reading in data and getting expression measures is the following:

1. Create a directory, move all the relevant *CEL* files to that directory
2. Start R in that directory.
3. If using the Rgui for Microsoft Windows make sure your working directory contains the *Cel* files (use "File -> Change Dir" menu item).
4. Load the library.

   ```
   R> library(gcrma) ##load the gcrma package
   ```

5. Read in the data and create an expression, using RMA for example.

   ```
   R> Data <-  ReadAffy() ##read data in working directory
   R> eset <- gcrma(Data)
   ```

5 More Examples with options

For illustration we will use the Dilution data.

```
R> require(affydata)
R> data(Dilution)
```

1. Obtain expression values For example,

   ```
   R> Dil.expr1<-gcrma(Dilution)
   ```

   To use the faster ad hoc procedure one can call

   ```
   R> Dil.expr2<-gcrma(Dilution,fast=TRUE)
   ```

   Suppose the user has his/her own NSB experiment and wants to compute affinity.info with that experiment.

   ```
   R> myNsbData <- ReadAffy(``mynsb.cel'')
   R> my.affinity.info <- compute.affinities.local(myNsbData)
   R> Dil.expr3 <- gcrma(myExprData,affinity.info=my.affinity.info)
   ```

   Suppose the user would like to use NC probes in the experimental data to estimate probe affinities. Here we use MM probes as example for NC probes.

   ```
   R> mmIndex <- unlist(indexProbes(Dilution,"mm"))
   R> Dil.expr4 <- gcrma(Dilution,affinity.source="local",NCprobe=mmIndex)
   ```

   Since the MM probes are default setting when NCprobe is not provided, the above gives identical result as

   ```
   R> Dil.expr5 <- gcrma(Dilution,affinity.source="local")
   ```

2. Background adjustment only The function `bg.adjust.gcrma` allows one to perform background adjustment only.

   ```
   R> Dil.bgadj <- bg.adjust.gcrma(Dilution)
   R> Dil.expr6 <- rma(Dil.bgadj,background=FALSE)
   ```

   `gcrma` also tries to adjust for specific binding using probe sequence. The user can turn off this feature by specifying *GSB.adjust=FALSE*.

   ```
   R> Dil.bgadj <- bg.adjust.gcrma(Dilution,GSB.adjust=FALSE)
   ```

6 Efficient use of gcrma

Most users deal with one or a few types of GeneChip arrays for repeatedly. To use gcrma efficiently, one can compute the `affinity.info` and save it, thus save the time to compute `affinity.info` every time `gcrma` (or `bg.adjust.gcrma` is called.

For example, the Dilution data is from hgu95av2 chips. We can compute affinity.info of chip type "hgu95av2" using the NSB data provided in *gcrma* and save it in a file.

```
R> affinity.info.hgu95av2 <- compute.affinities("hgu95av2")
R> save(affinity.info.hgu95av2,file = "affinity.hgu95av2.RData")
```

or

```
R> affinity.info.hgu95av2 <- compute.affinities(cdfName(Dilution))
R> save(affinity.info.hgu95av2,file = "affinity.hgu95av2.RData")
```

Now when you need to call `gcrma` for the same type of array, there is no need to compute `affinity.info` again:

```
R> library(gcrma)
R> data(Dilution)
R> load("affinity.hgu95av2.RData")
R> Dil.expr7 <- gcrma(Dilution,affinity.info=affinity.info.hgu95av2)
```