# HowTo BGX

Ernest Turro
Imperial College London

April 25, 2007

## 1 Introduction

This vignette describes how to use *bgx*, a C++ implementation of a Bayesian hierarchical integrated approach to the modelling and analysis of Affymetrix GeneChip arrays. The model and methodology is described in Hein et al, 2005.

There are two ways to run *bgx*: (1) through R and (2) as a standalone binary. Both ways make use of probe level GeneChip data, which you must obtain as GeneChip CEL files.

## 2 Reading in the CEL files

When you load *bgx*, several required packages from the Bioconductor[1] project are automatically loaded.

```
> library(bgx)
```

The *affy* package allows you to read CEL files into an `AffyBatch` object. This can be achieved by changing your working directory to wherever the CEL files are stored and executing:

```
> aData <- ReadAffy()
```

This will read in the CEL files in alphabetical order and save the data in the `aData` object. Alternatively, you can specify the specific files you would like to read in by adding their paths to the argument list, for example:

```
> aData <- ReadAffy("CEL/choe/chipC-rep1.CEL", "CEL/choe/chipS-rep2.CEL")
```

---

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

# 3 Running BGX through R

A basic execution of the program can be performed by simply passing an `AffyBatch` object as a single parameter to the `bgx` function and saving the result in an `ExpressionSet` object. The result will hold array-specific gene expression values and their corresponding standard errors in `assayData(eset)$exprs` and `assayData(eset)$se.exprs` respectively.

```
> eset <- bgx(aData)
```

A more elaborate scenario would involve splitting the arrays into a number of conditions using the *samplesets* argument[2]; specifying which genes to analyse with the *genes* argument; specifying whether to take into account probe affinity with *probeAff*; setting the number of burn-in and post burn-in runs with the *burnin* and *iter* arguments respectively; setting the set of parameters to save with the *output* argument[3]; and specifying where to save the runs with *rundir*. Execute `help(bgx)` in R for a full explanation of all the parameters.

As an example, let us analyse the `Dilution` data set and save the results in the current working directory (”.”):

```
> library(affydata)
> library(hgu95av2cdf)
> data(Dilution)
> eset <- bgx(Dilution, samplesets = c(2, 2), probeAff = FALSE,
+     burnin = 2048, iter = 8192, genes = c(12500:12599), output = "all")
```

The `eset` object will contain gene expression information for each gene under each condition (not necessarily each array). You may obtain the gene expression measure using the `exprs` function. For instance:

```
> exprs(eset)[10:40, ]
```

```
          condition 1 condition 2
947_at        6.55138     6.26038
948_s_at      4.81587     4.51607
949_s_at      4.84355     4.63399
950_at        4.56655     4.31742
951_at        2.43563     2.72552
952_at        2.00411     2.53086
```

---

[2] Note that if your `AffyBatch` object contains information on the experimental design in the `phenoData` slot, you do not need to use the *samplesets* argument.

[3] *output* can be set to either ”minimal”, ”trace” or ”all”. See the documentation for an explanation of what these levels mean

```
953_g_at     5.32531     4.90937
954_s_at     6.37378     6.07759
955_at       6.62420     6.35586
956_at       7.00169     6.71399
957_at       4.63236     4.33294
958_s_at     5.56037     5.18119
959_at       1.47410     1.40060
960_g_at     5.25870     4.96330
961_at       2.04249     1.34850
962_at       2.36443     1.57513
963_at       4.60603     4.34192
964_at       4.29928     4.09498
965_at       1.57227     2.06872
966_at       4.41798     4.10954
967_g_at     4.90560     4.61491
968_i_at     2.43029     3.71769
969_s_at     4.85067     4.55694
970_r_at     6.31518     6.17359
971_s_at     2.44516     2.49688
973_at       4.46834     4.12362
974_at       2.86450     2.29705
975_at       4.32882     4.10482
976_s_at     3.67115     2.81030
977_s_at     4.93549     4.60938
978_at       2.83836     2.19194
```

Run `help(ExpressionSet)` in R for more information.

Note that *samplesets* should be set to an array specifying the number of replicates in each condition. If set to `(3,2)`, `bgx` will treat the first three arrays read into R as replicates under condition 1 and the next two as replicates under condition 2. You should make sure that all condition 1 files are read in first and all condition 2 files are read in second by `ReadAffy()`. You may check the order of the samples in your `AffyBatch` object by using the `sampleNames` function:

```
> sampleNames(Dilution)

[1] "20A" "20B" "10A" "10B"
```

## 4 Running BGX as a standalone binary

Occasionally it may be useful to run `bgx` as a standalone binary from the command line[4]. In this case, you should use the `standalone.bgx` function instead of the `bgx` function. It takes the same arguments as `bgx`, with the addition of *dirname*, which should specify where you would like to save the input files required by the standalone binary.

```
aData <- ReadAffy() # Read in 6 arrays across two conditions
                    # in alphabetical order
standalone.bgx(aData, samplesets=c(3,3), genes=c(1:650,1000:1200),
  burnin=16384, iter=65536, output="minimal",
  dirname="input-choe3replicates")
```

Once you have saved the input files, you should locate the binary, make sure it is executable[5], and pass the path to the newly created `infile.txt` file as a single argument. For example:

```
./bgx ../input-choe3replicates/infile.txt
```

## 5 Detailed analysis of the output

If you wish to analyse the output in detail, you should first read the output into a list as follows:

```
> bgxOutput <- readOutput.bgx("run.1")
```

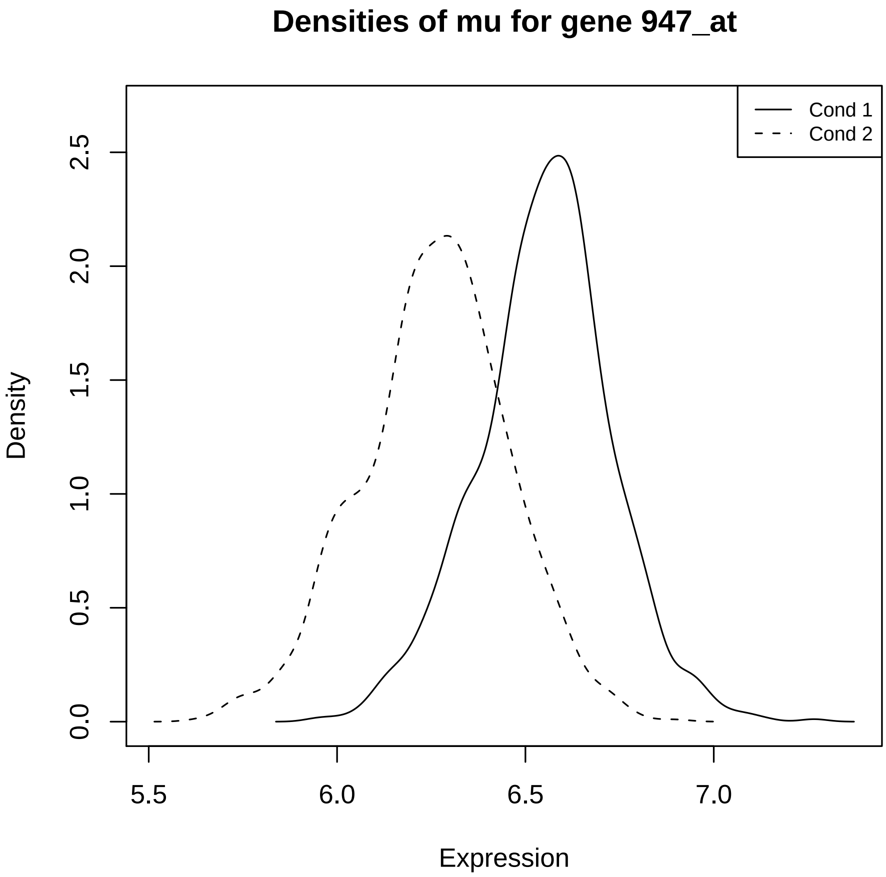
You may then pass the `bgxOutput` object to any of several analysis functions. For instance, to view the gene expression distributions under the various conditions for gene 10, you could do:

```
> plotExpressionDensity(bgxOutput, gene = 10)
```

---

[4] You can compile it by tweaking 'src/Makefile.standalone' to your specifications and running ‘make -f Makefile.standalone‘ from the 'src' directory.

[5] Under Unix-like environments, you can type `chmod +x bgx` at the command prompt to do this.



In order to get a list of ranked differential expression values, you could do:

```
> rankedGeneList <- rankByDE(bgxOutput)
> print(rankedGeneList[1:25, ])
```

```
                            Position DiffExpression
955_at                            18      33.222281
AFFX-HSAC07/X00351_5_at           83      31.507561
AFFX-HUMGAPDH/M33197_5_at         90      29.928834
956_at                            19      28.704045
941_at                             4      27.843495
947_at                            10      25.893534
AFFX-HSAC07/X00351_M_at           85      25.126291
AFFX-HUMGAPDH/M33197_M_at         92      24.585403
954_s_at                          17      22.634570
953_g_at                          16      22.067209
958_s_at                          21      20.690656
AFFX-hum_alu_at                   87      20.199556
```

```
AFFX-HUMGAPDH/M33197_3_at          88       19.762552
946_at                              9       19.069500
AFFX-BioDn-3_at                    70       16.881780
AFFX-HUMISGF3A/M97935_MB_at        97       13.474122
982_at                             44       12.165461
977_s_at                           39       12.136329
960_g_at                           23       11.765490
942_at                              5       11.391190
AFFX-HUMISGF3A/M97935_3_at         94       10.783033
AFFX-HUMISGF3A/M97935_MA_at        96        9.935485
969_s_at                           32        8.037317
AFFX-HSAC07/X00351_3_at            81        8.016859
967_g_at                           30        7.957880
```

Run `help(analysis.bgx)` for more detailed usage instructions on the analysis functions.