

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 Bioconductor¹ 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* argument²; 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* argument³; 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,
```

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,] # Shorthand for assayData(eset)\$exprs[10:40,]
```

| | condition 1 | condition 2 |
|----------|-------------|-------------|
| 947_at | 6.55658 | 6.24880 |
| 948_s_at | 4.82887 | 4.41459 |
| 949_s_at | 4.83150 | 4.52507 |
| 950_at | 4.50565 | 4.22432 |
| 951_at | 2.28303 | 3.12455 |
| 952_at | 1.76115 | 1.69608 |
| 953_g_at | 5.28602 | 4.84581 |

²Note 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

| | | |
|----------|---------|---------|
| 954_s_at | 6.39044 | 6.09729 |
| 955_at | 6.61334 | 6.34019 |
| 956_at | 7.00840 | 6.71031 |
| 957_at | 4.61623 | 4.29676 |
| 958_s_at | 5.54147 | 5.17050 |
| 959_at | 0.96296 | 1.10783 |
| 960_g_at | 5.18094 | 4.96274 |
| 961_at | 1.80088 | 1.16916 |
| 962_at | 1.49278 | 1.44293 |
| 963_at | 4.55301 | 4.20171 |
| 964_at | 4.24098 | 4.10523 |
| 965_at | 2.47125 | 1.42716 |
| 966_at | 4.51176 | 4.02216 |
| 967_g_at | 4.83973 | 4.61605 |
| 968_i_at | 3.23997 | 3.55578 |
| 969_s_at | 4.73917 | 4.47357 |
| 970_r_at | 6.31080 | 6.17463 |
| 971_s_at | 2.92644 | 2.82488 |
| 973_at | 4.33811 | 4.09892 |
| 974_at | 1.80054 | 1.45679 |
| 975_at | 4.33284 | 3.93199 |
| 976_s_at | 3.15728 | 3.31421 |
| 977_s_at | 4.88528 | 4.55692 |
| 978_at | 3.19099 | 2.48567 |

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 line⁴. In this case, you should use the `standalone.bgx` function instead of the `bgx` function.

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

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 executable⁵, 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)
```

⁵Under 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,]) # print top 25 DEG
```

| | Position | DiffExpression |
|---------------------------|----------|----------------|
| 955_at | 18 | 37.420388 |
| 941_at | 4 | 36.412272 |
| 956_at | 19 | 31.903408 |
| AFFX-HSAC07/X00351_5_at | 83 | 31.675462 |
| AFFX-HUMGAPDH/M33197_5_at | 89 | 28.416939 |
| AFFX-HUMGAPDH/M33197_M_at | 91 | 26.989338 |
| 947_at | 10 | 25.401802 |
| AFFX-HSAC07/X00351_M_at | 85 | 24.295601 |
| 954_s_at | 17 | 21.288885 |
| 946_at | 9 | 20.920797 |
| 958_s_at | 21 | 18.534143 |
| 953_g_at | 16 | 15.174308 |

| | | |
|-----------------------------|----|-----------|
| AFFX-HUMISGF3A/M97935_MA_at | 95 | 15.147713 |
| AFFX-HUMISGF3A/M97935_MB_at | 96 | 14.839909 |
| 982_at | 44 | 14.805079 |
| AFFX-HUMGAPDH/M33197_3_at | 87 | 14.393452 |
| AFFX-BioDn-3_at | 70 | 13.497089 |
| 948_s_at | 11 | 13.088445 |
| 949_s_at | 12 | 11.003962 |
| 957_at | 20 | 10.978025 |
| AFFX-HUMISGF3A/M97935_3_at | 93 | 10.345521 |
| 977_s_at | 39 | 10.079847 |
| 963_at | 26 | 9.595199 |
| 966_at | 29 | 8.188710 |
| 969_s_at | 32 | 8.120998 |

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