ArrayTools: Array Quality Assessment and Analysis Tool

Xiwei Wu and Xuejun Arthur Li

April 30, 2018

1 Introduction

The Affymetrix GeneChip is a commonly used tool to study gene expression profiles. The newly introduced Gene 1.0-ST arrays measure transcript expressions more accurately than the regular 3' -arrays. However, it lacks a tool to provide quality assessment and analysis for this type of array. This package is designed to provide solutions for quality assessment and to detect differentially expressed genes for the Affymetrix GeneChips, including both 3' -arrays and gene 1.0-ST arrays. The package provides functions that are easy to follow by biologists who have limited statistical backgrounds. The package generates comprehensive analysis reports in HTML format. Hyperlinks on the report page will lead to a series of QC plots, processed data, and differentially expressed gene lists. Differentially expressed genes are reported in tabular format with annotations hyperlinked to online biological databases. This guide will use an example dataset to demonstrate how to perform analysis of experiments with commonly used designs by using this package.

2 Data

We will use `exprsExample`, a simulated gene expression data, and its corresponding phenotype data file, `pDataExample` to illustrate some examples. Usually the expression data is generated from the Affymetrix Expression Console and pheno data file is created by the user.

```
> library(ArrayTools)
> data(exprsExample)
> head(exprsExample)
```

```
  probeset_id H1.CEL H2.CEL H3.CEL H4.CEL H5.CEL H6.CEL H7.CEL H8.CEL H9.CEL
1     7974617  20.94  37.84  21.05  19.02  17.88  24.10  29.54   9.67  10.31
2     8043502 334.29   3.39  14.28 256.20  94.44   9.41  25.26 321.78   6.19
3     7922008 416.82 774.87 659.94 346.52 361.60 307.01 372.93 416.47 417.56
4     7948123  21.54  13.90  35.51  36.36  38.81  36.12  41.32  29.70  24.05
5     7895635   0.01   5.58  22.13   4.59   8.99   1.32   5.11   0.30  14.22
6     8059985 323.61 306.80 241.76 355.07 410.07 583.40 411.06 315.05 315.75
```

```
  H10.CEL H11.CEL H12.CEL H13.CEL H14.CEL H15.CEL H16.CEL
1    2.72   34.72   43.47   18.94   72.74   38.31    9.38
2   67.32   44.75  170.68   25.78   63.70   71.56   27.01
3  222.75  363.14  336.07  403.53  218.79  367.89  193.06
4   39.06   43.96   48.48   12.68   13.66   15.16   53.27
5    6.44   13.41   14.66    3.10    2.37   18.83   11.09
6  463.78  470.00  312.35  583.83  484.79  394.48  530.53
```

```
> dim(exprsExample)
```

```
[1] 1000   17
```

```
> data(pDataExample)
> pDataExample
```

```
        Treatment Group
H1.CEL    Treated     A
H2.CEL    Treated     A
H3.CEL    Treated     A
H4.CEL    Treated     A
H5.CEL    Treated     B
H6.CEL    Treated     B
H7.CEL    Treated     B
H8.CEL    Treated     B
H9.CEL    Control     A
H10.CEL   Control     A
H11.CEL   Control     A
H12.CEL   Control     A
H13.CEL   Control     B
H14.CEL   Control     B
H15.CEL   Control     B
H16.CEL   Control     B
```

The expression data that created from Affymetrix Expression Console has an `probeset_id` column. To create an ExpressionSet, we need to use `probeset_id` as the rownames of expression data. Then we can create an `ExpressionSet` from the expression data and the phenotype data.

```
> rownames(exprsExample) <- exprsExample$probeset_id
> eSet <- createExpressionSet (pData=pDataExample,  exprs = exprsExample, annotation = "hugen
> dim(eSet)
```

```
Features  Samples
    1000       16
```

The `annotation` argument is important for the analysis for the Gene 1.0-ST arrays. Since we only support two types of Gene 1.0-ST arrays, please use either `hugene10sttranscriptcluster` or `mogene10sttranscriptcluster` as the value for the `annotation` argument.

3 Data Preprocessing

For Gene 1.0-ST arrays, data preprocessing include removing the `control` genes (default value is `rmControl =TRUE`) and takes the log2 of the expression value. Before taking log2, we added 1 to the expression value (the default value is `offset = 1`) because the expression value may have 0 value. If you want to output the preprocessed data to your local directory, you can use the `output = TRUE` option. Also, notice that the first argument is an `ExpressionSet`.

```
> normal <- preProcessGeneST (eSet, output=TRUE)
> normal

ExpressionSet (storageMode: lockedEnvironment)
assayData: 876 features, 16 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: H1.CEL H2.CEL ... H16.CEL (16 total)
  varLabels: Treatment Group
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: hugene10sttranscriptcluster

> dim(normal)

Features  Samples
     876       16
```

For 3' -arrays, data processing is done by using the `preProcess3prime` function, which is a wrapper function to perform normalization for the array. Instead of using the `ExpressionSet` as its argument, the `preProcess3prime` function requires an `AffyBatch` object. We can either choose `rma` or `gcrma` as the `method\verb` argument.

4 Quality Assessment

To generate the Quality Assessment Report, we need to use the Affymetrix Expression Console to generate a quality metric file. The sample quality metric file is similar to the `QC` file that can be obtained by using the `data` function. For Gene 1.0-ST arrays, we can use the `qaGeneST` function to create an HTML report. This report contains a series of plots, including Intensity Distribution , Mean Signal, BAC Spike, polya Spike, Pos Vs Neg Auc, Mad Residual Signal, RLE MEAN, and Hierarchical Clustering of Samples plots.

```
> data(QC)
> qaGeneST(normal, c("Treatment", "Group"), QC)
```

```
*** Output redirected to directory:  /tmp/RtmpeSPchU
*** Use HTMLStop() to end redirection.
```

For 3' -arrays, the `qa3prime` function is used to create a QC report. Instead of using `ExpressionSet` as its argument, an `AffyBatch` object is required. Furthermore, the QC file is not required for the 3' -Array.

5 Filtering

Before running analysis on the arrays, filtering out the uninformative genes may be an important step for your analysis. Three types of filtering methods are used in the `geneFilter` function. Suppose that if we want to remove genes with their inter-quartile range across the arrays with less than 10would also like to keep genes with at least 2 arrays with backgrounds greater than 4 at the same time, we can do the following:

```
> filtered <- geneFilter(normal, numChip = 2, bg = 4, iqrPct=0.1,  output=TRUE)
```

```
[1] "After Filtering, N = 763"
```

6 Analysis

The analysis takes a series of steps that includes creating a design and a contrast matrix, running regression, selecting significant genes, and creating an HTML report.

6.1 Design Matrix

A design matrix determines what type of model you are running. The design matrix is defined as a `designMatrix` class which can be created by the `new` function. To create a model with only one factor, which is equivalent to a one-way ANOVA model, we can do the following:

```
> design1<- new("designMatrix", target=pData(filtered), covariates = "Treatment")
> design1
```

```
   (Intercept) Treatment/Treated
1            1                 1
2            1                 1
3            1                 1
4            1                 1
5            1                 1
6            1                 1
7            1                 1
8            1                 1
9            1                 0
10           1                 0
```

```
11           1                  0
12           1                  0
13           1                  0
14           1                  0
15           1                  0
16           1                  0
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$Treatment
[1] "contr.treatment"
```

To create a model with two factors, we can do the following:

```
> design2<- new("designMatrix", target=pData(filtered), covariates = c("Treatment", "Group"))
> design2

   (Intercept) Treatment/Treated Group/B
1            1                  1       0
2            1                  1       0
3            1                  1       0
4            1                  1       0
5            1                  1       1
6            1                  1       1
7            1                  1       1
8            1                  1       1
9            1                  0       0
10           1                  0       0
11           1                  0       0
12           1                  0       0
13           1                  0       1
14           1                  0       1
15           1                  0       1
16           1                  0       1
attr(,"assign")
[1] 0 1 2
attr(,"contrasts")
attr(,"contrasts")$Treatment
[1] "contr.treatment"

attr(,"contrasts")$Group
[1] "contr.treatment"
```

6.2 Contrast Matrix

For the one-way ANOVA model based on the `design1`, if we want to compare `Treated` vs. `Control`, we can do the following:

```
> contrast1 <- new("contrastMatrix", design.matrix = design1,  compare1 = "Treated", compare2
> contrast1

     [,1]
[1,]    0
[2,]    1
```

To perform the same comparison and to controll the `Group` effect (Randomized Block Design), we can write:

```
> contrast2 <- new("contrastMatrix", design.matrix = design2, compare1 = "Treated", compare2 =
> contrast2

     [,1]
[1,]    0
[2,]    1
[3,]    0
```

6.3 Regression

To run the gene-wise regression, we can use the `regress` function. This function will create a `regressResult` object.

```
> result1 <- regress(filtered, contrast1)
> result2 <- regress(filtered, contrast2)
```

6.4 Select Significant Genes

We can select differentially expressed genes by using the `selectSigGene` function. To select differentially expressed genes based on p-values less than 0.05 and fold change greater than log2 of 1.5, we can write the following codes:

```
> sigResult1 <- selectSigGene(result1, p.value=0.05, fc.value=log2(1.5))
> sigResult2 <- selectSigGene(result2, p.value=0.05, fc.value=log2(1.5))
```

We can use the `Sort` function to sort the `regressResult` object by p-value in ascending order (`sorted.by = 'pValue'`) or log2 Ratio in descending order (`sorted.by = 'log2Ratio'`) or F statistics in descending order (`sorted.by = 'F'`).

```
> Sort(sigResult1, sorted.by = 'pValue')

There are  24  significant genes.

         ID Log2Ratio.1         F      pValue   adjPVal
180 8176933  -1.4476794 11.794504 0.003114917 0.6755821
682 7899348   0.8942033 11.423345 0.003506911 0.6755821
185 7961187  -1.6993006 10.579407 0.004622667 0.6755821
550 7967872  -0.6418379 10.517888 0.004718485 0.6755821
27  8133314  -1.1015420 10.079615 0.005469593 0.6755821
```

```
49  8176230  -0.6401176  9.715964 0.006196001 0.6755821
79  8121138  -0.6781879  9.715003 0.006198061 0.6755821
720 7986348  -0.6792892  9.040814 0.007852412 0.6755821
323 7973743   0.6500787  7.348170 0.014716392 0.6755821
246 8008517   0.8480338  7.314129 0.014911694 0.6755821
360 8170971   0.6443034  6.841945 0.017947995 0.6755821
751 8154223  -0.9634973  6.063748 0.024620521 0.6755821
364 8057004   1.1961724  5.930130 0.026030941 0.6755821
28  8139121  -1.1025219  5.642331 0.029393947 0.6755821
184 7945169   1.5171376  5.363939 0.033127890 0.6755821
248 7969493   2.1830182  5.348171 0.033355126 0.6755821
705 8146790  -0.7653335  5.206896 0.035472801 0.6755821
525 7968577  -0.9597539  5.091253 0.037321581 0.6755821
279 8006877   1.4172392  4.900766 0.040613733 0.6755821
732 8104607  -1.2819453  4.879652 0.040998796 0.6755821
```

```
> Sort(sigResult2, sorted.by = 'F')
```

```
There are  25  significant genes.

         ID Log2Ratio.1         F      pValue   adjPVal
180 8176933  -1.4476794 11.214364 0.004000748 0.6502387
682 7899348   0.8942033 11.155598 0.004076069 0.6502387
185 7961187  -1.6993006 10.545722 0.004960301 0.6502387
550 7967872  -0.6418379 10.425267 0.005159577 0.6502387
27  8133314  -1.1015420  9.563225 0.006882225 0.6502387
79  8121138  -0.6781879  9.408187 0.007256921 0.6502387
49  8176230  -0.6401176  9.152997 0.007925208 0.6502387
360 8170971   0.6443034  8.741519 0.009155414 0.6502387
720 7986348  -0.6792892  8.530873 0.009868294 0.6502387
323 7973743   0.6500787  7.382043 0.015066772 0.6502387
248 7969493   2.1830182  7.276650 0.015683249 0.6502387
246 8008517   0.8480338  6.905919 0.018092358 0.6502387
732 8104607  -1.2819453  6.372061 0.022343489 0.6502387
279 8006877   1.4172392  6.302352 0.022978852 0.6502387
364 8057004   1.1961724  5.848846 0.027656506 0.6502387
751 8154223  -0.9634973  5.844127 0.027710623 0.6502387
28  8139121  -1.1025219  5.386542 0.033587624 0.6502387
401 7927915  -0.9593178  5.294730 0.034933073 0.6502387
184 7945169   1.5171376  5.166801 0.036912820 0.6502387
705 8146790  -0.7653335  5.056408 0.038725850 0.6502387
```

6.5 Creating Reports

To output the differentially expressed genes along with annotations to an HTML file in your current working directory, we can use the `Output2HTML` function.

```
> Output2HTML(sigResult1)
> Output2HTML(sigResult2)
```

7 Detecting Interaction

Interaction is a statistical term referring to a situation when the relationship between the outcome and the variable of the main interest differs at different levels of the extraneous variable.

Just like before, we need to create the design and contrast matrices to detect the interaction effect.

```
> designInt <- new("designMatrix", target=pData(filtered), covariates = c("Treatment", "Group
> designInt

   (Intercept) Treatment/Treated Group/B Treatment/Treated:Group/B
1            1                 1       0                         0
2            1                 1       0                         0
3            1                 1       0                         0
4            1                 1       0                         0
5            1                 1       1                         1
6            1                 1       1                         1
7            1                 1       1                         1
8            1                 1       1                         1
9            1                 0       0                         0
10           1                 0       0                         0
11           1                 0       0                         0
12           1                 0       0                         0
13           1                 0       1                         0
14           1                 0       1                         0
15           1                 0       1                         0
16           1                 0       1                         0
attr(,"assign")
[1] 0 1 2 3
attr(,"contrasts")
attr(,"contrasts")$Treatment
[1] "contr.treatment"

attr(,"contrasts")$Group
[1] "contr.treatment"

> contrastInt <- new("contrastMatrix", design.matrix = designInt, interaction = TRUE)
> contrastInt

     [,1]
[1,]    0
[2,]    0
[3,]    0
[4,]    1
```

To identify genes with an interaction effect, we can use the same `regress` and `selectSigGene` functions:

```
> resultInt <- regress(filtered, contrastInt)
> sigResultInt <-selectSigGene(resultInt, p.value=0.05, fc.value=log2(1.5))
```

For genes with the interaction effect, they should be analyzed separately within each group. For genes without any interaction gene, they should be analyzed together. This step can be achieved by using the `postInteraction` function. The `postInteraction` function returns an object of `interactionResult` class. The components of the `interactionResult` object consist of a list of `regressResult` objects. The first component is a `regressResult` object for all the genes. The second component contains the result for genes without interaction. The third and the fourth components (since `Group` only contains two factors, `A` and `B`) contain results for genes with interaction only among groups `A` and `B`, respectively. Then we can use the `selectSigGeneInt` function again to select differently expressed genes within each component of the `interactionResult` object.

```
> intResult <- postInteraction(filtered, sigResultInt, mainVar ="Treatment",  compare1 = "Tre
> sigResultInt <- selectSigGeneInt(intResult, pGroup = 0.05, pMain = 0.05)
```

We can use the `Output2HTML` function again to output the differentially expressed genes along with annotations to an HTML file in your current working directory.

```
> Output2HTML(sigResultInt)
```

8 Creating Index File

We have created multiple outputs, including normalized data, filtered data, and differently expressed genes for multiple models. We can create an index file that can link all of these results.

```
> createIndex (sigResult1, sigResult2, intResult)

 *** Output redirected to directory:  /tmp/RtmpeSPchU
 *** Use HTMLStop() to end redirection.
```