Case studies of BioQC

Jitao David Zhang

June 1, 2014

Abstract

In this vignette we demonstrate the use of BioQC by a simulated expression dataset, and compare the performance of Wilcoxon-Mann-Whitney rank sum test algorithm implemented in BioQC to other implementations available in R. To use BioQC, the users only need to provide an expression dataset, in the form of a numeric matrix, or an *ExpressionSet* object. The BioQC package provides tissue-specific genes that can be used directly with the algorithm. The output is one score for each tissue type and each sample. The ranks of the score within each sample can be compared with prior knowledge about the sample to infer tissue heterogeneity. The hypotheses generated by BioQC should be further tested by follow-up experiments.

1 A dummy example

We demonstrate the basic use of the package by a dummy example. First we load BioQC library and the tissue signatures into the R session.

```
> library(Biobase)
> library(BioQC)
> gmtFile <- system.file("extdata/exp.tissuemark.affy.roche.symbols.gmt", package="BioQC")
> gmt <- readGmt(gmtFile)
```

Next we synthesize an ExpressionSet object, using randomly generated data.

```
> Nrow <- 2000L
> Nsample <- 5L
> gss <- unique(unlist(sapply(gmt, function(x) x$genes)))
> myEset <- new("ExpressionSet",
+               exprs=matrix(rnorm(Nrow*Nsample), nrow=Nrow),
+               featureData=new("AnnotatedDataFrame",
+                 data.frame(GeneSymbol=sample(gss, Nrow))))
```

Finally we run the BioQC algorithm and print the summary of the results. As expected, no single tissue scored significantly after multiple correction.

```
> dummyRes <- wmwTest(myEset, gmt, alternative="greater")
> summary(p.adjust(dummyRes, "BH"))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.8235  0.9145  1.0000  0.9630  1.0000  1.0000 
```

1.1 Using basic data structures

The dummy example above shows how to run BioQC algorithm with an *ExpressionSet* and a list read from GMT files (a file format capturing gene sets). Users can also use more basic data structures (e.g. matrices and list of integer indexes) to run the algorithm as shown by the following example. Other data structures will be coerced into these basic data structures; we refer to interested user to the documentation of *wmwTest* function.

```
> myMatrix <- matrix(rnorm(Nrow*Nsample),
+                    ncol=Nsample,
+                    dimnames=list(NULL, LETTERS[1:Nsample]))
> myList <- list(signature1=sample(1:Nrow, 100),
+                signature2=sample(1:Nrow, 50),
+                signature3=sample(1:Nrow, 200))
> wmwTest(myMatrix, myList)

                    A          B         C         D         E
signature1 0.96204402 0.04965141 0.7712248 0.8305735 0.9467891
signature2 0.83157528 0.74666765 0.1798944 0.6342453 0.2656264
signature3 0.07641411 0.18151649 0.9258939 0.5160349 0.2402755
```

2 Case study with a real data set

As another example, we have applied BioQC to a real gene expression profiling dataset. BioQC helped to generate hypotheses about potential contamination of rat kidney examples by pancreas tissues, which was confirmed by further qRT-PCR experiments.

The data and script used to perform the analysis can be found at `https://github.com/Accio/BioQC-example`.They are not included in the package due to size limitations.

3 Benchmarking against R implementation

In the core of *wmwTest*, an efficient C-implementation makes it feasible to run a large number of Wilcoxon-Mann-Whitney tests on large-scale expression profiling datasets and with many signature lists. Compared to native R implementations in *stats* (*wilcox.text*) and *limma* (*rankSumTestWithCorrelation*) packages, the *BioQC* implementation requires less memory and avoids repetitive statistical ranking of data.

The following code, though in a small scale, demonstrates the difference between the performances of two implementations.

```
> bm.Nrow <- 22000
> bw.Nsample <- 5
> bm.Ngs <- 5
> bm.Ngssize <- sapply(1:bm.Ngs, function(x) sample(1:bm.Nrow/2, replace=TRUE))
> ind <- lapply(1:bm.Ngs, function(i) sample(1:bm.Nrow, bm.Ngssize[i]))
> exprs <- matrix(round(rnorm(bm.Nrow*bw.Nsample),4), nrow=bm.Nrow)
> system.time(Cres <- wmwTest(exprs, ind, alternative="less"))

   user  system elapsed 
  0.135   0.000   0.135 

> wmwTestR <- function(matrix, index, alternative, stat) {
+   sub <- rep(FALSE, length(matrix))
+   sub[index]=TRUE
+   return(wilcox.test(matrix[sub], matrix[!sub], alternative=alternative)$p.value)
+ }
> system.time(Rres <- apply(exprs, 2, function(x)
+                           sapply(ind, function(y)
+                                  wmwTestR(x, y, alternative="less", stat=FALSE))))

   user  system elapsed 
  3.171   0.028   3.199 

>
```

With 22000 genes, five samples, and five gene sets, the BioQC implementation is about 20x faster than the R implementation (dependent on individual machines and settings). Our benchmark shows that with the same number of genes, 2000 samples and 200 gene sets (similar to the total number of tissues collected in the BioQC signature list), the BioQC implementation can be about 1000x faster than the R implementation.

4 Session Info

The script runs within the following session:

```
R Under development (unstable) (2014-05-31 r65803)
Platform: x86_64-unknown-linux-gnu (64-bit)
```

```
locale:
[1] C

attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods  
[8] base     

other attached packages:
[1] BioQC_1.1-5         Rcpp_0.11.1         Biobase_2.22.0      BiocGenerics_0.8.0

loaded via a namespace (and not attached):
[1] tools_3.2.0
```