HowTo: get pretty HTML output for my gene list

James W. MacDonald

October 13, 2014

1 Overview

The intent of this vignette is to show how to make reasonably nice looking HTML tables for presenting the results of a microarray analysis. These tables are a very nice format because you can insert clickable links to various public annotation databases, which facilitates the downstream analysis. In addition, the format is quite compact, can be posted on the web, and can be viewed using any number of free web browsers. One caveat; an HTML table is probably not the best format for presenting the results for *all* of the genes on a chip. For even a small (5000 gene) chip, the file could be 10 Mb or more, which would take an inordinate amount of time to open and view. Also note that the Bioconductor project supplies annotation packages for many of the more popular Affymetrix chips, as well as for many commercial spotted cDNA chips. For chips that have annotation packages, the *annaffy* package is the preferred method for making HTML tables.

To make an annotated HTML table, the only requirement is that we have some sort of annotation data for the microarray that we are using. Most manufacturers supply data in various formats that can be read into *R*. For instance, Affymetrix supplies CSV files that can be read into *R* using the `read.csv()` function `http://www.affymetrix.com/support/technical/byproduct.affx?cat=arrays`.

2 Alternate methods

Please note that one can also make these HTML tables by parsing data from e.g., an online (or local) Biomart database, using functions in the biomaRt package. This may be easier, and may result in more current annotation data. Please see the prettyOutput vignette in the biomaRt package for more information.

3 Data Analysis

I will assume that the reader is familiar with the analysis of microarray data, and has a set of genes that she would like to use. In addition, I will assume that the reader is familiar enough with *R* that she can subset the data based on a list of genes, and reorder based on a particular statistic. For any questions about subsetting or ordering data, please see "An Introduction to R". For questions regarding microarray analysis, please consult the vignettes for, say *limma*, *multtest*, or *marray*.

4 Getting Started

We first load the *annotate* package, as well as some data. These data will be from the Affymetrix HG-U95Av2 chip (for which we would normally use *annaffy*). To keep the HTML table small, we will take a subset of fifteen genes as an example.

```
> library("annotate")
> data(sample.ExpressionSet)
> igenes <- featureNames(sample.ExpressionSet)[246:260]
```

5 Annotation Data

For this vignette I have supplied the annotation data. In a normal situation, these data would be subset from the manufacturer's annotation data, using the manufacturer's gene identifiers (which is how I got these IDs).

First, we will look at the GenBank and LocusLink IDs. We will be able to use these IDs without further modification. Note that the LocusLink IDs contain some missing data ("—"). This will not pose a problem because LocusLink IDs are all numeric, so we have incorporated code in `htmlpage()` to automatically convert any non-numeric ID to an HTML empty cell character (" "). GenBank IDs (which often correspond to either RefSeq or GenBank IDs) are not as consistent, so any missing data would have to be manually converted to the HTML empty cell character. Missing data for LocusLink, UniGene and OMIM IDs are automatically converted, whereas Affymetrix, SwissProt and GenBank IDs have to be done manually. I will give examples of how to do this below.

```
> gb
```

```
 [1] "M57423"   "Z70218"   "L17328"   "S81916"   "U63332"   "M77235"
 [7] "X98175"   "AB019392" "J03071"   "D25272"   "D63789"   "D63789"
[13] "U19142"   "U19147"   "X16863"
```

```
> ll
```

```
 [1] "221823" "4330"   "9637"   "---"    "---"    "6331"   "841"
 [8] "27335"  "---"    "---"    "6375"   "---"    "2543"   "2578"
[15] "2215"
```

The UniGene and SwissProt IDs present different challenges, so we will modify them separately. For the UniGene IDs we need to strip off the extra information appended to each ID. If we didn't do this, our hyperlink would not work correctly.

```
> ug
```

```
 [1] "Hs.169284 // ---"          "Hs.268515 // full length"
 [3] "Hs.103419 // full length" "Hs.380429 // ---"
 [5] "--- // ---"                "Hs.169331 // full length"
 [7] "Hs.381231 // full length" "Hs.283781 // full length"
 [9] "--- // ---"                "--- // ---"
[11] "Hs.3195 // full length"    "--- // ---"
[13] "Hs.176660 // full length" "Hs.272484 // full length"
[15] "Hs.372679 // full length"
```

```
> ug <- sub(" //.*$", "", ug)
> ug
```

```
 [1] "Hs.169284" "Hs.268515" "Hs.103419" "Hs.380429" "---"
 [6] "Hs.169331" "Hs.381231" "Hs.283781" "---"       "---"
[11] "Hs.3195"   "---"       "Hs.176660" "Hs.272484" "Hs.372679"
```

The SwissProt IDs present a different challenge. Here there isn't any extra information. Instead, we have multiple IDs for some of the genes, and missing IDs for some of the others. Because the code for SwissProt IDs will not automatically handle missing data, we have to convert the missing data to an HTML empty cell identifier (" "). For `htmlpage()` to correctly handle multiple IDs, we have to convert the character vector into a *list* of character vectors.

```
> sp
```

```
 [1] "P21108"
 [2] "Q10571"
 [3] "Q9UHY8"
 [4] "Q16444"
 [5] "---"
 [6] "Q14524 /// Q8IZC9 /// Q8WTQ6 /// Q8WWN5 /// Q96J69"
 [7] "Q14790"
 [8] "Q9UBQ5"
 [9] "---"
[10] "---"
[11] "P47992"
[12] "---"
[13] "Q13065 /// Q8IYC5"
[14] "Q13070"
[15] "O75015"

> sp <- strsplit(sub("---"," ",as.character(sp)), "///")
> sp

[[1]]
[1] "P21108"

[[2]]
[1] "Q10571"

[[3]]
[1] "Q9UHY8"

[[4]]
[1] "Q16444"

[[5]]
[1] " "

[[6]]
[1] "Q14524 "  " Q8IZC9 " " Q8WTQ6 " " Q8WWN5 " " Q96J69"

[[7]]
[1] "Q14790"
```

```
[[8]]
[1] "Q9UBQ5"

[[9]]
[1] " "

[[10]]
[1] " "

[[11]]
[1] "P47992"

[[12]]
[1] " "

[[13]]
[1] "Q13065 " " Q8IYC5"

[[14]]
[1] "Q13070"

[[15]]
[1] "O75015"
```

We have converted the data to a list of character vectors, and also converted the "—" missing data identifier to the HTML character for an empty cell.

6 Build the Table

Usually we would like to include the expression values for our genes along with some statistics, say a t-statistic, fold change, and p-value. As an example, we will make a comparison using the first ten samples.

```
> dat <- exprs(sample.ExpressionSet)[igenes,1:10]
> FC <- rowMeans(dat[igenes,1:5]) - rowMeans(dat[igenes,6:10])
> pval <- esApply(sample.ExpressionSet[igenes,1:10], 1, function(x) t.test(x[1:5], x[
> tstat <- esApply(sample.ExpressionSet[igenes,1:10], 1, function(x) t.test(x[1:5], x
```

It is also usually a good idea to include gene names in the table. Normally the names would be subsetted from the annotation data, but here we have

to supply them. Again, we have to manually convert any missing names to the HTML empty cell character.

```
> name
```

```
 [1] "hypothetical protein LOC221823"
 [2] "meningioma (disrupted in balanced translocation) 1"
 [3] "fasciculation and elongation protein zeta 2 (zygin II)"
 [4] "Phosphoglycerate kinase {alternatively spliced}"
 [5] "---"
 [6] "sodium channel, voltage-gated, type V, alpha polypeptide"
 [7] "caspase 8, apoptosis-related cysteine protease"
 [8] "muscle specific gene"
 [9] "---"
[10] "---"
[11] "chemokine (C motif) ligand 1"
[12] "---"
[13] "G antigen 1"
[14] "G antigen 6"
[15] "Fc fragment of IgG, low affinity IIIb, receptor for (CD16)"
```

```
> name <- gsub("---", " ", name)
> name
```

```
 [1] "hypothetical protein LOC221823"
 [2] "meningioma (disrupted in balanced translocation) 1"
 [3] "fasciculation and elongation protein zeta 2 (zygin II)"
 [4] "Phosphoglycerate kinase {alternatively spliced}"
 [5] " "
 [6] "sodium channel, voltage-gated, type V, alpha polypeptide"
 [7] "caspase 8, apoptosis-related cysteine protease"
 [8] "muscle specific gene"
 [9] " "
[10] " "
[11] "chemokine (C motif) ligand 1"
[12] " "
[13] "G antigen 1"
[14] "G antigen 6"
[15] "Fc fragment of IgG, low affinity IIIb, receptor for (CD16)"
```

We can now build our HTML table. To make the process more transparent, this will be done in steps. In practice however, this can be done in

one line. Note here that the genelist consists of annotation data that will be hyperlinked to online databases, whereas othernames consists of other data that will not be hyperlinked.

```
> genelist <- list(igenes, ug, ll, gb, sp)
> filename <- "Interesting_genes.html"
> title <- "An Artificial Set of Interesting Genes"
> othernames <- list(name, round(tstat, 2), round(pval, 3), round(FC, 1), round(dat,
> head <- c("Probe ID", "UniGene", "LocusLink", "GenBank", "SwissProt", "Gene Name",
+           "Fold Change", "Sample 1", "Sample 2", "Sample 3", "Sample 4", "Sample 5"
+           "Sample 7", "Sample 8", "Sample 9", "Sample 10")
> repository <- list("affy", "ug", "en", "gb", "sp")
> htmlpage(genelist, filename, title, othernames, head, repository = repository)
```

7 Session Information

The version number of R and packages loaded for generating the vignette were:

```
R version 3.1.1 Patched (2014-09-25 r66681)
Platform: x86_64-apple-darwin13.1.0 (64-bit)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
 [1] grid      stats4    parallel  stats     graphics  grDevices
 [7] utils     datasets  methods   base

other attached packages:
 [1] GO.db_3.0.0          hgu95av2.db_3.0.0    org.Hs.eg.db_3.0.0
 [4] RSQLite_0.11.4       DBI_0.3.1            Rgraphviz_2.10.0
 [7] graph_1.44.0         xtable_1.7-4         annotate_1.44.0
[10] XML_3.98-1.1         AnnotationDbi_1.28.0 GenomeInfoDb_1.2.0
[13] IRanges_2.0.0        S4Vectors_0.4.0      Biobase_2.26.0
[16] BiocGenerics_0.12.0

loaded via a namespace (and not attached):
[1] tools_3.1.1
```