Annotation for Microarrays

Cavan Reilly

October 23, 2019

Table of contents

Overview

Gene Ontology

- Hypergeometric tests

Biomart

Database versions of annotation packages

Overview

We've seen that one can use tools in R to get useful information about the identity of genes that are represented on some microarrays.

There are also connections to databases that have other sorts of biological information.

This can help to interpret the results of an analysis that tests for differential gene expression.

There are a number of online resources that can be consulted for basic data about genomes.

Overview

A group I work with has interests in the relationship between certain nucleotide sequences in the 3' end of mRNAs and gene expression.

A postdoc in the lab a number of years ago downloaded a data set with nucleotide sequences from the 3' end of human transcripts.

As improvements to the annotation of the human genome occur all the time, we would like to update this data set periodically.

Overview

Websites constantly change, so even if the postdoc kept a record of what links he followed, that would quickly be out of date.

A much better approach: write a script that accesses information from biological databases.

This is straightforward to maintain and much more reproducible.

R has extensive capabilities to obtain and manage these sorts of data sets.

Overview

To explore this we will compare gene expression between 2 subtypes of ALL: the ALL1/AF4 mutation and the BCR/ABL mutation.
To start we will set up the dataset, much as we've seen previously

```
> library(ALL)
> data(ALL)
> types=c("ALL1/AF4", "BCR/ABL")
> bcell=grep("^B", as.character(ALL$BT))
> ALL_af4bcr=ALL[,intersect(bcell,
+  which(ALL$mol.biol %in% types))]
> ALL_af4bcr$mol.biol=factor(ALL_af4bcr$mol.biol)
```

Overview

The groups here are of different size so if we filter based on the IQR as before the calculation of variation will be dominated by the larger group

```
> table(ALL_af4bcr$mol.biol)

ALL1/AF4  BCR/ABL
      10       37
```

Overview

To avoid this we use a more extreme measure of the spread of the distribution: the extreme deciles.

To do this, we will define our own variance function and supply this to the `nsFilter` function that we've already seen.

Now our filtering excludes probes that don't have an Entrez gene ID or data from the gene ontology project about biological process (more on this later).

```
> qrange=function(x) diff(quantile(x, c(0.1,0.9)))
> library(genefilter)
> filt_af4bcr=nsFilter(ALL_af4bcr, require.entrez=T,
+  require.GOBP=T, var.func=qrange, var.cutoff=0.5)
> ALLfilt_af4bcr=filt_af4bcr$eset
```

Overview

We can examine the log from `nsFilter` to see the effect of filtering

```
> filt_af4bcr$filter.log
$numDupsRemoved
[1] 2747

$numLowVar
[1] 4033

$numRemoved.ENTREZID
[1] 1151

$numNoGO.BP
[1] 643

$feature.exclude
[1] 19
```

Overview

Then we load some packages and conduct some tests for differences between the 2 groups.

```
> library(Biobase)
> library(annotate)
Loading required package: XML
> library(hgu95av2.db)
> rt=rowttests(ALLfilt_af4bcr,"mol.biol")
> sum(p.adjust(rt$p,method="holm")<.05)
[1] 134
> sum(p.adjust(rt$p,method="BH")<.05)
[1] 587
```

Overview

We can obtain information about these genes using the get or mget functions as follows

```
> featureNames(ALLfilt_af4bcr)[p.adjust(rt$p,
+  method="holm")<.05]
  [1] "266_s_at"   "35831_at"   "41202_s_at" "41742_s_at"
...

> get("266_s_at",env=hgu95av2CHRLOC)
        Y          Y          Y          Y
-21152525 -21152525 -21152525 -21152525

> get("35831_at",env=hgu95av2CHRLOC)
      20
-50213313
```

Overview

This means that the gene with Affymetrix probe id 266_s_at resides on the Y chromosome on the negative strand about 21 million basepairs from the 5' end.

We use `mget` to retrieve this information for more than 1 probe set.

```
> mget(c("266_s_at","35831_at"),env=hgu95av2CHRLOC)
$`266_s_at`
        Y          Y          Y          Y
-21152525 -21152525 -21152525 -21152525

$`35831_at`
       20
-50213313
```

Why would a probe set return multiple start locations?

Overview

Alternatively we can use the `column` and `select` functions to get this information

```
> columns(hgu95av2.db)
> keys=head(keys(hgu95av2.db))
> select(hgu95av2.db,keys=keys,
+  columns=c("SYMBOL", "UNIGENE"))
```

Overview

With a few more tricks we can find out general properties of these genes, for example on which chromosome are they located.

```
> nm=names(unlist(mget(featureNames(ALLfilt_af4bcr)[p.adjust(rt$p,
+   method="holm")<.05],env=hgu95av2CHRLOC)))
> table(substr(nm,regexpr("\\.",nm)+1,200))
```

1	10	11	12	13	14
17	5	14	8	6	13
15	16	17	18	19	2
3	12	6	4	6	13
20	21	22	3	4	5
5	3	6	9	13	5
6	6_apd_hap1	6_cox_hap2	6_dbb_hap3	6_mann_hap4	6_mcf_hap5
19	8	10	10	10	10
6_qbl_hap6	6_ssto_hap7	7	8	9	X
10	10	22	5	10	4
Y					
4					

What's on chromosome 6? MHC region.

Gene Ontology

The gene ontology (GO) project is large project that attempts to classify genes and transcripts in terms of

1. Biological process
2. Molecular function
3. Cellular component

It does this in a hierarchical fashion, so that there are very general categories (e.g. mitochondrion or metabolic process) and very specific ones (e.g. oxidoreductase activity, acting on the CH-OH group of donors, NAD or NADP as acceptor).

All GO terms have a 7 digit number preceded by the letters GO.

Gene Ontology

We can determine relationships among terms using the `PARENT`, `CHILDREN` and `OFFSPRING` mappings.

```
> library(GO.db)
> as.list(GOMFCHILDREN["GO:0008094"])
$`GO:0008094`
      is_a       is_a       is_a       is_a       is_a
"GO:0003918" "GO:0004003" "GO:0015616" "GO:0033170" "GO:0033676"
      is_a       is_a
"GO:0033680" "GO:0043142"
> as.list(GOMFOFFSPRING["GO:0008094"])
$`GO:0008094`
 [1] "GO:0003689" "GO:0003918" "GO:0004003" "GO:0015616" "GO:0017116"
 [6] "GO:0033170" "GO:0033676" "GO:0033680" "GO:0033681" "GO:0033682"
[11] "GO:0036121" "GO:0039631" "GO:0043140" "GO:0043141" "GO:0043142"
[16] "GO:1990163" "GO:1990518"
```

Hypergeometric tests

Given our list of genes that differ we can test if there are classes of transcripts that are found in the list more frequently than we expect by chance.

This has been addressed by considering 2 by 2 tables and testing for an association using Fisher's exact test (which uses the hypergeometric distribution to compute the p-value).

We can use the `GOstats` package to do this: we must define the universe of genes that we are examining, get their Entrez gene ID, and get the Entrez IDs for our gene list.

Hypergeometric tests

We then create a `GOHyperGParams` object that holds the necessary information and provide a p-value cutoff.

We also need to specify if we want to use conditional or unconditional tests.

The conditional tests try to account for the massive and difficult to account for multiple hypothesis testing that is implicit in this analysis.

Hypergeometric tests

Here is the syntax to accomplish this

```
> BiocManager::install("GOstats")
> affyUniverse=featureNames(ALLfilt_af4bcr)
> uniId=hgu95av2ENTREZID[affyUniverse]
> entrezUniverse=unique(as.character(uniId))
> EGsub=featureNames(ALLfilt_af4bcr)[p.adjust(rt$p,
+  method="holm")<.05]
> EGsub1=as.character(hgu95av2ENTREZID[EGsub])
> params=new("GOHyperGParams",geneIds=EGsub1,
+  universeGeneIds=entrezUniverse,annotation=
+  "hgu95av2",ontology="BP", pvalueCutoff=0.001,
+  conditional=F, testDirection="over")
> mfhyper=hyperGTest(params)
```

Hypergeometric tests

We can visualize the results by looking at a histogram of results and generate an html report.

```
> hist(pvalues(mfhyper), col="mistyrose",
+  xlab=expression(italic(p)-values), main="")
> htmlReport(mfhyper,"ALL_GO_BP_summary.html")
```

Hypergeometric tests

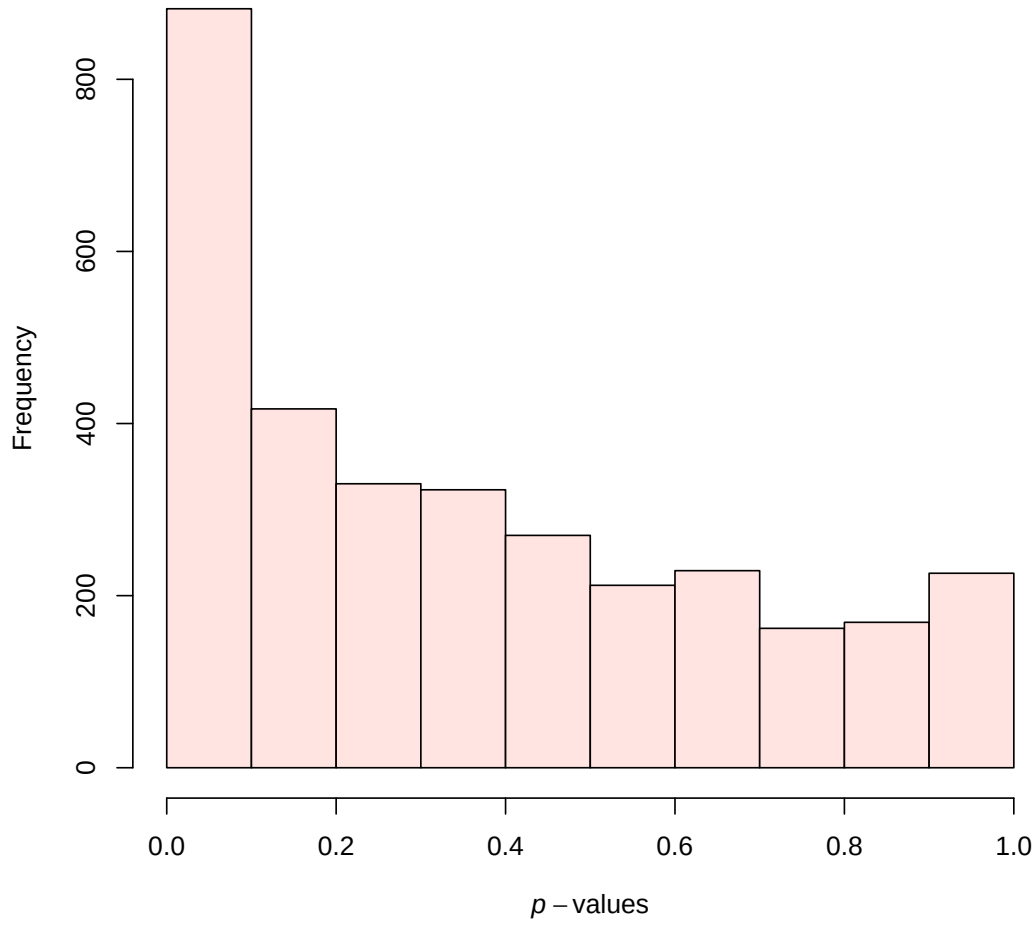

Figure: The p-value distribution for the hypergeometric tests ALL data set.

Hypergeometric tests

Unfortunately the links in the report don't work as is, but this is easy to fix in principle.

The problem is that GO changed the url: we need to replace

`http://www.godatabase.org/cgi-bin/amigo/go.cgi?view=details&search_constraint=terms&depth=0&query=`

with

`http:amigo.geneontology.org/amigo/term/`

Hypergeometric tests

which can be accomplished with code that uses tricks like the following (excusing the line break in sub: you can let commands wrap around the terminal window).

```
rpt=scan("ALL_GO_BP_summary.html",what="",sep="\n")
rpt1=rep(NA,length(rpt))
for(i in 1:length(rpt)) rpt1[i]=gsub('\\\"','"',rpt[i])
rpt2=rep(NA,length(rpt))
for(i in 1:length(rpt))
rpt2[i]=sub("http://www.godatabase.org/cgi-bin/amigo/go.cgi\\?view=details\\&search_constraint=terms\\

&depth=0\\&query=",

+  "http:amigo.geneontology.org/amigo/term/",rpt1[i])
write(rpt2,"ALL_GO_BP_summary1.html")
```

Biomart

Biomart is an online resource for obtaining a wide variety of data of interest to molecular biologists.

There is a convenient R interface that lets one get this data through R.

This is better than having to download over a browser as it can be documented, however if you work with others who are downloading from the site you could have different data sets.

The R package `biomaRt` allows for this functionality.

Biomart

Here is the usage: first you select a mart, then a data set.

```
> head(listMarts())
                biomart                 version
1 ENSEMBL_MART_ENSEMBL      Ensembl Genes 86
2   ENSEMBL_MART_MOUSE      Mouse strains 86
3     ENSEMBL_MART_SNP  Ensembl Variation 86
4 ENSEMBL_MART_FUNCGEN Ensembl Regulation 86
5    ENSEMBL_MART_VEGA                Vega 66
> mart=useMart("ensembl")
```

Biomart

```
> head(listDatasets(mart))
                         dataset                                  description
1         oanatinus_gene_ensembl     Ornithorhynchus anatinus genes (OANA5)
2        cporcellus_gene_ensembl            Cavia porcellus genes (cavPor3)
3        gaculeatus_gene_ensembl     Gasterosteus aculeatus genes (BROADS1)
4 itridecemlineatus_gene_ensembl Ictidomys tridecemlineatus genes (spetri2)
5         lafricana_gene_ensembl         Loxodonta africana genes (loxAfr3)
6        choffmanni_gene_ensembl        Choloepus hoffmanni genes (choHof1)
  version
1   OANA5
2 cavPor3
3 BROADS1
4 spetri2
5 loxAfr3
6 choHof1
```

Biomart

Then we select a data set (organism specific molecular biology data)

```
> ensembl=useDataset("hsapiens_gene_ensembl", mart=mart)
```

Once we've selected a dataset then we can see what is available for specific tasks, here we look for Affymetrix annotations.

Biomart

```
> listFilters(ensembl)[grep("Affy",listFilters(ensembl)[,2]),2]
 [1] "with Affymetrix Microarray huex 1 0 st v2 probeset ID(s)"
 [2] "with Affymetrix Microarray hc g110 probeset ID(s)"
 [3] "with Affymetrix Microarray hg Focus probeset ID(s)"
 [4] "with Affymetrix Microarray u133 x3p probeset ID(s)"
 [5] "with Affymetrix Microarray hg u133a probeset ID(s)"
 [6] "with Affymetrix Microarray hg u133a 2 probeset ID(s)"
 [7] "with Affymetrix Microarray hg u133 plus 2 probeset ID(s)"
 [8] "with Affymetrix Microarray hg u133b probeset ID(s)"
 [9] "with Affymetrix Microarray hg u95a probeset ID(s)"
[10] "with Affymetrix Microarray hg u95av2 ID(s) probeset"
...
```

Biomart

You can also look at what's available based on the attributes:

```
> listAttributes(ensembl)[grep("Tasmanian devil",
+  listAttributes(ensembl)[,2]),]
```

Biomart

You can retrieve information using the `getBM` function.

Here is how we could get the ensembl gene IDs and the GO IDs for our genes that appear to differ.

```
> hguIDs=getBM(attributes=c("ensembl_gene_id","go_id"),
+  filters="affy_hg_u95av2",values=EGsub,mart=ensembl)
> dim(hguIDs)
[1] 3004    2
```

Database versions of annotation packages

To extend the functionality and increase the speed of annotation queries, Bioconductor now uses connections to external SQLite databases to access annotation information.

All existing annotation packages have a database version, and the only difference in the names is that the database versions have a `db` at the end:

`hgu95av2` is replaced by `hgu95av2.db`.

To use these packages, you first establish a connection to the database, then run queries.

Database versions of annotation packages

Here is the syntax

```
> library(hgu133a.db)
> dbc=hgu133a_dbconn()
> get("201473_at",hgu133aSYMBOL)
[1] "JUNB"
> as.character(hgu133aSYMBOL["201473_at"])
201473_at
   "JUNB"
> hgu133aSYMBOL[["201473_at"]]
[1] "JUNB"
```

Database versions of annotation packages

Here is how to make a table of number of terms in each of the sets of GO categories.

```
> query="select ontology from go_term"
> goCats=dbGetQuery(GO_dbconn(), query)
> gCnums=table(goCats)[c("BP","CC","MF")]
> gCnums
goCats
   BP    CC    MF
28007  3827  9955
```

Database versions of annotation packages

This hints at the sort of more complex queries that are possible: here we search for genes on the U133 array that have the term transcription factor binding in their GO description.

```
> query=paste("select go_id from go_term where",
+   "term = 'transcription factor binding'")
> tfb=dbGetQuery(GO_dbconn(), query)
> tfbps=hgu133aGO2ALLPROBES[[tfb$go_id]]
> table(names(tfbps))

IBA IDA IEA IGI IPI ISS NAS TAS
 53 123 188   1 416  89  48  36
```

Database versions of annotation packages

One can also do reverse searches, for example finding Affymetrix probe set IDs from gene symbols.

```
> s1=revmap(hgu133aSYMBOL)
> s1$BCR
[1] "202315_s_at" "217223_s_at"
```