Gene set and data preparation

Weijun Luo (luo_weijun AT yahoo.com)

February 18, 2016

1 Introduction

In this short tutorial, we cover a few practical issues we frequently come cross in GAGE (**?**) and other types of gene set analysis. These issues include: gene set and expression data input, probe set ID, transcript and gene ID conversion. This tutorial is written for those who are less familiar with R/Bioconductor basics.

First we get started as decribed in the main vignette. Under R, first we load the *gage* package:

```
> library(gage)
```

2 Expression data input

The gage package provides dedicated functions for data input, partically reading in the experimental or gene expression data and gene set data. First, let's look at how to read in a gene expression dataset in tab-deliminated text format. You may open it in Excel for a better view. Now that genes are rows and samples are columns.

```
> filename=system.file("extdata/gse16873.demo", package = "gage")
> demo.data=readExpData(filename, row.names=1)
> #check the data
> head(demo.data)
```

```
             HN_1   DCIS_1     HN_2   DCIS_2     HN_3   DCIS_3     HN_4
10000     6.765984 6.458339 6.921720 6.774493 7.010564 6.986779 6.958761
10001     6.339474 6.755342 7.177369 6.842597 7.392611 6.879474 6.296571
10002     6.591755 6.790304 6.735359 6.773255 6.700016 7.041881 6.586285
10003     6.822092 6.590539 6.508452 6.411859 6.575640 6.470913 6.896886
100048912 7.356051 7.311144 7.385513 7.333481 7.392233 7.428623 7.314579
10004     6.941935 6.854373 6.883973 6.833695 6.855043 6.856864 7.021072
           DCIS_4     HN_5   DCIS_5     HN_6   DCIS_6
```

```
10000     6.888199 6.949912 6.948589 7.220427 7.070159
10001     6.130034 7.831222 7.942344 7.665248 7.799255
10002     6.501011 6.884931 7.652490 7.342505 7.500791
10003     6.848872 6.615143 6.407087 6.618174 6.651619
100048912 7.362658 7.370044 7.397249 7.383196 7.437643
10004     7.051310 6.767242 6.775276 6.873855 6.805247
```

```
> str(demo.data)
```

```
'data.frame':        100 obs. of  12 variables:
 $ HN_1  : num  6.77 6.34 6.59 6.82 7.36 ...
 $ DCIS_1: num  6.46 6.76 6.79 6.59 7.31 ...
 $ HN_2  : num  6.92 7.18 6.74 6.51 7.39 ...
 $ DCIS_2: num  6.77 6.84 6.77 6.41 7.33 ...
 $ HN_3  : num  7.01 7.39 6.7 6.58 7.39 ...
 $ DCIS_3: num  6.99 6.88 7.04 6.47 7.43 ...
 $ HN_4  : num  6.96 6.3 6.59 6.9 7.31 ...
 $ DCIS_4: num  6.89 6.13 6.5 6.85 7.36 ...
 $ HN_5  : num  6.95 7.83 6.88 6.62 7.37 ...
 $ DCIS_5: num  6.95 7.94 7.65 6.41 7.4 ...
 $ HN_6  : num  7.22 7.67 7.34 6.62 7.38 ...
 $ DCIS_6: num  7.07 7.8 7.5 6.65 7.44 ...
```

```
> #convert the data.frame into a matrix as to speed up the computing
> demo.data=as.matrix(demo.data)
> str(demo.data)
```

```
 num [1:100, 1:12] 6.77 6.34 6.59 6.82 7.36 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:100] "10000" "10001" "10002" "10003" ...
  ..$ : chr [1:12] "HN_1" "DCIS_1" "HN_2" "DCIS_2" ...
```

Then it is ready for running GAGE analysis on demo.data. Please check the main vignette: `Generally Applicable Gene-set/Pathway Analysis`, for details. Please do not actually run GAGE analysis on demo.data as it only has 100 genes, too few for any meaningful analysis.

3 Gene set data input

We may download gene set definitation from third-party databases like MSigDB in "GMT" format, or we may prepare our own gene set data in that format. Please check file

"c2.demo.gmt" in the "extdata" folder for details. We may read in such gene set data using readList function. You may also modify the function to accomodate your own gene set data format.

```
> #an example GMT gene set data derived from MSigDB data
> filename=system.file("extdata/c2.demo.gmt", package = "gage")
> demo.gs=readList(filename)
> demo.gs[1:3]

$`3AB_GAMMA_DN`
 [1] "ANXA3"     "BNIP2"     "COLQ"      "CSAD"      "CSNK1A1"   "DDX3Y"
 [7] "DPAGT1"    "FEN1"      "LOC653689" "MED31"     "NAT12"     "PTK9L"
[13] "THOC4"

$`4NQO_ESR_OLD_UNREG`
 [1] "EIF3S6IP" "FLOT1"    "GNPNAT1"  "HGF"      "LCK"      "NDRG1"
 [7] "SLC18A2"  "SOX9"     "SYK"      "TAF1C"    "TP53I11"

$`4NQO_ESR_WS_UNREG`
 [1] "ARPC1B"   "ATP5I"    "CEPT1"    "CNKSR1"   "COL17A1"  "CTCF"
 [7] "CTGF"     "FLOT1"    "FUT4"     "GABBR1"   "GALNT2"   "GNPNAT1"
[13] "HADH2"    "HGF"      "HLA-DQA1" "HNRPC"    "ITGB8"    "KIAA0146"
[19] "LCK"      "LRRFIP2"  "NDRG1"    "PDCD5"    "PRDX4"    "RHOB"
[25] "SGCB"     "SGPL1"    "SOX7"     "SOX9"     "SYK"      "TAF1C"
[31] "TP53I11"  "TTC19"    "TXNRD2"   "UGCG"     "USP1"     "ZNF507"

> #to use these gene sets with gse16873, need to convert the gene symbols
> #to Entrez IDs first
> data(egSymb)
> demo.gs.sym<-lapply(demo.gs, sym2eg)
> demo.gs.sym[1:3]

$`3AB_GAMMA_DN`
 [1] "306"    "663"    "8292"   "51380"  "1452"   "8653"   "1798"   "2237"
 [9] NA       "51003"  "122830" NA       "10189"

$`4NQO_ESR_OLD_UNREG`
 [1] NA      "10211" "64841" "3082"  "3932"  "10397" "6571"  "6662"  "6850"
[10] "9013"  "9537"

$`4NQO_ESR_WS_UNREG`
 [1] "10095" "521"   "10390" "10256" "1308"  "10664" "1490"  "10211" "2526"
```

```
[10] "2550"  "2590"  "64841" NA      "3082"  "3117"  NA      "3696"  "23514"
[19] "3932"  "9209"  "10397" "9141"  "10549" "388"   "6443"  "8879"  "83595"
[28] "6662"  "6850"  "9013"  "9537"  "54902" "10587" "7357"  "7398"  "22847"
```

4 Probe set ID conversion

Gene set or pathway analysis requires that gene sets and expression data use the same type of gene ID (Entrez Gene ID, Gene symbol or probe set ID etc). However, this is frequently not true for the data we have. For example, our gene sets mostly use Entrez Gene ID, but microarray datasets are frequently labeled by probe set ID (or RefSeq transcript ID etc). Therefore, we need to convert or map the probe set IDs to Entrez gene ID. The support data package *gageData* (the latest version: >= 2.0.0)provides a example microarray dataset with Affymetrix probe set IDs (labels with `_at` suffix), which are seen in most Afymetrix microarray datasets. From GEO GSE16873 record page or the CEL data file, we know that it uses Affymetrix Human Genome U133A Array. We need to use the corresponding Bioconductor annotation package, hgu133a.db. Of course, you need to have that installed first.

```
> library(gageData)
> data(gse16873.affyid)
> affyid=rownames(gse16873.affyid)
> library(hgu133a.db)
> egids2=hgu133aENTREZID[affyid]
> annots=toTable(egids2)
> str(annots)

'data.frame':        19609 obs. of  2 variables:
 $ probe_id: chr  "1053_at" "117_at" "121_at" "1255_g_at" ...
 $ gene_id : chr  "5982" "3310" "7849" "2978" ...

> gse16873.affyid=gse16873.affyid[annots$probe_id,]
> #if multiple probe sets map to a gene, select the one with maximal IQR
> iqrs=apply(gse16873.affyid, 1, IQR)
> sel.rn=tapply(1:nrow(annots), annots$gene_id, function(x){
+ x[which.max(iqrs[x])]
+ })
> gse16873.egid=gse16873.affyid[sel.rn,]
> rownames(gse16873.egid)=names(sel.rn)
> cn=colnames(gse16873.egid)
> hn=grep('HN',cn, ignore.case =T)
> dcis=grep('DCIS',cn, ignore.case =T)
```

```
> data(kegg.gs)
> gse16873.kegg.p.affy <- gage(gse16873.egid, gsets = kegg.gs,
+     ref = hn, samp = dcis)
> #result should be similar to that of using gse16873
```

5 gene or transcript ID conversion

Frequently, we need to convert other types of gene/transcript IDs to Entrez Gene ID or the reverse. The *gage* package provides functions eg2sym and sym2eg for such ID conversions on human genes, which uses a integrated ID mapping matrix, to access it do: data(egSymb); head(egSymb). The *pathview* package (**?**) provides two more comprehensive functions: eg2id and id2eg, which make use of the Bioconductor gene annotation packages or similar custom annotation pakages. These functions not only cover the latest gene annotations but also convert gene IDs for many common model organisms, for a list of model organisms and corresponding Bioconductor gene annotation packages, check data(bods) as below. These functions derive a ID mapping matrix, which then can be used to map the data in a seperate step because multiple transcript ID may map to the same Entrez Gene ID.

```
> library(pathview)
> data(bods)
> print(bods)
```

```
      package             species               kegg code id.type
 [1,] "org.Ag.eg.db"      "Anopheles"           "aga"     "eg"
 [2,] "org.At.tair.db"    "Arabidopsis"         "ath"     "tair"
 [3,] "org.Bt.eg.db"      "Bovine"              "bta"     "eg"
 [4,] "org.Ce.eg.db"      "Worm"                "cel"     "eg"
 [5,] "org.Cf.eg.db"      "Canine"              "cfa"     "eg"
 [6,] "org.Dm.eg.db"      "Fly"                 "dme"     "eg"
 [7,] "org.Dr.eg.db"      "Zebrafish"           "dre"     "eg"
 [8,] "org.EcK12.eg.db"   "E coli strain K12"   "eco"     "eg"
 [9,] "org.EcSakai.eg.db" "E coli strain Sakai" "ecs"     "eg"
[10,] "org.Gg.eg.db"      "Chicken"             "gga"     "eg"
[11,] "org.Hs.eg.db"      "Human"               "hsa"     "eg"
[12,] "org.Mm.eg.db"      "Mouse"               "mmu"     "eg"
[13,] "org.Mmu.eg.db"     "Rhesus"              "mcc"     "eg"
[14,] "org.Pf.plasmo.db"  "Malaria"             "pfa"     "orf"
[15,] "org.Pt.eg.db"      "Chimp"               "ptr"     "eg"
[16,] "org.Rn.eg.db"      "Rat"                 "rno"     "eg"
[17,] "org.Sc.sgd.db"     "Yeast"               "sce"     "orf"
```

```
[18,] "org.Ss.eg.db"      "Pig"                 "ssc"     "eg"
[19,] "org.Xl.eg.db"      "Xenopus"             "xla"     "eg"
```

```
> #simulated human expression data with RefSeq ID
> refseq.data <- sim.mol.data(mol.type = "gene", id.type = "REFSEQ",
+                 nexp = 2, nmol = 1000)
> #construct map between non-Entrez ID and Entrez Gene ID
> id.map.refseq <- id2eg(ids = rownames(refseq.data), category =
+                    "REFSEQ", org = "Hs")
> #Map data onto Entrez Gene IDs, note different sum.method can be used
> entrez.data <- mol.sum(mol.data = refseq.data, id.map = id.map.refseq,
+                    sum.method = "mean")
```