Generally Applicable Gene-set/Pathway Analysis

Weijun Luo (luo_weijun AT yahoo.com)

February 18, 2016

Abstract

In this vignette, we demonstrate the *gage* package for gene set (enrichment or GSEA) or pathway analysis. The *gage* package implement the GAGE method. GAGE is generally applicable independent of microarray and RNA-Seq data attributes including sample sizes, experimental designs, assay platforms, and other types of heterogeneity, and consistently achieves superior performance over other frequently used methods. We introduce functions and data for routine and advanced gene set (enrichment) analysis, as well as results presentation and interpretation. We also cover package installation, data preparation and common application errors. Two secondary vignettes, "Gene set and data preparation" and "RNA-Seq Data Pathway and Gene-set Analysis Workflows", demonstrate more applications and usages of GAGE.

1 Cite our work

Weijun Luo, Michael Friedman, Kerby Shedden, Kurt Hankenson, and Peter Woolf. GAGE: generally applicable gene set enrichment for pathway analysis. BMC Bioinformatics, 2009. doi:10.1186/1471-2105-10-161.

2 Quick start with demo data

This is the most basic use of *gage*, please check the full description below for more information.

```
> library(gage)
> data(gse16873)
> hn=(1:6)*2-1
> dcis=(1:6)*2
```

KEGG pathway analysis. Here you need to make sure species and gene ID type are the same in gene sets (kegg.gs) and expression data (gse16873). You can check by: head(kegg.gs[[1]]); head(rownames(gse16873))

```
> data(kegg.gs)
> gse16873.kegg.p <- gage(gse16873, gsets = kegg.gs, ref = hn, samp = dcis)
```

GO analysis, separate BP, MF and CC categories

```
> library(gageData)
> data(go.sets.hs)
> data(go.subs.hs)
> gse16873.bp.p <- gage(gse16873, gsets = go.sets.hs[go.subs.hs$BP], ref = hn, samp = dcis)
> gse16873.mf.p <- gage(gse16873, gsets = go.sets.hs[go.subs.hs$MF], ref = hn, samp = dcis)
> gse16873.cc.p <- gage(gse16873, gsets = go.sets.hs[go.subs.hs$CC], ref = hn, samp = dcis)
```

3 New features

gage ($\geq$ 2.11.3): A new secondary vignette, "RNA-Seq Data Pathway and Gene-set Analysis Workflows", demonstrates applications of GAGE/Pathview workflows in RNA-Seq data analysis.

gage ($\geq$ 2.11.3): A new function, `kegg.gsets`, has been introduced in the package. You may use it to compile pathway gene set data any time needed for one of the 3000 KEGG species and KEGG Orthology (with species="ko"). In addition, you get the most updated pathway gene set data as it is retrieved from KEGG in real time. Please check the Section Basic Analysis and the help info on the function for details.

gage ($\geq$ 2.13.5): Another new function, `go.gsets`, has been introduced in the package. You may use it to compile GO (Gene Ontology) gene set data any time needed for 19 common species annotated in Bioconductor and more others by the users. Please check the Section Basic Analysis and the help info on the function for details.

4 Introduction

Gene set analysis (GSA) is a powerful strategy to infer functional and mechanistic changes from high through microarray or sequencing data. GSA focuses on sets of related genes and has established major advantages over per-gene based different expression analyses, including greater robustness, sensitivity and biological relevance. However, classical GSA methods only have limited usage to a small number of microarray or RNA-Seq studies as they cannot handle datasets of different sample sizes, experimental designs and other types of heterogeneity (**?**). To address these limitations, we present a new GSA method called Generally Applicable Gene-set Enrichment (GAGE) (**?**). We have validated GAGE method extensively against the most widely used GSA methods in both applicability and performance (**?**). In this manual, we focus on the implementation and usage of the R package, *gage*.

In *gage* package, we provide a series of functions for basic GAGE analysis, result processing and presentation. We have also built pipeline routines for of multiple GAGE analysis on different comparisons or samples, the comparison of parallel analysis results, and even the combined analysis of heterogeneous data from different sources/studies.

This package also supplies an example microarray dataset. In GAGE paper (**?**) and the old versions of *gage* package, we used a BMP6 microarray experiment as the demo data. Here we choose to use another published microarray dataset from GEO, GSE16873 (**?**), as our primary demo data, as to show more and advanced functionality and applications of GAGE with this update package. GSE16873 is a breast cancer study (**?**), which covers twelve patient cases, each with HN (histologically normal), ADH (ductal hyperplasia), and DCIS (ductal carcinoma in situ) RMA samples. Hence, there are 12*3=36 microarray hybridizations or samples interesting to us plus 4 others less interesting in this full dataset. Due to the size limit of this package, we split this GSE16873 into two halves, each including 6 patients with their HN and DCIS but not ADH tissue types. Raw data for these two halves were processed using two different methods, FARMS (**?**) and RMA (**?**), respectively to generate the non-biological data heterogeneity. This *gage* package, we only include the first half dataset for 6 patients (processed using FARMS). The second half dataset plus the full dataset (including all HN, ADH and DCIS samples of all 12 patients, processed together using FARMS) and the original BMP6 dataset is supplied with a data package, *gageData*. Most of our demo analyses will be done on the first half dataset, except for the advanced analysis where we use both halves datasets with all 12 patients. While the *gage* package provides `kegg.gsets` and `go.gsets` to generate updated KEGG pathway and GO (Gene Ontology) gene set in real time, it also includes useful human gene set data derived from KEGG pathway and GO databases. We also supply extra gene set data for other species including mouse, rat and yeast in the data package, *gageData*, available from the bioconductor website. In addition, the users may derive other own gene sets using the `kegg.gsets` and `go.gsets` functions.

The manual is written by assuming the user has minimal R/Bioconductor knowledge. Some descriptions and code chunks cover very basic usage of R. The more experienced users may simply omit these parts.

5 Installation

Assume R has been correctly installed and accessible under current directory. Otherwise, please contact your system admin or follow the instructions on R website.

Start R: from Linux/Unix command line, type `R` (Enter); for Mac or Windows GUI, double click the R application icon to enter R console.

End R: type in `q()` when you are finished with the analysis using R, but not now.

Two options:

Either install with Bioconductor installation script `biocLite` directly:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite(c("gage","gageData"))
```

Or install without using Bioconductor: Download *gage* and *gageData* packages (make sure with proper version number and zip format) and save to `/your/local/directory/`.

```
> install.packages(c("/your/local/directory/gage_2.9.1.tar.gz",
+     "/your/local/directory/gageData_1.0.0.tar.gz"),
+     repos = NULL, type = "source")
```

6 Get Started

Under R, first we load the *gage* package:

```
> library(gage)
```

To see a brief overview of the package:

```
> library(help=gage)
```

To get help on any function (say the main function, `gage`), use the `help` command in either one of the following two forms:

```
> help(gage)
> ?gage
```

7 Basic Analysis

GAGE (?) is a widely applicable method for gene set or pathway analysis. In such analysis, we check for coordinated differential expression over gene sets instead of changes of individual genes. Gene sets are pre-defined groups of genes, which are functionally related. Commonly used gene sets include those derived from KEGG pathways, Gene Ontology terms, gene groups that share some other functional annotations, including common transcriptional regulators (like transcription factors, small interfering RNA's or siRNA etc), genomic locations etc. Consistent perturbations over such gene sets frequently suggest mechanistic changes. GSA is much more robust and sensitive than individual gene analysis (?), especially considering the functional redundancy of individual genes and the noise level of high throughput assay data. GAGE method implemented in this package makes GSA feasible to all microarray and RNA-Seq studies, irrespecitve of the sample size, experiment design, array platform etc. Let's start with the most basic gene set analysis. Note that both the demo expression data and gene set data are ready here, if not, please check vignette, ”Gene set and data preparation”, for data preparation.

We load and look at the demo microarray data first. Note that although we use microarray data as example, GAGE is equally applicable to RNA-Seq data and other types of gene/protein centered high throughput data. The other vignette, ”RNA-Seq Data Pathway and Gene-set Analysis Workflows”, demonstrates such applications of GAGE.

```
> data(gse16873)
> cn=colnames(gse16873)
> hn=grep('HN',cn, ignore.case =T)
> adh=grep('ADH',cn, ignore.case =T)
> dcis=grep('DCIS',cn, ignore.case =T)
> print(hn)
```

```
[1]  1  3  5  7  9 11
```

```
> print(dcis)
```

```
[1]  2  4  6  8 10 12
```

Check to make sure your gene sets and your expression data are using the same ID system (Entrez Gene ID, Gene symbol, or probe set ID etc). Entrez Gene IDs are integer numbers, Gene symbols are characters, and probe set IDs (Affymetrix GeneChip) are characters with `_at` suffix. When gene sets and expression data do use different types of ID, please check vignette, ”Gene set and data preparation”, for ID conversion solutions.

```
> data(kegg.gs)
> data(go.gs)
> lapply(kegg.gs[1:3],head)
```

```
$`hsa00010 Glycolysis / Gluconeogenesis`
[1] "10327" "124"   "125"   "126"   "127"   "128"  

$`hsa00020 Citrate cycle (TCA cycle)`
[1] "1431" "1737" "1738" "1743" "2271" "3417"

$`hsa00030 Pentose phosphate pathway`
[1] "2203"   "221823" "226"    "229"    "22934"  "230"   
```

```
> head(rownames(gse16873))
```

```
[1] "10000"     "10001"     "10002"     "10003"     "100048912" "10004"    
```

We normally do GAGE analysis using gene sets derived from on KEGG pathways or GO term groups. Here we use the human gene set data coming with this package with the human microarray dataset GSE16873. Note that `kegg.gs` has been updated since gage version 2.9.1. From then, `kegg.gs` only include the subset of canonical signaling and metabolic pathways from KEGG pathway database, and `kegg.gs.dise` is the subset of disease pathways. And it is recommended to do KEGG pathway analysis with either `kegg.gs` or `kegg.gs.dise` seperately (rather than combined altogether) for better defined results. `go.gs` derived from human GO database only includes 1000 gene sets due to size limit. For full `go.gs` or gene sets data for other species, we may always use the *gageData* package. If you don't find the gene set data for your target species, you may use `kegg.gsets` or `go.gsets` to compile pathway gene set data any time needed as long as it is one of the 3000 KEGG species or a species with gene annotation package supplied in Bioconductor or the user. You may want to save the new gene set data for later use. An another advantage of using `kegg.gsets` is that you get the most updated pathway gene set data as it is retrieved from KEGG in real time.

```
> kg.hsa=kegg.gsets("hsa")
> kegg.gs=kg.hsa$kg.sets[kg.hsa$sigmet.idx]
> #save(kegg.gs, file="kegg.hsa.sigmet.gsets.RData")

> #kegg.gsets works with 3000 KEGG species,for examples:
> data(korg)
> head(korg[,1:3])

     kegg.code scientific.name           common.name
[1,] "hsa"     "Homo sapiens"            "human"
[2,] "ptr"     "Pan troglodytes"         "chimpanzee"
[3,] "pps"     "Pan paniscus"            "bonobo"
[4,] "ggo"     "Gorilla gorilla gorilla" "western lowland gorilla"
[5,] "pon"     "Pongo abelii"            "Sumatran orangutan"
[6,] "mcc"     "Macaca mulatta"          "rhesus monkey"

> go.hs=go.gsets(species="human")
> go.bp=go.hs$go.sets[go.hs$go.subs$BP]
> go.mf=go.hs$go.sets[go.hs$go.subs$MF]
> go.cc=go.hs$go.sets[go.hs$go.subs$CC]
> #save(go.bp, go.mf, go.cc, file="go.hs.gsets.RData")

> #for Bioconductor species supported by go.gsets function:
> 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"
```

Here we look at the expression changes towards one direction (either up- or down- regulation) in the gene sets. The results of such 1-directional analysis is a list of two matrices, corresponding to the two directions. Each result matrix contains multiple columns of test statistics and p-/q-values for all tested gene sets. Among them, p.val is the global p-value and q.val is the corresponding FDR q-value. Gene sets are ranked by significance. Type `?gage` for more information.

```
> gse16873.kegg.p <- gage(gse16873, gsets = kegg.gs,
+     ref = hn, samp = dcis)
> #go.gs here only the first 1000 entries as a fast example.
> gse16873.go.p <- gage(gse16873, gsets = go.gs,
+     ref = hn, samp = dcis)
> #or we can do analysis on BP, MF, or CC subcategories if we've
> #generated the data as above.
> #gse16873.bp.p <- gage(gse16873, gsets = go.bp,
> #                        ref = hn, samp = dcis)
> str(gse16873.kegg.p, strict.width='wrap')
```

```
List of 3
$ greater: num [1:222, 1:11] 0.000216 0.001497 0.004771 0.003792 0.01862 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:222] "hsa04141 Protein processing in endoplasmic reticulum"
   "hsa00190 Oxidative phosphorylation" "hsa03050 Proteasome" "hsa04142
   Lysosome" ...
.. ..$ : chr [1:11] "p.geomean" "stat.mean" "p.val" "q.val" ...
$ less : num [1:222, 1:11] 0.00417 0.02714 0.02249 0.05413 0.01902 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:222] "hsa04510 Focal adhesion" "hsa04270 Vascular smooth muscle
   contraction" "hsa04151 PI3K-Akt signaling pathway" "hsa04921 Oxytocin
   signaling pathway" ...
.. ..$ : chr [1:11] "p.geomean" "stat.mean" "p.val" "q.val" ...
$ stats : num [1:222, 1:7] 3.52 2.85 2.63 2.53 2.12 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:222] "hsa04141 Protein processing in endoplasmic reticulum"
   "hsa00190 Oxidative phosphorylation" "hsa03050 Proteasome" "hsa04142
   Lysosome" ...
.. ..$ : chr [1:7] "stat.mean" "DCIS_1" "DCIS_2" "DCIS_3" ...
```

```
> head(gse16873.kegg.p$greater[, 1:5], 4)
```

```
                                                  p.geomean stat.mean
hsa04141 Protein processing in endoplasmic reticulum 0.0002164597  3.517131
hsa00190 Oxidative phosphorylation                 0.0014970021  2.848815
hsa03050 Proteasome                                0.0047708296  2.631099
hsa04142 Lysosome                                  0.0037922282  2.525800
                                                         p.val        q.val
hsa04141 Protein processing in endoplasmic reticulum 9.236559e-18 1.865785e-15
hsa00190 Oxidative phosphorylation                 3.279350e-12 3.312143e-10
hsa03050 Proteasome                                2.108534e-10 1.419746e-08
hsa04142 Lysosome                                  5.083020e-10 2.566925e-08
                                                 set.size
hsa04141 Protein processing in endoplasmic reticulum      144
hsa00190 Oxidative phosphorylation                        97
hsa03050 Proteasome                                       39
hsa04142 Lysosome                                        110
```

```
> head(gse16873.kegg.p$less[, 1:5], 4)
```

```
                                         p.geomean stat.mean        p.val
hsa04510 Focal adhesion                0.004174444 -2.335422 7.134012e-09
hsa04270 Vascular smooth muscle contraction 0.027138691 -1.817663 5.059782e-06
hsa04151 PI3K-Akt signaling pathway    0.022491982 -1.745629 1.014526e-05
hsa04921 Oxytocin signaling pathway    0.054131510 -1.568633 6.483178e-05
                                              q.val set.size
hsa04510 Focal adhesion                1.441070e-06      189
hsa04270 Vascular smooth muscle contraction 5.110380e-04      102
hsa04151 PI3K-Akt signaling pathway    6.831145e-04      300
hsa04921 Oxytocin signaling pathway    2.845448e-03      136
```

```
> head(gse16873.kegg.p$stats[, 1:5], 4)
```

```
                                                stat.mean   DCIS_1
hsa04141 Protein processing in endoplasmic reticulum  3.517131 4.184493
hsa00190 Oxidative phosphorylation               2.848815 4.066860
hsa03050 Proteasome                              2.631099 3.239365
hsa04142 Lysosome                                2.525800 2.351035
                                                  DCIS_2    DCIS_3
hsa04141 Protein processing in endoplasmic reticulum 4.526776 2.6444345
hsa00190 Oxidative phosphorylation              3.586117 0.6245743
hsa03050 Proteasome                             2.792570 2.0223035
hsa04142 Lysosome                               1.469745 4.4956840
                                                  DCIS_4
hsa04141 Protein processing in endoplasmic reticulum 3.242629
hsa00190 Oxidative phosphorylation              2.533425
hsa03050 Proteasome                             2.344753
hsa04142 Lysosome                               1.347200
```

As described in GAGE paper (?), it is worthy to run gage with same.dir=F option on KEGG pathways too to capture pathways perturbed towards both directions simultaneously. Note we normally do not use same.dir=F option on GO groups, which are mostly gene sets coregulated towards the same direction.

```
> gse16873.kegg.2d.p <- gage(gse16873, gsets = kegg.gs,
+     ref = hn, samp = dcis, same.dir = F)
> head(gse16873.kegg.2d.p$greater[,1:5], 4)
```

```
                                       p.geomean stat.mean        p.val
hsa04510 Focal adhesion              0.002163329  2.838093 2.588032e-12
hsa04512 ECM-receptor interaction    0.002393792  2.846662 3.958880e-12
hsa04974 Protein digestion and absorption 0.015991052  2.102221 1.938956e-07
hsa04151 PI3K-Akt signaling pathway  0.031062064  1.820065 4.329195e-06
                                            q.val set.size
hsa04510 Focal adhesion              3.998469e-10      189
hsa04512 ECM-receptor interaction    3.998469e-10       74
hsa04974 Protein digestion and absorption 1.305564e-05       74
hsa04151 PI3K-Akt signaling pathway  2.186244e-04      300
```

```
> head(gse16873.kegg.2d.p$stats[,1:5], 4)
```

```
                                     stat.mean   DCIS_1   DCIS_2   DCIS_3
hsa04510 Focal adhesion               2.838093 2.195874 3.417239 2.929209
```

```
hsa04512 ECM-receptor interaction          2.846662 2.497487 3.316319 3.337686
hsa04974 Protein digestion and absorption  2.102221 2.433950 1.981627 2.711449
hsa04151 PI3K-Akt signaling pathway        1.820065 1.422898 2.626514 2.238497
                                             DCIS_4
hsa04510 Focal adhesion                   2.2734695
hsa04512 ECM-receptor interaction         2.2355312
hsa04974 Protein digestion and absorption 0.7533766
hsa04151 PI3K-Akt signaling pathway       1.1321553
```

We may also do PAGE analysis (?) with or without different combinations of GAGE style options. The key difference is 1) to use z-test instead of two-sample t-test and 2) a group-on-group comparison scheme (by setting arguement `compare="as.group"`) instead of default one-on-one scheme (`compare="paired"`) as in GAGE. Here we only used z-test option to compare the net effect of using differet gene set tests, as detailed in ?.

```
> gse16873.kegg.page.p <- gage(gse16873, gsets = kegg.gs,
+     ref = hn, samp = dcis, saaTest = gs.zTest)
> head(gse16873.kegg.page.p$greater[, 1:5], 4)
```

```
                                                p.geomean stat.mean
hsa04141 Protein processing in endoplasmic reticulum 7.807248e-07  4.701752
hsa00190 Oxidative phosphorylation              2.660152e-05  3.782537
hsa04142 Lysosome                               6.607464e-05  3.453798
hsa03050 Proteasome                             2.562528e-04  3.384068
                                                    p.val        q.val
hsa04141 Protein processing in endoplasmic reticulum 5.422494e-31 1.095344e-28
hsa00190 Oxidative phosphorylation              9.727891e-21 9.825170e-19
hsa04142 Lysosome                               1.336397e-17 8.998409e-16
hsa03050 Proteasome                             5.698703e-17 2.877845e-15
                                                set.size
hsa04141 Protein processing in endoplasmic reticulum      144
hsa00190 Oxidative phosphorylation                    97
hsa04142 Lysosome                                    110
hsa03050 Proteasome                                   39
```

We get much smaller p-/q-values with PAGE than with GAGE. As described in GAGE paper (?), many significant calls made by PAGE are likely false positives.

With *gage*, we have much more options to tweak than those related to PAGE for more customized GAGE analysis. Here we show a few other alternative ways of doing GAGE analysis by setting other arguements. Use t-test statistics instead of fold change as per gene score:

```
> gse16873.kegg.t.p <- gage(gse16873, gsets = kegg.gs,
+     ref = hn, samp = dcis, use.fold = F)
```

If you are very concerned about the normality of expression level changes or the gene set sizes are mostly smaller than 10, we may do the non-parametric Mann Whitney U tests (rank test without distribution assumption) instead of the parametric two-sample t-tests on gene sets:

```
> gse16873.kegg.rk.p <- gage(gse16873, gsets = kegg.gs,
+     ref = hn, samp = dcis, rank.test = T)
```

Do the non-parametric Kolmogorov-Smirnov tests (like rank test, used in GSEA) instead of the parametric two-sample t-tests on gene sets:

```
> gse16873.kegg.ks.p <- gage(gse16873, gsets = kegg.gs,
+     ref = hn, samp = dcis, saaTest = gs.KSTest)
```

Frequently, the samples are not one-on-one paired in the experiment design. In such cases, we take the samples as unpaired:

```
> gse16873.kegg.unpaired.p <- gage(gse16873, gsets = kegg.gs,
+     ref = hn, samp = dcis, compare = "unpaired")
```

Since version 2.2.0, gage package does more robust p-value summarization using Stouffer's method through argument `use.stouffer=TRUE`. The original p-value summarization, i.e. negative log sum following a Gamma distribution as the Null hypothesis, may produce less stable global p-values for large or heterogenous datasets. In other words, the global p-value could be heavily affected by a small subset of extremely small individual p-values from pair-wise comparisons. Such sensitive global p-value leads to the "dual signficance" phenomenon. Dual-signficant means a gene set is called significant simultaneously in both 1-direction tests (up- and down-regulated). While not desirable in other cases, "dual signficance" could be informative in revealing the sub-types or sub-classes in big clinical or disease studies. If we preferred the original Gamma distribution based p-value summarization, we only need to set `use.stouffer=FALSE`:

```
> gse16873.kegg.gamma.p <- gage(gse16873, gsets = kegg.gs,
+     ref = hn, samp = dcis, use.stouffer=FALSE)
```

Sometimes we have expression data where genes are labelled by symbols (or other IDs). Let's make such a dataset `gse16873.sym` from `gse16873`, which is originally lable by Entrez Gene IDs.

```
> head(rownames(gse16873))

[1] "10000"     "10001"     "10002"     "10003"     "100048912" "10004"

> gse16873.sym<-gse16873
> data(egSymb)
> rownames(gse16873.sym)<-eg2sym(rownames(gse16873.sym))
> head(rownames(gse16873.sym))

[1] "AKT3"     "MED6"     "NR2E3"    "NAALAD2"  "CDKN2BAS" "NAALADL1"
```

If we want to do GAGE analysis, we can convert rownames of `gse16873.sym` back to Entrez Gene IDs or convert gene set definitions to gene symbols.

```
> kegg.gs.sym<-lapply(kegg.gs, eg2sym)
> lapply(kegg.gs.sym[1:3],head)

$`hsa00970 Aminoacyl-tRNA biosynthesis`
[1] "FARSB"  "WARS2"  "FARS2"  "PSTK"   "MTFMT"  "TARSL2"

$`hsa02010 ABC transporters`
[1] "ABCC5"  "ABCB6"  "ABCC9"  "ABCC4"  "ABCA7"  "ABCA10"

$`hsa03008 Ribosome biogenesis in eukaryotes`
[1] NA          "RCL1"      NA          NA          NA          "MPHOSPH10"

> gse16873.kegg.p2 <- gage(gse16873.sym, gsets = kegg.gs.sym,
+     ref = hn, samp = dcis)
```

8 Result Presentation and Intepretation

We may output full result tables from the analysis.

```
> write.table(gse16873.kegg.2d.p$greater, file = "gse16873.kegg.2d.p.txt",
+     sep = "\t")
> write.table(rbind(gse16873.kegg.p$greater, gse16873.kegg.p$less),
+     file = "gse16873.kegg.p.txt", sep = "\t")
```

Sort and count signficant gene sets based on q- or p-value cutoffs:

```
> gse16873.kegg.sig<-sigGeneSet(gse16873.kegg.p, outname="gse16873.kegg")

[1] "there are 23 signficantly up-regulated gene sets"
[1] "there are 25 signficantly down-regulated gene sets"

> str(gse16873.kegg.sig, strict.width='wrap')

List of 3
$ greater: num [1:23, 1:11] 0.000216 0.001497 0.004771 0.003792 0.01862 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:23] "hsa04141 Protein processing in endoplasmic reticulum"
   "hsa00190 Oxidative phosphorylation" "hsa03050 Proteasome" "hsa04142
   Lysosome" ...
.. ..$ : chr [1:11] "p.geomean" "stat.mean" "p.val" "q.val" ...
$ less : num [1:25, 1:11] 0.00417 0.02714 0.02249 0.05413 0.01902 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:25] "hsa04510 Focal adhesion" "hsa04270 Vascular smooth muscle
   contraction" "hsa04151 PI3K-Akt signaling pathway" "hsa04921 Oxytocin
   signaling pathway" ...
.. ..$ : chr [1:11] "p.geomean" "stat.mean" "p.val" "q.val" ...
$ stats : num [1:48, 1:7] 3.52 2.85 2.63 2.53 2.12 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:48] "hsa04141 Protein processing in endoplasmic reticulum"
   "hsa00190 Oxidative phosphorylation" "hsa03050 Proteasome" "hsa04142
   Lysosome" ...
.. ..$ : chr [1:7] "stat.mean" "DCIS_1" "DCIS_2" "DCIS_3" ...

> gse16873.kegg.2d.sig<-sigGeneSet(gse16873.kegg.2d.p, outname="gse16873.kegg")

[1] "there are 30 signficantly two-direction perturbed gene sets"

> str(gse16873.kegg.2d.sig, strict.width='wrap')

List of 2
$ greater: num [1:30, 1:11] 0.00216 0.00239 0.01599 0.03106 0.0319 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:30] "hsa04510 Focal adhesion" "hsa04512 ECM-receptor
   interaction" "hsa04974 Protein digestion and absorption" "hsa04151 PI3K-Akt
   signaling pathway" ...
.. ..$ : chr [1:11] "p.geomean" "stat.mean" "p.val" "q.val" ...
$ stats : num [1:30, 1:7] 2.84 2.85 2.1 1.82 1.78 ...
```

```
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:30] "hsa04510 Focal adhesion" "hsa04512 ECM-receptor
   interaction" "hsa04974 Protein digestion and absorption" "hsa04151 PI3K-Akt
   signaling pathway" ...
.. ..$ : chr [1:7] "stat.mean" "DCIS_1" "DCIS_2" "DCIS_3" ...
```

```
> write.table(gse16873.kegg.2d.sig$greater, file = "gse16873.kegg.2d.sig.txt",
+     sep = "\t")
> write.table(rbind(gse16873.kegg.sig$greater, gse16873.kegg.sig$less),
+     file = "gse16873.kegg.sig.txt", sep = "\t")
```

Heatmaps in Figure **??** visualize whole gene set perturbations, i.e. the gene set test statistics (or p-values) in pseudo-color.

There are frequently multiple significant gene sets that share multiple member genes or represent the same regulatory mechanism. Such redundancy in signficant gene set could be misleading too. A pathway or functional group itself is not relevant, but may be called signficant because it share multiple perturbed genes with one really significant gene set. We derive the non-redundant signficant gene set lists, with result output as tab-delimited text files automatically below. Here, function `esset.grp` calls redundant gene sets to be those overlap heavily in their core genes. Core genes are those member genes that really contribute to the signficance of the gene set in GAGE analysis. Arguement `pc` is the cutoff p-value for the overlap between gene sets, default to `10e-10`, may need trial-and-error to find the optimal value in minor cases. Note that we use small `pc` because redundant gene sets are not just signficant in overlap, but are HIGHLY significant so.

```
> gse16873.kegg.esg.up <- esset.grp(gse16873.kegg.p$greater,
+     gse16873, gsets = kegg.gs, ref = hn, samp = dcis,
+     test4up = T, output = T, outname = "gse16873.kegg.up", make.plot = F)
> gse16873.kegg.esg.dn <- esset.grp(gse16873.kegg.p$less,
+     gse16873, gsets = kegg.gs, ref = hn, samp = dcis,
+     test4up = F, output = T, outname = "gse16873.kegg.dn", make.plot = F)
> names(gse16873.kegg.esg.up)
```

```
[1] "essentialSets"      "setGroups"          "allSets"
[4] "connectedComponent" "overlapCounts"      "overlapPvals"
[7] "coreGeneSets"
```

```
> head(gse16873.kegg.esg.up$essentialSets, 4)
```

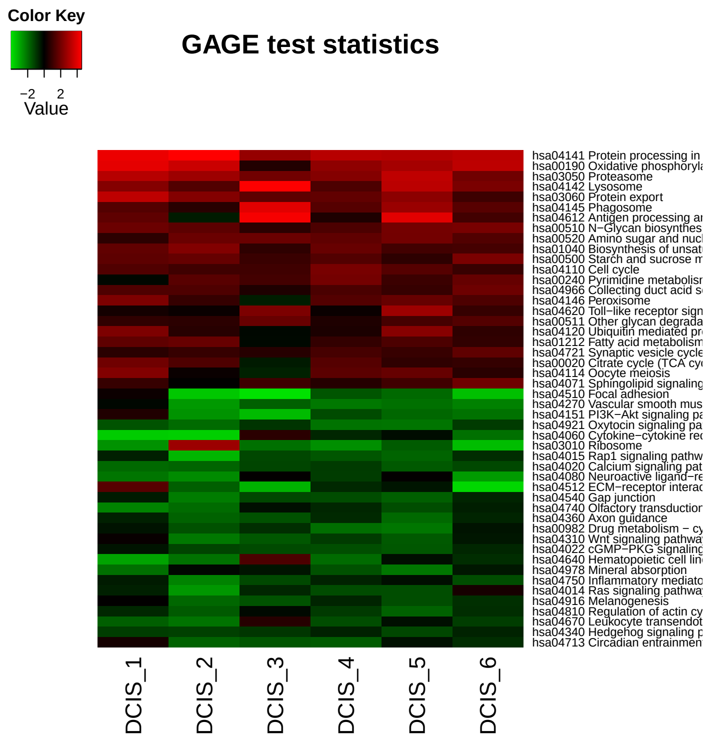
```
[1] "hsa04141 Protein processing in endoplasmic reticulum"
[2] "hsa00190 Oxidative phosphorylation"
[3] "hsa03050 Proteasome"
[4] "hsa04142 Lysosome"
```

```
> head(gse16873.kegg.esg.up$setGroups, 4)
```

```
[[1]]
[1] "hsa04141 Protein processing in endoplasmic reticulum"

[[2]]
[1] "hsa00190 Oxidative phosphorylation"

[[3]]
```



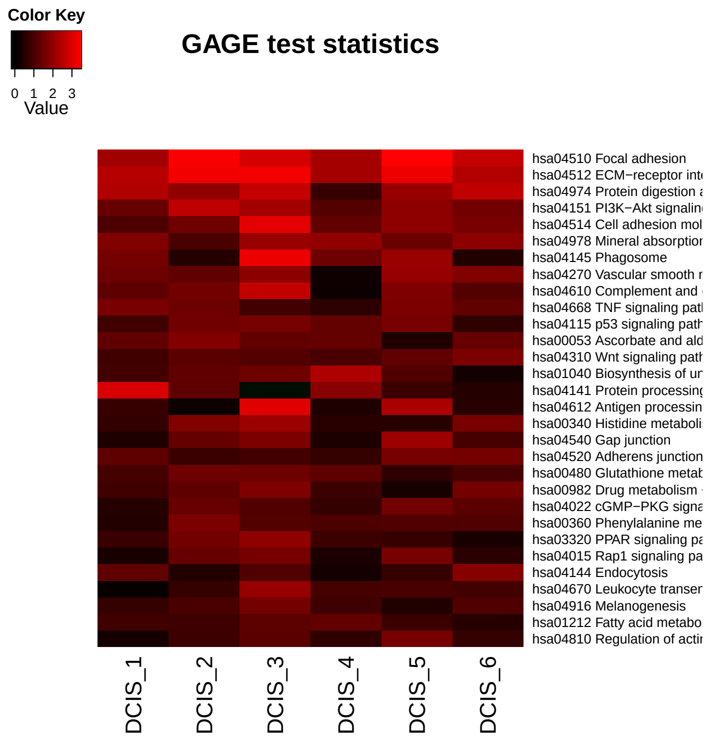


Figure 1: Example heatmaps generated with `sigGeneSet` function as to show whole gene set perturbations. Upper panel: signficant KEGG pathways in 1-directional (either up or down) test; lower panel: signficant KEGG pathways in 2-directional test. Only gene set test statistics are shown, heatmaps for -log10(p-value) look similar.

```
[1] "hsa03050 Proteasome"

[[4]]
[1] "hsa04142 Lysosome"                 "hsa00511 Other glycan degradation"
```

```
> head(gse16873.kegg.esg.up$coreGeneSets, 4)

$`hsa04141 Protein processing in endoplasmic reticulum`
 [1] "51128" "2923"  "10130" "3312"  "10970" "3301"  "9451"  "23480" "9601"
[10] "811"   "3309"  "7323"  "7494"  "6748"  "64374" "10525" "7466"  "10808"
[19] "6184"  "821"   "56886" "5887"  "6185"  "5034"  "6745"  "1603"  "7095"
[28] "79139" "3320"  "27102" "10294" "10483" "7415"  "7324"  "30001" "64215"
[37] "7184"  "5611"  "29927" "9871"  "9978"  "5601"  "22872" "6500"  "23471"
[46] "22926" "11231" "57134" "6396"  "10134"

$`hsa00190 Oxidative phosphorylation`
 [1] "1345"  "4720"  "4710"  "51382" "4709"  "51606" "27089" "533"   "29796"
[10] "9377"  "4711"  "528"   "514"   "10975" "51079" "10312" "8992"  "518"
[19] "4696"  "1537"  "1340"  "4708"  "521"   "54539" "537"   "4694"  "4701"
[28] "4702"  "509"

$`hsa03050 Proteasome`
 [1] "5691"  "10213" "5684"  "5685"  "5701"  "5688"  "51371" "5692"  "5714"
[10] "9861"  "5705"  "5720"  "5687"  "7979"  "5708"  "5690"  "5696"  "5718"
[19] "5707"

$`hsa04142 Lysosome`
 [1] "3920"  "1509"  "2517"  "1213"  "2720"  "1508"  "51606" "1512"  "54"
[10] "55353" "2990"  "1520"  "427"   "5660"  "533"   "10577" "27074" "5476"
[19] "9741"  "967"   "1514"  "10312" "2799"  "1075"  "3074"  "4126"  "8763"
[28] "10239" "7805"  "537"   "3988"
```

We may extract the gene expression data for top gene sets, output and visualize these expression data for further intepretation. Although we can check expression data in each top gene set one by one, here we work on the top 3 up-regulated KEGG pathways in a batch. The key functions we use here are, `essGene` (for extract genes with substiantial or above-background expression changes in gene sets) and `geneData` (for output and visualization of expression data in gene sets).

```
> rownames(gse16873.kegg.p$greater)[1:3]

[1] "hsa04141 Protein processing in endoplasmic reticulum"
[2] "hsa00190 Oxidative phosphorylation"
[3] "hsa03050 Proteasome"

> gs=unique(unlist(kegg.gs[rownames(gse16873.kegg.p$greater)[1:3]]))
> essData=essGene(gs, gse16873, ref =hn, samp =dcis)
> head(essData, 4)

         HN_1     HN_2      HN_3     HN_4      HN_5      HN_6    DCIS_1
1345  9.109413 9.373454 10.988181 9.161435 11.032016 11.231293 12.675099
5691  8.283191 7.716745  7.553621 8.381538  7.768811  7.635745  8.840405
```

```
51128 7.424312 7.970012  8.034436 6.806669  8.508019  8.523295  8.636513
2923  9.362371 9.150221  8.537944 8.828966  9.890736  9.980784 11.168409
         DCIS_2    DCIS_3    DCIS_4    DCIS_5    DCIS_6
1345  11.231271 12.547915  8.979639 13.470266 12.156052
5691   8.125168  7.782958 11.910352  7.956182  7.713826
51128  8.639654  8.740153  8.037517  9.060658  8.845460
2923   9.396683  9.176227  9.752254 10.529351 10.242883
```

```
> ref1=1:6
> samp1=7:12
> for (gs in rownames(gse16873.kegg.p$greater)[1:3]) {
+     outname = gsub(" |:|/", "_", substr(gs, 10, 100))
+     geneData(genes = kegg.gs[[gs]], exprs = essData, ref = ref1,
+         samp = samp1, outname = outname, txt = T, heatmap = T,
+         Colv = F, Rowv = F, dendrogram = "none", limit = 3, scatterplot = T)
+ }
```

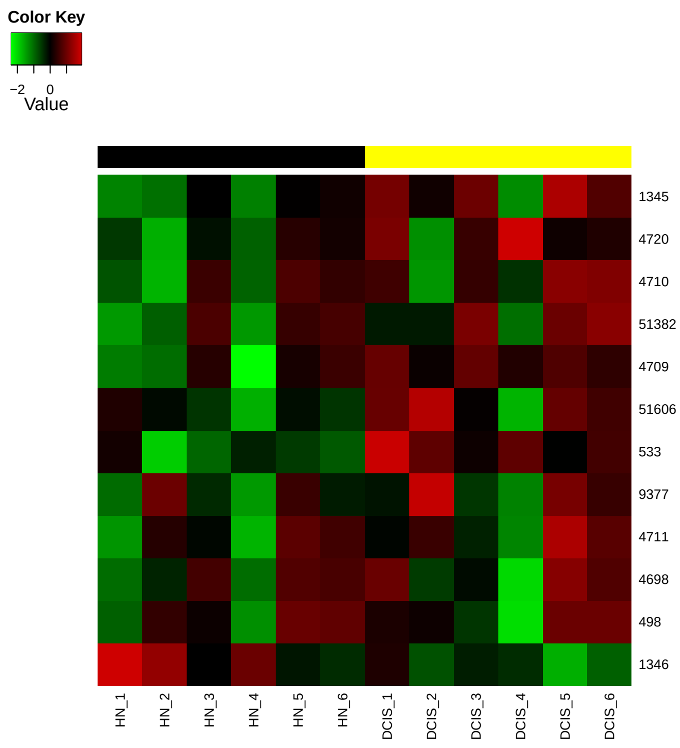
Figure **??** shows example heatmap and scatter plots generated by the code chunk above. Notice that this heatmap Figure **??** is somewhat inconsistent from the scatter plot, because the heatmap compares all samples with each other (hence unpaired samples assumed), yet the scatter plot Figure **??** incorporate the experiment design information and compare the match sample pairs only. The same inconsistency is also shown in Figure **??**.

Sometimes, we may also want to check the expression data for all genes in a top gene set, rather than just those above-background genes selected using `essGene` as above. Notice in this case (Figure **??**), the heatmap may less informative than the one in Figure 2 due to the inclusion of background noise.

```
> for (gs in rownames(gse16873.kegg.p$greater)[1:3]) {
+     outname = gsub(" |:|/", "_", substr(gs, 10, 100))
+     outname = paste(outname, "all", sep=".")
+     geneData(genes = kegg.gs[[gs]], exprs = gse16873, ref = hn,
+         samp = dcis, outname = outname, txt = T, heatmap = T,
+         Colv = F, Rowv = F, dendrogram = "none", limit = 3, scatterplot = T)
+ }
```

Starting from BioC 2.12 (R-2.16), we can visualize the KEGG pathway analysis results using a new package called *pathview* (**?**). Of course, you may manually installed the package with earlier BioC or R versions. Note that *pathview* can view our expression perturbation patterns on two styles of pathway graphs: KEGG view or Graphviz view ((Figure **??**). All we need is to supply our data (expression changes) and specify the target pathway. *Pathview* automatically downloads the pathway graph data, parses the data file, maps user data to the pathway, and renders pathway graph with the mapped data. For demonstratoin, let's look at a couple of selected pathways.

```
> library(pathview)
> gse16873.d <- gse16873[ ,dcis] - gse16873[ ,hn]
> path.ids=c("hsa04110 Cell cycle", "hsa00020 Citrate cycle (TCA cycle)")
> path.ids2 <- substr(path.ids, 1, 8)
> #native KEGG view
> pv.out.list <- sapply(path.ids2, function(pid) pathview(gene.data = gse16873.d[,
+                          1:2], pathway.id = pid, species = "hsa"))
> #Graphviz view
> pv.out.list <- sapply(path.ids2, function(pid) pathview(gene.data = gse16873.d[,
+                          1:2], pathway.id = pid, species = "hsa", kegg.native=F,
+                          sign.pos="bottomleft"))
```



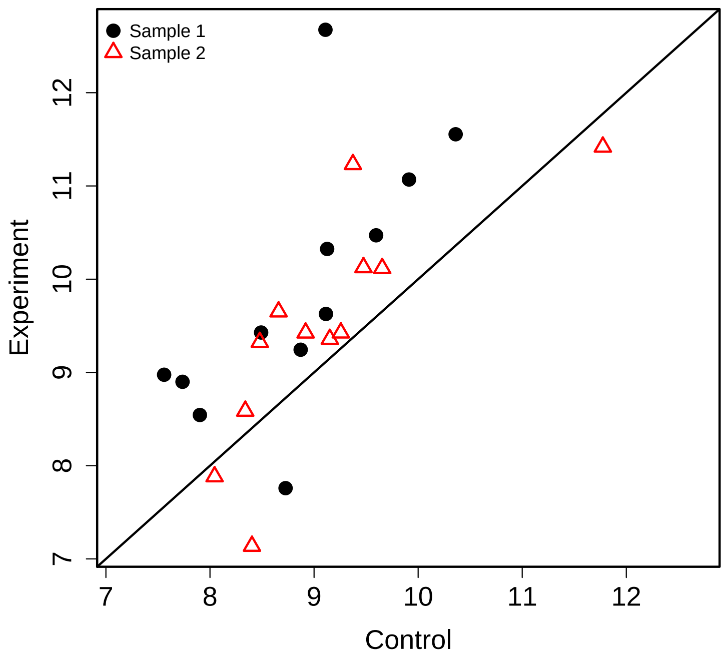


Figure 2: Example heatmap and scatter plot generated with `geneData` function to show the gene expresion perturbations in specified gene set(s). Only HN (control) and DCIS (experiment) data from the first two patients are plotted in the scatter plot.

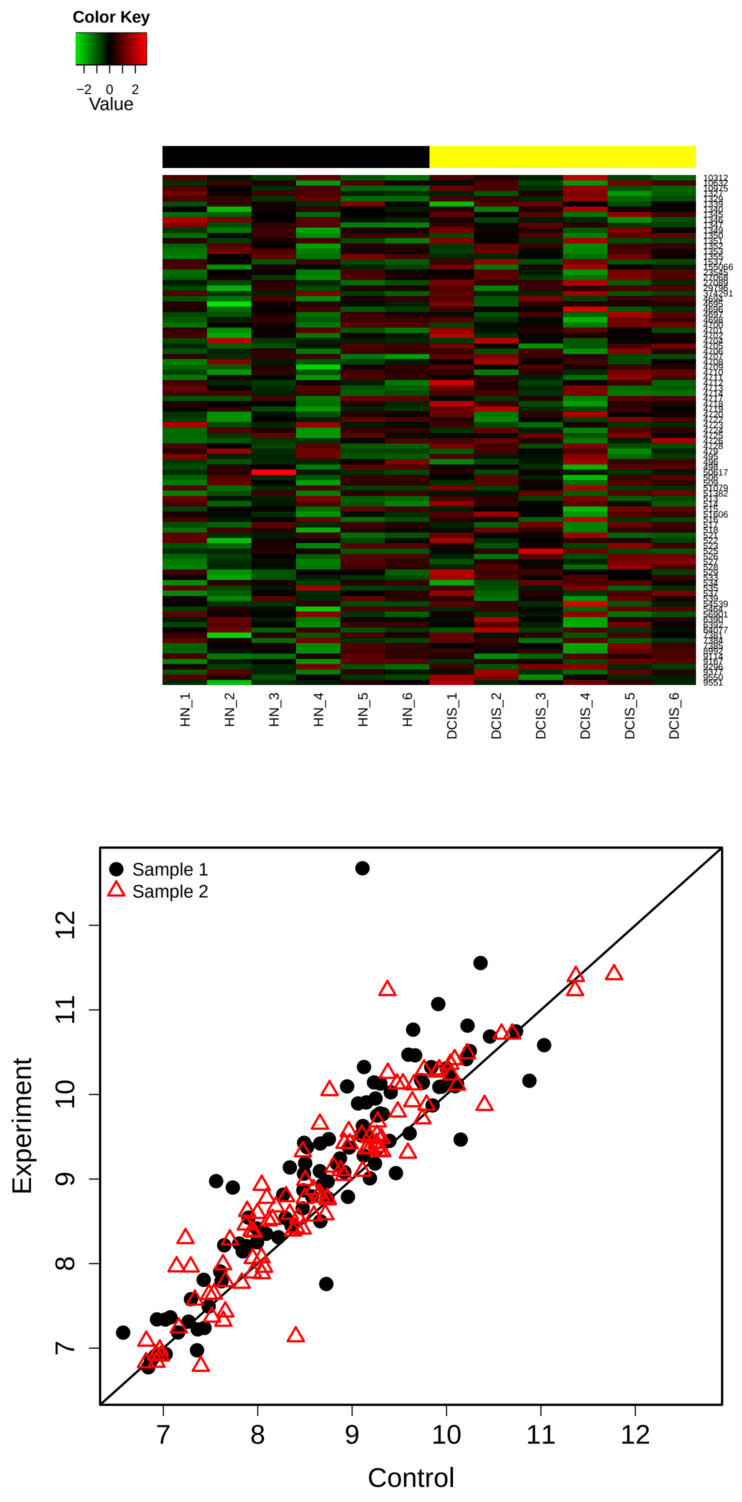


Figure 3: Example heatmap and scatter plot generated with `geneData` function to show the gene expresion perturbations in specified gene set(s). Only HN (control) and DCIS (experiment) data from the first two patients are plotted in the scatter plot. And all genes in a gene set are included here.

This pathway graph based data visualization can be simply applied to all selected pathways in a batch. In other words, in a few lines of code, we may connect *gage* to *pathview* for large-scale and fully automated pathway analysis and results visualization.

```
> sel <- gse16873.kegg.p$greater[, "q.val"] < 0.1 & !is.na(gse16873.kegg.p$greater[,
+ "q.val"])
> path.ids <- rownames(gse16873.kegg.p$greater)[sel]
> path.ids2 <- substr(path.ids, 1, 8)
> pv.out.list <- sapply(path.ids2, function(pid) pathview(gene.data = gse16873.d[,
+                         1:2], pathway.id = pid, species = "hsa"))
```

9 Advanced Analysis

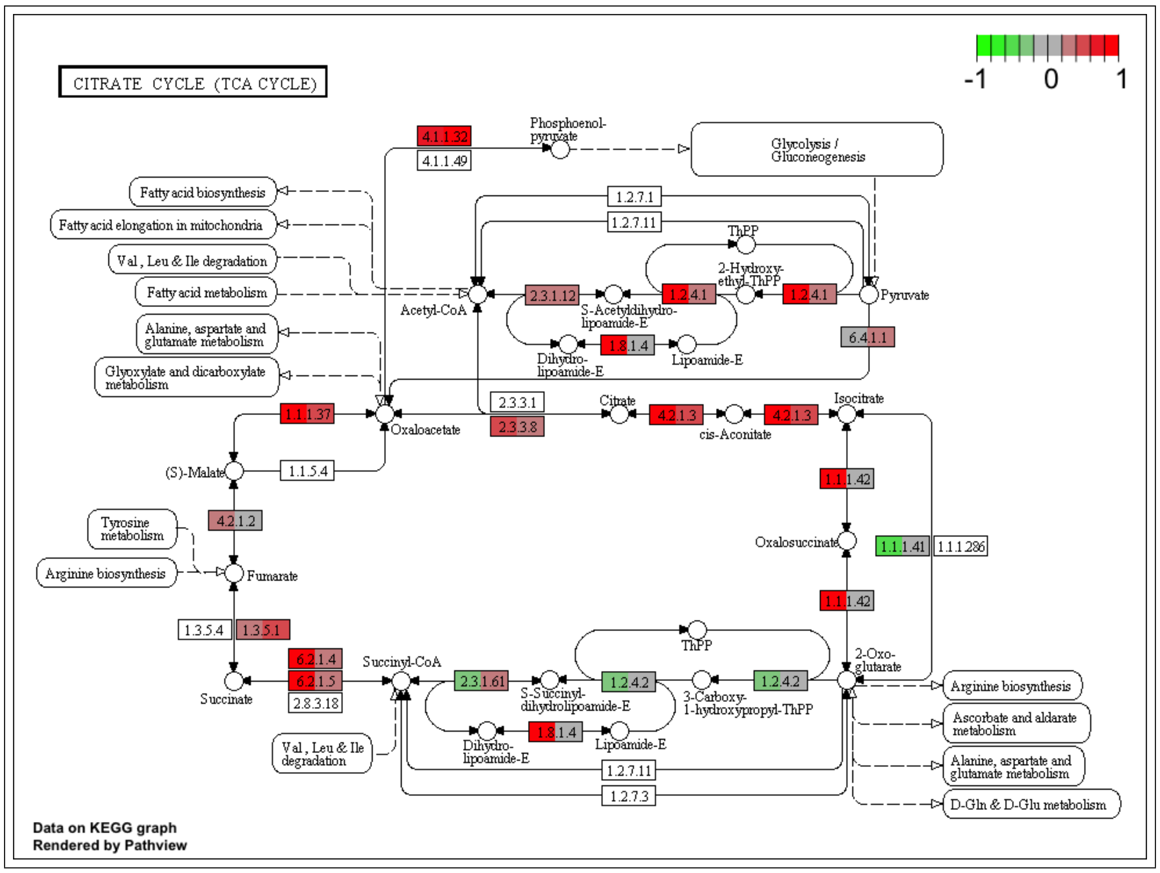
We frequently need to do GAGE analysis repetitively on mulitple comparisons (with different samples vs references) in a study, or even with different analysis options (paired or unpaired samples, use or not use rank test etc). We can carry these different analyses with different sub-datasets all at once using a composite funciton, `gagePipe`. Different from `gage`, `gagePipe` accepts lists of reference/sample column numbers, with matching lists/vectors of other arguements. Function `gagePipe` runs multiple rounds of GAGE in a batch without interference, and outputs signficant gene set lists in text format and save the results in `.RData` format.

```
> #introduce another half dataset
> library(gageData)
> data(gse16873.2)
> cn2=colnames(gse16873.2)
> hn2=grep('HN',cn2, ignore.case =T)
> dcis2=grep('DCIS',cn2, ignore.case =T)
> #batch GAGE analysis on the combined dataset
> gse16873=cbind(gse16873, gse16873.2)
> dataname='gse16873' #output data prefix
> sampnames=c('dcis.1', 'dcis.2')
> refList=list(hn, hn2+12)
> sampList=list(dcis, dcis2+12)
> gagePipe(gse16873, gsname = "kegg.gs", dataname = "gse16873",
+     sampnames = sampnames, ref.list = refList, samp.list = sampList,
+     comp.list = "paired")
```

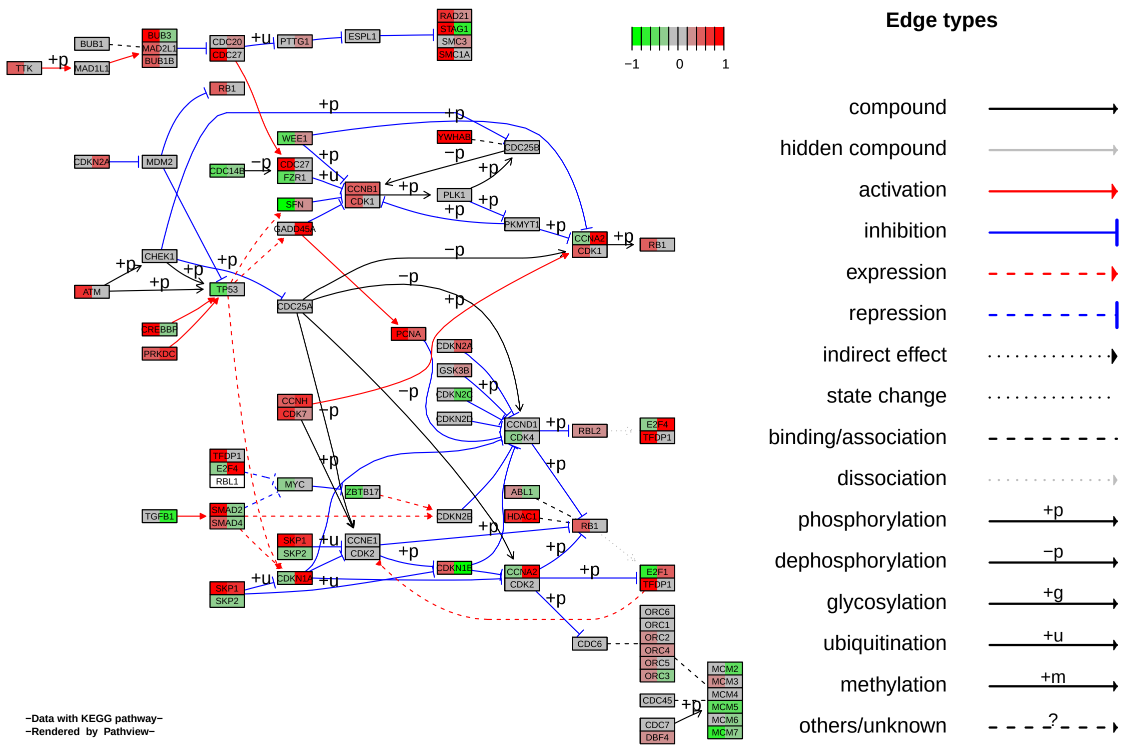
We may further loaded the `.RData` results, and do more analysis. For instance, we may compare the GAGE analysis results from the differen comparisons or different sub-datasets we have worked on. Here, the main function to use is `gageComp`. Comparison rsults between multiple GAGE analyses will be output as text files and optionally, venn diagram can be plotted in PDF format as shown in Figure **??**.

```
> load('gse16873.gage.RData')
> gageComp(sampnames, dataname, gsname = "kegg.gs",
+     do.plot = TRUE)
```

GAGE with single array analysis design also provide a framework for combined analysis accross heterogeneous microarray studies/experiments. The combined dataset of `gse16873` and `gse16873.2` provids a good example of heterogeneous data. As mentioned above, the two half-datasets were processed using FARMS (**?**) and RMA (**?**) methods separately. Therefore, the expression values and distributions are dramatically different between the two halves. Using function `heter.gage` we can do some combined analysis on such heterogeneous dataset(s). `heter.gage` is similar to `gagePipe` in that `ref.list` and `samp.list` arguements need to be lists of column numbers with matching vector of the `compare` arguement. Different from `gagePipe`,



(a)



(b)

Figure 4: Visualize GAGE pathway analysis results using *pathview* (a) KEGG view of hsa00020 Citrate cycle (TCA cycle) or (b) Graphviz view of hsa04110 Cell cycle.

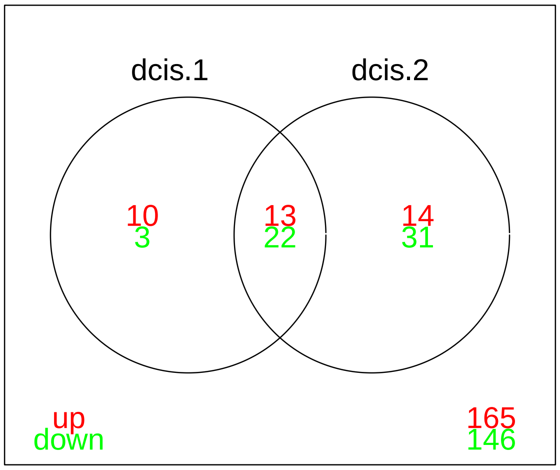


Figure 5: An example venn diagram generated with `gageComp` function. Compared are KEGG pathway results for the two half datasets when `gagePipe` was applied above.

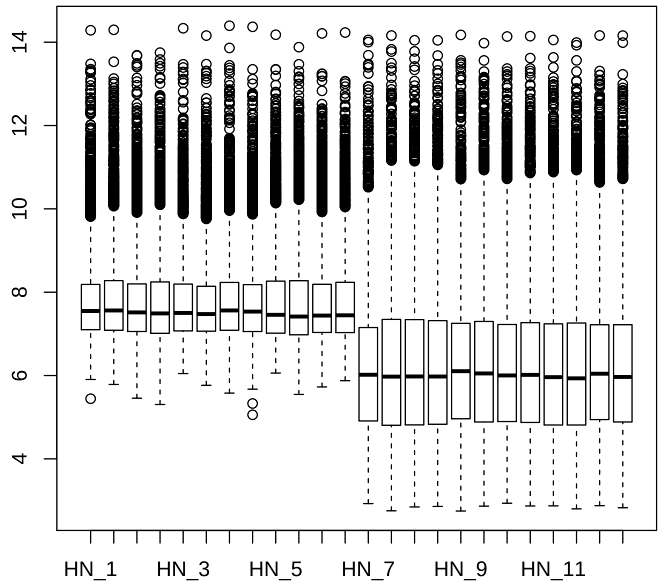


Figure 6: Sample-wise gene expression level distributions for GSE16873 with the two differently processed half datasets.

`heter.gage` does one combined GAGE analysis on all data instead of multiple separate analyses on different sub-datasets/comparisons.

Just to have an idea on how heterogeneous these two half datasets are, we may visualize the expression level distributions (Figure ??):

```
> boxplot(data.frame(gse16873))
> gse16873.kegg.heter.p <- heter.gage(gse16873, gsets = kegg.gs,
+     ref.list = refList, samp.list = sampList)
> gse16873.kegg.heter.2d.p <- heter.gage(gse16873, gsets = kegg.gs,
+     ref.list = refList, samp.list = sampList, same.dir = F)
```

We may compare the results from this combined analysis of the combined dataset vs the analysis on the first half dataset above. As expected the top gene sets from this combined analysis are consistent yet with smaller p-values due to the use of more data.

10 Common Errors

- Gene sets and expression data use different ID systems (Entrez Gene ID, Gene symbol or probe set ID etc). To correct, use functions like eg2sym or sym2eg, or check vignette, "Gene set and data preparation", for more solutions. If you used customized CDF file based on Entrez Gene to processed the raw data, do: `rownames(gse16873)=gsub('_at', '', rownames(gse16873))`.
- We use gene set data for a different species than the expression data, e.g. use `kegg.mm` instead of `kegg.gs` for human data. When running `gage` or `gagePipe` function, we get error message like, `Error in if (is.na(spval[ NA : argument is of length zero`.
- Expression data have multiple probesets (as in Affymetrix GeneChip Data) for a single gene, but gene set analysis requires one entry per gene. You may pick up the most differentially expressed probeset for a gene and discard the rest, or process the raw intensity data using customized probe set definition (CDF file), where probes are re-mapped on a gene by gene base. Check the Methods section of GAGE paper (?) for details.
- Expression data have genes as columns and samples/chips as rows, i.e. in a transposed form. To correct, do: `expdata=t(expdata)`.