PPInfer: Inferring functionally related proteins using protein interaction networks

Dongmin Jung, Xijin Ge

May 23, 2017

1 Introduction

Interactions between proteins occur in many, if not most, biological processes. Most proteins perform their functions in networks associated with other proteins and other biomolecules. This fact has motivated the development of a variety of experimental methods for the identification of protein interactions. This variety has in turn ushered in the development of numerous different computational approaches for modeling and predicting protein interactions. Sometimes an experiment is aimed at identifying proteins closely related to some interesting proteins. A network based statistical learning method is used to infer the putative functions of proteins from the known functions of its neighboring proteins on a PPI network. This package identifies such proteins often involved in the same or similar biological functions.

2 Graph

Graph data is ubiquitous and graph mining is the study that aims to discover novel and insightful knowledge from data that is represented as a graph. Graph mining differs from traditional data mining in a number of critical ways. For example, the topic of classification in data mining is often introduced in relation to vector data; however, these techniques are often unsuitable when applied to graphs, which require an entirely different approach such as the use of graph kernels (Samatova *et al.*, 2013).

A support vector machine only applies to datasets in the real space. Often, however, we want to use a SVM on a dataset that is not a subset of the real space. This occurs in the case of biology and chemistry problems to describe our data. Fortunately, there is a ready solution to this problem, formalized in the use of kernel functions (Werther & Seitz, 2008). We employ the kernel support vector machine (KSVM) based on the regularized Laplacian matrix (Smola & Kondor, 2003) for a graph. The kernel matrix K can now be used with a classification algorithm for predicting the class of vertices in the given dataset,

$$K = (I + \gamma L)^{-1},$$

where K is $N \times N$, I is an identity matrix, L is the normalized Laplacian matrix, and γ is an appropriate decay constant. The decay constant is typically regarded as an arbitrary constant that is less than one.

3 Support Vector Machine

We focus on the application of computational method using a support vector machine. Suppose we have a dataset in the real space and that each point in our dataset has a corresponding class label. Our goal is to separate the points in our dataset according to their class label. A SVM is a linear binary classifier. The idea behind nonlinear SVM is to find an optimal separating hyperplane in high-dimensional feature space just as we did for the linear SVM in original space. At the heart of kernel methods is the notion of a kernel function. Broadly speaking, kernels can be thought of as functions that produce similarity matrices (Kolaczyk & Csardi, 2014). One of the advantages of support vector machines is that we can improve performance by properly selecting kernels. In most applications, RBF kernels are widely used but kernels suited for specific applications are developed. Here, we select the graph kernel K for PPI.

Data in many biological problems are often compounded by imbalanced class distribution, known as the imbalanced data problem, in which the size of one class is significantly larger than that of the other class. Many classification algorithms such as a SVM are sensitive to data with imbalanced class distribution, and result in a suboptimal classification. It is desirable to compensate the imbalance effect in model training for more accurate classification. One possible solution to the imbalanced data problem is to use one-class SVMs by learning from the target class only, instead of traditional binary SVMs. In one-class classification, it is assumed that only information of one of the classes, the target class, is available, and no information is available from the other class, known as the background. The task of one-class classification is to define a boundary around the target class such that it accepts as much of the targets as possible and excludes the outliers as much as possible (Ma, 2014).

However, one-class classifiers seldom outperform two-class classifiers when the data from two class are available (Ma, 2014). So the OCSVM and classical SVM are sequentially used in this package. First, we apply the OCSVM by training a one-class classifier using the data from the known class only. Let n be the number of proteins in the target class. This model is used to identify distantly related proteins among remaining $N - n$ proteins in the background. Proteins with zero similarity with the target class are extracted. Then they are potentially defined as the other class by pseudo-absence selection methods (Senay *et al.*, 2013) from spatial statistics. The target class can be seen as real presence data. For the data to be balanced, assume that two classes contain the same number of proteins. Next, by the classical SVM, these two classes are used to identify closely related proteins among remaining $N - 2n$ proteins. Those found by this procedure can be functionally linked to the known class or interesting proteins.

4 Example

Consider a simple example about a graph representing the curated set of literature predicted protein-protein interactions, containing 2885 nodes, named using yeast standard names.

```
library(PPInfer)
data(litG)
litG <- graph_from_graphnel(litG)
```

```
summary(litG)
```

```
IGRAPH UN-- 2885 315 --
+ attr: name (v/c)
```

```
sg <- decompose(litG, min.vertices=50)
sg <- sg[[1]]           # largest subgraph
summary(sg)
```

```
IGRAPH UN-- 88 107 --
+ attr: name (v/c)
```

We use only the largest subnetwork in this example. There are 88 proteins and 107 interactions.

```
V(sg)$color <- "green"
V(sg)$label.font <- 3
V(sg)$label.cex <- 1
V(sg)$label.color <- "black"
V(sg)[1:10]$color <- "blue"
```

```
plot(sg,layout=layout.kamada.kawai(sg),vertex.size=10)
```

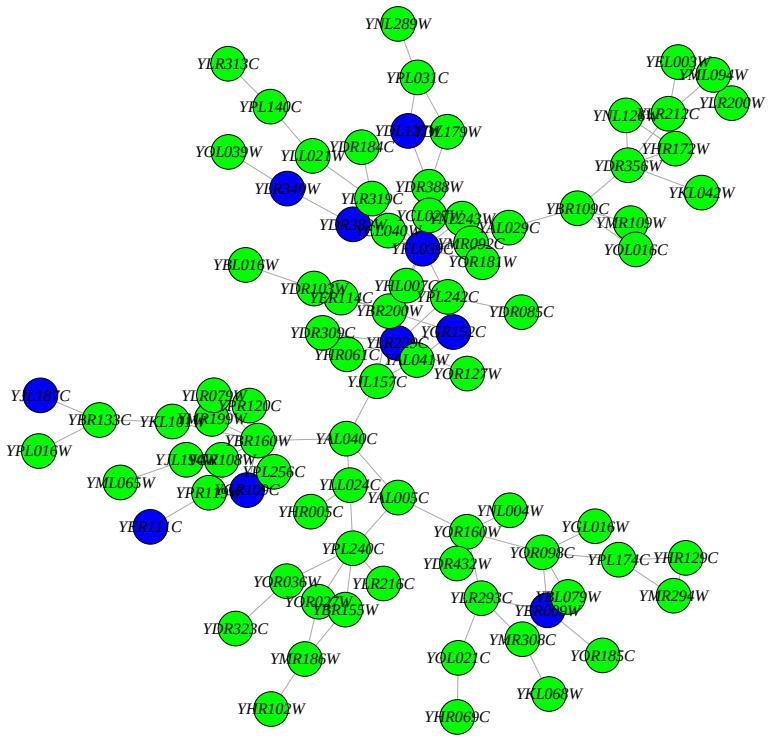

Figure 1: Network among yeast proteins with target class in blue and remaining proteins in green.

First, calculate the kernel matrix and choose 10 proteins as a target class. Then we can find proteins closely related to the target class by using the KSVM for a graph (Samatova *et al.*, 2013; Kolaczyk & Csardi, 2014). Network of interactions among proteins with target class in blue and backgrounds in green. Red vertices represent the top 20 proteins which are most closely related to the target class.

```
K <- net.kernel(sg)
litG.infer <- net.infer(names(V(sg))[1:10],K,top=20)
index <- match(litG.infer$top,names(V(sg)))
V(sg)[index]$color <- "red"

plot(sg,layout=layout.kamada.kawai(sg),vertex.size=10)
```

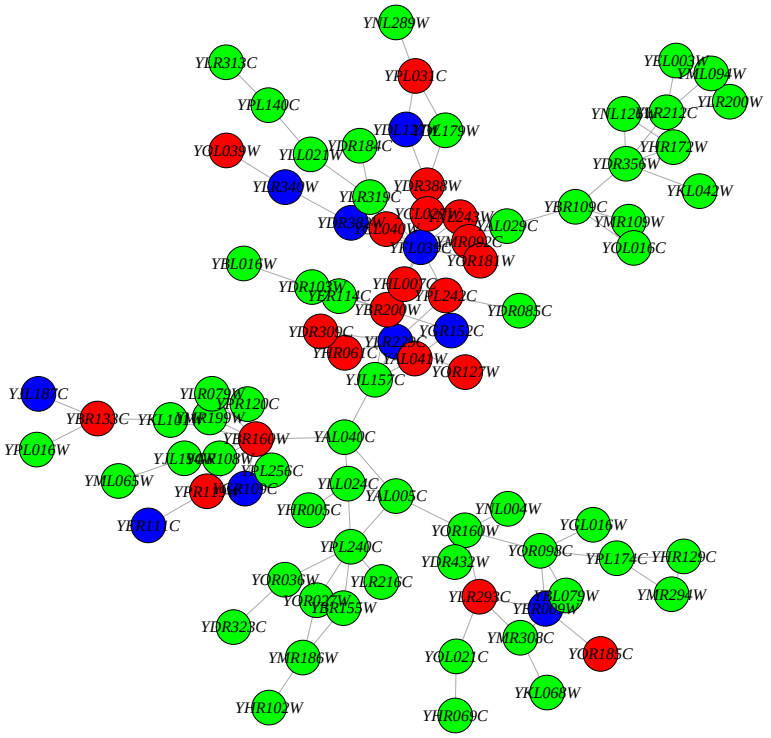

Figure 2: Red vertices denote the top 20 yeast proteins which are most closely related to 10 proteins of the target class.

Note that the number of proteins is not greater than a half of total proteins in a kernel matrix due to $N - 2n > 0$. Also, the number of top proteins to be inferred is less than or equal to $N - 2n$. If we use 50 proteins as a target class, then there is an error since $N - 2n = -12$. If we use 40 proteins as a target, and want to find top 20 proteins, then the number of available top proteins are only 8, which is the minimum of $N - 2n = 8$ and 20.

```
litG.infer <- try(net.infer(names(V(sg))[1:50],K,top=20))
cat(litG.infer)
```

```
Error in net.infer(names(V(sg))[1:50], K, top = 20) :
  size of list is too large
```

```
litG.infer <- net.infer(names(V(sg))[1:40],K,top=20)
litG.infer$top
```

```
[1] "YBR160W" "YDL179W" "YAL041W" "YDR323C" "YBR133C" "YPL174C" "YPL140C"
[8] "YDR356W"
```

5 Protein-protein interaction

5.1 Download the kernel matrix

We need a list of proteins and the kernel matrix to infer functionally related proteins. For the database for kernel matrix, we use the STRING data for protein-protein interactions for human and mouse. You do not have to calculate these kernel matrices. To save significant time, you can download kernel matrices at `http://ge-lab.org/dm/K9606.rds` for human and `http://ge-lab.org/dm/K10090.rds` for mouse.

```
# K.9606 <- readRDS(gzcon(url("http://ge-lab.org/dm/K9606.rds")))
# K.10090 <- readRDS(gzcon(url("http://ge-lab.org/dm/K10090.rds")))

library(httr)
download.kernel <- function(species, overwrite = FALSE)
{
  # If they exist, load the matrix. These kernels are calculated from
  # the STRING database with the version 10 and score  threshold 400.
  # There is a progress bar for downloading.
  # species : Only two species, "9606" for human or "10090" for mouse
  # overwrite : overwrite existing files (default: FALSE)

  URL <- paste("http://ge-lab.org/dm/K", species, ".rds", sep="")
  if(overwrite)
  {
    GET(URL, write_disk(paste("K", species, ".rds", sep=""),
                        overwrite = overwrite), progress())
  }
  else
  {
    if(!file.exists(paste("K", species, ".rds", sep="")))
```

```
    {
      GET(URL, write_disk(paste("K", species, ".rds", sep="")), progress())
    }
  }

  K <- readRDS(paste("K", species, ".rds", sep=""))
  # remove prefix
  rownames(K) <- sub('.*\\.','',rownames(K))
  colnames(K) <- sub('.*\\.','',colnames(K))
  K
}
```

5.2 Visualize the overrepresentation analysis

For the functional enrichment analysis, we can visualize or summarize the result from the overrepresentation analysis.

```
library(ggplot2)
ORA.plot <- function(object, categories = 10, p.adjust.methods = 'fdr',
                     color = c("red", "blue"), transparency = 0.5, plot = TRUE)
{
  # Visualize the ORA or summarize the result when plot = FALSE.
  # object : the output from hyperGTest
  # categories : the number of categories (default: 10)
  # p.adjust.methods : correction method (default: "fdr")
  # color : colors for low and high ends of the gradient.
  # (default: "red" for low and "blue" for high)
  # transparency : transparency (default: 0.5)
  # plot : summary of hyperGTest, if FALSE (default: TRUE)

  result <- summary(object, TRUE)
  adjPvalue <- p.adjust(pvalues(object), p.adjust.methods)
  index <- match(result[,1], names(adjPvalue))
  adjPvalue <- adjPvalue[index]
  GOID <- result[index,1]
  GeneRatio <- result[index,]$Count/result[index,]$Size
  results <- data.frame(GOID, GeneRatio, adjPvalue, result[index,-1])
  rownames(results) <- NULL
  head(results, categories)

  if(plot == TRUE)
```

```
  {
    sub.res <- results[1:categories,]
    sub.res$GOID <- factor(sub.res$GOID, levels=sub.res$GOID[nrow(sub.res):1])
    ggplot(sub.res, aes_string(x='GeneRatio', y="GOID",
                               size='Count', color='adjPvalue')) +
      geom_point(alpha = transparency) +
      scale_color_gradient(low = color[1], high = color[2]) + ylab('')
  }
  else
  {
    head(results, categories)
  }
}
```

5.3 PPI for human

Consider two examples for human. First, we can find proteins related to apoptosis. Then, load the kernel matrix for human.

```
library(graph)
library(GOstats)
library(hgu95av2.db)
# download kernel matrix
K.9606 <- download.kernel("9606")
# load target class from KEGG apoptosis pathway
data(apopGraph)
list.proteins <- nodes(apopGraph)
head(list.proteins)
```

```
[1] "FasL"           "TNFa"           "TNFb"           "CD40L"
[5] "Glucocorticoid" "NGF"
```

There are many types of protein ID or gene ID. By using `getBM` in `biomaRt`, we can change the format. For this reason, input and output formats must be available for `getBM`. Note that the number of proteins used as a target may be different from the number of proteins in the input since mapping between formats is not always one-to-one in `getBM`.

```
# find top 100 proteins
apoptosis.infer <- ppi.infer.human(list.proteins, K.9606, output="entrezgene", 100)
gene.id <- data.frame(apoptosis.infer$top)[,1]
head(gene.id)
```

```
[1] 1677  637 8738  638 1676 1149
```

```
 # functional enrichment
 params <- new("GOHyperGParams", geneIds=gene.id,annotation="org.Hs.eg.db",
              ontology="BP",pvalueCutoff=0.001,conditional=FALSE,
              testDirection="over")
 (hgOver <- hyperGTest(params))
```

```
Gene to GO BP  test for over-representation
2636 GO BP ids tested (470 have p < 0.001)
Selected gene set size: 99
    Gene universe size: 16655
    Annotation package: org.Hs.eg
```

```
 # top 10 categories
 (results = ORA.plot(hgOver, plot=FALSE))
```

```
         GOID  GeneRatio    adjPvalue       Pvalue OddsRatio  ExpCount Count
1  GO:0042981 0.05303030 8.146666e-59 3.456863e-62  38.64255  8.630922    77
2  GO:0043067 0.05263158 8.146666e-59 6.181082e-62  38.30808  8.696307    77
3  GO:0010941 0.05022537 1.471165e-58 1.674316e-61  37.97637  9.231282    78
4  GO:0008219 0.04187689 3.499725e-58 6.638324e-61  40.03854 11.781327    83
5  GO:0016265 0.04187689 3.499725e-58 6.638324e-61  40.03854 11.781327    83
6  GO:0006915 0.04359526 2.145320e-57 4.883126e-60  37.42572 11.044251    81
7  GO:0012501 0.04315397 4.172856e-57 1.108118e-59  36.98218 11.157190    81
8  GO:0097190 0.07582938 3.113794e-39 9.450058e-42  25.69492  3.762654    48
9  GO:2000116 0.15668203 3.037162e-37 1.036967e-39  46.79966  1.289883    34
10 GO:0043065 0.07666099 1.206292e-36 4.576223e-39  24.62177  3.489222    45
   Size                                                 Term
1  1452                    regulation of apoptotic process
2  1463                regulation of programmed cell death
3  1553                           regulation of cell death
4  1982                                         cell death
5  1982                                              death
6  1858                                 apoptotic process
7  1877                            programmed cell death
8   633                       apoptotic signaling pathway
9   217 regulation of cysteine-type endopeptidase activity
10  587           positive regulation of apoptotic process
```

```
ORA.plot(hgOver)
```

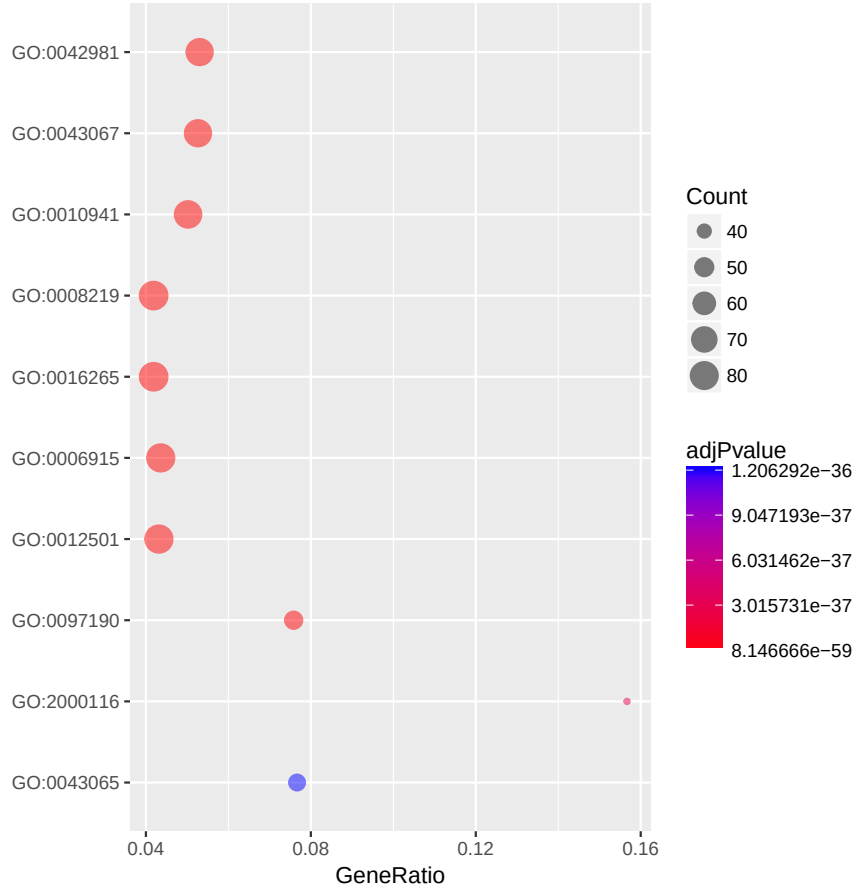

Figure 3: Top 10 biological functions related to apoptosis

We found functionally related top 100 proteins in terms of Entrez-ID by `ppi.infer.human`. However, we are often interested in biological functions about such inferred proteins. This is the top 10 categories from gene ontology. As we expected, inferred proteins have similar biological functions with the target, apoptosis. Thus this example supports that this model is reliable. Next, take another example. The protein p53 is known for inhibition of cancer. From KEGG pathway, we can find proteins for p53 signaling pathway. The procedure is the same to the previous example, but for the target class.

```
library(KEGGgraph)
library(KEGG.db)
# load target class for p53
pName <- "p53 signaling pathway"
pId <- mget(pName, KEGGPATHNAME2ID)[[1]]
p53 <- system.file("extdata/hsa04115.xml",package="KEGGgraph")
p53graph <- parseKGML2Graph(p53,expandGenes=TRUE)
entrez <- translateKEGGID2GeneID(nodes(p53graph))
head(entrez)
```

```
[1] "6477"  "3486"  "10912" "1647"  "4616"  "4194"
```

```
 # find top 100 proteins
 p53.infer <- ppi.infer.human(entrez,K.9606,input="entrezgene",
                              output="entrezgene",100)
 gene.id <- data.frame(p53.infer$top)[,1]
 head(gene.id)
```

```
[1] 23612  7832 57103 28984 94241 10848
```

```
 # functional enrichment
 params <- new("GOHyperGParams", geneIds=gene.id,annotation="hgu95av2.db",
               ontology="BP",pvalueCutoff=0.001,conditional=FALSE,
               testDirection="over")
 (hgOver <- hyperGTest(params))
```

```
Gene to GO BP  test for over-representation
3344 GO BP ids tested (536 have p < 0.001)
Selected gene set size: 82
    Gene universe size: 8080
    Annotation package: hgu95av2
```

```
 # top 10 categories
 (results = ORA.plot(hgOver, plot=FALSE))
```

```
         GOID  GeneRatio    adjPvalue       Pvalue OddsRatio  ExpCount Count
1  GO:0006915 0.04461538 5.053693e-25 1.986494e-28 13.145733 13.193069    58
2  GO:0012501 0.04427481 5.053693e-25 3.022543e-28 13.021432 13.294554    58
3  GO:0008219 0.04181687 5.698476e-24 6.816358e-27 12.126975 14.075990    58
4  GO:0016265 0.04181687 5.698476e-24 6.816358e-27 12.126975 14.075990    58
5  GO:0097190 0.08016878 7.838142e-23 1.171971e-25 14.978941  4.810396    38
6  GO:0097193 0.11711712 1.431821e-18 2.569057e-21 18.481414  2.252970    26
7  GO:0006974 0.06910569 5.604246e-18 1.328336e-20 11.661208  4.993069    34
8  GO:0042981 0.04393505 5.604246e-18 1.340729e-20  8.931679 10.625495    46
9  GO:0043067 0.04360190 6.845840e-18 1.842481e-20  8.850732 10.706683    46
10 GO:0010942 0.07127430 7.334717e-18 2.289018e-20 11.853061  4.698762    33
   Size                                    Term
1  1300                    apoptotic process
2  1310                programmed cell death
3  1387                           cell death
4  1387                                death
5   474           apoptotic signaling pathway
6   222  intrinsic apoptotic signaling pathway
7   492 cellular response to DNA damage stimulus
```

```
8  1047          regulation of apoptotic process
9  1055       regulation of programmed cell death
10  463         positive regulation of cell death
```

```
ORA.plot(hgOver)
```

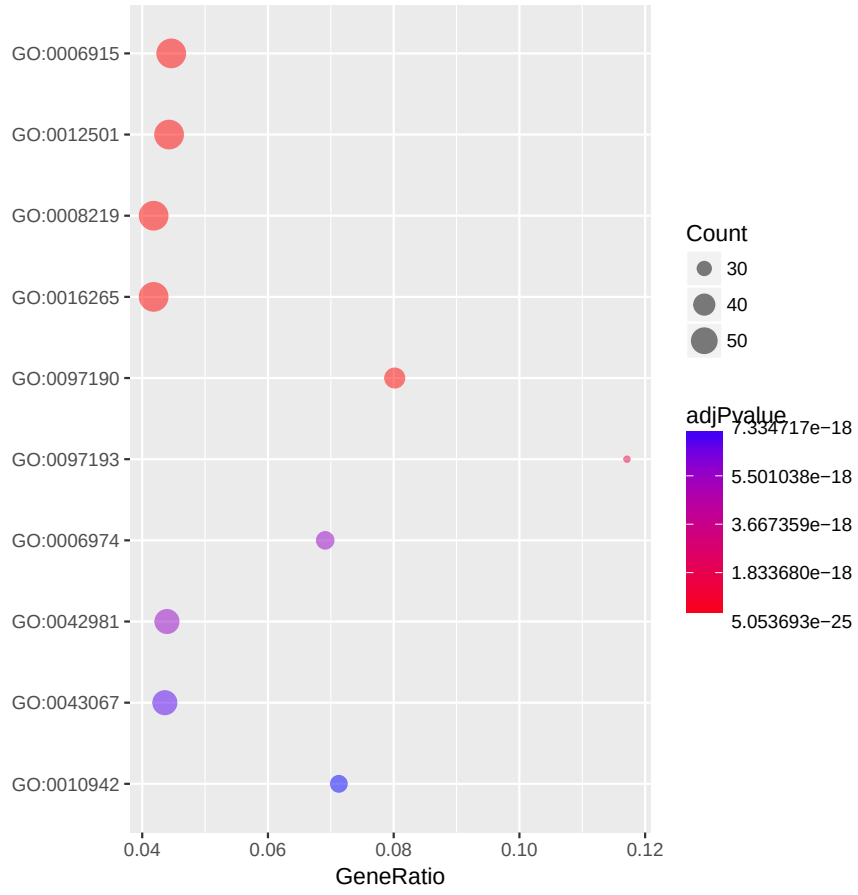

Figure 4: Top 10 biological functions related to p53 signaling pathway

5.4 PPI for mouse

For mouse, we can infer functionally related proteins by `ppi.infer.mouse` with the kernel matrix for mouse. The first example is Acute myeloid leukemia.

```
library(limma)
library(org.Mm.eg.db)
# download kernel matrix
K.10090 <- download.kernel("10090")
# load target class
mget('Acute myeloid leukemia', KEGGPATHNAME2ID)
```

```
$`Acute myeloid leukemia`
[1] "05221"
```

```
kegg.mmu <- getGeneKEGGLinks(species.KEGG='mmu')
index <- match(kegg.mmu[,2],'path:mmu05221')
path.05221 <- 1:length(index)
index2 <- path.05221[index]
path.05221 <- path.05221[is.na(index2)==FALSE]
path.05221 <- kegg.mmu[path.05221,1]
head(path.05221)
```

```
[1] "109880" "110157" "11651"  "11652"  "11836"  "12015"
```

```
# find top 100 proteins
path.05221.infer <- ppi.infer.mouse(path.05221,K.10090,
                                    input="entrezgene",output="entrezgene",100)
gene.id <- data.frame(path.05221.infer$top)[,1]
head(gene.id)
```

```
[1]  16369 110279  16367  12445  26420  14389
```

```
# functional enrichment
params <- new("GOHyperGParams", geneIds=gene.id,annotation="org.Mm.eg.db",
              ontology="BP",pvalueCutoff=0.001,conditional=FALSE,
              testDirection="over")
(hgOver <- hyperGTest(params))
```

```
Gene to GO BP  test for over-representation
4038 GO BP ids tested (1406 have p < 0.001)
Selected gene set size: 99
    Gene universe size: 23276
    Annotation package: org.Mm.eg
```

```
# top 10 categories
(results = ORA.plot(hgOver, plot=FALSE))
```

	GOID	GeneRatio	adjPvalue	Pvalue	OddsRatio	ExpCount	Count
1	GO:0048522	0.01937899	8.812719e-47	2.182446e-50	33.63845	19.314272	88
2	GO:0048518	0.01749214	2.636829e-44	1.306007e-47	32.36331	21.640832	89
3	GO:0031325	0.02680867	9.904453e-44	9.211670e-47	21.74849	11.581758	73
4	GO:0035556	0.03125000	9.904453e-44	9.811246e-47	21.92404	9.255198	68
5	GO:0009893	0.02414231	2.584960e-43	3.200793e-46	21.62562	13.389414	76
6	GO:0042981	0.04208999	1.351533e-42	2.228950e-45	23.42398	5.861059	58
7	GO:0006915	0.03632103	1.351533e-42	2.342925e-45	21.93353	7.260397	62
8	GO:0043067	0.04160689	2.164363e-42	4.287990e-45	23.12652	5.929112	58
9	GO:0012501	0.03575548	2.704815e-42	6.028562e-45	21.55228	7.375236	62

```
10 GO:0010941 0.03922872 5.837841e-42 1.445726e-44  22.18318  6.396975    59
   Size                                          Term
1  4541           positive regulation of cellular process
2  5088         positive regulation of biological process
3  2723 positive regulation of cellular metabolic process
4  2176                 intracellular signal transduction
5  3148          positive regulation of metabolic process
6  1378                   regulation of apoptotic process
7  1707                               apoptotic process
8  1394               regulation of programmed cell death
9  1734                           programmed cell death
10 1504                         regulation of cell death
```

```
ORA.plot(hgOver)
```

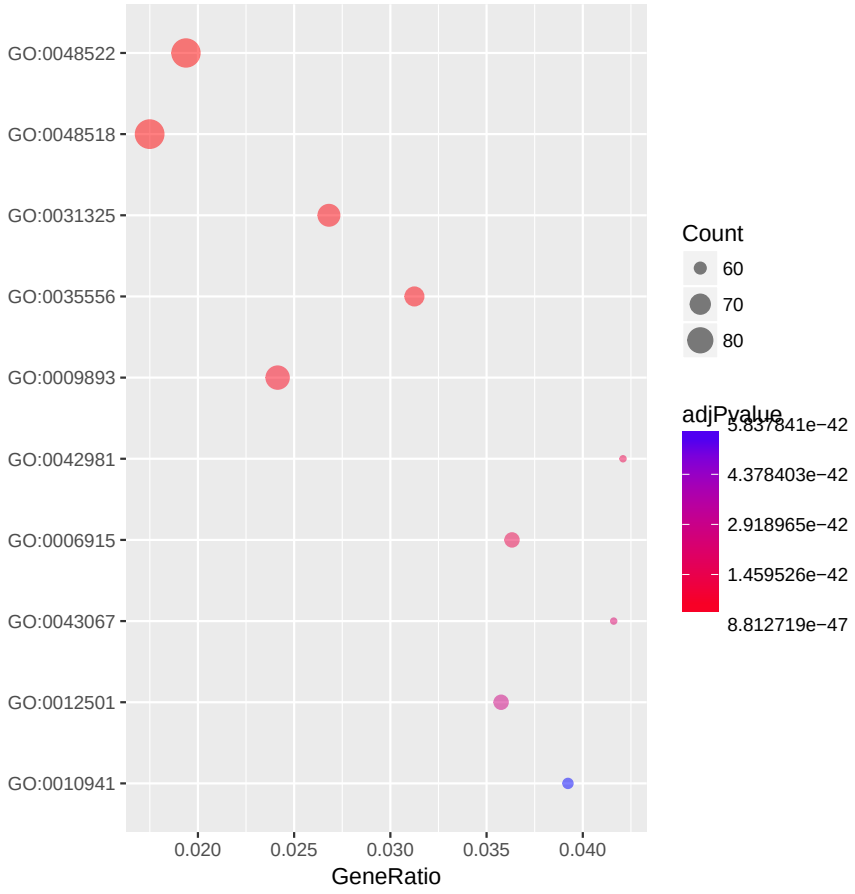

Figure 5: Top 10 biological functions related to Acute myeloid leukemia

The second example is Ras signaling pathway. The Ras proteins are GTPases that function as molecular switches for signaling pathways regulating cell proliferation, survival, growth, migration, differentiation or cytoskeletal dynamism.

```
# load target class
```

```
 mget('Ras signaling pathway', KEGGPATHNAME2ID)
```

```
$`Ras signaling pathway`
[1] "04014"
```

```
 kegg.mmu <- getGeneKEGGLinks(species.KEGG='mmu')
 index <- match(kegg.mmu[,2],'path:mmu04014')
 path.04014 <- 1:length(index)
 index2 <- path.04014[index]
 path.04014 <- path.04014[is.na(index2)==FALSE]
 path.04014 <- kegg.mmu[path.04014,1]
 head(path.04014)
```

```
[1] "100043507" "100503710" "109905"    "110157"    "11350"     "11352"
```

```
 # find top 100 proteins
 path.04014.infer <- ppi.infer.mouse(path.04014,K.10090,
                                     input="entrezgene",output="entrezgene",100)
 gene.id <- data.frame(path.04014.infer$top)[,1]
 head(gene.id)
```

```
[1]  14181  11836  17444 327826 107971  71520
```

```
 # functional enrichment
 params <- new("GOHyperGParams", geneIds=gene.id,annotation="org.Mm.eg.db",
              ontology="BP",pvalueCutoff=0.001,conditional=FALSE,
              testDirection="over")
 (hgOver <- hyperGTest(params))
```

```
Gene to GO BP  test for over-representation
3622 GO BP ids tested (1063 have p < 0.001)
Selected gene set size: 98
    Gene universe size: 23276
    Annotation package: org.Mm.eg
```

```
 # top 10 categories
 (results = ORA.plot(hgOver, plot=FALSE))
```

```
         GOID  GeneRatio    adjPvalue       Pvalue OddsRatio  ExpCount Count
1  GO:0007169 0.08536585 9.015548e-41 2.489108e-44  37.88000  2.071490    42
2  GO:0016310 0.03088608 2.013615e-37 1.111880e-40  18.31602  8.315432    61
3  GO:0007167 0.05625000 1.569972e-36 1.319457e-39  25.21642  3.368276    45
4  GO:0006796 0.02451518 1.569972e-36 1.733817e-39  16.62880 11.506874    67
5  GO:0006793 0.02412676 3.475272e-36 4.797448e-39  16.32372 11.692129    67
```

```
6  GO:0006468 0.03329370 1.517144e-35 2.513215e-38  17.67282  7.081801    56
7  GO:0035556 0.02757353 2.633654e-34 5.089890e-37  15.71635  9.161712    60
8  GO:1902531 0.03351955 1.975145e-29 4.362551e-32  15.11723  6.029215    48
9  GO:0042325 0.03379868 4.066157e-28 1.010365e-30  14.70749  5.730280    46
10 GO:0042127 0.03188602 8.480999e-28 2.341524e-30  14.04698  6.206049    47
   Size                                                                Term
1   492 transmembrane receptor protein tyrosine kinase signaling pathway
2  1975                                                      phosphorylation
3   800                  enzyme linked receptor protein signaling pathway
4  2733                   phosphate-containing compound metabolic process
5  2777                                        phosphorus metabolic process
6  1682                                             protein phosphorylation
7  2176                                  intracellular signal transduction
8  1432                  regulation of intracellular signal transduction
9  1361                                     regulation of phosphorylation
10 1474                                   regulation of cell proliferation
```

ORA.plot(hgOver)

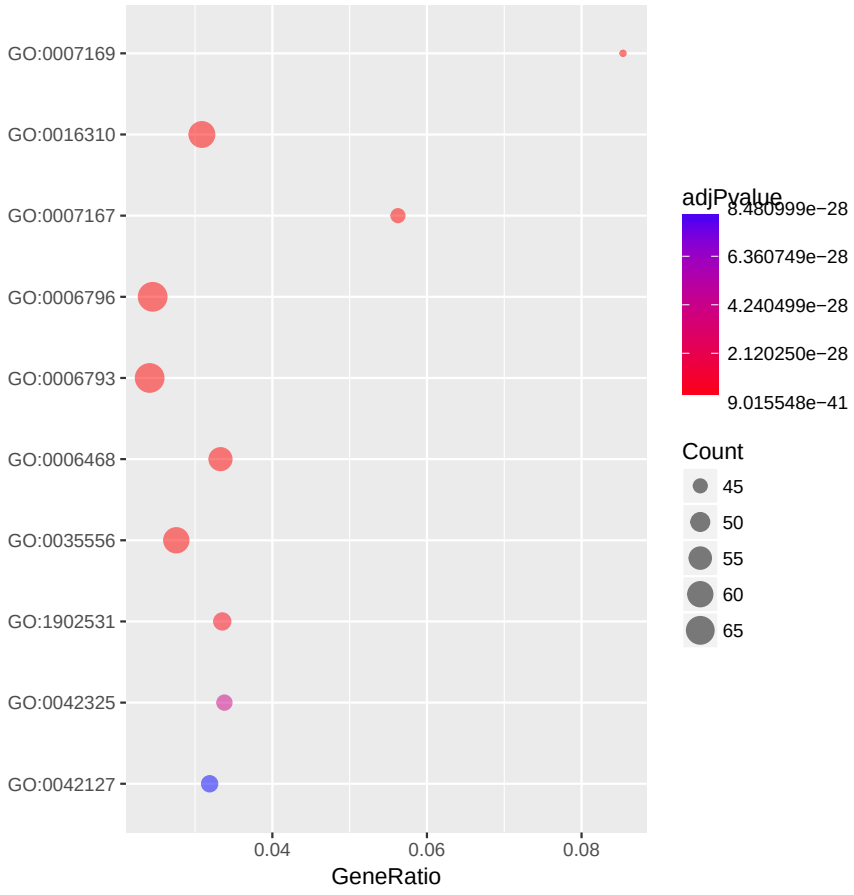

Figure 6: Top 10 biological functions related to Ras signaling pathway

We discussed about how to infer functionally related proteins for human and mouse. Two functions `ppi.infer.human` and `ppi.infer.mouse` are specially designed because popular organisms are human and mouse. However, other kinds of species are also available in `net.infer` if kernel matrices are given.

5.5 PPI for other species

```
# yeast
string.db.4932 <- STRINGdb$new(version='10', species = 4932)
string.db.4932.graph <- string.db.4932$get_graph()
K.4932 <- net.kernel(string.db.4932.graph)
rownames(K.4932) <- sub('.*\\.','',rownames(K.4932))
colnames(K.4932) <- sub('.*\\.','',colnames(K.4932))
net.infer(rownames(K.4932)[1:100], K.4932, top=100)
```

6 Session Information

```
sessionInfo()
```

```
R version 3.3.1 (2016-06-21)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.11.6 (El Capitan)

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

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

other attached packages:
 [1] org.Mm.eg.db_3.3.0   limma_3.28.21        KEGG.db_3.2.3
 [4] KEGGgraph_1.30.0     GO.db_3.3.0          hgu95av2.db_3.2.3
 [7] org.Hs.eg.db_3.3.0   GOstats_2.38.1       Category_2.38.0
[10] Matrix_1.2-6         AnnotationDbi_1.34.4 IRanges_2.8.2
[13] S4Vectors_0.12.2     Biobase_2.32.0       ggplot2_2.2.0
[16] httr_1.2.1           PPInfer_1.0.1        yeastExpData_0.20.0
[19] graph_1.50.0         BiocGenerics_0.20.0  STRINGdb_1.12.0
[22] kernlab_0.9-24       igraph_1.0.1         biomaRt_2.28.0

loaded via a namespace (and not attached):
 [1] genefilter_1.54.2      gtools_3.5.0           splines_3.3.1
```

```
 [4] lattice_0.20-33        colorspace_1.2-6       hash_2.2.6
 [7] chron_2.3-47           XML_3.98-1.4           RBGL_1.48.1
[10] survival_2.40-1        DBI_0.5-1              RColorBrewer_1.1-2
[13] gsubfn_0.6-6           plyr_1.8.4             munsell_0.4.3
[16] gtable_0.2.0           caTools_1.17.1         labeling_0.3
[19] GSEABase_1.34.1        proto_0.3-10           Rcpp_0.12.6
[22] KernSmooth_2.23-15     xtable_1.8-2           scales_0.4.1
[25] gdata_2.17.0           plotrix_3.6-3          annotate_1.50.0
[28] gplots_3.0.1           digest_0.6.10          png_0.1-7
[31] grid_3.3.1             tools_3.3.1            bitops_1.0-6
[34] magrittr_1.5           sqldf_0.4-10           RCurl_1.95-4.8
[37] lazyeval_0.2.0         RSQLite_1.0.0          tibble_1.2
[40] assertthat_0.1         AnnotationForge_1.14.2 R6_2.1.3
```

7 References

Kolaczyk, E. D. & Csardi, G. (2014). *Statistical analysis of network data with R*. Springer.

Ma, Y. (2014). *Support vector machines applications*. G. Guo (Ed.). Springer.

Samatova, *et al.* (Eds.). (2013). *Practical graph mining with R*. CRC Press.

Senay, S. D. *et al.* (2013). Novel three-step pseudo-absence selection technique for improved species distribution modelling. *PLOS ONE*. **8(8)**, e71218.

Smola, A. J. & Kondor, R. (2003). Kernels and regularization on graphs. *In Learning theory and kernel machines*. 144-158. Springer Berlin Heidelberg.

Werther, M., & Seitz, H. (Eds.). (2008). *Protein-protein interaction*. Springer.