

PPIInfer: Inferring functionally related proteins using protein interaction networks

Dongmin Jung, Xijin Ge

July 1, 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.

Semi-supervised learning can be applied to make use of large unlabeled data and small labeled data. Some of these methods directly try to label the unlabeled data. Self-training is a commonly used semi-supervised learning technique (Zhu, 2006). Self-training is an incremental algorithm that initially builds a classifier using a small amount of labeled data. So it iteratively predicts the labels of the unlabeled data and then predicted labels are added to the labeled data. Here, the function `net.infer.ST` is the self-training method for SVM. Also, the function `net.infer` is the special case of `net.infer.ST` where a single iteration is conducted.

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)
```

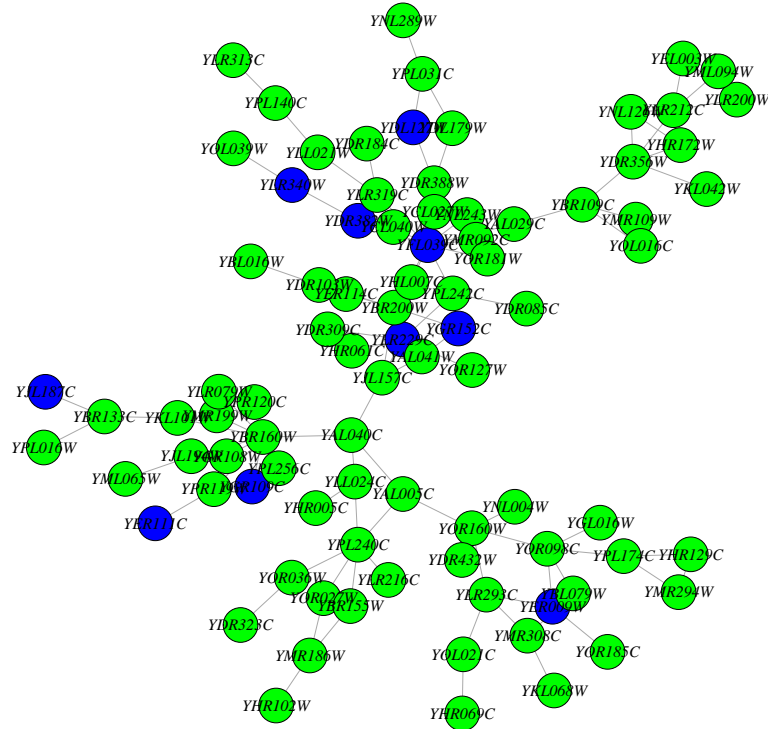


Figure 1: Network among yeast proteins with target class in blue and remaining proteins in green.

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)
set.seed(123)
litG.infer <- net.infer(names(V(sg))[1:10], K, top=20, cross = 10)
litG.infer$Cverror

[1] 0.45

index <- match(litG.infer$top,names(V(sg)))
V(sg)[index]$color <- "red"
```

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

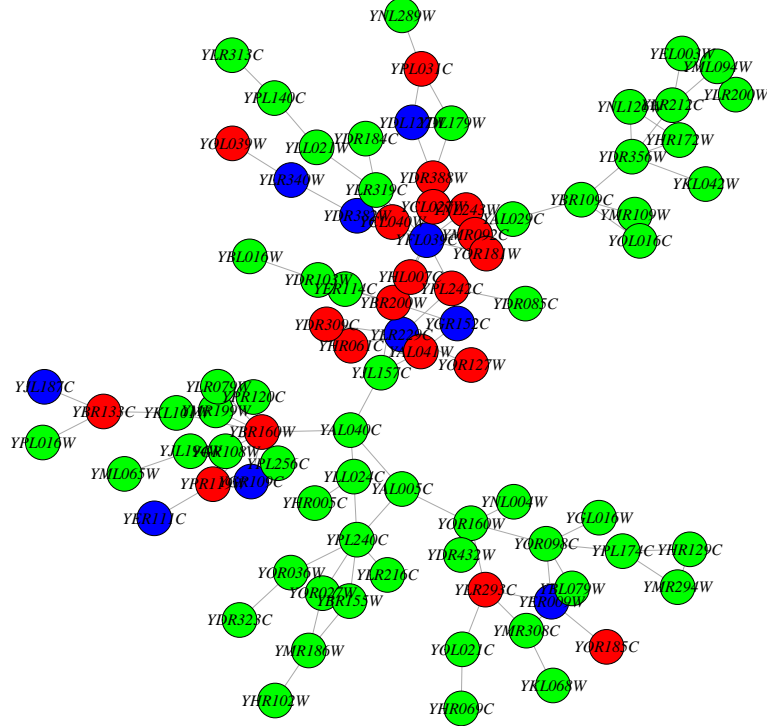


Figure 2: Red vertices denote the top 20 yeast proteins which are most closely related to 10 proteins of the target class.

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)
  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(GOstats)
# 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))
```


	G0ID	GeneRatio	adjPvalue	Pvalue	OddsRatio	ExpCount	Count
1	G0:0042981	0.05303030	8.146666e-59	3.456863e-62	38.64255	8.630922	77
2	G0:0043067	0.05263158	8.146666e-59	6.181082e-62	38.30808	8.696307	77
3	G0:0010941	0.05022537	1.471165e-58	1.674316e-61	37.97637	9.231282	78
4	G0:0008219	0.04187689	3.499725e-58	6.638324e-61	40.03854	11.781327	83
5	G0:0016265	0.04187689	3.499725e-58	6.638324e-61	40.03854	11.781327	83
6	G0:0006915	0.04359526	2.145320e-57	4.883126e-60	37.42572	11.044251	81
7	G0:0012501	0.04315397	4.172856e-57	1.108118e-59	36.98218	11.157190	81
8	G0:0097190	0.07582938	3.113794e-39	9.450058e-42	25.69492	3.762654	48
9	G0:2000116	0.15668203	3.037162e-37	1.036967e-39	46.79966	1.289883	34
10	G0:0043065	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)
```

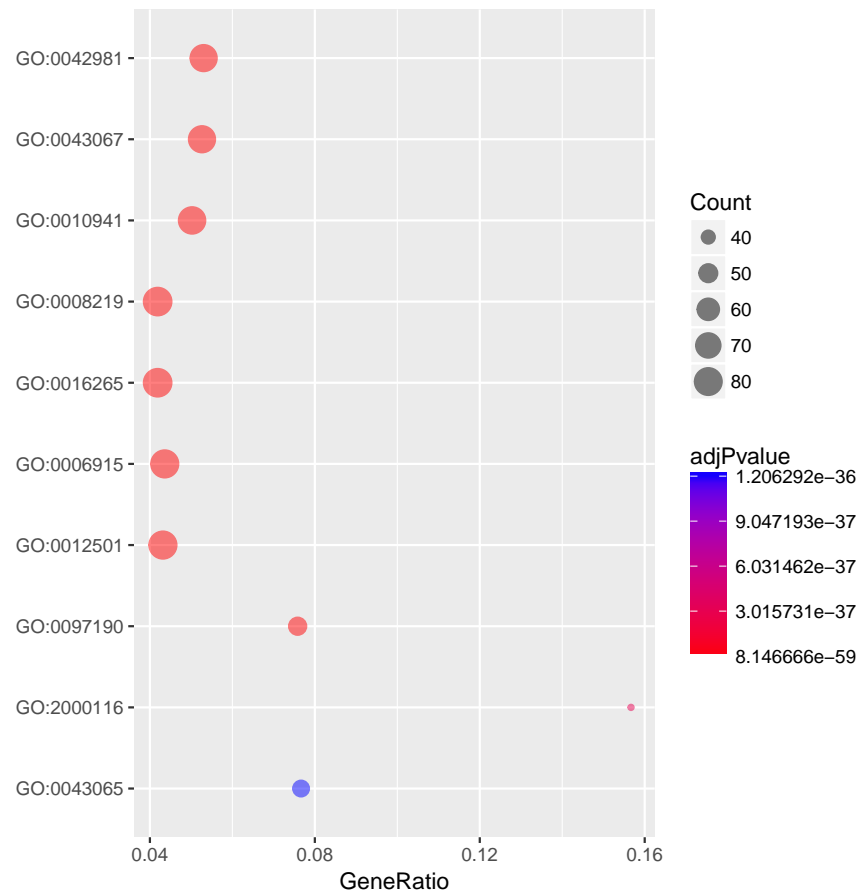


Figure 3: Top 10 biological functions related to apoptosis

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(KEGG.db)
library(limma)
# load target class for p53
mget('p53 signaling pathway', KEGGPATHNAME2ID)

$p53 signaling pathway`
[1] "04115"

kegg.hsa <- getGeneKEGGLinks(species.KEGG='hsa')
index <- match(kegg.hsa[,2], 'path:hsa04115')
path.04115 <- 1:length(index)
```

```

index2 <- path.04115[index]
path.04115 <- path.04115[is.na(index2)==FALSE]
path.04115 <- kegg.hsa[path.04115,1]
head(path.04115)

[1] "1017" "1019" "1021" "1026" "1029" "10912"

hsa04115.infer <- ppi.infer.human(path.04115,K.9606,input="entrezgene",
                                output="entrezgene",100)
gene.id <- data.frame(hsa04115.infer$top)[,1]
head(gene.id)

[1] 23612 7832 57103 28984 94241 10848

rm(K.9606)

# 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
3594 GO BP ids tested (790 have p < 0.001)
Selected gene set size: 98
  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:0006915 0.03713671 5.138835e-40 1.451012e-43 19.64095 10.932693   69
2  GO:0012501 0.03676079 5.138835e-40 2.859675e-43 19.40954 11.044491   69
3  GO:0008219 0.03481332 9.594813e-39 1.067870e-41 18.21360 11.662324   69
4  GO:0016265 0.03481332 9.594813e-39 1.067870e-41 18.21360 11.662324   69
5  GO:0097190 0.06951027 1.226541e-33 1.706373e-36 22.08992 3.724647   44
6  GO:0042981 0.03925620 1.760436e-32 2.938959e-35 15.11031 8.543741   57
7  GO:0043067 0.03896104 2.272697e-32 4.426510e-35 14.98121 8.608466   57
8  GO:0010941 0.03734707 3.514031e-32 7.821994e-35 14.60863 9.138037   58
9  GO:0010942 0.06720000 1.817816e-31 4.552126e-34 20.54974 3.677574   42
10 GO:0043065 0.06814310 5.463825e-30 1.520263e-32 20.18534 3.453978   40

      Size      Term
1  1858      apoptotic process

```

2	1877	programmed cell death
3	1982	cell death
4	1982	death
5	633	apoptotic signaling pathway
6	1452	regulation of apoptotic process
7	1463	regulation of programmed cell death
8	1553	regulation of cell death
9	625	positive regulation of cell death
10	587	positive regulation of apoptotic process

ORA.plot(hgOver)

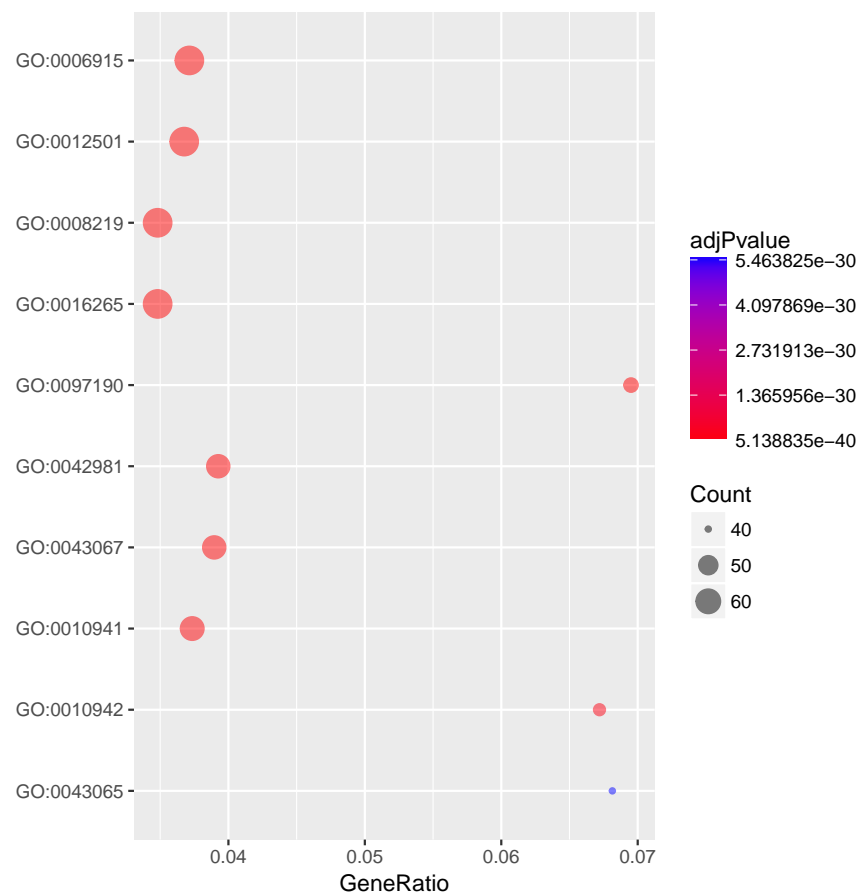


Figure 4: Top 10 biological functions related to p53 signaling pathway

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.

```
# 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
3986 GO BP ids tested (1431 have p < 0.001)
Selected gene set size: 100
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 5.872658e-46 1.473321e-49 30.83360 19.509366 88
2  GO:0043067 0.04304161 3.478326e-44 1.745271e-47 24.55997 5.989002 60
3  GO:0012501 0.03690888 4.108708e-44 3.092354e-47 22.89395 7.449734 64
4  GO:0009893 0.02445997 5.203936e-44 5.506297e-47 21.91730 13.524661 77

```

5	G0:0010941	0.04055851	5.203936e-44	6.527768e-47	23.55692	6.461591	61
6	G0:0048518	0.01749214	6.205747e-44	9.341316e-47	29.41957	21.859426	89
7	G0:0008219	0.03494624	6.412225e-44	1.126081e-46	22.12121	7.991064	65
8	G0:0016265	0.03483387	6.876083e-44	1.380047e-46	22.04133	8.016841	65
9	G0:0042981	0.04281567	9.522044e-44	2.149985e-46	23.84591	5.920261	59
10	G0:0006915	0.03690685	1.051319e-43	2.637529e-46	22.30085	7.333734	63

	Size	Term
1	4541	positive regulation of cellular process
2	1394	regulation of programmed cell death
3	1734	programmed cell death
4	3148	positive regulation of metabolic process
5	1504	regulation of cell death
6	5088	positive regulation of biological process
7	1860	cell death
8	1866	death
9	1378	regulation of apoptotic process
10	1707	apoptotic process

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

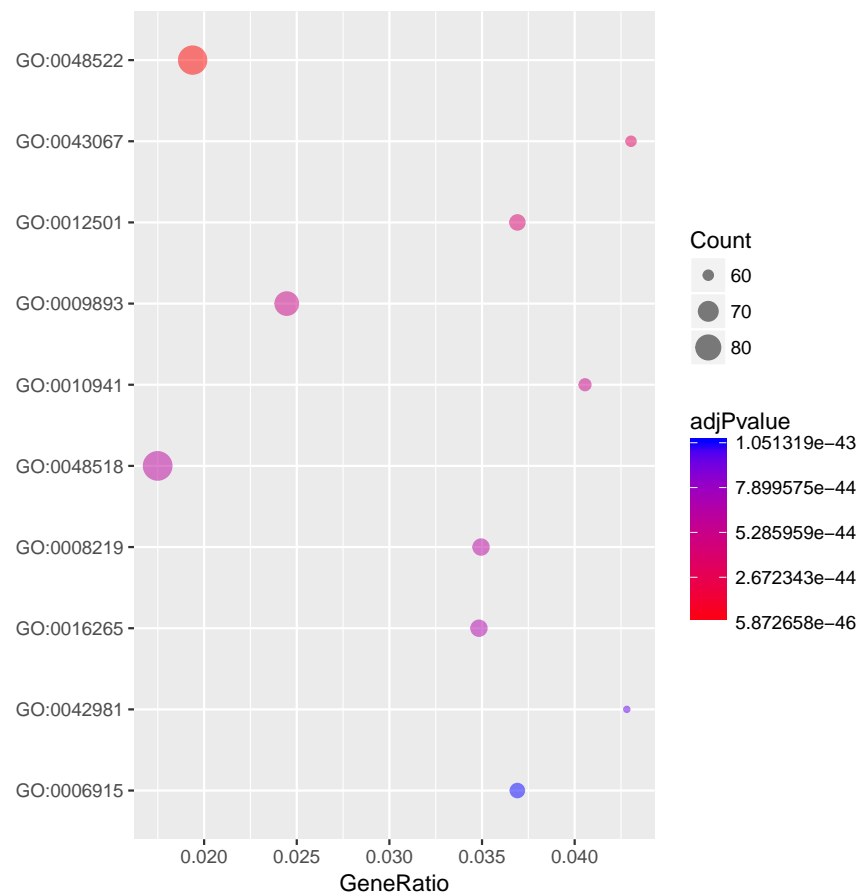


Figure 5: Top 10 biological functions related to Acute myeloid leukemia

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

rm(K.10090)
# 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)
```

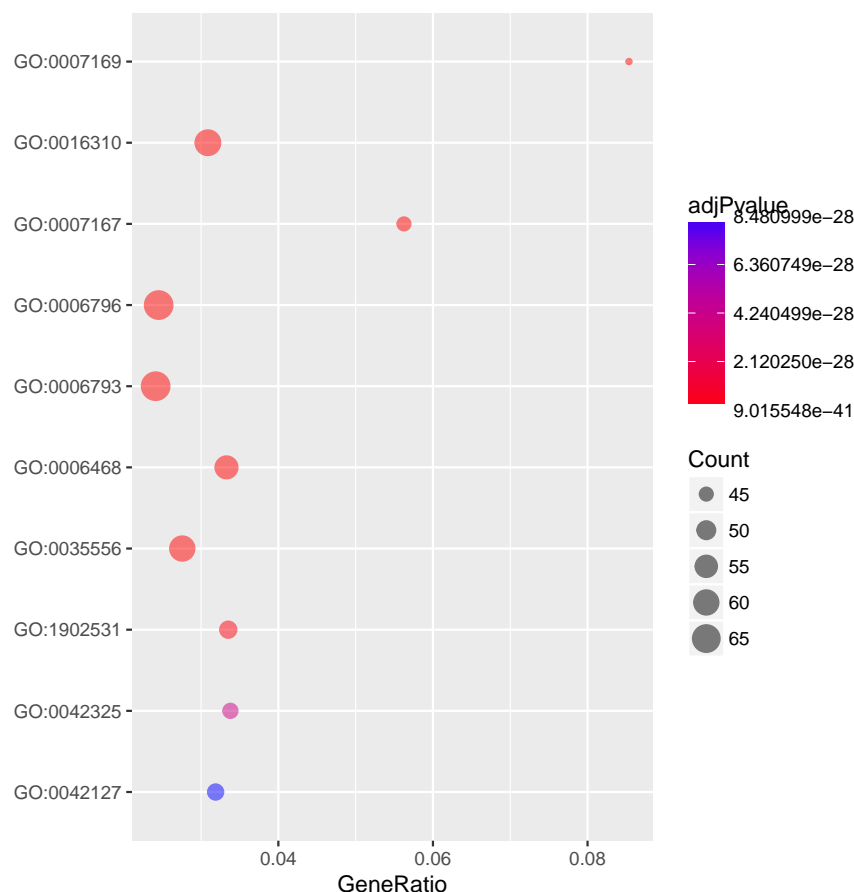


Figure 6: Top 10 biological functions related to Ras signaling pathway

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 organisms

```

##### E. coli
string.db.511145 <- STRINGdb$new(version='10', species = 511145)
string.db.511145.graph <- string.db.511145$get_graph()
K.511145 <- net.kernel(string.db.511145.graph)
rownames(K.511145) <- sub("[:digit:]]+.", "", rownames(K.511145))
colnames(K.511145) <- sub("[:digit:]]+.", "", colnames(K.511145))
# load target class (DNA replication)

```

```

kegg.eco <- getGeneKEGGLinks(species.KEGG='eco')
index <- match(kegg.eco[,2], 'path:eco03030')
path.03030 <- 1:length(index)
index2 <- path.03030[index]
path.03030 <- path.03030[is.na(index2)==FALSE]
path.03030 <- kegg.eco[path.03030,1]
head(path.03030)

[1] "b0183" "b0184" "b0214" "b0215" "b0470" "b0640"

sce03030.infer <- net.infer(path.03030, K.511145, top=100)
gene.id <- data.frame(sce03030.infer$top)[,1]
head(gene.id)

[1] b1652 b1863 b3822 b3652 b3813 b2509
100 Levels: b0060 b0082 b0094 b0095 b0098 b0143 b0160 b0177 b0185 ... b4389

rm(K.511145)

##### 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)
# saveRDS(K.4932, 'K4932.rds')
K.4932 <- readRDS("K4932.rds")
dim(K.4932)

[1] 6418 6418

rownames(K.4932) <- sub("[:digit:]+.", "", rownames(K.4932))
colnames(K.4932) <- sub("[:digit:]+.", "", colnames(K.4932))
# load target class (Cell cycle)
kegg.sce <- getGeneKEGGLinks(species.KEGG='sce')
index <- match(kegg.sce[,2], 'path:sce04111')
path.04111 <- 1:length(index)
index2 <- path.04111[index]
path.04111 <- path.04111[is.na(index2)==FALSE]
path.04111 <- kegg.sce[path.04111,1]
head(path.04111)

[1] "YAL016W" "YAL024C" "YAL040C" "YAR019C" "YBL016W" "YBL023C"

sce04111.infer <- net.infer(path.04111, K.4932, top=100)
gene.id <- data.frame(sce04111.infer$top)[,1]
head(gene.id)

```

```
[1] YPL209C YEL061C YOR058C YMR078C YKL108W YJL090C
100 Levels: YAR007C YAR018C YBL002W YBL003C YBL035C YBL063W YBL088C ... YPR200C
```

```
rm(K.4932)
# functional enrichment
params <- new("GOHyperGParams", geneIds=gene.id,annotation="org.Sc.sgd.db",
              ontology="BP",pvalueCutoff=0.001,conditional=FALSE,
              testDirection="over")
(hgOver <- hyperGTest(params))
```

Gene to GO BP test for over-representation

936 GO BP ids tested (265 have p < 0.001)

Selected gene set size: 100

Gene universe size: 6419

Annotation package: org.Sc.sgd

```
ORA.plot(hgOver, plot = FALSE)
```

	GOID	GeneRatio	adjPvalue	Pvalue	OddsRatio	ExpCount	Count
1	GO:0007049	0.09293194	5.102715e-41	5.451618e-44	19.87590	11.902165	71
2	GO:0022402	0.09459459	2.822661e-35	6.031327e-38	16.14038	10.375448	63
3	GO:0006260	0.21153846	3.431233e-27	1.247582e-29	24.81107	2.430285	33
4	GO:1903047	0.11842105	3.431233e-27	1.466339e-29	14.61493	5.919925	45
5	GO:0000278	0.11363636	1.706156e-26	9.114083e-29	13.91142	6.169185	45
6	GO:0051276	0.12209302	9.853057e-26	6.316062e-28	14.42761	5.359090	42
7	GO:0006261	0.20143885	2.799126e-22	2.093363e-24	21.74975	2.165446	28
8	GO:0006259	0.08614232	4.250729e-22	3.633102e-24	10.17858	8.319053	46
9	GO:0000280	0.11048159	4.363949e-22	4.196104e-24	12.22695	5.499299	39
10	GO:0048285	0.10714286	1.249156e-21	1.334568e-23	11.79148	5.670665	39

	Size	Term
1	764	cell cycle
2	666	cell cycle process
3	156	DNA replication
4	380	mitotic cell cycle process
5	396	mitotic cell cycle
6	344	chromosome organization
7	139	DNA-dependent DNA replication
8	534	DNA metabolic process
9	353	nuclear division
10	364	organelle fission

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

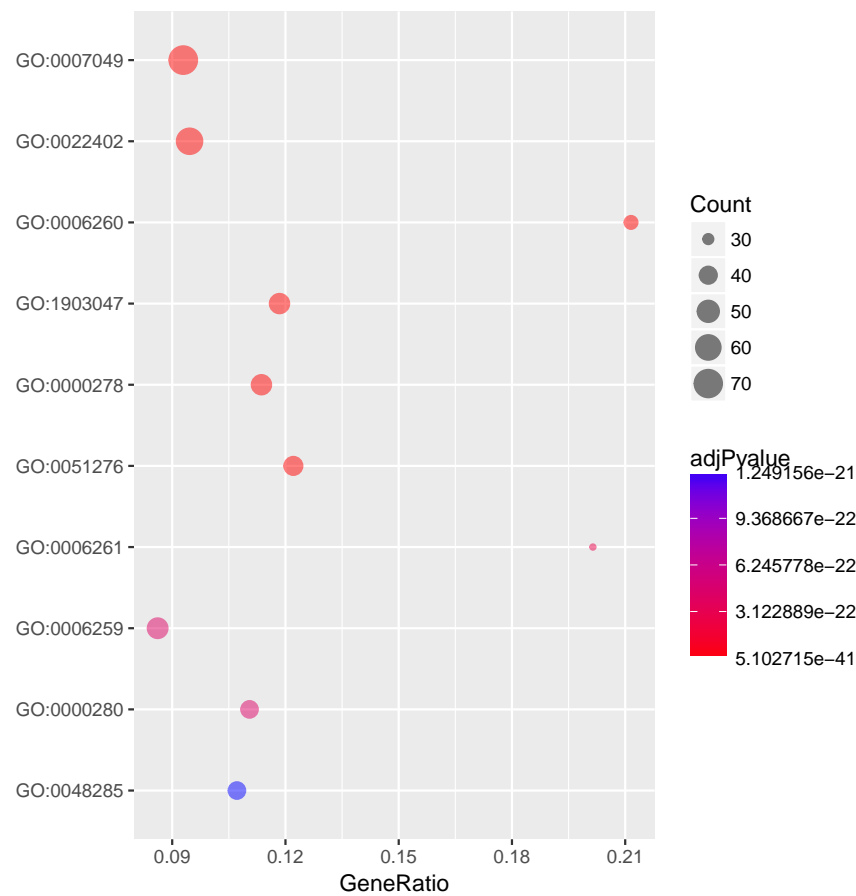


Figure 7: Top 10 biological functions related to Cell cycle

```
##### C. elegans
# string.db.6239 <- STRINGdb$new(version='10', species = 6239)
# string.db.6239.graph <- string.db.6239$get_graph()
# K.6239 <- net.kernel(string.db.6239.graph)
# saveRDS(K.6239, 'K6239.rds')
K.6239 <- readRDS("K6239.rds")
dim(K.6239)

[1] 15830 15830

rownames(K.6239) <- sub("[:digit:]+.", "", rownames(K.6239))
colnames(K.6239) <- sub("[:digit:]+.", "", colnames(K.6239))
# load target class (DNA replication)
kegg.cel <- getGeneKEGGLinks(species.KEGG='cel')
index <- match(kegg.cel[,2], 'path:cel03030')
path.03030 <- 1:length(index)
index2 <- path.03030[index]
path.03030 <- path.03030[is.na(index2)==FALSE]
```

```

path.03030 <- kegg.cel[path.03030,1]
path.03030 <- sub('.*\\_',' ',path.03030)
head(path.03030)

[1] "C04F12.9" "C25D7.6" "C29A12.3" "C39E9.13" "C54G10.2" "F08B4.5"

cel03030.infer <- net.infer(path.03030, K.6239, top=100)
gene.id <- data.frame(cel03030.infer$top)[,1]
head(gene.id)

[1] Y113G7B.24a F59A6.6c T09F3.4 K12D12.5 C24H12.12 C24H12.2
100 Levels: C01F6.8b C07H6.1 C11G6.2 C23H4.6a C24H12.12 C24H12.2 ... ZK675.2

rm(K.6239)
library(org.Ce.eg.db)
gene.id2 <- as.vector(na.omit(select(org.Ce.eg.db,
                                   keys=as.character(gene.id), "ENTREZID",
                                   keytype = 'SYMBOL')[,2])))
# functional enrichment
params <- new("GOHyperGParams", geneIds=gene.id2,annotation="org.Ce.eg.db",
              ontology="BP",pvalueCutoff=0.001,conditional=FALSE,
              testDirection="over")
(hgOver <- hyperGTest(params))

Gene to GO BP test for over-representation
192 GO BP ids tested (27 have p < 0.001)
Selected gene set size: 16
Gene universe size: 10846
Annotation package: org.Ce.eg

ORA.plot(hgOver, plot = FALSE)

      GOID   GeneRatio   adjPvalue      Pvalue OddsRatio   ExpCount Count
1  GO:0006974 0.037037037 4.685554e-08 2.440393e-10  51.06731 0.31864282    8
2  GO:0006259 0.028225806 1.582553e-06 2.882181e-08  34.17381 0.36584916    7
3  GO:0000723 0.166666667 1.582553e-06 3.296986e-08 180.16667 0.03540476    4
4  GO:0032200 0.166666667 1.582553e-06 3.296986e-08 180.16667 0.03540476    4
5  GO:0033554 0.018691589 2.061652e-06 5.368885e-08  24.78571 0.63138484    8
6  GO:0060249 0.137931034 2.347761e-06 7.336754e-08 144.06667 0.04278075    4
7  GO:0006281 0.036764706 3.087761e-05 1.125746e-06  37.12353 0.20062696    5
8  GO:0032392 0.076923077 4.972099e-04 2.330671e-05  69.19231 0.05753273    3
9  GO:0032508 0.076923077 4.972099e-04 2.330671e-05  69.19231 0.05753273    3
10 GO:0006950 0.008163265 5.516468e-04 2.873160e-05  10.14198 1.44569427    8

```

	Size	Term
1	216	cellular response to DNA damage stimulus
2	248	DNA metabolic process
3	24	telomere maintenance
4	24	telomere organization
5	428	cellular response to stress
6	29	anatomical structure homeostasis
7	136	DNA repair
8	39	DNA geometric change
9	39	DNA duplex unwinding
10	980	response to stress

`ORA.plot(hgOver)`

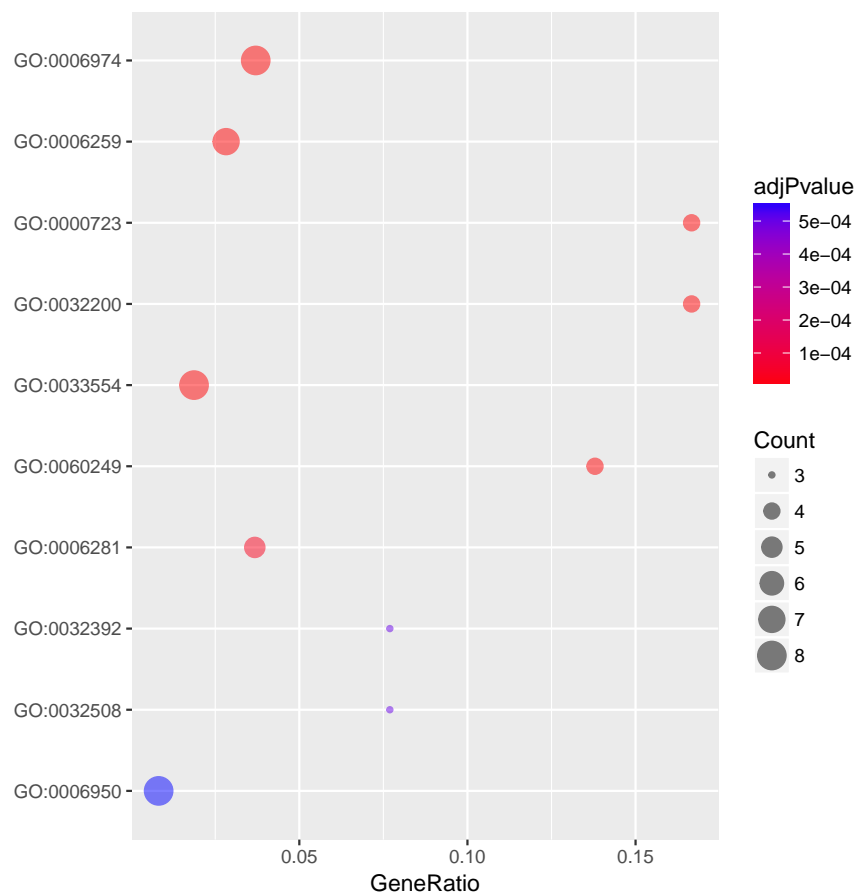


Figure 8: Top 10 biological functions related to DNA replication

```
##### Drosophila melanogaster
# string.db.7227 <- STRINGdb$new(version='10', species = 7227)
# string.db.7227.graph <- string.db.7227$get_graph()
# K.7227 <- net.kernel(string.db.7227.graph)
# saveRDS(K.7227, 'K7227.rds')
```

```

K.7227 <- readRDS("K7227.rds")
dim(K.7227)

[1] 13113 13113

rownames(K.7227) <- sub("[:digit:]]+.", "", rownames(K.7227))
colnames(K.7227) <- sub("[:digit:]]+.", "", colnames(K.7227))
# load target class (Proteasome)
kegg.dme <- getGeneKEGGLinks(species.KEGG='dme')
index <- match(kegg.dme[,2], 'path:dme03050')
path.03050 <- 1:length(index)
index2 <- path.03050[index]
path.03050 <- path.03050[is.na(index2)==FALSE]
path.03050 <- kegg.dme[path.03050,1]
path.03050 <- sub('.*\_\_', '', path.03050)
head(path.03050)

[1] "CG10149" "CG10230" "CG10370" "CG10938" "CG1100" "CG11552"

library(org.Dm.eg.db)
path2.03050 <- select(org.Dm.eg.db, keys=path.03050,
                      "FLYBASEPROT", keytype = 'ALIAS')[,2]
dme03050.infer <- net.infer(path2.03050, K.7227, top=100)
gene.id <- data.frame(dme03050.infer$top)[,1]
head(gene.id)

[1] FBpp0304246 FBpp0073558 FBpp0083036 FBpp0085535 FBpp0111932 FBpp0078317
100 Levels: FBpp0070118 FBpp0070654 FBpp0070907 FBpp0071222 ... FBpp0305676

rm(K.7227)
gene.id2 <- as.vector(na.omit(select(org.Dm.eg.db,
                                     keys=as.character(gene.id), "ENTREZID",
                                     keytype = 'FLYBASEPROT')[,2]))

# functional enrichment
params <- new("GOHyperGParams", geneIds=gene.id2, annotation="org.Dm.eg.db",
             ontology="BP", pvalueCutoff=0.001, conditional=FALSE,
             testDirection="over")

(hgOver <- hyperGTest(params))

Gene to GO BP test for over-representation
1199 GO BP ids tested (170 have p < 0.001)
Selected gene set size: 90
Gene universe size: 12699
Annotation package: org.Dm.eg

```

```
ORA.plot(hgOver, plot = FALSE)
```

	G0ID	GeneRatio	adjPvalue	Pvalue	OddsRatio	ExpCount	Count
1	G0:0070647	0.14074074	3.188680e-37	2.659449e-40	38.98591	1.913536	38
2	G0:0044257	0.14545455	1.741093e-31	4.356363e-34	36.45194	1.559178	32
3	G0:0051603	0.14545455	1.741093e-31	4.356363e-34	36.45194	1.559178	32
4	G0:0006508	0.06064516	5.122698e-31	1.708990e-33	17.83820	5.492558	47
5	G0:0030163	0.13733906	7.049892e-31	2.963471e-33	34.05867	1.651311	32
6	G0:0044265	0.11846690	7.049892e-31	3.527886e-33	29.65161	2.034018	34
7	G0:0006511	0.14485981	1.088221e-30	6.353251e-33	35.67713	1.516655	31
8	G0:0019941	0.14351852	1.287204e-30	8.588517e-33	35.28575	1.530829	31
9	G0:0043632	0.14220183	1.541968e-30	1.157441e-32	34.90275	1.545004	31
10	G0:0009057	0.10240964	6.536568e-29	5.451683e-31	25.08233	2.352941	34
	Size	Term					
1	270	protein modification by small protein conjugation or removal					
2	220	cellular protein catabolic process					
3	220	proteolysis involved in cellular protein catabolic process					
4	775	proteolysis					
5	233	protein catabolic process					
6	287	cellular macromolecule catabolic process					
7	214	ubiquitin-dependent protein catabolic process					
8	216	modification-dependent protein catabolic process					
9	218	modification-dependent macromolecule catabolic process					
10	332	macromolecule catabolic process					


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

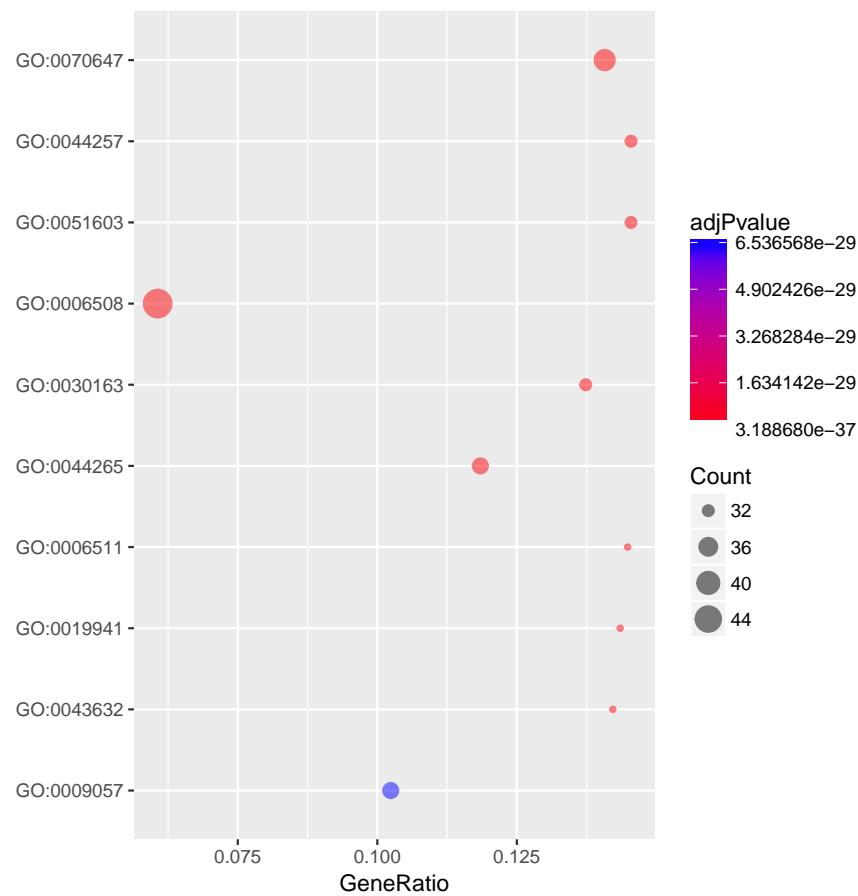


Figure 9: Top 10 biological functions related to Proteasome

```
##### Arabidopsis thaliana
# string.db.3702 <- STRINGdb$new(version='10', species = 3702)
# string.db.3702.graph <- string.db.3702$get_graph()
# K.3702 <- net.kernel(string.db.3702.graph)
# saveRDS(K.3702, 'K3702.rds')
K.3702 <- readRDS("K3702.rds")
dim(K.3702)

[1] 24283 24283

rownames(K.3702) <- sub("[:digit:]+.", "", rownames(K.3702))
colnames(K.3702) <- sub("[:digit:]+.", "", colnames(K.3702))
rownames(K.3702) <- gsub("\\.*", "", rownames(K.3702))
colnames(K.3702) <- gsub("\\.*", "", colnames(K.3702))
# load target class (Photosynthesis)
kegg.ath <- getGeneKEGGLinks(species.KEGG='ath')
index <- match(kegg.ath[,2], 'path:ath00195')
path.00195 <- 1:length(index)
```

```

index2 <- path.00195[index]
path.00195 <- path.00195[is.na(index2)==FALSE]
path.00195 <- kegg.ath[path.00195,1]
path.00195 <- sub('.*\\_',' ',path.00195)
head(path.00195)

[1] "AT1G03130" "AT1G03600" "AT1G06680" "AT1G08380" "AT1G10960" "AT1G14150"

ath00195.infer <- net.infer(path.00195, K.3702, top=100)
gene.id <- data.frame(ath00195.infer$top)[,1]
head(gene.id)

[1] AT2G40100 AT2G34420 AT1G29930 AT2G34430 ATCG00270 AT4G10340
100 Levels: AT1G09310 AT1G10360 AT1G15820 AT1G18730 AT1G21500 ... ATCG01060

rm(K.3702)
# functional enrichment
params <- new("GOHyperGParams", geneIds=gene.id, annotation="org.At.tair.db",
              ontology="BP",pvalueCutoff=0.001,conditional=FALSE,
              testDirection="over")
(hgOver <- hyperGTest(params))

Gene to GO BP test for over-representation
566 GO BP ids tested (166 have p < 0.001)
Selected gene set size: 96
  Gene universe size: 20904
  Annotation package: org.At.tair

ORA.plot(hgOver, plot = FALSE)

      GOID  GeneRatio    adjPvalue      Pvalue OddsRatio  ExpCount  Count
1  GO:0015979 0.13126492 6.953915e-66 1.228607e-68   75.34307   1.9242250    55
2  GO:0019684 0.12933754 5.830976e-47 2.060416e-49   55.45534   1.4557979    41
3  GO:0006091 0.06829268 1.590856e-36 8.432097e-39   27.46655   2.8243398    42
4  GO:0010207 0.11864407 5.773037e-22 4.079885e-24   37.06769   0.8128588    21
5  GO:0009637 0.14062500 4.410735e-20 3.896409e-22   43.42238   0.5878301    18
6  GO:0006364 0.08606557 3.424191e-19 3.885311e-21   25.84664   1.1205511    21
7  GO:0016072 0.08571429 3.424191e-19 4.234865e-21   25.73000   1.1251435    21
8  GO:0010218 0.16161616 6.210523e-19 8.778125e-21   49.93976   0.4546498    16
9  GO:0010114 0.15384615 1.272641e-18 2.023634e-20   47.09091   0.4776119    16
10 GO:0034470 0.07526882 3.643896e-18 6.437979e-20   22.30233   1.2812859    21

      Size      Term
1    419      photosynthesis

```

2	317	photosynthesis, light reaction
3	615	generation of precursor metabolites and energy
4	177	photosystem II assembly
5	128	response to blue light
6	244	rRNA processing
7	245	rRNA metabolic process
8	99	response to far red light
9	104	response to red light
10	279	ncRNA processing

ORA.plot(hgOver)

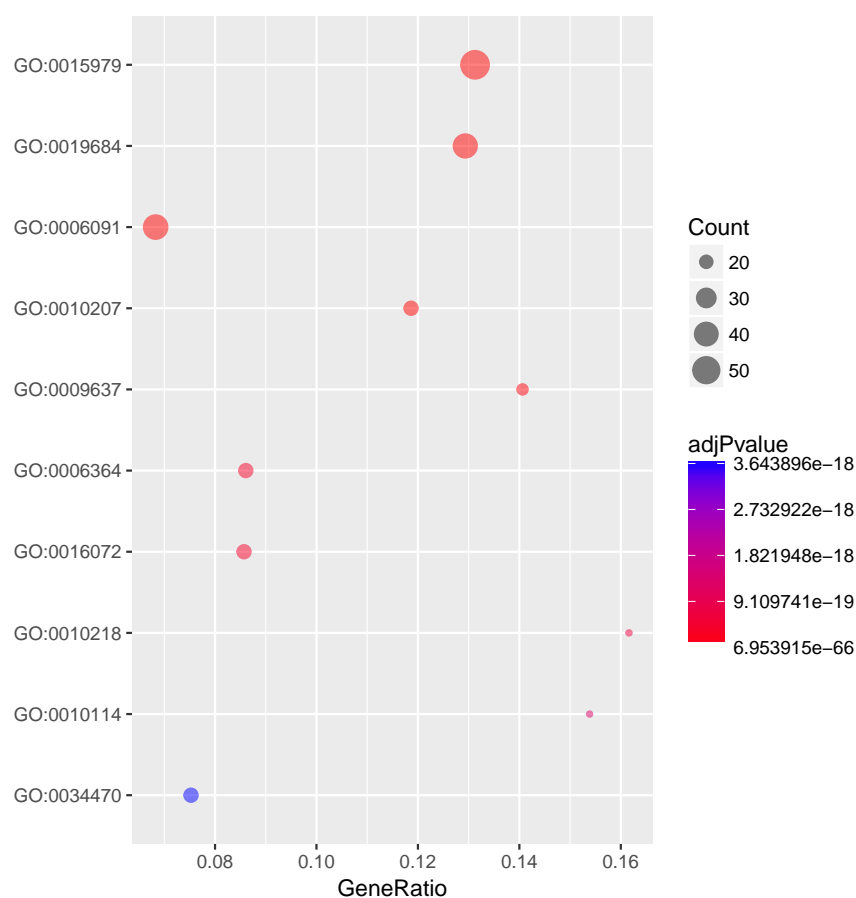


Figure 10: Top 10 biological functions related to Photosynthesis

```
##### Zebra fish
# string.db.7955 <- STRINGdb$new(version='10', species = 7955)
# string.db.7955.graph <- string.db.7955$get_graph()
# K.7955 <- net.kernel(string.db.7955.graph)
# saveRDS(K.7955, 'K7955.rds')
K.7955 <- readRDS("K7955.rds")
dim(K.7955)
```

```
[1] 23369 23369
```

```
rownames(K.7955) <- sub("[:digit:]]+.", "", rownames(K.7955))
colnames(K.7955) <- sub("[:digit:]]+.", "", colnames(K.7955))
# load target class (ErbB signaling pathway)
kegg.dre <- getGeneKEGGLinks(species.KEGG='dre')
index <- match(kegg.dre[,2], 'path:dre04012')
path.04012 <- 1:length(index)
index2 <- path.04012[index]
path.04012 <- path.04012[is.na(index2)==FALSE]
path.04012 <- kegg.dre[path.04012,1]
path.04012 <- sub('.*\_\_', '', path.04012)
head(path.04012)
```

```
[1] "100000252" "100000720" "100002225" "100002263" "100003514" "100006675"
```

```
library(org.Dr.eg.db)
path2.04012 <- select(org.Dr.eg.db, path.04012, c("ENSEMBLPROT"))[,2]
dre04012.infer <- net.infer(path2.04012, K.7955, top=100)
gene.id <- data.frame(dre04012.infer$top)[,1]
head(gene.id)
```

```
[1] ENSDARP00000109319 ENSDARP00000123439 ENSDARP00000120120 ENSDARP00000111479
```

```
[5] ENSDARP00000056821 ENSDARP00000106632
```

```
100 Levels: ENSDARP00000007188 ENSDARP00000007758 ... ENSDARP00000126522
```

```
rm(K.7955)
gene.id2 <- as.vector(na.omit(select(org.Dr.eg.db,
                                     keys=as.character(gene.id), "ENTREZID",
                                     keytype = 'ENSEMBLPROT')[,2])))
```

```
# functional enrichment
```

```
params <- new("GOHyperGParams", geneIds=gene.id2, annotation="org.Dr.eg.db",
              ontology="BP", pvalueCutoff=0.001, conditional=FALSE,
              testDirection="over")
```

```
(hgOver <- hyperGTest(params))
```

```
Gene to GO BP test for over-representation
```

```
451 GO BP ids tested (81 have p < 0.001)
```

```
Selected gene set size: 45
```

```
Gene universe size: 16411
```

```
Annotation package: org.Dr.eg
```

```
ORA.plot(hgOver, plot = FALSE)
```

	GOID	GeneRatio	adjPvalue	Pvalue	OddsRatio	ExpCount	Count
1	GO:0007165	0.011666168	9.296688e-19	2.061350e-21	25.69703	9.1667174	39
2	GO:0044700	0.010964296	3.566136e-18	2.134513e-20	23.73849	9.7535190	39
3	GO:0023052	0.010933558	3.566136e-18	2.372153e-20	23.65278	9.7809396	39
4	GO:0007154	0.010800332	4.240007e-18	3.760539e-20	23.28135	9.9015904	39
5	GO:0046578	0.112000000	1.292028e-17	1.432404e-19	66.13484	0.3427579	14
6	GO:0035556	0.023360288	1.639647e-17	2.181348e-19	19.23469	3.0519164	26
7	GO:0051716	0.010174798	2.269763e-17	3.522914e-19	21.53875	10.5103284	39
8	GO:0051056	0.101449275	3.393824e-17	6.020087e-19	59.15401	0.3784047	14
9	GO:0050896	0.008843831	5.662341e-17	1.129957e-18	26.25740	12.7122052	41
10	GO:0007265	0.092715232	9.940884e-17	2.204187e-18	53.49800	0.4140516	14
	Size		Term				
1	3343		signal transduction				
2	3557		single organism signaling				
3	3567		signaling				
4	3611		cell communication				
5	125		regulation of Ras protein signal transduction				
6	1113		intracellular signal transduction				
7	3833		cellular response to stimulus				
8	138		regulation of small GTPase mediated signal transduction				
9	4636		response to stimulus				
10	151		Ras protein signal transduction				

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

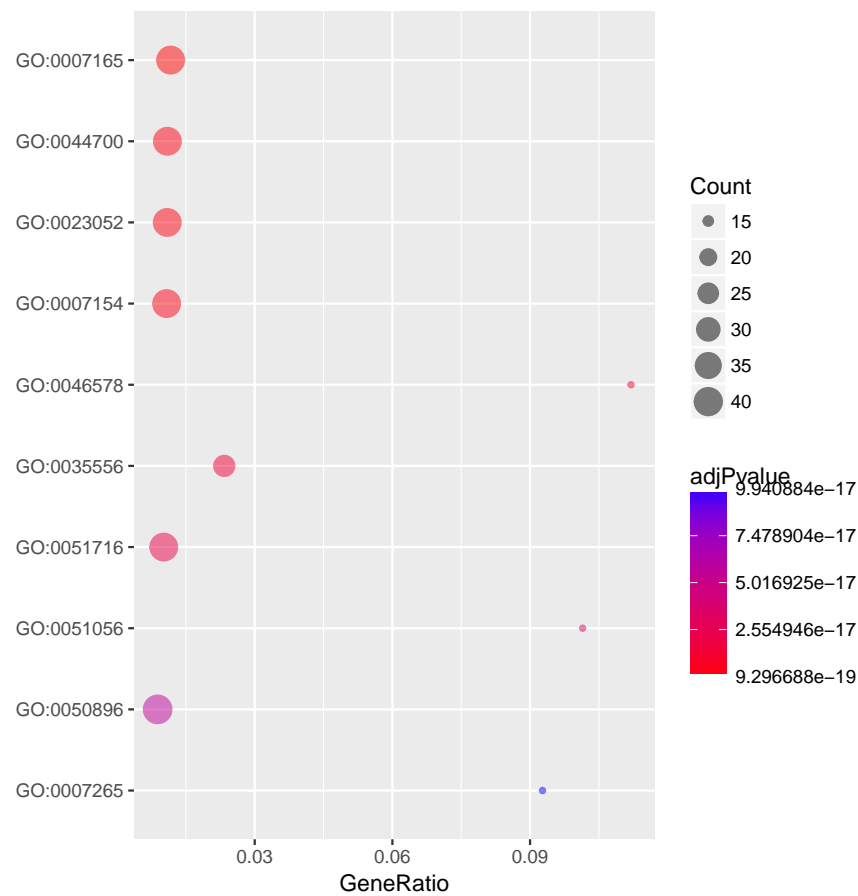


Figure 11: Top 10 biological functions related to ErbB signaling pathway

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.Dr.eg.db_3.4.0  org.At.tair.db_3.4.0 org.Dm.eg.db_3.4.0
```

[4] org.Ce.eg.db_3.3.0	org.Sc.sgd.db_3.4.0	org.Mm.eg.db_3.3.0
[7] limma_3.28.21	KEGG.db_3.2.3	G0.db_3.3.0
[10] org.Hs.eg.db_3.3.0	G0stats_2.38.1	Category_2.38.0
[13] Matrix_1.2-6	AnnotationDbi_1.36.2	IRanges_2.8.2
[16] S4Vectors_0.12.2	Biobase_2.32.0	ggplot2_2.2.0
[19] httr_1.2.1	PPInfer_1.1.2	yeastExpData_0.20.0
[22] graph_1.50.0	BiocGenerics_0.20.0	STRINGdb_1.12.0
[25] kernlab_0.9-24	igraph_1.0.1	biomaRt_2.28.0

loaded via a namespace (and not attached):

[1] Rcpp_0.12.6	lattice_0.20-33	png_0.1-7
[4] gtools_3.5.0	assertthat_0.1	digest_0.6.10
[7] R6_2.1.3	plyr_1.8.4	chron_2.3-47
[10] RSQLite_1.0.0	sqldf_0.4-10	gplots_3.0.1
[13] lazyeval_0.2.0	annotate_1.50.0	gdata_2.17.0
[16] hash_2.2.6	gsubfn_0.6-6	proto_0.3-10
[19] labeling_0.3	splines_3.3.1	RCurl_1.95-4.8
[22] munsell_0.4.3	tibble_1.2	XML_3.98-1.4
[25] AnnotationForge_1.14.2	bitops_1.0-6	grid_3.3.1
[28] RBGL_1.48.1	xtable_1.8-2	GSEABase_1.34.1
[31] gtable_0.2.0	DBI_0.5-1	magrittr_1.5
[34] scales_0.4.1	KernSmooth_2.23-15	genefilter_1.54.2
[37] RColorBrewer_1.1-2	tools_3.3.1	plotrix_3.6-3
[40] survival_2.40-1	colorspace_1.2-6	caTools_1.17.1

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.
- Zhu, X. (2006). Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*. 2(3), 4.