KEGGgraph: a graph approach to KEGG PATHWAY in R and Bioconductor

Jitao David Zhang and Stefan Wiemann

October 29, 2019

Abstract

We demonstrate the capabilities of the *KEGGgraph* package, an interface between KEGG pathways and graph model in R as well as a collection of tools for these graphs. Superior to preceding approaches, *KEGGgraph* maintains the pathway topology and allows further analysis or dissection of pathway graphs. It parses the regularly updated KGML (KEGG XML) files into graph models maintaining all essential pathway attributes.

1 Introduction

Since its first introduction in 1995, KEGG PATHWAY has been widely used as a reference knowledge base for understanding biological pathways and functions of cellular processes. The knowledge from KEGG has proven of great value by numerous work in a wide range of fields [Kanehisa *et al.*, 2008].

Pathways are stored and presented as graphs on the KEGG server side, where nodes are molecules (protein, compound, etc) and edges represent relation types between the nodes, e.g. activation or phosphorylation. The graph nature raised our interest to investigate them with powerful graph tools implemented in R and Bioconductor [Gentleman *et al.*, 2004], including *graph*, *RBGL* and *Rgraphviz* [Carey *et al.*, 2005]. While it is barely possible to query the graph characteristics by manual parsing, a native and straightforward client-side tool is currently missing to parse pathways and analyze them consequently with tools for graph in R.

To address this problem, we developed the open-source software package *KEGGgraph*, an interface between KEGG pathway and graph object as well as a collection of tools to analyze, dissect and visualize these graphs.

The package requires KGML (KEGG XML) files, which can be downloaded from KEGG FTP site (`ftp://ftp.genome.jp/pub/kegg/xml`) without license permission for academic purposes. To demonstrate the functionality, in 'extdata/' sub-directory of *KEGGgraph* we have pre-installed several KGML files.

2 Software features

KEGGgraph offers the following functionalities:

Parsing: It should be noted that one 'node' in KEGG pathway does not necessarily map to merely one gene product, for example the node 'ERK' in the human TGF-Beta signaling pathway contains two homologues, MAPK1 and MAPK3. Therefore, among several parsing options, user can set whether to expand these nodes topologically. Beyond facilitating the interpretation of pathways in a gene-oriented manner, the approach also entitles unique identifiers to nodes, enabling merging graphs from different pathways.

Graph operations: Two common operations on graphs are subset and merge. A subgraph of selected nodes and the edges in between are returned when subsetting, while merging produces a new graph that contains nodes and edges of individual ones. Both are implemented in *KEGGgraph*.

Visualization: *KEGGgraph* provides functions to visualize KEGG graphs with custom style. Nevertheless users are not restricted by them, alternatively they are free to render the graph with other tools like the ones in *Rgraphviz*.

Besides the functionalities described above, *KEGGgraph* also has tools for remote KGML file retrieval, graph feature study and other related tasks. We will demonstrate them later in this vignette.

3 Case studies

We load the *KEGGgraph* by typing or pasting the following codes in R command line:

```
> library(KEGGgraph)
```

3.1 Get KGML files

There are at least two possibilities to get KGML (KEGG XML) files:

- Manual download from KEGG FTP site at `ftp://ftp.genome.jp/pub/kegg/xml/`.
- Automatic retrieval from KEGG FTP site with the function `retrieveKGML`.

To retrieve KGML file automatically from KEGG FTP site, one has to know the pathway identifier (in the form of [a-z]3[0-9]5, where the three-alphabet code represent the organism and the five digits represent pathway). One method to find the mapping between pathway name and identifier is use `KEGGPATHNAME2ID` environment in *KEGG.db*. For example, the following codes retrieve p53 signaling pathway of *C.elegans* from KEGG FTP site.

```
> library(KEGG.db)
> tmp <- tempfile()
```

```
> pName <- "p53 signaling pathway"
> pId <- mget(pName, KEGGPATHNAME2ID)[[1]]
> retrieveKGML(pId, organism="cel", destfile=tmp, method="wget", quiet=TRUE)
```

Note: `retrieveKGML` uses a *try-download* mechanism (since the *KEGGgraph version 1.1.2*) to retrieve the KGML file from remote KEGG FTP server. Since August 2009 the KGML files are stored separately in different subdirectories depending on whether they record metabolic or non-metabolic pathways. Since from the KEGG pathway accession ID alone (e.g. `hsa00010`) alone it is not possible to determine its content, the `retrieveKGML` first tries to download the file from the non-metabolic subdirectory, and tries the metabolic directory in case no file was found in the non-metabolic category (the *try* step). In case the corresponding file is found, it is downloaded (the *download* step). Even if the file is found in the first try round, it still needs to be downloaded in the *download* step. However, this does not actually need to the network overhead, since thanks to the common cache system the file is only downloaded once.

3.2 Parsing and graph feature query

First we read in KGML file for human MAPK signaling pathway (with KEGG ID hsa04010):

```
> mapkKGML <- system.file("extdata/hsa04010.xml",
+                         package="KEGGgraph")
```

Once the file is ready, we can either parse them into an object of *KEGGPathway* or an object of *graph*. *KEGGPathway* object maintains the information of the pathway (title, link, organism, etc), while *graph* objects are more natural approach and can be directly plugged in many other tools. To convert the pathway into graph, we use

```
> mapkG <- parseKGML2Graph(mapkKGML,expandGenes=TRUE)
> mapkG

A graphNEL graph with directed edges
Number of Nodes = 265
Number of Edges = 876
```

Alternatively we can parse the KGML file first into an object of *KEGGpathway*, which can be later converted into the graph object, as the following lines show:

```
> mapkpathway <- parseKGML(mapkKGML)
> mapkpathway

KEGG Pathway
[ Title ]: MAPK signaling pathway
```

```
[ Name ]: path:hsa04010
[ Organism ]: hsa
[ Number ] :04010
[ Image ] :http://www.genome.jp/kegg/pathway/hsa/hsa04010.gif
[ Link ] :http://www.genome.jp/dbget-bin/show_pathway?hsa04010
------------------------------------------------------------
Statistics:
        136 node(s)
        171 edge(s)
        0 reaction(s)
------------------------------------------------------------
```

```
> mapkG2 <- KEGGpathway2Graph(mapkpathway, expandGenes=TRUE)
> mapkG2

A graphNEL graph with directed edges
Number of Nodes = 265
Number of Edges = 876
```

There is no difference between graph objects derived from two approaches.

The option 'expandGenes' in parsing controls whether the nodes of paralogues in pathways should be expanded or not. Since one 'node' in KEGG pathway does not necessarily map to only one gene/gene product (e.g. 'ERK' maps to MAPK1 and MAPK3), the option allows expanding these nodes and takes care of copying existing edges.

Another option users may find useful is 'genesOnly', when set to TRUE, the nodes of other types than 'gene' (compounds, for example) are neglected and the result graph consists only gene products. This is especially desired when we want to query network characteristics of gene products. Its value is set to 'TRUE' by default.

The following commands extract node and edge information:

```
> mapkNodes <- nodes(mapkG)
> nodes(mapkG)[1:3]

[1] "hsa:5923"  "hsa:5924"  "hsa:11072"

> mapkEdges <- edges(mapkG)
> edges(mapkG)[1]

$`hsa:5923`
[1] "hsa:22800" "hsa:22808" "hsa:3265"  "hsa:3845"  "hsa:4893"  "hsa:6237"
```

Edges in KEGG pathways are directional, that is, an edge starting at node A pointing to node B does not guarantee a reverse relation, although reciprocal edges are also allowed.

When listing edges, a list indexed with node names is returned. Each item in the list records the nodes pointed to.

We can also extract the node attributes specified by KEGG with `getKEGGnodeData`:

```
> mapkGnodedata <- getKEGGnodeData(mapkG)
> mapkGnodedata[[2]]

KEGG Node (Entry 'hsa:5924'):
------------------------------------------------------------
[ displayName ]: RASGRF1, GRF1...
[ Name ]: hsa:5924
[ Type ]: gene
[ Link ]: http://www.genome.jp/dbget-bin/www_bget?hsa+5923+5924
------------------------------------------------------------
```

An alternative to use `gettKEGGnodeData` is

```
> getKEGGnodeData(mapkG, 'hsa:5924')
```

, returning identical results.

Similarly the `getKEGGedgeData` is able to extract edge information:

```
> mapkGedgedata <- getKEGGedgeData(mapkG)
> mapkGedgedata[[4]]

  KEGG Edge (Type: PPrel):
------------------------------------------------------------
[ Entry 1 ID ]: hsa:5923
[ Entry 2 ID ]: hsa:3845
[ Subtype ]:
  [ Subtype name ]: activation
  [ Subtype value ]: -->
------------------------------------------------------------
```

Alternatively the query above can be written as:

```
> getKEGGedgeData(mapkG,'hsa:627~hsa:4915')
```

For `KEGGNode` and `KEGGEdge` objects, methods are implemented to fetch their attributes, for example `getName`, `getType` and `getDisplayName`. Guides to use these methods as well as examples can be found in help pages.

This case study finishes with querying the degree attributes of the nodes in the graph. We ask the question which nodes have the highest out- or in-degrees. Roughly speaking the out-degree (number of out-going edges) reflects the regulatory role, while the in-degree (number of in-going edges) suggests the subjectability of the protein to intermolecular regulations.

```
> mapkGoutdegrees <- sapply(edges(mapkG), length)
> mapkGindegrees <- sapply(inEdges(mapkG), length)
> topouts <- sort(mapkGoutdegrees, decreasing=T)
> topins <- sort(mapkGindegrees, decreasing=T)
> topouts[1:3]

hsa:5594 hsa:5595 hsa:1432
      26       26       13

> topins[1:3]

 hsa:5923  hsa:5924 hsa:10125
       26        26        26
```

3.3 Graph subset and merge

We demonstrate the subsetting of the graph with 25 randomly chosen nodes of MAPK pathway graph:

```
> library(Rgraphviz)
> set.seed(123)
> randomNodes <- sample(nodes(mapkG), 25)
> mapkGsub <- subGraph(randomNodes, mapkG)
> mapkGsub

A graphNEL graph with directed edges
Number of Nodes = 25
Number of Edges = 6
```

The subgraph is visualized in figure 1, where nodes with in-degree or out-degree in red and others in grey.[1]. And in the example we also demonstrate how to convert KEGG ID into other other identifiers via the Entrez GeneID. More details on the conversion of IDs can be found on page 11.

Another common operation on graphs is merging, that is, combining different graphs together. It is inspired by the fact that many KEGG pathways embed other pathway, for example MAPK signaling pathway embeds 6 pathways including Wnt signaling pathway. `mergeGraphs` provides the possibility to merge them into one graph for further analysis. Next we merge MAPK and Wnt signaling pathway into one graph. The graphs to be merged should be organized into a list, and it is commandary to use 'expandGenes=TRUE' option when parsing to make sure the nodes are unique and indexed by KEGGID.

[1]The `makeAttr` function is used to assign nodes with rendering attributes, whose code can be found in the Rnw file.

```
> outs <- sapply(edges(mapkGsub), length) > 0
> ins <- sapply(inEdges(mapkGsub), length) > 0
> ios <- outs | ins
> ## translate the KEGG IDs into Gene Symbol
> if(require(org.Hs.eg.db)) {
+   ioGeneID <- translateKEGGID2GeneID(names(ios))
+   nodesNames <- sapply(mget(ioGeneID, org.Hs.egSYMBOL, ifnotfound=NA), "[[",1)
+ } else {
+   nodesNames <- names(ios)
+ }
> names(nodesNames) <- names(ios)
> nAttrs <- list();
> nAttrs$fillcolor <- makeAttr(mapkGsub, "lightgrey", list(orange=names(ios)[ios]))
> nAttrs$label <- nodesNames
> plot(mapkGsub, "neato", nodeAttrs=nAttrs,
+      attrs=list(node=list(fillcolor="lightgreen",
+                   width="0.75", shape="ellipse"),
+        edge=list(arrowsize="0.7")))
```

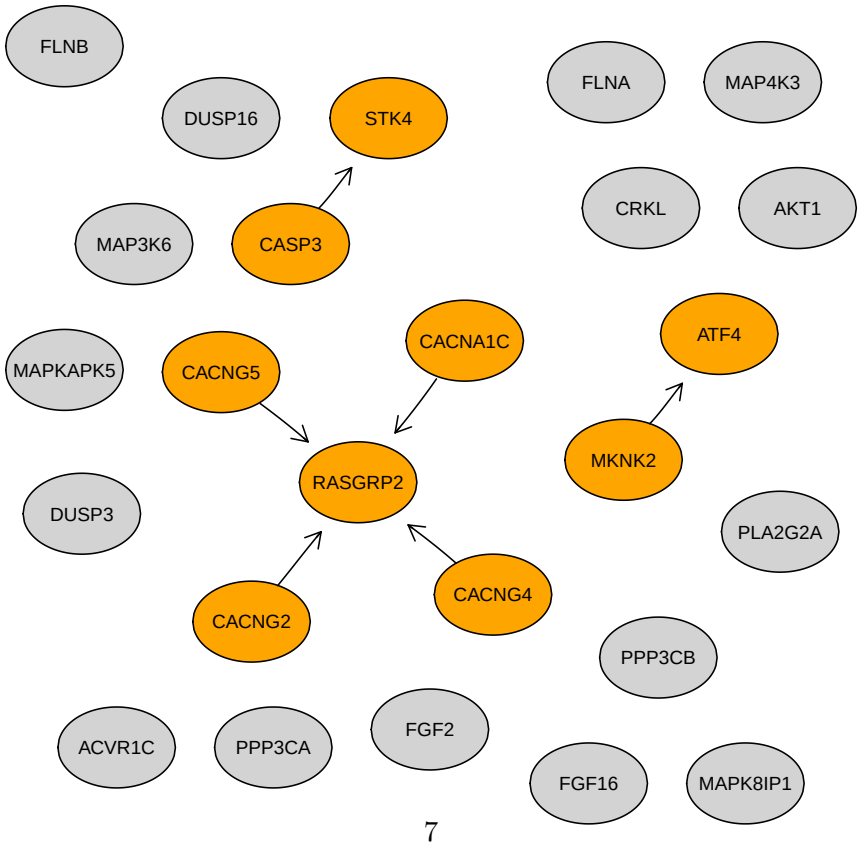

Figure 1: A random subgraph of MAPK signaling pathway

```
> wntKGML <- system.file("extdata/hsa04310.xml",package="KEGGgraph")
> wntG <- parseKGML2Graph(wntKGML)
> graphs <- list(mapk=mapkG, wnt=wntG)
> merged <- mergeGraphs(graphs)
> merged

A graphNEL graph with directed edges
Number of Nodes = 386
Number of Edges = 1628
```

We observe that the node number in the merged graph (386) is less than the sum of two graphs (265 and 148 for MAPK and Wnt pathway respectively), reflecting the crosstalk between the pathways by sharing nodes.

3.4 Using other graph tools

In R and Bioconductor there'are powerful tools for graph algorithms and operations, including *graph*, *Rgraphviz* and *RBGL*. The KEGG graphs can be analyzed with their functionalities to describe characterisitcs of the pathway and to answer biological relevant questions.

Here we demonstrate the use of other graph tools with asking the question which nodes are of the highest importance in MAPK signalling pathway. To this end we turn to relative betweenness centrality [Aittokallio and Schwikowski, 2006, Carey *et al.*, 2005]. Betweenness is a centrality measure of a node within a graph. Nodes that occur on many shortest paths between other vertices have higher betweenness than those that do not. It is scaled by the factor of (n-1)(n-2)/2 to get relative betweenness centrality, where n is the number of nodes in the graph. Both measurements estimate the importance or the role of the node in the graph.

With the function implemented in *RBGL*, our aim is to identify most important nodes in MAPK signalling pathway.

Note: the current version of `RBGL` (version 1.59.5) reports the error that `BGL_brandes_betweeness_central` not available for `.Call()` for package "RBGL". Therefore the execution has been suppressed for now.

```
> library(RBGL)
> bcc <- brandes.betweenness.centrality(mapkG)
> rbccs <- bcc$relative.betweenness.centrality.vertices[1L,]
> toprbccs <- sort(rbccs,decreasing=TRUE)[1:4]
> toprbccs
```

We identify the top 4 important nodes judged by betweenness centrality as MAP3K1 (hsa:4214), GRB2 (hsa:2885), MAP2K2 (hsa:5605) and MAP2K1 (hsa:5604) (the mapping between KEGG ID and gene symbol is done using *biomaRt*, see page 11). In figure 2 we illustrate them as well as their interacting partners in MAPK pathway.

4 Other funtionalities

Besides the ability to parse and operate on KEGG PATHWAY graph objects, the *KEGGgraph* package also provides functionalities to complement tasks related to deal with KEGG pathways. We introduce some of them here, for a full list of functions please see the package help file:

```
> help(package=KEGGgraph)
```

4.1 Parsing chemical compound reaction network

KEGG PATHWAY captures two kinds of network:the protein network and the chemical network. The protein network consists *relations* (*edges*) between gene products, while the chemical network illustrate the *reactions* between chemical compounds. Since the metabolic pathway can be viewed both as a network of proteins (enzymes) and as a network of chemical compounds, metabolic pathways can be viewed as both protein networks and chemical networks, whereas regulatory pathways are always viewed as protein networks only.`KEGGPathway` provides methods to access this network.

We show the example of Glycine, serine and threonine metabolism pathway.

```
> mapfile <-  system.file("extdata/map00260.xml",package="KEGGgraph")
> map <- parseKGML(mapfile)
> map

KEGG Pathway
[ Title ]: Glycine, serine and threonine metabolism
[ Name ]: path:map00260
[ Organism ]: map
[ Number ] :00260
[ Image ] :http://www.genome.jp/kegg/pathway/map/map00260.gif
[ Link ] :http://www.genome.jp/dbget-bin/show_pathway?map00260
------------------------------------------------------------
Statistics:
        144 node(s)
        371 edge(s)
        68 reaction(s)
------------------------------------------------------------

> reactions <- getReactions(map)
> reactions[[1]]

KEGG Reaction(rn:R08211)
------------------------------------------------------------
```

```
[ Name ]: rn:R08211
[ Type ]: irreversible
[ Substrate Name ]: cpd:C00576
[ Product Name ]: cpd:C00719
```

Figure 3 shows how to extract reactions from the pathway and to build a directed graph with them.

4.2 Expand embedded pathways

Function `parseKGMLexpandMaps` is a function to handle with pathways embedding other pathways. For example, pancreatic cancer pathway embeds 9 other pathways including MAPK and ErbB signaling pathway, cell cycle and apoptosis pathway, etc. To parse them into one graph, the users only have to download the KGML file and feed the file name to `parseKGMLexpandMaps`, the function parses the file, analyze the embedded pathways, download their files from KEGG FTP site automatically (alternatively a local repository can be specified for KGML files) and merge the individual pathways into a single graph. For example, the following single line parses MAPK signaling pathway with all its embedded pathways

```
> mapkGembed <- parseKGMLexpandMaps(mapkKGML)
```

As its name suggests, function `subGraphByNodeType` subsets the graph by node type, the nodes to subset are those of the type given by the user. It is useful when the KGML file was parsed with 'genesOnly=FALSE' option and later on the user wants only certain kind of node, 'gene' for example, remained. The following example shows how to use it.

```
> mapkGall <- parseKGML2Graph(mapkKGML,genesOnly=FALSE)
> mapkGall

A graphNEL graph with directed edges
Number of Nodes = 277
Number of Edges = 891

> mapkGsub <- subGraphByNodeType(mapkGall, "gene")
> mapkGsub

A graphNEL graph with directed edges
Number of Nodes = 265
Number of Edges = 876
```

4.3 Annotation

`translateKEGGID2GeneID` translates KEGG identifiers (KEGGID) into Entrez GeneID. For example, if we want to find the Entrez GeneID of the nodes in MAPK pathway having the highest relative betweenness centrality, the following codes do the job.

```
> toprbccKEGGID <- names(toprbccs)
> toprbccKEGGID
> toprbccGeneID <- translateKEGGID2GeneID(toprbccKEGGID)
> toprbccGeneID
```

To convert GeneID to other identifiers, we recommend genome wide annotation packages, for human it is *org.Hs.eg.db* and the packages for other organisms can be fount at `http://www.bioconductor.org/packages/release/data/annotation/`. To demonstrate its use, we draw the sub-network in the figure 2 again, whereas nodes are now labeled with gene symbols.

```
> if(require(org.Hs.eg.db)) {
+   tnodes <- nodes(toprSub)
+   tgeneids <- translateKEGGID2GeneID(tnodes)
+   tgenesymbols <- sapply(mget(tgeneids, org.Hs.egSYMBOL, ifnotfound=NA), "[[",1)
+   toprSubSymbol <- toprSub
+   nodes(toprSubSymbol) <- tgenesymbols
+   plot(toprSubSymbol, "neato",attrs=list(node=list(font=5, fillcolor="lightblue")))
+ }
```

Alternatively, users could use R package *biomaRt* [Durinck *et al.*, 2005, Durinck and Huber, 2008] for ID conversion, whereas it assumes that the user has an internet connection. The following example shows how to translate the node hits we acquired in the example above into HGNC symbols:

```
> library(biomaRt)
> hsapiens <- useMart("ensembl","hsapiens_gene_ensembl" )
> filters <- listFilters(hsapiens)
> getBM(attributes=c("entrezgene","hgnc_symbol"),
+       filters="entrezgene",
+       values=toprbccGeneID, mart=hsapiens)
```

5 Acknowledgement

We thank Vincent Carey, Holger Fröhlich and Wolfgang Huber for comments and suggestions on the package, and the reviewers from the Bioconductor community.

Many users have provided very helpful feedback. Here is an incomplete list of them: Martin Morgan, Paul Shannon, Juliane Manitz and Alexander Gulliver Bjørnholt Grønning.

6 Conclusion

Before the release of *KEGGgraph*, several R/Bioconductor packages have been introduced and proven their usefulness in understanding biological pathways with KEGG. However, *KEGGgraph* is the first package able to parse any KEGG pathways from KGML files into graphs. In comparison, existing tools can not achieve the results we present here. They either neglects the graph topology (*KEGG.db*), do not parse pathway networks (*keggorth*), or are specialized for certain pathways (*cMAP* and *pathRender*).

With *KEGGgraph*, we contribute a direct and natural approach to KEGG pathways, and the possibilities to study them in R and Bioconductor.

7 Session Info

The script runs within the following session:

```
R version 3.6.1 (2019-07-05)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server 2012 R2 x64 (build 9600)

Matrix products: default

locale:
[1] LC_COLLATE=C
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

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

other attached packages:
[1] org.Hs.eg.db_3.10.0  AnnotationDbi_1.48.0 IRanges_2.20.0
[4] S4Vectors_0.24.0     Biobase_2.46.0       Rgraphviz_2.30.0
[7] graph_1.64.0         BiocGenerics_0.32.0  KEGGgraph_1.46.0
```

```
loaded via a namespace (and not attached):
 [1] Rcpp_1.0.2      bit_1.1-14      rlang_0.4.1     blob_1.2.0
 [5] tools_3.6.1     DBI_1.0.0       digest_0.6.22   bit64_0.9-7
 [9] tibble_2.1.3    crayon_1.3.4    vctrs_0.2.0     bitops_1.0-6
[13] RCurl_1.95-4.12 zeallot_0.1.0   memoise_1.1.0   RSQLite_2.1.2
[17] pillar_1.4.2    compiler_3.6.1  backports_1.1.5 XML_3.98-1.20
[21] pkgconfig_2.0.3
```

References

[Gentleman *et al.*, 2004] Gentleman *et al.* (2004) Bioconductor: open software development for computational biology and bioinformatics, *Genome Biology*, **5**, R80.

[Carey *et al.*, 2005] Carey *et al.* (2005) Network structures and algorithms in Bioconductor, *Bioinformatics*, **21**, 135-136.

[Kanehisa *et al.*, 2008] Kanehisa *et al.* (2008) KEGG for linking genomes to life and the environment, *Nucleic Acids Research, Database issue*, **36**, 480-484.

[Klukas and Schreiber, 2007] Klukas and Schreiber. (2007) Dynamic exploration and editing of KEGG pathway diagrams, *Bioinformatics*, **23**, 344-350.

[Aittokallio and Schwikowski, 2006] Aittokallio and Schwikowski (2006) Graph-based methods for analysing networks in cell biology, *Briefings in Bioinformatics*, **7**, 243-255.

[Durinck *et al.*, 2005] Durinck *et al.* (2005) BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis, *Bioinformatcs*, **21**, 3439-3440.

[Durinck and Huber, 2008] Durinck and Huber (2008) R/Bioconductor package *biomaRt*, 2008

```
> toprbccName <- names(toprbccs)
> toprin <- sapply(toprbccName, function(x) inEdges(mapkG)[x])
> toprout <- sapply(toprbccName, function(x) edges(mapkG)[x])
> toprSubnodes <- unique(unname(c(unlist(toprin), unlist(toprout), toprbccName)))
> toprSub <- subGraph(toprSubnodes, mapkG)
> nAttrs <- list()
> tops <- c("MAPK3K1","GRB2","MAP2K2","MAP2K1")
> topLabels <- lapply(toprbccName, function(x) x); names(topLabels) <- tops
> nAttrs$label <- makeAttr(toprSub, "", topLabels)
> nAttrs$fillcolor <- makeAttr(toprSub, "lightblue", list(orange=toprbccName))
> nAttrs$width <- makeAttr(toprSub,"",list("0.8"=toprbccName))
> plot(toprSub, "twopi", nodeAttrs=nAttrs, attrs=list(graph=list(start=2)))
```

Figure 2: Nodes with the highest relative betweenness centrality in MAPK pathway (in orange) and their interacting partners (in blue).

```
> chemicalGraph <- KEGGpathway2reactionGraph(map)
> outDegrees <- sapply(edges(chemicalGraph), length)
> maxout <- names(sort(outDegrees,decreasing=TRUE))[1:3]
> nAttrs <- list()
> maxoutlabel <- as.list(maxout); names(maxoutlabel) <- maxout
> nAttrs$label <- makeAttr(chemicalGraph, "", maxoutlabel)
> nAttrs$fillcolor <- makeAttr(chemicalGraph, "lightblue", list(orange=maxout))
> nAttrs$width <- makeAttr(chemicalGraph,"0.8", list("1.8"=maxout))
> plot(chemicalGraph, nodeAttrs=nAttrs)
```

Figure 3: Reaction network built of chemical compounds: the orange nodes are the three compounds with maximum out-degree in this network.