RamiGO: an R interface for AmiGO

Markus Schröder[1,2], Daniel Gusenleitner[1], Aedín Culhane[1], Benjamin Haibe-Kains[1], and John Quackenbush[1]

[1]*Biostatistics and Computational Biology, Dana-Farber Cancer Institute, Harvard School of Public Health, Boston, USA*
[2]*Computational Genomics, Center for Biotechnology, Bielefeld University, Germany*

June 8, 2011

Contents

1 Introduction

The *RamiGO* package is providing functions to interact with AmiGO and retrieving GO (Gene Ontology) trees in various formats. The most common requests would be as png or svg. RamiGO also provides a parser for the GraphViz DOT format that returns a graph object and meta data.

2 Getting started

The DOT format parser uses the `strapply` function from the *gsubfn* package to extract the information from the DOT format file. Therefor the *gsubfn* package has to be installed on your system, as well as the following packages:

```
> library(RamiGO)
> library(gsubfn)
> library(igraph)
> library(png)
> library(RCurl)
```

strapply enables perl-like regular expression in R, as do grep, sub or gsub. In particular, it enables the use of the perl variables $1, $2, ... for extracting information from within a regular expression. The code below shows an example of the use of strapply. The string within brackets (...) is returned in a list by strapply.

```
> strapply(c("node25 -> node30"), "node([\\d]+) -> node([\\d]+)",
+     c, backref = -2)

[[1]]
[1] "25" "30"
```

The *RCurl* package is useful for communicating with webserver and sending GET or POST requests. RamiGO uses the postForm() function to communicate with the AmiGO webserver. The *png* package is used to convert the webserver response for an png request into an actual png file. The *igraph* package is used to build an graph object representing the tree that was parsed from an DOT format file.

3 Example

The RamiGO package currently provides two functions that enable the user to retrieve directed acyclic trees from AmiGO and parse the GraphViz DOT format. An example on how to use the functions is given below.

To retrieve a tree from AmiGO, the user has to provide a vector of GO ID's. For example GO:0051130, GO:0019912, GO:0005783, GO:0043229 and GO:0050789. These GO ID's represent entries from the three GO categories: Biological Process, Molecular Function and Cellular Component. The given GO ID's can be highlighted with different colors within the tree, therefor the user has to provide a vector of colors for each GO ID. A request could look like this:

```
> goIDs <- c("GO:0051130", "GO:0019912", "GO:0005783", "GO:0043229",
+     "GO:0050789")
> color <- c("lightblue", "red", "yellow", "green", "pink")
> pngRes <- getAmigoTree(goIDs = goIDs, color = color, filename = "example",
+     picType = "png", saveResult = TRUE)

png: 1407 x 965 [8], 5628 bytes, 0x6, 0, 0
   -filter-> 8-bits, 5628 bytes, 0x6
```

The GO tree representing the given GO ID's is dowloaded to the file ”example.png” (see Figure 1); the file extension is created automatically according to picType. The request for a svg file is similar:

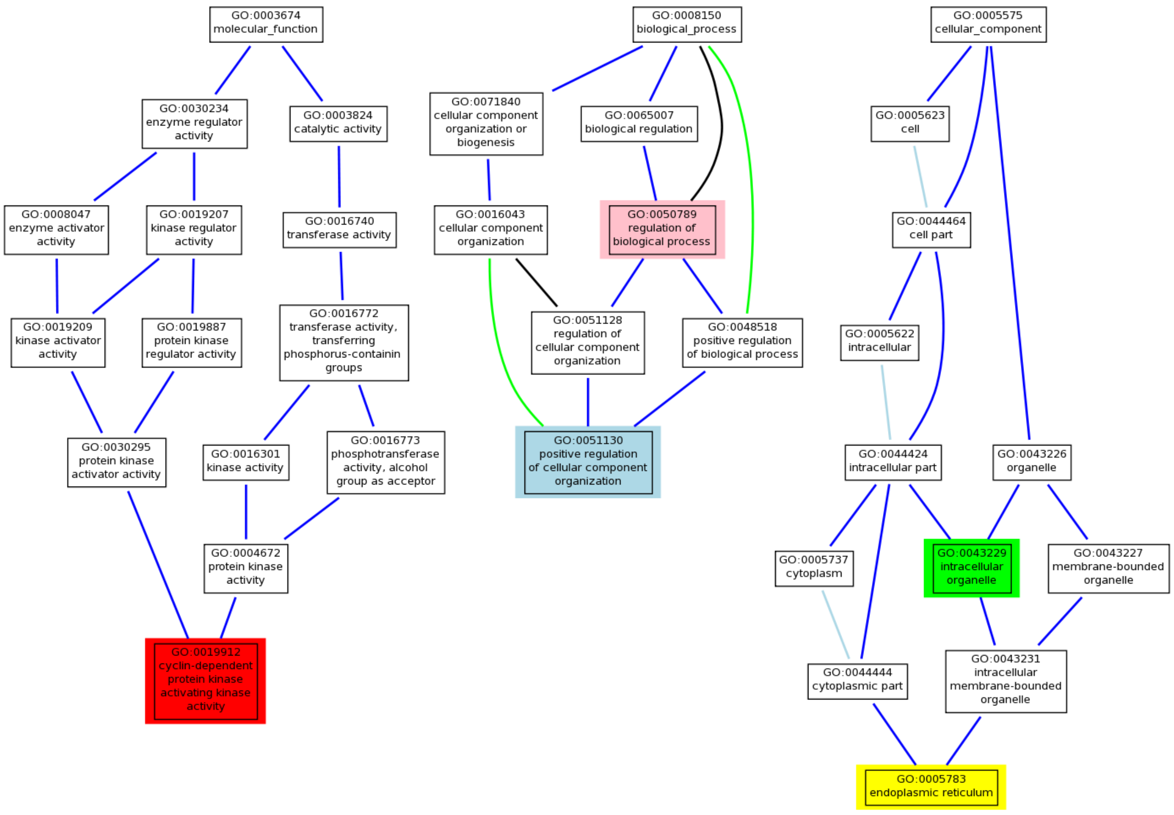
GO:0003674 molecular_function
GO:0008150 biological_process
GO:0005575 cellular_component
GO:0030234 enzyme regulator activity
GO:0003824 catalytic activity
GO:0071840 cellular component organization or biogenesis
GO:0065007 biological regulation
GO:0005623 cell
GO:0008047 enzyme activator activity
GO:0019207 kinase regulator activity
GO:0016740 transferase activity
GO:0016043 cellular component organization
GO:0050789 regulation of biological process
GO:0044464 cell part
GO:0019209 kinase activator activity
GO:0019887 protein kinase regulator activity
GO:0016772 transferase activity, transferring phosphorus-containin groups
GO:0051128 regulation of cellular component organization
GO:0048518 positive regulation of biological process
GO:0005622 intracellular
GO:0030295 protein kinase activator activity
GO:0016301 kinase activity
GO:0016773 phosphotransferase activity, alcohol group as acceptor
GO:0051130 positive regulation of cellular component organization
GO:0044424 intracellular part
GO:0043226 organelle
GO:0004672 protein kinase activity
GO:0005737 cytoplasm
GO:0043229 intracellular organelle
GO:0043227 membrane-bounded organelle
GO:0019912 cyclin-dependent protein kinase activating kinase activity
GO:0044444 cytoplasmic part
GO:0043231 intracellular membrane-bounded organelle
GO:0005783 endoplasmic reticulum

Figure 1: Example PNG returned from AmiGO.

```
> svgRes <- getAmigoTree(goIDs = goIDs, color = color, filename = "example",
+     picType = "svg", saveResult = TRUE)
```

svgRes is a vector with the svg picture in xml format. In order to further analyze the tree, RamiGO provides the possibility to retrieve the tree in the GraphViz DOT format. The function readAmigoTree parses these DOT format files and returns multiple objects. A graph object, an adjacency matrix representing the graph, a data.frame with the annotation for each node, the relations (edges) between the nodes and a data.frame with the leaves of the tree and their annotation. An example could look like this:

```
> dotRes <- getAmigoTree(goIDs = goIDs, color = color, filename = "example",
+     picType = "dot", saveResult = TRUE)
> tt <- readAmigoDot(object = dotRes)
> ## reading the file would also work!
> ## tt <- readAmigoDot(filename='example.dot')
> str(tt)
```

```
List of 5
```

```
 $ graph    :List of 9
  ..$ : num 34
  ..$ : logi TRUE
  ..$ : num [1:46] 0 1 2 3 3 4 5 5 6 6 ...
  ..$ : num [1:46] 1 18 3 8 27 5 7 27 2 4 ...
  ..$ : num [1:46] 0 1 2 3 4 5 6 7 8 9 ...
  ..$ : num [1:46] 22 0 24 8 2 9 5 10 6 11 ...
  ..$ : num [1:35] 0 1 2 3 5 6 8 12 13 13 ...
  ..$ : num [1:35] 0 1 3 4 5 6 8 8 10 13 ...
  ..$ :List of 4
  .. ..$ : num [1:2] 1 0
  .. ..$ : Named list()
  .. ..$ :List of 1
  .. .. ..$ name: chr [1:34] "node1" "node2" "node3" "node4" ...
  .. ..$ : list()
  ..- attr(*, "class")= chr "igraph"
 $ adjMatrix: num [1:34, 1:34] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:34] "node1" "node2" "node3" "node4" ...
  .. ..$ : chr [1:34] "node1" "node2" "node3" "node4" ...
 $ annot    :'data.frame':        34 obs. of  6 variables:
  ..$ node       : chr [1:34] "node1" "node2" "node3" "node4" ...
  ..$ GO_ID      : chr [1:34] "GO:0008047" "GO:0019209" "GO:0071840" "GO:0016043" ...
  ..$ description: chr [1:34] "enzyme activator activity" "kinase activator activity" "cel
  ..$ color      : chr [1:34] "#000000" "#000000" "#000000" "#000000" ...
  ..$ fillcolor  : chr [1:34] "#ffffff" "#ffffff" "#ffffff" "#ffffff" ...
  ..$ fontcolor  : chr [1:34] "#000000" "#000000" "#000000" "#000000" ...
 $ relations:'data.frame':        46 obs. of  6 variables:
  ..$ parent   : chr [1:46] "node1" "node2" "node3" "node4" ...
  ..$ child    : chr [1:46] "node2" "node19" "node4" "node9" ...
  ..$ arrowhead: chr [1:46] "none" "none" "none" "none" ...
  ..$ arrowtail: chr [1:46] "normal" "normal" "normal" "normal" ...
  ..$ color    : chr [1:46] "blue" "blue" "blue" "green" ...
  ..$ style    : chr [1:46] "bold" "bold" "bold" "bold" ...
 $ leaves   :'data.frame':        3 obs. of  6 variables:
  ..$ node       : chr [1:3] "node9" "node22" "node34"
  ..$ GO_ID      : chr [1:3] "GO:0051130" "GO:0019912" "GO:0005783"
  ..$ description: chr [1:3] "positive regulation of cellular component organization" "cyc
  ..$ color      : chr [1:3] "#000000" "#000000" "#000000"
  ..$ fillcolor  : chr [1:3] "lightblue" "red" "yellow"
  ..$ fontcolor  : chr [1:3] "#000000" "#000000" "#000000"
```

The leaves of the tree are returned in `tt$leaves`:

```
> tt$leaves[, c("node", "GO_ID", "description")]
```

```
     node      GO_ID                                                 description
9   node9 GO:0051130     positive regulation of cellular component organization
22 node22 GO:0019912 cyclin-dependent protein kinase activating kinase activity
34 node34 GO:0005783                                       endoplasmic reticulum
```

In order to export the tree to an GML file that is readable by Cytoscape, you have to call the `adjM2gml` with some of the results from the `readAmigoDot` function. The following example creates a GML file by internally calling the `exportCytoGML`:

```
> adjM2gml(tt$adjMatrix, tt$relations$color, tt$annot$fillcolor,
+     tt$annot$GO_ID, tt$annot$description, "example")
```

The results is a GML file named example.gml that can be imported into Cytoscape as a network file.

4 A usefull extension to GSEA

The *RamiGO* package provides an extremely helpful extension to the GSEA software, in java as well as in R, if run with genesets from GO (C5 in MSigDB). *RamiGO* provides a mapping from GO terms returned from GSEA to official GO ID's. The mapping is stored in the data object `c5.go.mapping`.

```
> data(c5.go.mapping)
> head(c5.go.mapping)
```

```
                   description       goid
1                  NUCLEOPLASM GO:0005654
2 EXTRINSIC_TO_PLASMA_MEMBRANE GO:0019897
3               ORGANELLE_PART GO:0044422
4         CELL_PROJECTION_PART GO:0044463
5 CYTOPLASMIC_VESICLE_MEMBRANE GO:0030659
6               GOLGI_MEMBRANE GO:0000139
```

One of the ways to avoid running GSEA in R is to call the java application of GSEA from R with the `system()` function. An example for a preranked GSEA would be:

```
> ## paths to gsea jar and gmt file
> exe.path <- exe.path.string
> gmt.path <- gmt.path.string
> gsea.collapse <- "false"
> ## number of permutations
> nperm <- 10000
> gsea.seed <- 54321
> gsea.out <- "out-folder"
> ## build GSEA command
> gsea.report <- "report-file"
> rnk.path <- "rank-file"
> gsea.cmd <- sprintf("java -Xmx4g -cp %s xtools.gsea.GseaPreranked -gmx %s
-collapse %s -nperm %i -rnk %s -scoring_scheme weighted -rpt_label %s
-include_only_symbols true -make_sets true -plot_top_x 75 -rnd_seed %i
-set_max 500 -set_min 15 -zip_report true -out %s -gui false",
+     exe.path, gmt.path, gsea.collapse, nperm, rnk.path, gsea.report,
+     gsea.seed, gsea.out)
> ## execute command on the system
> system(gsea.cmd)
```

The results are stored in a folder with the name specified in `gsea.out`. The subfolder `gsea.report` has the detailed results in comma separated files and html pages. In the `gsea.cmd` string above we specified a few parameters which can be changed according to the type of analysis.

- plot_top_x: the number of results that should have an individual result page linked to the main index.html.
- set_max and set_min: limits the analysis to genesets that have more than 15 and less than 500 genes.

Once the GSEA analysis is finished, the important result files are xls files in the `gsea.report` folder. Named gsea_report_for_na_pos_<some number>.xls and gsea_report_for_na_neg_<some number>.xls. We can read them into R with the following command:

```
> resn <- "xxx"
> tt <- rbind(read.table(sprintf("%s/%s/gsea_report_for_na_pos_%s.xls",
+     gsea.out, gsea.report, resn), stringsAsFactors = FALSE, sep = "\t",
+     header = TRUE),
read.table(sprintf("%s/%s/gsea_report_for_na_neg_%s.xls",
+     gsea.out, gsea.report, restn), stringsAsFactors = FALSE,
+     sep = "\t", header = TRUE))
```

With all results from the GSEA analysis stored in `tt`, you can extract information from the results and call the `getAmigoTree` mentioned in the example section.

5 View and edit GO trees in Cytoscape

The `adjM2gml` function in *RamiGO* creates a Cytoscape specific GML file (see example section above) that can be imported into Cytoscape and further edited (for example for publication purposes). The GO tree from the example above, parsed with the `readAmigoDot` function, exported with the `adjM2gml` and imported into Cytoscape as a network, looks like Figure 2.

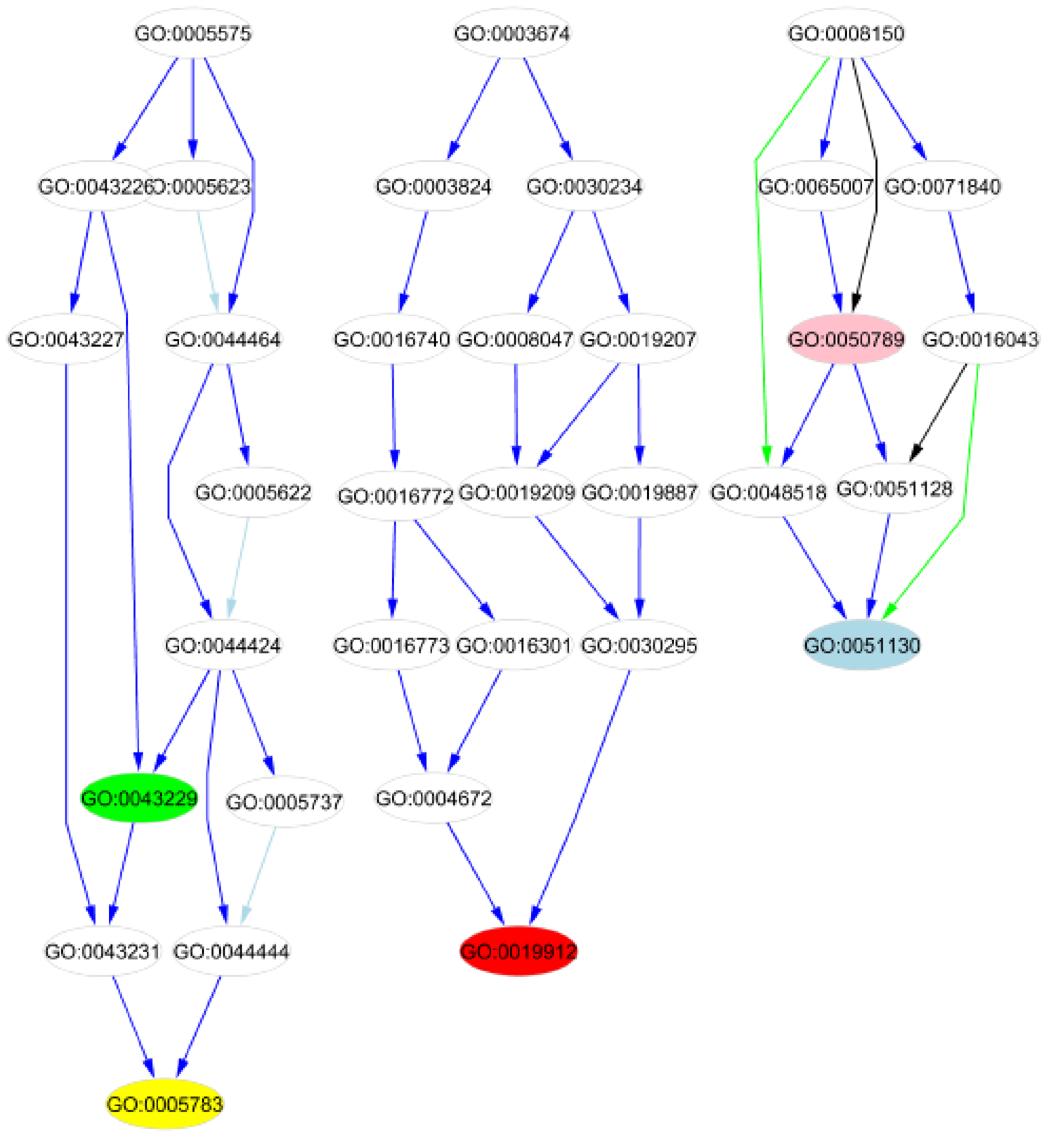

Figure 2: Example GML imported in Cytoscape.

6 Session Info

- R version 2.12.1 (2010-12-16), `i386-pc-mingw32`
- Locale: `LC_COLLATE=C`, `LC_CTYPE=English_United States.1252`, `LC_MONETARY=English_United States.1252`, `LC_NUMERIC=C`, `LC_TIME=English_United States.1252`
- Base packages: base, datasets, grDevices, graphics, methods, stats, tcltk, tools, utils
- Other packages: RCurl 1.5-0.1, RamiGO 0.2, Rcpp 0.9.2, bitops 1.0-4.1,

cacheSweave 0.4-5, codetools 0.2-6, filehash 2.1-1, formatR 0.2-0, getopt 1.15, gsubfn 0.5-5, highlight 0.2-5, igraph 0.5.5-2, optparse 0.9.1, parser 0.0-13, pgfSweave 1.2.1, png 0.1-2, proto 0.3-9.2, stashR 0.3-3, tikzDevice 0.6.1

- Loaded via a namespace (and not attached): digest 0.4.2