

How to Use pkgDepTools

Seth Falcon

Program in Computational Biology
Fred Hutchinson Cancer Research Center

August 28, 2007

1 Introduction

The `pkgDepTools` package provides tools for computing and analyzing dependency relationships among R packages. With it, you can build a graph-based representation of the dependencies among all packages in a list of CRAN-style package repositories. There are utilities for computing installation order of a given package and, if the `RCurl` package is available, estimating the download size required to install a given package and its dependencies.

This vignette demonstrates the basic features of the package.

2 Graph Basics

A graph consists of a set of nodes and a set of edges representing relationships between pairs of nodes. The relationships among the nodes of a graph are binary; either there is an edge between a pair of nodes or there is not. To model package dependencies using a graph, let the set of packages be the nodes of the graph with directed edges originating from a given package to each of its dependencies. Figure 1 shows a part of the Bioconductor dependency graph for the `Category` package. Since circular dependencies are not allowed, the resulting dependency graph will be a directed acyclic graph (DAG).

3 Building a Dependency Graph

```
> library("pkgDepTools")
> library("Biobase")
> library("Rgraphviz")
```

The `makeDepGraph` function retrieves the meta data for all packages of a specified type (source, win.binary, or mac.binary) from each repository in a list of repository URLs and builds a *graphNEL*¹ instance representing the packages and their dependency relationships.

The function takes four arguments: 1) `repList` a character vector of CRAN-style package repository URLs; 2) `suggests.only` a logical value indicating whether the resulting graph should represent relations from the `Depends` field (`FALSE`, default) or the `Suggests` field (`TRUE`); 3) `type` a string indicating the type of packages to search for, the default is `getOption("pkgType")`; 4) `keep.builtin` which will keep packages that come with a standard R install in the dependency graph (the default is `FALSE`).

Here we use `makeDepGraph` to build dependency graphs of the BioC and CRAN packages. Each dependency graph is a *graphNEL* instance. The out-edges of a given node list its direct dependencies (as shown for package `annotate`). The node attribute “size” gives the size of the package in megabytes (only if `RCurl` is available).

```
> biocUrl <- biocReposList()["bioc"]
> cranUrl <- biocReposList()["cran"]
> biocDeps <- makeDepGraph(biocUrl, type = "source")
> cranDeps <- makeDepGraph(cranUrl, type = "source")

> biocDeps
```

```
A graphNEL graph with directed edges
Number of Nodes = 285
Number of Edges = 447
```

```
> cranDeps
```

```
A graphNEL graph with directed edges
Number of Nodes = 1156
Number of Edges = 697
```

¹See `help("graphNEL-class")`

```

> edges(biocDeps)["annotate"]

$annotate
[1] "Biobase"

> nodeData(biocDeps, n = "annotate", attr = "size")

$annotate
[1] 1.453631

```

4 Using the Dependency Graph

The dependencies of a given package can be visualized using the graph generated by `makeDepGraph` and the `Rgraphviz` package. The graph shown in Figure 1 was produced using the code shown below. The `acc` method from the `graph` package returns a vector of all nodes that are accessible from the given node. Here, it has been used to obtain the complete list of `Category`'s dependencies.

```

> categoryNodes <- c("Category", names(acc(biocDeps, "Category")[[1]]))
> categoryGraph <- subGraph(categoryNodes, biocDeps)
> nn <- makeNodeAttrs(categoryGraph, shape = "ellipse")
> plot(categoryGraph, nodeAttrs = nn)

```

In R, there is no easy way to preview a given package's dependencies and estimate the amount of data that needs to be downloaded even though the `install.packages` function will search for and install package dependencies if you ask it to by specifying `dependencies=TRUE`. The `getInstallOrder` function provides such a "preview".

For computing installation order, it is useful to have a single graph representing the relationships among all packages in all available repositories. Below, we create such a graph combining all CRAN and Bioconductor packages.

```

> allDeps <- makeDepGraph(biocReposList(), type = "source",
+   keep.builtin = TRUE)

```

Calling `getInstallOrder` for package `GOstats`, we see a listing of only those packages that need to be installed. Your results will be different based upon your installed packages.

```
> getInstallOrder("GOstats", allDeps)
```

```
$packages  
character(0)
```

```
$total.size  
numeric(0)
```

When `needed.only=FALSE`, the complete dependency list is returned regardless of what packages are currently installed.

```
> getInstallOrder("GOstats", allDeps, needed.only = FALSE)
```

```
$packages  
      ?MB      ?MB      ?MB      ?MB      0.21MB  
"methods"  "stats"  "graphics"  "utils"  "cluster"  
  0.31MB  16.62MB      ?MB  1.71MB  1.45MB  
"graph"    "GO"      "tools"    "Biobase"  "annotate"  
  1.29MB   0.23MB      ?MB   0.79MB   0.23MB  
"RBGL"     "KEGG"    "splines"  "survival" "genefilter"  
   ?MB      ?MB      0.26MB   1.21MB   0.21MB  
"grid"    "grDevices"  "lattice"  "Matrix"  "Category"  
  1.27MB  
"GOstats"
```

```
$total.size  
[1] 25.79567
```

The edge directions of the dependency graph can be reversed and the resulting graph used to determine the set of packages that make use of (even indirectly) a given package. For example, one might like to know which packages make use of the `methods` package. Here is one way to do that:

```
> allDepsOnMe <- reverseEdgeDirections(allDeps)  
> usesMethods <- dijkstra.sp(allDepsOnMe, start = "methods")$distance  
> usesMethods <- usesMethods[is.finite(usesMethods)]  
> length(usesMethods) - 1
```

```
[1] 778
```

```
> table(usesMethods)
```

```
usesMethods
```

```
 0  1  2  3  4
1 251 176 350  1
```

```
> toLatex(sessionInfo())
```

- R version 2.5.1 (2007-06-27), x86_64-unknown-linux-gnu
- Locale: LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=
- Base packages: base, datasets, graphics, grDevices, methods, stats, tools, utils
- Other packages: annotate 1.14.1, Biobase 1.14.1, geneplotter 1.14.0, graph 1.14.2, lattice 0.16-3, pkgDepTools 1.2.1, RBGL 1.12.0, RCurl 0.8-0, Rgraphviz 1.14.1

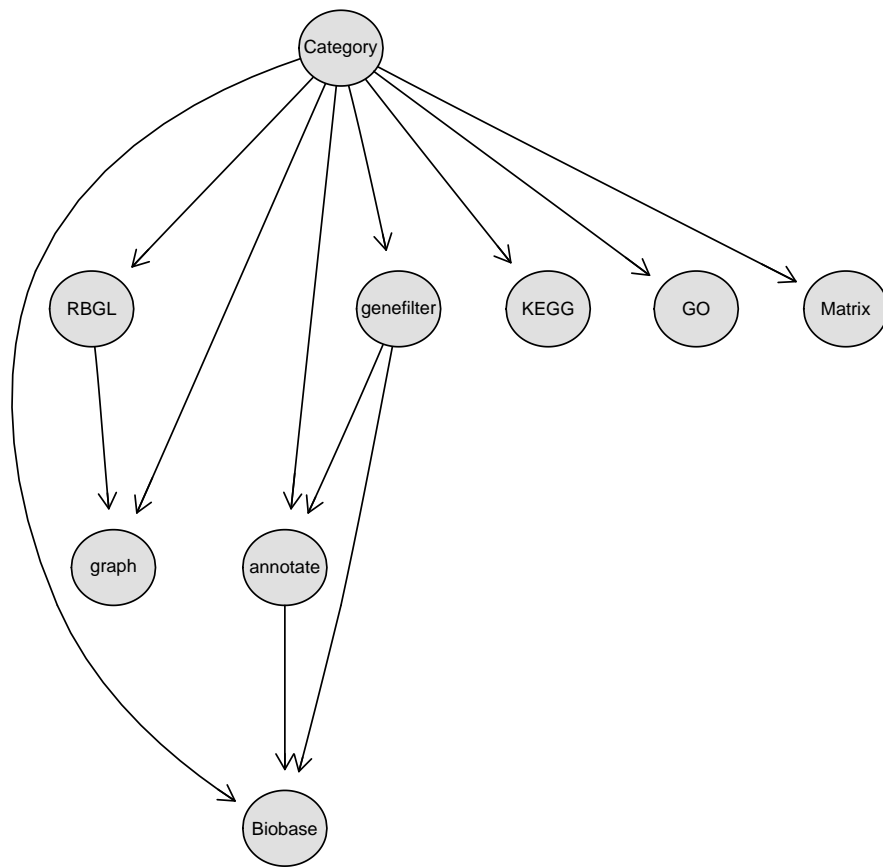


Figure 1: The dependency graph for the `Category` package.