Package 'packdep'

July 2, 2014

Type Package

Title Mapping dependencies among R packages

Version 0.3.1

Date 2014-02-11

Depends R (>= 2.9.0), igraph (>= 0.5.2), utils

Suggests Rgraphviz (>= 1.20.3)

Author Radhakrishnan Nagarajan <rnagarajan@uky.edu> and Marco Scutari <marco.scutari@gmail.com>

Maintainer Marco Scutari <marco.scutari@gmail.com>

Description packdep elucidates the dependencies between user-contributed R packages and identifies key packages according to social network analysis metrics.

License GPL (>= 2)

LazyLoad yes

LazyData yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-02-12 13:07:47

R topics documented:

packdep-package 2
centrality 3
dependencies 4
funcmap 5
map.depends 6
plot centrality 7
plot dependencies 8
related.packages 9

Index **11**

packdep-package *Mapping dependencies among R packages*

Description

Mapping dependencies among R packages, both on CRAN and BioConductor.

Details

Package:	packdep
Type:	Package
Version:	0.3.1
Date:	2014-02-11
License:	GPLv2 or later

Re-using user-contributed packages is encouraged in R open-source environment and promotes transparency, reproducibility while minimizing redundancy. However, understanding package inter-dependencies can significantly improve sustainability.

User-contributed R packages undergo revisions with time, while some of the packages are orphaned, others are re-used extensively as ("Suggests", "Imports" and "Depends"). **packdep** elucidates the dependencies between user contributed R packages in CRAN and BioConductor. It subsequently identifies key packages by ranking them using a battery of social network analysis metrics. **packdep** can also be used to identify function names that are shared across packages. This is especially important, since a given function name across two different packages can have distinct purposes. With the number of user-contributed packages increasing steadily with time, understanding the dependencies between R packages and minimizing the reuse of existing function names may be critical for sustainable software development. It is believed **packdep** may also provide preliminary insights into identifying recommended packages and eventual resource allocation in the R open-source environment.

Author(s)

Radhakrishnan Nagarajan <rnagarajan@uams.edu>
Department of Biostatistics
University of Arkansas for Medical Sciences

Marco Scutari <marco.scutari@stat.unipd.it>
Department of Statistical Sciences
University of Padova

References

Wasserman S, Faust K (2007) *Social Network Analysis*. Cambridge University Press.

Examples

```
library(packdep)

## Not run:
# map dependencies among CRAN packages.
d1 = map.depends()
# use a specific CRAN mirror.
d2 = map.depends(contriburl = contrib.url("http://cran.r-project.org"))
d2
# map dependencies among BioConductor packages.
d3 = map.depends(repository = "bioc")
d3

## End(Not run)
```

`centrality` *Packages' betweenness and closeness*

Description

Estimate packages' betweenness and closeness indexes.

Usage

```
centrality(x, order.by = c("none", "betweenness", "incloseness",
    "outcloseness", "indegree", "outdegree", "degree"),
    decreasing = TRUE)
```

Arguments

`x`	an object of class `igraph` returned by `map.depends`.
`order.by`	a character string. Possible values are `none` (rows in the returned data frame are not sorted), `betweenness` (rows are sorted in order of decreasing betweenness) or `closeness` (rows are sorted in order of decreasing closeness).
`decreasing`	a boolean value. Should the sort order be increasing or decreasing?

Value

A data frame with the following columns:

`package`	the name of the package.
`betweenness`	the betweenness centrality measure for the package.
`closeness`	the closeness centrality measure for the package.

The class of the data frame is changed to `c("packdep.centrality", "data.frame")` to override the dispatch of the plotting methods (currently `plot` and `hist`).

Author(s)

Radhakrishnan Nagarajan and Marco Scutari

References

Wasserman S, Faust K (2007) *Social Network Analysis*. Cambridge University Press.

See Also

`map.depends` and `plot centrality`.

Examples

```
## Not run:
d = map.depends()
centrality(d)

# write a table containing the packages on CRAN,
# ordered by decreasing betweenness, to a specified file.
res = centrality(d, order.by = "betweenness")
write.table(res, file = "centrality.txt", row.names = FALSE)

## End(Not run)
```

`dependencies` *Count dependencies and reverse dependencies*

Description

Count dependencies and reverse dependencies.

Usage

```
dependencies(x, order.by = c("none", "dependencies", "reverse"),
    decreasing = TRUE)
```

Arguments

`x`	an object of class `igraph` returned by `map.depends`.
`order.by`	a character string. Possible values are `none` (rows in the returned data frame are not sorted), `dependencies` (rows are sorted in order of decreasing number of dependencies) or `reverse` (rows are sorted in order of decreasing number of reverse dependencies).
`decreasing`	a boolean value. Should the sort order be increasing or decreasing?

Details

This function uses `available.packages` to get the Depends, Imports and Suggests for all packages in a CRAN or BioConductor mirror.

If `repository` is set to `bioc`, the `contriburl` parameter is ignored and the URL of BioConductor is set via the `biocinstallRepos` function.

Value

A data frame with the following columns:

`package`	the name of the package.
`dependencies`	number dependencies of the package.
`reverse`	number of reverse dependencies (i.e. the number of packages that have the package in question as a dependency) of the package.

The class of the data frame is changed to `c("packdep.dependencies", "data.frame")` to override the dispatch of the plotting methods (currently `plot` and `hist`).

Author(s)

Radhakrishnan Nagarajan and Marco Scutari

See Also

`map.depends` and `plot dependencies`.

Examples

```
## Not run:
d = map.depends()
dependencies(d)

# write a table containing the packages on CRAN,
# ordered by decreasing betweenness, to a specified file.
res = dependencies(d, order.by = "reverse")
write.table(res, file = "reverse.dependencies.txt", row.names = FALSE)

## End(Not run)
```

`funcmap` *Map similar funtionality in CRAN's packages*

Description

Find out which packages share similar functions or topics.

Usage

```
funcmap()
```

Details

This functions retrieves help pages' titles from all the packages on CRAN and judges whether the latter are related by intersection (i.e. if the same title appears in two packages they are considered related).

Value

An adjacency matrix; elements with a value of 1 denote packages for which at least one match have been found.

Author(s)

Radhakrishnan Nagarajan

Examples

```
## Not run:
z = funcmap()
z["bnlearn", "deal"]
z["bnlearn", "ggm"]
z["bnlearn", "igraph"]

## End(Not run)
```

---

map.depends *Retrieve packages' dependencies*

Description

Retrieve packages' dependencies from either a CRAN mirror or BioConductor.

Usage

```
map.depends(repository = c("cran", "bioc"),
  contriburl = contrib.url(getOption("repos")),
  dependencies = c("Imports", "Depends", "Suggests"))
```

Arguments

repository	a character string, either cran (for a CRAN mirror) or bioc (for BioConductor).
contriburl	URL(s) of the contrib sections of the repositories.
dependencies	a character vector, specifying which kind of dependencies are to be evaluated. Possible values are one or more among "Imports", "Depends" and "Suggests".

Details

This function uses `available.packages` to get the Depends, Imports and Suggests for all packages in a CRAN or BioConductor mirror.

If `repository` is set to `bioc`, the `contriburl` parameter is ignored and the URL of BioConductor is set via the `biocinstallRepos` function.

Value

An object of class `igraph`, representing the dependencies as an directed acyciclic graph.

Author(s)

Radhakrishnan Nagarajan and Marco Scutari

`plot centrality` *Plot closeness and betweenness.*

Description

Plot closeness and betweenness from the object returned by `centrality`.

Usage

```
## S3 method for class 'packdep.centrality'
plot(x, type = c("betweenness", "incloseness",
    "outcloseness", "indegree", "outdegree", "degree"),
     logscale = c("", "x", "y", "xy"), breaks = 10, freq = FALSE,
     xlab = NULL, ylab = NULL, ...)
## S3 method for class 'packdep.centrality'
hist(x, type = c("betweenness", "incloseness",
     "outcloseness", "indegree", "outdegree", "degree"),
     xlab = NULL, ylab = "frequency", main = "", ...)
```

Arguments

`x`	an object of class `packdep.centrality`.
`type`	a character string. Possible choices are `closeness` (closeness values are plotted) and `betweenness` (betweenness values are plotted).
`logscale`	a character string, which indicates which axes are to to be plotted on a logarithmic scale (none by default).
`breaks`	the breaks used in the plot; see `hist` for details.
`freq`	a logical value. If `TRUE` absolute frequencies (counts) are used; if `FALSE` relative frequencies are used instead.
`xlab`	a title for the x axis.
`ylab`	a title for the y axis.
`main`	an overall title for the plot.
`...`	other parameters to be passed through to plotting functions.

Author(s)

Radhakrishnan Nagarajan and Marco Scutari

See Also

centrality.

Examples

```
## Not run:
d = map.depends()
c = centrality(d)

hist(c, freq = TRUE)
plot(c, freq = FALSE, logscale = "y")

## End(Not run)
```

---

plot dependencies *Plot dependencies and reverse dependencies.*

Description

Plot dependencies and reverse dependencies from the object returned by dependencies.

Usage

```
## S3 method for class 'packdep.dependencies'
plot(x, type = c("dependencies", "reverse"),
  logscale = c("", "x", "y", "xy"), breaks = 10, freq = FALSE,
  xlab = NULL, ylab = NULL, ...)
## S3 method for class 'packdep.dependencies'
hist(x, type = c("dependencies", "reverse"),
  xlab = NULL, ylab = "frequency", main = "", ...)
```

Arguments

x	an object of class packdep.dependencies.
type	a character string. Possible choices are dependencies (dependencies are plotted) and reverse (reverse dependencies are plotted).
logscale	a character string, which indicates which axes are to to be plotted on a logarithmic scale (none by default).
breaks	the breaks used in the plot; see hist for details.
freq	a logical value. If TRUE absolute frequencies (counts) are used; if FALSE relative frequencies are used instead.
xlab	a title for the x axis.

ylab	a title for the y axis.
main	an overall title for the plot.
...	other parameters to be passed through to plotting functions.

Author(s)

Radhakrishnan Nagarajan and Marco Scutari

See Also

dependencies.

Examples

```
## Not run:
d = map.depends()
c = dependencies(d)

hist(c, freq = TRUE)
plot(c, freq = FALSE, logscale = "y")

## End(Not run)
```

---

related.packages *Plot a package's neighbourhood*

Description

Plot a graph containing the set of packages closely related by forward or reverse dependencies to a specified package.

Usage

```
related.packages(x, node = NULL, order = 1, graphviz = FALSE)
```

Arguments

x	an object of class igraph returned by map.depends.
node	a character string, the name of a package present in x.
order	a positive integer, giving the order of the neighbourhood (i.e. the maximum distance from the package specified by the node parameter).
graphviz	a boolean value. If TRUE **Rgraphviz** is loaded and used to produce the plot; if FALSE **igraph** plotting functions are used instead.

Value

The object of class igraph used for the plot.

Author(s)

Marco Scutari

Examples

```
## Not run:
d = map.depends()
related.packages(d, 1, "boot")

## End(Not run)
```

Index

∗Topic **graphs**
centrality, 3
dependencies, 4
funcmap, 5
map.depends, 6
related.packages, 9
∗Topic **hplot**
plot centrality, 7
plot dependencies, 8
related.packages, 9
∗Topic **package**
packdep-package, 2

centrality, 3, *8*

dependencies, 4, *9*

funcmap, 5

hist, *7, 8*
hist.packdep.centrality *(*plot centrality*)*, 7
hist.packdep.dependencies *(*plot dependencies*)*, 8

map.depends, *4, 5*, 6

packdep *(*packdep-package*)*, 2
packdep-package, 2
plot centrality, 7
plot dependencies, 8
plot.packdep.centrality *(*plot centrality*)*, 7
plot.packdep.dependencies *(*plot dependencies*)*, 8

related.packages, 9