

# Package ‘packdep’

January 2, 2012

**Type** Package

**Title** Mapping dependencies among R packages

**Version** 0.2

**Date** 2009-07-26

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

**Author** Radhakrishnan Nagarajan <rnagarajan@uams.edu> and Marco Scutari  
<marco.scutari@stat.unipd.it>

**Maintainer** Marco Scutari <marco.scutari@stat.unipd.it>

**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

**Repository** CRAN

**Date/Publication** 2009-07-26 17:51:33

## R topics documented:

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

<b>Index</b>	<b>9</b>
--------------	----------

---

packdep-package

*Mapping dependencies among R packages*

---

## Description

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

## Details

Package: packdep  
Type: Package  
Version: 0.2  
Date: 2009-07-26  
License: GPLv2 or later

Reusing user-contributed packages is encouraged in R open-source environment and promotes transparency, reproducibility while minimizing redundancy. However, package inter-dependencies can have a significant impact on their sustainability.

User-contributed R-packages undergo revisions with time and may even be orphaned in the absence of an active maintainer. **packdep** elucidates the dependencies (“Suggests”, “Imports” and “Depends”) 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. With the number of user-contributed packages increasing steadily with time, understanding dependencies between R packages may be critical for sustainable software development. **packdep** may also provide preliminary insights into identifying “recommended” packages.

## 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", "closeness"))
```

## 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).

## Value

A data frame with the following columns:

<code>package</code>	the name of the package.
<code>betweenness</code>	the betweenness centrality measure for the package.
<code>closeness</code>	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"))
```

**Arguments**

x	an object of class <code>igraph</code> returned by <code>map.depends</code> .
order.by	a character string. Possible values are <code>none</code> (rows in the returned data frame are not sorted), <code>dependencies</code> (rows are sorted in order of decreasing number of dependencies) or <code>reverse</code> (rows are sorted in order of decreasing number of reverse dependencies).

**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)
```

---

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")))
```

**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.

**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 acyclic 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", "closeness"),
     logscale = c("", "x", "y", "xy"), breaks = 10, freq = FALSE,
     xlab = NULL, ylab = NULL, ...)
## S3 method for class 'packdep.centrality'
hist(x, type = c("betweenness", "closeness"),
     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 be plotted on a logarithmic scale (none by default).
breaks	the breaks used in the plot; see <a href="#">hist</a> 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 be plotted on a logarithmic scale (none by default).
breaks	the breaks used in the plot; see <a href="#">hist</a> 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	<i>Plot a package's neighbourhood</i>
------------------	---------------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

x	an object of class <code>igraph</code> returned by <code>map.depends</code> .
order	a positive integer, giving the order of the neighbourhood (i.e. the maximum distance from the package specified by the <code>node</code> parameter).
node	a character string, the name of a package present in <code>x</code> .

**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](#)
- map.depends, [5](#)
- related.packages, [8](#)

## \*Topic **hplot**

- plot centrality, [6](#)
- plot dependencies, [7](#)
- related.packages, [8](#)

## \*Topic **package**

- packdep-package, [2](#)

centrality, [3](#), [6](#)

dependencies, [4](#), [8](#)

hist, [6](#), [7](#)

hist.packdep.centrality (plot  
centrality), [6](#)

hist.packdep.dependencies (plot  
dependencies), [7](#)

map.depends, [3](#), [5](#), [5](#)

packdep (packdep-package), [2](#)

packdep-package, [2](#)

plot centrality, [3](#), [6](#)

plot dependencies, [5](#), [7](#)

plot.packdep.centrality (plot  
centrality), [6](#)

plot.packdep.dependencies (plot  
dependencies), [7](#)

related.packages, [8](#)