

Package ‘pkgndep’

November 15, 2022

Type Package

Title Analyze Dependency Heaviness of R Packages

Version 1.2.1

Date 2022-11-13

Depends R (>= 4.0.0)

Imports ComplexHeatmap (>= 2.6.0), GetoptLong, GlobalOptions, utils, grid, hash, methods, BiocManager, brew, BiocVersion

Suggests knitr, rmarkdown, svglite, callr, rjson, Rook, igraph, ggplot2, ggrepel, base64, testthat, cowplot

Description A new metric named 'dependency heaviness' is proposed that measures the number of additional dependency packages that a parent package brings to its child package and are unique to the dependency packages imported by all other parents. The dependency heaviness analysis is visualized by a customized heatmap. The package is described in <[doi:10.1093/bioinformatics/btac449](https://doi.org/10.1093/bioinformatics/btac449)>. We have also performed the dependency heaviness analysis on the CRAN/Bioconductor package ecosystem and the results are implemented as a web-based database which provides comprehensive tools for querying dependencies of individual R packages.

URL <https://github.com/jokergoo/pkgndep>

VignetteBuilder knitr

License MIT + file LICENSE

NeedsCompilation no

Author Zuguang Gu [aut, cre] (<<https://orcid.org/0000-0002-7395-8709>>)

Maintainer Zuguang Gu <z.gu@dkfz.de>

Repository CRAN

Date/Publication 2022-11-15 19:50:02 UTC

R topics documented:

all_pkg_stat_snapshot	2
check_pkg	3

child_dependency	4
co_heaviness	4
dependency_database	5
dependency_heatmap	6
dependency_report	7
dependency_website	7
downstream_dependency	8
get_all_functions_imported_to_children	8
gini_index	9
heaviness	9
heaviness_db	10
heaviness_from_upstream	10
heaviness_on_children	11
heaviness_on_downstream	12
is_parent	13
is_upstream	13
loaded_packages	14
load_all_pkg_dep	15
load_from_pkgndep_db	15
load_pkg_db	16
load_pkg_downstream_dependency_path_snapshot	17
load_pkg_stat_snapshot	17
parent_dependency	18
pkgndep	19
pkgndep_opt	20
plot.pkgndep	20
print.heaviness_db_info	21
print.pkgndep	22
reformat_db	22
required_dependency_packages	23
upstream_dependency	24
Index	25

all_pkg_stat_snapshot *The complete table of dependency heaviness for all CRAN/Bioconductor packages*

Description

The complete table of dependency heaviness for all CRAN/Bioconductor packages

Usage

```
all_pkg_stat_snapshot()
```

Value

The columns are self-explanatory from the column names.

Examples

```
# There is no example  
NULL
```

check_pkg	<i>Check whether a package is available</i>
-----------	---

Description

Check whether a package is available

Usage

```
check_pkg(pkg, bioc = FALSE)
```

Arguments

pkg	The name of the package.
bioc	Whether it is a Bioconductor package.

Details

One of the suggestions to avoid heavy dependencies is to put parent packages that are not frequently used to 'Suggests' and to load them when the corresponding functions are used. Here the [check_pkg](#) function helps to check whether these parent packages are available and if not, it prints messages to guide users to install the corresponding packages.

Examples

```
# There is no example  
NULL
```

child_dependency *Get child dependency for a package*

Description

Get child dependency for a package

Usage

```
child_dependency(package, fields = NULL, snapshot = TRUE)
```

Arguments

package	Package name.
fields	Which fields in DESCRIPTION? Values should be in Depends, Imports, LinkingTo, Suggests and Enhances.
snapshot	If it is TRUE, the package database generated on 2021-10-28 is used. If it is FALSE, the package database is directly retrieved from CRAN/Bioconductor.

Details

The dependency information is based on packages retrieved from CRAN/Bioconductor on 2021-10-28.

Value

A data frame with child packages as well as its heaviness on its child packages. If snapshot is set to FALSE, heaviness on child packages is set to NA.

Examples

```
## Not run:
child_dependency("ComplexHeatmap")

## End(Not run)
```

co_heaviness *Co-heaviness for pairs of parent packages*

Description

Co-heaviness for pairs of parent packages

Usage

```
co_heaviness(x, rel = FALSE, a = 10, jaccard = FALSE)
```

Arguments

x	An object returned by <code>pkgndep</code> .
rel	Whether to return the absolute measure or the relative measure.
a	A constant added for calculating the relative measure.
jaccard	Whether to return Jaccard coefficient?

Details

Denote a package as P and its two strong parent packages as A and B, i.e., parent packages in "Depends", "Imports" and "LinkingTo", the co-heaviness for A and B is calculated as follows.

Denote S_A as the set of reduced dependency packages when only moving A to "Suggests" of P, and denote S_B as the set of reduced dependency packages when only moving B to "Suggests" of P, denote S_{AB} as the set of reduced dependency packages when moving A and B together to "Suggests" of P, the co-heaviness of A, B on P is calculated as $\text{length}(\text{setdiff}(S_{AB}, \text{union}(S_A, S_B)))$, which is the number of reduced package only caused by co-action of A and B.

Note the co-heaviness is only calculated for parent packages in "Depends", "Imports" and "LinkingTo".

When `jaccard` is set to TRUE, the function returns jaccard coefficient. $\text{setdiff}(S_{AB}, \text{union}(S_A, S_B))$ is actually the set of dependencies imported by and only by two parent packages A and B. Thus the jaccard coefficient is calculated as $\text{length}(\text{setdiff}(S_{AB}, \text{union}(S_A, S_B))) / \text{length}(S_{AB})$.

Examples

```
## Not run:  
x = pkgndep("DESeq2")  
hm = co_heaviness(x)  
ComplexHeatmap::Heatmap(hm)  
co_heaviness(x, jaccard = TRUE)  
  
## End(Not run)
```

dependency_database *The database of package dependency analysis*

Description

The database of package dependency analysis

Usage

```
dependency_database()
```

Details

The analysis is based on all 22076 CRAN/Bioconductor packages retrieved on 2022-06-08.

Examples

```
if(interactive()) {
  dependency_database()
}
```

dependency_heatmap *Make the dependency heatmap*

Description

Make the dependency heatmap

Usage

```
dependency_heatmap(x, pkg_fontsize = 10*cex, title_fontsize = 12*cex,
  legend_fontsize = 10*cex, fix_size = !dev.interactive(), cex = 1,
  help = TRUE, file = NULL, res = 144)
```

Arguments

x	An object from pkgndep .
pkg_fontsize	Font size for the package names.
title_fontsize	Font size for the title.
legend_fontsize	Font size for the legends.
fix_size	Should the rows and columns in the heatmap have fixed size?
cex	A factor multiplied to all font sizes.
help	Whether to print help message?
file	A path of the figure. The size of the figure is automatically calculated.
res	Resolution of the figure (only for png and jpeg).

Details

If `fix_size` is set to `TRUE`. The size of the whole plot can be obtained by:

```
size = dependency_heatmap(x, fix_size = TRUE)
```

where `size` is a numeric vector of length two which are the width and height of the whole heatmap.

If `file` argument is set, the size of the figure is automatically calculated.

If there are no dependency packages stored in `x`, `NULL` is returned.

Value

A vector of two numeric values (in inches) that correspond to the width and height of the plot.

Examples

```
# See examples in `pkgndep()`.
```

dependency_report *HTML report for package dependency analysis*

Description

HTML report for package dependency analysis

Usage

```
dependency_report(pkg, file = NULL)
```

Arguments

pkg	An object from pkgndep .
file	The path of the html file. If it is not specified, the report will be automatically opened in the web browser.

Value

The path of the HTML file of the report.

Examples

```
if(interactive()) {  
  x = readRDS(system.file("extdata", "ComplexHeatmap_dep.rds", package = "pkgndep"))  
  dependency_report(x)  
}
```

dependency_website *The database of package dependency analysis*

Description

The database of package dependency analysis

Usage

```
dependency_website()
```

Details

The analysis is based on all 22076 CRAN/Bioconductor packages retrieved on 2022-06-08.

Examples

```
if(interactive()) {  
  dependency_website()  
}
```

downstream_dependency *Get downstream dependency for a package*

Description

Get downstream dependency for a package

Usage

```
downstream_dependency(package, snapshot = TRUE)
```

Arguments

package	Package name.
snapshot	If it is TRUE, the package database generated on 2021-10-28 is used. If it is FALSE, the package database is directly retrieved from CRAN/Bioconductor.

Details

Downstream packages with relations of Depends, Imports and LinkingTo are retrieved.

Value

A data frame with all downstream packages.

Examples

```
## Not run:  
downstream_dependency("ComplexHeatmap")  
  
## End(Not run)
```

get_all_functions_imported_to_children
Get functions that are imported to child packages

Description

Get functions that are imported to child packages

Usage

```
get_all_functions_imported_to_children(package)
```

Arguments

package	Package name.
---------	---------------

Value

It returns a list of names of functions that are imported to every of its child packages.

Examples

```
# There is no example
NULL
```

gini_index	<i>Gini index</i>
------------	-------------------

Description

Gini index

Usage

```
gini_index(v)
```

Arguments

`v` A numeric vector.

Examples

```
x = readRDS(system.file("extdata", "ComplexHeatmap_dep.rds", package = "pkgndep"))
gini_index(x$heaviness[x$which_required])
```

heaviness	<i>Heaviness from parent packages</i>
-----------	---------------------------------------

Description

Heaviness from parent packages

Usage

```
heaviness(x, rel = FALSE, a = 10, only_strong_dep = FALSE)
```

Arguments

`x` An object returned by [pkgndep](#).
`rel` Whether to return the absolute measure or the relative measure.
`a` A constant added for calculating the relative measure.
`only_strong_dep` Whether to only return the heaviness for strong parents.

Details

The heaviness from a parent package is calculated as follows: If package B is in the Depends/Imports/LinkingTo fields of package A, which means, package B is necessary for package A, denote v_1 as the total numbers of packages required for package A, and v_2 as the total number of required packages if moving package B to Suggests (which means, now B is not necessary for A). The absolute measure is simply $v_1 - v_2$ and relative measure is $(v_1 + a)/(v_2 + a)$.

In the second scenario, if B is in the Suggests/Enhances fields of package A, now v_2 is the total number of required packages if moving B to Imports, the absolute measure is $v_2 - v_1$ and relative measure is $(v_2 + a)/(v_1 + a)$.

Value

A numeric vector.

Examples

```
x = readRDS(system.file("extdata", "ComplexHeatmap_dep.rds", package = "pkgndep"))
heaviness(x)
heaviness(x, rel = TRUE)
```

heaviness_db	<i>Version of the heaviness database</i>
--------------	--

Description

Version of the heaviness database

Usage

```
heaviness_db
```

Examples

```
heaviness_db
```

heaviness_from_upstream	<i>Heaviness from all upstream packages</i>
-------------------------	---

Description

Heaviness from all upstream packages

Usage

```
heaviness_from_upstream(package)
```

Arguments

package A package name.

Value

A named vector.

Examples

```
# There is no example  
NULL
```

heaviness_on_children *Heaviness on all child packages*

Description

Heaviness on all child packages

Usage

```
heaviness_on_children(package, add_values_attr = FALSE)
```

Arguments

package A package name.
add_values_attr Whether to include "values" attribute? Internally used.

Value

The value is the mean heaviness of the package on all its child packages.

Examples

```
## Not run:  
heaviness_on_children("ComplexHeatmap")  
  
## End(Not run)
```

 heaviness_on_downstream

Heaviness on all downstream packages

Description

Heaviness on all downstream packages

Usage

```
heaviness_on_downstream(package, add_values_attr = FALSE, via = NULL)
```

Arguments

package	A package name.
add_values_attr	Whether to include "values" attribute? Internally used.
via	Whether to only consider downstream packages via a intermediate package?

Value

The value is the mean heaviness of the package on all its downstream packages. Denote n as the number of all its downstream packages, k_i as the number of required packages for package i , v_1 as the total number of required packages for all downstream packages, i.e. $v_1 = \sum_i k_i$. Denote p_i as the number of required packages if moving package to Suggests, and v_2 as the total number of required packages, i.e. $v_2 = \sum_i p_i$. The final heaviness on downstream packages is $(v_1 - v_2)/n$.

Note since the interaction from package to its downstream packages may go through several intermediate packages, which means, the reduction of required packages for a downstream package might be joint effects from all its upstream packages, thus, to properly calculate the heaviness of a package to its downstream packages, we first make a copy of the package database and move package to Suggests for all packages which depends on package. Then for all downstream packages of package, dependency analysis by [pkgndep](#) is redone with the modified package database. Finally, the heaviness on downstream packages is collected and the mean heaviness is calculated.

Examples

```
## Not run:
heaviness_on_downstream("ComplexHeatmap")

## End(Not run)
```

is_parent	<i>Test the parent-child relationship</i>
-----------	---

Description

Test the parent-child relationship

Usage

```
is_parent(parent, child, ...)
```

Arguments

parent	A vector of package names.
child	A single package name.
...	Pass to parent_dependency .

Value

A logical vector.

Examples

```
# There is no example  
NULL
```

is_upstream	<i>Test upstream - downstream relationship</i>
-------------	--

Description

Test upstream - downstream relationship

Usage

```
is_upstream(upstream, package, ...)
```

Arguments

upstream	A vector of package names.
package	A single package name.
...	Pass to upstream_dependency .

Value

A logical vector.

Examples

```
# There is no example  
NULL
```

loaded_packages	<i>Loaded packages</i>
-----------------	------------------------

Description

Loaded packages

Usage

```
loaded_packages(pkg, verbose = TRUE)
```

Arguments

pkg	A package name.
verbose	Whether to print messages.

Details

It loads pkg into a new R session and collects which other packages are loaded by parsing the output from [sessionInfo](#).

Value

A data frame.

Examples

```
loaded_packages("ComplexHeatmap")
```

load_all_pkg_dep	<i>Load dependency data of all packages</i>
------------------	---

Description

Load dependency data of all packages

Usage

```
load_all_pkg_dep(hash = TRUE)
```

Arguments

hash Whether to convert the named list to a hash table by [hash](#).

Details

It loads the package dependency analysis for all CRAN/Bioconductor packages done on 2022-06-08.

Value

A list (as a hash table) of pkgndep objects where each element corresponds to the analysis on one package.

Examples

```
## Not run:  
lt = load_all_pkg_dep()  
length(lt)  
head(names(lt))  
lt[["ggplot2"]]  
  
## End(Not run)
```

load_from_pkgndep_db	<i>Load pre-computed results</i>
----------------------	----------------------------------

Description

Load pre-computed results

Usage

```
load_from_pkgndep_db(file)
```

Arguments

file File name.

Details

Internally used.

Examples

```
# There is no example
NULL
```

load_pkg_db	<i>Load package database</i>
-------------	------------------------------

Description

Load package database

Usage

```
load_pkg_db(lib = NULL, snapshot = FALSE, verbose = TRUE, online = TRUE)
```

Arguments

lib Local library path. If the value is NA, only remote package database is used.

snapshot Internally used. If it is TRUE, the package database generated on 2022-06-08 is used.

verbose Whetehr to print messages.

online If the value is TRUE, it will directly use the package database file from CRAN/Bioconductor. If the value is FALSE, it uses the cached package database retrieved on 2022-06-08.

Details

It loads the package database from CRAN/Bioconductor and locally installed packages. The database object internally is cached for repeated use of other functions in this package.

Value

A pkg_db class object.

Examples

```
## Not run:  
pkg_db = load_pkg_db(lib = NA)  
pkg_db  
  
## End(Not run)
```

```
load_pkg_downstream_dependency_path_snapshot  
Load downstream dependency paths for all packages
```

Description

Load downstream dependency paths for all packages

Usage

```
load_pkg_downstream_dependency_path_snapshot()
```

Details

It loads the package dependency analysis for all CRAN/Bioconductor packages done on 2022-06-08.

Value

A list.

Examples

```
## Not run:  
downstream_path_list = load_pkg_downstream_dependency_path_snapshot()  
downstream_path_list[["ComplexHeatmap"]]  
  
## End(Not run)
```

```
load_pkg_stat_snapshot  
Load all package dependency statistics
```

Description

Load all package dependency statistics

Usage

```
load_pkg_stat_snapshot()
```

Details

It loads the package dependency analysis for all CRAN/Bioconductor packages done on 2022-06-08.

Value

A data frame of various columns.

Examples

```
## Not run:
df = load_pkg_stat_snapshot()
head(df)

## End(Not run)
```

parent_dependency	<i>Get parent dependency for a package</i>
-------------------	--

Description

Get parent dependency for a package

Usage

```
parent_dependency(package, fields = NULL, snapshot = TRUE)
```

Arguments

package	Package name.
fields	Which fields in DESCRIPTION? Values should be in Depends, Imports, LinkingTo, Suggests and Enhances.
snapshot	If it is TRUE, the package database generated on 2021-10-28 is used. If it is FALSE, the package database is directly retrieved from CRAN/Bioconductor.

Details

The dependency information is based on packages retrieved from CRAN/Bioconductor on 2021-10-28.

Value

A data frame with parent packages as well as their heaviness on package. If snapshot is set to FALSE, heaviness on child packages is set to NA.

Examples

```
## Not run:
parent_dependency("ComplexHeatmap")

## End(Not run)
```

pkgndep	<i>Package dependency analysis</i>
---------	------------------------------------

Description

Package dependency analysis

Usage

```
pkgndep(package, load = FALSE, verbose = TRUE, online = TRUE)
```

Arguments

package	Package name. The value should be 1. a CRAN/Bioconductor package, 2. an installed package, 3. a path of a local package, 4. URL of a GitHub repository.
load	Check which other packages are loaded into R session (directly or indirectly) when loading pkg.
verbose	Whether to show messages.
online	If the value is TRUE, it will directly use the package database file from CRAN/Bioconductor. If the value is FALSE, it uses the cached package database retrieved on 2021-10-28.

Details

The package database for dependency analysis is always directly retrieved from CRAN/Bioconductor.

Value

A pkgndep object.

Examples

```
## Not run:
x = pkgndep("ComplexHeatmap")

## End(Not run)
# The `x` variable generated by `pkgndep()` is already saved in this package.
x = readRDS(system.file("extdata", "ComplexHeatmap_dep.rds", package = "pkgndep"))
x
dependency_heatmap(x)
```

pkgndep_opt *Global parameters for pkgndep*

Description

Global parameters for pkgndep

Usage

```
pkgndep_opt(..., RESET = FALSE, READ.ONLY = NULL, LOCAL = FALSE, ADD = FALSE)
```

Arguments

...	Arguments for the parameters, see "details" section
RESET	Reset to default values.
READ.ONLY	Please ignore.
LOCAL	Please ignore.
ADD	Please ignore.

Details

There are following parameters:

`bioc_version` The bioconductor version. By default it is the version corresponding to the R version under use.

Examples

```
pkgndep_opt
```

plot.pkgndep *Make the dependency heatmap*

Description

Make the dependency heatmap

Usage

```
## S3 method for class 'pkgndep'
plot(x, ...)
```

Arguments

x	An object from pkgndep .
...	Other arguments.

Details

Please use [dependency_heatmap](#) instead.

Value

A vector of two numeric values (in inches) that correspond to the width and height of the plot.

Examples

```
# See examples in `pkgndep()`.
```

```
print.heaviness_db_info
```

Print the heaviness_db_info object

Description

Print the `heaviness_db_info` object

Usage

```
## S3 method for class 'heaviness_db_info'  
print(x, ...)
```

Arguments

<code>x</code>	A <code>heaviness_db_info</code> object.
<code>...</code>	Other arguments

Value

No value is returned.

Examples

```
heaviness_db
```

print.pkgndep	<i>Print method</i>
---------------	---------------------

Description

Print method

Usage

```
## S3 method for class 'pkgndep'  
print(x, ...)
```

Arguments

x	An object from pkgndep .
...	Other arguments.

Value

No value is returned.

Examples

```
# See examples in `pkgndep()`.
```

reformat_db	<i>Format the package database</i>
-------------	------------------------------------

Description

Format the package database

Usage

```
reformat_db(db)
```

Arguments

db	A data frame returned from available.packages or installed.packages .
----	---

Details

It reformats the data frame of the package database into a pkg_db class object.

Value

A pkg_db class object. There are the following methods:

pkg_db\$get_meta(package, field=NULL) field can take values in "Package", "Version" and "Repository".

pkg_db\$get_dependency_table(package) Get the dependency table.

pkg_db\$get_rev_dependency_table(package) Get the reverse dependency table.

pkg_db\$package_dependencies(package, recursive=FALSE, reverse=FALSE, which="strong", simplify=FALSE)
All the arguments are the same as in [package_dependencies](#). Argument simplify controls whether to return a data frame or a simplified vector.

Examples

```
## Not run:
db = available.packages()
db2 = reformat_db(db)

# a pkg_db object generated on 2021-10-28 can be loaded by load_pkg_db()
db2 = load_pkg_db(snapshot = TRUE)
db2
db2$get_meta("ComplexHeatmap")
db2$get_dependency_table("ComplexHeatmap")
db2$get_rev_dependency_table("ComplexHeatmap")
db2$package_dependencies("ComplexHeatmap")
db2$package_dependencies("ComplexHeatmap", recursive = TRUE)

## End(Not run)
```

required_dependency_packages
Required dependency packages

Description

Required dependency packages

Usage

```
required_dependency_packages(x, all = FALSE)
```

Arguments

x An object from [pkgndep](#).

all Whether to include the packages required if also including packages from "Suggests"/"Enhances" field.

Details

The function returns all upstream packages.

Value

A vector of package names.

Examples

```
## Not run:  
x = readRDS(system.file("extdata", "ComplexHeatmap_dep.rds", package = "pkgndep"))  
required_dependency_packages(x)  
  
## End(Not run)
```

upstream_dependency *Get upstream dependency for a package*

Description

Get upstream dependency for a package

Usage

```
upstream_dependency(package, snapshot = TRUE)
```

Arguments

package	Package name.
snapshot	If it is TRUE, the package database generated on 2021-10-28 is used. If it is FALSE, the package database is directly retrieved from CRAN/Bioconductor.

Details

Upstream packages with relations of "Depends", "Imports" and "LinkingTo" are retrieved.

Value

A data frame with all upstream packages.

Examples

```
## Not run:  
upstream_dependency("ComplexHeatmap")  
  
## End(Not run)
```


Index

all_pkg_stat_snapshot, [2](#)
available.packages, [22](#)

check_pkg, [3, 3](#)
child_dependency, [4](#)
co_heaviness, [4](#)

dependency_database, [5](#)
dependency_heatmap, [6, 21](#)
dependency_report, [7](#)
dependency_website, [7](#)
downstream_dependency, [8](#)

get_all_functions_imported_to_children,
[8](#)
gini_index, [9](#)

hash, [15](#)
heaviness, [9](#)
heaviness_db, [10](#)
heaviness_from_upstream, [10](#)
heaviness_on_children, [11](#)
heaviness_on_downstream, [12](#)

installed.packages, [22](#)
is_parent, [13](#)
is_upstream, [13](#)

load_all_pkg_dep, [15](#)
load_from_pkgndep_db, [15](#)
load_pkg_db, [16](#)
load_pkg_downstream_dependency_path_snapshot,
[17](#)
load_pkg_stat_snapshot, [17](#)
loaded_packages, [14](#)

package_dependencies, [23](#)
parent_dependency, [13, 18](#)
pkgndep, [5–7, 9, 12, 19, 20, 22, 23](#)
pkgndep_opt, [20](#)
plot.pkgndep, [20](#)
print.heaviness_db_info, [21](#)
print.pkgndep, [22](#)
reformat_db, [22](#)
required_dependency_packages, [23](#)
sessionInfo, [14](#)
upstream_dependency, [13, 24](#)