

# Package ‘pkgndep’

May 9, 2026

**Type** Package

**Title** Analyze Dependency Heaviness of R Packages

**Version** 2.0.0

**Date** 2026-04-08

**Depends** R (>= 4.0.0)

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

**Suggests** knitr, rmarkdown, svglite, base64, testthat

**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, described in <[doi:10.1016/j.jss.2023.111610](https://doi.org/10.1016/j.jss.2023.111610)>.

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

**VignetteBuilder** knitr

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**NeedsCompilation** no

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

**Maintainer** Zuguang Gu <[guzuguang@suat-sz.edu.cn](mailto:guzuguang@suat-sz.edu.cn)>

**Repository** CRAN

**Date/Publication** 2026-04-08 09:40:08 UTC

## Contents

co_heaviness . . . . .	2
current_bioc_version . . . . .	3

heaviness . . . . .	3
heaviness_report . . . . .	4
load_pkg_db . . . . .	5
pkgndep . . . . .	5
pkgndep_opt . . . . .	6
plot.pkgndep . . . . .	7
print.pkgndep . . . . .	8
reformat_db . . . . .	8

<b>Index</b>	<b>10</b>
--------------	-----------

---

co_heaviness	<i>Co-heaviness for pairs of parent packages</i>
--------------	--

---

## 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:
# DESeq version 1.36.0, the dependencies have been changed in later versions.
x = readRDS(system.file("extdata", "DESeq2_dep.rds", package = "pkgndep"))
hm = co_heaviness(x)
ComplexHeatmap::Heatmap(hm)
co_heaviness(x, jaccard = TRUE)

## End(Not run)
```

---

current\_bioc\_version    *Current Bioconductor Version*

---

**Description**

Current Bioconductor Version

**Usage**

```
current_bioc_version()
```

**Details**

The version information is from <https://bioconductor.org/config.yaml>

---

heaviness                    *Heaviness from parent packages*

---

**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  $(v_2 + a)/(v_1 + a)$ .

**Examples**

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

---

heaviness_report	<i>HTML report for package dependency heaviness analysis</i>
------------------	--

---

**Description**

HTML report for package dependency heaviness analysis

**Usage**

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

**Arguments**

pkg	An object from <a href="#">pkgndep()</a> .
file	The path of the html file. If it is not specified, the report will be automatically opened in the web browser.
...	pass to <a href="#">heaviness_report()</a> .

**Value**

The path of the HTML file of the report.

**Examples**

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

---

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

---

**Description**

Load package database

**Usage**

```
load_pkg_db(  
  lib = .libPaths(),  
  bioc_version = pkgndep_opt$bioc_version,  
  verbose = TRUE  
)
```

**Arguments**

lib	Local library. The value can be a character vector specifying the local library paths, or NULL which directly uses <code>.libPaths()</code> . A value of NA means not to use local libraries.
bioc_version	Biconductor version.
verbose	Whether to print messages.

**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.

---

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

---

**Description**

Package dependency analysis

**Usage**

```
pkgndep(  
  package,  
  lib = .libPaths(),  
  bioc_version = pkgndep_opt$bioc_version,  
  verbose = TRUE  
)
```

**Arguments**

package	Package name. The value can be 1. a CRAN/Bioconductor package, 2. an installed package, 3. a path of a local package, 4. URL of a GitHub repository.
lib	Local library. The value can be a character vector specifying the local library paths, or NULL which directly uses <code>.libPaths()</code> . A value of NA means not to use local libraries.
bioc_version	Bioconductor version.
verbose	Whether to show messages.

**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. Use numeric version for devel branch.

---

plot.pkgndep	<i>Make the dependency heatmap</i>
--------------	------------------------------------

---

### Description

Make the dependency heatmap

### Usage

```
## S3 method for class 'pkgndep'  
plot(x, ...)  
  
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 <code>pkgndep()</code> .
...	Other arguments.
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.

---

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.

---

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

---

**Description**

Format the package database

**Usage**

```
reformat_db(db, version = NULL)
```

**Arguments**

db	A data frame returned from <code>utils::available.packages()</code> or <code>utils::installed.packages()</code> .
version	Version of the database, a self-defined text.

**Details**

It reformats the data frame of the package database into a pkg\_db object.

**Value**

A pkg\_db 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 [tools::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)  
  
## End(Not run)
```

# Index

`.libPaths()`, [5](#), [6](#)

`co_heaviness`, [2](#)  
`current_bioc_version`, [3](#)

`dependency_heatmap (plot.pkgndep)`, [7](#)  
`dependency_report (heaviness_report)`, [4](#)

`heaviness`, [3](#)  
`heaviness_report`, [4](#)  
`heaviness_report()`, [4](#)

`load_pkg_db`, [5](#)

`pkgndep`, [5](#)  
`pkgndep()`, [2-4](#), [7](#)  
`pkgndep_opt`, [6](#)  
`plot.pkgndep`, [7](#)  
`print.pkgndep`, [8](#)

`reformat_db`, [8](#)

`tools::package_dependencies()`, [9](#)

`utils::available.packages()`, [8](#)  
`utils::installed.packages()`, [8](#)