

# Package: pkgnet (via r-universe)

October 10, 2024

**Type** Package

**Title** Get Network Representation of an R Package

**Version** 0.5.0.9999

**Maintainer** Brian Burns <brian.burns.opensource@gmail.com>

**Description** Tools from the domain of graph theory can be used to quantify the complexity and vulnerability to failure of a software package. That is the guiding philosophy of this package. 'pkgnet' provides tools to analyze the dependencies between functions in an R package and between its imported packages. See the pkgnet website for vignettes and other supplementary information.

**Imports** assertthat, covr, data.table, DT, futile.logger, glue, igraph(>= 1.3), knitr, magrittr, methods, R6, rlang, rmarkdown(>= 1.9), tools, visNetwork

**Suggests** ggplot2, pkgdown, testthat, webshot, withr

**License** BSD\_3\_clause + file LICENSE

**URL** <https://github.com/uptake/pkgnet>, <https://uptake.github.io/pkgnet/>

**BugReports** <https://github.com/uptake/pkgnet/issues>

**RoxygenNote** 7.3.1

**Repository** <https://uptake.r-universe.dev>

**RemoteUrl** <https://github.com/uptake/pkgnet>

**RemoteRef** HEAD

**RemoteSha** 994723de87c39a4a76434100543a639e3ff2d286

## Contents

|                                 |   |
|---------------------------------|---|
| CreatePackageReport . . . . .   | 2 |
| CreatePackageVignette . . . . . | 3 |
| DefaultReporters . . . . .      | 3 |
| DependencyReporter . . . . .    | 4 |

|                               |    |
|-------------------------------|----|
| DirectedGraph . . . . .       | 5  |
| FunctionReporter . . . . .    | 6  |
| InheritanceReporter . . . . . | 8  |
| PackageReport . . . . .       | 9  |
| SummaryReporter . . . . .     | 10 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>12</b> |
|--------------|-----------|

---

CreatePackageReport *pkgnet Analysis Report for an R package*

---

## Description

Create a standalone HTML report about a package and its networks.

## Usage

```
CreatePackageReport(
  pkg_name,
  pkg_reporters = DefaultReporters(),
  pkg_path = NULL,
  report_path = tempfile(pattern = pkg_name, fileext = ".html")
)
```

## Arguments

|               |   |
|---------------|---|
| pkg_name      | (string) name of a package  |
| pkg_reporters | (list) a list of package reporters  |
| pkg_path      | (string) The path to the package repository. If given, coverage will be calculated for each function. pkg_path can be an absolute or relative path. |
| report_path   | (string) The path and filename of the output report. Default report will be produced in the temporary directory.                                    |

## Value

an instantiated [PackageReport](#) object

---

CreatePackageVignette *pkgnet Report as Vignette*

---

### Description

Create pkgnet package report as an R Markdown vignette. This vignette can be rendered into a standard HTML vignette with the `knitr::rmarkdown` vignette engine into HTML vignettes upon package building. It is also compatible with #' `pkgdown` sites. See the vignette "[Publishing Your pkgnet Package Report](#)" for details about how to use this function, as well as [our example for pkgnet](#).

### Usage

```
CreatePackageVignette(  
  pkg = ".",  
  pkg_reporters = list(DependencyReporter$new(), FunctionReporter$new()),  
  vignette_path = file.path(pkg, "vignettes", "pkgnet-report.Rmd")  
)
```

### Arguments

`pkg` (string) path to root directory of package of interest

`pkg_reporters` (list) a list of initialized package reporters

`vignette_path` (string) The location of a file to store the output vignette file at. Must be an .Rmd file. By default, this will be '`<pkg>/vignettes/pkgnet-report.Rmd`' relative to the input to `pkg`

---

DefaultReporters *Default Reporters*

---

### Description

Instantiates a list of default reporters to feed into [CreatePackageReport](#).

### Usage

```
DefaultReporters()
```

### Details

Default reporters are:

- [SummaryReporter](#)
- [DependencyReporter](#)
- [FunctionReporter](#)

Note, [InheritanceReporter](#) is not included in the default list.

If desired, append a new instance of [InheritanceReporter](#) to the DefaultReporters list.

ex: `c(DefaultReporters(), InheritanceReporter$new())`

## Value

list of instantiated reporter objects

---

DependencyReporter      *Recursive Package Dependency Reporter*

---

## Description

This reporter looks at the recursive network of its dependencies on other packages. This allows a developer to understand how individual dependencies might lead to a much larger set of dependencies, potentially informing decisions on including or removing them.

## Super classes

[pkgnet::AbstractPackageReporter](#) -> [pkgnet::AbstractGraphReporter](#) -> DependencyReporter

## Active bindings

`report_markdown_path` (character string) path to R Markdown template for this reporter. Read-only.

## Methods

### Public methods:

- [DependencyReporter\\$new\(\)](#)
- [DependencyReporter\\$clone\(\)](#)

**Method** `new()`: Initialize an instance of the reporter.

*Usage:*

```
DependencyReporter$new(
  dep_types = c("Imports", "Depends", "LinkingTo"),
  installed = TRUE
)
```

*Arguments:*

`dep_types` (character vector) The sections within the DESCRIPTION file to be counted as dependencies. By default, `c("Imports", "Depends", "LinkingTo")` is chosen.

`installed` (logical) If TRUE, consider only installed packages when building dependency network.

*Returns:* Self, invisibly.

*Examples:*

```

\donttest{
# Instantiate an object
reporter <- DependencyReporter$new()

# Seed it with a package
reporter$set_package("ggplot2")

}

```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
DependencyReporter$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

### See Also

Other Network Reporters: [FunctionReporter](#), [InheritanceReporter](#)

Other Package Reporters: [FunctionReporter](#), [InheritanceReporter](#), [SummaryReporter](#)

### Examples

```

## -----
## Method `DependencyReporter$new`
## -----

# Instantiate an object
reporter <- DependencyReporter$new()

# Seed it with a package
reporter$set_package("ggplot2")

```

---

DirectedGraph

*Directed Graph Network Model*

---

### Description

R6 class defining a directed graph model for representing a network, including methods to calculate various measures from graph theory. The [igraph](#) package is used as a backend for calculations.

This class isn't intended to be initialized directly; instead, [network reporter objects](#) will initialize it as its `pkg_graph` field. If you have a network reporter named `reporter`, then you access this object's public interface through `pkg_graph`—for example,

```
reporter$pkg_graph$node_measures('hubScore')
```

**Super class**

`pkgnet::AbstractGraph` -> DirectedGraph

**Active bindings**

`default_node_measures` character vector of default node measures. See *Node Measures* section in [DirectedGraphMeasures](#) for details about each measure. Read-only.

`default_graph_measures` character vector of default graph measures. See *Graph Measures* section in [DirectedGraphMeasures](#) for details about each measure. Read-only.

**Methods****Public methods:**

- [DirectedGraph\\$clone\(\)](#)

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
DirectedGraph$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

**See Also**

[DirectedGraphMeasures](#)

---

FunctionReporter

*Function Interdependency Reporter*

---

**Description**

This reporter looks at the network of interdependencies of its defined functions. Measures of centrality from graph theory can indicate which function is most important to a package. Combined with unit test coverage information—also provided by this reporter—it can be used as a powerful tool to prioritize test writing.

**Details**

**R6 Method Support:** R6 classes are supported, with their methods treated as functions by the reporter.

- R6 methods will be named like `<classname>$<methodtype>$<methodname>`, e.g., `FunctionReporter$private_method`
- Note that the class name used will be the **name of the generator object in the package's namespace**.
- The `classname` attribute of the class is **not** used. In general, it is not required to be defined or the same as the generator object name. This attribute is used primarily for S3 dispatch.

**Known Limitations::**

- Using non-standard evaluation to refer to things (e.g, dataframe column names) that have the same name as a function will trick FunctionReporter into thinking the function was called. This can be avoided if you don't use reuse function names for other purposes.
- Functions stored as list items and not assigned to the package namespace will be invisible to FunctionReporter.
- Calls to methods of instantiated R6 or reference objects will not be recognized. We don't have a reliable way of identifying instantiated objects, or identifying their class.
- Reference class methods are not yet supported. They will not be identified as nodes by FunctionReporter.

**Super classes**

`pkgnet::AbstractPackageReporter` -> `pkgnet::AbstractGraphReporter` -> `FunctionReporter`

**Active bindings**

`report_markdown_path` (character string) path to R Markdown template for this reporter. Read-only.

**Methods****Public methods:**

- `FunctionReporter$calculate_default_measures()`
- `FunctionReporter$clone()`

**Method** `calculate_default_measures()`: Calculates the default node and network measures for this reporter.

*Usage:*

```
FunctionReporter$calculate_default_measures()
```

*Returns:* Self, invisibly.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
FunctionReporter$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

**See Also**

Other Network Reporters: [DependencyReporter](#), [InheritanceReporter](#)

Other Package Reporters: [DependencyReporter](#), [InheritanceReporter](#), [SummaryReporter](#)

---

InheritanceReporter    *Class Inheritance Reporter*

---

## Description

This reporter takes a package and traces the class inheritance structure. Currently the following object-oriented systems are supported:

- S4 Classes
- Reference Classes (sometimes informally called "R5")
- R6 Classes

S3 classes are not supported, as their inheritance is defined on an ad hoc basis per object and not formally by class definitions.

## Details

Note the following details about class naming:

- Reference Classes : The name passed as `Class` in `setRefClass` is used as the node name by this reporter. This is the class name that is used when specifying inheritance. The generator object returned by `setRefClass` does not have to be assigned and can have a different name.
- R6 Classes : The name of the generator object in the package namespace is used as the node name by this reporter. The generator object returned by `R6::R6Class` is what is used when specifying inheritance. The name passed as `classname` passed to `R6::R6Class` can be a different name or even `NULL`.

For more info about R's built-in object-oriented systems, check out the relevant chapter in [Hadley Wickham's \*Advanced R\*](#). For more info about R6, check out their [docs website](#) or the chapter in [Advanced R's second edition](#).

## Super classes

`pkgnet::AbstractPackageReporter` -> `pkgnet::AbstractGraphReporter` -> `InheritanceReporter`

## Active bindings

`report_markdown_path` (character string) path to R Markdown template for this reporter. Read-only.

## Methods

### Public methods:

- `InheritanceReporter$clone()`

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*



```
InheritanceReporter$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

### See Also

Other Network Reporters: [DependencyReporter](#), [FunctionReporter](#)

Other Package Reporters: [DependencyReporter](#), [FunctionReporter](#), [SummaryReporter](#)

---

PackageReport

*R6 Class Representing an R Package Report*

---

### Description

pkgnet compiles one or more package reporters into a package report for a specified package. PackageReport is an R6 class that holds all of those reporters and has a method `render_report()` to generate an HTML report file. You can access each individual reporter and modify it using its methods if you wish.

The function `CreatePackageReport()` is a shortcut for both generating a PackageReport object with instantiated reporters and creating the HTML report in one call.

### Value

Self, invisibly.

### Active bindings

pkg\_name (character string) name of package. Read-only.

pkg\_path (character string) path to source code of the package. Read-only.

report\_path (character string) path and filename of output report.

SummaryReporter Instantiated pkgnet [SummaryReporter](#) object

DependencyReporter Instantiated pkgnet [DependencyReporter](#) object

FunctionReporter Instantiated pkgnet [FunctionReporter](#) object

InheritanceReporter Instantiated pkgnet [InheritanceReporter](#) object

### Methods

#### Public methods:

- [PackageReport\\$new\(\)](#)
- [PackageReport\\$add\\_reporter\(\)](#)
- [PackageReport\\$render\\_report\(\)](#)
- [PackageReport\\$clone\(\)](#)

**Method** `new()`: Initialize an instance of a package report object.

*Usage:*

```
PackageReport$new(
  pkg_name,
  pkg_path = NULL,
  report_path = tempfile(pattern = pkg_name, fileext = ".html")
)
```

*Arguments:*

`pkg_name` (character string) name of package

`pkg_path` (character string) optional directory path to source code of the package. It is used for calculating test coverage. It can be an absolute or relative path.

`report_path` (character string) The path and filename of the output report. Default report will be produced in the temporary directory.

*Returns:* Instantiated package report object.

**Method** `add_reporter()`: Add a reporter to the package report.

*Usage:*

```
PackageReport$add_reporter(reporter)
```

*Arguments:*

`reporter` Instantiated package reporter object

*Returns:* Self, invisibly

**Method** `render_report()`: Render html pkgnet package report.

*Usage:*

```
PackageReport$render_report()
```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
PackageReport$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

---

SummaryReporter

*Package Summary Reporter*

---

**Description**

This reporter provides a high-level overview of a package via its package DESCRIPTION file.

**Super class**

[pkgnet::AbstractPackageReporter](#) -> SummaryReporter

**Active bindings**

report\_markdown\_path (character string) path to R Markdown template for this reporter. Read-only.

**Methods****Public methods:**

- [SummaryReporter\\$get\\_summary\\_view\(\)](#)
- [SummaryReporter\\$clone\(\)](#)

**Method** [get\\_summary\\_view\(\)](#): Returns an htmlwidget object that summarizes the analysis of the reporter. Used when creating a [package report](#).

*Usage:*

```
SummaryReporter$get_summary_view()
```

*Returns:* Self, invisibly.

**Method** [clone\(\)](#): The objects of this class are cloneable with this method.

*Usage:*

```
SummaryReporter$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**See Also**

Other Package Reporters: [DependencyReporter](#), [FunctionReporter](#), [InheritanceReporter](#)

# Index

## \* Graph Classes

DirectedGraph, 5

## \* Main Functions

CreatePackageReport, 2

CreatePackageVignette, 3

## \* Network Reporters

DependencyReporter, 4

FunctionReporter, 6

InheritanceReporter, 8

## \* Package Reporters

DependencyReporter, 4

FunctionReporter, 6

InheritanceReporter, 8

SummaryReporter, 10

## \* Reporters

DefaultReporters, 3

DependencyReporter, 4

FunctionReporter, 6

InheritanceReporter, 8

PackageReport, 9

SummaryReporter, 10

CreatePackageReport, 2, 3, 9

CreatePackageVignette, 3

DefaultReporters, 3

DependencyReporter, 3, 4, 7, 9, 11

DirectedGraph, 5

DirectedGraphMeasures, 6

FunctionReporter, 3, 5, 6, 9, 11

igraph, 5

InheritanceReporter, 4, 5, 7, 8, 9, 11

knitr::rmarkdown, 3

network reporter objects, 5

package report, 11

PackageReport, 2, 9

pkgdown, 3

pkgnet::AbstractGraph, 6

pkgnet::AbstractGraphReporter, 4, 7, 8

pkgnet::AbstractPackageReporter, 4, 7, 8,  
10

R6::R6Class, 8

setRefClass, 8

SummaryReporter, 3, 5, 7, 9, 10