Package: depcheck (via r-universe)

October 24, 2024

Type Package

Title Dependency Use Checker for R Packages

Version 0.1.0

Description A way to check R packages and shiny applications for unused package dependencies, or packages with low usage. Reduces the requirement to have unneccessary dependencies in a project.

License MIT + file LICENSE

Imports formatR

Encoding UTF-8

Language en-GB

Suggests covr, testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.1.2

Repository https://ashbaldry.r-universe.dev

RemoteUrl https://github.com/ashbaldry/depcheck

RemoteRef HEAD

RemoteSha 3f028933244a3137fb8bad3417da38c91d64c94e

Contents

`checkFunctionUse` *Check Function Usage in Code*

Description

Counting the frequency that a function is called within selected code.

Usage

```
checkFunctionUse(function_name, code, package_name = NULL)
```

Arguments

`function_name`	Name of the function to check against
`code`	A character vector of code chunks to check for function use
`package_name`	Name of the package that `function_name` is contained in

Details

If `package_name` is left as `NULL`, then there is a chance the checks will find times where the function is used, but is explicitly called from another package.

There are three stages of checking: - If the function name exists anywhere in the code chunks - If it has not been previously assigned in a code chunk - If the function name is not quoted

At each stage, only keep chunks where the function has been called to improve speed

---

`checkPackageDependencyUse`
Check Package Dependencies

Description

'checkPackageDependencyUse()' checks the `DESCRIPTION` file for packages mentioned in the `Depends` and `Imports` fields.

Using these packages, it then checks the scripts in the R directory to check for function use of each package.

Usage

```
checkPackageDependencyUse(path = ".", include_suggests = FALSE, verbose = TRUE)
```

Arguments

path	Path to the package root directory.
include_suggests	Logical, should the "Suggests" field also be checked for package dependencies or just the Depends and Imports?
verbose	Logical, should informative messages be printed during the dependency evaluation?

Value

An object of class `multi_package_usage`, a named list of the dependencies used, each containing a `data.frame` of all the functions within the package and how often they are used within the project.

The result will flag any packages that have been rarely used in the project.

See Also

`checkProjectDependencyUse`, `checkShinyDependencyUse`

Examples

```
## Not run:
dependency_use <- checkPackageDependencyUse()
summary(dependency_use)

## End(Not run)
```

---

checkPackagesUsage *Check Package Use in Code*

Description

This will check what functions contained within a specified package are used in code.

Usage

```
checkPackagesUsage(package_names, code, verbose = TRUE)

checkPackageUsage(package_name, code)
```

Arguments

package_names, package_name	Name(s) of the package to check against
code	A character vector of code chunks to check for package use
verbose	Logical, should informative messages be printed during the dependency evaluation?

Value

checkPackageUsage will return a data.frame of class package_usage. When printed it will show a summary of the package usage.

checkPackagesUsage will return a list of class multi_package_usage. When printed it will show a summary of all packages mentioned, and flag any rarely used.

---

checkProjectDependencyUse

Check Project Package Dependencies

Description

'checkProjectDependencyUse()' checks R scripts within a project for package calling through the use of library, require and requireNamespace. It also finds packages explicitly used via the pkg::function notation.

Using these packages, it then checks the same R scripts to check for function use of each package.

Usage

```
checkProjectDependencyUse(path = ".", recursive = TRUE, verbose = TRUE)
```

Arguments

path	Path to the directories to search for R scripts. By default looks in the current working directory.
recursive	Logical. Should the R file search recurse into directories?
verbose	Logical, should informative messages be printed during the dependency evaluation?

Value

An object of class multi_package_usage, a named list of the dependencies used, each containing a data.frame of all the functions within the package and how often they are used within the project.

The result will flag any packages that have been rarely used in the project.

See Also

checkPackageDependencyUse, checkShinyDependencyUse

Examples

```
## Not run:
dependency_use <- checkProjectDependencyUse()
summary(dependency_use)

## End(Not run)
```

checkShinyDependencyUse *Check Shiny Application Dependencies*

Description

'checkShinyDependencyUse()' checks the global.R and app.R (or ui.R and server.R) scripts within a project, and any specified directory within the project, for package calling through the use of `library`, `require` and `requireNamespace`. It also finds packages explicitly used via the `pkg::function` notation.

Using these packages, it then checks the same R scripts to check for function use of each package.

Usage

```
checkShinyDependencyUse(path = ".", r_scripts_dir = "R", verbose = TRUE)
```

Arguments

`path`	Path to the application root directory
`r_scripts_dir`	Subdirectories in the shiny application that contain R scripts, write as relative paths to `path`. Default is set to `R`
`verbose`	Logical, should informative messages be printed during the dependency evaluation?

Value

An object of class `multi_package_usage`, a named list of the dependencies used, each containing a `data.frame` of all the functions within the package and how often they are used within the project.

The result will flag any packages that have been rarely used in the project.

See Also

checkProjectDependencyUse, checkPackageDependencyUse

Examples

```
## Not run:
dependency_use <- checkShinyDependencyUse()
summary(dependency_use)

## End(Not run)
```

extractPackageCalls *Extract Package Calls*

Description

The ability to search for package loading (via `library()` or `requireNamespace`), or by direct referencing through `"pkg::function"`

Usage

```
extractPackageCalls(code)
```

Arguments

code	A character vector of code chunks to check for package calls

getPackageFunctions *Get Package Functions*

Description

In order to check if packages are included, need to extract all functions from the package.

Usage

```
getPackageFunctions(package_name)

getInternalPackageFunctions(package_name)
```

Arguments

package_name	Name of the package to find function names for

print.multi_package_usage *Package Usage Printing Options*

Description

Package Usage Printing Options

Usage

```
## S3 method for class 'multi_package_usage'
print(
  x,
  warn_percent_usage = 0.2,
  warn_number_usage = 3,
  ignore_low_usage_packages = character(),
  ...
)
```

Arguments

x	A package_usage data.frame
warn_percent_usage	Minimum percent of functions to be used within a dependent package. Default is 20%
warn_number_usage	Minimum number of functions to be used within a dependent package. Default is 3
ignore_low_usage_packages	A vector of packages to ignore the low usage of, usually when already aware of the low usage, but the dependent package is necessary for the project.
...	Not used

Details

Package usage must be below both thresholds for the warning to appear. With the defaults, if a package has fewer than 5 functions then only 1 function is required to prevent a warning message to appear.

print.package_usage *Package Usage Printing Options*

Description

Package Usage Printing Options

Usage

```
## S3 method for class 'package_usage'
print(x, warn_percent_usage = 0.2, warn_number_usage = 3, ...)
```

Arguments

x	A package_usage data.frame
warn_percent_usage	Minimum percent of functions to be used within a dependent package. Default is 20%
warn_number_usage	Minimum number of functions to be used within a dependent package. Default is 3
...	Not used

Details

Package usage must be below both thresholds for the warning to appear. With the defaults, if a package has fewer than 5 functions then only 1 function is required to prevent a warning message to appear.

readPackageRFiles *Read R Files*

Description

Reading R files for package dependency checks.

readPackageFiles assumes the selected path is a package, and will check the R subdirectory for files to read. readDirectoryFiles is a more generic version of readPackageFiles

Usage

```
readPackageRFiles(path = ".")

readDirectoryRFiles(path = ".")

readRFiles(files, path = NULL)
```

Arguments

path	Location of the package or directory to import R files from
files	Vector of R file paths to import

Details

The default directory for readFiles is NULL, to allow the reading of files from multiple directories.

Value

A character vector containing unique code chunks in the specified directory/files.

Examples

```
## Not run:
readPackageRFiles()

# When running outside of a package
readDirectoryRFiles("R")

## End(Not run)
```

---

summary.multi_package_usage *Package Usage Summary*

Description

Package Usage Summary

Usage

```
## S3 method for class 'multi_package_usage'
summary(
  object,
  warn_percent_usage = 0.2,
  warn_number_usage = 3,
  ignore_low_usage_packages = character(),
  ...
)
```

Arguments

`object`	A `package_usage` data.frame
`warn_percent_usage`	Minimum percent of functions to be used within a dependent package. Default is `20%`
`warn_number_usage`	Minimum number of functions to be used within a dependent package. Default is `3`
`ignore_low_usage_packages`	A vector of packages to ignore the low usage of, usually when already aware of the low usage, but the dependent package is necessary for the project.
`...`	Not used

Details

Package usage must be below both thresholds for the warning to appear. With the defaults, if a package has fewer than 5 functions then only 1 function is required to prevent a warning message to appear.

`summary.package_usage` *Function Usage Summary*

Description

Function Usage Summary

Usage

```
## S3 method for class 'package_usage'
summary(object, warn_percent_usage = 0.2, warn_number_usage = 3, ...)
```

Arguments

`object`	A `package_usage` data.frame
`warn_percent_usage`	Minimum percent of functions to be used within a dependent package. Default is `20%`
`warn_number_usage`	Minimum number of functions to be used within a dependent package. Default is `3`
`...`	Not used

Details

Function usage must be below both thresholds for the warning to appear. With the defaults, if a package has fewer than 5 functions then only 1 function is required to prevent a warning message to appear.

Examples

```
## Not run:
package_use <- checkPackageUsage()
dependency_use <- checkPackageDependencyUse()
summary(dependency_use)

## End(Not run)
```

Index