Package 'risk.assessr'

July 10, 2025

Title Assessing Package Risk Metrics

Version 2.0.0

Description Provides a structured approach to assess the quality and trustworthiness of R packages (documentation, testing, popularity, dependencies), supporting informed decisions in production or research by highlighting strengths and potential risks in adoption or development.

License GPL (>= 2)

URL https://sanofi-public.github.io/risk.assessr/

BugReports https://github.com/Sanofi-Public/risk.assessr/issues

Depends R (>= 4.1.0)

Imports remotes, callr, checkmate, covr, desc, devtools, dplyr, fs, methods, purrr, rcmdcheck, rlang, xml2, stringr, tidyr, utils, curl, gh, jsonlite

Suggests forcats, glue, here, htmltools, kableExtra, knitr, magrittr, openxlsx, readr, roxygen2, testthat (>= 3.0.0), tidyselect, tools, rmarkdown, withr, mockery

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

Config/build/clean-inst-doc false

VignetteBuilder knitr

NeedsCompilation no

Author Edward Gillian [cre, aut] (ORCID: <https://orcid.org/0000-0003-2732-5107>), Hugo Bottois [aut] (ORCID: <https://orcid.org/0000-0003-4674-0875>), Paulin Charliquart [aut], Andre Couturier [aut], Sanofi [cph, fnd]

Maintainer Edward Gillian <edward.gillian-ext@sanofi.com>

Repository CRAN

Date/Publication 2025-07-10 15:20:02 UTC

Contents

`assess_pkg` *Assess package*

Description

assess package for risk metrics

Usage

```
assess_pkg(pkg_source_path, rcmdcheck_args, covr_timeout = Inf)
```

Arguments

pkg_source_path	- source path for install local
rcmdcheck_args	- arguments for R Cmd check - these come from setup_rcmdcheck_args
covr_timeout	- setting for covr time out

Value

list containing results - list containing metrics, covr, tm - trace matrix, and R CMD check

Examples

```
## Not run:
# set CRAN repo to enable running of reverse dependencies
r = getOption("repos")
r["CRAN"] = "http://cran.us.r-project.org"
old <- options(repos = r)

pkg_source_path <- system.file("test-data", "here-1.0.1.tar.gz",
   package = "risk.assessr")
pkg_name <- sub("\\.tar\\.gz$", "", basename(pkg_source_path))
modified_tar_file <- modify_description_file(pkg_source_path)

# Set up the package using the temporary file
install_list <- set_up_pkg(modified_tar_file)

# Extract information from the installation list
build_vignettes <- install_list$build_vignettes
package_installed <- install_list$package_installed
pkg_source_path <- install_list$pkg_source_path
rcmdcheck_args <- install_list$rcmdcheck_args

# check if the package needs to be installed locally
package_installed <- install_package_local(pkg_source_path)

# Check if the package was installed successfully
if (package_installed == TRUE) {
  # Assess the package
  assess_package <- assess_pkg(pkg_source_path, rcmdcheck_args)
  # Output the assessment result
} else {
  message("Package installation failed.")
}
options(old)

## End(Not run)
```

`assess_pkg_r_package` *Assess an R Package riskmetric with package name and version*

Description

This function use 'risk.assessr::assess_pkg' assessment function with only the package name and version

Usage

```
assess_pkg_r_package(package_name, version = NA, repos = NULL)
```

Arguments

`package_name`	A character string specifying the name of the package to assess.
`version`	A character string specifying the version of the package to assess. Default is 'NA', which assesses the latest version.
`repos`	A character string specifying the repo directly. Default is NULL, which uses the CRAN mirrors

Details

This function follows these steps:

1. Downloads the specified R package
2. Installs the downloaded package in a temporary location.
3. Runs the 'risk.assessr::assess_pkg' function to assess the package for risks and issues.
4. Returns the results of the assessment.

Value

The function returns a list of assessment results generated by the 'risk.assessr::assess_pkg' function. If the package cannot be downloaded or installed, an error message is returned.

Examples

```
## Not run:
# Example usage of assess_pkg_r_package
results <- assess_pkg_r_package("here", version = "1.0.1")
print(results)

## End(Not run)
```

`calc_overall_risk_score` *Calculate overall package risk scores*

Description

Calculate overall package risk scores

Usage

```
calc_overall_risk_score(data, default_weights = FALSE)
```

Arguments

`data`	risk metric data
`default_weights`	logical T/F for weights choice

Value

pkg_score

Examples

```
data <- list(
  pkg_name = "synapser",
  pkg_version = "0.2.1",
  pkg_source_path = "/tmp/RtmpNpDlUz/temp_file_1fe56774aacc/synapser",
  has_bug_reports_url = 1,
  has_examples = 1,
  has_maintainer = 1,
  size_codebase = 0.06702413,
  has_news = 0,
  has_source_control = 0,
  has_vignettes = 1,
  has_website = 1,
  news_current = 0,
  export_help = 1,
  export_calc = 0.586281,
  check = .7,
  covr = .1084,
  dep_score = .9706878,
  revdep_score = .1260338
  )
  overall_risk_score <-
    calc_overall_risk_score(data,
                            default_weights = TRUE)
```

calc_risk_profile *Calculate package risk profile*

Description

Calculate package risk profile

Usage

```
calc_risk_profile(risk_data)
```

Arguments

risk_data	overall risk score

Value

risk_profile

Examples

```
## Not run:
# Toy dataset
toy_data <- data.frame(score = c(0.1, 0.2, 0.3, 0.4, 0.8, 1.2))

calc_risk_profile(toy_data)

## End(Not run)
```

check_and_fetch_cran_package *Check and Fetch CRAN Package*

Description

This function checks if a package exists on CRAN and retrieves the corresponding package URL and version details. If a specific version is not provided, the latest version is used.

Usage

```
check_and_fetch_cran_package(package_name, package_version = NULL)
```

Arguments

package_name	A character string specifying the name of the package to check and fetch.
package_version	An optional character string specifying the version of the package to fetch. Defaults to 'NULL'.

Value

A list containing: - 'package_url': URL to download the package tarball. - 'last_version': Latest version available - 'version': The requested version of the package (or 'NULL' if not specified). - 'all_versions': A character vector of all available package versions - 'error': If the package or version is not found, an error message is included.

Examples

```
## Not run:
# Check and fetch a specific version of "ggplot2"
result <- check_and_fetch_cran_package("ggplot2", package_version = "3.3.5")
print(result)

## End(Not run)
```

---

check_cran_package *Check if a Package Exists on CRAN*

Description

This function checks if a given package is available on CRAN.

Usage

```
check_cran_package(package_name)
```

Arguments

package_name	A character string specifying the name of the package to check.

Value

A logical value: 'TRUE' if the package exists on CRAN, 'FALSE' otherwise.

Examples

```
## Not run:
# Check if the package "ggplot2" exists on CRAN
check_cran_package("ggplot2")

## End(Not run)
```

check_suggested_exp_funcs

Function to check suggested exported functions

Description

This function checks the exported functions of target package against the exported functions of Suggested dependencies to see if any Suggested packages should be in Imports in the DESCRIPTION file

Usage

```
check_suggested_exp_funcs(pkg_name, pkg_source_path, suggested_deps)
```

Arguments

pkg_name	- name of the target package
pkg_source_path	- source of the target package
suggested_deps	- dependencies in Suggests

Value

- data frame with results of Suggests check

contains_vignette_folder

Check for Vignette Folder and .Rmd Files in a .tar File

Description

This function checks if a given .tar file contains a 'vignettes' folder and if there is at least one .Rmd file within that folder. If both 'vignettes' and 'inst/doc' folders exist, the function will return FALSE.

Usage

```
contains_vignette_folder(tar_file)
```

Arguments

tar_file	A character string specifying the path to the .tar file to be checked.

Details

The function checks if the specified file exists and has a valid .tar extension using `utils::untar`. If the file is empty or any error occurs during the extraction, the function stops and returns an error message. If both 'vignettes' and 'inst/doc' folders exist, the function returns `FALSE`. If the 'vignettes' folder exists and contains at least one .Rmd file, the function returns `TRUE`. Otherwise, it returns `FALSE`.

Value

A logical value: `TRUE` if the 'vignettes' folder exists and contains at least one .Rmd file, and neither 'vignettes' nor 'inst/doc' folders are present, `FALSE` otherwise.

Examples

```
## Not run:
  tar_file <- system.file("test-data", "here-1.0.1.tar.gz",
   package = "risk.assessr")
  result <- contains_vignette_folder(tar_file)
  print(result)

## End(Not run)
```

---

create_weights_profile

Create weights profile

Description

This creates a specific weights profile for all risk metrics

Usage

```
create_weights_profile()
```

Value

- numeric vector with weights profile

Examples

```
create_weights_profile()
```

extract_version *Extract Package Version from File Path*

Description

This function extracts the version number from a package source file name based on the package name and expected file pattern.

Usage

```
extract_version(path, package_name)
```

Arguments

path	A character string specifying the file path or URL.
package_name	A character string specifying the name of the package.

Value

A character string representing the extracted version number, or 'NULL' if no match is found.

Examples

```
## Not run:
link <- "https://bioconductor.org/packages/3.14/bioc/src/contrib/GenomicRanges_1.42.0.tar.gz"
extract_version(link, "GenomicRanges")

## End(Not run)
```

fetch_bioconductor_package_info *Fetch Bioconductor Package Information*

Description

This function retrieves information about a specific Bioconductor package for a given Bioconductor version. It fetches the package details, such as version, source package URL, and archived versions if available.

Usage

```
fetch_bioconductor_package_info(bioconductor_version, package_name)
```

Arguments

bioconductor_version	A character string specifying the Bioconductor version (e.g., "3.14").
package_name	A character string specifying the name of the package.

Value

A list containing package details, including the latest version, package URL, source package link, and any archived versions if available. Returns 'FALSE' if the package does not exist or cannot be retrieved.

Examples

```
## Not run:
fetch_bioconductor_package_info("3.14", "GenomicRanges")


## End(Not run)
```

---

fetch_bioconductor_releases

Fetch Bioconductor Release Announcements

Description

This function retrieves the Bioconductor release announcements page and returns its HTML content for further processing.

Usage

```
fetch_bioconductor_releases()
```

Value

An XML document from bioconductor version page.

Examples

```
## Not run:
html_content <- fetch_bioconductor_releases()

## End(Not run)
```

generate_html_report *Generate HTML Report for Package Assessment*

Description

Generates an HTML report for the package assessment results using rmarkdown.

Usage

```
generate_html_report(assessment_results, output_dir)
```

Arguments

assessment_results	List containing the results from risk_assess_pkg function.
output_dir	Character string indicating the directory where the report will be saved.

Value

Path to the generated HTML report.

Examples

```
## Not run:
assessment_results <- risk_assess_pkg()
generate_html_report(assessment_results, output_dir = "path/to/save/report")

## End(Not run)
```

get_bioconductor_package_url *Retrieve Bioconductor Package URL*

Description

This function fetches the source package URL for a given Bioconductor package. If no version is specified, it retrieves the latest available version. Currently, this function is not able to fetch archived package version for a bioconductor version

Usage

```
get_bioconductor_package_url(
  package_name,
  package_version = NULL,
  release_data
)
```

Arguments

package_name	A character string specifying the name of the Bioconductor package.
package_version	(Optional) A character string specifying the package version. Defaults to 'NULL', which retrieves the latest version.
release_data	A list containing Bioconductor release information.

Value

A list containing the following elements:

url	The URL of the source package (if available).
version	The specified or latest available package version.
last_version	The last available version of the package.
all_versions	A vector of all discovered versions of the package.
bioconductor_version_package	The Bioconductor version associated with the package.
archived	A logical value indicating whether the package is archived.

Examples

```
## Not run:
release_data <- list(
  list(release = "3.12"),
  list(release = "3.13"),
  list(release = "3.14")
)

get_bioconductor_package_url("GenomicRanges", release_data = release_data)



## End(Not run)
```

---

get_cran_package_url *Get CRAN Package URL*

Description

This function constructs the CRAN package URL for a specified package and version.

Usage

```
get_cran_package_url(package_name, version, last_version, all_versions)
```

Arguments

package_name	A character string specifying the name of the package.
version	An optional character string specifying the version of the package.
last_version	A character string specifying the latest available version of the package.
all_versions	A character vector of all available versions of the package.

Value

A character string containing the URL to download the package tarball, or 'NULL' if the version is not found in the list of available versions.

Examples

```
url_result <- get_cran_package_url("dplyr", NULL, "1.0.10", c("1.0.0", "1.0.10"))
```

---

get_github_data *Fetch GitHub Repository Data*

Description

This function retrieves metadata about a GitHub repository, including creation date, stars, forks, and the number of recent commits within the last 30 days.

Usage

```
get_github_data(owner, repo)
```

Arguments

owner	A character string specifying the owner of the repository (e.g., GitHub username).
repo	A character string specifying the name of the repository. A github Personal Access Token (PAT) will be needed for some request or to help with the rate limit. Use Sys.setenv(GITHUB_TOKEN = "personal_access_token") or store your token in a .Renviron file (GitHub fine grained token are not yet covered by gh)

Details

If the 'owner' parameter is 'NA' or empty, the function returns an empty response object. Repository data is fetched using the GitHub API via the 'gh' package.

Value

A list containing: - ‘created_at‘: Creation date of the repository. - ‘stars‘: Number of stars the repository - ‘forks‘: Number of forks of the repository. - ‘date‘: acquisition date in the format "YYYY-MM-DD". - ‘recent_commits_count‘: count of commits in the last 30 days (from acquisition date).

Examples

```
## Not run:
# Fetch data for the "ggplot2" repository owned by "tidyverse"
result <- get_github_data("tidyverse", "ggplot2")
print(result)

## End(Not run)
```

---

get_internal_package_url

Get Internal Package URL

Description

This function retrieves the URL of an internal package on your internal Mirror, its latest version, and a list of all available versions.

Usage

```
get_internal_package_url(
  package_name,
  version = NULL,
  base_url = "http://cran.us.r-project.org",
  internal_path = "/src/contrib/"
)
```

Arguments

package_name	A character string specifying the name of the package.
version	An optional character string specifying the version of the package. Defaults to ‘NULL‘, in which case the latest version will be used.
base_url	a character string of internal package manager link
internal_path	a character string of internal package mirror link

Value

A list containing: - ‘url‘: A character string of the package URL (or ‘NULL‘ if not found). - ‘last_version‘: A character string of the latest version of the package. - ‘all_versions‘: A character vector of all available package versions.

Examples

```
## Not run:

# Retrieve a specific version URL of a package
result <- get_internal_package_url("internalpackage", version = "1.0.1")
print(result)

## End(Not run)
```

`get_package_download` *Get CRAN Package Download Count*

Description

Retrieves the download count for a given CRAN package from the CRAN logs API.

Usage

```
get_package_download(package_name, timeline = "grand-total")
```

Arguments

`package_name`	A character string specifying the package name.
`timeline`	A character string specifying the timeline ('last-month', or 'grand-total').

Value

An integer representing the total number of downloads.

Examples

```
## Not run:
total_download_result <- get_package_download('ggplot2')

month_download_result <- get_package_download('dplyr', 'last-month')

## End(Not run)
```

get_pkg_name *get package name for display*

Description

get package name for display

Usage

```
get_pkg_name(input_string)
```

Arguments

input_string	- string containing package name

Value

pkg_disp - package name for display

Examples

```
pkg_source_path <- "/home/user/R/test.package.0001_0.1.0.tar.gz"
pkg_disp_1 <- get_pkg_name(pkg_source_path)
print(pkg_disp_1)

pkg <- "TxDb.Dmelanogaster.UCSC.dm3.ensGene_3.2.2.tar.gz"
pkg_disp_2 <- get_pkg_name(pkg)
print(pkg_disp_2)
```

get_session_dependencies *Get Dependencies*

Description

This function extracts the version information of imported and suggested packages for a given package from the current R session.

Usage

```
get_session_dependencies(deps_list)
```

Arguments

deps_list	A data frame containing the dependency information of the package (provided by calc_dependencies function)

Value

A list with two elements:

`imports`	A named list of packages in the "Imports" section along with their corresponding versions
`suggests`	A named list of packages in the "Suggests" section along with their corresponding versions

Examples

```
deps_list <- data.frame(
  package = c("dplyr", "ggplot2", "testthat", "knitr"),
  type = c("Imports", "Imports", "Suggests", "Suggests")
)
get_session_dependencies(deps_list)
```

`get_suggested_exp_funcs`

Function to get suggested exported functions

Description

This function gets exported functions for all packages in the Suggests section of the target package's DESCRIPTION file

Usage

```
get_suggested_exp_funcs(data)
```

Arguments

`data`	- all packages listed in the DESCRIPTION file

Value

- data with package names and exported functions

get_versions *Get Package Versions*

Description

This function retrieves all available versions including last version from parse_html_version function'

Usage

```
get_versions(table, package_name)
```

Arguments

table	A list of parsed package data, where each element contains package details including package_version.
package_name	A character string specifying the name of the package to fetch versions for.

Value

A list containing: - 'all_versions': A character vector of all unique package versions. - 'last_version': A character string of the latest version fetched from the RStudio Package Manager, or 'NULL' if not available.

Examples

```
## Not run:
# Define the input table
table <- list(
  list(
    package_name = "here",
    package_version = "0.1",
    link = "here_0.1.tar.gz",
    date = "2017-05-28 08:13",
    size = "3.5K"
  ),
  list(
    package_name = "here",
    package_version = "1.0.0",
    link = "here_1.0.0.tar.gz",
    date = "2020-11-15 18:10",
    size = "32K"
  )
)

# Use the get_versions function
result <- get_versions(table, "here")

# Example output
```

```
print(result)

## End(Not run)
```

`install_package_local` *Install package locally*

Description

Install package locally

Usage

```
install_package_local(pkg_source_path)
```

Arguments

`pkg_source_path`	- source path for install local

Value

logical. Returns ‘TRUE‘ if the package was successfully installed, ‘FALSE‘ otherwise.

Examples

```
## Not run:
results <- install_package_local("pkg_source_path")
print(results)

## End(Not run)
```

`modify_description_file` *Modify the DESCRIPTION File in a R Package Tarball*

Description

This function recreate a ‘.tar.gz‘ R package file after modifying its ‘DESCRIPTION‘ file by appending Config/build/clean-inst-doc: false parameter.

Usage

```
modify_description_file(tar_file)
```

Arguments

`tar_file`	A string representing the path to the ‘.tar.gz‘ file that contains the R package.

Value

A string containing the path to the newly created modified '.tar.gz' file.

Examples

```
## Not run:
  modified_tar <- modify_description_file("path/to/mypackage.tar.gz")
  print(modified_tar)

## End(Not run)
```

`parse_bioconductor_releases`
Parse Bioconductor Release Announcements

Description

This function extracts Bioconductor release details such as version number, release date, number of software packages, and required R version from the release announcements HTML page.

Usage

```
parse_bioconductor_releases(html_content)
```

Arguments

`html_content`	The parsed HTML document from 'fetch_bioconductor_releases'.

Value

A list of lists containing Bioconductor release details: release version, date, number of software packages, and corresponding R version.

Examples

```
## Not run:
html_content <- fetch_bioconductor_releases()
release_data <- parse_bioconductor_releases(html_content)


## End(Not run)
```

parse_package_info *Parse Package Information from CRAN Archive*

Description

This function retrieves the package archive information from the CRAN Archive.

Usage

```
parse_package_info(name)
```

Arguments

name	A character string specifying the name of the package to fetch information for.

Value

A character string containing the raw HTML content of the package archive page, or ‘NULL‘ if the request fails or the package is not found.

Examples

```
## Not run:
# Fetch package archive information for "dplyr"
result <- parse_package_info("dplyr")

print(result)


## End(Not run)
```

risk_assess_pkg *Assess package - simplified*

Description

simplified input to assess package for risk metrics

Usage

```
risk_assess_pkg(path = NULL)
```

Arguments

path	(optional) path of locally stored package source code

Value

list containing results - list containing metrics, covr, tm - trace matrix, and R CMD check

Examples

```
## Not run:
risk_assess_package <- risk_assess_pkg()

OR

risk_assess_package <- risk_assess_pkg(path/to/package.tar.gz)

## End(Not run)
```

---

risk_assess_pkg_lock_files

Process lock files

Description

This function processes ‘renv.lock‘ and ‘pak.lock‘ files to produce risk metric data

Usage

```
risk_assess_pkg_lock_files(input_data)
```

Arguments

input_data	- path to a lock file

Value

assessment_results - nested list containing risk metric data

Examples

```
## Not run:
  input_data <- ("path/to/mypak.lock")
  pak_results <- risk_assess_pkg_lock_files(input_data)
  print(pak_results)

## End(Not run)
```

score_reverse_dependencies *Scoring method for number of reverse dependencies a package has*

Description

Score a package for the number of reverse dependencies it has; regularized Convert the number of reverse dependencies `length(x)` into a validation score [0,1]

$$1/(1 + exp(-0.5 * (sqrt(length(x)) + sqrt(20))))$$

Usage

```
score_reverse_dependencies(x)
```

Arguments

x	number of dependencies

Details

The scoring function is the classic logistic curve

$$1/(1 + exp(-k(x - x[0]))$$

with a square root scale for the number of reverse dependencies $x = sqrt(length(x))$, sigmoid midpoint is 20 reverse dependencies, ie. $x[0] = sqrt(15)$, and logistic growth rate of $k = 0.5$.

$$1/(1 + -0.5 * exp(sqrt(length(x)) - sqrt(20)))$$

Value

numeric value between `1` (high number of reverse dependencies) and `0` (low number of reverse dependencies)

set_up_pkg *Creates information on package installation*

Description

Creates information on package installation

Usage

```
set_up_pkg(dp, check_type = "1")
```

Arguments

dp	data path and name for the package.
check_type	basic R CMD check type - "1" CRAN R CMD check_type - "2"

Value

list with local package install

Examples

```
## Not run:
set_up_pkg(path/to/package, "mypackage")

## End(Not run)
```

---

update_results_doc_scores

update results doc_metrics

Description

This updates results list for documentation risk metrics

Usage

```
update_results_doc_scores(results, doc_scores)
```

Arguments

results	list with results
doc_scores	results from documentation risk metrics

Value

- list with updated risk result values

Index