Package 'rworkflows'

May 6, 2026

Type Package

Title Test, Document, Containerise, and Deploy R Packages

Version 1.0.12

Description Reproducibility is essential to the progress of research,
yet achieving it remains elusive even in computational fields.
Continuous Integration (CI) platforms offer a powerful way to launch automated workflows
to check and document code, but often require considerable time, effort,
and technical expertise to setup. We therefore developed the rworkflows suite
to make robust CI workflows easy and freely accessible to all R package developers.
rworkflows consists of 1) a CRAN/Bioconductor-compatible R package template,
2) an R package to quickly implement a standardised workflow, and
3) a centrally maintained GitHub Action.

URL https://github.com/neurogenomics/rworkflows,
https://CRAN.R-project.org/package=rworkflows

BugReports https://github.com/neurogenomics/rworkflows/issues

Encoding UTF-8

biocViews Software, WorkflowManagement

Depends R (>= 4.1)

Imports stats, here, yaml, utils, desc, badger, renv, methods,
BiocManager, data.table

Suggests markdown, rmarkdown, remotes, knitr, covr, testthat (>=
3.0.0), htmltools, jsonlite, BiocStyle, BiocPkgTools,
biocViews, reticulate, rvest, curl

VignetteBuilder knitr

License GPL-3

Config/testthat/edition 3

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Brian Schilder [aut] (ORCID: <https://orcid.org/0000-0001-5949-2191>),
Alan Murphy [aut, ctb] (ORCID: <https://orcid.org/0000-0002-2487-8753>),
Hiranyamaya (Hiru) Dash [ctb, cre] (ORCID:
<https://orcid.org/0009-0005-5514-505X>),
Nathan Skene [aut] (ORCID: <https://orcid.org/0000-0002-6807-3180>)

Maintainer Hiranyamaya (Hiru) Dash <hdash.work@gmail.com>

Repository CRAN

Date/Publication 2026-05-05 22:30:09 UTC

Contents

`biocpkgtools_db` *Static Bioc packages list*

Description

A static snapshot of all Bioconductor packages from biocPkgList. Last updated: Sept. 06 2023

Usage

```
data("biocpkgtools_db")
```

Format

An object of class `data.table` (inherits from `data.frame`) with 100 rows and 53 columns.

Source

```
as_ascii <- function(dt, cols=names(dt)){ cols <- cols[cols func <- function(v){ Encoding(v)
<- "latin1" iconv(v, "latin1", "UTF-8") } for(col in cols){ if(is.character(dt[[col]])){
dt[[col]] <- func(dt[[col]]) } } return(dt) } biocpkgtools_db <- get_description_repo_biocpkgtools(repo=
biocpkgtools_db <- as_ascii(biocpkgtools_db[seq(100)]) usethis::use_data(biocpkgtools_db,
overwrite = TRUE)
```

---

bioc_r_versions *Bioconductor / R versions*

Description

Get the respective version of R for a given version of Bioconductor.

Usage

```
bioc_r_versions(bioc_version = NULL, depth = NULL, return_opts = FALSE)
```

Arguments

`bioc_version`	Version of Bioc to return info for. Can be: **"devel"** Get the current development version of Bioc. **"release"** Get the current release version of Bioc. **<numeric>** A specific Bioc version number (e.g. `3.16`). **NULL** Return info for all Bioc versions.
`depth`	How many levels deep into the R version to include. For example, is the R version number is "4.2.0", the following depths would return: `depth=NULL`: "4.2.0" `depth=1`: "4" `depth=2`: "4.2" `depth=3`: "4.2.0"
`return_opts`	Return a character vector of all valid Bioc version names.

Value

Named list of Bioc/R versions

Examples

```
if (requireNamespace("curl", quietly = TRUE) && curl::has_internet()) {
  ver <- bioc_r_versions(bioc_version="devel")
} else {
  message("No internet connection available, skipping example.")
}
```

conda_export *Conda export*

Description

Get a list of installed packages within a conda environment. Generates a requirements.txt file.

Usage

```
conda_export(
  name,
  save_path = tempfile(fileext = "_requirements.txt"),
  preview = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

name	Name of conda environment.
save_path	Path to save the requirements file. If the file ends with .yml or .yaml, a conda-style yaml file will be generated. If the file ends with requirements.txt, a pip-style requirements.txt file will be generated.
preview	Print the requirements file to the R console.
verbose	Print messages.
...	Optional arguments, reserved for future expansion.

Value

Path to requirements file.

Source

https://stackoverflow.com/a/55687210

Examples

```
## Not run:
conda_export()

## End(Not run)
```

construct_authors *Construct authors*

Description

Helper function to construct an author list for a *DESCRIPTION* file. Returns a template when `authors` is not provided (default).

Usage

```
construct_authors(
  authors = NULL,
 template = c(utils::person(given = "yourGivenName", family = "yourFamilyName", role =
  c("cre"), email = "yourEmail@email.com", comment = c(ORCID = "0000-0001-2345-6789")))
)
```

Arguments

`authors`	A list of authors who contributed to your R package, each provided as objects of class person. By default, if an `Authors` field already exists in the *DESCRIPTION* file, the original values are kept. Otherwise, a template person list is created using the construct_authors.
`template`	Default value to return when `authors=NULL`.

Value

Named list in person format.

Examples

```
authors <- construct_authors()
```

construct_conda_yml *Construct a conda yaml*

Description

Construct a yaml file to be used for building a given conda environment.

Usage

```
construct_conda_yml(
  name = "test",
  channels = list("conda-forge", "nodefaults"),
  dependencies = list(),
  pip = NULL,
  save_path = here::here(paste0(name, "_conda.yml")),
  return_path = TRUE,
  preview = FALSE,
  verbose = TRUE
)
```

Arguments

name	Name of conda env.
channels	conda channels to use.
dependencies	Packages to install via conda.
pip	Packages to install via pip.
save_path	Path to save the yaml file to.
return_path	Return the path to the saved *yaml* workflow file (default: TRUE), or return the *yaml* object directly.
preview	Print the yaml file to the R console.
verbose	Print messages.

Value

description

Path or yaml object.

Examples

```
yml <- construct_conda_yml(name="myenv",
                           dependencies=c("anndata","scanpy"),
                           return_path = FALSE,
                           ## Writing to temp only for example
                           save_path=tempfile(fileext="myenv_conda.yml"))
```

---

construct_cont *Construct containers list*

Description

Construct containers list

Usage

```
construct_cont(
  default_registry = c("ghcr.io/", "docker.io/", "mcr.microsoft.com/"),
  default_cont = "bioconductor/bioconductor_docker",
  default_tag = "devel",
  cont = list(paste(default_cont, default_tag, sep = ":"), NULL, NULL),
  versions_explicit = FALSE,
  run_check_cont = FALSE,
  verbose = TRUE
)
```

Arguments

`default_registry`
: The default container registry to use. Options include:

  "ghcr.io/" : GitHub Container Registry

  "docker.io/" : DockerHub

  "mcr.microsoft.com/" : Microsoft Container Registry

`default_cont`
: The DockerHub container to default to. Used when it's detected that only the tag has been given in one or more `cont` entry.

`default_tag`
: The DockerHub container tag to default to.

`cont`
: Which Docker container to use on each OS (`NULL` means no container will be used for that OS). See here for a list of all official Bioconductor Docker container versions.

`versions_explicit`
: Specify R/Bioc versions explicitly (e.g. `r: 4.2.0`, `bioc: 3.16`) as opposed to flexibly (e.g. `r: "latest"`, `bioc: "release"`).

`run_check_cont`
: Check whether the requested container repo (and the tag, if specified) exist using check_cont.

`verbose`
: Print messages.

Value

Named list of containers

Examples

```
cont <- construct_cont()
```

`construct_runners` *Construct runners*

Description

Construct runner configurations across multiple Operating Systems (OS) for GitHub Actions workflow.

Usage

```
construct_runners(
  os = c("ubuntu-latest", "macOS-latest", "windows-latest"),
  bioc = list("devel", "release", "release"),
  r = list("auto", "auto", "auto"),
  python_version = list(NULL, NULL, NULL),
  versions_explicit = FALSE,
  run_check_cont = FALSE,
 cont = construct_cont(default_tag = bioc[[1]], run_check_cont = run_check_cont),
  rspm = list(NULL, NULL, NULL),
  verbose = TRUE
)
```

Arguments

`os`	Which OS to launch GitHub Actions on. See here for all options.
`bioc`	Which Bioconductor version to use on each OS. See bioc_r_versions documentation for all options.
`r`	Which R version to use on each OS.
`python_version`	Which python version to use on each OS (e.g. "3.10", "3.7.5", or "3.x"). (`NULL` means python will not be installed on that OS). See here or `rworkflows:::gha_python_versions()` for all available python versions. See here for details on the `actions/setup-miniconda` action. See here for details on the `actions/setup-python` action.
`versions_explicit`	Specify R/Bioc versions explicitly (e.g. `r: 4.2.0`, `bioc: 3.16`) as opposed to flexibly (e.g. `r: "latest"`, `bioc: "release"`).
`run_check_cont`	Check whether the requested container repo (and the tag, if specified) exist using check_cont.
`cont`	Which Docker container to use on each OS (`NULL` means no container will be used for that OS). See here for a list of all official Bioconductor Docker container versions.
`rspm`	Which R repository manager to use on each OS (`NULL` means the default will be used for that OS).
`verbose`	Print messages.

Value

Named list of configurations for each runner OS.

Examples

```
if (requireNamespace("curl", quietly = TRUE) && curl::has_internet()) {
  runners <- construct_runners()
} else {
  message("No internet connection available, skipping example.")
}
```

`dt_to_desc` *data.table to desc*

Description

Convert data.table containing the parsed *DESCROPTION* file data and convert each of them to to desc format.

Usage

```
dt_to_desc(db, refs = NULL, verbose = TRUE)
```

Arguments

`db`	A data.table where each row is a different R package and each column is a field from the *DESCROPTION* file.
`refs`	Reference for one or more GitHub repository in owner/repo format (e.g.`"neurogenomics/rworkflows"`), or an R package name (e.g. `"rworkflows"`).
`verbose`	Print messages.

Value

A named list of desc objects.

Examples

```
#### Updated data ####
# db <- BiocPkgTools::biocPkgList()
#### Static data ####
 db <- rworkflows::biocpkgtools_db
 dl <- dt_to_desc(db=db, refs="ABSSeq")
```

`fill_description` *Fill* DESCRIPTION

Description

Fill out a *DESCRIPTION* file, such as (but not limited to) the one provided by the templateR R package template. For any given field, set its corresponding argument as follows to get certain behaviour:

`NULL`: Keeps the current value.

`NA`: Removes the field from the *DESCRIPTION* file entirely.

Usage

```
fill_description(
  path = here::here("DESCRIPTION"),
  package,
  title,
  description,
  github_owner = NULL,
  github_repo = package,
  authors = construct_authors(authors = NULL),
 depends = paste0("R ", "(>= ", bioc_r_versions(bioc_version = "devel", depth = 2)$r,
    ")"),
  imports = infer_deps(path = path, which = "Imports", add_newlines = TRUE),
  suggests = infer_deps(path = path, which = "Suggests", add_newlines = TRUE),
  remotes = NULL,
  version = NULL,
  license = NULL,
  encoding = NULL,
  vignettebuilder = NULL,
  biocviews = infer_biocviews(pkgdir = dirname(path), add_newlines = TRUE),
  url = paste0("https://github.com/", github_owner, "/", github_repo),
  bugreports = paste0(url, "/issues"),
  save_path = path,
  verbose = TRUE,
  fields = list()
)
```

Arguments

`path`	Path to the *DESCRIPTION* file.
`package`	The name of your R package.
`title`	The title of your R package.
`description`	The description of your R package.

github_owner	The owner of your R package's GitHub repository. Can be inferred from the URL field in the *DESCRIPTION* file if this has already been filled out.
github_repo	The name of your R package's GitHub repository.
authors	A list of authors who contributed to your R package, each provided as objects of class person. By default, if an Authors field already exists in the *DESCRIPTION* file, the original values are kept. Otherwise, a template person list is created using the construct_authors.
depends	R package Depends. Defaults to the version of R that the current development version of Bioconductor depends on.
imports	R package Imports. These dependencies will be automatically installed with your R package.
suggests	R package Suggests. These dependencies will NOT be automatically installed with your R package, unless otherwise specified by users during installation
remotes	R package Remotes
version	The current version of your R package (e.g 0.99.0).
license	R package license. See here for guidance.
encoding	R package Encoding.
vignettebuilder	R package VignetteBuilder.
biocviews	Standardised biocViews terms used to describe your package. Defaults to automatically recommending terms using the infer_biocviews function. Note that non-Bioconductor packages (e.g. CRAN) can also use this field.
url	URL where your R package is distributed from (e.g. GitHub repository, Bioconductor page, and/or CRAN page). Can be a single character string or a character vector.
bugreports	A URL where users of your package should go if they encounter bugs or have feature requests.
save_path	Path to save the updated *DESCRIPTION* file to. Defaults to overwriting the input file (path). Set to NULL if you wish to only return the description object without writing to any file.
verbose	Print messages.
fields	A named list of additional fields to fill the *DESCRIPTION* file with: e.g. list(RoxygenNote=7.2.3)

Value

An object of class description.

Examples

```
if (requireNamespace("curl", quietly = TRUE) && curl::has_internet()) {
  #### Get example DESCRIPTION file ####
  url <- "https://github.com/neurogenomics/templateR/raw/master/DESCRIPTION"
  path <- tempfile(fileext = "DESCRIPTION")
  utils::download.file(url,path)
```

```
  #### Fill out DESCRIPTION file ####
  d <- fill_description(
    path = path,
    package = "MyPackageName",
    title = "This Package Does Awesome Stuff",
    description = paste(
      "MyPackageName does several awesome things.",
      "Describe thing1.",
      "Describe thing2.",
      "Describe thing3."
    ),
    github_owner = "OwnerName",
    biocviews = c("Genetics", "SystemsBiology"))
} else {
  message("No internet connection available, skipping example.")
}
```

`get_description` *Get DESCRIPTION*

Description

The Liam Neeson of *DESCRIPTION* file functions.

1. I will look for you,
2. I will find you,
3. —and I will import you into a neatly parsed R object.

Uses a variety of alternative methods, including searching locally and on GitHub (whenever possible). Prioritises the fastest methods that do not involve downloading files first.

Usage

```
get_description(
  refs = NULL,
  paths = here::here("DESCRIPTION"),
  db = NULL,
  cache_dir = tempdir(),
  force_new = FALSE,
  use_wd = TRUE,
  use_repos = FALSE,
  repo = c("BioCsoft", "BioCann", "BioCexp", "BioCworkflows", "CRAN"),
  verbose = TRUE
)
```

Arguments

refs	Reference for one or more GitHub repository in owner/repo format (e.g."neurogenomics/rworkflows"), or an R package name (e.g. "rworkflows").
paths	Paths to *DESCRIPTION* file(s) R package(s).
db	A data.table of R package metadata generated by biocPkgList.
cache_dir	Directory where to cache downloaded files.
force_new	Ignore cached files and re-download them instead.
use_wd	Search the local working directory (and the one above it) for *DESCRIPTION* files.
use_repos	Use R standard R package repositories like CRAN and Bioc to find *DESCRIPTION* files.
repo	character(1) The requested Bioconductor repository. The default is to pull from the "BioCsoft" repository. Possible repositories include "BioCsoft", "BioCexp", "BioCworkflows", "BioCann", and "CRAN". Note that not all repos are available for all versions, particularly older versions.
verbose	Print messages.

Value

A named list of packageDescription objects.

Examples

```
if (requireNamespace("curl", quietly = TRUE) && curl::has_internet()) {
  d <- get_description(refs="neurogenomics/rworkflows")
} else {
  message("No internet connection available, skipping example.")
}
```

---

get_hex *Get hex*

Description

Get the URL of a hex sticker for a given R package (if one exists).

Usage

```
get_hex(
  refs = NULL,
  paths = here::here("DESCRIPTION"),
  hex_path = "inst/hex/hex.png",
  branch = c("master", "main", "dev"),
  hex_height = 300,
  check_url = TRUE,
```

```
  add_html = TRUE,
  verbose = TRUE
)
```

Arguments

refs	Reference for one or more GitHub repository in owner/repo format (e.g."neurogenomics/rworkflows"), or an R package name (e.g. "rworkflows").
paths	Paths to *DESCRIPTION* file(s) R package(s).
hex_path	Path to hex sticker file.
branch	Name of the GitHub repository branch to use.
hex_height	Height of the hex sticker in pixels (when add_hex=TRUE).
check_url	Check whether the URL actually exists.
add_html	Wrap the URL in an html "img" tag and set its height with hex_height.
verbose	Print messages.

Value

URL

Examples

```
if (requireNamespace("curl", quietly = TRUE) && curl::has_internet()) {
  hex_url <- get_hex(refs=c("neurogenomics/rworkflows",
                            "neurogenomics/echolocatoR"))
} else {
  message("No internet connection available, skipping example.")
}
```

---

infer_biocviews *Infer biocViews*

Description

Infer the best terms to fill the biocViews field in your *DESCRIPTION* file based on the code within your R package. By default, also includes any biocViews that are already present in the *DESCRIPTION* file. Please see the Bioconductor website for more details.

Usage

```
infer_biocviews(
  pkgdir = here::here(),
  branch = c("Software", "AnnotationData", "ExperimentData")[1],
  type = c("recommended", "current", "remove"),
  keep_current = TRUE,
  include_branch = TRUE,
```

```
  biocviews = NULL,
  add_newlines = FALSE,
  verbose = TRUE
)
```

Arguments

pkgdir	The path of the package Directory.
branch	The branch which your package will belong to. It can be either 'Software', 'AnnotationData' or 'ExperimentData'.
type	Which element of the recommendBiocViews results list to return. If a vector is supplied, only the first value will be used.
keep_current	Keep any biocViews terms that are already included in the *DESCRIPTION* file.
include_branch	Whether to include the branch argument as one of the returned biocViews.
biocviews	User-supplied biocViews terms to include in addition to the automated recommendations.
add_newlines	Prefix each package name with a newline character and two spaces. This is useful for formatting *DESCRIPTION* files.
verbose	Print messages.

Value

A character vector of biocviews.

Examples

```
## Don't run simply bc biocViews::recommendBiocViews is unable
## to find the DESCRIPTION file when running examples.
## Not run:
biocviews <- infer_biocviews()

## End(Not run)
```

---

infer_deps *Infer dependencies*

Description

Infers the R packages that your R package depends on.

Usage

```
infer_deps(
  path = here::here("DESCRIPTION"),
  which = c("Imports", "Suggests"),
  imports_thresh = 2,
  imports = NULL,
 suggests = c("testthat", "rmarkdown", "markdown", "knitr", "remotes", "knitr", "covr"),
  errors = c("reported", "fatal", "ignored"),
  dev = FALSE,
  progress = TRUE,
  add_newlines = FALSE
)
```

Arguments

path	The path to a .R, .Rmd, .qmd, DESCRIPTION, a directory containing such files, or an R function. The default uses all files found within the current working directory and its children.
which	Which types of dependencies to return.
imports_thresh	The minimum number of times that a package has to be called within your package to assign it as an Import. If is called less times than this threshold, it will instead be assigned as a Suggest, which means it will not be installed by default.
imports	R packages that are exempt from the suggests_thresh rule and are instead automatically assigned as Imports.
suggests	R packages that are exempt from the suggests_thresh rule and are instead automatically assigned as Suggests.
errors	How should errors that occur during dependency enumeration be handled? • "reported" (the default): errors are reported to the user, but otherwise ignored. • "fatal": errors are fatal and stop execution. • "ignored": errors are ignored and not reported to the user.
dev	Boolean; include development dependencies? These packages are typically required when developing the project, but not when running it (i.e. you want them installed when humans are working on the project but not when computers are deploying it). Development dependencies include packages listed in the Suggests field of a DESCRIPTION found in the project root, and roxygen2 or devtools if their use is implied by other project metadata. They also include packages used in ~/.Rprofile if config$user.profile() is TRUE.
progress	Boolean; report progress output while enumerating dependencies?
add_newlines	Prefix each package name with a newline character and two spaces. This is useful for formatting *DESCRIPTION* files.

Value

A character vector of R package names.

Examples

```
if (requireNamespace("curl", quietly = TRUE) && curl::has_internet()) {
  #### Get example DESCRIPTION file ####
  url <- "https://github.com/neurogenomics/templateR/raw/master/DESCRIPTION"
  path <- tempfile(fileext = "DESCRIPTION")
  utils::download.file(url,path)

  deps <- infer_deps(path = path)
} else {
  message("No internet connection available, skipping example.")
}
```

`infer_docker_org` *Infer Docker registry organisation name*

Description

Infer Docker registry organisation name from DESCRIPTION file.

Usage

```
infer_docker_org(docker_org = NULL, docker_registry, verbose = TRUE, ...)
```

Arguments

`docker_org`	Docker registry organization name. Can simply be your registry username instead. If `NULL`, `docker_org` will be inferred as the R package's GitHub owner.
`docker_registry`	Docker container registry to push to. Options include: **"ghcr.io" :** GitHub Container Registry **"docker.io" :** DockerHub
`verbose`	Print messages.
`...`	Arguments passed on to `get_description` `refs` Reference for one or more GitHub repository in owner/repo format (e.g.`"neurogenomics/rworkfl` or an R package name (e.g. `"rworkflows"`). `paths` Paths to *DESCRIPTION* file(s) R package(s). `cache_dir` Directory where to cache downloaded files. `force_new` Ignore cached files and re-download them instead. `use_wd` Search the local working directory (and the one above it) for *DESCRIPTION* files. `use_repos` Use R standard R package repositories like CRAN and Bioc to find *DESCRIPTION* files. `db` A data.table of R package metadata generated by biocPkgList. `repo character(1)` The requested Bioconductor repository. The default is to pull from the "BioCsoft" repository. Possible repositories include "BioCsoft", "BioCexp", "BioCworkflows", "BioCann", and "CRAN". Note that not all repos are available for all versions, particularly older versions.

Value

Docker registry organisation name.

Examples

```
infer_docker_org(docker_org="myorg", docker_registry="ghcr.io")
```

---

is_gha *Is GitHub Action*

Description

Tests whether a function is currently being run within a GitHub Actions workflow or not.

Usage

```
is_gha(var = "GITHUB_ACTION", verbose = TRUE)
```

Arguments

var	Environmental variable to check.
verbose	Print messages.

Value

Boolean

Source

GitHub Actions docs

Examples

```
is_gha()
```

use_badges *Use badges*

Description

Create one or more badges showing the status of your R package. Uses the package **badger**.

Usage

```
use_badges(
  ref = NULL,
  add_hex = TRUE,
  add_actions = "rworkflows",
  add_doi = NULL,
  add_lifecycle = FALSE,
  add_github_version = TRUE,
  add_commit = TRUE,
  add_code_size = TRUE,
  add_license = TRUE,
  add_authors = TRUE,
  add_codecov = TRUE,
  add_codecov_graphs = "icicle",
  add_bioc_release = FALSE,
  add_bioc_download_month = FALSE,
  add_bioc_download_total = FALSE,
  add_bioc_download_rank = FALSE,
  add_cran_release = FALSE,
  add_cran_checks = FALSE,
  add_cran_download_month = FALSE,
  add_cran_download_total = FALSE,
  branch = "master",
  as_list = FALSE,
  sep = "\n",
  hex_height = 300,
  codecov_graph_width = 200,
  colors = list(github = "black", bioc = "green", cran = "black", default = "blue",
    lifecycle = NULL),
  verbose = TRUE
)
```

Arguments

ref	Reference for a GitHub repository. If NULL (the default), the reference is determined by the URL field in the DESCRIPTION file.
add_hex	Add a hex sticker. If add_hex=TRUE, will assume the sticker is located at the following relative path: "inst/hex/hex.png". If add_hex is a character string, this will instead be used as the relative hex path (e.g. "/images/mysticker.png").

`add_actions` The name of one or more GitHub Actions to show the status for with badge_github_actions (e.g. c("rworkflows","rworkflows_static")).

`add_doi` Add the DOI of a given package or publication associated with the package using badge_doi. Must be provided as a character string, e.g.: `"10.1111/2041-210X.12628"`

`add_lifecycle` Add package lifecycle stage. If not `FALSE`, must be a character string indicating one of the following valid lifecycle stage:

- "stable"
- "deprecated"
- "superseded"
- "experimental"

See lifecycle.r-lib.org for further details.

`add_github_version` Add package version with badge_github_version.

`add_commit` Add the last GitHub repo commit date with badge_last_commit.

`add_code_size` Add code size with badge_code_size.

`add_license` Add license info with badge_license.

`add_authors` Add author names inferred from the `DESCRIPTION` file.

`add_codecov` Add Codecov status with badge_codecov. See the Codecov site for more information about these badges.

`add_codecov_graphs` Add Codecov graphs visualising results of code coverage tests. Options include:

- "sunburst"
- "tree"
- "icicle"

See the Codecov site for more information about each plot type.

`add_bioc_release` Add Bioc release version with badge_bioc_release.

`add_bioc_download_month` Add the number of Bioc downloads last month badge_bioc_download.

`add_bioc_download_total` Add the number of Bioc downloads total badge_bioc_download.

`add_bioc_download_rank` Add the download rank of the package on Bioc badge_bioc_download_rank.

`add_cran_release` Add Bioc release version with badge_cran_release.

`add_cran_checks` Add whether package is passing all checks on CRAN with badge_cran_checks.

`add_cran_download_month` Add the number of CRAN downloads last month badge_cran_download.

`add_cran_download_total` Add the number of CRAN downloads total badge_cran_download.

`branch` Name of the GitHub repository branch to use.

as_list	Return the header as a named list (TRUE), or a collapsed text string (default: FALSE).
sep	Character to separate each item in the list with using paste.
hex_height	Height of the hex sticker in pixels (when add_hex=TRUE).
codecov_graph_width	Width of each Codecov graph in pixels (when add_codecov_graph!=FALSE).
colors	Colors to assign to each group of badges (when possible).
verbose	Print messages.

Value

A named list of selected badges in markdown format.

Examples

```
if (requireNamespace("curl", quietly = TRUE) && curl::has_internet()) {
  badges <- rworkflows::use_badges(ref = "neurogenomics/rworkflows")
} else {
  message("No internet connection available, skipping example.")
}
```

---

use_codespace *Use Codespace*

Description

Generate a dev container config file to set up a GitHub Codespace.

Usage

```
use_codespace(
  template = "devcontainer.json",
  image = "ghcr.io/neurogenomics/rworkflows:dev",
  features = list(`ghcr.io/devcontainers/features/conda:1` = list()),
  customizations = list(vscode = list(settings = list(), extensions =
    list("reditorsupport.r", "visualstudioexptteam.vscodeintellicode",
    "ionutvmi.path-autocomplete"))),
  save_dir = here::here(".devcontainer"),
  path = file.path(save_dir, template),
  force_new = FALSE,
  show = FALSE,
  verbose = TRUE
)
```

Arguments

`template`	Dev container config template to use.
`image`	Base Docker image to use for the Codespace.
`features`	Named list of features to add to the Codespace. See here for details.
`customizations`	Named list of customizations to add to the Codespace. See here for details.
`save_dir`	Directory to save the file to.
`path`	Path to the file.
`force_new`	If the file already exists, overwrite it (default: `FALSE`).
`show`	Print the contents of the file in the R console.
`verbose`	Print messages.

Value

Path to dev container config file.

Examples

```
path <- use_codespace(save_dir=tempdir())
```

`use_dockerfile` *Use Dockerfile*

Description

Creates a Docker file to be used with the GitHub Actions (GHA) workflows distributed by **rworkflows**.

Usage

```
use_dockerfile(
  save_dir = here::here(),
  path = file.path(save_dir, "Dockerfile"),
  base_image = construct_cont()[[1]],
  force_new = FALSE,
  show = FALSE,
  verbose = TRUE
)
```

Arguments

`save_dir`	Directory to save the Docker file to.
`path`	Path to the Docker file.
`base_image`	Base Docker image to use.
`force_new`	If a Docker file already exists, overwrite it (default: `FALSE`).
`show`	Print the contents of the Docker file in the R console.
`verbose`	Print messages.

Value

Path to Docker file.

Examples

```
path <- use_dockerfile(save_dir=tempdir())
```

---

use_issue_template *Use Issue Template*

Description

Creates one or more Issue Templates to be used in a GitHub repository.

Usage

```
use_issue_template(
  templates = c("bug_report.yml", "feature_request.yml", "config.yml"),
  save_dir = here::here(".github", "ISSUE_TEMPLATE"),
  path = file.path(save_dir, templates),
  force_new = FALSE,
  show = FALSE,
  verbose = TRUE
)
```

Arguments

templates	The names of templates to be used.
save_dir	Directory to save the Docker file to.
path	Path to the Docker file.
force_new	If a Docker file already exists, overwrite it (default: FALSE).
show	Print the contents of the Docker file in the R console.
verbose	Print messages.

Value

Path to Issue Templates.

Examples

```
path <- use_issue_template(save_dir=tempdir())
```

use_readme *Use README*

Description

Creates an rmarkdown README file that autofills using metadata from the R package *DESCRIPTION* file.

Usage

```
use_readme(
  save_dir = here::here(),
  path = file.path(save_dir, "README.Rmd"),
  force_new = FALSE,
  show = FALSE,
  verbose = TRUE
)
```

Arguments

save_dir	Directory to save the file to.
path	Path to the file.
force_new	If the file already exists, overwrite it (default: FALSE).
show	Print the contents of the file in the R console.
verbose	Print messages.

Value

Path to README file.

Examples

```
## use default save_dir in practice
path <- use_readme(save_dir = tempdir())
```

use_vignette_docker *Use vignette: Docker*

Description

Creates a vignette rmarkdown file demonstrates how to create a Docker/Singularity image from a container stored in Dockerhub.

Usage

```
use_vignette_docker(
  package = names(get_description()),
  docker_org = NULL,
  docker_registry = "ghcr.io",
 cont = construct_cont(cont = paste(docker_org, package, sep = "/"), default_registry =
    docker_registry)[[1]],
  title = "Docker/Singularity Containers",
  vignette_index_entry = "docker",
  save_dir = here::here(),
  path = file.path(save_dir, "vignettes", "docker.Rmd"),
 output = list(`BiocStyle::html_document` = list(md_extensions = "-autolink_bare_uris")),
  port_in = 8787,
  port_out = 8900,
  force_new = FALSE,
  show = FALSE,
  verbose = TRUE
)
```

Arguments

package	R package name.
docker_org	Docker registry organization name. Can simply be your registry username instead. If NULL, docker_org will be inferred as the R package's GitHub owner.
docker_registry	Docker container registry to push to. Options include: **"ghcr.io" :** GitHub Container Registry **"docker.io" :** DockerHub
cont	Which Docker container to use on each OS (NULL means no container will be used for that OS). See here for a list of all official Bioconductor Docker container versions.
title	Title of vignette.
vignette_index_entry	Index entry of the vignette, which is used when creating the navigation bar in the **pkgdown** site.
save_dir	Directory to save the file to.
path	Path to the file.
output	Vignette output style. Defaults to html_document.
port_in	Port number to route into the docker container. See the Docker docs for further details.
port_out	Port number to route out of docker container. See the Docker docs for further details.
force_new	If the file already exists, overwrite it (default: FALSE).
show	Print the contents of the file in the R console.
verbose	Print messages.

Value

Path to vignette file.

Examples

```
path <- use_vignette_docker(package = "mypackage",
                            docker_org = "neurogenomics",
                            ## use default save_dir in practice
                            save_dir = tempdir())
```

---

use_vignette_getstarted

Use vignette: Get started

Description

Creates a "Get started" rmarkdown vignette file.

Usage

```
use_vignette_getstarted(
  package = names(get_description()),
  title = "Get started",
  vignette_index_entry = package,
  save_dir = here::here(),
  path = file.path(save_dir, "vignettes", paste0(package, ".Rmd")),
  output = "BiocStyle::html_document",
  force_new = FALSE,
  show = FALSE,
  verbose = TRUE
)
```

Arguments

package	R package name.
title	Title of vignette.
vignette_index_entry	Index entry of the vignette, which is used when creating the navigation bar in the **pkgdown** site.
save_dir	Directory to save the file to.
path	Path to the file.
output	Vignette output style. Defaults to html_document.
force_new	If the file already exists, overwrite it (default: FALSE).
show	Print the contents of the file in the R console.
verbose	Print messages.

Value

Path to vignette file.

Examples

```
path <- use_vignette_getstarted(package = "mypackage",
                                ## use default save_dir in practice
                                save_dir = tempdir())
```

`use_workflow` *Use GitHub Actions workflow*

Description

Create workflow that calls an rworkflows GitHub Actions (GHA)

Usage

```
use_workflow(
  template = "rworkflows",
  name = template,
  tag = "@master",
  on = c("push", "pull_request"),
  branches = c("master", "main", "devel", "RELEASE_**"),
  runners = construct_runners(),
  github_token = "${{ secrets.GITHUB_TOKEN }}",
  cache_version = "cache-v1",
  enable_act = FALSE,
  ncpus = 2,
  timeout = 2000,
  force_install = FALSE,
  run_telemetry = TRUE,
  free_diskspace = FALSE,
  run_bioccheck = FALSE,
  run_rcmdcheck = TRUE,
  as_cran = TRUE,
  run_vignettes = TRUE,
  has_testthat = TRUE,
  has_runit = FALSE,
  run_lintr = TRUE,
  run_spelling = TRUE,
  run_covr = TRUE,
  codecov_token = "${{ secrets.CODECOV_TOKEN }}",
  has_latex = FALSE,
  tinytex_installer = "TinyTeX-1",
  tinytex_version = NULL,
  pandoc_version = "2.19",
```

```
  run_pkgdown = TRUE,
  run_docker = FALSE,
  docker_registry = "ghcr.io",
  docker_user = NULL,
  docker_org = docker_user,
  docker_token = "${{ secrets.DOCKER_TOKEN }}",
  miniforge_variant = FALSE,
  miniforge_version = NULL,
  activate_environment = "test",
  environment_file = NULL,
  channels = NULL,
  save_dir = here::here(".github", "workflows"),
  return_path = TRUE,
  force_new = FALSE,
  preview = FALSE,
  verbose = TRUE
)
```

Arguments

`template`	Workflow template name. **"rworkflows"** A short workflow script that calls the GitHub action from the GitHub Marketplace. The action is continually updated so users do not need to worry about maintaining it. **"rworkflows_static"** A longer workflow scripts that explicitly copies all steps from the **rworkflows** action into a static file. Users may need to update this file themselves over time, though this does allow for a fully customisable workflow. Optionally, you can include the suffix ":<branch>" to specify which branch you would like to download the "action.yml" file from to create the static workflow template.
`name`	An arbitrary name to call the workflow.
`tag`	Which version of the `rworkflows` action to use. Can be a branch name on the GitHub repository (e.g. "\@master"), or a Release Tag (e.g. "\@v1").
`on`	GitHub trigger conditions.
`branches`	GitHub trigger branches.
`runners`	Runner configurations for multiple Operating Systems (OS), including R versions, Bioc versions, and container sources. Can use the construct_runners functions to assist in constructing customized runners configurations.
`github_token`	GitHub authentication token with permissions to push to the R package's GitHub repository. Also used to bypass GitHub download limits. By default, uses {{ secrets.GITHUB_TOKEN }} which is automatically set up by GitHub. However users can also choose to pass a custom GitHub secret variable (e.g. {{ secrets.PAT_GITHUB }}) which allows access to private repositories. Read here for more details.
`cache_version`	Name of the cache sudirectory to be used when reinstalling software in GHA.

`enable_act` Whether to add extra lines to the yaml to enable local workflow checking with act.

`ncpus` Number of CPUs to use for R package installation. Higher values can speed up the dependency installation process but may result in spurious errors. (default = 2)

`timeout` The maximum time to wait for long R processes like dependency installations, downloads, and code checks. (default = `2000`)

`force_install` Whether to force install packages. If `true`, all packages will be reinstalled, bypassing the cache.

`run_telemetry` Whether to run the workflow telemetry action: https://github.com/catchpoint/workflow-telemetry-action

`free_diskspace` Whether to free up additional disk space by deleting non-essential software.

`run_bioccheck` Run Bioconductor checks using `BiocCheck::BiocCheck()`. Must pass in order to continue workflow.

`run_rcmdcheck` Run R CMD checks using `rcmdcheck::rcmdcheck()`. Must pass in order to continue workflow.

`as_cran` When running R CMD checks, use the '–as-cran' flag to apply CRAN standards

`run_vignettes` Build and check R package vignettes.

`has_testthat` Run unit tests and report results.

`has_runit` Run R Unit tests.

`run_lintr` Run `lintr::lint_package()` and emit each lint as a GitHub workflow annotation. Does not fail the workflow.

`run_spelling` Run `spelling::spell_check_package()` and emit each misspelling as a GitHub workflow annotation. Does not fail the workflow.

`run_covr` Run code coverage tests and publish results to codecov.

`codecov_token` Codecov repository token needed to upload coverage reports. Providing this token helps prevent report upload failures by bypassing Codecov's GitHub API rate limits. Read here for more details.

`has_latex` Install a suite of LaTeX dependencies used for rendering Sweave (.rnw) and other documentation files.

`tinytex_installer`
Which release of tinytex (bundles of LaTeX packages) to use. All options can be found here. Note, 'TinyTeX-2' is only available for `tinytex_version='daily'`.

`tinytex_version`
Which version of tinytex to use. When set to '', uses the latest daily build. All versions can be found here.

`pandoc_version` Which version of pandoc to use. For details see here.

`run_pkgdown` Knit the *README.Rmd* (if available), build documentation website, and deploy to *gh-pages* branch.

`run_docker` Whether to build and push a Docker container to DockerHub.

`docker_registry`
Docker container registry to push to. Options include:

"ghcr.io" : GitHub Container Registry

"docker.io" : DockerHub

`docker_user` Docker registry username. Not used when `docker_registry="ghcr.io"`.

`docker_org` Docker registry organization name. Is the same as `docker_user` by default. Not used when `docker_registry="ghcr.io"`.

`docker_token` Docker registry token. Not used when `docker_registry="ghcr.io"`.

`miniforge_variant`
: If provided, this variant of Miniforge will be downloaded and installed. If `miniforge_variant=false`, Miniforge will not be installed at all. If `miniforge_variant=""`, the "Miniforge3" variant will be installed. If `miniforge_version` is not provided, the `latest` version will be used. Currently-known values: - "Miniforge3" (default) - "Miniforge-pypy3" - "Mambaforge" - "Mambaforge-pypy3". Visit https://github.com/conda-forge/miniforge/releases/ for more information on available variants.

`miniforge_version`
: If provided, this version of the given Miniforge variant will be downloaded and installed. If `miniforge_variant` is not provided, "Miniforge3" will be used. Visit https://github.com/conda-forge/miniforge/releases/ for more information on available versions.

`activate_environment`
: Environment name (or path) to activate on all shells. Default is "test" which will be created in *$CONDA/envs/test*. If an empty string is used, no environment is activated by default (For "base" activation see the `auto-activate-base` option). If the environment does not exist, it will be created and activated. If `environment-file` is used and you want that to be the environment used, you need to explicitly provide the name of that environment on `activate-environment`. If using `sh/bash/cmd.exe` shells please read the IMPORTANT! section on the README.md! to properly activate conda environments on these shells.

`environment_file`
: Path or URL to a .yml file to build the conda environment with. For more information see here.

`channels` Conda configuration. Comma separated list of channels to use in order of priority. See here for more information.

`save_dir` Directory to save workflow to.

`return_path` Return the path to the saved *yaml* workflow file (default: `TRUE`), or return the *yaml* object directly.

`force_new` If the GHA workflow yaml already exists, overwrite with new one (default: `FALSE`).

`preview` Print the yaml file to the R console.

`verbose` Print messages.

Value

Path or yaml object.

Source

Issue reading in "on:"/"y","n" elements.

Issue writing "on:" as "'as':"

Examples

```
if (requireNamespace("curl", quietly = TRUE) && curl::has_internet()) {
  path <- use_workflow(save_dir = file.path(tempdir(),".github","workflows"))
} else {
  message("No internet connection available, skipping example.")
}
```

Index