Package: fledge (via r-universe)

October 15, 2024

Title Smoother Change Tracking and Versioning for R Packages

Version 0.1.99.9030

Date 2024-09-15

Description Streamlines the process of updating changelogs (NEWS.md) and versioning R packages developed in git repositories.

License GPL-3

URL https://fledge.cynkra.com/, https://github.com/cynkra/fledge

BugReports https://github.com/cynkra/fledge/issues

Imports brio, callr, cli, curl, desc (>= 1.4.1), devtools, glue, httr2, lifecycle, memoise, pandoc, parsedate, purrr (>= 0.3.2), rematch2, rlang (>= 0.4.12), tibble, usethis (>= 2.1.5), whoami, xml2

Suggests clipr, covr, digest, fansi, foghorn, fs, gert (>= 1.4.0), gh, httptest2, jsonlite, knitr, pkgbuild, pkgload, rmarkdown, rstudioapi, rversions, testthat (>= 3.1.10), vctrs, withr

Config/Needs/readme r-lib/downlit@f-readme-document

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2.9000

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first demo, parse-news-items, bump_version

Config/autostyle/scope line_breaks

Config/autostyle/strict true

Config/Needs/website cynkra/cynkratemplate

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

RemoteUrl https://github.com/cynkra/fledge

RemoteRef HEAD

RemoteSha b2c96870e6c4246b1361ab227fb2b817fb1d4f39

Contents

`bump_version` *Bump package version*

Description

Calls the following functions:

1. Verify that the current branch is the main branch if `check_default_branch = TRUE` (the default).
2. Check if there were changes since the last version.
3. `update_news()`, using the `which` argument
4. Depending on the `which` argument:
   - If `"dev"`, `finalize_version()` with `push = FALSE`
   - Otherwise, `commit_version()`.

Usage

```
bump_version(
  which = c("dev", "patch", "pre-minor", "minor", "pre-major", "major"),
  ...,
  no_change_behavior = c("bump", "noop", "fail"),
  check_default_branch = TRUE
)
```

Arguments

`which` Component of the version number to update. Supported values are

- `"auto"` (default: `"samedev"` or `"dev"`, depending on contents of `NEWS.md`),
- `"samedev"` (a.b.c.900x with stable version),
- `"dev"` (a.b.c.9xxx),

- "patch" (a.b.x),
- "pre-minor" (a.b.99.9000),
- "minor" (a.x.0),
- "pre-major" (a.99.99.9000),
- "major" (x.0.0).

`...` These dots are for future extensions and must be empty.

`no_change_behavior`
What to do if there was no change since the last version: "bump" for bump the version; "noop" for do nothing; "fail" for erroring.

`check_default_branch`
Whether to check that the current branch is the default branch.

Value

TRUE if NEWS.md and DESCRIPTION have been updated, FALSE otherwise. Do not rely on this behavior.

Bumped too soon?

Have you just run bump_version(), then realized "oh shoot, I forgot to merge that PR"? Fear not, run unbump_version(), merge that PR, run bump_version().

See Also

unbump_version()

Examples

```
# Create mock package in a temporary directory.
# Set open to TRUE if you want to play in the mock package.
with_demo_project({
  # Use functions as if inside the newly created package project.
  # (Or go and actually run code inside the newly created package project!)
  # Add a new R file.
  usethis::use_r("cool-function", open = FALSE)
  # Pretend we added useful code inside it.
  # Track the new R file with Git.
  gert::git_add("R/cool-function.R")
  gert::git_commit("- Add cool function.")
  # Bump version with fledge.
  fledge::bump_version()
})
```

commit_version *Commits NEWS.md and DESCRIPTION to Git*

Description

Commits changes to `NEWS.md` and `DESCRIPTION`, amending a previous commit created by **fledge** if necessary.

Usage

```
commit_version()
```

Value

Invisibly: `TRUE` if a previous commit for that version has been amended, `FALSE` if not.

Examples

```
# Create mock package in a temporary directory.
# Set open to TRUE if you want to play in the mock package.
with_demo_project({
  # Use functions as if inside the newly created package project.
  # (Or go and actually run code inside the newly created package project!)
  # Add a new R file.
  usethis::use_r("cool-function", open = FALSE)
  # Pretend we added useful code inside it.
  # Track the new R file with Git.
  gert::git_add("R/cool-function.R")
  gert::git_commit("- Add cool function.")
  # Bump version with fledge.
  fledge::bump_version()
  desc::desc_add_author(given = "Jane", family = "Doe", role = "ctb")
  fledge::commit_version()
})
```

create_demo_project *Create example repo for fledge demos*

Description

Create example repo for fledge demos

Usage

```
create_demo_project(
  open = rlang::is_interactive(),
  name = "tea",
  maintainer = NULL,
  email = NULL,
  date = "2021-09-27",
  dir = file.path(tempdir(), "fledge"),
  news = FALSE
)
```

Arguments

open	Whether to open the new project.
name	Package name.
maintainer	Name for DESCRIPTION and git.
email	Email for DESCRIPTION and git.
date	String of time for DESCRIPTION and git.
dir	Directory within which to create the mock package folder.
news	If TRUE, create a NEWS.md file.

Value

The path to the newly created mock package.

---

finalize_version *Finalize package version*

Description

Calls the following functions:

1. commit_version()
2. tag_version(), setting force = TRUE if and only if commit_version() amended a commit.
3. Force-pushes the created tag to the "origin" remote, if push = TRUE.

Usage

```
finalize_version(push = FALSE)
```

Arguments

push	If TRUE, push the created tag.

Value

None

Examples

```
# Create mock package in a temporary directory.
# Set open to TRUE if you want to play in the mock package.
with_demo_project({
  # Use functions as if inside the newly created package project.
  # (Or go and actually run code inside the newly created package project!)
  # Add a new R file.
  usethis::use_r("cool-function", open = FALSE)
  # Pretend we added useful code inside it.
  # Track the new R file with Git.
  gert::git_add("R/cool-function.R")
  gert::git_commit("- Add cool function.")
  # Bump version with fledge.
  fledge::bump_version()
  # Edit news by hand
  # ...
  # Once done
  fledge::finalize_version()
})
```

---

get_last_tag *The most recent tag*

Description

Returns the most recent Git tag.

Usage

```
get_last_tag()
```

Value

A one-row tibble with columns `name`, `ref` and `commit`. For annotated tags (as created by fledge), `commit` may be different from the SHA of the commit that this tag points to. Use `gert::git_log()` to find the actual commit.

Examples

```
# Create mock package in a temporary directory.
# Set open to TRUE if you want to play in the mock package.
with_demo_project({
  # Use functions as if inside the newly created package project.
  # (Or go and actually run code inside the newly created package project!)
  # Add a new R file.
```

```
  usethis::use_r("cool-function", open = FALSE)
  # Pretend we added useful code inside it.
  # Track the new R file with Git.
  gert::git_add("R/cool-function.R")
  gert::git_commit("- Add cool function.")
  # Bump version with fledge.
  fledge::bump_version()
  fledge::finalize_version()
  print(get_top_level_commits(since = NULL))
  print(fledge::get_last_tag())
})
```

`get_top_level_commits` *All top-level commits*

Description

Return all top-level commits since a particular version as commit objects.

Usage

```
get_top_level_commits(since = NULL)
```

Arguments

`since`	A commit SHA, e.g. as returned from `get_last_tag()`. If `NULL`, the entire log is retrieved.

Value

A tibble::tibble with at least two columns:

- `commit`: the commit SHA
- `message`: the commit message

Examples

```
# Create mock package in a temporary directory.
# Set open to TRUE if you want to play in the mock package.
with_demo_project({
  # Use functions as if inside the newly created package project.
  # (Or go and actually run code inside the newly created package project!)
  # Add a new R file.
  usethis::use_r("cool-function", open = FALSE)
  # Pretend we added useful code inside it.
  # Track the new R file with Git.
  gert::git_add("R/cool-function.R")
  gert::git_commit("- Add cool function.")
  # Bump version with fledge.
  fledge::bump_version()
```

```
  fledge::finalize_version()
  print(get_top_level_commits(since = NULL))
  print(fledge::get_last_tag())
})
```

---

release *Automating CRAN release*

Description

`init_release()` and `pre_release()` are run when a milestone in the development of a package is reached and it is ready to be sent to CRAN.

`release()` sends to CRAN after performing several checks, and offers help with accepting the submission.

`post_release()` should be called after the submission has been accepted.

Usage

```
init_release(which = "next", force = FALSE)

pre_release(force = FALSE)

release()

post_release()
```

Arguments

`which`	Component of the version number to update. Supported values are • `"next"` (`"major"` if the current version is `x.99.99.9yz`, `"minor"` if the current version is `x.y.99.za`, `"patch"` otherwise), • `"patch"` • `"minor"`, • `"major"`.
`force`	Create branches and tags even if they exist. Useful to recover from a previously failed attempt.

Details

`init_release()`:

- Ensures that no modified files are in the git index.
- Creates a release branch and bumps version to a non-development version which should be sent to CRAN.
- Writes/updates `cran-comments.md` with useful information about the current release process.

- Prompts the user to run `urlchecker::url_update()`, `devtools::check_win_devel()`, and `rhub::check_for_cran()`.

`pre_release()`:

- Opens a pull request for the release branch for final checks.

`tag_version` *Create a new version tag*

Description

Parses `NEWS.md` and creates/updates the tag for the most recent version.

Usage

```
tag_version(force = FALSE)
```

Arguments

`force`	Re-tag even if the last commit wasn't created by `bump_version()`. Useful when defining a CRAN release.

Value

The created tag, invisibly.

None

Examples

```
# Create mock package in a temporary directory.
# Set open to TRUE if you want to play in the mock package.
with_demo_project({
  # Use functions as if inside the newly created package project.
  # (Or go and actually run code inside the newly created package project!)
  # Add a new R file.
  usethis::use_r("cool-function", open = FALSE)
  # Pretend we added useful code inside it.
  # Track the new R file with Git.
  gert::git_add("R/cool-function.R")
  gert::git_commit("- Add cool function.")
  # Bump version with fledge.
  fledge::bump_version()
  fledge::update_news(c("- something I forgot", "- blabla"), which = "patch")
  gert::git_add("NEWS.md")
  gert::git_commit(message = "release notes tweaking")
  fledge::tag_version()
  print(fledge::get_last_tag())
})
```

unbump_version *Undoes bumping the package version*

Description

This undoes the effect of a `bump_version()` call, with a safety check.

Usage

```
unbump_version()
```

Value

`NULL`, invisibly. This function is called for its side effects.

None

See Also

bump_version

Examples

```
# Create mock package in a temporary directory.
# Set open to TRUE if you want to play in the mock package.
with_demo_project({
  # Use functions as if inside the newly created package project.
  # (Or go and actually run code inside the newly created package project!)
  # Add a new R file.
  usethis::use_r("cool-function", open = FALSE)
  # Pretend we added useful code inside it.
  # Track the new R file with Git.
  gert::git_add("R/cool-function.R")
  gert::git_commit("- Add cool function.")
  # Bump version with fledge.
  fledge::bump_version()
  # Oh no, we forgot to also add the awesome function for that version!
  # UNBUMP
  fledge::unbump_version()
  # Add a new R file.
  usethis::use_r("awesome-function", open = FALSE)
  # Pretend we added awesome code inside it.
  # Track the new R file with Git.
  gert::git_add("R/awesome-function.R")
  gert::git_commit("- Add awesome function.")
  # Bump version with fledge.
  fledge::bump_version()
  #'
})
```

update_news *Update NEWS.md with messages from top-level commits*

Description

Lists all commits from a range (default: top-level commits since the most recent tag) and adds bullets from their body to `NEWS.md`. Creates `NEWS.md` if necessary.

Usage

```
update_news(
  messages = NULL,
  which = c("auto", "samedev", "dev", "patch", "pre-minor", "minor", "pre-major",
    "major")
)
```

Arguments

`messages`	A character vector of commit messages, e.g. as in the `message` column in the return value of `get_top_level_commits()`. The default uses the top level commits since the last tag as retrieved by `get_last_tag()`.
`which`	Component of the version number to update. Supported values are • `"auto"` (default: `"samedev"` or `"dev"`, depending on contents of `NEWS.md`), • `"samedev"` (a.b.c.900x with stable version), • `"dev"` (a.b.c.9xxx), • `"patch"` (a.b.x), • `"pre-minor"` (a.b.99.9000), • `"minor"` (a.x.0), • `"pre-major"` (a.99.99.9000), • `"major"` (x.0.0).

Value

None

Examples

```
# Create mock package in a temporary directory.
# Set open to TRUE if you want to play in the mock package.
with_demo_project({
  # Use functions as if inside the newly created package project.
  # (Or go and actually run code inside the newly created package project!)
  # Add a new R file.
  usethis::use_r("cool-function", open = FALSE)
  # Pretend we added useful code inside it.
  # Track the new R file with Git.
```

```
  gert::git_add("R/cool-function.R")
  gert::git_commit("- Add cool function.")
  # Bump version with fledge.
  fledge::bump_version()
  fledge::update_news(c("- something I forgot", "- blabla"), which = "patch")
  gert::git_add("NEWS.md")
  gert::git_commit(message = "release notes tweaking")
  fledge::tag_version()
  print(fledge::get_last_tag())
})
```

`with_demo_project` *Run code in temporary project*

Description

Run code in temporary project

Usage

```
with_demo_project(code, dir = NULL, news = TRUE, quiet = FALSE)

local_demo_project(
  dir = NULL,
  news = TRUE,
  quiet = FALSE,
  .local_envir = parent.frame()
)
```

Arguments

`code`	Code to run with temporary active project
`dir`	Directory within which to create the mock package folder.
`news`	If TRUE, create a NEWS.md file.
`quiet`	Whether to show messages from usethis
`.local_envir`	The environment to use for scoping. Defaults to current execution environment.

Value

`with_demo_project()` returns the result of evaluating `code`.

`local_demo_project()` is called for its side effect and returns `NULL`, invisibly.

Examples

```
with_demo_project({
  # Add a new R file.
  usethis::use_r("cool-function", open = FALSE)
  # Pretend we added useful code inside it.
  # Track the new R file with Git.
  gert::git_add("R/cool-function.R")
  gert::git_commit("- Add cool function.")
  # Bump version with fledge.
  fledge::bump_version()
})
```

Index