

# Package: fledge (via r-universe)

October 28, 2024

**Title** Smoother Change Tracking and Versioning for R Packages

**Version** 0.1.99.9031

**Date** 2024-10-28

**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, htr2, 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, fohorn, 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://poissonconsulting.r-universe.dev>

**RemoteUrl** <https://github.com/cynkra/fledge>

**RemoteRef** HEAD

**RemoteSha** 38391e26ac6e8dfacc88a756ff7c7e55679ecde6

## Contents

bump_version . . . . .	2
commit_version . . . . .	4
create_demo_project . . . . .	4
finalize_version . . . . .	5
get_last_tag . . . . .	6
get_top_level_commits . . . . .	7
release . . . . .	8
tag_version . . . . .	9
unbump_version . . . . .	10
update_news . . . . .	11
with_demo_project . . . . .	12

<b>Index</b>	<b>14</b>
--------------	-----------

---

bump_version	<i>Bump package version</i>
--------------	-----------------------------

---

## 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

<code>which</code>	Component of the version number to update. Supported values are <ul style="list-style-type: none"> <li>• "auto" (default: "samedev" or "dev", depending on contents of NEWS.md),</li> <li>• "samedev" (a.b.c.900x with stable version),</li> <li>• "dev" (a.b.c.9xxx),</li> </ul>
--------------------	---

- "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	<i>Commits NEWS.md and DESCRIPTION to Git</i>
----------------	---

---

### 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	<i>Create example repo for fledge demos</i>
---------------------	---

---

### 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	<i>Finalize package version</i>
------------------	---------------------------------

---

**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

<code>which</code>	Component of the version number to update. Supported values are <ul style="list-style-type: none"> <li>• "next" ("major" if the current version is x.99.99.9yz, "minor" if the current version is x.y.99.za, "patch" otherwise),</li> <li>• "patch"</li> <li>• "minor",</li> <li>• "major".</li> </ul>
<code>force</code>	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	<i>Create a new version tag</i>
-------------	---------------------------------

---

## 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	<i>Undoes bumping the package version</i>
----------------	---

---

### 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

<code>messages</code>	A character vector of commit messages, e.g. as in the message column in the return value of <code>get_top_level_commits()</code> . The default uses the top level commits since the last tag as retrieved by <code>get_last_tag()</code> .
<code>which</code>	Component of the version number to update. Supported values are <ul style="list-style-type: none"><li>• "auto" (default: "samedev" or "dev", depending on contents of NEWS.md),</li><li>• "samedev" (a.b.c.900x with stable version),</li><li>• "dev" (a.b.c.9xxx),</li><li>• "patch" (a.b.x),</li><li>• "pre-minor" (a.b.99.9000),</li><li>• "minor" (a.x.0),</li><li>• "pre-major" (a.99.99.9000),</li><li>• "major" (x.0.0).</li></ul>

### 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	<i>Run code in temporary project</i>
-------------------	--------------------------------------

---

## 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

bump\_version, [2](#)  
bump\_version(), [9](#), [10](#)  
bump\_version\_impl (bump\_version), [2](#)

commit\_version, [4](#)  
commit\_version(), [2](#), [5](#)  
create\_demo\_project, [4](#)

finalize\_version, [5](#)  
finalize\_version(), [2](#)  
finalize\_version\_impl  
    (finalize\_version), [5](#)

gert::git\_log(), [6](#)  
get\_last\_tag, [6](#)  
get\_last\_tag(), [7](#), [11](#)  
get\_top\_level\_commits, [7](#)  
get\_top\_level\_commits(), [11](#)

init\_release (release), [8](#)

local\_demo\_project (with\_demo\_project),  
    [12](#)

post\_release (release), [8](#)  
pre\_release (release), [8](#)

release, [8](#)

tag\_version, [9](#)  
tag\_version(), [5](#)  
tibble::tibble, [7](#)

unbump\_version, [10](#)  
unbump\_version(), [3](#)  
unbump\_version\_impl (unbump\_version), [10](#)  
update\_news, [11](#)  
update\_news(), [2](#)

with\_demo\_project, [12](#)