# Package: fusen (via r-universe)

August 27, 2024

**Title** Build a Package from Rmarkdown Files

**Version** 0.6.0.9003

**Description** Use Rmarkdown First method to build your package. Start your package with documentation, functions, examples and tests in the same unique file. Everything can be set from the Rmarkdown template file provided in your project, then inflated as a package. Inflating the template copies the relevant chunks and sections in the appropriate files required for package development.

**License** MIT + file LICENSE

**URL** https://thinkr-open.github.io/fusen/, https://github.com/Thinkr-open/fusen

**Depends** R (>= 3.5.0)

**Imports** attachment, cli, covr, desc, devtools, glue, here (>= 1.0.0), lightparser, magrittr, methods, pkgload, roxygen2, stats, stringi, tibble, tidyr, tools, usethis (>= 2.0.0), utils, yaml

**Suggests** gert, knitr, rcmdcheck, rmarkdown, rstudioapi, styler, testthat (>= 3.2.0), withr

**VignetteBuilder** knitr

**Config/fusen/version** 0.6.0.9002

**Config/testthat/edition** 3

**Config/testthat/parallel** false

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**Repository** https://thinkr-open.r-universe.dev

**RemoteUrl** https://github.com/ThinkR-open/fusen

**RemoteRef** HEAD

**RemoteSha** 1f7fd59336da2dd87afc0868f9b8b7bae261d6bc

# Contents



---

`add_additional` *Add flat Rmd file that drives package development*

---

## Description

Add flat Rmd file that drives package development

## Usage

```
add_additional(
  pkg = ".",
  dev_dir = "dev",
  flat_name = "additional",
  overwrite = FALSE,
  open = TRUE
)

add_minimal_flat(
  pkg = ".",
  dev_dir = "dev",
  flat_name = "minimal",
  overwrite = FALSE,
  open = TRUE
)

add_minimal_package(
  pkg = ".",
```

```
  dev_dir = "dev",
  flat_name = "minimal",
  overwrite = FALSE,
  open = TRUE
)

add_full(
  pkg = ".",
  dev_dir = "dev",
  flat_name = "full",
  overwrite = FALSE,
  open = TRUE
)

add_dev_history(pkg = ".", dev_dir = "dev", overwrite = FALSE, open = TRUE)

add_flat_template(
 template = c("full", "minimal_package", "minimal_flat", "additional", "teaching",
    "dev_history"),
  pkg = ".",
  dev_dir = "dev",
  flat_name = NULL,
  overwrite = FALSE,
  open = TRUE
)
```

## Arguments

| | |
|---|---|
| `pkg` | Path where to save file |
| `dev_dir` | Name of directory for development Rmarkdown files. Default to "dev". |
| `flat_name` | Name of the file to write in dev. Use the name of the main function of your template to get chunks pre-filled with this function name. |
| `overwrite` | Whether to overwrite existing flat Rmd template file with same name |
| `open` | Logical. Whether to open file after creation |
| `template` | Name of the template to use. See details. |

## Details

Choose `template` among the different templates available:

- "full": The full template with a reproducible package that can directly be inflated. It comes along with the "dev_history" template. Default.
- "minimal_package": Minimal template to start a new package when you already know 'fusen', along with the "dev_history" template. Note that this is called "minimal" in `create_fusen()`.
- "minimal_flat" or "additional": Template for a new minimal flat file only.
- "teaching": Template with a reproducible package, simpler than "full", but everything to teach the minimal structure of a package.

- "dev_history": Template with functions commonly used during package development. This does not contain chunks to write your own functions.

Abbreviated names can also be used for the different templates: "add" for additional, "minflat" for minimal_flat, "minpkg" for minimal_package "teach" for teaching, "dev" for "dev_history".

`add_additional()`, `add_minimal_flat()`, `add_dev_history()`, `add_minimal_package()`, `add_full()` are wrapper around `add_flat_template("additional")`, ...

## Value

Create flat Rmd file(s) template(s) and return its (their) path

## Examples

```
# Create a new project
dummypackage <- tempfile("dummy.package.flat")
dir.create(dummypackage)

# Add
add_flat_template(template = "teaching", pkg = dummypackage)
# Delete dummy package
unlink(dummypackage, recursive = TRUE)

# For classical use in your package
## Not run:
# first time ever using 'fusen'
add_flat_template("full")

# first time in your new package
add_flat_template("minimal_package")

# add new flat file for new functions
add_flat_template("add")
add_additional()
add_minimal_flat()

# add only the dev_history file in an existing package
add_dev_history()

# add new flat template for teaching (a reduced full template)
add_flat_template("teaching")

## End(Not run)
```

---

`add_fusen_chunks` *Add 'fusen' chunks*

---

## Description

Create 'fusen' chunks inside your Rmd

### Usage

```
add_fusen_chunks(
  function_name = NULL,
  export = getOption("fusen.export.functions")
)
```

### Arguments

| | |
|---|---|
| function_name | Name of the function to create. If NULL (the default), the user will be prompted to enter it. |
| export | Should the function be exported? Default is `getOption("fusen.export.functions")`. If NULL, the user will be prompted to enter it. |

### Value

A list with the context and the content, invisibly.

### Examples

```
## Not run:
add_fusen_chunks("this", export = TRUE)

## End(Not run)
```

---

## check_not_registered_files

*Show in a table files that are not already registered in the yaml config file*

---

### Description

If user start their package without 'fusen' or with version < 0.4, they need to create the config file, with already existing functions.

### Usage

```
check_not_registered_files(
  path = ".",
  config_file,
  guess = TRUE,
  to_csv = TRUE,
  open = FALSE
)
```

### Arguments

| | |
|---|---|
| path | Path to package to check for not registered files |
| config_file | Path to the configuration file |
| guess | Logical. Guess if the file was inflated by a specific flat file |
| to_csv | Logical. Whether to store along the config file, the outputs in a csv for the user to clean it manually |
| open | Logical. Whether to open the csv of unregistered files. |

### Value

Path to csv file if `to_csv` is TRUE. `dput()` of the dataframe otherwise.

### See Also

`register_all_to_config()` for automatically registering all files already present in the project, `inflate_all()` to inflate every flat files according to the configuration file.

### Examples

```
## Not run:
# Run this on the current package in development
out_csv <- check_not_registered_files()
file.edit(out_csv)

## End(Not run)

# Or you can try on the reproducible example
dummypackage <- tempfile("clean")
dir.create(dummypackage)

# {fusen} steps
fill_description(pkg = dummypackage, fields = list(Title = "Dummy Package"))
dev_file <- suppressMessages(add_flat_template(pkg = dummypackage, overwrite = TRUE, open = FALSE))
flat_file <- dev_file[grepl("flat_", dev_file)]
# Inflate once
usethis::with_project(dummypackage, {
  suppressMessages(
    inflate(
      pkg = dummypackage, flat_file = flat_file,
      vignette_name = "Get started", check = FALSE,
      open_vignette = FALSE
    )
  )

  # Add a not registered file to the package
  cat("# test R file\n", file = file.path(dummypackage, "R", "to_keep.R"))

  # Use the function to check the list of files
  out_csv <- check_not_registered_files(dummypackage)
  out_csv
```

```
  # Read the csv to see what is going on
  content_csv <- read.csv(out_csv, stringsAsFactors = FALSE)
  content_csv
  # Keep it all or delete some files, and then register all remaining
  out_config <- register_all_to_config()
  out_config
  # Open the out_config file to see what's going on
  yaml::read_yaml(out_config)
})
unlink(dummypackage, recursive = TRUE)
```

---

`create_fusen` *Create a new fusen project*

---

## Description

Create a new fusen project

## Usage

```
create_fusen(
  path,
  template = c("full", "minimal", "teaching", "dev_history"),
  flat_name = template,
  open = TRUE,
  overwrite = FALSE,
  with_git = FALSE
)
```

## Arguments

| | |
|---|---|
| `path` | Character. Path where to create the new fusen project. |
| `template` | Character. Name of the template to be used among "full", "minimal", "teaching" and "dev_history". |
| `flat_name` | Character. Filename of the flat file created. This is also used to name the first function of the file in `minimal` template. |
| `open` | Logical. Should the newly created project be opened ? |
| `overwrite` | Logical. Allow to overwrite 'dev/' files if path exists. |
| `with_git` | Logical. Should git be initialized in the newly created project ? |

## Details

See `add_flat_template` for details about the different options for `template`. Template "additional" is not available here as it is meant to be used with an already existing 'fusen'.

## Value

Path to dev and flat files. Side-effect: Create a new directory to build a package

## Examples

```
my_path <- tempfile("mypkg")
create_fusen(path = my_path, template = "full", open = FALSE)
```

---

`deprecate_flat_file` *Deprecate a flat file*

---

## Description

It is not inflated again during [`inflate_all()`] as it is identified as deprecated in the config file. Previously generated files get "do not edit by hand" message removed. The flat file is moved to "dev/flat_history".

## Usage

```
deprecate_flat_file(flat_file)
```

## Arguments

| | |
|---|---|
| `flat_file` | Path to the flat file to deprecate |

## Value

Used for side effect. Move flat file to "dev/flat_history", update config file, and remove "do not edit by hand" message.

## Examples

```
## Not run:
# These functions change the current user workspace
dev_file <- suppressMessages(
  add_flat_template(
    template = "add",
    pkg = dummypackage, overwrite = TRUE, open = FALSE
  )
)
deprecate_flat_file(flat_file = "dev/flat_additional.Rmd")

## End(Not run)
```

`fill_description` *Fill DESCRIPTION file of the package*

## Description

Fill DESCRIPTION file of the package

## Usage

```
fill_description(pkg = ".", fields, overwrite = FALSE)
```

## Arguments

| | |
|---|---|
| `pkg` | Path to package |
| `fields` | A named list of fields to add to DESCRIPTION, potentially overriding default values. See `use_description` for how you can set personalized defaults using package options |
| `overwrite` | Whether to overwrite existing DESCRIPTION |

## Value

Fill DESCRIPTION file with fields. Return path to file.

## Examples

```
# Create a new project
dummypackage <- tempfile("dummy.package.desc")
dir.create(dummypackage)

fill_description(
  pkg = dummypackage,
  fields = list(
    Title = "Build A Package From Rmarkdown file",
    Description = paste(
      "Use Rmd First method to build your package.",
      "Start your package with documentation.",
      "Everything can be set from a Rmarkdown file in your project."
    ),
    `Authors@R` = c(
      person("John", "Doe",
        email = "john@email.me",
        role = c("aut", "cre"), comment = c(ORCID = "0000-0000-0000-0000")
      )
    )
  )
)

# Delete dummy package
unlink(dummypackage, recursive = TRUE)
```

## get_all_created_funs *Get all functions created in a R file*

### Description

Get all functions created in a R file

### Usage

```
get_all_created_funs(file)
```

### Arguments

| | |
|---|---|
| file | A R file |

### Value

A character vector of function names

### Examples

```
file_path <- tempfile(fileext = ".R")
cat(
  "my_fun <- function() {1}",
  "my_fun2 <- function() {2}",
  sep = "\n",
  file = file_path
)
get_all_created_funs(file_path)
```

## get_package_structure *Get structure and information of a 'fusen' built package for developers*

### Description

Get structure and information of a 'fusen' built package for developers

Draw a tree of the package structure in the console

### Usage

```
get_package_structure(config_file, emoji = TRUE, silent = FALSE)

draw_package_structure(structure_list, silent = FALSE)
```

### Arguments

| | |
|---|---|
| config_file | Path to a source configuration file to get the structure from |
| emoji | Add emojis to the output |
| silent | Whether to print messages |
| structure_list | A list of information about the package as issued from [get_package_structure()] |

### Value

A list of information about the package

### Examples

```
## Not run:
# This only works inside a 'fusen' built package
pkg_structure <- get_package_structure()
draw_package_structure(pkg_structure)

## End(Not run)

# Example with a dummy package
dummypackage <- tempfile("drawpkg.structure")
dir.create(dummypackage)

# {fusen} steps
fill_description(pkg = dummypackage, fields = list(Title = "Dummy Package"))
dev_file <- suppressMessages(
  add_flat_template(pkg = dummypackage, overwrite = TRUE, open = FALSE)
)
flat_file <- dev_file[grepl("flat_", dev_file)]

usethis::with_project(dummypackage, {
  # Add an extra R file with internal function
  # to list in "keep"
  dir.create("R")
  cat("extra_fun <- function() {1}\n", file = "R/my_extra_fun.R")

  # Works with classical package
  pkg_structure <- get_package_structure()
  draw_package_structure(pkg_structure)
})

usethis::with_project(dummypackage, {
  # Works with 'fusen' package
  suppressMessages(
    inflate(
      pkg = dummypackage, flat_file = flat_file,
      vignette_name = "Get started", check = FALSE,
      open_vignette = FALSE
    )
  )
```

```
  pkg_structure <- get_package_structure()
  draw_package_structure(pkg_structure)
})
```

---

`inflate` *Inflate Rmd to package*

---

## Description

Inflate Rmd to package

## Usage

```
inflate(
  pkg = ".",
  flat_file,
  vignette_name = "Get started",
  open_vignette = TRUE,
  check = TRUE,
  document = TRUE,
  overwrite = "ask",
  clean = "ask",
  update_params = TRUE,
  codecov = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `pkg` | Path to package |
| `flat_file` | Path to Rmarkdown file to inflate |
| `vignette_name` | Character. Title of the resulting vignette. Use `NA` if you do not want to create a vignette. |
| `open_vignette` | Logical. Whether to open vignette file at the end of the process |
| `check` | Logical. Whether to check package after Rmd inflating |
| `document` | Logical. Whether to document your package using `att_amend_desc` |
| `overwrite` | Logical (TRUE, FALSE) or character ("ask", "yes", "no). Whether to overwrite vignette and functions if already exists. |
| `clean` | Logical (TRUE, FALSE) or character ("ask", "yes", "no) Whether to delete files that are not anymore created by the current flat file. Typically, if you have deleted or renamed a function in the flat file. Default to "ask". |
| `update_params` | Logical. Whether to update the inflate parameters in the configuration file. |
| `codecov` | Logical. Whether to compute code coverage (with `covr::package_coverage()` or `covr::report()`). |
| `...` | Arguments passed to `devtools::check()`. For example, you can do `inflate(check = TRUE, quiet = TRUE)`, where `quiet` is passed to `devtools::check()`. |

### Value

Package structure. Return path to current package.

### See Also

`inflate_all()` to inflate every flat files according to the configuration file.

### Examples

```
# Create a new project
dummypackage <- tempfile("dummy.package")
dir.create(dummypackage)

# {fusen} steps
dev_file <- add_flat_template(template = "full", pkg = dummypackage, overwrite = TRUE)
flat_file <- dev_file[grepl("flat", dev_file)]
fill_description(pkg = dummypackage, fields = list(Title = "Dummy Package"))
inflate(
  pkg = dummypackage, flat_file = flat_file,
  vignette_name = "Exploration of my Data", check = FALSE
)

# Explore directory of the package
# browseURL(dummypackage)

# Try pkgdown build
# usethis::use_pkgdown()
# pkgdown::build_site(dummypackage)
# Delete dummy package
unlink(dummypackage, recursive = TRUE)
```

---

## `inflate_all` *Inflate all your flat files*

---

### Description

Inflate all the flat files stored in "dev/" and starting with "flat_"

### Usage

```
inflate_all(
  pkg = ".",
  document = TRUE,
  check = TRUE,
  open_vignette = FALSE,
  overwrite = TRUE,
  check_unregistered = TRUE,
  codecov = FALSE,
```

```
  stylers,
  ...
)

inflate_all_no_check(
  pkg = ".",
  document = TRUE,
  open_vignette = FALSE,
  overwrite = TRUE,
  check_unregistered = TRUE,
  codecov = FALSE,
  stylers,
  ...
)
```

## Arguments

| | |
|---|---|
| pkg | Path to package |
| document | Logical. Whether to document your package using att_amend_desc |
| check | Logical. Whether to check package after Rmd inflating |
| open_vignette | Logical. Whether to open vignette file at the end of the process |
| overwrite | Logical (TRUE, FALSE) or character ("ask", "yes", "no). Whether to overwrite vignette and functions if already exists. |
| check_unregistered | Logical. Whether to help detect unregistered files. Typically files not created from a flat file and added manually in the repository. |
| codecov | Logical. Whether to compute code coverage (with covr::package_coverage() or covr::report()). |
| stylers | Function to be run at the end of the process, like styler::style_pkg or lintr::lint_package or a lambda function combining functions like: function() {styler::style_pkg(); lintr::lint_package()}. For a unique function, use the format without parenthesis () at the end of the command. |
| ... | Arguments passed to devtools::check(). For example, you can do inflate(check = TRUE, quiet = TRUE), where quiet is passed to devtools::check(). |

## Details

This requires to inflate() all flat files individually at least once, so that their specific inflate configurations are stored.

This also requires to register all R, tests and vignettes files of your package, even if not created with an inflate. Run inflate_all() once and read the messages. The first time, you will probably need to run register_all_to_config() if your package is not new.

For more information, read the vignette("inflate-all-your-flat-files", package = "fusen")

## Value

side effect. Inflates all your flat files that can be inflated.

## See Also

inflate() for the options of a single file inflate, check_not_registered_files() for the list of files not already associated with a flat file in the config file, register_all_to_config() for automatically registering all files already present in the project before the first inflate_all()

## Examples

```
## Not run:
# Usually, in the current package run inflate_all() directly
# These functions change the current user workspace
inflate_all()
# Or inflate_all_no_check() to prevent checks to run
inflate_all_no_check()
# Or inflate with the styler you want
inflate_all(stylers = styler::style_pkg)

## End(Not run)

# You can also inflate_all flats of another package as follows
# Example with a dummy package with a flat file
dummypackage <- tempfile("inflateall.otherpkg")
dir.create(dummypackage)
fill_description(pkg = dummypackage, fields = list(Title = "Dummy Package"))
flat_files <- add_minimal_package(
  pkg = dummypackage,
  overwrite = TRUE,
  open = FALSE
)
flat_file <- flat_files[grep("flat", basename(flat_files))]
# Inflate the flat file once
usethis::with_project(dummypackage, {
  # if you are starting from a brand new package, inflate_all() will crash
  # it's because of the absence of a fusen config file
  #
  # inflate_all() # will crash

  # Add licence
  usethis::use_mit_license("John Doe")

  # you need to inflate manually your flat file first
  inflate(
    pkg = dummypackage,
    flat_file = flat_file,
    vignette_name = "Get started",
    check = FALSE,
    open_vignette = FALSE,
    document = TRUE,
    overwrite = "yes"
  )

  # your config file has been created
  config_yml_ref <-
```

```
    yaml::read_yaml(getOption("fusen.config_file", default = "dev/config_fusen.yaml"))
})

# Next time, you can run inflate_all() directly
usethis::with_project(dummypackage, {
  # now you can run inflate_all()
  inflate_all(check = FALSE, document = TRUE)
})

# If you wish, the code coverage can be computed
usethis::with_project(dummypackage, {
  # now you can run inflate_all()
  inflate_all(check = FALSE, document = TRUE, codecov = TRUE)
})

# Clean the temporary directory
unlink(dummypackage, recursive = TRUE)
```

---

`init_share_on_github` *Initiate GitHub to share your package on a website*

---

## Description

This uses 'pkgdown' to share the documentation of the package through GitHub Actions. You may need to run `usethis::create_github_token()`, then `gitcreds::gitcreds_set()` before.

## Usage

```
init_share_on_github(ask = TRUE, organisation = NULL)
```

## Arguments

| | |
|---|---|
| `ask` | Logical. `TRUE` (default) to ask the user to apply the instructions each time needed, or `FALSE` if the user already know what to do. |
| `organisation` | If supplied, the repo will be created under this organisation, instead of the login associated with the GitHub token discovered for this `host`. The user's role and the token's scopes must be such that you have permission to create repositories in this `organisation`. |

## Details

`init_share_on_github()` runs multiple steps to be able to share a proper package on GitHub:

- Start versionning with git if not already
- Connect to your GitHub account
- Create a minimal DESCRIPTION file if missing
  - You will have to update its content with your information after deployment

- Add NEWS file to present modifications of your releases
- Add README.Rmd and knit it to README.md to quickly present the aim and the use of your package
- Init continuous integration (CI)
  - Check the package on Linux, Windows and MacOS
  - Calculate code coverage. Note that you may need to connect to https://about.codecov.io/ to see the results of the code coverage.
- Init continuous deployment (CD) of the 'pkgdown' website documentation
- Commit and push to GitHub
- List remaining manual steps to make the website public

Read `vignette("share-on-a-github-website", package = "fusen")`

## Value

The URL of the website created

## Examples

```
## Not run:
# This modifies the current directory and send it on GitHub
init_share_on_github()

## End(Not run)
```

---

`list_flat_files` *List all flat files present in the package*

---

## Description

Search for flat files listed in fusen config file, and for rmd and qmd files starting with "flat_" in dev/ folder, and dev/flat_history folder

## Usage

```
list_flat_files(pkg = ".")
```

## Arguments

| | |
|---|---|
| pkg | Path to package |

## Value

a vector of flat files paths

## load_flat_functions *Load the code of all 'function' chunk in a flat file*

### Description

Load the code of all 'function' chunk in a flat file

### Usage

```
load_flat_functions(flat_file, envir = globalenv())
```

### Arguments

| | |
|---|---|
| flat_file | Path to the flat to load functions from |
| envir | the environment in which expr is to be evaluated. May also be NULL, a list, a data frame, a pairlist or an integer as specified to sys.call. |

### Value

Path to flat file loaded. Used for side effect: Load functions in the global environment.

### Examples

```
## Not run:
# Use this command directly in the console
fusen::load_flat_functions()

# Or choose a flat file to load functions from
load_flat_functions(flat_file = "dev/flat_full.Rmd")
load_flat_functions(flat_file = "dev/flat_clean_fusen_files.Rmd")

## End(Not run)
```

## register_all_to_config *Include all existing package files in the config file*

### Description

Helps transition from non-fusen packages or made with earlier version

### Usage

```
register_all_to_config(pkg = ".", config_file)
```

### Arguments

| | |
|---|---|
| pkg | Path to the package from which to add file to configuration file |
| config_file | Path to the configuration file |

### Value

Invisible path to 'fusen' configuration file

### See Also

check_not_registered_files() for the list of files not already associated with a flat file in the config file,

### Examples

```
## Not run:
# Usually run this one inside the current project
# Note: running this will write "dev/config_fusen.yaml" in your working directory
register_all_to_config()

## End(Not run)

# Or you can try on the reproducible example
dummypackage <- tempfile("register")
dir.create(dummypackage)

# {fusen} steps
fill_description(pkg = dummypackage, fields = list(Title = "Dummy Package"))
dev_file <- suppressMessages(add_flat_template(pkg = dummypackage, overwrite = TRUE, open = FALSE))
flat_file <- dev_file[grepl("flat_", dev_file)]
# Inflate once
usethis::with_project(dummypackage, {
  suppressMessages(
    inflate(
      pkg = dummypackage, flat_file = flat_file,
      vignette_name = "Get started", check = FALSE,
      open_vignette = FALSE
    )
  )
  out_path <- register_all_to_config(dummypackage)

  # Look at the output
  yaml::read_yaml(out_path)
})
```

## rename_flat_file *Rename a flat file*

### Description

Rename a flat file

### Usage

```
rename_flat_file(flat_file, new_name)
```

### Arguments

| | |
|---|---|
| flat_file | Path to the flat file to rename |
| new_name | New name for the flat file |

### Value

Used for side effect. Flat file renamed, config file updated, inflated files modified when needed.

### Examples

```
## Not run:
# These functions change the current user workspace
dev_file <- suppressMessages(
  add_flat_template(
    template = "add",
    pkg = ".", overwrite = TRUE, open = FALSE
  )
)
rename_flat_file(
  flat_file = "dev/flat_additional.Rmd",
  new_name = "flat_new.Rmd"
)

## End(Not run)
```

## sepuku *Clean a package from fusen-related files and tags*

### Description

This function will delete all the flat files (i.e files listed in the fusen configuration file, as well as rmd or qmd files starting with "flat" in the "dev/" and "dev/flat_history" folders). It will also remove the fusen-related tags added by calls to `fusen::inflate()` in files located in the "R/", "tests/" and "vignettes/" folders. Finally, it will also remove the fusen configuration file if it exists.

## Usage

```
sepuku(pkg = ".", force = FALSE, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| pkg | Character. Path of the current package |
| force | logical. whether to force the cleaning or not, without asking if the user is sure about making this operation (default: FALSE) |
| verbose | logical. whether to display the files that will be deleted or modified (default: FALSE) |

## Value

side effect. A package cleaned from fusen-related files or tags

## Examples

```
## Not run:
sepuku()
# If you want to force the cleaning, you can use the force argument
sepuku(force = TRUE)

# Example with a dummy package
dummypackage <- tempfile("sepuku.example")
dir.create(dummypackage)
fill_description(pkg = dummypackage, fields = list(Title = "Dummy Package"))

usethis::with_project(dummypackage, {
  # Add licence
  usethis::use_mit_license("John Doe")

  dir.create(file.path(dummypackage, "dev"))
  dir.create(file.path(dummypackage, "dev", "flat_history"))

  # We add 2 flat files in the package and inflate them
  dev_file1 <- add_minimal_flat(
    pkg = dummypackage,
    flat_name = "flat1.Rmd",
    open = FALSE
  )

  dev_file2 <- add_minimal_flat(
    pkg = dummypackage,
    flat_name = "flat2.Rmd",
    open = FALSE
  )

  inflate(
    pkg = dummypackage,
    flat_file = dev_file1,
    vignette_name = "Get started",
```

```
  check = FALSE,
  open_vignette = FALSE,
  document = TRUE,
  overwrite = "yes"
)

inflate(
  pkg = dummypackage,
  flat_file = dev_file2,
  vignette_name = "Get started 2",
  check = FALSE,
  open_vignette = FALSE,
  document = TRUE,
  overwrite = "yes"
)

# We deprecate the first flat file, which will be moved to the flat_history folder
deprecate_flat_file(
  file.path(dummypackage, "dev", "flat_flat1.Rmd")
)

# We create 2 flat files with the qmd extension
file.create(file.path(dummypackage, "dev", "flat_history", "flat_old.qmd"))
file.create(file.path(dummypackage, "dev", "flat_qmd.qmd"))

sepuku(force = TRUE)

# We check that the fusen configuration file has been deleted
file.exists(
  file.path(dummypackage, "dev", "config_fusen.yaml")
)

# We check that all the flat files have been deleted
length(
  list.files(file.path(dummypackage, "dev"),
    pattern = "^flat.*\\.Rmd"
  )
)

length(
  list.files(file.path(dummypackage, "dev"),
    pattern = "^flat.*\\.qmd"
  )
)


length(
  list.files(file.path(dummypackage, "dev", "flat_history"),
    pattern = "^flat.*\\.Rmd"
  )
)
```

```
  length(
    list.files(file.path(dummypackage, "dev", "flat_history"),
      pattern = "^flat.*\\.qmd"
    )
  )

  # We check that all the files with fusen tags have been cleaned
  length(fusen:::find_files_with_fusen_tags(pkg = dummypackage))
})

# Clean the temporary directory
unlink(dummypackage, recursive = TRUE)

## End(Not run)
```

# Index