

# Package ‘pkglite’

February 21, 2024

**Title** Compact Package Representations

**Version** 0.2.2

**Description** A tool, grammar, and standard to represent and exchange R package source code as text files. Converts one or more source packages to a text file and restores the package structures from the file.

**License** GPL-3

**URL** <https://merck.github.io/pkglite/>, <https://github.com/Merck/pkglite>

**BugReports** <https://github.com/Merck/pkglite/issues>

**Encoding** UTF-8

**VignetteBuilder** knitr

**Depends** R (>= 3.5.0)

**Imports** crayon, magrittr, remotes

**Suggests** covr, knitr, rmarkdown, testthat

**RoxygenNote** 7.3.1

**NeedsCompilation** no

**Author** Nan Xiao [aut, cre] (<<https://orcid.org/0000-0002-0250-5673>>),  
Yilong Zhang [aut],  
Keaven Anderson [aut],  
Amin Shirazi [ctb],  
Jeff Cheng [ctb],  
Danfeng Fu [ctb],  
Merck & Co., Inc., Rahway, NJ, USA and its affiliates [cph]

**Maintainer** Nan Xiao <[nan.xiao1@merck.com](mailto:nan.xiao1@merck.com)>

**Repository** CRAN

**Date/Publication** 2024-02-21 05:20:02 UTC

## R topics documented:

collate . . . . .	2
ext_binary . . . . .	3

ext_text . . . . .	4
file_auto . . . . .	4
file_default . . . . .	5
file_ectd . . . . .	6
file_name_patterns . . . . .	6
file_spec . . . . .	7
file_spec_templates . . . . .	8
is_file_collection . . . . .	9
is_file_spec . . . . .	9
merge.file_collection . . . . .	10
pack . . . . .	11
print.file_collection . . . . .	12
print.file_spec . . . . .	12
prune . . . . .	13
remove_content . . . . .	14
sanitize . . . . .	15
sanitize_file_collection . . . . .	15
unpack . . . . .	16
verify_ascii . . . . .	17

**Index** **19**

---

collate	<i>Evaluate a list of file specifications</i>
---------	---

---

**Description**

Evaluate a list of file specifications and bind the results as a file collection.

**Usage**

```
collate(pkg = ".", ...)
```

**Arguments**

pkg	Path to the package directory.
...	One or more file specification objects.

**Value**

A file collection object containing the package name, file paths, and file format types.

**Specification**

- Get package metadata, for example, the package name.
- Flatten the input list of file specification(s).
- Evaluate the file specification(s) under the pkg directory.
- Remove duplicated files from all evaluation results and store the file path and format information in a data frame.
- Return the package metadata and the data frame in a list.

**Examples**

```
system.file("examples/pkg1/", package = "pkglite") %>%  
  collate(file_default())
```

---

ext_binary	<i>Common file extensions (binary)</i>
------------	--

---

**Description**

Common file extensions (binary)

**Usage**

```
ext_binary(flat = FALSE)
```

**Arguments**

flat                      Flatten the list and return a vector?

**Value**

A list or vector of standard binary file extensions.

**Specification**

- Return a named list of common binary file extensions in R packages.

**Examples**

```
ext_binary()
```

---

ext_text	<i>Common file extensions (text)</i>
----------	--------------------------------------

---

**Description**

Common file extensions (text)

**Usage**

```
ext_text(flat = FALSE)
```

**Arguments**

flat                    Flatten the list and return a vector?

**Value**

A list or vector of standard text file extensions.

**Specification**

- Return a named list of common text file extensions in R packages.

**Examples**

```
ext_text()
```

---

file_auto	<i>File specification (automatic guess)</i>
-----------	---

---

**Description**

Lists all files under a folder recursively and guesses the file format type (text or binary) based on the file extension.

**Usage**

```
file_auto(path)
```

**Arguments**

path                    The directory's relative path (relative to the package root), for example, "inst/".

**Value**

A list of file specifications.

**Specification**

- Use `file_spec()` to cover all text files by their file extensions under the relative path of the package, recursively.
- Use `file_spec()` to cover all binary files by their file extensions under the relative path of the package, recursively.
- Return the combination of the two file specifications in a list.

**Examples**

```
file_auto("inst/")
```

---

file_default	<i>File specification (default combination)</i>
--------------	---

---

**Description**

A default combination of common file specifications.

**Usage**

```
file_default()
```

**Value**

A list of file specifications.

**Specification**

- Use file specification template functions to generate and return a list of file specifications that cover a default set of the common naming patterns of files in source R packages.

**Examples**

```
file_default()
```

---

file_ectd	<i>File specification (eCTD submission)</i>
-----------	---

---

**Description**

A combination of file specifications for eCTD submissions.

**Usage**

```
file_ectd()
```

**Value**

A list of file specifications.

**Specification**

- Use file specification template functions to generate and return a list of file specifications that cover the a set of the common naming patterns of files in source R packages for eCTD submissions.

**Examples**

```
file_ectd()
```

---

file_name_patterns	<i>Common File name patterns</i>
--------------------	----------------------------------

---

**Description**

Common File name patterns

**Usage**

```
pattern_file_root_core()
```

```
pattern_file_root_all()
```

```
pattern_file_sanitize()
```

**Value**

A vector of file name patterns.

**Specification**

- Return a vector of filename patterns for matching files.

---

file_spec	<i>Create a file specification</i>
-----------	------------------------------------

---

## Description

Specify which files to include

## Usage

```
file_spec(  
  path,  
  pattern = NULL,  
  format = c("binary", "text"),  
  recursive = TRUE,  
  ignore_case = TRUE,  
  all_files = FALSE  
)
```

## Arguments

path	Path relative to the package root), for example, "inst/".
pattern	Regular expression for matching the file names.
format	File format type, one of "binary" or "text".
recursive	List files in the sub-directories?
ignore_case	Should pattern-matching be case-insensitive?
all_files	List all files including the invisible ones?

## Details

Most of the parameters are passed through `list.files()`.

## Value

A file specification object.

## Specification

- Define a list of parameters that can be passed to `list.files()`.
- Store the parameters in a named list.
- Assign class `file_spec` to the list.
- Return the `file_spec` object.

**Examples**

```
file_spec(  
  "R/",  
  pattern = "\\..R$", format = "text",  
  recursive = FALSE, ignore_case = TRUE, all_files = FALSE  
)
```

---

file\_spec\_templates    *File specification templates*

---

**Description**

- file\_root\_core() - core files under the package root
- file\_root\_all() - all files under the package root
- file\_r() - files under R/
- file\_man() - files under man/
- file\_src() - files under src/
- file\_vignettes() - files under vignettes/
- file\_data() - files under data/
- file\_tests() - files under tests/

**Usage**

```
file_root_core()
```

```
file_root_all()
```

```
file_r()
```

```
file_man()
```

```
file_src()
```

```
file_vignettes()
```

```
file_data()
```

```
file_tests()
```

**Value**

A file specification or a list of file specifications.

**Specification**

- Use file\_spec() to generate and return a file specification or a list of file specifications that cover the common naming patterns of files under directories in source R packages.



---

is\_file\_collection      *Is this a file collection object?*

---

**Description**

Is this a file collection object?

**Usage**

```
is_file_collection(object)
```

**Arguments**

object                  Any R object.

**Value**

Logical. TRUE if it is a file collection object, FALSE otherwise.

**Specification**

- Check if the object has the class file\_collection.

**Examples**

```
system.file("examples/pkg1/", package = "pkglite") %>%  
  collate(file_default()) %>%  
  is_file_collection()
```

---

is\_file\_spec              *Is this a file specification object?*

---

**Description**

Is this a file specification object?

**Usage**

```
is_file_spec(object)
```

**Arguments**

object                  Any R object

**Value**

Logical. TRUE if it is a file specification object, FALSE otherwise.

**Specification**

- Check if the input object class contains file\_spec.

**Examples**

```
file_spec(
  "R/",
  pattern = "\\..R$", format = "text",
  recursive = FALSE, ignore_case = TRUE, all_files = FALSE
) %>%
  is_file_spec()
```

---

merge.file\_collection *Merge file collections*

---

**Description**

Merge file collections

**Usage**

```
## S3 method for class 'file_collection'
merge(x, y, ...)
```

**Arguments**

x	File collection.
y	Another file collection.
...	Additional file collections.

**Value**

Merged file collection.

**Specification**

- Capture the file collection objects and store in a list.
- Check if all objects are file collection objects.
- Check if the file collections are for the same package.
- Bind the data frames from the file collections together and remove duplicated rows.
- Create a new file collection object with the merged data frame.

**Examples**

```
pkg <- system.file("examples/pkg1/", package = "pkglite")
fc1 <- pkg %>% collate(file_root_core())
fc2 <- pkg %>% collate(file_r(), file_man())
merge(fc1, fc2)
```

---

pack *Pack packages into a text file*

---

## Description

Pack packages into a text file

## Usage

```
pack(..., output, quiet = FALSE)
```

## Arguments

...	One or more file collection objects generated by <code>collate()</code> .
output	Path to the output text file. If empty, will create a txt file using the lower-cased package name in the current working directory. For multiple packages, will use "pkglite.txt".
quiet	Suppress printing of progress?

## Value

The output file path.

## Specification

- Get the package metadata, for example, package names, from the input file collection(s).
- If unspecified, generate a default output file name by the number of packages.
- Read each file in each package as DCF blocks.
- Add header and write to the output file.

## Examples

```
# pack two packages
pkg1 <- system.file("examples/pkg1", package = "pkglite")
pkg2 <- system.file("examples/pkg2", package = "pkglite")

fc1 <- pkg1 %>% collate(file_default())
fc2 <- pkg2 %>% collate(file_default())

txt <- tempfile(fileext = ".txt")
pack(fc1, fc2, output = txt, quiet = TRUE)

txt %>%
  readLines() %>%
  head() %>%
  cat(sep = "\n")
txt %>%
  readLines() %>%
  length()
```

print.file\_collection *Print a file collection*

---

**Description**

Print a file collection

**Usage**

```
## S3 method for class 'file_collection'  
print(x, ...)
```

**Arguments**

x                    An object of class file\_collection.  
...                  Additional parameters for `print()` (not used).

**Value**

The input file\_collection object.

**Specification**

- Print the metadata and the data frame in a file collection object.

**Examples**

```
fc <- system.file("examples/pkg1/", package = "pkglite") %>%  
  collate(file_default())  
fc
```

---

print.file\_spec            *Print a file specification*

---

**Description**

Print a file specification

**Usage**

```
## S3 method for class 'file_spec'  
print(x, ...)
```

**Arguments**

x                    An object of class file\_spec.  
...                  Additional parameters for `print()` (not used).

**Value**

The input `file_spec` object.

**Specification**

- Print the elements in the file specification object with `cli`.

**Examples**

```
fs <- file_spec(
  "R/",
  pattern = "\\..R$", format = "text",
  recursive = FALSE, ignore_case = TRUE, all_files = FALSE
)
fs
```

---

prune	<i>Remove files from a file collection</i>
-------	--

---

**Description**

Remove files from a file collection

**Usage**

```
prune(x, path)

## S3 method for class 'file_collection'
prune(x, path)
```

**Arguments**

<code>x</code>	File collection.
<code>path</code>	Character vector. Relative paths of the files to remove.

**Value**

Pruned file collection.

**Specification**

- Remove the rows from the data frame whose relative paths match the given paths exactly.
- Create a new file collection object with the pruned data frame.

**Examples**

```
system.file("examples/pkg1/", package = "pkglite") %>%
  collate(file_default()) %>%
  prune(path = c("NEWS.md", "man/figures/logo.png"))
```

---

remove_content	<i>Remove content lines from a pkglite file</i>
----------------	---

---

### Description

Remove content lines from a pkglite file

### Usage

```
remove_content(input, x, quiet = FALSE)
```

### Arguments

input	Path to the text file.
x	A character vector. Exactly matched lines in the file content will be removed.
quiet	Suppress printing of progress?

### Value

The input file path.

### Specification

- Read the input file as a character vector.
- Identify the line numbers that belong to the Content field by the indentation.
- Extract these lines and remove the two heading whitespaces and store as a vector.
- Find the elements that match the values in the input vector x.
- Remove the matching elements.
- Write the file back with the matching elements removed.
- If there are any matching elements and quiet = FALSE, print the line numbers being removed.

### Examples

```
pkg <- system.file("examples/pkg1", package = "pkglite")
txt <- tempfile(fileext = ".txt")

pkg %>%
  collate(file_default()) %>%
  pack(output = txt, quiet = TRUE) %>%
  remove_content(c("## New Features", "## Improvements"), quiet = TRUE)
```

---

sanitize	<i>Sanitize file collection</i>
----------	---------------------------------

---

**Description**

Remove commonly excluded files from a file collection.

**Usage**

```
sanitize(x)

## S3 method for class 'file_collection'
sanitize(x)
```

**Arguments**

x                   File collection.

**Value**

Sanitized file collection.

**Specification**

- Remove the files whose names match certain patterns from the file collection and return a sanitized file collection.

**Examples**

```
system.file("examples/pkg1/", package = "pkglite") %>%
  collate(file_default()) %>%
  sanitize()
```

---

sanitize_file_collection	<i>Sanitize file collection (deprecated)</i>
--------------------------	--

---

**Description**

Remove commonly excluded files from a file collection.

**Usage**

```
sanitize_file_collection(x)
```

**Arguments**

x                   File collection.

**Value**

Sanitized file collection.

**Specification**

- Remove the files whose names match certain patterns from the file collection and return a sanitized file collection.

**Examples**

```
system.file("examples/pkg1/", package = "pkglite") %>%
  collate(file_default()) %>%
  sanitize()
```

---

unpack	<i>Unpack packages from a text file</i>
--------	---

---

**Description**

Unpack packages from a text file

**Usage**

```
unpack(input, output = ".", install = FALSE, quiet = FALSE, ...)
```

**Arguments**

input	Path to the text file.
output	Path to the output directory. Each package is placed under a subdirectory named after the package name. Default is the current working directory.
install	Try install the unpacked package(s)?
quiet	Suppress printing of progress?
...	Additional parameters for <a href="#">remotes::install_local()</a> .

**Details**

If `install = TRUE`, the packages will be installed by the order of appearance in the input file. When internal dependencies exist between these packages, make sure they are packed in the order where the low-level dependencies appear first.

**Value**

The output directory path.



**Specification**

- Read input file.
- Construct output file paths.
- Write all files in all packages to their destination in the order of their appearance in the input file.
- Install the unpacked packages if `install = TRUE`.

**Examples**

```
# pack two packages
pkg1 <- system.file("examples/pkg1", package = "pkglite")
pkg2 <- system.file("examples/pkg2", package = "pkglite")

fc1 <- pkg1 %>% collate(file_default())
fc2 <- pkg2 %>% collate(file_default())

txt <- tempfile(fileext = ".txt")
pack(fc1, fc2, output = txt, quiet = TRUE)

# unpack the two packages
out <- file.path(tempdir(), "twopkgs")
txt %>% unpack(output = out, quiet = TRUE)

out %>%
  file.path("pkg1") %>%
  list.files()
out %>%
  file.path("pkg2") %>%
  list.files()
```

---

 verify\_ascii

*Check if a file contains only ASCII characters*


---

**Description**

Check if a file contains only ASCII characters

**Usage**

```
verify_ascii(input, quiet = FALSE)
```

**Arguments**

input	Path to the text file.
quiet	Print the elements containing non-ASCII characters?

**Value**

Logical. TRUE if the file only contains ASCII characters, FALSE otherwise.

**Specification**

- Read the file and check for non-ASCII characters in its content.
- If there are non-ASCII characters, return FALSE, otherwise TRUE.
- If quiet = FALSE and there are non-ASCII characters, print the corresponding line numbers and the non-ASCII characters.

**Examples**

```
pkg <- system.file("examples/pkg1", package = "pkglite")
txt <- tempfile(fileext = ".txt")
```

```
pkg %>%
  collate(file_default()) %>%
  pack(output = txt, quiet = TRUE) %>%
  verify_ascii()
```

# Index

collate, [2](#)  
collate(), [11](#)

ext\_binary, [3](#)  
ext\_text, [4](#)

file\_auto, [4](#)  
file\_data (file\_spec\_templates), [8](#)  
file\_default, [5](#)  
file\_ectd, [6](#)  
file\_man (file\_spec\_templates), [8](#)  
file\_name\_patterns, [6](#)  
file\_r (file\_spec\_templates), [8](#)  
file\_root\_all (file\_spec\_templates), [8](#)  
file\_root\_core (file\_spec\_templates), [8](#)  
file\_spec, [7](#)  
file\_spec\_templates, [8](#)  
file\_src (file\_spec\_templates), [8](#)  
file\_tests (file\_spec\_templates), [8](#)  
file\_vignettes (file\_spec\_templates), [8](#)

is\_file\_collection, [9](#)  
is\_file\_spec, [9](#)

list.files(), [7](#)

merge.file\_collection, [10](#)

pack, [11](#)  
pattern\_file\_root\_all  
    (file\_name\_patterns), [6](#)  
pattern\_file\_root\_core  
    (file\_name\_patterns), [6](#)  
pattern\_file\_sanitize  
    (file\_name\_patterns), [6](#)  
print(), [12](#)  
print.file\_collection, [12](#)  
print.file\_spec, [12](#)  
prune, [13](#)

remotes::install\_local(), [16](#)

remove\_content, [14](#)

sanitize, [15](#)  
sanitize\_file\_collection, [15](#)

unpack, [16](#)

verify\_ascii, [17](#)