Package 'checkhelper'

May 13, 2026

Title Deal with Check Outputs

Version 1.0.0

Description Deal with packages 'check' outputs and reduce the risk of rejection by 'CRAN' by following policies.

License MIT + file LICENSE

URL https://thinkr-open.github.io/checkhelper/, https://github.com/ThinkR-open/checkhelper

BugReports https://github.com/ThinkR-open/checkhelper/issues

Depends R (>= 4.0)

Imports cli, covr, desc, devtools, dplyr, glue, knitr, lifecycle, magrittr, pkgbuild, pkgload, purrr, rcmdcheck, roxygen2, stringi, stringr, tibble, tools, utils, whisker (>= 0.4), withr

Suggests attachment, callr, rmarkdown, testthat, usethis

VignetteBuilder knitr

Config/Needs/website ThinkR-open/thinkrtemplate

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

NeedsCompilation no

Author Vincent Guyader [aut, cre] (ORCID: <https://orcid.org/0000-0003-0671-9270>), Sebastien Rochette [aut] (ORCID: <https://orcid.org/0000-0002-1565-9313>, previous maintainer), Arthur Bréant [aut] (ORCID: <https://orcid.org/0000-0003-1668-0963>), Murielle Delmotte [aut] (ORCID: <https://orcid.org/0000-0002-1339-2424>), ThinkR [cph]

Maintainer Vincent Guyader <vincent@thinkr.fr>

Repository CRAN

Date/Publication 2026-05-13 11:30:02 UTC

Contents

`asciify_file` *Apply the asciify rewrite to a single R / R-related file*

Description

Writes the rewritten file in place (unless `dry_run = TRUE`). Files that do not contain any non-ASCII characters are skipped without rewriting, so file mtimes stay clean.

Usage

```
asciify_file(
  path,
  strategy = c("auto", "escape", "translit", "report"),
  identifiers = c("error", "warn", "skip"),
  dry_run = FALSE
)
```

Arguments

`path`	path to a file. Suffixes `.R`, `.r`, `.Rmd`, `.qmd` are handled as R source. `.Rnw` (Sweave) and any other suffix are scanned read-only - Sweave's `<<>>= ... @` chunk syntax is not handled by the rewriter.
`strategy`	one of:

- "auto" (default): per-token policy - \\uXXXX escape inside string literals (so they remain semantically equivalent and CRAN-safe); Latin-ASCII transliteration (drops diacritics, e.g. an accented e becomes plain e) inside comments and roxygen blocks (where escapes would not be interpreted).
- "escape": force \\uXXXX escape on every non-identifier token.
- "translit": force ASCII transliteration on every non-identifier token.
- "report": rewrite nothing, just return the input unchanged. Useful in conjunction with find_nonascii_tokens() for a dry run.

identifiers — what to do when a non-ASCII identifier (variable, function name, formal, slot...) is found:

- "error" (default): stop. Renaming an identifier changes the API surface and is not safe to automate.
- "warn": warn and leave the token unchanged.
- "skip": silently leave the token unchanged.

dry_run — logical. If TRUE, report what would change but do not write the file. Default FALSE.

Value

invisibly, a list with:

- path: path of the file
- changed: TRUE if the file was rewritten (or would be in dry-run)
- n_chars: number of non-ASCII characters in the file
- n_tokens: number of distinct source locations to rewrite (an accented string literal counts once regardless of how many non-ASCII characters it contains)
- text: rewritten content if changed, else the original

---

asciify_r_source *Rewrite non-ASCII characters inside a string of R source code*

Description

Rewrite non-ASCII characters inside a string of R source code

Usage

```
asciify_r_source(
  text,
  strategy = c("auto", "escape", "translit", "report"),
  identifiers = c("error", "warn", "skip")
)
```

Arguments

`text`	character(1), the full source of an R script.
`strategy`	one of: • `"auto"` (default): per-token policy - `\\uXXXX` escape inside string literals (so they remain semantically equivalent and CRAN-safe); `Latin-ASCII` transliteration (drops diacritics, e.g. an accented e becomes plain e) inside comments and roxygen blocks (where escapes would not be interpreted). • `"escape"`: force `\\uXXXX` escape on every non-identifier token. • `"translit"`: force ASCII transliteration on every non-identifier token. • `"report"`: rewrite nothing, just return the input unchanged. Useful in conjunction with `find_nonascii_tokens()` for a dry run.
`identifiers`	what to do when a non-ASCII identifier (variable, function name, formal, slot...) is found: • `"error"` (default): stop. Renaming an identifier changes the API surface and is not safe to automate. • `"warn"`: warn and leave the token unchanged. • `"skip"`: silently leave the token unchanged.

Details

This function does **not** touch identifiers, even with `identifiers = "skip"`: CRAN's policy is to forbid non-ASCII identifiers, but rewriting them automatically is unsafe (it would silently rename the user's exported API). Use `find_nonascii_tokens()` to surface them.

Strings declared with the R 4.0 raw form (`r"(...)"`, `R"---(...)---"`) are detected - by default they are still treated like regular `STR_CONST` (escaped); pass `strategy = "translit"` if you want to keep them raw and lose the accents instead.

Value

character(1), the rewritten source code. The original is returned unchanged if no non-ASCII tokens are found.

Examples

```
src <- '
# accent dans un commentaire: ete
x <- "deja vu"
'
cat(asciify_r_source(src))
```

audit_ascii *Audit non-ASCII characters in a package*

Description

Lists every line containing non-ASCII bytes across the package files. CRAN raises a NOTE when source code or documentation contains non-ASCII characters that are not properly escaped. Wraps `find_nonascii_files()`.

Usage

```
audit_ascii(
  pkg = ".",
  scope = c("R", "tests", "vignettes", "man", "DESCRIPTION", "NAMESPACE"),
 ignore_ext = c("png", "jpg", "jpeg", "gif", "rds", "rda", "rdata", "pdf", "ico", "svg"),
  size_limit = 5e+05
)
```

Arguments

`pkg`	Path to the package to audit.
`scope`	Character vector of subdirectories / files to scan.
`ignore_ext`	Extensions to skip (binary assets, snapshots).
`size_limit`	Skip files larger than this many bytes.

Value

A data frame with columns `file`, `line`, `text`, `n_tokens`.

See Also

`fix_ascii()` to apply the rewrite, `find_nonascii_files()`.

Examples

```
## Not run:
pkg <- create_example_pkg()
audit_ascii(pkg)

## End(Not run)
```

audit_check *Run R CMD check with CRAN environment*

Description

Invokes `R CMD check` with the env vars and options used by CRAN's incoming-pretest scripts, so local checks match CRAN as closely as possible. Wraps `check_as_cran()`.

Usage

```
audit_check(
  pkg = ".",
 check_output = file.path(dirname(normalizePath(pkg, mustWork = FALSE)), "check"),
  scratch = tempfile("scratch_dir"),
  Ncpus = 1,
  as_command = FALSE,
  clean_before = TRUE,
  open = FALSE,
  repos = getOption("repos")
)
```

Arguments

`pkg`	Path to the package to check.
`check_output`	Where to store check outputs.
`scratch`	Where to store temporary files.
`Ncpus`	Number of CPUs.
`as_command`	Run as Linux command line instead of in R. **Currently a no-op**: the `as_command = TRUE` branch only sets a local variable and returns invisibly without invoking the check. Keep `FALSE` (default) until that branch is implemented.
`clean_before`	Wipe `check_output` before running.
`open`	Open the check directory at the end.
`repos`	Repositories used for dependency resolution.

Value

The `rcmdcheck` results object.

See Also

`check_as_cran()`.

Examples

```
## Not run:
audit_check(".")

## End(Not run)
```

`audit_citation` *Audit* `inst/CITATION` *for old-style calls flagged by CRAN*

Description

Surfaces every call to `personList()`, `as.personList()` or `citEntry()` in the package's `inst/CITATION` file, with the line number and a one-line suggestion of the modern equivalent. CRAN rejects new submissions whose CITATION file still uses these (`Package CITATION file contains call(s) to old-style ...`).

Usage

```
audit_citation(pkg = ".")
```

Arguments

`pkg`	Path to the package to audit.

Details

Detection is purely static (parse + token walk via `utils::getParseData()`), so it does not source the file and never executes user code.

Value

A tibble with columns `call`, `line`, `suggestion`. Empty when the file is modern, or when there is no `inst/CITATION`.

Examples

```
## Not run:
audit_citation(".")

## End(Not run)
```

audit_dataset_doc *Audit dataset documentation*

Description

Lists every dataset under `data/` and reports whether a roxygen documentation file is found in `R/`. CRAN raises a NOTE for undocumented datasets.

Usage

```
audit_dataset_doc(pkg = ".")
```

Arguments

`pkg`	Path to the package to audit.

Value

A tibble with columns `name` and `has_doc`.

See Also

`fix_dataset_doc()`.

Examples

```
## Not run:
audit_dataset_doc(".")

## End(Not run)
```

audit_description *Audit unquoted package names in DESCRIPTION's* `Description` *field*

Description

CRAN policy: package names (and software names in general) inside the `Description` field of a `DESCRIPTION` file must be wrapped in single quotes (e.g. `'jsonlite'`, `'httr'`). An unquoted package name produces the `Package names should be quoted in the Description field` warning on CRAN incoming pretest.

Usage

```
audit_description(pkg = ".")
```

Arguments

pkg	Path to the package to audit.

Details

audit_description() reads the Description field, tokenises it, and surfaces every word that matches an installed package name yet is not wrapped in single quotes. Detection is purely static: no package is loaded, no namespace is touched.

Value

A tibble with columns word, position and suggestion. Empty when every match is already quoted, when the Description field is plain prose, or when the package has no DESCRIPTION.

Examples

```
## Not run:
audit_description(".")

## End(Not run)
```

---

audit_dontrun *Audit* \dontrun{} *blocks across the package's Rd files*

Description

Surfaces every \dontrun{} block in man/*.Rd, with the source Rd file, the documented topic, the line number, and a one-line suggestion of the modern equivalent. CRAN policy is that \dontrun{} should only wrap example code that genuinely cannot be executed (missing API key, missing system dependency, side effect on the user's filespace). The contributor should otherwise use \donttest{}, which still gets exercised by R CMD check --run-donttest but is skipped by default.

Usage

```
audit_dontrun(pkg = ".")
```

Arguments

pkg	Path to the package to audit.

Details

Detection is purely static: each Rd file is read line-by-line and the literal \dontrun{ opener is matched (commented-out forms starting with % are ignored). The function never sources the file and never executes user code.

Value

A tibble with columns `rd_file`, `topic`, `line`, `suggestion`. Empty when the package has no `\dontrun{}` blocks, or when there is no `man/` directory.

Examples

```
## Not run:
audit_dontrun(".")

## End(Not run)
```

`audit_downloads` *Audit calls to known download / network functions*

Description

CRAN policy: package code that downloads files or hits the network at install / runtime must degrade gracefully when the network is unavailable (offline build farms, sandboxed CI, locked-down user environment). Common rejection causes: downloads from inside `.onLoad()`, `.onAttach()`, vignettes or examples that have no `tryCatch()` / `skip_if_offline()` / `\dontrun{}` guard.

Usage

```
audit_downloads(pkg = ".")
```

Arguments

`pkg`	Path to the package to audit.

Details

`audit_downloads()` walks `R/`, `tests/`, `vignettes/` and `inst/` and surfaces every call to a known download or HTTP function so the dev can review each one. Detection is purely static: each file is parsed and the AST is walked; nothing is sourced.

Value

A tibble with columns `file`, `line`, `function` and `suggestion`. Empty when no download call is found, or when the package has none of the watched directories.

Examples

```
## Not run:
audit_downloads(".")

## End(Not run)
```

audit_globals *Audit globals to declare in R/globals.R*

Description

Runs `R CMD check` on the package and extracts every `no visible binding for global variable` and `no visible global function` note. Use `fix_globals()` to print or write the corresponding `globalVariables()` block. Wraps `get_no_visible()`.

Usage

```
audit_globals(pkg = ".", checks = NULL)
```

Arguments

pkg	Path to the package to audit.
checks	Optional. A pre-computed `rcmdcheck::rcmdcheck()` result (a list with at least a `notes` element). When supplied, `audit_globals()` reuses it instead of re-running `R CMD check` on `pkg`. Useful to share a single check between `audit_globals()` and `fix_globals()`.

Value

A list with three tibbles, `globalVariables`, `functions` and `operators`, or `NULL` if the package has no global notes. `globalVariables` collects names that need a `utils::globalVariables()` declaration; `functions` collects external functions to import via `@importFrom`; `operators` collects NSE tokens / data.table / rlang pronouns (`:=`, `.SD`, `.data`, `!!`, ...) that also need `@importFrom` rather than a `globalVariables()` entry.

See Also

`fix_globals()`, `get_no_visible()`.

Examples

```
## Not run:
pkg <- create_example_pkg()

# One-shot:
audit_globals(pkg)

# Shared check (avoid running R CMD check twice):
chk <- rcmdcheck::rcmdcheck(pkg)
audit_globals(pkg, checks = chk)
fix_globals(pkg, checks = chk)

## End(Not run)
```

audit_tags *Audit roxygen tags expected by CRAN*

Description

Reports exported functions that lack `@return`, and documented internal functions that lack `@noRd` (these trigger CRAN's `Please add \value to .Rd files` message). Wraps `find_missing_tags()`.

Usage

```
audit_tags(pkg = ".")
```

Arguments

pkg	Path to the package to audit.

Value

A list with three tibbles: `package_doc`, `data`, `functions`.

See Also

`find_missing_tags()`

Examples

```
## Not run:
pkg <- create_example_pkg()
audit_tags(pkg)

## End(Not run)
```

audit_userspace *Audit files left in user space by checks*

Description

Runs the package's tests, examples, vignettes and full check, and lists files that were created or modified outside the check directory. CRAN raises a NOTE for "non-standard things in the check directory". Wraps `check_clean_userspace()`.

Usage

```
audit_userspace(pkg = ".", check_output = tempfile("dircheck"))
```

Arguments

pkg	Path to the package to audit.
check_output	Where to store check outputs (defaults to a tempfile).

Value

A tibble of leaked files with columns source, problem, where, file.

See Also

check_clean_userspace().

Examples

```
## Not run:
pkg <- create_example_pkg()
audit_userspace(pkg)

## End(Not run)
```

---

check_n_covr *Run R CMD check and code coverage in one pass*

Description

Splits the work between devtools::check(args = "--no-tests") (the static / install / vignette parts of R CMD check) and covr::package_coverage(type = "tests") (the test runner, with coverage instrumentation). The unit tests run exactly once, on the coverage side, instead of twice (once for the check, once for the coverage). On a package with a slow suite this halves the wait.

Usage

```
check_n_covr(pkg = ".", args = character(0), quiet = TRUE)
```

Arguments

pkg	Path to the package.
args	Character vector of extra args passed to devtools::check(). --no-tests is always prepended; pass --as-cran, --no-manual etc. here.
quiet	Logical. Forwarded to both inner runners (devtools::check(quiet = ...) and covr::package_coverage(quiet = ...)). Default TRUE.

Value

A named list with two elements:

- check: the rcmdcheck result returned by devtools::check().
- coverage: the package_coverage object returned by covr::package_coverage().

Examples

```
## Not run:
res <- check_n_covr(".")
res$check
covr::percent_coverage(res$coverage)

## End(Not run)
```

`create_example_pkg` *Create a package example producing notes and errors*

Description

Create a package example producing notes and errors

Usage

```
create_example_pkg(
  path = tempfile(pattern = "pkg-"),
  with_functions = TRUE,
  with_extra_notes = FALSE,
  with_nonascii = FALSE,
  with_undocumented_data = FALSE
)
```

Arguments

`path`	Path where to store the example package
`with_functions`	Logical. Whether there will be functions or not (with notes)
`with_extra_notes`	Logical. Whether there are extra notes or not
`with_nonascii`	Logical. If `TRUE`, copy a fixture file containing non-ASCII characters (French comments, string literals, message text) so `audit_ascii()` / `fix_ascii()` have something to surface.
`with_undocumented_data`	Logical. If `TRUE`, save a small `data.frame` to `data/` *without* writing a roxygen block for it, so `audit_dataset_doc()` flags it as undocumented.

Value

Path where the example package is stored.

Examples

```
create_example_pkg()
```

find_nonascii_tokens *Find non-ASCII tokens inside a piece of R source code*

Description

Parses `text` with `base::parse()` and `utils::getParseData()` and returns the token rows whose source text is not pure ASCII. Used as the building block of `asciify_r_source()` and `asciify_pkg()`.

Usage

```
find_nonascii_tokens(text)
```

Arguments

`text`	character(1), R source code (one element, possibly with embedded newlines).

Details

Compared to a hand-rolled regex (e.g. the one used by `dreamRs/prefixer`), this catches every relevant context exactly once: string literals, comments, identifiers, numeric literals, etc., without false matches on lookalike characters that appear inside larger tokens.

Value

a data.frame, the subset of `getParseData()` whose `text` field contains at least one non-ASCII byte. An extra logical column `is_identifier` flags symbol-like tokens that should not be auto-rewritten.

See Also

`asciify_r_source()` to apply the rewrite, `find_nonascii_files()` to scan a whole directory.

fix_ascii *Rewrite non-ASCII characters in a package*

Description

Escapes non-ASCII string literals to `\uXXXX` and transliterates comments / roxygen so the package passes CRAN's "non-ASCII characters" check. Dry-run by default: pass `dry_run = FALSE` to actually rewrite files. Wraps `asciify_pkg()`.

Usage

```
fix_ascii(
  pkg = ".",
  scope = c("R", "tests", "vignettes"),
  strategy = c("auto", "escape", "translit", "report"),
  identifiers = c("error", "warn", "skip"),
  dry_run = TRUE
)
```

Arguments

pkg	Path to the package to rewrite.
scope	Subdirectories to rewrite.
strategy	Rewrite strategy (see asciify_r_source()).
identifiers	What to do when a non-ASCII identifier is found.
dry_run	If TRUE (default), only report what would change.

Value

Invisibly, a data frame of the changes per file.

See Also

audit_ascii(), asciify_pkg().

Examples

```
## Not run:
pkg <- create_example_pkg()
fix_ascii(pkg, dry_run = TRUE)

## End(Not run)
```

---

fix_dataset_doc *Generate a roxygen skeleton for a dataset*

Description

Writes R/{prefix}{name}.R with a roxygen documentation skeleton for the data/{name}.rda dataset. Wraps use_data_doc().

Usage

```
fix_dataset_doc(
  name,
  pkg = ".",
  prefix = "doc_",
  description = "Description",
  source = "Source",
  overwrite = FALSE
)
```

Arguments

name	Name of the dataset (without extension).
pkg	Path to the package.
prefix	Prefix for the generated R file. Defaults to "doc_".
description	Description shown in the roxygen block.
source	Source attribution shown in the roxygen block.
overwrite	If FALSE (default), error when the doc file already exists. Set TRUE to regenerate it in place.

Value

Invisibly, the path of the generated file.

See Also

audit_dataset_doc(), use_data_doc().

Examples

```
## Not run:
fix_dataset_doc("my_data", description = "My data", source = "Internal")

## End(Not run)
```

---

fix_globals *Print or write the globalVariables block to declare*

Description

Convenience wrapper that runs the audit, then either prints the globalVariables(...) block to console (default) or writes it to R/globals.R. Wraps print_globals().

Usage

```
fix_globals(pkg = ".", write = FALSE, checks = NULL)
```

Arguments

pkg	Path to the package.
write	If TRUE, write a single globalVariables(...) call to <pkg>/R/globals.R, **merging** with whatever names that file already declares (the freshly detected names are added on top of the existing ones, deduplicated). Default FALSE (print the block to the console for manual paste).
checks	Optional. A pre-computed rcmdcheck::rcmdcheck() result (a list with at least a notes element). When supplied, fix_globals() reuses it instead of re-running R CMD check on pkg. Useful to share a single check between audit_globals() and fix_globals().

Value

Invisibly, the path written (when write = TRUE) or NULL.

See Also

audit_globals(), print_globals().

Examples

```
## Not run:
pkg <- create_example_pkg()
fix_globals(pkg)
fix_globals(pkg, write = TRUE)

# Reuse a single R CMD check across both audit and fix:
chk <- rcmdcheck::rcmdcheck(pkg)
audit_globals(pkg, checks = chk)
fix_globals(pkg, checks = chk, write = TRUE)

## End(Not run)
```

Index