# Package 'checktor'

July 2, 2026

**Title** Extra CRAN Submission Checks

**Version** 0.1.0

**Description** Provides automated checks for common Comprehensive R Archive Network (CRAN) submission issues that are not caught by standard 'R CMD check'. Consolidates ad-hoc requirements that CRAN reviewers enforce but standard checks do not surface, helping 'R' package maintainers identify and fix issues before submission to reduce rejection rates. Covers code-pattern issues, DESCRIPTION-field formatting, documentation problems, and general package structure concerns.

**License** AGPL (>= 3)

**URL** https://github.com/coatless-rpkg/checktor, https://r-pkg.thecoatlessprofessor.com/checktor/

**BugReports** https://github.com/coatless-rpkg/checktor/issues

**Depends** R (>= 3.5.0)

**Imports** utils, tools, cli (>= 3.0.0), generics, xml2, xmlparsedata

**Suggests** testthat (>= 3.0.0), quarto

**VignetteBuilder** quarto

**Encoding** UTF-8

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** James Joseph Balamuta [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-2826-8458>)

**Maintainer** James Joseph Balamuta <james.balamuta@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-07-02 18:50:02 UTC

# Contents



---

`checktor` *Diagnose Package for CRAN Submission Issues*

---

## Description

Runs a comprehensive diagnostic suite for common CRAN submission issues that are not caught by standard R CMD check. Like a doctor for your package, this function examines your code, DESCRIPTION file, documentation, general package structure, and CRAN policy compliance to identify potential problems that could cause CRAN submission delays or rejections.

## Usage

```
checktor(
  path = ".",
  verbose = getOption("checktor.verbose", TRUE),
  progress = getOption("checktor.progress", verbose)
)
```

## Arguments

| | |
|---|---|
| path | Character. Path to the R package directory. Defaults to current directory (".") |
| verbose | Logical. Whether to print detailed diagnostic output to console. Defaults to getOption("checktor.verbose", TRUE). |
| progress | Logical. Whether to show progress bars during diagnostics. Defaults to getOption("checktor.progres verbose). |

## Details

The function runs five categories of diagnostics: **Code**, **DESCRIPTION**, **Documentation**, **General**, and **Policy**. See diagnose_code_issues(), diagnose_description_issues(), diagnose_documentation_issues diagnose_general_issues(), and diagnose_policy_violations() for the specific checks within each category.

The metadata$total_issues figure counts the total number of distinct issues found across all checks (e.g., 80 lines using T/F count as 80, not 1). The metadata$failed_checks figure counts how many individual checks reported any issue at all.

## Value

A checktor_results object (list) containing:

- code_issues: Results from code diagnostics
- description_issues: Results from DESCRIPTION file diagnostics
- documentation_issues: Results from documentation diagnostics
- general_issues: Results from general package diagnostics
- policy_issues: Results from CRAN policy violation diagnostics
- metadata: List with package path, diagnosis time, total issue count, total failed-check count, and checktor version

Each diagnostic category contains a passed element showing which individual checks passed/failed, plus detailed results for each check.

## See Also

health_report() to generate detailed reports, prescribe() for treatment recommendations, checkup() for quick health checks

## Examples

```
# Run against a synthetic package with known T/F issues
pkg <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                 show_content = FALSE)
results <- checktor(pkg, verbose = FALSE, progress = FALSE)

results              # the diagnosis summary
summary(results)     # per-category overview
issues(results)      # every issue as a tidy data frame
is_healthy(results)  # FALSE
```

---

`checktor_category_result` *Create a Multi-Category Diagnostic Result Object*

---

## Description

Constructor function for creating diagnostic category result objects used by multi-category diagnostic functions like `diagnose_code_issues()`.

## Usage

```
checktor_category_result(...)
```

## Arguments

| | |
|---|---|
| `...` | Named arguments where each is a checktor_check_result object representing individual checks within the category. |

## Value

An object of class `checktor_category_result` containing:

- Individual checktor_check_result objects for each check
- `passed`: Named logical vector showing which individual checks passed

## See Also

Multi-category functions like `diagnose_code_issues()`, `diagnose_documentation_issues()`

## Examples

```
# Create individual check results
tf_check <- checktor_check_result(FALSE, "file.R:5", "T/F usage check")
seed_check <- checktor_check_result(TRUE, character(0), "Seed setting check")

# Create category result
code_results <- checktor_category_result(
  tf_usage = tf_check,
```

```
  seed_setting = seed_check
)
print(code_results)
```

---

`checktor_check_result` *Create a Standard Diagnostic Check Result Object*

---

## Description

Constructor function for creating consistent diagnostic check result objects used by all individual diagnostic functions.

## Usage

```
checktor_check_result(passed, issues, message, ...)
```

## Arguments

| | |
|---|---|
| `passed` | Logical. TRUE if the check passed, FALSE if issues were found. |
| `issues` | Character vector. Specific issues found, typically in "file:line" format. |
| `message` | Character. Description of what was checked. |
| `...` | Additional named elements specific to the particular check. |

## Value

An object of class `checktor_check_result` containing:

- `passed`: The passed status
- `issues`: Vector of issues found
- `message`: Description of the check
- Additional elements passed via `...`

## See Also

Individual diagnostic functions like `diagnose_tf_usage()`, `diagnose_seed_setting()`

## Examples

```
# Create a passing check result
result <- checktor_check_result(
  passed = TRUE,
  issues = character(0),
  message = "Example check"
)
print(result)

# Create a failing check result with additional elements
```

```
result <- checktor_check_result(
  passed = FALSE,
  issues = c("file1.R:5", "file2.R:10"),
  message = "T/F usage check",
  file_issues = list("file1.R" = 5, "file2.R" = 10)
)
print(result)
```

---

checkup *Quick Health Check*

---

## Description

Runs `checktor()` with minimal output, suitable for CI/CD pipelines.

## Usage

```
checkup(path = ".")
```

## Arguments

| | |
|---|---|
| `path` | Character. Path to the R package directory. Default: `"."`. |

## Value

Logical. `TRUE` if no issues were found, `FALSE` otherwise.

## Examples

```
# A clean synthetic package passes; a known-bad one does not
pkg_bad <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                     show_content = FALSE)
checkup(pkg_bad)
```

---

configure_doctor *Configure Package Doctor Defaults*

---

## Description

Sets session-wide defaults for `checktor()` behavior. Subsequent calls to `checktor()` (and helpers that delegate to it) pick up these defaults via `getOption()`.

## Usage

```
configure_doctor(verbose_default = TRUE, progress_default = TRUE, color = TRUE)
```

### Arguments

| | |
|---|---|
| verbose_default | Logical. Default verbosity for `checktor()`. |
| progress_default | Logical. Default progress-bar setting. |
| color | Logical. Whether `cli` should emit ANSI color. Sets `cli.num_colors` via `options()`. |

### Value

Invisibly returns the previous values of the changed options, so the call can be reversed with `options(.)`.

### Examples

```
# Save defaults so we can restore them after the example runs
old <- options(checktor.verbose = NULL, checktor.progress = NULL)
on.exit(options(old), add = TRUE)

configure_doctor(verbose_default = FALSE)
getOption("checktor.verbose")
```

---

## diagnose_code_issues *Diagnose Code Health Issues*

---

### Description

Runs comprehensive diagnostics on R source code to identify common CRAN submission issues and coding best-practice violations.

### Usage

```
diagnose_code_issues(path = ".", verbose = TRUE)
```

### Arguments

| | |
|---|---|
| path | Character. Path to the R package directory. Default: `"."`. |
| verbose | Logical. Whether to print detailed diagnostic output. Default: `TRUE`. |

### Details

Each source file is parsed once with `parse(keep.source = TRUE)`; checks run XPath queries against the parsed XML representation, so identifiers that appear only inside string literals or comments do not false-positive. Multi-line constructs (`set.seed(\n123\n)`), formula `~` versus path `~`, and scope-aware patterns (an `options()` call guarded by a sibling `on.exit()` in the same function body) are all handled correctly.

### Value

List of named checktor_check_result() objects (e.g., tf_usage, seed_setting) plus a passed named logical vector summarizing pass/fail for each sub-check.

### See Also

checktor() for complete package diagnostics

### Examples

```
pkg <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                 show_content = FALSE)
code_results <- diagnose_code_issues(pkg, verbose = FALSE)
summary(code_results)   # per-category overview
issues(code_results)    # the issues found
```

---

## diagnose_cran_comments_file

*Diagnose a Missing cran-comments.md File*

---

### Description

A cran-comments.md file carries the submission notes CRAN reviewers read (test environments, R CMD check results, downstream-dependency notes). Its absence is flagged so it can be added before submission.

### Usage

```
diagnose_cran_comments_file(path, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| path | Character. Path to package directory |
| verbose | Logical. Print diagnostic messages |

### Details

This check is opt-in: it is **not** part of the default checktor() / diagnose_general_issues() run, because a cran-comments.md is a workflow convention rather than a CRAN requirement. Call it directly to use it.

### Value

checktor_check_result() with passed, issues, message.

## Examples

```
pkg_path <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                      show_content = FALSE)
file.remove(file.path(pkg_path, "cran-comments.md"))  # failing case
issues(diagnose_cran_comments_file(pkg_path, verbose = FALSE))
```

---

`diagnose_description_issues` *Diagnose DESCRIPTION File Issues*

---

## Description

Runs diagnostics against the package DESCRIPTION file. Fields are parsed with `base::read.dcf()` so that multi-line fields like `Description` and `Title` are inspected in full, not just their first physical line.

## Usage

```
diagnose_description_issues(path = ".", verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `path` | Character. Path to the R package directory. Default: `"."`. |
| `verbose` | Logical. Whether to print diagnostic output. Default: `TRUE`. |

## Value

List containing one named element per check. Each element is a list with at least `passed`, `issues`, and `message` (see `checktor_check_result()`).

## See Also

`checktor()` for complete package diagnostics

## Examples

```
pkg_path <- example_diagnose_scenario("description_examples/bad_description.txt",
                                      show_content = FALSE)
results <- diagnose_description_issues(pkg_path, verbose = FALSE)
issues(results)     # description-field problems, if any
```

`diagnose_documentation_issues` *Diagnose Documentation Issues*

## Description

Runs diagnostics on package documentation to identify common issues that can cause CRAN submission problems or a poor user experience.

## Usage

```
diagnose_documentation_issues(path = ".", verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `path` | Character. Path to package directory. Default: `"."`. |
| `verbose` | Logical. Print diagnostic output. Default: `TRUE`. |

## Details

This function checks for:

- Missing `\value` tags in function documentation
- Exported functions missing an `\examples` section
- Roxygen2 usage
- Example structure (appropriate use of `\dontrun{}`)
- Examples that use Suggested packages without a guard

`.Rd` files are parsed structurally via `tools::parse_Rd()` so analyses look at sections by their `Rd_tag` rather than grepping LaTeX text.

## Value

List of `checktor_check_result()` objects plus a `passed` named logical vector summarizing pass/fail per check.

## See Also

`checktor()` for complete package diagnostics

## Examples

```
pkg_path <- example_diagnose_scenario("documentation_examples/missing_value_tag.Rd",
                                      show_content = FALSE)
doc_results <- diagnose_documentation_issues(pkg_path, verbose = FALSE)
summary(doc_results)
issues(doc_results)
```

## diagnose_example_structure

*Diagnose Example Structure*

### Description

Walks \examples{} sections via tools::parse_Rd() and flags \dontrun{} subtrees that don't appear to have a justifying reason (interactive, network, credentials, long-running, etc.).

### Usage

```
diagnose_example_structure(path, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| path | Character. Path to package directory |
| verbose | Logical. Print diagnostic messages |

### Value

checktor_check_result() with passed, issues, message.

### Examples

```
pkg_path <- example_diagnose_scenario("network_examples/bad_network_example.Rd",
                                      show_content = FALSE)
diagnose_example_structure(pkg_path, verbose = FALSE)
```

## diagnose_general_issues

*Diagnose General Package Issues*

### Description

Runs general diagnostics on package structure and content that don't fit into specific code, documentation, or DESCRIPTION categories.

### Usage

```
diagnose_general_issues(path = ".", verbose = TRUE)
```

### Arguments

| | |
|---|---|
| path | Character. Path to package directory. Default: ".". |
| verbose | Logical. Print diagnostic output. Default: TRUE. |

## Details

This function checks:

- Package size — measured against the files that would ship in the tarball (`.Rbuildignore` and standard scratch dirs are excluded), with a 5 MB warning threshold matching CRAN's recommendation.
- Invalid or problematic URLs in package files.
- Presence of a `NEWS` file documenting user-facing changes.
- Relative links in the `README` that would break on CRAN.

`diagnose_cran_comments_file()` is intentionally not part of this default run, since a `cran-comments.md` is a workflow convention rather than a CRAN requirement; call it directly to opt in.

## Value

List of `checktor_check_result()` objects plus a `passed` summary.

## See Also

`checktor()` for complete package diagnostics

## Examples

```
pkg_path <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                      show_content = FALSE)
general_results <- diagnose_general_issues(pkg_path, verbose = FALSE)
general_results$package_size$size_mb
```

---

# diagnose_missing_examples

*Diagnose Exported Functions Missing Examples*

---

## Description

CRAN expects exported functions to carry a runnable `\examples{}` section. Walks `.Rd` files via `tools::parse_Rd()` and reports exported function topics that lack one. Data, class, methods, package-level, and re-export topics are skipped, and only topics whose name appears in NAMESPACE `export()` are considered (so internal helpers and S3 methods aren't required to have examples). Genuinely side-effect-only functions may be false positives and can be ignored.

## Usage

```
diagnose_missing_examples(path, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `path` | Character. Path to package directory |
| `verbose` | Logical. Print diagnostic messages |

## Value

checktor_check_result() with passed, issues, missing, message.

## Examples

```
pkg_path <- example_diagnose_scenario(
  "documentation_examples/missing_examples_bad.Rd", show_content = FALSE)
writeLines("export(undocumented_fn)", file.path(pkg_path, "NAMESPACE"))
issues(diagnose_missing_examples(pkg_path, verbose = FALSE))
```

---

# diagnose_news_file *Diagnose a Missing NEWS File*

---

## Description

CRAN expects packages (especially on resubmission) to document user-facing changes in a NEWS file. Accepts NEWS.md, NEWS, or NEWS.Rd at the package root or under inst/.

## Usage

```
diagnose_news_file(path, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| path | Character. Path to package directory |
| verbose | Logical. Print diagnostic messages |

## Value

checktor_check_result() with passed, issues, message.

## Examples

```
pkg_path <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                      show_content = FALSE)
file.remove(file.path(pkg_path, "NEWS.md"))   # demonstrate the failing case
issues(diagnose_news_file(pkg_path, verbose = FALSE))
```

## diagnose_package_size *Diagnose Package Size*

### Description

Estimates the size of the source package that would be shipped to CRAN (files matched by `.Rbuildignore`, plus standard scratch directories like `.git`, `.Rproj.user`, are excluded). Warns at the 5 MB threshold.

### Usage

```
diagnose_package_size(path, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| `path` | Character. Path to package directory |
| `verbose` | Logical. Print diagnostic messages |

### Value

`checktor_check_result()` with `passed`, `issues`, `message`, and `size_mb`.

### Examples

```
pkg_path <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                      show_content = FALSE)
diagnose_package_size(pkg_path, verbose = FALSE)$size_mb
```

## diagnose_policy_violations *Check for Common CRAN Policy Violations*

### Description

Runs additional diagnostics focused on CRAN policy: leftover `browser()` calls, raw system invocations, file writes outside `tempdir()`, and unwrapped network access in examples or vignettes. Code-side checks use the parsed AST so string/comment matches don't false-positive; Rd-side checks use `tools::parse_Rd()` for the same reason.

### Usage

```
diagnose_policy_violations(path = ".", verbose = TRUE)
```

### Arguments

| | |
|---|---|
| `path` | Character. Path to the R package directory. Default: `"."`. |
| `verbose` | Logical. Whether to print diagnostic output. Default: `TRUE`. |

## Value

List of checktor_check_result() objects, plus a passed named logical vector summarizing pass/fail per check.

## See Also

checktor() for complete package diagnostics

## Examples

```
pkg <- example_diagnose_scenario("code_examples/browser_calls_bad.R",
                                 show_content = FALSE)
policy <- diagnose_policy_violations(pkg, verbose = FALSE)
summary(policy)
issues(policy)
```

---

# diagnose_print_cat_usage

*Diagnose Print/Cat Usage in Functions*

---

## Description

Flags print() / cat() calls not guarded by an enclosing if(), for(), or while(). The check uses the ancestor axis, so guard detection is robust regardless of formatting.

## Usage

```
diagnose_print_cat_usage(path, verbose = TRUE, parsed = NULL)
```

## Arguments

| | |
|---|---|
| path | Character. Path to package directory. |
| verbose | Logical. Print diagnostic messages. |
| parsed | Internal. Pre-parsed source cache; if NULL, files are read from path on demand. |

## Value

checktor_check_result() with passed, issues, message.

## Examples

```
pkg <- example_diagnose_scenario("code_examples/print_cat_bad.R",
                                 show_content = FALSE)
diagnose_print_cat_usage(pkg, verbose = FALSE)
```

## diagnose_readme_relative_links

*Diagnose Relative Links in the README*

### Description

Relative links in `README.md`/`README.Rmd` render on GitHub but break on CRAN when the target is not shipped in the built tarball. This flags relative links whose target is missing on disk or excluded by `.Rbuildignore` (and therefore absent after `R CMD build`). Relative links to files that do ship (e.g. `man/figures/logo.png`) are not flagged.

### Usage

```
diagnose_readme_relative_links(path, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| `path` | Character. Path to package directory |
| `verbose` | Logical. Print diagnostic messages |

### Value

`checktor_check_result()` with `passed`, `issues`, `message`.

### Examples

```
pkg_path <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                      show_content = FALSE)
writeLines("See [the guide](docs/guide.md) for details.",
           file.path(pkg_path, "README.md"))
issues(diagnose_readme_relative_links(pkg_path, verbose = FALSE))
```

## diagnose_roxygen_usage

*Diagnose Roxygen2 Usage*

### Description

Informational check: reports whether the package appears to use roxygen2.

### Usage

```
diagnose_roxygen_usage(path, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| path | Character. Path to package directory |
| verbose | Logical. Print diagnostic messages |

### Value

checktor_check_result() with passed (always TRUE), has_roxygen, message.

### Examples

```
diagnose_roxygen_usage(".", verbose = FALSE)$has_roxygen
```

---

## diagnose_seed_setting *Diagnose Hardcoded Seed Setting*

---

### Description

Flags set.seed(<numeric>) calls. Multi-line forms are handled because the check matches the call AST node, not raw text.

### Usage

```
diagnose_seed_setting(path, verbose = TRUE, parsed = NULL)
```

### Arguments

| | |
|---|---|
| path | Character. Path to package directory. |
| verbose | Logical. Print diagnostic messages. |
| parsed | Internal. Pre-parsed source cache; if NULL, files are read from path on demand. |

### Value

checktor_check_result() with passed, issues, message.

### Examples

```
pkg <- example_diagnose_scenario("code_examples/seed_setting_bad.R",
                                 show_content = FALSE)
diagnose_seed_setting(pkg, verbose = FALSE)   # prints PASSED/FAILED
```

## diagnose_suggested_in_examples *Diagnose Suggested Packages Used in Examples Without a Guard*

### Description

Under CRAN's noSuggests check a package must work without its Suggested packages installed. This flags \examples{} that load a Suggested package (library()/require()/pkg::) in code that runs unconditionally and is not guarded by requireNamespace() / rlang::is_installed() (the form @examplesIf and if (requireNamespace(...)) produce). Usage inside \dontrun{} or \donttest{} is not flagged.

### Usage

```
diagnose_suggested_in_examples(path, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| path | Character. Path to package directory |
| verbose | Logical. Print diagnostic messages |

### Value

checktor_check_result() with passed, issues, message.

### Examples

```
pkg_path <- example_diagnose_scenario(
  "documentation_examples/suggested_in_examples_bad.Rd", show_content = FALSE)
cat("Suggests: somesuggest\n",
    file = file.path(pkg_path, "DESCRIPTION"), append = TRUE)
issues(diagnose_suggested_in_examples(pkg_path, verbose = FALSE))
```

## diagnose_tf_usage *Diagnose* T/F *Usage in R Code*

### Description

Flags bare T / F symbols that should be TRUE / FALSE. Operates on the parsed AST, so T inside string literals or comments is not flagged (a long-standing source of regex false positives). Named-argument names (f(T = 1)) and $T / @T extractions are excluded.

### Usage

```
diagnose_tf_usage(path, verbose = TRUE, parsed = NULL)
```

### Arguments

| | |
|---|---|
| path | Character. Path to package directory. |
| verbose | Logical. Print diagnostic messages. |
| parsed | Internal. Pre-parsed source cache; if NULL, files are read from path on demand. |

### Value

checktor_check_result() with passed, issues, message.

### Examples

```
pkg <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                 show_content = FALSE)
issues(diagnose_tf_usage(pkg, verbose = FALSE))
```

---

## diagnose_urls *Diagnose URL Issues in Package Files*

---

### Description

Checks common package files for http:// URLs (should usually be https://) and known URL shortener domains.

### Usage

```
diagnose_urls(path, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| path | Character. Path to package directory |
| verbose | Logical. Print diagnostic messages |

### Value

checktor_check_result() with passed, issues, message.

### Examples

```
pkg_path <- example_diagnose_scenario("description_examples/bad_description.txt",
                                      show_content = FALSE)
issues(diagnose_urls(pkg_path, verbose = FALSE))
```

## diagnose_value_tags *Diagnose Missing Value Tags in Documentation*

### Description

Walks `.Rd` files via `tools::parse_Rd()` and reports topics that are missing a `\value{}` section. Data, class, methods, package-level, and re-export topics are skipped (they don't need `\value{}`).

### Usage

```
diagnose_value_tags(path, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| `path` | Character. Path to package directory |
| `verbose` | Logical. Print diagnostic messages |

### Value

`checktor_check_result()` with `passed`, `issues`, `missing`, `message`.

### Examples

```
pkg_path <- example_diagnose_scenario("documentation_examples/missing_value_tag.Rd",
                                      show_content = FALSE)
issues(diagnose_value_tags(pkg_path, verbose = FALSE))
```

## example_diagnose_scenario *Create Example Diagnostic Scenario*

### Description

Creates a temporary package structure with a specified example file for testing diagnostic functions. This is primarily used in documentation examples to demonstrate diagnostic capabilities with known problematic code.

### Usage

```
example_diagnose_scenario(
  example_path,
  show_content = TRUE,
  description_type = "minimal",
  cleanup = FALSE
)
```

## Arguments

| | |
|---|---|
| `example_path` | Character. Relative path to example file within inst/diagnose/. Should include subdirectory and filename (e.g., "code_examples/tf_usage_bad.R"). |
| `show_content` | Logical. Whether to display the example file content in the console. Default: `TRUE`. |
| `description_type` | Character. Type of DESCRIPTION file to create. Options: "minimal" (basic fields only), "bad" (with known issues), "good" (properly formatted). Default: "minimal". |
| `cleanup` | Logical. Whether to register cleanup of temporary directory on exit. Default: `FALSE` (user manages cleanup). |

## Details

This function:

1. Locates the specified example file in the package's `inst/diagnose/` directory
2. Creates a temporary package directory structure
3. Copies the example file to the appropriate location
4. Optionally displays the example file content
5. Returns the path to the temporary package for diagnostic testing

The temporary package includes minimal structure (`R/`, `man/`, etc.) needed for running diagnostics, plus a basic `DESCRIPTION` file.

## Value

Character. Path to the temporary package directory containing the example file. Returns `NULL` if the example file cannot be found.

## Example File Structure

The temporary package created has this structure:

```
/tmp/checktor_example_XXXX/
|-- DESCRIPTION          # Basic or custom DESCRIPTION file
|-- R/                   # Contains copied example R files
|   `-- example.R        # The example file with issues
|-- man/                 # Empty directory for .Rd files
`-- tests/               # Empty directory for test files
```

## See Also

Used in examples for diagnostic functions like `diagnose_tf_usage()`, `diagnose_seed_setting()`, etc.

## Examples

```
# Create scenario with T/F usage issues
pkg_path <- example_diagnose_scenario("code_examples/tf_usage_bad.R")
result <- diagnose_tf_usage(pkg_path, verbose = TRUE)
issues(checktor(pkg_path, verbose = FALSE, progress = FALSE))

# Create scenario without showing file content
pkg_path <- example_diagnose_scenario("code_examples/seed_setting_bad.R",
                                      show_content = FALSE)

# Create scenario with problematic DESCRIPTION file
pkg_path <- example_diagnose_scenario("description_examples/bad_description.txt",
                                      description_type = "bad")
desc_result <- diagnose_description_issues(pkg_path)

# Manual cleanup when done
unlink(pkg_path, recursive = TRUE)

# Or use with automatic cleanup
pkg_path <- example_diagnose_scenario("code_examples/browser_calls_bad.R",
                                      cleanup = TRUE)
# Cleanup happens automatically when R session ends
```

---

# health_report *Comprehensive Health Report*

---

## Description

Creates a comprehensive report with specific treatment instructions

## Usage

```
health_report(results, file = NULL, format = "markdown")
```

## Arguments

| | |
|---|---|
| `results` | List. Results from checktor() |
| `file` | Character. Output file path (optional) |
| `format` | Character. Report format: "markdown", "html", or "text" |

## Value

Character vector with report content

## Examples

```
pkg <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                 show_content = FALSE)
results <- checktor(pkg, verbose = FALSE, progress = FALSE)
report <- health_report(results, format = "text")
head(report)
```

---

`issues` *Extract issues, checks, or a per-category summary from checktor results*

---

## Description

Plain accessors over the objects returned by `checktor()` and the `diagnose_*_issues()` functions, so you never navigate nested sublists.

## Usage

```
issues(x, ...)

## S3 method for class 'checktor_check_result'
issues(x, ...)

## S3 method for class 'checktor_category_result'
issues(x, ...)

## S3 method for class 'checktor_results'
issues(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `checktor_results`, `checktor_category_result`, or `checktor_check_result` object. |
| `...` | Unused. |

## Value

`issues()` returns a `data.frame` with one row per issue. At the results level the columns are `category`, `check`, `file`, `line`, `location`, `message`; a single category drops `category`; a single check drops `category` and `check`. A healthy object yields a 0-row frame.

## Examples

```
pkg <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                 show_content = FALSE)
results <- checktor(pkg, verbose = FALSE, progress = FALSE)
issues(results)
```

predicates *Status predicates for checktor results*

## Description

Status predicates for checktor results

## Usage

```
passed(x, ...)

## S3 method for class 'checktor_check_result'
passed(x, ...)

## S3 method for class 'checktor_category_result'
passed(x, ...)

## S3 method for class 'checktor_results'
passed(x, ...)

is_healthy(x, ...)

## S3 method for class 'checktor_check_result'
is_healthy(x, ...)

## S3 method for class 'checktor_category_result'
is_healthy(x, ...)

## S3 method for class 'checktor_results'
is_healthy(x, ...)

n_issues(x, ...)

## S3 method for class 'checktor_check_result'
n_issues(x, ...)

## S3 method for class 'checktor_category_result'
n_issues(x, ...)

## S3 method for class 'checktor_results'
n_issues(x, ...)

n_failed_checks(x, ...)

## S3 method for class 'checktor_category_result'
n_failed_checks(x, ...)
```

```
## S3 method for class 'checktor_results'
n_failed_checks(x, ...)

failed_checks(x, ...)

## S3 method for class 'checktor_category_result'
failed_checks(x, ...)

## S3 method for class 'checktor_results'
failed_checks(x, ...)
```

## Arguments

| | |
|---|---|
| x | A checktor_results, checktor_category_result, or checktor_check_result object. |
| ... | Unused. |

## Value

passed(): logical — a single value for a check, a named logical by check for a category, and a named logical by category for results. is_healthy(): a single logical. n_issues() / n_failed_checks(): integer counts. failed_checks(): character vector of failing check names (qualified "category.check" at the results level).

## Examples

```
pkg <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                 show_content = FALSE)
results <- checktor(pkg, verbose = FALSE, progress = FALSE)
is_healthy(results)
failed_checks(results)
```

---

# prescribe *Treatment Recommendations*

---

## Description

Prints specific treatment recommendations for issues found by checktor().

## Usage

```
prescribe(results)
```

## Arguments

| | |
|---|---|
| results | A checktor_results object. |

## Value

Invisibly returns NULL. Called for the side effect of printing recommendations.

## Examples

```
pkg <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                 show_content = FALSE)
results <- checktor(pkg, verbose = FALSE, progress = FALSE)
prescribe(results)
```

---

# print.checktor_category_result *Print Method for checktor_category_result Objects*

---

## Description

Print Method for checktor_category_result Objects

## Usage

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

## Arguments

| | |
|---|---|
| x | A checktor_category_result object |
| ... | Additional arguments (unused) |

## Value

Returns x invisibly

---

# print.checktor_check_result *Print Method for checktor_check_result Objects*

---

## Description

Print Method for checktor_check_result Objects

## Usage

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

### Arguments

| | |
|---|---|
| x | A checktor_check_result object |
| ... | Additional arguments (unused) |

### Value

Returns x invisibly

---

## print.checktor_results *Print Method for checktor_results Objects*

---

### Description

Provides a clean, formatted summary of diagnostic results from checktor().

### Usage

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

### Arguments

| | |
|---|---|
| x | A checktor_results object from checktor() |
| ... | Additional arguments passed to print methods (currently unused) |

### Value

Returns x invisibly. Called primarily for its side effect of printing a formatted summary to the console.

### See Also

checktor() to generate results, health_report() for detailed reports

### Examples

```
pkg <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                 show_content = FALSE)
results <- checktor(pkg, verbose = FALSE, progress = FALSE)
print(results)
```

show_example_files *Show Available Example Files*

## Description

Lists all available example files in the `inst/diagnose/` directory that can be used with `example_diagnose_scenario()`.

## Usage

```
show_example_files(category = "all", pattern = NULL)
```

## Arguments

| | |
|---|---|
| `category` | Character. Optional category filter. One of "code", "description", "documentation", "network", "temp", or "all". Default: "all". |
| `pattern` | Character. Optional regex pattern to filter filenames. Default: `NULL` (no filtering). |

## Value

Character vector of relative paths to example files that can be used with `example_diagnose_scenario()`.

## See Also

`example_diagnose_scenario()` to create test scenarios with these files

## Examples

```
# List all available examples
show_example_files()

# List only code examples
show_example_files("code")

# List files matching a pattern
show_example_files(pattern = "bad")

# Use with example_diagnose_scenario
examples <- show_example_files("code")
pkg_path <- example_diagnose_scenario(examples[1])
```

## summary.checktor_category_result *Per-category summary of checktor results*

### Description

Per-category summary of checktor results

### Usage

```
## S3 method for class 'checktor_category_result'
summary(object, ...)

## S3 method for class 'checktor_results'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | A `checktor_results` or `checktor_category_result` object. |
| `...` | Unused. |

### Value

For results: a 5-row `data.frame` (`category, checks, passed, failed, issues`). For a category: a 1-row `data.frame` (`checks, passed, failed, issues`).

### Examples

```
pkg <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                 show_content = FALSE)
results <- checktor(pkg, verbose = FALSE, progress = FALSE)
summary(results)
```

## tidy.checktor_results *Tidy a checktor result into a per-check data frame*

### Description

Tidy a checktor result into a per-check data frame

## Usage

```
## S3 method for class 'checktor_results'
tidy(x, ...)

## S3 method for class 'checktor_category_result'
tidy(x, ...)

## S3 method for class 'checktor_results'
as.data.frame(x, ...)

## S3 method for class 'checktor_category_result'
as.data.frame(x, ...)
```

## Arguments

| | |
|---|---|
| x | A checktor_results or checktor_category_result object. |
| ... | Unused. |

## Value

A data.frame with one row per check: category (results level only), check, passed, n_issues, message.

## Examples

```
pkg <- example_diagnose_scenario("code_examples/tf_usage_bad.R",
                                 show_content = FALSE)
results <- checktor(pkg, verbose = FALSE, progress = FALSE)
tidy(results)
```

# Index