

Package: printtree (via r-universe)

October 12, 2026

Type Package

Title Print Directory Trees for R Projects and Folders

Version 0.3.0

Description Quickly visualize 'R' project directory structures with automatic project detection and clean tree output.

License MIT + file LICENSE

URL <https://github.com/PrigasG/printtree>,
<https://prigasg.github.io/printtree/>

BugReports <https://github.com/PrigasG/printtree/issues>

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0), webshot2, withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Repository <https://prigasg.r-universe.dev>

Date/Publication 2026-10-12 20:57:33 UTC

RemoteUrl <https://github.com/prigasg/printtree>

RemoteRef HEAD

RemoteSha 4dc5232376e551c975cb1b3788eaea44d22914f7

Contents

print_rtree	2
tree_diff_dot	4
tree_diff_mermaid	5
tree_to_dot	6

tree_to_html	7
tree_to_mermaid	8
tree_to_mindmap	9
view_mermaid	10
write_tree	12
Index	14

print_rtree	<i>Print an R Project or Directory Tree</i>
-------------	---

Description

Prints a directory tree for a given path. Optionally, detects an RStudio project (.Rproj) and can print from a project root.

Usage

```
print_rtree(  
  path = NULL,  
  ignore = c("renv", ".git", ".Rproj.user", "__pycache__", ".DS_Store", "node_modules",  
    ".Rhistory"),  
  ignore_type = c("auto", "fixed", "glob", "regex"),  
  max_depth = NULL,  
  show_hidden = FALSE,  
  project = c("auto", "root", "none"),  
  search_paths = c(".", "..", "~/Documents", "~/Projects"),  
  root_markers = c(".Rproj", "DESCRIPTION"),  
  format = c("ascii", "unicode"),  
  return_lines = FALSE,  
  quiet = FALSE,  
  count_footer = TRUE,  
  git = FALSE,  
  git_legend = TRUE,  
  prune = FALSE,  
  snapshot = FALSE,  
  snapshot_file = "tree.png",  
  snapshot_width = 800,  
  snapshot_bg = c("white", "black"),  
  snapshot_path = "."  
)
```

Arguments

path	Character. Directory path, project name, or .Rproj file. If NULL, uses current directory.
------	---

ignore	Character vector. Basenames to exclude (e.g., ".git", "renv"). With ignore_type = "auto", entries containing wildcard characters are treated as glob patterns, so values such as "*.log" or "test_*" work.
ignore_type	One of "auto", "fixed", "glob", or "regex". Controls how ignore is matched against basenames.
max_depth	Integer. Maximum depth to traverse. NULL for unlimited.
show_hidden	Logical (TRUE/FALSE). Whether to include hidden files/directories (starting with ".").
project	One of "auto", "root", "none". • "auto": alias of "none"; use path as-is (no upward search) • "root": walk upward from path to find a project root (via root_markers) and use it if found • "none": never attempt root detection; print the tree from path
search_paths	Character vector. Used only when path is not an existing directory (treated as a project name). Paths are searched in order.
root_markers	Character vector. Markers used when project = "root" to detect a root directory. Special value ".Rproj" means "any file ending in .Rproj". Common markers include "DESCRIPTION" (R package root) and "_quarto.yml" (Quarto project root).
format	One of "ascii" or "unicode". "ascii" is portable for all terminals.
return_lines	Logical. If TRUE, invisibly return the printed character vector of lines.
quiet	Logical. If TRUE, suppress console output. Useful with return_lines = TRUE.
count_footer	Logical. If TRUE, append a summary like "3 directories, 12 files".
git	Logical. If TRUE, annotate files and directories with porcelain git status markers when path is inside a Git work tree.
git_legend	Logical. If TRUE and git = TRUE, append a short legend explaining the Git status markers.
prune	Logical. If TRUE, omit directories with no displayable children.
snapshot	Logical. If TRUE, gives a visual snapshot of tree.
snapshot_file	Text. Snapshot PNG name if snapshot is set as TRUE.
snapshot_width	Integer between 1 and 15000. Default 800.
snapshot_bg	Either white or black for snapshot background. If white, tree text appears black and vice.
snapshot_path	Character. If snapshot_path is provided, the file is saved there.

Value

Invisible NULL, or a character vector of printed lines if return_lines = TRUE.

Examples

```
# Create a small example directory tree
demo <- file.path(tempdir(), "printtree-demo")
if (dir.exists(demo)) unlink(demo, recursive = TRUE)
dir.create(demo, recursive = TRUE)
dir.create(file.path(demo, "R"))
file.create(file.path(demo, "R", "hello.R"))
file.create(file.path(demo, "README.md"))

# Print the tree
print_rtree(demo)

# Limit depth
print_rtree(demo, max_depth = 1)

# Save a PNG snapshot to a temporary file
png_file <- tempfile(fileext = ".png")
print_rtree(demo, snapshot = TRUE, snapshot_file = png_file)
```

tree_diff_dot

Diff Two Directory Trees as a Graphviz DOT Graph

Description

Builds both trees with the same options and renders the difference as a Graphviz DOT graph: nodes only in after are filled green, nodes only in before are filled red and dashed, and unchanged nodes are neutral. Nodes are matched by their path relative to each tree root.

Usage

```
tree_diff_dot(
  before,
  after,
  rankdir = c("TB", "LR", "RL", "BT"),
  file = NULL,
  ...
)
```

Arguments

before	Character. Directory path for the "before" tree.
after	Character. Directory path for the "after" tree.
rankdir	Character. Graph direction: "TB" (top-bottom), "LR" (left-right), "RL", or "BT".
file	Character or NULL. If given, the diagram text is written to this file with writeLines() .
...	Additional arguments passed to build_tree() for both trees (e.g. <code>ignore</code> , <code>max_depth</code> , <code>show_hidden</code> , <code>prune</code>).

Value

Invisibly, the diagram text as a single string (or file when file is given).

Examples

```
old <- file.path(tempdir(), "printtree-diff-old")
new <- file.path(tempdir(), "printtree-diff-new")
for (d in c(old, new)) {
  if (dir.exists(d)) unlink(d, recursive = TRUE)
  dir.create(d, recursive = TRUE)
}
file.create(file.path(old, "gone.txt"))
file.create(file.path(new, "added.txt"))

cat(tree_diff_dot(old, new))

# Left-to-right layout
cat(tree_diff_dot(old, new, rankdir = "LR"))
```

tree_diff_mermaid	<i>Diff Two Directory Trees as a Mermaid Flowchart</i>
-------------------	--

Description

Builds both trees with the same options and renders the difference as a Mermaid flowchart: nodes only in after are green, nodes only in before are red and dashed, and unchanged nodes are neutral. Nodes are matched by their path relative to each tree root.

Usage

```
tree_diff_mermaid(
  before,
  after,
  direction = c("TD", "LR", "RL", "BT"),
  file = NULL,
  ...
)
```

Arguments

before	Character. Directory path for the "before" tree.
after	Character. Directory path for the "after" tree.
direction	Character. Flowchart direction: "TD" (top-down), "LR" (left-right), "RL", or "BT".
file	Character or NULL. If given, the diagram text is written to this file with writeLines() .
...	Additional arguments passed to build_tree() for both trees (e.g. <code>ignore</code> , <code>max_depth</code> , <code>show_hidden</code> , <code>prune</code>).

Value

Invisibly, the diagram text as a single string (or file when file is given).

Examples

```
old <- file.path(tempdir(), "printtree-diff-old")
new <- file.path(tempdir(), "printtree-diff-new")
for (d in c(old, new)) {
  if (dir.exists(d)) unlink(d, recursive = TRUE)
  dir.create(d, recursive = TRUE)
}
file.create(file.path(old, "gone.txt"))
file.create(file.path(new, "added.txt"))
file.create(file.path(old, "kept.txt"))
file.create(file.path(new, "kept.txt"))

cat(tree_diff_mermaid(old, new))

# Left-to-right layout
cat(tree_diff_mermaid(old, new, direction = "LR"))
```

tree_to_dot

Convert a Directory Tree to a Graphviz DOT Graph

Description

Builds the directory tree with the same options as `build_tree()` and returns it as **Graphviz** DOT text. The output renders in Quarto documents (in a `{dot}` chunk when Graphviz is installed) and with any DOT-compatible tool. Directories use the folder shape and files use the note shape.

Usage

```
tree_to_dot(
  path = NULL,
  rankdir = c("TB", "LR", "RL", "BT"),
  file = NULL,
  git_colors = FALSE,
  repo_url = NULL,
  repo_branch = "main",
  ...
)
```

Arguments

path	Character. Directory path, project name, or .Rproj file. If NULL, uses the current directory.
rankdir	Character. Graph direction: "TB" (top-bottom), "LR" (left-right), "RL", or "BT".
file	Character or NULL. If given, the diagram text is written to this file with <code>writelnLines()</code> .

git_colors	Logical. If TRUE, nodes are filled by their Git status (amber for modified, green for untracked, blue for staged). Implies git = TRUE unless git is set explicitly.
repo_url	Character or NULL. Base URL of the Git repository, e.g. "https://github.com/user/repo". If given, every node gets a URL attribute linking to the file (blob) or directory (tree) on repo_branch, so SVG output is clickable.
repo_branch	Character. Branch used to build the repo_url links.
...	Additional arguments passed to build_tree() (e.g. ignore, max_depth, show_hidden, prune).

Value

Invisibly, the diagram text as a single string (or file when file is given). Use `cat()` to print it.

Examples

```
demo <- file.path(tempdir(), "printtree-dot-demo")
if (dir.exists(demo)) unlink(demo, recursive = TRUE)
dir.create(file.path(demo, "R"), recursive = TRUE)
file.create(file.path(demo, "R", "hello.R"))
file.create(file.path(demo, "README.md"))

cat(tree_to_dot(demo))

# Left-to-right layout with clickable nodes
cat(tree_to_dot(demo, rankdir = "LR",
                repo_url = "https://github.com/PrigasG/printtree"))
```

tree_to_html	<i>Convert a Directory Tree to a Collapsible HTML Page</i>
--------------	--

Description

Builds the directory tree with the same options as `build_tree()` and returns it as a self-contained HTML document: directories expand and collapse on click, with no external dependencies. When the tree is built with `git = TRUE`, Git status badges are shown next to changed entries.

Usage

```
tree_to_html(path = NULL, title = NULL, file = NULL, git_colors = FALSE, ...)
```

Arguments

path	Character. Directory path, project name, or .Rproj file. If NULL, uses the current directory.
title	Character or NULL. Page title and heading. Defaults to "Directory tree".
file	Character or NULL. If given, the HTML is written to this file with <code>writelnLines()</code> .

git_colors	Logical. If TRUE, nodes are colored by their Git status (badges for modified, untracked, and staged files). Implies git = TRUE.
...	Additional arguments passed to build_tree() (e.g. ignore, max_depth, show_hidden, prune, git).

Value

Invisibly, the HTML document as a single string (or file when file is given).

Examples

```
demo <- file.path(tempdir(), "printtree-html-demo")
if (dir.exists(demo)) unlink(demo, recursive = TRUE)
dir.create(file.path(demo, "R"), recursive = TRUE)
file.create(file.path(demo, "R", "hello.R"))

html <- tree_to_html(demo, title = "Demo project")
out <- tempfile(fileext = ".html")
tree_to_html(demo, file = out)

# Inside a Git checkout, git_colors = TRUE adds status badges
# next to modified, untracked, and staged files
```

tree_to_mermaid

Convert a Directory Tree to a Mermaid Flowchart

Description

Builds the directory tree with the same options as build_tree() and returns it as **Mermaid** flowchart text. The output renders natively in Quarto documents (in a {mermaid} chunk), GitHub Markdown, and anywhere else Mermaid is supported. Directories use the stadium shape and files use rectangles.

Usage

```
tree_to_mermaid(
  path = NULL,
  direction = c("TD", "LR", "RL", "BT"),
  file = NULL,
  git_colors = FALSE,
  subgraph = FALSE,
  repo_url = NULL,
  repo_branch = "main",
  ...
)
```

Arguments

path	Character. Directory path, project name, or .Rproj file. If NULL, uses the current directory.
direction	Character. Flowchart direction: "TD" (top-down), "LR" (left-right), "RL", or "BT".
file	Character or NULL. If given, the diagram text is written to this file with <code>writelines()</code> .
git_colors	Logical. If TRUE, nodes are colored by their Git status (amber for modified, green for untracked, blue for staged). Implies git = TRUE unless git is set explicitly.
subgraph	Logical. If TRUE, each directory is wrapped in a Mermaid subgraph block so directories render as visual containers.
repo_url	Character or NULL. Base URL of the Git repository, e.g. "https://github.com/user/repo". If given, every node becomes clickable and links to the file (blob) or directory (tree) on repo_branch.
repo_branch	Character. Branch used to build the repo_url links.
...	Additional arguments passed to <code>build_tree()</code> (e.g. ignore, max_depth, show_hidden, prune).

Value

Invisibly, the diagram text as a single string (or file when file is given). Use `cat()` to print it, e.g. into a Quarto `{mermaid}` chunk.

Examples

```
demo <- file.path(tempdir(), "printtree-mermaid-demo")
if (dir.exists(demo)) unlink(demo, recursive = TRUE)
dir.create(file.path(demo, "R"), recursive = TRUE)
file.create(file.path(demo, "R", "hello.R"))
file.create(file.path(demo, "README.md"))

cat(tree_to_mermaid(demo))

# Directories as visual containers, laid out left to right
cat(tree_to_mermaid(demo, subgraph = TRUE, direction = "LR"))

# Nodes that link into the repository on GitHub
cat(tree_to_mermaid(demo,
  repo_url = "https://github.com/PrigasG/printtree"))
```

Description

Builds the directory tree with the same options as `build_tree()` and returns it as **Mermaid** mindmap text: a radial alternative to the flowchart produced by `tree_to_mermaid()`. The root uses the cloud shape and every level is indented beneath its parent.

Usage

```
tree_to_mindmap(path = NULL, file = NULL, ...)
```

Arguments

<code>path</code>	Character. Directory path, project name, or .Rproj file. If NULL, uses the current directory.
<code>file</code>	Character or NULL. If given, the diagram text is written to this file with <code>writelnLines()</code> .
<code>...</code>	Additional arguments passed to <code>build_tree()</code> (e.g. <code>ignore</code> , <code>max_depth</code> , <code>show_hidden</code> , <code>prune</code>).

Value

Invisibly, the diagram text as a single string (or file when file is given). Use `cat()` to print it, e.g. into a Quarto `{mermaid}` chunk.

Examples

```
demo <- file.path(tempdir(), "printtree-mindmap-demo")
if (dir.exists(demo)) unlink(demo, recursive = TRUE)
dir.create(file.path(demo, "R"), recursive = TRUE)
file.create(file.path(demo, "R", "hello.R"))

cat(tree_to_mindmap(demo))

# Save a mindmap file, ready for a Quarto {mermaid} chunk
tree_to_mindmap(demo, file = tempfile(fileext = ".mmd"))
```

view_mermaid

Preview a Directory Tree as a Mermaid Diagram

Description

Writes an HTML page that renders `tree_to_mermaid()` output as an SVG diagram in a web browser. The page loads Mermaid from its public CDN, so rendering requires an internet connection. Browser opening is always suppressed in non-interactive sessions, including package checks and automated jobs.

Usage

```
view_mermaid(
  path = NULL,
  file = tempfile("printtree-mermaid-", fileext = ".html"),
  title = "Directory tree",
  theme = c("default", "neutral", "dark", "forest", "base"),
  open = interactive(),
  pan_zoom = TRUE,
  save = NULL,
  ...
)
```

Arguments

path	Character. Directory path, project name, or .Rproj file. If NULL, uses the current directory.
file	Character. HTML output path. By default, a temporary file is used.
title	Character. Page title and heading.
theme	Mermaid theme: "default", "neutral", "dark", "forest", or "base".
open	Logical. Open the generated page when running interactively. Browser opening is skipped when <code>interactive()</code> is false, even if this is explicitly set to TRUE.
pan_zoom	Logical. Enable pan/zoom controls for navigating large diagrams (via the <code>svg-pan-zoom</code> library).
save	Character or NULL. If given, render the diagram to an image file instead of the HTML page: one of "png", "jpeg" (or "jpg"), or "pdf". Requires the webshot2 package and a Chrome/Chromium browser. When save is given, file is treated as the image output path; if file was not explicitly supplied, the path is derived from title and the save extension. A missing extension is appended; a conflicting extension is an error. PNG/JPEG capture only the diagram element; PDF uses print CSS to hide the viewer UI.
...	Additional arguments passed to <code>tree_to_mermaid()</code> , such as <code>direction</code> , <code>ignore</code> , <code>max_depth</code> , <code>git_colors</code> , or <code>subgraph</code> .

Value

Invisibly, the generated HTML file path (or the image path when save is given).

Examples

```
demo <- file.path(tempdir(), "printtree-view-demo")
if (dir.exists(demo)) unlink(demo, recursive = TRUE)
dir.create(file.path(demo, "R"), recursive = TRUE)
file.create(file.path(demo, "R", "hello.R"))
file.create(file.path(demo, "README.md"))

preview <- view_mermaid(demo, open = FALSE)
file.exists(preview)
```

write_tree

Write a Directory Tree to a Text, Markdown, Diagram, or Quarto File

Description

Builds a directory tree with the same options as [print_rtree\(\)](#) and writes it to a plain text, Markdown, diagram, or Quarto file.

Usage

```
write_tree(
  path = NULL,
  file,
  format = c("txt", "md", "mermaid", "dot", "qmd", "mindmap", "html"),
  title = NULL,
  create_dirs = TRUE,
  ...
)
```

Arguments

path	Character. Directory path, project name, or .Rproj file. If NULL, uses current directory.
file	Character. Output file path.
format	One of "txt", "md", "mermaid", "dot", "qmd", "mindmap", or "html". "mermaid" writes a Mermaid flowchart (renders in Quarto {mermaid} chunks and GitHub Markdown), "dot" writes a Graphviz DOT graph, "qmd" writes a minimal Quarto document embedding the Mermaid flowchart, "mindmap" writes a Mermaid mindmap, and "html" writes a self-contained collapsible HTML tree.
title	Optional heading: used as the Markdown heading when format = "md", as the Quarto document title when format = "qmd", and as the page title and heading when format = "html".
create_dirs	Logical. If TRUE, create the output file's parent directory when it does not exist.
...	Additional arguments passed to the underlying tree builder (the same tree options as print_rtree() , such as ignore, max_depth, git, or prune). Diagram engines also accept direction (Mermaid), rankdir (DOT), git_colors, subgraph, and repo_url/repo_branch; see tree_to_mermaid() and tree_to_dot() .

Value

Invisibly returns the output file path.

Examples

```
demo <- file.path(tempdir(), "printtree-write-demo")
if (dir.exists(demo)) unlink(demo, recursive = TRUE)
dir.create(demo, recursive = TRUE)
file.create(file.path(demo, "README.md"))

out <- tempfile(fileext = ".md")
write_tree(demo, out, format = "md")

# Mermaid flowchart and a Quarto document embedding it
write_tree(demo, tempfile(fileext = ".mmd"), format = "mermaid")
write_tree(demo, tempfile(fileext = ".qmd"), format = "qmd",
           title = "Demo project tree")

# Mermaid mindmap and a collapsible HTML tree
write_tree(demo, tempfile(fileext = ".mmd"), format = "mindmap")
write_tree(demo, tempfile(fileext = ".html"), format = "html",
           title = "Demo project tree")
```

Index

`cat()`, [7](#), [9](#), [10](#)

`interactive()`, [11](#)

`print_rtree`, [2](#)

`print_rtree()`, [12](#)

`tree_diff_dot`, [4](#)

`tree_diff_mermaid`, [5](#)

`tree_to_dot`, [6](#)

`tree_to_dot()`, [12](#)

`tree_to_html`, [7](#)

`tree_to_mermaid`, [8](#)

`tree_to_mermaid()`, [10–12](#)

`tree_to_mindmap`, [9](#)

`view_mermaid`, [10](#)

`write_tree`, [12](#)

`writelines()`, [4–7](#), [9](#), [10](#)