

Package: prefab (via r-universe)

October 12, 2026

Title Opinionated Project Scaffolding via Composable Themes

Version 0.6.0

Description Provides a composable theme system for project scaffolding. Themes are ordered lists of steps that deploy files, inline text, or execute functions, with per-file merge strategies for handling pre-existing content. Includes pre-set themes for R analysis projects, targets workflows, and Claude Code agent configuration.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

URL <https://economic.github.io/prefab/>,
<https://github.com/Economic/prefab>

BugReports <https://github.com/Economic/prefab/issues>

Config/testthat/edition 3

Imports cli, fs, glue, jsonlite, rlang, rprojroot, withr

Suggests knitr, rmarkdown, rstudioapi, testthat (>= 3.0.0), yaml

VignetteBuilder knitr

Config/Needs/website economic/epitemplate, usethis

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libuv1-dev

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

Date/Publication 2026-08-25 02:45:22 UTC

RemoteUrl <https://github.com/economic/prefab>

RemoteRef HEAD

RemoteSha b3579119edad9e150f1e3dd7b7d8f086c50d013c

Contents

claude_r_analysis	2
claude_r_package	3
claude_r_targets	3
create_project	4
from_dir	5
from_package	5
launch_project	6
load_themes	7
new_theme	8
r_analysis	9
r_targets	9
step_file	10
step_run	10
step_text	11
theme_code	11
theme_from_dir	12
use_theme	13
Index	14

claude_r_analysis	<i>Claude Code configuration theme for R analysis projects</i>
-------------------	--

Description

Creates a theme that deploys Claude Code agent settings and rules for an R analysis project.

Usage

```
claude_r_analysis(settings_json = TRUE)
```

Arguments

settings_json Logical. If TRUE (default), merges the package settings.json into .claude/settings.json.

Value

A prefab_theme object.

Examples

```
claude_r_analysis()
```

claude_r_package	<i>Claude Code configuration theme for R packages</i>
------------------	---

Description

Creates a theme that deploys Claude Code agent settings and rules for an R package.

Usage

```
claude_r_package()
```

Value

A `prefab_theme` object.

Examples

```
claude_r_package()
```

claude_r_targets	<i>Claude Code configuration theme for R targets projects</i>
------------------	---

Description

Creates a theme that deploys Claude Code agent settings and rules for an R targets project.

Usage

```
claude_r_targets(settings_json = TRUE)
```

Arguments

`settings_json` Logical. If TRUE (default), merges the package `settings.json` into `.claude/settings.json`.

Value

A `prefab_theme` object.

Examples

```
claude_r_targets()
```

create_project	Create a new project and apply a theme
----------------	--

Description

Creates a new project directory and applies a theme to it.

Usage

```
create_project(path, theme, open = rlang::is_interactive())
```

Arguments

path	Path for the new project directory. Resolved to an absolute path via <code>fs::path_abs()</code> .
theme	A <code>prefab_theme</code> object created by <code>new_theme()</code> , <code>theme_from_dir()</code> , or a pre-set theme function, or a path to a directory of template files.
open	Whether to activate the new project after creating it, following the convention of <code>usethis::create_project()</code> and <code>usethis::create_package()</code> : when TRUE, the project opens in a new session/window in RStudio/Positron, or in other editors the working directory of the active session is changed into the new project. Defaults to <code>rlang::is_interactive()</code> , so it opens interactively but stays inert in scripts, tests, and non-interactive callers.

Value

The normalized project path (invisibly).

See Also

`theme_from_dir()` for how a template directory is read.

Examples

```
## Not run:
create_project("~/projects/my-analysis", r_analysis())
create_project("my-targets-project", r_targets() + claude_r_targets())

# A directory of template files
create_project("~/projects/my-report", "~/templates/quarto-report")

## End(Not run)
```

from_dir	Create a step-builder from a local directory
----------	--

Description

Returns a step-builder function that resolves source paths relative to a local directory. The directory path is resolved to an absolute path at creation time.

Usage

```
from_dir(path)
```

Arguments

path	Path to the directory. Resolved to absolute via <code>fs:path_abs()</code> at creation time. Must exist.
------	--

Value

A function with signature `function(source, dest, strategy = "overwrite", data = NULL)` that returns a `step_file()` object.

from_package	Create a step-builder from an installed package
--------------	---

Description

Returns a step-builder function that resolves source paths relative to a package's `inst/` directory. Works with both installed packages and during development with `devtools::load_all()`.

Usage

```
from_package(package)
```

Arguments

package	Package name (string).
---------	------------------------

Value

A function with signature `function(source, dest, strategy = "overwrite", data = NULL)` that returns a `step_file()` object.

launch_project

Launch a new project from an editor keyboard shortcut

Description

Interactive project launcher: prompts for a project name via `rstudioapi::showPrompt()`, builds a path from root (and an optional dated subdirectory), and creates the project with `create_project()`. It is designed to be bound to a key chord so that a single keystroke scaffolds and opens a fresh project. The prompt shows the destination directory (including the dated subdirectory when `date = TRUE`), and re-prompts with a hint if the entered name is invalid rather than aborting.

Usage

```
launch_project(
  theme,
  root = getOption("prefab.project_root", "~"),
  date = FALSE,
  slug = getOption("prefab.project_slug", TRUE),
  open = rlang::is_interactive(),
  label = NULL
)
```

Arguments

theme	A <code>prefab_theme</code> object created by <code>new_theme()</code> , <code>theme_from_dir()</code> , or a preset theme function, or a path to a directory of template files.
root	Directory under which the project is created. Defaults to the <code>prefab.project_root</code> option, or the home directory (" <code>~</code> ") if unset.
date	Whether to nest the project under a YYYY-MM-DD subdirectory of root (useful for dated, one-off projects). Defaults to <code>FALSE</code> .
slug	Whether to slugify the entered name into a filesystem-safe directory name (lower-case, non-alphanumeric runs collapsed to <code>_</code>). Defaults to the <code>prefab.project_slug</code> option, or <code>TRUE</code> if unset. When <code>FALSE</code> , the name is used verbatim and validated, erroring on characters that are unsafe in a path component.
open	Whether to activate the new project after creating it. Passed to <code>create_project()</code> , following the convention of <code>usethis::create_project()</code> : when <code>TRUE</code> , the project opens in a new session/window in RStudio/Positron, or in other editors the working directory of the active session is changed into the new project. Defaults to <code>rlang::is_interactive()</code> .
label	Optional prompt-window title, used to show which theme a key chord will apply (e.g. " <code>r_targets + claude_r_targets</code> "). Because a <code>prefab_theme</code> object does not retain the names used to build it, pass a human-readable label here. Defaults to "New project folder".

Details

Bind it in Positron by running it via `workbench.action.executeCode.console`, for example in `keybindings.json`:

```
{
  "key": "Ctrl+P N",
  "command": "workbench.action.executeCode.console",
  "args": {
    "langId": "r",
    "code": "prefab::launch_project(prefab::r_analysis(), label = 'r_analysis')",
    "focus": false
  }
}
```

For programmatic (non-interactive) project creation from a known path, use `create_project()` directly.

Value

The normalized project path (invisibly), or `NULL` if the prompt is cancelled or left empty.

See Also

`theme_from_dir()` for how a template directory is read.

load_themes	<i>Load custom theme definitions</i>
-------------	--------------------------------------

Description

Sources an R file containing custom theme functions into the global environment, making them available for use with `use_theme()` and `create_project()`.

Usage

```
load_themes(file = NULL)
```

Arguments

file	Path to an R file defining theme functions. If <code>NULL</code> , falls back to the <code>PREFAB_THEMES</code> environment variable, then <code>~/prefab-themes.R</code> .
------	---

Details

The file is resolved in order: the explicit file argument, then the `PREFAB_THEMES` environment variable, then `~/prefab-themes.R`.

Value

The file path that was sourced (invisibly), or NULL invisibly if no file was found.

Examples

```
## Not run:
# Source from default locations
load_themes()

# Source from a specific file
load_themes("~/my-org-themes.R")

## End(Not run)
```

new_theme	<i>Create a theme from steps</i>
-----------	----------------------------------

Description

Constructs a theme from step objects. NULL arguments are silently dropped, enabling conditional steps via `if (cond) step(...)`.

Usage

```
new_theme(...)
```

Arguments

... Step objects ([step_file\(\)](#), [step_text\(\)](#), [step_run\(\)](#)).

Value

A list with class "prefab_theme" containing a steps element.

Composing themes with +

Themes compose with `+`, which concatenates their steps with the left operand's steps first. Either operand may be a path to a directory of template files, coerced with [theme_from_dir\(\)](#) using its defaults:

```
r_analysis() + clauder_analysis()
r_analysis() + "~/my-extras"
~/my-baseline + r_analysis()
```

Steps cannot be added to a theme directly; wrap them in `new_theme()` first. Anything that is neither a theme nor a path to an existing directory is an error.

One limitation: an `fs::path()` value does not work as a `+` operand in either position, because `fs` defines its own `+` method and R refuses to choose between the two. Wrap it in [theme_from_dir\(\)](#) instead. A plain character path, and `fs::path()` values at [use_theme\(\)](#), [create_project\(\)](#), and [launch_project\(\)](#), are all fine.

See Also

[theme_from_dir\(\)](#) for how a template directory is read.

r_analysis	<i>R analysis project theme</i>
------------	---------------------------------

Description

Creates a theme that scaffolds a simple R analysis project with `main.R`, `README.md`, and `.gitignore`.

Usage

```
r_analysis(data_dirs = TRUE)
```

Arguments

`data_dirs` Logical. If TRUE (default), creates directories `./data_raw` and `./data_processed`.

Value

A `prefab_theme` object.

Examples

```
r_analysis()
```

r_targets	<i>R targets project theme</i>
-----------	--------------------------------

Description

Creates a theme that scaffolds an R targets project with `_targets.R`, `packages.R`, `README.md`, and `.gitignore`.

Usage

```
r_targets()
```

Value

A `prefab_theme` object.

Examples

```
r_targets()
```

step_file	<i>Create a file deployment step</i>
-----------	--------------------------------------

Description

Creates a step that deploys a file or directory to the project.

Usage

```
step_file(source, dest, strategy = "overwrite", data = NULL)
```

Arguments

source	Absolute path to the source file or directory. Typically produced by a source helper (<code>from_package()</code> , <code>from_dir()</code>) rather than written by hand.
dest	Path to the destination, relative to the project root.
strategy	How to handle a pre-existing destination file. One of "overwrite", "skip", "union", "append", or "merge_json".
data	NULL (default) for static file copy, "auto" to interpolate using only auto-discovered project variables (project_dir, package_name, year, date), or a named list of variables to interpolate into the file via <code>{{var}}</code> syntax before deploying.

Value

A list with class "prefab_step_file".

step_run	<i>Create a function execution step</i>
----------	---

Description

Creates a step that executes an R function for its side effects.

Usage

```
step_run(fn, ..., .label = NULL)
```

Arguments

fn	A function to execute.
...	Additional arguments passed to fn at execution time.
.label	Optional label for display. When NULL (default), captured via <code>deparse(substitute(fn))</code> .

Value

A list with class "prefab_step_run".

step_text	<i>Create an inline text deployment step</i>
-----------	--

Description

Creates a step that deploys inline text content to the project. Like `step_file()` but takes a character vector instead of a source file path.

Usage

```
step_text(content, dest, strategy = "overwrite")
```

Arguments

content	Character vector of lines to deploy (one element per line). <code>character(0)</code> is allowed except with <code>strategy = "merge_json"</code> .
dest	Path to the destination, relative to the project root.
strategy	How to handle a pre-existing destination file. One of "overwrite", "skip", "union", "append", or "merge_json".

Value

A list with class "prefab_step_text".

theme_code	<i>Print R code that reproduces a theme</i>
------------	---

Description

Prints the R code that would reproduce the given theme via `cat()`, and returns the code invisibly as a single character string.

Usage

```
theme_code(theme)
```

Arguments

theme	A <code>prefab_theme</code> object. Unlike <code>use_theme()</code> , <code>create_project()</code> , and <code>launch_project()</code> , this does not accept a path to a directory of template files; pass <code>theme_from_dir()</code> the path instead.
-------	--

Value

The generated R code as a single character string (invisibly).

 theme_from_dir

 Create a theme from a directory of template files

Description

This function reads a directory tree and makes a theme. Each file becomes a `step_file()`. Each empty directory becomes a step that makes the directory again. An optional `_prefab.yml` file at the root of the directory can change the strategy and the template data for each file.

Usage

```
theme_from_dir(path, strategy = "skip")
```

Arguments

path	The path to a directory of template files. The directory must exist. <code>prefab</code> expands a leading <code>~</code> . <code>prefab</code> also makes a relative path absolute against the working directory. <code>prefab</code> does these two operations only to find the files. Each step keeps the path in the spelling that you supply in its provenance record.
strategy	The default merge strategy for all files. A <code>_prefab.yml</code> sidecar can change the strategy for each file. Use one of "overwrite", "skip", "union", "append", or "merge_json".

Details

Each file below path becomes a `step_file()`. The destination of the step is the path of the file relative to path. `prefab` includes hidden files. `prefab` also includes hidden empty directories.

`prefab` excludes these directories at all levels of the tree:

- `.git`
- `.Rproj.user`
- `_targets`
- `renv/library`

`prefab` does not deploy the symbolic links below path.

This function is called by `use_theme()`, `create_project()`, `launch_project()`, and `+` when they process a path. They call this function with the default arguments.

Value

A `prefab_theme` object.

Examples

```
## Not run:
# Make a theme from a directory of configuration files
use_theme(theme_from_dir("~/my-template"))

# The short form. It calls theme_from_dir() with the default arguments.
use_theme("~/my-template")

# Add the theme to another theme
use_theme(r_analysis() + theme_from_dir("~/my-extras"))

## End(Not run)
```

use_theme	<i>Apply a theme to the current project</i>
-----------	---

Description

Discovers the project root and executes the theme against it.

Usage

```
use_theme(theme)
```

Arguments

theme	A prefab_theme object created by new_theme() , theme_from_dir() , or a pre-set theme function, or a path to a directory of template files.
-------	--

Value

The project root path (invisibly).

See Also

[theme_from_dir\(\)](#) for how a template directory is read.

Examples

```
## Not run:
use_theme(r_analysis())
use_theme(r_analysis() + clauder_analysis())

# A directory of template files
use_theme("~/my-template")

## End(Not run)
```

Index

`cat()`, [11](#)
`clauder_analysis`, [2](#)
`clauder_package`, [3](#)
`clauder_targets`, [3](#)
`create_project`, [4](#)
`create_project()`, [6–8](#), [11](#), [12](#)

`devtools::load_all()`, [5](#)

`from_dir`, [5](#)
`from_dir()`, [10](#)
`from_package`, [5](#)
`from_package()`, [10](#)
`fs::path()`, [8](#)
`fs::path_abs()`, [4](#), [5](#)

`launch_project`, [6](#)
`launch_project()`, [8](#), [11](#), [12](#)
`load_themes`, [7](#)

`new_theme`, [8](#)
`new_theme()`, [4](#), [6](#), [13](#)

`r_analysis`, [9](#)
`r_targets`, [9](#)
`rlang::is_interactive()`, [4](#), [6](#)
`rstudioapi::showPrompt()`, [6](#)

`step_file`, [10](#)
`step_file()`, [5](#), [8](#), [11](#), [12](#)
`step_run`, [10](#)
`step_run()`, [8](#)
`step_text`, [11](#)
`step_text()`, [8](#)

`theme_code`, [11](#)
`theme_from_dir`, [12](#)
`theme_from_dir()`, [4](#), [6–9](#), [11](#), [13](#)

`use_theme`, [13](#)
`use_theme()`, [7](#), [8](#), [11](#), [12](#)
`usethis::create_package()`, [4](#)
`usethis::create_project()`, [4](#), [6](#)