

Package: pipeline (via r-universe)

September 19, 2026

Type Package

Title Orchestrate Functional Workflows

Version 0.0.0.9165

Description Make it easy for users to adopt a functional approach to their analysis. Pipelines are defined using a reduced subset of R and, when run, use an intelligent approach to work planning and object caching. Whilst the goal is to be ``good enough" for many pieces of analysis, by encouraging a functional approach, it remains easy to move to more advanced alternatives such as targets (<[doi:10.32614/CRAN.package.targets](https://doi.org/10.32614/CRAN.package.targets)>).

URL <https://timtaylor.codeberg.page/pipeline/>

BugReports <https://codeberg.org/TimTaylor/pipeline/issues>

License GPL-2 | GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports carrier, cli, datasets, dplyr, globals, igraph, rlang, ronfig
(>= 0.0.10), stats, tibble, tools, utils, vctrs

Suggests fs, litedown, testthat (>= 3.0.0)

Depends R (>= 4.5.0)

Config/testthat/edition 3

Config/testthat/load-all list(export_all = FALSE, helpers = FALSE)

Config/testthat/parallel true

VignetteBuilder litedown

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libglib-dev libxml2-dev

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

Date/Publication 2026-05-22 13:16:45 UTC

RemoteUrl <https://codeberg.org/TimTaylor/pipeline>

RemoteRef v0.0.0.9165

RemoteSha 1c1161b04e28c12f1ff933e24255a831971bb7ce
RemoteSubdir pkg

Contents

pipeline_attach	2
pipeline_run	3
pipeline_skeleton	6
Index	7

pipeline_attach	<i>Attach (and detach) the pipeline functions and configuration</i>
-----------------	---

Description

These functions allow users to attach (and detach) the pipeline functions and scenario to the R search path which can be useful during development. They are attached as local:pipelinedevenv. Whether this is a good idea or not is to be determined.

Usage

```
pipeline_attach(  
  scenario = NULL,  
  scenario_default = getOption("pipeline.scenario_default", "default"),  
  pipeline_dir = ".",  
  relative_config_file = getOption("pipeline.config_file", "config.R"),  
  relative_function_dir = getOption("pipeline.function_dir", "R"),  
  relative_output_dir = getOption("pipeline.output_dir", "output")  
)  
  
pipeline_detach()
```

Arguments

- scenario** Configuration you wish to attach.
If NULL, any configuration file (see config_file) will be read in as is.
Character input will be treated as configuration you wish to consider. In this case the configuration file will first be parsed looking for a named-list entry corresponding to the value of the scenario_default argument. The chosen scenario is then layered on top of this default using utils::modifyList().
- scenario_default** The default scenario to consider.
If config is not NULL, this represents the default scenario on which the specified scenarios are layered.

```

pipeline_dir    [character]
                The directory you wish to run the the pipeline relative to.
relative_config_file
                [character]
                The configuration file.
                Must be a none-nested and given relative to pipeline_dir.
relative_function_dir
                [character]
                The directory to look for user defined pipeline functions.
                Must be a none-nested and given relative to pipeline_dir.
relative_output_dir
                [character]
                The directory for output.
                Must be a none-nested and given relative to pipeline_dir.

```

Value

NULL (invisibly). Called only for side effects.

Examples

```

# generate a demo pipeline with a single scenario
dir <- pipeline_skeleton(tempfile())

## Not run:
# If we attach the 'default' scenario then both the CONFIG and the
#   defined functions will be available to us.
pipeline_attach(scenario = "default", pipeline_dir = dir)
CONFIG
exists("load_dat") && is.function(load_dat)
pipeline_attach(scenario = "production", pipeline_dir = dir)
CONFIG
pipeline_detach()
exists("load_dat")

## End(Not run)

unlink(dir)

```

pipeline_run

Run a pipeline

Description

pipeline_run() manages the running of a user defined pipeline simplifying object caching and configuration management across different scenarios.

Usage

```

pipeline_run(
  x,
  ...,
  scenario = NULL,
  scenario_default = getOption("pipeline.scenario_default", "default"),
  pipeline_dir = ".",
  relative_config_file = getOption("pipeline.config_file", "config.R"),
  relative_function_dir = getOption("pipeline.function_dir", "R"),
  relative_output_dir = getOption("pipeline.output_dir", "output"),
  force = FALSE,
  saveRDS_args = list(),
  readRDS_args = list(),
  return = TRUE,
  gc = FALSE
)

```

Arguments

<code>x</code>	[expression] R Expression of pipeline assignments. Normally this will involve multiple assignments and will need to be embraced to represent a single expression.
<code>...</code>	Not currently used.
<code>scenario</code>	If NULL, any configuration file (see <code>config_file</code>) will be read in as is. Character input will be treated as configurations you wish to loop over. In this case the configuration file will first be parsed looking for a named-list entry corresponding to the value of the <code>scenario_default</code> argument. The chosen scenarios are then layered on top of this default using <code>utils::modifyList()</code> .
<code>scenario_default</code>	[character] The default scenario to consider. If <code>scenario</code> is NULL this will represent the folder under the <code>output_dir</code> where file output and cached objects are stored. If <code>scenario</code> is not NULL, this represents the default scenario on which the specified scenarios are layered.
<code>pipeline_dir</code>	[character] The directory you wish to run the the pipeline relative to.
<code>relative_config_file</code>	[character] The configuration file. Must be a none-nested and given relative to <code>pipeline_dir</code> .
<code>relative_function_dir</code>	[character] The directory to look for user defined pipeline functions. Must be a none-nested and given relative to <code>pipeline_dir</code> .

relative_output_dir	[character] The directory for output. Must be a none-nested and given relative to pipeline_dir.
force	[bool] Do you want to force a run of the pipeline. If TRUE, then cached objects are removed and the pipeline is (re)run. If a character vector then the corresponding object(s) are removed from the cache and the pipeline is rerun.
saveRDS_args	[list] List of additional arguments passed to saveRDS(). This argument allows you to pass additional arguments to that function (e.g. list(compress = "zstd"))
readRDS_args	[list] List of additional arguments passed to readRDS().
return	[bool] Should the output be returned. Defaults to TRUE, but when running across multiple scenarios and with outputs that use a large amount of memory it can be useful to set FALSE. If FALSE, a successful run will return NULL invisibly.
gc	[bool] Should we force calls to gc() whilst running the pipeline. For pipeline's creating large objects, setting this to TRUE <i>may</i> help reduce memory consumption.

Value

A named list of outputs for each configuration.

Examples

```
# generate a demo pipeline with a single scenario
dir <- pipeline_skeleton(tempfile(), single = TRUE)

# Note the configuration file and the folder of R functions
list.files(dir, all.files = TRUE, recursive = TRUE, no.. = TRUE)

# Run the pipeline
out <- pipeline_run(
  {
    raw  <- load_dat(CONFIG$in_csv)
    clean <- wrangle_dat(raw, CONFIG$rows)
    plot <- plot_dat(clean, CONFIG$out_plot)
  },
  pipeline_dir = dir
)

# output in a list
```

```
str(out$default$raw)
str(out$default$clean)
out$default$plot

unlink(dir)
```

pipeline_skeleton	<i>Create a demo pipeline</i>
-------------------	-------------------------------

Description

Sets up an example pipeline which can be used as the basis for your own.

Usage

```
pipeline_skeleton(dir = "DemoPipeline", single = FALSE)
```

Arguments

dir	Directory you wish to create the pipeline in.
single	Whether the demo should use a single or multiple scenarios in the demo.

Value

Absolute path of the created pipeline directory (invisibly).

Examples

```
dir <- pipeline_skeleton(tempfile())

# Note the configuration file and the folder of R functions
list.files(dir, all.files = TRUE, recursive = TRUE, no.. = TRUE)

unlink(dir)
```

Index

`pipeline_attach`, [2](#)
`pipeline_detach (pipeline_attach)`, [2](#)
`pipeline_run`, [3](#)
`pipeline_skeleton`, [6](#)