Package ‘rmonad’

February 14, 2020

Type Package

Version 0.7.0

Title A Monadic Pipeline System

Description A monadic solution to pipeline analysis. All operations -- and the errors, warnings and messages they emit -- are merged into a directed graph. Infix binary operators mediate when values are stored, how exceptions are handled, and where pipelines branch and merge. The resulting structure may be queried for debugging or report generation. 'rmonad' complements, rather than competes with, non-monadic pipeline packages like 'magrittr' or 'pipeR'. This work is funded by the NSF (award number 1546858).

URL https://github.com/arendsee/rmonad

BugReports https://github.com/arendsee/rmonad/issues

Depends R (>= 3.2.0)

Imports igraph, methods, magrittr, glue, pryr, digest

Suggests testthat, covr, knitr, rmarkdown, readr, stringr, tidyr, dplyr, Nozzle.R1

VignetteBuilder knitr

LazyData yes

RoxygenNote 7.0.2

License GPL-3

Encoding UTF-8

NeedsCompilation no

Author Zebulun Arendsee [aut, cre]

Maintainer Zebulun Arendsee <zbwrnz@gmail.com>

Repository CRAN

Date/Publication 2020-02-14 07:00:02 UTC

R topics documented:

apply_rewriters . . . 3
clear_cache . . . 3
const . . . 4
crunch . . . 4
esc . . . 5
fail_cache . . . 5
false . . . 6
false_as_error . . . 6
first . . . 7
get_dependency_matrix . . . 7
gff . . . 8
infix . . . 8
is_rmonad . . . 9
loop . . . 10
make_cacher . . . 10
make_recacher . . . 11
memory_cache . . . 12
missues . . . 13
mtabulate . . . 13
nothing . . . 14
no_cache . . . 14
null_as_error . . . 15
plot.Rmonad . . . 15
print.Rmonad . . . 16
report . . . 17
rmonad . . . 17
rmonad_checkers . . . 19
rmonad_getters . . . 20
second . . . 22
size . . . 23
splice_function . . . 23
tag . . . 24
toss . . . 24
true . . . 25
view . . . 25
viewID . . . 26
viewIDs . . . 26
views . . . 27
void_cache . . . 27
x_to_monad . . . 28

Index **30**

`apply_rewriters` *Apply rewriters to an Rmonad*

Description

Rewriters are functions stored in an Rmonad's metadata list that operate on an Rmonad after it has evaluated its code.

Usage

```
apply_rewriters(x, meta = .single_meta(x))
```

Arguments

`x`	The Rmonad
`meta`	A metadata list

`clear_cache` *Clear cached values and delete temporary files*

Description

Clear cached values and delete temporary files

Usage

```
clear_cache(m, index = .get_ids(m))
```

Arguments

`m`	Rmonad object
`index`	indices to clear (all indices by default)

Value

Rmonad object

See Also

Other cache: `fail_cache`, `make_cacher`, `make_recacher`, `memory_cache`, `no_cache`, `void_cache`

Examples

```
256 %v>% sqrt %>>% sqrt %>>% sqrt  -> m
m
clear_cache(m)
```

`const` *Ignore the first input, return the second*

Description

This function can be used to change the value in the lhs of a monadic sequence

Usage

```
const(x, r)
```

Arguments

x	ignored value
r	replacing value

See Also

Other helper_functions: `false_as_error`, `false`, `null_as_error`, `toss`, `true`

`crunch` *Cache all large values that are stored in memory*

Description

Cache all large values that are stored in memory

Usage

```
crunch(m)
```

Arguments

m	Rmonad object

Examples

```
## Not run:
set.seed(42)
m <- as_monad(runif(1e6), tag="a") %>>%
     sqrt %>% tag("b") %>>%
     log %>% tag("c") %>>% prod(2) %>>% prod(3)
m1 <- crunch(m)
get_value(m,  1:3) %>% lapply(head)
get_value(m1, 1:3) %>% lapply(head)

## End(Not run)
```

`esc` *Returns the value a monad holds*

Description

If the monad is in the passing state, return the wrapped value. Otherwise, raise an appropriate error.

Usage

```
esc(m, quiet = FALSE)
```

Arguments

`m`	An Rmonad
`quiet`	If TRUE, print the exact messages that are raised, without extra context.

Details

Regardless of pass/fail status, `esc` raises all collected warnings and prints all messages. Terminating a monadic sequence with `esc` should obtain a result very close to running the same code outside the monad. The main difference is that Rmonad appends the toplevel code that generated the error.

See Also

Other from_Rmonad: `missues`, `mtabulate`, `report`

Examples

```
library(magrittr)
256 %>>% sqrt %>% esc
```

`fail_cache` *Represent a dummy value for a node downstream of a failing node*

Description

Returns a ValueManager that represents a dummy value for a node downstream of a failing node. Unlike `void_cache`, this presence of this manager in a pipeline is not pathological, so does not raise a warning by default.

Usage

```
fail_cache()
```

Value

A function that represents an unrun node

See Also

Other cache: `clear_cache`, `make_cacher`, `make_recacher`, `memory_cache`, `no_cache`, `void_cache`

`false` *Return false for all input*

Description

Return false for all input

Usage

```
false(...)
```

Arguments

`...`	whatever

See Also

Other helper_functions: `const`, `false_as_error`, `null_as_error`, `toss`, `true`

`false_as_error` *Make NULL values an error*

Description

Make NULL values an error

Usage

```
false_as_error(x)
```

Arguments

`x`	Input value

See Also

Other helper_functions: `const`, `false`, `null_as_error`, `toss`, `true`

`first` *Given two arguments, return the first*

Description

Given two arguments, return the first

Usage

```
first(x, y)
```

Arguments

`x`	anything
`y`	anything

See Also

Other help_functions: `nothing`, `second`

`get_dependency_matrix` *Get dependencies of local variables on inputs*

Description

Get dependencies of local variables on inputs

Usage

```
get_dependency_matrix(declarations, bound_vars)
```

Arguments

`declarations`	A list of declarations
`bound_vars`	Character vector of variables names that are bound as arguments to the function

Value

logical matrix

`gff` *Data for GFF processing vignette*

Description

Contains 4 files:

1. good - a valid GFF string
2. not_a_gff1 - a string that is not a GFF file at all
3. invalid_type - a table with invalid types
4. good_result - the final pipeline produced using the good gff

Usage

```
gff
```

Format

List

`infix` *Infix operators*

Description

Infix monadic sequence operators

Usage

```
lhs %>>% rhs

lhs %v>% rhs

lhs %*>% rhs

lhs %>_% rhs

lhs %>^% rhs

lhs %^>% rhs

lhs %|>% rhs

lhs %||% rhs

lhs %__% rhs
```

Arguments

`lhs`	left hand side value
`rhs`	right hand side value

Details

See the main package help page (`?rmonad`) or the `intro` and `cheatsheet` vignettes for more information.

Examples

```
256 %>>% sqrt
256 %v>% sqrt
list(1,2,3) %*>% sum
iris %>_% plot %>>% summary
1:10 %>^% rgamma(10, 5) %>^% rgamma(10, 6) %^>% cor
1:10 %>>% colSums %|>% sum
stop("die") %||% 4 %>>% sqrt
16 %>>% sqrt %__% 25 %>>% sqrt
```

`is_rmonad` *Determine whether something is an Rmonad object*

Description

Determine whether something is an Rmonad object

Usage

```
is_rmonad(m)
```

Arguments

`m`	Rmonad object

Value

logical TRUE if m is an Rmonad

loop — *Apply an rmonad pipeline function to each element in a rmonad bound list*

Description

Apply an rmonad pipeline function to each element in a rmonad bound list

Usage

```
loop(m, FUN, looper = lapply, ...)
```

Arguments

`m`	Rmonad object wrapping a vector
`FUN`	function of an element from the vector stored in `m` that returns an Rmonad object.
`looper`	function that applies each element in the input vector to `FUN`. The default it `lapply`.
`...`	Additional arguments sent to `FUN`

Value

Rmonad object wrapping a vector of the values wrapped by the outputs of `FUN`

Examples

```
foo <- function(x) { x %>>% sqrt }
c(256, 6561) %v>% sqrt %>% loop(foo) %>>% lapply(sqrt)
```

make_cacher — *Make Cacher object*

Description

Make Cacher object

Usage

```
make_cacher(f_path = function() getOption("rmonad.cache_dir"),
  f_save = saveRDS, f_get = readRDS, f_del = unlink,
  f_ext = function(cls) ".Rdata")
```

Arguments

`f_path`	A function for finding the directory in which to cache results
`f_save`	function of x and filename that saves x to the path filename
`f_get`	function of filename that retrieves the cached data
`f_del`	function of filename that deletes the cached data
`f_ext`	function of class(x) that determines the filename extension

Value

A function that builds a local cache function for a value

See Also

Other cache: `clear_cache`, `fail_cache`, `make_recacher`, `memory_cache`, `no_cache`, `void_cache`

`make_recacher` *Make a function that takes an Rmonad and recaches it*

Description

Make a function that takes an Rmonad and recaches it

Usage

```
make_recacher(cacher, preserve = TRUE)
```

Arguments

`cacher`	A function of a data value
`preserve`	logical Should the cached value be preserved across bind operations?

Value

A function that swaps the cache function of an Rmonad

See Also

Other cache: `clear_cache`, `fail_cache`, `make_cacher`, `memory_cache`, `no_cache`, `void_cache`

Examples

```
## Not run:
  recacher <- make_recacher(make_local_cacher())
  m <- iris %>>% summary %>% recacher
  # load the data from a local file
  .single_value(m)

  recacher <- make_recacher(memory_cache)
  m <- iris %>>% summary %>% recacher
  # load the data from memory
  .single_value(m)

## End(Not run)

add1 <- function(x) x+1
add2 <- function(x) x+2
add3 <- function(x) x+3
cc <- make_recacher(make_local_cacher())
3 %>>% add1 %>% cc %>>% add2 %>>% add3 -> m
m
```

---

memory_cache *Store a value in memory*

Description

Store a value in memory

Usage

```
memory_cache(x)
```

Arguments

x	Value to be stored

Value

A function that returns a value stored in memory

See Also

Other cache: clear_cache, fail_cache, make_cacher, make_recacher, no_cache, void_cache

Examples

```
foo <- 45
foo_proxy <- memory_cache(foo)
foo
foo_proxy@get()
```

missues — *Tabulates all errors, warnings and notes*

Description

Tabulates all errors, warnings and notes

Usage

```
missues(m)
```

Arguments

`m`	An Rmonad

See Also

Other from_Rmonad: `esc`, `mtabulate`, `report`

Examples

```
data(gff)
m <- gff$good_result
missues(m)
```

mtabulate — *Make tabular summary of a pipeline*

Description

Make tabular summary of a pipeline

Usage

```
mtabulate(m, code = FALSE)
```

Arguments

`m`	An Rmonad
`code`	logical Should the code by included?

See Also

Other from_Rmonad: `esc`, `missues`, `report`

Examples

```
data(gff)
m <- gff$good_result
mtabulate(m)
```

nothing — *Do nothing*

Description

Do nothing

Usage

```
nothing(...)
```

Arguments

`...`	anything

Value

nothing

See Also

Other help_functions: `first`, `second`

no_cache — *Represent a value that has been deleted*

Description

By default, the value of a node that has already been executed will be set to this function.

Usage

```
no_cache()
```

Value

A function that represents a deleted value

See Also

Other cache: `clear_cache`, `fail_cache`, `make_cacher`, `make_recacher`, `memory_cache`, `void_cache`

`null_as_error` *Make NULL values an error*

Description

Currently not exported.

Usage

```
null_as_error(x)
```

Arguments

`x`	Input value

See Also

Other helper_functions: `const`, `false_as_error`, `false`, `toss`, `true`

`plot.Rmonad` *Render an Rmonad graph*

Description

Convert the Rmonad object to a DiagrammeR graph and then render it

Usage

```
## S3 method for class 'Rmonad'
plot(x, y, label = NULL, color = "status", ...)
```

Arguments

`x`	An Rmonad object
`y`	This variable is currently ignored
`label`	The node labels. If NULL, the node labels will equal node ids. It may be one of the strings ['code', 'time', 'space', 'value', 'depth']. If 'value' is selected, nodes with no value cached are represented with '-'. Alternatively, it may be a function that maps a single Rmonad object to a string.
`color`	How to color the nodes. Default is 'status', which colors green for passing, orange for warning, and red for error. Alternatively, color can be a function of an Rmonad object, which will be applied to each node.
`...`	Additional arguments passed to plot.igraph. These arguments may override rmonad plotting defaults and behavior specified by the 'label' and 'color' parameters.

Details

The nodes in the graph represent both a function and the function's output. The edges are relationships between nodes. In an unnested pipeline, every edge represents data flow from source to sink (solid black edges). Nested pipelines contain three additional edge types: a transitive edge, where a node is dependent on a value that was passed to its parent (dotted gray line); a nest edge linking a node to the nested node that produced its value (solid red line); a 'prior' edge for pipelines coupled with the `%__%` operator (thick dotted blue line).

Examples

```
data(gff)
# default plot
plot(gff$good_result)
# turn off vertex labels and set vertex size
plot(gff$good_result, vertex.size=10, vertex.label=NA)
```

`print.Rmonad` *Rmonad print generic function*

Description

Rmonad print generic function

Usage

```
## S3 method for class 'Rmonad'
print(x, verbose = FALSE, value = TRUE, ...)
```

Arguments

`x`	An Rmonad object
`verbose`	logical print verbose output (include benchmarking)
`value`	logical print the value wrapped in the Rmonad
`...`	Additional arguments (unused)

Examples

```
m1 <- 256 %v>% sqrt %>>% sqrt %>>% sqrt
print(m1)
print(m1, verbose=TRUE)
```

`report` *Convert a pipeline to Rmarkdown*

Description

Plots an rmonad workflow, summarizes the nodes, lists issues, and lists details for each node. This function is likely to change extensively in the future. It should be seen as one example of the kind of report that can be generated by rmonad, rather than THE report.

Usage

```
report(m, prefix = "report")
```

Arguments

`m`	An Rmonad
`prefix`	A file prefix for the generated report

See Also

Other from_Rmonad: `esc`, `missues`, `mtabulate`

Examples

```
## Not run:
report(-1:2 %>>% log %>>% sqrt %__% "asdf" %>>% sqrt)

## End(Not run)
```

`rmonad` *rmonad: handling pipes, errors, and everything with monads*

Description

Rmonad merges blocks of code into a graph containing the history of all past operations, and optionally their values. It consists mainly of a set of monadic bind operators for controlling a pipeline and handling error. It also contains functions for operating on monads, evaluating expressions into monads, and extracting values from them. I will briefly introduce the most useful of these here. For more information see the `introduction` vignette.

Basic Operators

%>>% monadic bind: applies rhs function to the lhs value

%v>% monadic bind: store intermediate result

%*>% bind lhs list as arguments to right. The lhs may be a literal list or a monad bound list.

%>_% perform rhs action, discard result, pass the lhs

%>^% Bind as a new branch, pass input on main. This differs from `%>_%` in that future operations do not depend on its pass/fail status. Use `unbranch` to extract all branches from an Rmonad object.

%||% if input is error, use rhs value instead

%|>% if input is error, run rhs on last passing result

%__% keep parents from the lhs (errors ignored). This allows chaining of independent operations.

Operators targeted for deprecation

%^>% Monadic bind and record input in monad. Perform rhs operation on lhs branches. I may deprecate this operator.

x to monad functions

`as_monad` - evaluate an expression into a monad (capturing error)

`funnel` - evaluate expressions into a list inside a monad

monad to monad functions

`forget` - erase parents from a monad

`combine` - combine a list of monads into a list in a monad

monad to x functions

`esc` - extract the result from a computation

`mtabulate` - summarize all steps in a pipeline into a table

`missues` - tabulate all warnings and errors from a pipeline

`unbranch` - extract all branches from the pipeline

Examples

```
# chain operations
cars %>>% colSums

# chain operations with intermediate storing
cars %v>% colSums

# handle failing monad
iris %>>% colSums %|>% head
cars %>>% colSums %|>% head
```

```
# run an effect
cars %>_% plot %>>% colSums

# return first successful operation
read.csv("a.csv") %||% iris %>>% head

# join two independent pipelines, preserving history
cars %>>% colSums %__% cars %>>% lapply(sd) %>>% unlist

# load an expression into a monad, catching errors
as_monad(stop("instant death"))

# convert multiple expressions into a list inside a monad
funnel(stop("oh no"), runif(5), sqrt(-1))
```

`rmonad_checkers` *Vectorized existence checkers for public Rmonad fields*

Description

Vectorized existence checkers for public Rmonad fields

Usage

```
has_code(m, ...)

has_tag(m, ...)

has_error(m, ...)

has_doc(m, ...)

has_warnings(m, ...)

has_notes(m, ...)

has_meta(m, ...)

has_time(m, ...)

has_mem(m, ...)

has_value(m, ...)

has_parents(m, ...)

has_dependents(m, ...)
```

```
has_prior(m, ...)

has_nest(m, ...)

has_summary(m, ...)
```

Arguments

`m`	An Rmonad object
`...`	Additional arguments passed to `get_*` functions

Examples

```
data(gff)
m <- gff$good_result

has_code(m)
has_dependents(m)
has_doc(m)
has_error(m)
has_mem(m)
has_meta(m)
has_nest(m)
has_notes(m)
has_parents(m)
has_prior(m)
has_summary(m)
has_time(m)
has_value(m)
has_warnings(m)

# find root nodes
which(!has_parents(m))

# find terminal (output) nodes
which(!has_dependents(m))

# count number of independent chains
sum(has_prior(m)) + 1
```

`rmonad_getters` *Vectorized getters for public Rmonad fields*

Description

Vectorized getters for public Rmonad fields

Usage

```
get_parents(m, index = .get_ids(m), tag = NULL)

get_dependents(m, index = .get_ids(m), tag = NULL)

get_nest(m, index = .get_ids(m), tag = NULL)

get_prior(m, index = .get_ids(m), tag = NULL)

get_depth(m, index = .get_ids(m), tag = NULL)

get_nest_depth(m, index = .get_ids(m), tag = NULL)

get_value(m, index = .get_ids(m), tag = NULL, warn = TRUE)

get_key(m, index = .get_ids(m), tag = NULL)

get_id(m, index = .get_ids(m), tag = NULL)

get_OK(m, index = .get_ids(m), tag = NULL)

get_code(m, index = .get_ids(m), tag = NULL)

get_tag(m, index = .get_ids(m), tag = NULL)

get_error(m, index = .get_ids(m), tag = NULL)

get_warnings(m, index = .get_ids(m), tag = NULL)

get_notes(m, index = .get_ids(m), tag = NULL)

get_doc(m, index = .get_ids(m), tag = NULL)

get_meta(m, index = .get_ids(m), tag = NULL)

get_time(m, index = .get_ids(m), tag = NULL)

get_mem(m, index = .get_ids(m), tag = NULL)

get_summary(m, index = .get_ids(m), tag = NULL)
```

Arguments

`m`	An Rmonad object
`index`	Selection of indices to extract (all by default). The indices may be a vector of integers, node names, or igraph vertices (`igraph.vs`).
`tag`	character vector specifying the tags that must be associated with extracted nodes
`warn`	logical In get_value, raise a warning on an attempt to access an uncached node

Examples

```
data(gff)
m <- gff$good_result

# vectorized accessors for all stored slots
get_value(m, warn=FALSE)
get_OK(m)
get_code(m)
get_dependents(m)
get_doc(m)
get_error(m)
get_id(m)
get_mem(m)
get_meta(m)
get_nest(m)
get_nest_depth(m)
get_notes(m)
get_parents(m)
get_prior(m)
get_summary(m)
get_time(m)
get_warnings(m)

# get the code associated with long running functions
get_code(m)[get_time(m) > 0.1]

# Calculate the average node degree
nparents <- sapply(get_parents(m), length)
nchildren <- sapply(get_dependents(m), length)
sum(nparents + nchildren) / size(m)
```

`second` *Given two arguments, return the second*

Description

Given two arguments, return the second

Usage

```
second(x, y)
```

Arguments

x	anything
y	anything

See Also

Other help_functions: `first`, `nothing`

`size` *Return the number of nodes in the workflow*

Description

Return the number of nodes in the workflow

Usage

```
size(m)
```

Arguments

`m`	Rmonad object

Examples

```
m <- 256 %>>% sqrt %>>% sqrt
size(m)
```

`splice_function` *Take a monadic bind operation's result and splice histories*

Description

We need to link input variables to the nodes in the nested pipeline that use them.

Usage

```
splice_function(f, m, ms, ...)
```

Arguments

`f`	The function
`m`	The monadic result of running f(ms)
`ms`	The list of inputs passed to f
`...`	additional arguments passed to add_transitive_edges

`tag` *Set the tag of an Rmonad object*

Description

Set the tag of an Rmonad object

Usage

```
tag(m, ..., index = m@head)
```

Arguments

`m`	Rmonad object
`...`	one or more tags for the given nodes
`index`	character or integer vector, specifying the nodes which will be assigned the new tag

Value

Rmonad object with new tags

Examples

```
library(magrittr)
1 %>>% prod(2) %>% tag('a/b') %>>% prod(3) %>% get_tag
```

`toss` *Take input and do nothing with it*

Description

Take input and do nothing with it

Usage

```
toss(...)
```

Arguments

`...`	whatever

See Also

Other helper_functions: `const`, `false_as_error`, `false`, `null_as_error`, `true`

`true` *Return true for all input*

Description

Return true for all input

Usage

```
true(...)
```

Arguments

`...`	whatever

See Also

Other helper_functions: `const`, `false_as_error`, `false`, `null_as_error`, `toss`

`view` *Set the head of an Rmonad to a particular tag*

Description

Will split on '/'

Usage

```
view(m, ...)
```

Arguments

`m`	Rmonad object
`...`	one or more tag strings specifying a unique node in the pipeline

Value

Rmonad object with head reset

Examples

```
library(magrittr)
m <- 256 %v>% sqrt %>% tag('a', 'b') %v>% sqrt
esc(view(m, 'a/b'))
funnel(view(m, 'a'), m) %*>% sum
```

`viewID` *Move head to this id*

Description

Move head to this id

Usage

```
viewID(m, id)
```

Arguments

`m`	rmonad object
`id`	integer index

`viewIDs` *Return a list of Rmonad objects at these positions*

Description

Return a list of Rmonad objects at these positions

Usage

```
viewIDs(m, ids)
```

Arguments

`m`	rmonad object
`ids`	integer vector index

`views` *Get a list of Rmonad objects matching the given tag*

Description

Get a list of Rmonad objects matching the given tag

Usage

```
views(m, ...)
```

Arguments

`m`	Rmonad object
`...`	one or more tags

Value

list of Rmonad objects

Examples

```
library(magrittr)
1 %>>% prod(2) %>% tag('a/b') %>>%
       prod(2) %>% tag('a/c') %>>%
       prod(2) %>% tag('a/c') %>>%
       prod(2) %>% tag('g/a') -> m
views(m, 'a')
```

`void_cache` *Represent a value that has not been set*

Description

This is the default value of RmonadData@value. It should always be replaced shortly after the object is created, thus should only be encountered if 1) the user is directly creating RmonadData objects (in which case they should be spoken to sternly) or 2) there is a bug in rmonad.

Usage

```
void_cache()
```

Value

A function that represents a void, uncached value

See Also

Other cache: `clear_cache`, `fail_cache`, `make_cacher`, `make_recacher`, `memory_cache`, `no_cache`

`x_to_monad` *Conversions to monads*

Description

These functions convert possibly non-monadic inputs into monads.

Usage

```
as_monad(expr, desc = NULL, tag = NULL, doc = .default_doc(),
  key = NULL, env = parent.frame(), lossy = FALSE)

funnel(..., env = parent.frame(), keep_history = TRUE)

combine(xs, keep_history = TRUE, desc = .default_code())
```

Arguments

`expr`	An expression
`desc`	A description of the monad (usually the producing code)
`tag`	Character vector specifying the tag to associate with a node
`doc`	A docstring to associate with the monad
`key`	16 byte raw vector
`env`	Evaluation environment
`lossy`	logical Should unnesting with record be done?
`...`	multiple expressions
`keep_history`	merge the histories of all monads
`xs`	A list of elements to join into a monad

Details

For each of these functions, failure of any part causes failure of the whole. Any non-monadic inputs will be converted to monads. Any exceptions raised in the inputs will be caught.

`as_monad` evaluate a single expression into an Rmonad. If the value is already an Rmonad, it will be nested.

`funnel` evaluates multiple arguments into one Rmonad. It can be used within pipelines to create multi-input nodes (works well with `%*>%`).

`combine` takes a list of Rmonads and joins the elements into one Rmonad. The values of the original monadic containers joined into a list in the child Rmonad. The list Rmonads are recorded as the new Rmonad's parents.

Examples

```
as_monad(stop(1))
as_monad(1:10)
as_monad(5 %>>% sqrt)

## merge failing inputs
funnel( 1:10, stop(1), sqrt(-3:3) )

## join pipelines
b2 <- letters[1:10] %>>% sqrt
b3 <- -3:6 %>>% log
1:10 %>% funnel(b2,b3) %>>%
  {data.frame(b1=.[[1]], b2=.[[2]], b3=.[[3]])}

z <- list(
  x = rnorm(10) %>>% sqrt,
  y = 1 %>>% colSums
)
combine(z)
```

Index

∗Topic **datasets**
    gff, 8
%*>% (infix), 8
%>>% (infix), 8
%>_% (infix), 8
%>^% (infix), 8
%__% (infix), 8
%^>% (infix), 8
%v>% (infix), 8

apply_rewriters, 3
as_monad (x_to_monad), 28

clear_cache, 3, *6*, *11*, *12*, *14*, *28*
combine (x_to_monad), 28
const, 4, *6*, *15*, *24*, *25*
crunch, 4

esc, 5, *13*, *17*

fail_cache, *3*, 5, *11*, *12*, *14*, *28*
false, *4*, *6*, 6, *15*, *24*, *25*
false_as_error, *4*, *6*, 6, *15*, *24*, *25*
first, 7, *14*, *22*
funnel (x_to_monad), 28

get_code (rmonad_getters), 20
get_dependency_matrix, 7
get_dependents (rmonad_getters), 20
get_depth (rmonad_getters), 20
get_doc (rmonad_getters), 20
get_error (rmonad_getters), 20
get_id (rmonad_getters), 20
get_key (rmonad_getters), 20
get_mem (rmonad_getters), 20
get_meta (rmonad_getters), 20
get_nest (rmonad_getters), 20
get_nest_depth (rmonad_getters), 20
get_notes (rmonad_getters), 20
get_OK (rmonad_getters), 20
get_parents (rmonad_getters), 20
get_prior (rmonad_getters), 20
get_summary (rmonad_getters), 20
get_tag (rmonad_getters), 20
get_time (rmonad_getters), 20
get_value (rmonad_getters), 20
get_warnings (rmonad_getters), 20
gff, 8

has_code (rmonad_checkers), 19
has_dependents (rmonad_checkers), 19
has_doc (rmonad_checkers), 19
has_error (rmonad_checkers), 19
has_mem (rmonad_checkers), 19
has_meta (rmonad_checkers), 19
has_nest (rmonad_checkers), 19
has_notes (rmonad_checkers), 19
has_parents (rmonad_checkers), 19
has_prior (rmonad_checkers), 19
has_summary (rmonad_checkers), 19
has_tag (rmonad_checkers), 19
has_time (rmonad_checkers), 19
has_value (rmonad_checkers), 19
has_warnings (rmonad_checkers), 19

infix, 8
is_rmonad, 9

loop, 10

make_cacher, *3*, *6*, 10, *11*, *12*, *14*, *28*
make_recacher, *3*, *6*, *11*, 11, *12*, *14*, *28*
memory_cache, *3*, *6*, *11*, 12, *14*, *28*
missues, *5*, *13*, 13, *17*
mtabulate, *5*, *13*, 13, *17*

no_cache, *3*, *6*, *11*, *12*, 14, *28*
nothing, *7*, 14, *22*
null_as_error, *4*, *6*, 15, *24*, *25*

plot.Rmonad, 15
print.Rmonad, 16

report, *5*, *13*, 17
rmonad, 17
rmonad-package *(*rmonad*)*, 17
rmonad_checkers, 19
rmonad_getters, 20

second, *7*, *14*, 22
size, 23
splice_function, 23

tag, 24
toss, *4*, *6*, *15*, 24, *25*
true, *4*, *6*, *15*, *24*, 25

view, 25
viewID, 26
viewIDs, 26
views, 27
void_cache, *3*, *6*, *11*, *12*, *14*, 27

x_to_monad, 28