

Package: rhylythyme (via r-universe)

October 12, 2026

Title Read and Plot 'Rhylythyme' Schedules with 'ggplot2'

Version 0.0.0.9000

Description Reads 'Rhylythyme' programs (JSON descriptions of parallel tracks of timed steps: recipes, lab protocols, run-of-shows) and plots them as Gantt charts with 'ggplot2'. Step times come from the same timing engine that <<https://www.rhylythyme.com>> uses, embedded through 'V8', so a plot agrees with the website to the second.

License Apache License (>= 2)

URL <https://github.com/rhylythyme/rhylythyme-r>,
<https://www.rhylythyme.com>

BugReports <https://github.com/rhylythyme/rhylythyme-r/issues>

Depends R (>= 4.1)

Imports curl, generics, ggplot2 (>= 3.5.0), grDevices, grid, jsonlite, jsonvalidate, stats, utils, V8

Suggests dplyr, testthat (>= 3.0.0), rlang, vdiff, withr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libssl-dev libnode-dev

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

Date/Publication 2026-10-12 03:31:08 UTC

RemoteUrl <https://github.com/rhylythyme/rhylythyme-r>

RemoteRef HEAD

RemoteSha 5eb3ffadf0bbe48a025150a5fd82ab5ad4390ce1

Contents

as.list.rhylthyme_program	2
autoplot.rhylthyme_programs	3
fortify.rhylthyme_program	4
geom_rhylthyme	5
read_rhylthyme	8
read_rhylthyme_run	9
rhylthyme_engine_version	10
rhylthyme_palettes	10
rhylthyme_programs	11
rhylthyme_scales	12
rhylthyme_tables	13
summary.rhylthyme_program	14
theme_rhylthyme	15
validate_rhylthyme	16
Index	17

as.list.rhylthyme_program
<i>The program as a list</i>

Description

The parsed program JSON, unchanged. `jsonlite::toJSON(as.list(p), auto_unbox = TRUE, null = "null", digits = NA)` turns it back into the program’s JSON, and `p$json` holds the text exactly as it was read.

Usage

```
## S3 method for class 'rhylthyme_program'
as.list(x, ...)
```

Arguments

- x A rhylthyme_program.
- ... Unused.

Value

A list.

`autoplot.rhylthyme_programs`*A finished plot of a Rhylthyme program*

Description

`ggplot(program) + geom_rhylthyme()` with `theme_rhylthyme()` and the program's name as the title. With one lane per track the fill legend only repeats the lane names, so it is left out. Add to it like any `ggplot`.

Usage

```
## S3 method for class 'rhylthyme_programs'
autoplot(object, ...)
```

```
## S3 method for class 'rhylthyme_run'
autoplot(object, ...)
```

```
## S3 method for class 'rhylthyme_program'
autoplot(object, ...)
```

Arguments

<code>object</code>	A <code>rhylthyme_program</code> .
<code>...</code>	Passed on to <code>geom_rhylthyme()</code> (e.g. <code>lanes = "task"</code> , <code>collapse_replicates = TRUE</code>).

Value

A `ggplot`.

Examples

```
library(ggplot2)
p <- read_rhylthyme(system.file("extdata", "breakfast_schedule.json", package = "rhylthyme"))
autoplot(p)
autoplot(p, lanes = "task")
```

fortify.rhylthyme_program

Turn programs into plotting data

Description

One row per step: the columns of `steps()`, plus program (a factor, for faceting), step_key (the step id, prefixed by the program within a set of programs, so it is unique), track (a factor in authored order; tracks with the same name share a level, and a colour), the lane factors lane (by track), step_lane and task_lane (see lanes in `geom_rhylthyme()`; levels run bottom to top so the first track or step is drawn at the top, and within a set each program's lanes are its own), and range_min / range_max (a variable step's shortest and longest end; NA otherwise). start, end and the ranges carry a class that makes ggplot2 use `scale_x_rhylthyme()` (or `scale_x_rhylthyme_clock()`) by default, as track does for `scale_fill_rhylthyme()` and the lanes for `scale_y_rhylthyme()`. The program rides along as the "rhylthyme_program" attribute, for `geom_rhylthyme()`'s arrows and replicate collapsing.

Usage

```
## S3 method for class 'rhylthyme_program'
fortify(model, data, ...)

## S3 method for class 'rhylthyme_programs'
fortify(model, data, ...)

## S3 method for class 'rhylthyme_run'
fortify(model, data, ...)
```

Arguments

model	A rhylthyme_program or <code>rhylthyme_programs()</code> .
data	Ignored.
...	Ignored.

Value

A data frame.

Description

geom_rhylthyme() draws one bar per step, from its start to its end, in one lane per track, with arrows for dependencies between lanes and step names on (or beside) the bars. With a program as the plot's data (ggplot(program) + geom_rhylthyme()), the aesthetics default to xmin = start, xmax = end, y = lane, fill = track, step = step_id and label = name, and the arrows come from the program's [dependencies\(\)](#); anything you map or pass yourself wins. The geom also works by name: geom = "rhylthyme".

Usage

```
geom_rhylthyme(  
  mapping = NULL,  
  data = NULL,  
  stat = "rhylthyme",  
  position = "identity",  
  ...,  
  arrows = TRUE,  
  labels = TRUE,  
  ranges = TRUE,  
  estimates = TRUE,  
  lanes = c("track", "step", "task"),  
  collapse_replicates = FALSE,  
  dependencies = NULL,  
  label_size = 3,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)  
  
stat_rhylthyme(  
  mapping = NULL,  
  data = NULL,  
  geom = "rhylthyme",  
  position = "identity",  
  ...,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() . If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of
---------	---

	the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code>. A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).
stat	The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: • A Stat ggproto subclass, for example <code>StatCount</code>. • A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>. • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The <code>position</code> argument accepts the following: • The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as <code>"jitter"</code>. • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to <code>ggplot2::layer()</code> , such as <code>colour = "white"</code> .
arrows	Draw dependency arrows between lanes, as curves the way <code>rhylthyme.com</code> draws them: a sweep into the waiting step's start, or an S-curve between the two bars when they are close. Solid for a step that waits for another; dashed when it starts before the other ends (a negative offset); dotted for a step that runs only if the other aborts.
labels	Draw each bar's label: inside the bar when it fits, else just after it when the lane has room, else not at all.
ranges	Show each variable step's range: a faded extension from its planned end to its longest, and a thin line under the bar from its shortest to its longest.
estimates	Outline, dotted, the bars whose length is an estimate (instrument steps with no duration, indefinite steps, and durations <code>rhylthyme plan</code> filled in), as <code>rhylthyme.com</code> does.

lanes	What each lane is, for a program as the data: "track" (the default; one lane per track, as on rhylthyme.com), "step" (one lane per step, a classic Gantt; step names are then on the axis, so bars get no label unless you map one) or "task" (one lane per resource type, so contention shows). In any mode, bars that overlap in a lane are stacked so none hides another.
collapse_replicates	Draw one bar per authored step instead of one per replicate instance, spanning all its instances and labelled with the count (only for a program as the data; steps() is unaffected).
dependencies	A data frame with from_step, to_step and relation (and offset), as dependencies() returns, matched to the step aesthetic. Taken from the program when the plot's data is one.
label_size	Label text size, in mm (as ggplot2::geom_text() 's size).
na.rm	If FALSE, the default, steps with no computed time are removed with a warning; if TRUE, silently.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders() .
geom	The geometric object to use to display the data for this layer. When using a <code>stat_*()</code> function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example <code>GeomPoint</code>. • A string naming the geom. To give the geom as a string, strip the function name of the <code>geom_</code> prefix. For example, to use <code>geom_point()</code>, give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.

Details

Aesthetics: `xmin`, `xmax` and `y` (required); `fill`, `colour`, `alpha`, `linewidth`, `linetype`; `label` (the text for each bar), `step` (the id the dependencies table refers to), `xlower` and `xupper` (a variable step's shortest and longest end) and `estimate` (TRUE for an estimated length). For a program these default to `range_min`, `range_max` and `estimate`.

Value

A layer to add to a plot.

Examples

```
library(ggplot2)
p <- read_rhylthyme(system.file("extdata", "breakfast_schedule.json", package = "rhylthyme"))
ggplot(p) + geom_rhylthyme()
ggplot(p) + geom_rhylthyme(labels = FALSE)
```

read_rhylthyme	<i>Read a Rhylthyme program</i>
----------------	---------------------------------

Description

Reads a Rhylthyme program (JSON), checks it against the program schema and times every step with the same engine rhylthyme.com uses.

Usage

```
read_rhylthyme(x, validate = TRUE, start_at = NULL, finish_at = NULL)
```

Arguments

x	The program: a path to a JSON file, an <code>http(s)://</code> URL (a rhylthyme.com program page, <code>.../p/<id></code> , or share link, <code>...?share=<id></code> , is read through the site's public API), a JSON string, or a list (as from <code>jsonlite::read_json()</code> , or built in R; a length-one vector becomes a JSON scalar, so wrap a one-element array in <code>list()</code> or <code>I()</code>).
validate	If TRUE, the default, a program that breaks the schema is an error listing every problem (see <code>validate_rhylthyme()</code>). FALSE reads it anyway, for drafts; steps are then timed as far as the engine can.
start_at, finish_at	Anchor the program to the clock: when it starts, or when its last step should end ("everything ready at 6 pm"). A <code>POSIXct</code> or a string <code>as.POSIXct()</code> understands, in the local time zone. Give one at most. <code>steps()</code> then adds <code>start_time</code> and <code>end_time</code> , and plots have a clock-time axis.

Value

A `rhylthyme_program`: a list with the program's name, description, `schema_version` and source, the steps, tracks and dependencies tables (see `steps()`), the parsed program and its json text, and `start_at` when anchored. Times are seconds from program start.

Examples

```
p <- read_rhylthyme(system.file("extdata", "breakfast_schedule.json", package = "rhylthyme"))
p
read_rhylthyme('{
  "programId": "tea", "name": "Tea", "resourceConstraints": [],
  "tracks": [
    {"trackId": "kettle", "name": "Kettle",
     "steps": [
       {"stepId": "boil", "name": "Boil", "duration": {"type": "fixed", "seconds": 180},
```

```

    "startTrigger": {"type": "programStart"}}]]}]')
# Everything ready at 6 pm
p <- read_rhylthyme(system.file("extdata", "breakfast_schedule.json", package = "rhylthyme"),
  finish_at = "2026-11-26 18:00")
steps(p)[c("name", "start_time", "end_time")]

```

read_rhylthyme_run	<i>Read a Rhylthyme run record</i>
--------------------	------------------------------------

Description

A run record (the runs schema) is what actually happened when a program was run: when each step really started and ended, and what ended it. It is written by the rhylthyme runner and by rhylthyme.com.

Usage

```
read_rhylthyme_run(x, program = NULL, validate = TRUE, clock = TRUE)
```

Arguments

x	The run record: a path, an <code>http(s)://</code> URL, a JSON string or a list, as for read_rhylthyme() .
program	The program that was run (a <code>rhylthyme_program</code> , or anything read_rhylthyme() accepts). Optional, but it gives each step its track, name and task, maps replicate instances to the program's step ids, and brings the dependency arrows; without it, steps are named by id and share one lane.
validate	Check the record against the runs schema (see read_rhylthyme()).
clock	If TRUE, the default, times are anchored at the record's <code>startedAt</code> : steps() adds <code>start_time</code> and <code>end_time</code> , and plots have a clock-time axis. FALSE keeps elapsed seconds.

Value

A `rhylthyme_run`: a list with `name`, `run_id`, `program_id`, `program_version`, `started_at`, `ended_at`, `outcome` (completed, aborted or abandoned), `context`, the steps table and the parsed record. The steps table has one row per step in the record: the program's `step_id` and `track` columns, the actual start, end and duration (seconds from `startedAt`; NA for a step that never started), `status` (completed, aborted, skipped, unfinished or `not_started`), `ended_by`, `failed` (an instrument command failed at least once) and `last_code` (its last reply), and the plan the runner had (`planned_start`, `planned_end`).

Examples

```
extdata <- function(f) system.file("extdata", f, package = "rhylythyme")
dinner <- read_rhylythyme(extdata("thanksgiving_one_oven.json"))
run <- read_rhylythyme_run(extdata("thanksgiving_one_oven.run.json"), program = dinner)
run
steps(run)[c("name", "start", "end", "status")]
autoplot(run)
```

rhylythyme_engine_version

Versions of the embedded engine and the schemas it reads

Description

rhylythyme_engine_version() is the version of the [@rhylythyme/timeline](#) timing engine this package carries: step times match rhylythyme.com's for that engine version. rhylythyme_schema_versions() lists the program schemaVersions [read_rhylythyme\(\)](#) accepts.

Usage

```
rhylythyme_engine_version()
```

```
rhylythyme_schema_versions()
```

Value

A character vector.

Examples

```
rhylythyme_engine_version()
rhylythyme_schema_versions()
```

rhylythyme_palettes

The web timeline's colour palettes

Description

Read from the embedded engine, so they are exactly rhylythyme.com's.

Usage

```
rhylythyme_palettes()
```

Value

A named list of character vectors of hex colours.

Examples

```
names(rhythyme_palettes())
rhythyme_palettes()$classic
```

rhythyme_programs	<i>Several Rhythyme programs together</i>
-------------------	---

Description

Combines programs so they can be tabulated and plotted together: `steps()`, `tracks()`, `dependencies()` and `resources()` stack each program's table with a program column, and a plot keeps each program's lanes separate (facet with `facet_wrap(~ program, scales = "free_y")`, or use [autoplot\(\)](#)). Step ids may repeat between programs: plots match dependencies within a program only. `c(p1, p2)` does the same.

Usage

```
rhythyme_programs(...)
```

Arguments

... rhythyme_programs, runs from [read_rhythyme_run\(\)](#), or sets of them. Argument names become the program labels; otherwise each program's name is used, made unique.

Value

A `rhythyme_programs`.

Examples

```
library(ggplot2)
breakfast <- read_rhythyme(system.file("extdata", "breakfast_schedule.json", package = "rhythyme"))
both <- rhythyme_programs(Saturday = breakfast, Sunday = breakfast)
both
autoplot(both)
```

Description

A program plotted with `geom_rhylythyme()` picks these by default: its bars are filled by track from rhylythyme.com's palette, and its time axis reads as elapsed time (or clock time, when the program was read with `start_at` or `finish_at`). Add one yourself to change its options; any other ggplot2 scale replaces them as usual.

Usage

```
scale_fill_rhylythyme(palette = "classic", ..., aesthetics = "fill")

scale_colour_rhylythyme(palette = "classic", ..., aesthetics = "colour")

scale_color_rhylythyme(palette = "classic", ..., aesthetics = "colour")

scale_x_rhylythyme(
  name = ggplot2::waiver(),
  breaks = elapsed_breaks,
  labels = ggplot2::waiver(),
  ...
)

scale_y_rhylythyme(...)

scale_x_rhylythyme_clock(name = ggplot2::waiver(), ...)
```

Arguments

<code>palette</code>	A palette name: one of <code>names(rhylythyme_palettes())</code> .
<code>aesthetics</code>	The aesthetics the scale applies to.
<code>name, breaks, labels, ...</code>	Passed on to the underlying ggplot2 scale.

Details

- `scale_fill_rhylythyme()` / `scale_colour_rhylythyme()`: the web timeline's palettes, in the same order (colours repeat, as on the web, when there are more tracks than colours). See [rhylythyme_palettes\(\)](#).
- `scale_x_rhylythyme()`: seconds from program start, labelled `m:ss`, or `h:mm` once the axis reaches an hour, at round breaks (1, 5, 15 or 30 minutes, 1 or 2 hours, ...). Titled "Elapsed (`m:ss`)" or "Elapsed (`h:mm`)" unless you give a title.
- `scale_x_rhylythyme_clock()`: clock times, for an anchored program; a `ggplot2::scale_x_datetime()` titled "Clock time".

- `scale_y_rhythm()`: the lanes, labelled by track, step or task name (within a set of `rhythm_programs()` each program's lanes are its own even when names repeat).

Value

A ggplot2 scale.

Examples

```
library(ggplot2)
p <- read_rhythm(system.file("extdata", "breakfast_schedule.json", package = "rhythm"))
ggplot(p) + geom_rhythm() + scale_fill_rhythm("cookbook")
ggplot(p) + geom_rhythm() + scale_x_rhythm(name = "Time since start")
```

rhythm_tables	<i>Tables from a Rhythm program</i>
---------------	-------------------------------------

Description

Each returns a plain data frame, ready for dplyr and ggplot2.

Usage

```
steps(x, ...)

tracks(x, ...)

dependencies(x, ...)

resources(x, ...)

## S3 method for class 'rhythm_program'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

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

Arguments

x	A rhythm_program.
...	Unused.
row.names, optional	Unused; for compatibility with <code>as.data.frame()</code> .

Details

- `steps()`: one row per step, replicate instances included, in authored order: `track_id`, `track_name`, `track_index`, `step_id`, `name`, `task`, `start` and `end` (seconds from program start, as the engine times them), `duration`, `resolved` (FALSE when the engine could not place the step), `duration_type` (fixed, variable, indefinite or instrument), `min_seconds` and `max_seconds` (the authored range), `estimate` (the planned length is a guess: instrument steps with no duration, indefinite steps, and durations rhylthyme plan filled in), `trigger_type`, and for a replicate instance `instance_of` (the authored step) and `instance_index`. For a program read with `start_at` or `finish_at`, also `start_time` and `end_time` (POSIXct). `as.data.frame()` and `tidy()` give the same table.
- `tracks()`: one row per track as the engine lays them out (replicate and template tracks included): `track_id`, `name`, `track_index`, `parent_track_id`, `template_id`, `n_steps`, `start`, `end`.
- `dependencies()`: one row per trigger that refers to another step: `from_step`, `to_step`, `relation` (`end_to_start`, `start_to_start` or `on_abort`), `offset` in seconds (buffers included; negative for a hand-off before the other step ends), `trigger_type`, and `logic` (all or any) for a trigger inside a compound one.
- `resources()`: one row per task: `task`, `max_concurrent` (NA for a task steps use that has no constraint), `description`, `n_steps`.

Value

A data frame.

Examples

```
p <- read_rhylthyme(system.file("extdata", "breakfast_schedule.json", package = "rhylthyme"))
steps(p)
tracks(p)
dependencies(p)
resources(p)
```

```
summary.rhylthyme_program
```

Summarise a Rhylthyme program

Description

Summarise a Rhylthyme program

Usage

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

Arguments

object	A rhylthyme_program.
...	Unused.

Value

A summary.rhylthyme_program: a list with name, schema_version, makespan (seconds), n_steps, n_estimated, n_unresolved, and the [tracks\(\)](#) and [resources\(\)](#) tables. Printing it shows each track's steps and time span, and each resource's limit.

Examples

```
summary(read_rhylthyme(system.file("extdata", "breakfast_schedule.json", package = "rhylthyme")))
```

theme_rhylthyme	<i>A clean theme for Rhylthyme timelines</i>
-----------------	--

Description

A minimal theme for Gantt charts: light vertical grid lines on the time axis, none between lanes, no y-axis title (the lanes name themselves) and the legend underneath.

Usage

```
theme_rhylthyme(base_size = 11, base_family = "")
```

Arguments

base_size, base_family	
	Base font size (pt) and family.

Value

A ggplot2 theme.

Examples

```
library(ggplot2)
p <- read_rhylthyme(system.file("extdata", "breakfast_schedule.json", package = "rhylthyme"))
ggplot(p) + geom_rhylthyme() + theme_rhylthyme()
```

validate_rhylthyme	<i>Check a Rhylthyme program against the schema</i>
--------------------	---

Description

Lists every problem the program schema finds, each tied to the step (or track) it is in. [read_rhylthyme\(\)](#) runs the same check and stops on the first invalid program; use this to see all the problems as a table.

Usage

```
validate_rhylthyme(x)
```

Arguments

x	A program: anything read_rhylthyme() accepts, or a rhylthyme_program.
---	---

Value

A data frame with one row per problem (none when the program is valid): track_id, step_id (NA where the problem is not inside a track or step, or the step has no id), where (the field, relative to the step or track), message and path (a JSON pointer into the program).

Examples

```
validate_rhylthyme('{"name": "Draft", "tracks": [{"trackId": "a", "steps": [{"name": "x"}]}]}')
```

Index

aes(), 5
annotation_borders(), 7
as.data.frame(), 13
as.data.frame.rhylthyme_program
 (rhylthyme_tables), 13
as.list.rhylthyme_program, 2
as.POSIXct(), 8
autoplot(), 11
autoplot.rhylthyme_program
 (autoplot.rhylthyme_programs),
 3
autoplot.rhylthyme_programs, 3
autoplot.rhylthyme_run
 (autoplot.rhylthyme_programs),
 3

dependencies(rhylthyme_tables), 13
dependencies(), 5, 7

fortify(), 6
fortify.rhylthyme_program, 4
fortify.rhylthyme_programs
 (fortify.rhylthyme_program), 4
fortify.rhylthyme_run
 (fortify.rhylthyme_program), 4

geom_rhylthyme, 5
geom_rhylthyme(), 3, 4, 12
GeomRhylthyme(geom_rhylthyme), 5
ggplot(), 6
ggplot2::geom_text(), 7
ggplot2::layer(), 6
ggplot2::scale_x_datetime(), 12

I(), 8

jsonlite::read_json(), 8

layer geom, 7
layer position, 6
layer stat, 6

list(), 8

read_rhylthyme, 8
read_rhylthyme(), 9, 10, 16
read_rhylthyme_run, 9
read_rhylthyme_run(), 11
resources(rhylthyme_tables), 13
resources(), 15
rhylthyme_engine_version, 10
rhylthyme_palettes, 10
rhylthyme_palettes(), 12
rhylthyme_programs, 11
rhylthyme_programs(), 4, 13
rhylthyme_scales, 12
rhylthyme_schema_versions
 (rhylthyme_engine_version), 10
rhylthyme_tables, 13

scale_color_rhylthyme
 (rhylthyme_scales), 12
scale_colour_rhylthyme
 (rhylthyme_scales), 12
scale_fill_rhylthyme
 (rhylthyme_scales), 12
scale_fill_rhylthyme(), 4
scale_x_rhylthyme(rhylthyme_scales), 12
scale_x_rhylthyme(), 4
scale_x_rhylthyme_clock
 (rhylthyme_scales), 12
scale_x_rhylthyme_clock(), 4
scale_y_rhylthyme(rhylthyme_scales), 12
scale_y_rhylthyme(), 4
stat_rhylthyme(geom_rhylthyme), 5
StatRhylthyme(geom_rhylthyme), 5
steps(rhylthyme_tables), 13
steps(), 4, 7–9
summary.rhylthyme_program, 14

theme_rhylthyme, 15
theme_rhylthyme(), 3

`tidy.rhylthyme_program`
 `(rhylthyme_tables)`, [13](#)
`tracks(rhylthyme_tables)`, [13](#)
`tracks()`, [15](#)

`validate_rhylthyme`, [16](#)
`validate_rhylthyme()`, [8](#)