

Package: jinx (via r-universe)

October 4, 2026

Title RLadies+ GitHub Organisation Management

Version 0.2.0.9006

Description GitHub App for managing the RLadies+ organisation. Handles onboarding and offboarding of global team members, chapter onboarding, PR review automation, org-wide reporting, Airtable sync, website automation, chapter monitoring, and interactive issue commands.

Depends R (>= 4.4.0)

License MIT + file LICENSE

URL <https://github.com/rladies/jinx>, <https://rladies.github.io/jinx/>

BugReports <https://github.com/rladies/jinx/issues>

Imports bskyr, cli, cloudflarer, gh (>= 1.4.0), glue, httr2, jsonlite, jsonvalidate, lubridate, magick, meetupr, openssl, rmarkdown, rroot, rvest, stats, stringi, stringr, utils, xml2, yaml

Suggests callr, httptest2, knitr, pkgdown, reprex, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/Needs/website rladies/cloak, rprojroot

Remotes drmowinckels/cloudflarer

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake git make libmagick++-dev gsfonts libicu-dev libuv1-dev libxml2-dev libssl-dev libnode-dev libx11-dev

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

Date/Publication 2026-10-04 10:11:51 UTC

RemoteUrl <https://github.com/rladies/jinx>

RemoteRef main

RemoteSha 4c2d1be1048286e7184633e16113b1a037557d5f

Contents

airtable_base_allowed	7
airtable_record_update	8
airtable_webhook_process	8
analytics_collect_chapter_activity	9
analytics_collect_contributor_growth	10
analytics_compute_trends	10
analytics_format_markdown	11
analytics_generate_dashboard	11
announce_create_message	12
announce_post	12
announce_post_bluesky	13
announce_post_mastodon	13
announce_post_reply	14
announce_send_newsletter	14
blog_add_pr	15
blog_check_links	15
blog_check_links_repo	16
blog_create_entry	16
blog_feed_blocks	17
blog_feed_collect	18
blog_feed_format	18
blog_feed_items	19
blog_feed_new_items	19
blog_feed_og	20
blog_feed_pending	21
blog_feed_post	21
blog_feed_run	22
blog_feed_sources	24
cf_ops_format_workers_report	24
cf_ops_get_kv_value	25
cf_ops_kv_delete	25
cf_ops_kv_put	26
cf_ops_list_kv_keys	27
cf_ops_list_workers_scripts	27
cf_ops_purge_cache	28
cf_ops_workers_invocations	29
cfp_check_deadlines	29
cfp_create_issue	30
cfp_list_open	31
cfp_recommend_speaker	31
channel_copy_apply	32
channel_copy_plan	33
channel_copy_proposals	33
channel_index_load	34
channel_promo_build	35
channel_promo_format	36

channel_promo_post	36
channel_rename_proposals	37
chapter_check_health	38
chapter_checklist_tick	38
chapter_confirm_send	39
chapter_create	40
chapter_create_pr	41
chapter_create_setup	42
chapter_create_update	42
chapter_duplicate_check	43
chapter_duplicate_comment	44
chapter_duplicate_report	45
chapter_email_approvals	46
chapter_email_provision	46
chapter_find_issue	47
chapter_initiated_date	48
chapter_mailbox_create	48
chapter_meetup_brief	49
chapter_meetup_groups	50
chapter_meetup_logo_upload	51
chapter_meetup_name	51
chapter_meetup_request	52
chapter_meetup_urlname	52
chapter_meta_fetch	53
chapter_monitor_status	54
chapter_onboard_blocker	54
chapter_onboard_state	55
chapter_onboard_summary	55
chapter_onboard_website	56
chapter_organiser_submissions	57
chapter_remind_stale	57
chapter_repo_audit	58
chapter_repo_audit_report	59
chapter_repo_create	59
chapter_repo_name	60
chapter_report_health	60
chapter_slack_prompt	61
chapter_slack_sent	61
chapter_status_reconcile	62
chapter_status_reconcile_report	63
chapter_status_report	64
chapter_team_audit	64
chapter_team_audit_report	65
chapter_team_create	66
chapter_team_description	66
chapter_team_name	67
chapter_team_slug	68
chapter_validate_files	68

chapter_validate_report	69
chunk_markdown	69
cloudflare_account_id	70
cloudflare_embed	70
cloudflare_generate	71
cloudflare_vectorize_upsert	72
cmd_attach_slack_context	72
cmd_authorize	73
cmd_execute	74
cmd_parse	74
conference_generate_report	75
conference_list_speakers	75
contributor_format	76
contributor_list	76
contributor_list_org	77
contributor_update	77
copilot_request_review	78
copilot_review_pr	78
copilot_sync_repo	79
directory_crop_image	80
directory_mark_synced	81
directory_optimize_image	82
directory_purge_submissions	82
directory_sync_airtable	83
directory_validate_filename	84
directory_verify_handles	84
event_authorize	85
event_create_summary	85
event_execute	86
event_list_chapter	86
event_parse	87
event_sync_chapters	87
gather_rag_source	88
gh_branch_upsert	88
gh_greet_contributor	89
gh_open_or_update_pr	90
gh_post_checklist	90
gh_thank_contributor	91
gh_thank_issue_contributors	92
gh_welcome_contributor	92
gha_generate_dashboard	93
gt_check_invitations	94
gt_create_offboarding	94
gt_finalize_offboarding	95
gt_finalize_onboarding	95
gt_invite	96
gt_remind_stale	96
gt_schedule_onboarding_meeting	97

guide_email_template	98
guide_template	99
il8n_check_coverage	100
il8n_get_chapter_language	100
il8n_list_languages	101
il8n_translate_template	101
il8n_validate_translations	102
li_get_version	102
li_media_upload	103
li_oauth	103
li_post_write	104
li_req	104
li_urn_me	105
load_copilot_review_config	105
load_labels_config	105
load_rag_sources	106
load_review_rules	106
load_teams_config	106
mail_send	107
meeting_poll_best	107
meeting_poll_create	108
meeting_poll_format_best	109
meeting_poll_format_created	109
meeting_poll_get	110
meeting_poll_ics	110
meeting_poll_lock	111
meetup_urlname_status	111
pending_link_consume	112
prepare_inactivity_emails	112
promo_blurb	113
promo_eligible_channels	114
promo_pick_channel	114
question_coding_declined_count	115
question_content_gaps	115
question_digest_build	116
question_digest_format	116
question_digest_post	117
question_downvoted_rank	118
question_draft_guide_snippet	118
question_feedback_format	119
question_feedback_summary	119
question_gaps_rank	120
question_log_format	121
question_log_purge	121
question_log_query	122
question_vote_apply	123
r_challenge_adversarial_check	123
r_challenge_allowed_packages	124

r_challenge_difficulties	125
r_challenge_difficulty_badge	125
r_challenge_difficulty_label	126
r_challenge_draft_build	126
r_challenge_draft_post	127
r_challenge_format_issue	128
r_challenge_format_slack	128
r_challenge_generate	129
r_challenge_issue_title	129
r_challenge_open_issue	130
r_challenge_parse	130
r_challenge_render_statement	131
r_challenge_reprex_check	131
rag_chunk_id	132
rag_index_build	132
random_emoji	133
reaction_direction	133
reaction_event_apply	134
reaction_log_increment	134
report_format_markdown	135
report_generate	135
review_check_pr_naming	136
review_run	136
rum_collect_analytics	137
rum_format_analytics	137
rum_generate_report	138
setup_channel_process	139
short_url	139
slack_api_call	140
slack_bot_token	140
slack_channel_id_lookup	141
slack_channel_index	142
slack_channel_mention	142
slack_conversations_open	143
slack_interaction_process	143
slack_invite_approval_checklist_blocks	144
slack_invite_batch	144
slack_invite_request	145
slack_invite_request_blocks	146
slack_post_message	146
slack_response_url_post	147
slack_subscribe_rss	148
slack_workspace_for_team	148
tags2hash	149
validate_blog_entry	149
validate_directory_entries	150
validate_directory_pr	150
website_collect_analytics	151

airtable_base_allowed

7

website_format_analytics

152

website_generate_report

152

website_merge_pending

153

website_publish_report

153

welcome_message_render

154

welcome_send

155

Index

156

airtable_base_allowed

Check whether an Airtable base is within the configured token’s scope

Description

Caches the allowed-bases list in KV (same AIRTABLE_BASES namespace and 1h TTL the Worker used) rather than calling Airtable’s Meta API on every webhook. R port of `airtable_base_is_allowed()`/`airtable_allowed_bases_get()` from the deleted `worker/src/airtable-meta.js`.

Usage

```
airtable_base_allowed(  
  base_id,  
  namespace_id = airtable_bases_namespace_id(),  
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),  
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),  
  api_key = Sys.getenv("AIRTABLE_API_KEY")  
)
```

Arguments

- base_id

Airtable base ID to check.
- namespace_id

KV namespace ID for AIRTABLE_BASES.
- account_id

Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
- api_token

Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
- api_key

Airtable API key. Defaults to env AIRTABLE_API_KEY.

Value

TRUE if the base is within the token’s scope.

`airtable_record_update`*Update fields on an Airtable record*

Description

R's first Airtable write primitive - everything in R/airtable-sync.R is read-only. R port of `airtable_record_update()` from the deleted worker/src/airtable-invite.js.

Usage

```
airtable_record_update(  
  base_id,  
  table_id,  
  record_id,  
  fields,  
  api_key = Sys.getenv("AIRTABLE_API_KEY")  
)
```

Arguments

<code>base_id</code>	Airtable base ID.
<code>table_id</code>	Airtable table ID.
<code>record_id</code>	Airtable record ID.
<code>fields</code>	Named list of fields to update.
<code>api_key</code>	Airtable API key. Defaults to env <code>AIRTABLE_API_KEY</code> .

Value

Invisibly, TRUE.

`airtable_webhook_process`*Process an Airtable invite-request webhook*

Description

The `airtable_webhook` event handler registered in `jinx_events()`. Checks the base is within the configured Airtable token's scope, builds the approval-request card, and posts it to the community-invite channel. R port of the remainder of `airtable_webhook_handle()` (after shared-secret verification, which stays in the Worker) from the deleted worker/src/airtable-invite.js.

Usage

```
airtable_webhook_process(
  email,
  name = NULL,
  chapter = NULL,
  record_id,
  base_id,
  table_id,
  channel = Sys.getenv("SLACK_COMMUNITY_INVITE_CHANNEL")
)
```

Arguments

email	Applicant email.
name	Applicant name, or NULL.
chapter	Chapter name, or NULL.
record_id	Airtable record ID.
base_id	Airtable base ID.
table_id	Airtable table ID.
channel	Slack channel to post the request to. Defaults to env SLACK_COMMUNITY_INVITE_CHANNEL.

Value

Invisibly, TRUE if posted, FALSE if the base was rejected.

analytics_collect_chapter_activity
Collect chapter activity data

Description

Queries the GitHub API for repository activity metrics across all non-meetup repos in the organization.

Usage

```
analytics_collect_chapter_activity(
  org = "rladies",
  months = 12,
  exclude_pattern = "^meetup-"
)
```

Arguments

org	GitHub organization.
months	Number of months of history.
exclude_pattern	Regex pattern to exclude repos.

Value

Data frame with columns: chapter, month, commits, prs, issues.

analytics_collect_contributor_growth
<i>Collect contributor growth data</i>

Description

Collect contributor growth data

Usage

```
analytics_collect_contributor_growth(org = "rladies", months = 12)
```

Arguments

org	GitHub organization.
months	Number of months of history.

Value

Data frame with columns: month, new_contributors, total_contributors, active_repos.

analytics_compute_trends
<i>Compute activity trends</i>

Description

Calculates month-over-month changes and rolling averages from chapter activity data.

Usage

```
analytics_compute_trends(activity_data)
```

Arguments

activity_data Data frame from [analytics_collect_chapter_activity\(\)](#).

Value

Data frame with trend columns added.

analytics_format_markdown

Format analytics as markdown

Description

Format analytics as markdown

Usage

```
analytics_format_markdown(trends, contributor_growth)
```

Arguments

trends Data frame from [analytics_compute_trends\(\)](#).

contributor_growth

 Data frame from [analytics_collect_contributor_growth\(\)](#).

Value

Formatted markdown string.

analytics_generate_dashboard

Generate analytics dashboard

Description

Orchestrates data collection, trend computation, and formatting.

Usage

```
analytics_generate_dashboard(org = "rladies", months = 12, output_path = NULL)
```

Arguments

org GitHub organization.

months Number of months of history.

output_path Optional path to write JSON data.

Value

Named list with trends, growth, and markdown (invisibly).

announce_create_message	<i>Create a formatted announcement message</i>
-------------------------	--

Description

Create a formatted announcement message

Usage

```
announce_create_message(frontmatter, uri)
```

Arguments

frontmatter	Named list from YAML front matter (needs title, description, tags).
uri	URL of the post.

Value

Formatted message string.

announce_post	<i>Announce a blog post across multiple platforms</i>
---------------	---

Description

Reads YAML front matter from a blog post, creates an announcement message, and posts to selected platforms.

Usage

```
announce_post(  
  post,  
  platforms = c("bluesky", "linkedin", "mastodon"),  
  newsletter = TRUE  
)
```

Arguments

post	File path to the blog post (markdown with YAML front matter).
platforms	Character vector of platforms to post to. Defaults to all: "bluesky", "linkedin", "mastodon".
newsletter	Logical, whether to also send a newsletter. Defaults to TRUE.

Value

Invisibly returns a named list with the posting result for each platform.

announce_post_bluesky *Post an announcement to Bluesky*

Description

Post an announcement to Bluesky

Usage

```
announce_post_bluesky(text, image = NULL, image_alt = NULL)
```

Arguments

text	Post text.
image	Optional file path to an image.
image_alt	Alt text for the image.

Value

Post response (invisibly).

announce_post_mastodon
Post an announcement to Mastodon

Description

Post an announcement to Mastodon

Usage

```
announce_post_mastodon(text, image = NULL, image_alt = NULL)
```

Arguments

text	Post text.
image	Optional file path to an image.
image_alt	Alt text for the image.

Value

Post response (invisibly).

announce_post_reply	<i>Post a reply comment on an issue or PR</i>
---------------------	---

Description

Posts as the authenticated user (jinx[bot] when using a GitHub App token).

Usage

```
announce_post_reply(owner, repo, issue_number, body)
```

Arguments

owner	Repository owner.
repo	Repository name.
issue_number	Issue or PR number.
body	Comment body (markdown).

Value

The API response (invisibly).

announce_send_newsletter	<i>Send a newsletter via ConvertKit</i>
--------------------------	---

Description

Send a newsletter via ConvertKit

Usage

```
announce_send_newsletter(frontmatter, url)
```

Arguments

frontmatter	Named list with seo, image, title fields.
url	URL of the post.

Value

API response (invisibly).

blog_add_pr	<i>Add a community blog entry via a pull request</i>
-------------	--

Description

Builds an entry from the URL's metadata and opens a PR adding it to the awesome-rladies-creations content directory. If an entry for the URL's domain already exists, no PR is opened.

Usage

```
blog_add_pr(  
  url,  
  org = "rladies",  
  repo = "awesome-rladies-creations",  
  content_path = "data/content",  
  language = "en",  
  author_name = NULL,  
  base = "main"  
)
```

Arguments

url	Blog URL.
org, repo, content_path	Location of the blog entry JSON files.
language	Language code. Defaults to "en".
author_name	Author name. When NULL, taken from page metadata.
base	Base branch. Defaults to "main".

Value

A list with status ("created" or "exists"), filename, and url (the PR URL when created).

blog_check_links	<i>Check blog URLs and RSS feeds for broken links</i>
------------------	---

Description

Check blog URLs and RSS feeds for broken links

Usage

```
blog_check_links(blogs_path)
```

Arguments

blogs_path Path to directory containing blog JSON files.

Value

A data frame with columns file, url, rss_feed, url_status, rss_status.

blog_check_links_repo *Check community blog links from the awesome-rladies-creations repo*

Description

Lists the blog entry JSON files in the content directory via the GitHub API and checks each entry's url and rss_feed for broken links.

Usage

```
blog_check_links_repo(
  org = "rladies",
  repo = "awesome-rladies-creations",
  content_path = "data/content"
)
```

Arguments

org, repo, content_path
 Location of the blog entry JSON files.

Value

A data frame with columns file, url, rss_feed, url_status, rss_status.

blog_create_entry *Auto-generate a blog entry JSON file from a URL*

Description

Fetches OpenGraph metadata from the URL and writes a JSON entry compatible with awesome-rladies-creations format.

Usage

```
blog_create_entry(url, language = "en", author_name, output_dir = ".")
```

Arguments

url	Blog URL.
language	Language code. Defaults to "en".
author_name	Author name. Required.
output_dir	Directory to write the JSON file. Defaults to ".".

Value

File path of the created JSON (invisibly).

blog_feed_blocks	<i>Build a new post as Slack blocks</i>
------------------	---

Description

The heading, a linked bold title and the post's own description go in one section; the preview image follows as its own block so Slack renders it full width rather than as a thumbnail; the byline sits in a context block, which Slack renders small and grey so it reads as attribution rather than as part of the post.

Usage

```
blog_feed_blocks(item, source, og = list())
```

Arguments

item	A one-row data frame from <code>blog_feed_items()</code> .
source	A one-row data frame from <code>blog_feed_sources()</code> .
og	Preview data from <code>blog_feed_og()</code> . Omitted fields are fine.

Details

Every piece of text here comes from a feed or a page that contributors control, so all of it is escaped - the link markup around the title is the only markup the message is allowed to introduce.

Value

A list of Slack Block Kit blocks.

blog_feed_collect	<i>Collect every recent post across the curated feeds</i>
-------------------	---

Description

Polls each source in turn, keeping going when an individual feed is down so one dead blog can't silence the rest, then returns the items sorted oldest first so a channel reads in publication order.

Usage

```
blog_feed_collect(
  entries = NULL,
  max_age_days = 14,
  now = as.numeric(Sys.time())
)
```

Arguments

entries	Curated content entries. Fetched when NULL.
max_age_days	Only announce items published within this window.
now	Current time, as a unix timestamp.

Details

Deliberately knows nothing about what any workspace has already announced: both blog channels want the same posts, so a run polls the feeds once and each workspace filters the result against its own seen-set with `blog_feed_pending()`. Polling per workspace would double the requests every contributor's blog receives.

Value

A list with posts (a list of item/source pairs, oldest first), feedless (titles with no `rss_feed`), and sources (how many feeds were polled).

blog_feed_format	<i>Format one new post as a Slack mrkdwn message</i>
------------------	--

Description

Titles and author names come from feeds and curated JSON that contributors control, so both are escaped before they go into a broadcast message - that neutralises injected links and `<!channel><!everyone>` mass-pings while leaving the message's own link markup intact.

Usage

```
blog_feed_format(item, source)
```

Arguments

item	A one-row data frame from <code>blog_feed_items()</code> .
source	A one-row data frame from <code>blog_feed_sources()</code> .

Value

Character scalar Slack mrkdown message.

blog_feed_items	<i>Parse an RSS or Atom document into a data frame of items</i>
-----------------	---

Description

Matched on `local-name()` rather than a namespace prefix: the feeds in the curated list are a mix of RSS 2.0, Atom, and RSS 1.0/RDF written by half a dozen site generators, and only the element names are reliably shared between them.

Usage

```
blog_feed_items(xml, base = NULL, max_items = 25L)
```

Arguments

xml	Feed document as text.
base	Feed URL, used to resolve relative item links.
max_items	Keep at most this many items from the top of the feed.

Value

A data frame with columns `id`, `title`, `link`, and `date` (a unix timestamp, 0 when the feed gave no usable date).

blog_feed_new_items	<i>Keep the items worth announcing</i>
---------------------	--

Description

Drops anything already announced and anything older than `max_age_days`

- a blog joining the curated list arrives with its whole back catalogue in the feed, and without the age gate its first run would announce years of posts at once. Items with no usable date are treated as too old for the same reason.

Usage

```
blog_feed_new_items(
  items,
  seen = character(),
  max_age_days = 14,
  now = as.numeric(Sys.time())
)
```

Arguments

<code>items</code>	Data frame from <code>blog_feed_items()</code> .
<code>seen</code>	Character vector of item ids already announced.
<code>max_age_days</code>	Only announce items published within this window.
<code>now</code>	Current time, as a unix timestamp.

Value

A subset of `items`, oldest first.

<code>blog_feed_og</code>	<i>Read a post's Open Graph preview data</i>
---------------------------	--

Description

One extra request per *new* post - not per feed, and not per item - so at a post a week this is nothing next to polling the feeds themselves. Every failure returns empty: a post with no preview is still worth announcing, so nothing here is allowed to abort the run.

Usage

```
blog_feed_og(url)
```

Arguments

<code>url</code>	The post's URL.
------------------	-----------------

Value

A list with `description`, `image`, and `image_alt`, each a character scalar that may be `""`.

blog_feed_pending	<i>Keep the posts one workspace still owes its channel</i>
-------------------	--

Description

Keep the posts one workspace still owes its channel

Usage

```
blog_feed_pending(posts, seen = character(), limit = 20L)
```

Arguments

posts	List of item/source pairs from blog_feed_collect() .
seen	Character vector of item ids already announced there.
limit	Most posts to announce in one run.

Value

The subset still to announce, oldest first.

blog_feed_post	<i>Announce new community posts in one workspace's blog channel</i>
----------------	---

Description

Replaces the Slack Feed app's per-blog subscriptions with one run driven by awesome-rladies-creations: a blog added to the curated list is announced without anyone touching Slack.

Usage

```
blog_feed_post(
  workspace = c("community", "organiser"),
  channel = env_default("SLACK_BLOG_CHANNEL", "blogs-by-rladies"),
  dry_run = FALSE,
  posts = NULL,
  entries = NULL,
  max_age_days = 14,
  limit = 20L,
  slack_token = NULL,
  og_fetch = blog_feed_og_memo(),
  namespace_id = slack_tokens_namespace_id(),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")
)
```

Arguments

workspace	Which Slack workspace to post in.
channel	Channel to post to. Defaults to env SLACK_BLOG_CHANNEL, falling back to "blogs-by-rladies".
dry_run	When TRUE, log the messages and record nothing.
posts	Collected posts from <code>blog_feed_collect()</code> . Collected here when NULL.
entries	Curated content entries. Fetched when NULL.
max_age_days	Only announce items published within this window.
limit	Most posts to announce in one run.
slack_token	Bot token for workspace. Resolved from the workspace when unset.
og_fetch	Reader for a post's preview data. Defaults to a per-call cache; <code>blog_feed_run()</code> passes one shared across workspaces so a post's page is read once however many channels announce it.
namespace_id	KV namespace ID for SLACK_TOKENS.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.

Details

Announced ids are recorded in KV per workspace, and only after a successful post - a failed run re-announces nothing and loses nothing.

Use `blog_feed_run()` to serve both workspaces, which polls the feeds once for the pair rather than once each.

Value

Invisibly, the number of posts announced.

blog_feed_run	<i>Serve every workspace's blog channel from one poll of the feeds</i>
---------------	--

Description

The scheduled entry point. Both channels want the same posts, so the feeds are polled once and each workspace is then filtered against its own seen-set - at a ten-minute cadence, polling per workspace would double the requests every contributor's blog receives for no gain.

Usage

```

blog_feed_run(
  workspaces = c("community", "organiser"),
  channel = env_default("SLACK_BLOG_CHANNEL", "blogs-by-rladies"),
  dry_run = FALSE,
  seed = FALSE,
  entries = NULL,
  max_age_days = 14,
  limit = 20L,
  namespace_id = slack_tokens_namespace_id(),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")
)

```

Arguments

workspaces	Workspaces to announce in.
channel	Channel to post to. Defaults to env SLACK_BLOG_CHANNEL, falling back to "blogs-by-rladies".
dry_run	When TRUE, log the messages and record nothing.
seed	When TRUE, record every current feed item as announced in each workspace without posting. Run once at cutover so the first real run doesn't repeat what the Feed app already posted.
entries	Curated content entries. Fetched when NULL.
max_age_days	Only announce items published within this window.
limit	Most posts to announce in one run.
namespace_id	KV namespace ID for SLACK_TOKENS.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.

Details

One workspace failing does not stop the others: each is attempted, and the error is re-raised at the end so the run still goes red.

Value

Invisibly, a named integer of posts announced per workspace.

blog_feed_sources	<i>Split curated entries into pollable sources and gaps</i>
-------------------	---

Description

An entry is pollable when it carries an `rss_feed`, whatever its type

- YouTube channels and plain websites serve the same Atom/RSS XML as blogs do. Entries without a feed can't be polled at all, so they are returned separately for the run to report rather than dropped silently.

Usage

```
blog_feed_sources(entries)
```

Arguments

entries	List of entry records from the curated content list.
---------	--

Value

A list with sources (a data frame with columns `title`, `site`, `feed`, `author`, and `type`) and `feedless` (a character vector of titles).

cf_ops_format_workers_report	<i>Format Workers invocation metrics as a markdown table</i>
------------------------------	--

Description

Format Workers invocation metrics as a markdown table

Usage

```
cf_ops_format_workers_report(df)
```

Arguments

df	Data frame from <code>cf_ops_workers_invocations()</code> .
----	---

Value

Character string with markdown-formatted report.

cf_ops_get_kv_value	<i>Read a single value from a Cloudflare KV namespace</i>
---------------------	---

Description

Read-only, for interactive/console use during an incident — see [cf_ops_list_kv_keys\(\)](#) for why this isn't wired into a /jinx command.

Usage

```
cf_ops_get_kv_value(  
    account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),  
    namespace_id,  
    key_name,  
    token = cf_ops_token()  
)
```

Arguments

account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
namespace_id	KV namespace ID.
key_name	Key to read.
token	Cloudflare API token. Defaults to env CLOUDFLARE_OPS_API_TOKEN, falling back to CLOUDFLARE_API_TOKEN.

Value

Character string with the stored value.

cf_ops_kv_delete	<i>Delete a single value from a Cloudflare KV namespace</i>
------------------	---

Description

Wraps [cloudflarer::cf_delete_kv_value\(\)](#). See [cf_ops_kv_put\(\)](#) for why this is narrowly scoped rather than a general-purpose tool.

Usage

```
cf_ops_kv_delete(  
    account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),  
    namespace_id,  
    key_name,  
    token = cf_ops_token()  
)
```

Arguments

account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
namespace_id	KV namespace ID.
key_name	Key to delete.
token	Cloudflare API token. Defaults to env CLOUDFLARE_OPS_API_TOKEN, falling back to CLOUDFLARE_API_TOKEN.

Value

The API response (invisibly).

cf_ops_kv_put	<i>Write a single value to a Cloudflare KV namespace</i>
---------------	--

Description

Wraps `cloudflarer::cf_put_kv_value()`. Unlike `cf_ops_list_kv_keys()/cf_ops_get_kv_value()`, this is not a general-purpose "read/write any key" tool exposed to a `/jinx` command — it's called internally by event/command handlers against a small set of hardcoded key patterns (e.g. `pending_link:{email}`, `channel_index:{team_id}`), the same scoping the exfiltration concern in those two functions' docs warns about.

Usage

```
cf_ops_kv_put(
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  namespace_id,
  key_name,
  value,
  ttl_seconds = NULL,
  token = cf_ops_token()
)
```

Arguments

account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
namespace_id	KV namespace ID.
key_name	Key to write.
value	Character value to store.
ttl_seconds	Optional expiration TTL in seconds.
token	Cloudflare API token. Defaults to env CLOUDFLARE_OPS_API_TOKEN, falling back to CLOUDFLARE_API_TOKEN.

Value

The API response (invisibly).

cf_ops_list_kv_keys	<i>List keys in a Cloudflare KV namespace</i>
---------------------	---

Description

Read-only, for interactive/console use during an incident — not wired into a /jinx command, since KV namespaces on this account (SLACK_TOKENS, AIRTABLE_BASES) hold operational secrets, and a chat-triggered arbitrary-namespace read is a real exfiltration path.

Usage

```
cf_ops_list_kv_keys(  
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),  
  namespace_id,  
  prefix = NULL,  
  token = cf_ops_token()  
)
```

Arguments

account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
namespace_id	KV namespace ID.
prefix	Optional key-name prefix filter.
token	Cloudflare API token. Defaults to env CLOUDFLARE_OPS_API_TOKEN, falling back to CLOUDFLARE_API_TOKEN.

Value

Data frame of key records.

cf_ops_list_workers_scripts	<i>List Workers scripts deployed on the account</i>
-----------------------------	---

Description

Thin wrapper over `cloudflarer::cf_list_workers_scripts()`.

Usage

```
cf_ops_list_workers_scripts(  
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),  
  token = cf_ops_token()  
)
```

Arguments

account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
token	Cloudflare API token. Defaults to env CLOUDFLARE_OPS_API_TOKEN, falling back to CLOUDFLARE_API_TOKEN.

Value

Data frame of Workers script records.

cf_ops_purge_cache	<i>Purge specific URLs or prefixes from the Cloudflare cache</i>
--------------------	--

Description

Thin wrapper over `cloudflarer::cf_purge_cache()` that deliberately does not expose `purge_everything` — a maintainer who genuinely needs a full-zone wipe calls `cloudflarer::cf_purge_cache(purge_everything = TRUE)` directly. That extra step is intentional friction on the most dangerous call this integration can make.

Usage

```
cf_ops_purge_cache(
  zone_id = Sys.getenv("CLOUDFLARE_ZONE_ID"),
  files = NULL,
  prefixes = NULL,
  token = cf_ops_token()
)
```

Arguments

zone_id	Cloudflare zone ID. Defaults to env CLOUDFLARE_ZONE_ID.
files	Optional character vector of URLs to purge.
prefixes	Optional character vector of URL prefixes to purge (without scheme), e.g. "rladies.org/blog".
token	Cloudflare API token. Defaults to env CLOUDFLARE_OPS_API_TOKEN, falling back to CLOUDFLARE_API_TOKEN.

Value

A named list with the purge job id.

cf_ops_workers_invocations

Query Workers invocation/error/CPU metrics

Description

Thin wrapper over `cloudflarer::cf_workers_invocations()`, defaulting `script_name` to "jinx" (matches `wrangler.jsonc`'s worker name) rather than requiring the caller to know it.

Usage

```
cf_ops_workers_invocations(
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  since = Sys.Date() - 1,
  until = Sys.Date(),
  script_name = "jinx",
  token = cf_ops_token()
)
```

Arguments

<code>account_id</code>	Cloudflare account ID. Defaults to env <code>CLOUDFLARE_ACCOUNT_ID</code> .
<code>since, until</code>	Date range. Defaults to the last day.
<code>script_name</code>	Worker script name. Defaults to "jinx".
<code>token</code>	Cloudflare API token. Defaults to env <code>CLOUDFLARE_OPS_API_TOKEN</code> , falling back to <code>CLOUDFLARE_API_TOKEN</code> .

Value

Data frame with columns `date`, `script`, `requests`, `errors`, `subrequests`, `cpu_p50_us`, `cpu_p99_us`.

cfp_check_deadlines

Check CFP deadlines and post reminders

Description

Check CFP deadlines and post reminders

Usage

```
cfp_check_deadlines(org = "rladies", repo = "global-team", warn_days = 7)
```

Arguments

org	GitHub organization.
repo	Repository where CFP issues are tracked.
warn_days	Number of days before deadline to warn.

Value

Data frame of approaching CFPs (invisibly).

cfp_create_issue	<i>Create a CFP tracking issue</i>
------------------	------------------------------------

Description

Create a CFP tracking issue

Usage

```
cfp_create_issue(  
  conference,  
  deadline,  
  url,  
  topics = character(0),  
  org = "rladies",  
  repo = "global-team"  
)
```

Arguments

conference	Conference name.
deadline	Submission deadline (YYYY-MM-DD).
url	CFP URL.
topics	Character vector of topic tags.
org	GitHub organization.
repo	Repository to create the issue in.

Value

Issue URL (invisibly).

cfp_list_open	<i>List open CFPs tracked as GitHub issues</i>
---------------	--

Description

List open CFPs tracked as GitHub issues

Usage

```
cfp_list_open(org = "rladies", repo = "global-team")
```

Arguments

org	GitHub organization.
repo	Repository where CFP issues are tracked.

Value

Data frame with columns: conference, deadline, url, number, status.

cfp_recommend_speaker	<i>Recommend a speaker for a conference</i>
-----------------------	---

Description

Adds a speaker recommendation as a comment on the CFP tracking issue.

Usage

```
cfp_recommend_speaker(
  conference,
  speaker_name,
  speaker_github = NULL,
  expertise = character(0),
  org = "rladies",
  repo = "global-team"
)
```

Arguments

conference	Conference name (matched against open CFP issues).
speaker_name	Speaker name or GitHub username.
speaker_github	Optional GitHub username.
expertise	Character vector of expertise areas.
org	GitHub organization.
repo	Repository where CFP issues are tracked.

Value

Comment URL (invisibly).

channel_copy_apply	<i>Apply a channel copy plan</i>
--------------------	----------------------------------

Description

Only rows with status "apply" are sent to Slack. Everything else is returned untouched so the caller keeps a full record of what was skipped and why.

Usage

```
channel_copy_apply(  
  plan,  
  token,  
  dry_run = TRUE,  
  skip = character(),  
  join = TRUE,  
  include_drift = FALSE,  
  user_token = NULL  
)
```

Arguments

- plan A plan from `channel_copy_plan()`.
- token Slack bot token.
- dry_run When TRUE (the default), report what would be sent without calling Slack.
- skip Channel names to leave alone.
- join Join channels the bot is not a member of first. Slack refuses `conversations.setTopic`, `setPurpose` and `rename` with `not_in_channel` otherwise.
- include_drift Also apply rows whose live value has changed since the review. Off by default: drift means someone edited the channel after the copy was written, so applying overwrites the newer text with older reviewed text.
- user_token Optional Slack user token used for `rename` rows only. Slack refuses `conversations.rename` from a bot token for a channel the bot did not create; a grant from a workspace owner passes that check. Topics and descriptions always use token.

Value

plan with applied and error columns added.

channel_copy_plan	<i>Plan the channel copy and rename pass for a workspace</i>
-------------------	--

Description

Compares the reviewed proposals against the workspace's live state and classifies every intended change. A row is "unchanged" when Slack already holds the proposed value, "drift" when the live value no longer matches what the review recorded (someone edited it since, so the proposal may be stale), "missing" when the channel is not in the workspace, and "apply" otherwise.

Usage

```
channel_copy_plan(
  workspace = c("organiser", "community"),
  index = NULL,
  proposals = NULL,
  renames = NULL
)
```

Arguments

workspace	Either "organiser" or "community".
index	Optional pre-fetched channel index, for testing.
proposals, renames	Optional proposal data frames, for testing.

Details

Renames are planned last so a rename cannot hide the channel from the copy rows in the same pass.

Value

A data frame of planned changes.

channel_copy_proposals	<i>Read the reviewed channel copy proposals</i>
------------------------	---

Description

Read the reviewed channel copy proposals

Usage

```
channel_copy_proposals(workspace = c("organiser", "community"), path = NULL)
```

Arguments

workspace	Either "organiser" or "community".
path	Optional path to the proposals CSV, for testing.

Value

A data frame of proposals for that workspace, excluding channels that do not exist yet.

channel_index_load	<i>Load (and cache) a workspace's full channel name-to-id index</i>
--------------------	---

Description

Callers resolving several channel names at once (e.g. [welcome_message_render\(\)](#)'s starter-channel list) should call this once and look names up in the result, rather than calling [slack_channel_id_lookup\(\)](#) per name - each call independently checks the KV cache, so looking up N names one-by-one means N redundant KV reads (and N cache-miss races) for data that doesn't change between them.

Usage

```
channel_index_load(
  team_id,
  workspace,
  namespace_id = slack_tokens_namespace_id(),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")
)
```

Arguments

team_id	Slack team id.
workspace	"organiser" or "community".
namespace_id	KV namespace ID for SLACK_TOKENS.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.

Value

A named list mapping channel name to channel id, or NULL if the channel list couldn't be fetched.

channel_promo_build	<i>Build the weekly channel-spotlight message</i>
---------------------	---

Description

Lists the community workspace's public channels, keeps those with a description, picks the next one round-robin (state stored in KV), and drafts a Jinx-voiced invitation for it - falling back to the plain description if the model call fails.

Usage

```
channel_promo_build(
  team_id = Sys.getenv("SLACK_COMMUNITY_TEAM_ID"),
  target_channel = env_default("SLACK_PROMO_CHANNEL", "general"),
  skip = promo_skip_channels(),
  namespace_id = slack_tokens_namespace_id(),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),
  model = workers_ai_chat_model()
)
```

Arguments

team_id	Community Slack team id. Defaults to env SLACK_COMMUNITY_TEAM_ID.
target_channel	Channel the spotlight is posted to (and never promotes). Defaults to env SLACK_PROMO_CHANNEL, falling back to "general".
skip	Additional channel names to exclude. Defaults to the comma-separated env SLACK_PROMO_SKIP.
namespace_id	KV namespace ID for SLACK_TOKENS.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
model	Workers AI chat model used for drafting.

Value

A list with text (the Slack message), channel_id, channel_name, and recent (the updated round-robin state to persist), or NULL when no channel is eligible.

channel_promo_format	<i>Format a channel spotlight as a Slack mrkdwn message</i>
----------------------	---

Description

Links the featured channel in the header and appends the invitation. The blurb is escaped with `escape_markdown()` before it goes out: a channel's description is user-controlled and flows through the model into a broadcast message, so this neutralises injected links and `<!channel><!everyone>` mass-pings while leaving `*bold*/_italic_` intact.

Usage

```
channel_promo_format(id, name, blurb)
```

Arguments

id	Featured channel id.
name	Featured channel name (without #).
blurb	Invitation text from <code>promo_blurb()</code> or <code>promo_fallback_blurb()</code> .

Value

Character scalar Slack mrkdwn message.

channel_promo_post	<i>Build and post the weekly channel spotlight to the community workspace</i>
--------------------	---

Description

Posts one channel's spotlight to the target channel, then records the pick in KV so next week's run features a different channel. The round-robin state is only advanced after a successful post, so a failed run doesn't burn a channel's turn.

Usage

```
channel_promo_post(
  team_id = Sys.getenv("SLACK_COMMUNITY_TEAM_ID"),
  target_channel = env_default("SLACK_PROMO_CHANNEL", "general"),
  slack_token = slack_bot_token("community"),
  skip = promo_skip_channels(),
  namespace_id = slack_tokens_namespace_id(),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),
  model = workers_ai_chat_model()
)
```

Arguments

team_id	Community Slack team id. Defaults to env SLACK_COMMUNITY_TEAM_ID.
target_channel	Channel the spotlight is posted to (and never promotes). Defaults to env SLACK_PROMO_CHANNEL, falling back to "general".
slack_token	Community bot token. Defaults to <code>slack_bot_token()</code> for the community workspace.
skip	Additional channel names to exclude. Defaults to the comma-separated env SLACK_PROMO_SKIP.
namespace_id	KV namespace ID for SLACK_TOKENS.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
model	Workers AI chat model used for drafting.

Value

Invisibly, TRUE if a spotlight was posted, FALSE if there was nothing eligible to promote.

channel_rename_proposals
<i>Read the reviewed channel rename proposals</i>

Description

Read the reviewed channel rename proposals

Usage

```
channel_rename_proposals(workspace = c("organiser", "community"), path = NULL)
```

Arguments

workspace	Either "organiser" or "community".
path	Optional path to the proposals CSV, for testing.

Value

A data frame of old/new channel names for that workspace.

chapter_check_health	<i>Check chapter health across the organization</i>
----------------------	---

Description

Identifies chapters that have been inactive (no events) for a specified number of months.

Usage

```
chapter_check_health(  
  months = 12,  
  org = "rladies",  
  data_repo = "meetup_archive"  
)
```

Arguments

months	Number of months without activity to consider inactive. Defaults to 12.
org	GitHub organization. Defaults to "rladies".
data_repo	Repository containing chapter/event data. Defaults to "meetup_archive".

Value

A data frame with columns chapter, last_event, status, and months_inactive.

chapter_checklist_tick	<i>Tick a checklist item on an onboarding issue</i>
------------------------	---

Description

Tick a checklist item on an onboarding issue

Usage

```
chapter_checklist_tick(  
  issue_number,  
  pattern,  
  org = "rladies",  
  onboarding_repo = "new-chapters-onboarding"  
)
```

Arguments

issue_number	Onboarding issue number.
pattern	Text identifying the item to tick.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository holding onboarding issues.

Value

TRUE when an item was ticked, FALSE otherwise.

chapter_confirm_send *Confirm a completed onboarding to the chapter organisers*

Description

Sends the confirmation email from `jinx@rladies.org`, cc'ing `chapters@rladies.org` so the onboarding inbox keeps the thread, and notes on the issue that it went out.

Usage

```
chapter_confirm_send(
  issue_number,
  to,
  city = NULL,
  email = NULL,
  meetup_url = NULL,
  force = FALSE,
  org = "rladies",
  onboarding_repo = "new-chapters-onboarding"
)
```

Arguments

issue_number	Onboarding issue number.
to	Organiser email addresses.
city	Chapter city. Read from the issue when NULL.
email	Chapter email address to quote in the message.
meetup_url	Meetup URL to quote in the message.
force	Send even with checklist steps outstanding.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository holding onboarding issues.

Details

Refuses to send while checklist steps are outstanding, since the email tells the organisers their chapter is set up. Pass `force = TRUE` to override, which is occasionally right - a step may be recorded elsewhere - but it should be a decision rather than an accident.

Recipient addresses are passed in rather than read from the issue: organiser emails are deliberately not stored in the issue metadata, and this function does not put them back there either. The issue comment records how many organisers were written to, not who.

Value

The sent message id (invisibly).

chapter_create	Create a new chapter JSON file
----------------	--------------------------------

Description

Generates a chapter JSON entry for the website.

Usage

```
chapter_create(
  city,
  country,
  organizers,
  region = NULL,
  social_media = list(),
  status = "prospective",
  output_dir = "."
)
```

Arguments

city	City name.
country	Country name.
organizers	Character vector of organizer names.
region	State, region or province, or NULL.
social_media	Named list of social media links (e.g. <code>list(meetup = "...", email = "city@rladies.org")</code>).
status	Chapter status. Defaults to "prospective".
output_dir	Directory to write the JSON file.

Value

File path of the created JSON (invisibly).

chapter_create_pr	Create a chapter JSON PR on the website repo
-------------------	--

Description

Generates the chapter JSON entry, validates it against the bundled schema, and opens a PR adding it to the website. A prospective chapter has no Meetup group or chapter email yet, so both are optional and are simply left out of the entry.

Usage

```
chapter_create_pr(  
  city,  
  country,  
  region = NULL,  
  meetup_urlname = NULL,  
  email = NULL,  
  organizers = character(0),  
  status = "prospective",  
  social_media = list(),  
  org = "rladies",  
  website_repo = "rladies.github.io",  
  team_reviewers = "leadership"  
)
```

Arguments

city	Chapter city.
country	Chapter country.
region	State/region/province (optional).
meetup_urlname	Meetup group URL name, or NULL when the group does not exist yet.
email	Chapter email address, or NULL when it does not exist yet.
organizers	Character vector of organizer names.
status	Chapter status. Defaults to "prospective".
social_media	Named list of social media handles (optional).
org	GitHub organization. Defaults to "rladies".
website_repo	Website repository name.
team_reviewers	Teams to request review from. Defaults to "leadership", which the onboarding process requires.

Value

PR URL (invisibly).

chapter_create_setup *Create a new chapter setup issue*

Description

Opens a tracking issue in the new-chapters-onboarding repo with the full checklist for setting up a new RLadies+ chapter.

Usage

```
chapter_create_setup(
  city,
  country,
  organizers,
  region = NULL,
  org = "rladies",
  onboarding_repo = "new-chapters-onboarding"
)
```

Arguments

city	Chapter city name.
country	Chapter country.
organizers	Character vector of organizer names.
region	State/region/province, or NULL. Used to narrow the duplicate check's proximity search.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository for chapter onboarding issues.

Value

Issue URL (invisibly).

chapter_create_update *Create a chapter update issue*

Description

Opens a tracking issue for updating an existing chapter's infrastructure.

Usage

```
chapter_create_update(  
  city,  
  country,  
  org = "rladies",  
  onboarding_repo = "new-chapters-onboarding"  
)
```

Arguments

city	Chapter city name.
country	Chapter country.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository for chapter onboarding issues.

Value

Issue URL (invisibly).

chapter_duplicate_check

Check whether a requested chapter already exists or has neighbours

Description

Performs the first step of the chapter onboarding checklist: searches the website's data/chapters entries for a chapter in the same city, for chapters with a near-identical city name, and for chapters in the same country within radius_km of the requested city.

Usage

```
chapter_duplicate_check(  
  city,  
  country,  
  region = NULL,  
  radius_km = 100,  
  org = "rladies",  
  website_repo = "rladies.github.io"  
)
```

Arguments

city	Requested chapter city.
country	Requested chapter country.
region	State/region/province, or NULL.
radius_km	Radius in kilometres for the proximity search. Set to NULL to skip geocoding entirely.
org	GitHub organization. Defaults to "rladies".
website_repo	Repository holding data/chapters.

Details

Only chapters in the requested country, or whose filename contains the requested city slug, are fetched in full, so the check costs one directory listing plus a handful of file reads.

Value

A data frame with one row per match and columns city, country, status, email, meetup, match, and distance_km. match is one of "exact", "similar", or "nearby".

chapter_duplicate_comment

Post the duplicate check onto an onboarding issue

Description

Runs `chapter_duplicate_check()` and comments the rendered report on the issue. Failures are reported but never abort onboarding; the check is an aid to the team, not a gate.

Usage

```
chapter_duplicate_comment(
  issue_number,
  city,
  country,
  region = NULL,
  org = "rladies",
  onboarding_repo = "new-chapters-onboarding"
)
```

Arguments

issue_number	Onboarding issue number.
city	Requested chapter city.
country	Requested chapter country.

region	State/region/province, or NULL.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository holding onboarding issues.

Value

The rendered report (invisibly), or NULL on failure.

chapter_duplicate_report

Render a duplicate check as an onboarding issue comment

Description

Turns the output of `chapter_duplicate_check()` into the markdown the onboarding team reads first: whether the city already has a chapter, which nearby chapters should be cc'd, and which names are close enough to be worth a second look.

Usage

```
chapter_duplicate_report(matches, city, country, radius_km = 100)
```

Arguments

matches	Data frame from <code>chapter_duplicate_check()</code> .
city	Requested chapter city.
country	Requested chapter country.
radius_km	Radius used for the proximity search.

Value

Markdown body as a single string.

`chapter_email_approvals`*Find the approvals on a chapter onboarding issue*

Description

An approval is a comment containing the approval marker, written by a confirmed member of an approving team. Bot comments never count, so jinx cannot approve its own work.

Usage

```
chapter_email_approvals(  
  issue_number,  
  org = "rladies",  
  onboarding_repo = "new-chapters-onboarding"  
)
```

Arguments

<code>issue_number</code>	Onboarding issue number.
<code>org</code>	GitHub organization. Defaults to "rladies".
<code>onboarding_repo</code>	Repository holding onboarding issues.

Value

Character vector of approver logins, possibly empty.

`chapter_email_provision`*Provision a chapter mailbox once a human has approved it*

Description

Reads the chapter from the onboarding issue's own machine-readable block, checks that a member of an approving team has left an approval comment, and only then asks the worker to create the mailbox. The address is posted back on the issue and the email step of the checklist is ticked.

Usage

```
chapter_email_provision(  
  issue_number,  
  org = "rladies",  
  onboarding_repo = "new-chapters-onboarding"  
)
```

Arguments

issue_number	Onboarding issue number.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository holding onboarding issues.

Details

The approval is a separate, deliberate act on the specific issue: being allowed to run the command is not on its own enough to create an account, so a stray dispatch or a mistaken command cannot provision anything.

Value

The created address (invisibly).

chapter_find_issue	<i>Find the open onboarding issue for a chapter</i>
--------------------	---

Description

Matches on the issue title, which onboarding issues open as ", chapter setup".

Usage

```
chapter_find_issue(  
  city,  
  org = "rladies",  
  onboarding_repo = "new-chapters-onboarding"  
)
```

Arguments

city	Chapter city.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository holding onboarding issues.

Value

The issue number, or NULL when no open issue matches.

chapter_initiated_date

When a chapter first appeared in the website repository

Description

Used as the start of the six-month window in which a chapter with no events counts as unbegun rather than inactive.

Usage

```
chapter_initiated_date(files, repo = ".", bulk_import = as.Date("2023-01-04"))
```

Arguments

files	Chapter file paths, relative to repo.
repo	Path to the website repository.
bulk_import	Date of the bulk import to treat as unknown, or NULL to keep every date.

Details

Two things make this fragile, and both are handled here. Renaming a chapter file would otherwise reset its date to the rename, so the lookup follows renames. A shallow clone has no history to read, so the caller's checkout needs `fetch-depth: 0`; without it every chapter looks as though it appeared today.

Dates before `bulk_import` are not meaningful: 118 chapters were added to the website in one import on 2023-01-04, so that date records the import rather than when any of those chapters started. Those come back as NA.

Value

A Date vector, NA where the date is unknown or untrusted.

chapter_mailbox_create

Create a chapter mailbox in Google Workspace

Description

Asks the jinx worker to provision <city>@rladies.org in the /Chapters organisational unit. The Workspace service account key lives in the worker rather than in CI, so this call carries only the worker API key: the most a stolen CI token can do is ask for a mailbox whose name passes the worker's validation.

Usage

```
chapter_mailbox_create(
  city,
  given_name = NULL,
  family_name = NULL,
  base_url = "https://jinx.rldadies.org",
  api_key = worker_api_key()
)
```

Arguments

city	Chapter city. The worker transliterates and validates it, and refuses reserved addresses.
given_name	Given name on the account. Defaults to "RLadies+".
family_name	Family name on the account. Defaults to the city.
base_url	Worker base URL.
api_key	Worker API key. Defaults to the JINX_WORKER_API_KEY environment variable.

Details

The signing and Directory API calls could be done here in R - openssl signs RS256 perfectly well - but that would require the service account key on the Actions runner that runs this function, which is the thing the current arrangement avoids. `worker/src/workspace.js` carries the full reasoning; it is a deliberate exception to moving worker logic into this package.

The generated password is never returned. The account is created with change-password-at-next-login set, and the onboarding team issues the handover from the Admin console.

Value

The created address (invisibly).

chapter_meetup_brief *Render the Meetup setup brief for a chapter*

Description

Everything the Meetup Pro team needs in one comment: the exact group name and URL, the settings the organisational guidelines call for, and the standard description ready to paste.

Usage

```
chapter_meetup_brief(city, country, status = NULL)
```

Arguments

city	Chapter city.
country	Chapter country.
status	Result of <code>meetup_urlname_status()</code> , or NULL to skip the availability line.

Value

Markdown body as a single string.

chapter_meetup_groups *Chapters that have a Meetup group, read from the website data*

Description

The website's data/chapters entries are the source of truth for which chapters exist and which Meetup group each uses, so the sync derives its list from there rather than from a hand-maintained one that would drift the day it was written.

Usage

```
chapter_meetup_groups(  
  org = "rladies",  
  website_repo = "rladies.github.io",  
  statuses = c("active", "prospective")  
)
```

Arguments

org	GitHub organization. Defaults to "rladies".
website_repo	Repository holding data/chapters.
statuses	Chapter statuses to include.

Details

Retired chapters are excluded by default: they have no upcoming events and querying them only spends requests.

Value

A data frame with columns file, city, country, status, and urlname.

`chapter_meetup_logo_upload`*Set a chapter's Meetup group photo to the RLadies+ logo*

Description

Meetup uploads in two steps: a mutation registers the photo and returns a signed `uploadUrl`, then the bytes are sent to that URL. The mutation is named for event photos, but `photoType` carries a `GROUP_PHOTO` value and `setAsMain` makes it the group's image.

Usage

```
chapter_meetup_logo_upload(urlname, image = chapter_meetup_logo_path())
```

Arguments

<code>urlname</code>	Meetup group urlname.
<code>image</code>	Path to the image. Defaults to the bundled RLadies+ profile logo.

Value

The uploaded image path reported by Meetup (invisibly).

`chapter_meetup_name` *The Meetup group name for a chapter*

Description

The Meetup group name for a chapter

Usage

```
chapter_meetup_name(city)
```

Arguments

<code>city</code>	Chapter city, used as written apart from trimming.
-------------------	--

Value

The group name.

chapter_meetup_request

Post the Meetup setup brief on an onboarding issue

Description

Reads the chapter from the issue's own machine-readable block, so in the usual case only the issue number is needed.

Usage

```
chapter_meetup_request(
  issue_number,
  city = NULL,
  country = NULL,
  org = "rladies",
  onboarding_repo = "new-chapters-onboarding"
)
```

Arguments

issue_number	Onboarding issue number.
city	Chapter city. Read from the issue when NULL.
country	Chapter country. Read from the issue when NULL.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository holding onboarding issues.

Value

The rendered brief (invisibly).

chapter_meetup_urlname

The Meetup URL name for a chapter

Description

meetup.com/rladies-<city>, hyphenated for multi-word cities and transliterated the way the website filenames already are. This value and the group name cannot be changed after the group is created, so getting them right first time is the whole point of rendering them here rather than having someone type them.

Usage

```
chapter_meetup_urlname(city)
```

Arguments

city Chapter city.

Value

The urlname, without the meetup.com prefix.

chapter_meta_fetch	<i>Read the machine-readable block off an onboarding issue</i>
--------------------	--

Description

Read the machine-readable block off an onboarding issue

Usage

```
chapter_meta_fetch(  
  issue_number,  
  org = "rladies",  
  onboarding_repo = "new-chapters-onboarding"  
)
```

Arguments

issue_number Onboarding issue number.
org GitHub organization. Defaults to "rladies".
onboarding_repo Repository holding onboarding issues.

Value

A named list of chapter details, or NULL when the issue carries no block.

chapter_monitor_status

Monitor chapter activity status

Description

Determines chapter status based on meetup event data. Chapters are classified as active, inactive, or unbegun based on their last event and founding date.

Usage

```
chapter_monitor_status(data_path = NULL, inactive_months = 6, org = "rladies")
```

Arguments

data_path	Path to a CSV export from Meetup Pro Dashboard, or NULL to fetch from meetup_archive.
inactive_months	Months without events to consider inactive. Defaults to 6.
org	GitHub organization.

Value

Data frame with chapter status information.

chapter_onboard_blocker

Which team is holding a chapter onboarding up

Description

The owner of the first unchecked item, or NA when the checklist is complete or carries no owner.

Usage

```
chapter_onboard_blocker(state)
```

Arguments

state	Data frame from chapter_onboard_state() .
-------	---

Value

A team slug such as "rladies/email", or NA_character_.

chapter_onboard_state *Read the checklist state of a chapter onboarding issue*

Description

Parses the task list in an onboarding issue body into one row per checklist item, carrying the section it sits under and the team that section says is responsible for it. Nested items inherit their parent's owner.

Usage

```
chapter_onboard_state(
  issue_number,
  org = "rladies",
  onboarding_repo = "new-chapters-onboarding"
)
```

Arguments

issue_number Onboarding issue number.

org GitHub organization. Defaults to "rladies".

onboarding_repo Repository holding onboarding issues.

Value

A data frame with columns section, owner, item, and done.

chapter_onboard_summary

Summarise an onboarding checklist as markdown

Description

Reports progress per section, names the next unchecked item and who owns it, and says plainly when everything is done.

Usage

```
chapter_onboard_summary(state, title = "Chapter onboarding")
```

Arguments

state Data frame from [chapter_onboard_state\(\)](#).

title Chapter title to put in the heading.

Value

Markdown body as a single string.

chapter_onboard_website

Add a prospective chapter to the website from its onboarding issue

Description

Opens the website PR with the chapter's prospective entry, requests review from the leadership team, links the PR on the onboarding issue, and ticks the website step of the checklist.

Usage

```
chapter_onboard_website(
  issue_number,
  city = NULL,
  country = NULL,
  region = NULL,
  organizers = NULL,
  org = "rladies",
  onboarding_repo = "new-chapters-onboarding",
  website_repo = "rladies.github.io"
)
```

Arguments

issue_number	Onboarding issue number.
city	Chapter city. Read from the issue when NULL.
country	Chapter country. Read from the issue when NULL.
region	State/region/province. Read from the issue when NULL.
organizers	Character vector of organizer names. Read from the issue when NULL.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository holding onboarding issues.
website_repo	Website repository name.

Details

A prospective entry deliberately carries no chapter email and no Meetup group: neither exists at this point in onboarding, and the entry is filled in as those steps complete.

Anything left NULL is read from the issue's own machine-readable block, so in the usual case only the issue number is needed.

Value

The website PR URL (invisibly).

 chapter_organiser_submissions

Find organiser form submissions for a chapter

Description

The form's chapter field is free text and inconsistently written - "Cologne, Germany" beside "Wake Forest, North Carolina EE.UU" - so matching is by city slug appearing anywhere in it, and the caller is expected to show a human what was found rather than act on it.

Usage

```
chapter_organiser_submissions(city, api_key = Sys.getenv("AIRTABLE_API_KEY"))
```

Arguments

city	Chapter city.
api_key	Airtable API key.

Value

A data frame with record_id, name, chapter, status, and type; empty when nothing matches.

 chapter_remind_stale *Nudge the team holding up a stale chapter onboarding*

Description

Finds open onboarding issues with no activity for days days and comments on each, naming the team that owns the first unchecked step rather than pinging everyone. Onboardings stall on the email and Meetup Pro teams more than anywhere else, so the ping is targeted.

Usage

```
chapter_remind_stale(
  org = "rladies",
  onboarding_repo = "new-chapters-onboarding",
  days = 14,
  labels = c("new chapter", "chapter update")
)
```

Arguments

org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository holding onboarding issues.
days	Days without activity before nudging. Defaults to 14.
labels	Issue labels to consider.

Value

Invisibly, a list of the issues nudged.

chapter_repo_audit	<i>Audit the GitHub references in chapter data</i>
--------------------	--

Description

Each chapter may carry a `social_media$github` value. Two shapes are legitimate: `owner/repo` for a chapter that keeps its files in a repository, and a bare organisation name for a chapter that runs its own GitHub organisation. Anything that resolves to neither is a dead link on the chapter's page.

Usage

```
chapter_repo_audit(chapters_dir, org = "rladies")
```

Arguments

chapters_dir	Directory of chapter JSON files.
org	Organisation the conventional repository would live in.

Details

Where a reference is dead and the conventional `<org>/meetup-presentations_<city>` repository exists, that is reported as the likely correction.

Value

A data frame with columns `file`, `github`, `state` and `suggestion`. `state` is `"repo"`, `"org"`, `"url"`, `"prefixed"` or `"missing"`. A `"url"` row resolves but is written as a full URL; a `"prefixed"` row is an organisation the chapter owns, written as though it were a repository under `org`. Both carry the same reference in the expected shape as their suggestion. Only rows needing attention are returned.

`chapter_repo_audit_report`*Render a chapter GitHub audit as markdown*

Description

Render a chapter GitHub audit as markdown

Usage

```
chapter_repo_audit_report(audit)
```

Arguments

`audit` Data frame from `chapter_repo_audit()`.

Value

A markdown string.

`chapter_repo_create`*Create a chapter's presentations repository*

Description

Opt-in: not every chapter wants one, and only 111 of the 122 existing teams have a repository, so this is never called as part of onboarding without someone asking for it.

Usage

```
chapter_repo_create(team_slug, city, country, region = NULL, org = "rladies")
```

Arguments

`team_slug` Team slug, from `chapter_team_slug()`.
`city` Chapter city.
`country` Chapter country.
`region` State/region/province, or NULL.
`org` GitHub organization. Defaults to "rladies".

Details

The repository is public and the chapter's team is given admin, matching every existing chapter repository - organisers administer their own.

Value

The repository name (invisibly).

chapter_repo_name	<i>The presentations repository name for a chapter</i>
-------------------	--

Description

meetup-presentations_<team slug>, matching the 127 repositories the org already has. Hyphens are kept: san-diego, sao-paulo and kansas-city all have them, and sanfrancisco is the outlier rather than the rule.

Usage

```
chapter_repo_name(team_slug)
```

Arguments

team_slug Team slug, from [chapter_team_slug\(\)](#).

Value

The repository name.

chapter_report_health	<i>Generate a chapter health report</i>
-----------------------	---

Description

Analyzes chapter activity data and returns a formatted markdown summary.

Usage

```
chapter_report_health(months = 6, org = "rladies")
```

Arguments

months Inactivity threshold in months. Defaults to 6.
org GitHub organization. Defaults to "rladies".

Value

Markdown report body (invisibly), or NULL if no data.

chapter_slack_prompt	<i>Prompt the onboarding issue for the organiser Slack invite</i>
----------------------	---

Description

Prompt the onboarding issue for the organiser Slack invite

Usage

```
chapter_slack_prompt(  
  issue_number,  
  city = NULL,  
  org = "rladies",  
  onboarding_repo = "new-chapters-onboarding"  
)
```

Arguments

issue_number	Onboarding issue number.
city	Chapter city. Read from the issue when NULL.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository holding onboarding issues.

Value

The posted body (invisibly).

chapter_slack_sent	<i>Record that the organisers Slack invite was sent</i>
--------------------	---

Description

Record that the organisers Slack invite was sent

Usage

```
chapter_slack_sent(  
  issue_number,  
  org = "rladies",  
  onboarding_repo = "new-chapters-onboarding"  
)
```

Arguments

issue_number	Onboarding issue number.
org	GitHub organization. Defaults to "rladies".
onboarding_repo	Repository holding onboarding issues.

Value

TRUE when the checklist step was ticked.

chapter_status_reconcile

Reconcile stored chapter status against actual Meetup activity

Description

The website's status field is set by hand, so it drifts from what the chapters are actually doing. This compares it against the event history in `meetup_archive` and reports the two mismatches worth acting on: a chapter running events that is not marked active, and a chapter marked active that has gone quiet.

Usage

```
chapter_status_reconcile(
  chapters_dir,
  health = NULL,
  months = 6,
  retire_months = 12,
  initiated = NULL
)
```

Arguments

chapters_dir	Directory of chapter JSON files, typically <code>data/chapters</code> in the website repository.
health	Output of <code>chapter_check_health()</code> , or NULL to fetch it. Passed in to avoid refetching when reconciling repeatedly.
months	Months without an event before a chapter is flagged. Defaults to 6, matching the guide.
retire_months	Months without an event before a chapter should simply be marked inactive. Defaults to 12.
initiated	Optional Date vector, one per file in <code>chapters_dir</code> , from <code>chapter_initiated_date()</code> . A chapter younger than <code>retire_months</code> is never marked inactive, because it has not had time to run anything yet. NA counts as old enough.

Details

The categories follow the guide: a chapter is active when it has run an event in the last months, and inactive when it has not.

Chapters create their events on Meetup even when they promote them elsewhere, precisely so activity can be tracked, so an absence of Meetup events is a real signal rather than an artefact of where a chapter advertises. That holds for every chapter, so one with no Meetup group at all is included here rather than skipped: having no group means it never began, or stopped before it started.

Value

A data frame with one row per chapter that has a urlname,

A data frame with one row per chapter, with columns file, urlname, stored, last_event, months_inactive and action. action is "promote to active", "flag quiet", "mark inactive", or NA when nothing needs doing.

`chapter_status_reconcile_report`*Render a status reconciliation as markdown*

Description

Render a status reconciliation as markdown

Usage

```
chapter_status_reconcile_report(reconciled, months = 6)
```

Arguments

reconciled	Data frame from <code>chapter_status_reconcile()</code> .
months	Months used for the inactivity window, for the wording.

Value

A markdown string.

`chapter_status_report` *Report the state of a chapter onboarding*

Description

Accepts either an issue number or a city name, and renders the checklist summary that `/jinx chapter-status` replies with.

Usage

```
chapter_status_report(
  ref,
  org = "rladies",
  onboarding_repo = "new-chapters-onboarding"
)
```

Arguments

<code>ref</code>	Issue number, or a city name to look one up by.
<code>org</code>	GitHub organization. Defaults to "rladies".
<code>onboarding_repo</code>	Repository holding onboarding issues.

Value

Markdown body as a single string.

`chapter_team_audit` *Compare the chapter teams against the website chapter data*

Description

Reports rather than repairs. Renaming a team changes its URL and where its notifications go, so which of these to act on is a human decision.

Usage

```
chapter_team_audit(
  org = "rladies",
  website_repo = "rladies.github.io",
  parent_team = "chapters",
  statuses = "active"
)
```

Arguments

org	GitHub organization. Defaults to "rladies".
website_repo	Repository holding data/chapters.
parent_team	Slug of the parent team.
statuses	Chapter statuses expected to have a team.

Details

Each row is one of:

- ok - the team matches a chapter's urlname
- slug_mismatch - a chapter matches by city, but its urlname would give a different slug
- no_chapter - no chapter in the website data explains this team
- missing_team - an active chapter with a Meetup group and no team

Value

A data frame with finding, team, display, chapter, and expected_slug.

`chapter_team_audit_report`*Render a team audit as markdown*

Description

Render a team audit as markdown

Usage

```
chapter_team_audit_report(audit)
```

Arguments

audit	Data frame from <code>chapter_team_audit()</code> .
-------	---

Value

Markdown body as a single string.

chapter_team_create	<i>Create a chapter's GitHub team</i>
---------------------	---------------------------------------

Description

Creates the team nested under the chapters parent, named so that its slug matches the chapter's Meetup urlname, with the chapter spelled out in the description.

Usage

```
chapter_team_create(
  urlname,
  city,
  country,
  region = NULL,
  org = "rladies",
  parent_team = "chapters",
  overrides = character(0)
)
```

Arguments

urlname	Meetup urlname for the chapter.
city	Chapter city.
country	Chapter country.
region	State/region/province, or NULL.
org	GitHub organization. Defaults to "rladies".
parent_team	Slug of the parent team.
overrides	Passed to <code>chapter_team_slug()</code> .

Value

The created team's slug (invisibly).

chapter_team_description	<i>The chapter description shown on the team</i>
--------------------------	--

Description

Free text, so unlike the name it can spell the chapter out in full without changing the slug.

Usage

```
chapter_team_description(city, country, region = NULL)
```

Arguments

city	Chapter city.
country	Chapter country.
region	State/region/province, or NULL.

Value

The description.

chapter_team_name	<i>The display name for a chapter's GitHub team</i>
-------------------	---

Description

GitHub derives a team's slug from its name and gives no way to set the slug directly, so the name has to be chosen such that it slugifies back to the chapter's urlname. That rules out "Portland, Oregon, USA" as a name - it would produce portland-oregon-usa rather than pdx.

Usage

```
chapter_team_name(slug, acronyms = chapter_team_acronyms())
```

Arguments

slug	Team slug, from <code>chapter_team_slug()</code> .
acronyms	Slugs to render upper case rather than title case.

Details

The readable description of the chapter goes in the team's description instead, which is free text and does not affect the slug.

Value

The display name.

chapter_team_slug	<i>The GitHub team slug for a chapter</i>
-------------------	---

Description

The Meetup urlname, minus the `rladies-` prefix. This is the convention 108 of the 122 existing chapter teams already follow, and it is the only chapter field that is unique: seven city names are shared by more than one chapter, and it is the urlname that separates them - `london` from `ldnont`, `pdx` from `portland-maine`.

Usage

```
chapter_team_slug(urlname, overrides = character(0))
```

Arguments

urlname	Meetup urlname, with or without the <code>rladies-</code> prefix.
overrides	Named character vector mapping a urlname to a slug, for chapters whose urlname makes a poor team name. Warsaw's is <code>Spotkania-Entuzjastow-R-Warsaw-R-Users-Group-Meet</code>

Value

The team slug, or `NA_character_` when there is no urlname.

chapter_validate_files	<i>Validate website chapter data files</i>
------------------------	--

Description

Runs the checks that keep `data/chapters/*.json` on the website consistent with the conventions `jinx` relies on elsewhere: the schema, the filename derivation, and the agreement between urlname and `social_media$meetup`.

Usage

```
chapter_validate_files(paths, added = character())
```

Arguments

paths	Character vector of paths to chapter JSON files.
added	Character vector of paths (a subset of paths) that the pull request adds rather than modifies. Filename mismatches are errors for added files and warnings for modified ones, because 32 files already in the repository predate the convention.

Details

Designed to run over the files changed in a pull request, so that pre-existing drift in untouched files does not block unrelated work.

Value

A data frame with columns file, level ("error" or "warning"), check, and message. Zero rows when everything passes.

chapter_validate_report	<i>Render chapter validation issues as markdown</i>
-------------------------	---

Description

Render chapter validation issues as markdown

Usage

```
chapter_validate_report(issues, n_files = NA_integer_)
```

Arguments

- | | |
|---------|---|
| issues | Data frame returned by chapter_validate_files() . |
| n_files | Number of files that were checked. |

Value

A markdown string suitable for a pull request comment.

chunk_markdown	<i>Chunk markdown into retrieval-sized pieces</i>
----------------	---

Description

Splits a markdown document by H1/H2 sections, then packs paragraphs into chunks of roughly target_chars characters. Returns a list of chunk records with heading, title, and lastmod metadata.

Usage

```
chunk_markdown(markdown, meta, target_chars = 1800, min_chars = 200)
```

Arguments

markdown	Markdown text (may include YAML frontmatter).
meta	Named list with repo, path, url, fallback_title, optional date, lastmod (both unix seconds).
target_chars	Approximate target chunk size.
min_chars	Minimum chunk size; smaller chunks are dropped.

Value

List of chunk records (one list per chunk).

cloudflare_account_id	<i>Discover the Cloudflare account ID for a token</i>
-----------------------	---

Description

Fails if the token has access to zero or more than one account.

Usage

```
cloudflare_account_id(api_token)
```

Arguments

api_token	Cloudflare API token.
-----------	-----------------------

Value

Account ID string.

cloudflare_embed	<i>Embed texts with a Cloudflare Workers AI model</i>
------------------	---

Description

Calls the Cloudflare REST endpoint `accounts/{id}/ai/run/{model}` and returns the embedding vectors. Custom because `cloudflare` does not wrap Workers AI inference; built on its `cloudflarer::cf_request()` and `cloudflarer::cf_resp()` for consistent auth and error handling.

Usage

```
cloudflare_embed(  
  texts,  
  account_id,  
  api_token,  
  model = "@cf/baai/bge-base-en-v1.5"  
)
```

Arguments

texts	Character vector of texts.
account_id	Cloudflare account ID.
api_token	Cloudflare API token.
model	Workers AI embedding model.

Value

List of numeric vectors, one per input text.

cloudflare_generate	<i>Generate chat-completion text with a Cloudflare Workers AI model</i>
---------------------	---

Description

Calls the Cloudflare REST endpoint `accounts/{id}/ai/run/{model}` with a chat-style messages array and returns the generated text. Sibling of `cloudflare_embed()` - same auth/retry/error-handling idiom, built on `cloudflarer::cf_request()` and `cloudflarer::cf_resp()` because cloudflarer does not wrap Workers AI inference.

Usage

```
cloudflare_generate(  
  messages,  
  account_id,  
  api_token,  
  model = workers_ai_chat_model(),  
  max_tokens = 256  
)
```

Arguments

messages	List of chat messages, each <code>list(role = , content =)</code> .
account_id	Cloudflare account ID.
api_token	Cloudflare API token.
model	Workers AI chat/instruct model.
max_tokens	Maximum tokens to generate.

Value

Character scalar with the model's response text.

cloudflare_vectorize_upsert

Upsert vectors into a Cloudflare Vectorize index

Description

Custom because `cloudflarer` does not wrap Vectorize v2; built on its `cloudflarer::cf_request()` and `cloudflarer::cf_resp()` for consistent auth and error handling.

Usage

```
cloudflare_vectorize_upsert(vectors, account_id, api_token, index_name)
```

Arguments

vectors	List of vector records, each with id, values, metadata.
account_id	Cloudflare account ID.
api_token	Cloudflare API token.
index_name	Vectorize index name.

Value

The unwrapped result payload from the Cloudflare response.

cmd_attach_slack_context

Attach Slack context fields to a parsed command

Description

A handful of dispatched commands ("feedback", "setup-channel") act on the Slack workspace/channel the command was run from, rather than only on fields parsed from the command text itself. Owning that merge here - mirroring how `event_parse()` owns shaping the slack-event payload - keeps the calling workflow YAML a thin, untested plumbing layer instead of the place command semantics live.

Usage

```
cmd_attach_slack_context(command, team_id, channel_id, channel_name)
```

Arguments

command	Parsed command list from <code>cmd_parse()</code> , or NULL.
team_id	Slack team id the command was run from.
channel_id	Slack channel id the command was run from.
channel_name	Slack channel display name.

Value

command with team_id/channel_id/channel_name added, or NULL unchanged if command was NULL.

cmd_authorize	<i>Authorize a parsed command before execution</i>
---------------	--

Description

Privileged commands (those with a `jinx_gated` keyword in the command registry, i.e. anything not labelled `jinx_safe`) may only be run by members of the global team, identified via the Airtable member directory. Read-only commands are always allowed. The check fails closed: an unknown actor is denied, and a directory lookup error is also denied (with a distinct "try again" message).

Usage

```
cmd_authorize(
    command,
    actor = NULL,
    source = c("github", "slack"),
    workspace = NULL,
    api_key = Sys.getenv("AIRTABLE_API_KEY")
)
```

Arguments

command	Parsed command list from <code>cmd_parse()</code> , or NULL.
actor	The requesting actor: a GitHub login when source is "github", or a Slack user id when source is "slack".
source	Origin of the command, "github" or "slack".
workspace	Originating Slack workspace for source = "slack": "organiser" or "community". Ignored for GitHub commands.
api_key	Airtable API key used for the directory lookup.

Details

Slack identity is only trusted in the organisers workspace. The community workspace is openly joinable and its user ids come from a different Slack team, so an id from there must not authorize privileged actions; privileged Slack commands are only honored when workspace is "organiser".

Value

A list with `ok` (logical) and `message` (a refusal string when `ok` is FALSE, otherwise NULL).

cmd_execute	<i>Execute a parsed jinx command</i>
-------------	--------------------------------------

Description

Returns the response message as a character string. The caller is responsible for routing the message to the right destination (GitHub issue comment, Slack, R console, etc.).

Usage

```
cmd_execute(command)
```

Arguments

command	Parsed command list from cmd_parse() .
---------	--

Value

Character string with the response message.

cmd_parse	<i>Parse a jinx command from an issue comment</i>
-----------	---

Description

Parses commands like:

- `/jinx invite @username to website`
- `/jinx offboard @username from blog`
- `/jinx report weekly`
- `/jinx help`

Usage

```
cmd_parse(body)
```

Arguments

body	Character string, the comment body.
------	-------------------------------------

Value

A named list with action and action-specific fields, or NULL if the comment is not a jinx command.

`conference_generate_report`*Generate conference coordination report*

Description

Summarizes active CFPs, speaker recommendations, and upcoming deadlines.

Usage

```
conference_generate_report(org = "rladies", repo = "global-team")
```

Arguments

<code>org</code>	GitHub organization.
<code>repo</code>	Repository where CFP issues are tracked.

Value

Formatted markdown string.

`conference_list_speakers`*List speaker recommendations for a conference*

Description

List speaker recommendations for a conference

Usage

```
conference_list_speakers(conference, org = "rladies", repo = "global-team")
```

Arguments

<code>conference</code>	Conference name.
<code>org</code>	GitHub organization.
<code>repo</code>	Repository where CFP issues are tracked.

Value

Data frame with columns: speaker, expertise, recommended_by.

contributor_format	<i>Generate a contributors markdown section</i>
--------------------	---

Description

Creates a markdown table or image grid of contributors for inclusion in README or other documents.

Usage

```
contributor_format(contributors, format = c("table", "grid"), cols = 7)
```

Arguments

contributors	Data frame from <code>contributor_list()</code> .
format	Output format: "table" or "grid".
cols	Number of columns for grid format. Defaults to 7.

Value

Character string with markdown content.

contributor_list	<i>List contributors for a repository</i>
------------------	---

Description

Collects all contributors from merged PRs and commits, returning a deduplicated list with contribution counts.

Usage

```
contributor_list(owner, repo, include_bots = FALSE)
```

Arguments

owner	Repository owner.
repo	Repository name.
include_bots	Whether to include bot accounts. Defaults to FALSE.

Value

Data frame with login, contributions, avatar_url, and profile_url columns, sorted by contribution count.

contributor_list_org *Collect contributors across multiple repos*

Description

Aggregates contributor data across all non-meetup repos in the org.

Usage

```
contributor_list_org(org = "rladies", exclude_pattern = "^meetup-")
```

Arguments

org	GitHub organization.
exclude_pattern	Regex to exclude repos.

Value

Data frame with login, repos, contributions, avatar_url, and profile_url. contributions is the sum across all repos; repos is the count of distinct repos contributed to.

contributor_update *Generate and update a contributors list for a repo*

Description

Fetches contributors and commits the rendered list directly to the default branch. No PR is opened.

Usage

```
contributor_update(
  owner,
  repo,
  file_path = ".github/contributors.md",
  format = "grid",
  branch = "main"
)
```

Arguments

owner	Repository owner.
repo	Repository name.
file_path	Path in the repo to update with the contributor list. Defaults to ".github/contributors.md".
format	Format for the contributor list.
branch	Branch to commit to. Defaults to "main".

Value

Commit URL if changes were made, NULL otherwise (invisibly).

copilot_request_review	<i>Request GitHub Copilot as a reviewer on a pull request</i>
------------------------	---

Description

Request GitHub Copilot as a reviewer on a pull request

Usage

```
copilot_request_review(  
  owner,  
  repo,  
  pr_number,  
  reviewer = copilot_reviewer_login()  
)
```

Arguments

owner	Repository owner.
repo	Repository name.
pr_number	Pull request number.
reviewer	Copilot reviewer login.

Value

TRUE if the request was accepted, FALSE otherwise.

copilot_review_pr	<i>Summon a GitHub Copilot review on a pull request</i>
-------------------	---

Description

Requests Copilot as a reviewer and posts a short scoping comment naming the grimoire gate. Copilot then applies the instruction files synced by `copilot_sync_repo()`. Returns a status message suitable for replying to the requester.

Usage

```
copilot_review_pr(
  owner,
  repo,
  pr_number,
  gate = NULL,
  comment = TRUE,
  config = load_copilot_review_config()
)
```

Arguments

owner	Repository owner.
repo	Repository name.
pr_number	Pull request number.
gate	Gate key (brand, blog, social, translation), or NULL for a full review.
comment	Whether to post the scoping comment. Defaults to TRUE.
config	Config from load_copilot_review_config() .

Value

A status string.

copilot_sync_repo	<i>Sync grimoire review skills into a repo as Copilot instructions</i>
-------------------	--

Description

Fetches the four grimoire review-gate skills (plus the foundation brand ruleset) and lands them in owner/repo as .github/copilot-instructions.md and path-scoped .github/instructions/*.instructions.md files, via a pull request. This is what makes GitHub Copilot's code review grimoire-aware. Re-run whenever grimoire changes to refresh the instructions.

Usage

```
copilot_sync_repo(
  owner = "rladies",
  repo,
  base = "main",
  branch = "jinx/copilot-review-instructions",
  config = load_copilot_review_config()
)
```

Arguments

owner	Target repository owner. Defaults to "rladies".
repo	Target repository name.
base	Base branch. Defaults to "main".
branch	Working branch. Defaults to "jinx/copilot-review-instructions".
config	Config from <code>load_copilot_review_config()</code> .

Value

A list with status ("created" or "unchanged") and url (the PR URL when created).

directory_crop_image *Crop and resize a directory profile image*

Description

Uses magick to crop to a square and resize for web display.

Usage

```
directory_crop_image(
  path,
  width = 400,
  height = 400,
  gravity = "Center",
  output = path
)
```

Arguments

path	Path to the image file.
width	Target width in pixels. Defaults to 400.
height	Target height in pixels. Defaults to 400.
gravity	Crop gravity. Defaults to "Center".
output	Output path. Defaults to overwriting the input.

Value

Output path (invisibly).

directory_mark_synced *Flag handled submissions as synced.*

Description

Reconciles Airtable against the directory and flips the `directory_synced_field()` checkbox on every submission whose work is done, so future syncs skip it:

Usage

```
directory_mark_synced(  
  base_id = directory_base_id(),  
  api_key = Sys.getenv("AIRTABLE_API_KEY"),  
  org = "rladies",  
  directory_repo = "directory"  
)
```

Arguments

<code>base_id</code>	Airtable base ID. Defaults to the directory base.
<code>api_key</code>	Airtable API key. Defaults to AIRTABLE_API_KEY env var.
<code>org</code>	GitHub organization. Defaults to "rladies".
<code>directory_repo</code>	Directory repository name.

Details

- an **update** is done once applying it to main would change nothing — its entry is already fully incorporated (the same predicate the sync uses to decide whether to commit);
- a **delete request** is done once its entry file is absent from main — the reviewed purge workflow has removed it.

Stateless and idempotent — safe to run on every sync-PR merge and after a purge completes.

Value

Character vector of marked record ids (invisibly).

directory_optimize_image

Optimize an image for web display

Description

Resizes to max width and compresses.

Usage

```
directory_optimize_image(path, max_width = 800, quality = 85, output = path)
```

Arguments

path	Path to the image file.
max_width	Maximum width in pixels. Defaults to 800.
quality	JPEG quality (1-100). Defaults to 85.
output	Output path. Defaults to overwriting the input.

Value

Output path (invisibly).

directory_purge_submissions

Erase every Airtable submission for the given directory slugs.

Description

GDPR right-to-erasure: after the purge workflow removes a member's entry from the directory, every submissions row that resolves to one of slugs (by `directory_id`, falling back to `identifier`) is **deleted** from Airtable so no submitted PII remains — and so a later sync cannot re-create the entry from a leftover row. Unlike `directory_mark_synced()`, which only flags the synced checkbox, this destroys the rows.

Usage

```
directory_purge_submissions(
  slugs,
  base_id = directory_base_id(),
  api_key = Sys.getenv("AIRTABLE_API_KEY")
)
```

Arguments

slugs	Character vector of directory slugs to erase.
base_id	Airtable base ID. Defaults to the directory base.
api_key	Airtable API key. Defaults to the AIRTABLE_API_KEY env var.

Value

Character vector of deleted record ids (invisibly).

directory_sync_airtable

Sync directory entries from Airtable

Description

Fetches directory submissions from Airtable, transforms each eligible record into the directory entry schema, and opens (or updates) a PR on the directory repo containing only the entries whose content changed.

Usage

```
directory_sync_airtable(
  base_id = directory_base_id(),
  api_key = Sys.getenv("AIRTABLE_API_KEY"),
  org = "rladies",
  directory_repo = "directory"
)
```

Arguments

base_id	Airtable base ID. Defaults to the directory base.
api_key	Airtable API key. Defaults to AIRTABLE_API_KEY env var.
org	GitHub organization. Defaults to "rladies".
directory_repo	Directory repository name.

Details

Only submissions marked `minority_gender == "yes"` and not flagged as a DEFAULT ROW are processed. Returning submitters are matched to their existing entry by slug (`directory_id`, falling back to `identifier`), and their submission is merged onto the existing file: fields listed in `clear_fields` are wiped first, then submitted fields overlay the rest, so a partial update never drops data the submitter left blank.

Delete requests are collected and reported in the PR body but are **not** executed here; destructive removal stays with the reviewed purge workflow.

Value

PR URL if changes found, NULL otherwise (invisibly).

directory_validate_filename
<i>Validate a directory entry filename</i>

Description

Checks that filenames follow conventions: lowercase, no hashes, ASCII-only, .json extension.

Usage

directory_validate_filename(filename)

Arguments

filename Filename (not full path) to validate.

Value

A named list with valid (logical) and issues (character vector of problems found).

directory_verify_handles
<i>Verify that social media handles exist</i>

Description

Performs HTTP HEAD requests to check if profiles resolve.

Usage

directory_verify_handles(entry)

Arguments

entry Named list representing a directory entry, with a social_media sub-list containing handles/URLs.

Value

A data frame with columns platform, handle, status, and valid.

event_authorize	<i>Authorize a parsed event before execution</i>
-----------------	--

Description

Events are gated on the originating team being an allowed workspace, not on actor privilege (there is no "actor" for a passive event like a reaction or a team join - anyone triggers these). `airtable_webhook` events have no `team_id` at all (Airtable webhooks aren't Slack-team scoped) and are trusted by construction: the Worker already verified the webhook's shared secret before dispatching. The other three kinds are checked against the organiser/community team allowlist, mirroring the Worker's own `slack_team_is_allowed()`.

Usage

```
event_authorize(
  event,
  organiser_id = Sys.getenv("SLACK_ORGANIZER_TEAM_ID"),
  community_id = Sys.getenv("SLACK_COMMUNITY_TEAM_ID")
)
```

Arguments

<code>event</code>	Parsed event list from <code>event_parse()</code> , or NULL.
<code>organiser_id</code>	Organiser workspace team id. Defaults to env <code>SLACK_ORGANIZER_TEAM_ID</code> .
<code>community_id</code>	Community workspace team id. Defaults to env <code>SLACK_COMMUNITY_TEAM_ID</code> .

Value

A list with `ok` (logical) and `message` (a refusal string when `ok` is FALSE, otherwise NULL).

event_create_summary	<i>Create a formatted event summary</i>
----------------------	---

Description

Create a formatted event summary

Usage

```
event_create_summary(events, period = c("weekly", "monthly"))
```

Arguments

<code>events</code>	Data frame from <code>event_list_chapter()</code> or <code>event_sync_chapters()</code> .
<code>period</code>	Summary period label.

Value

Formatted markdown string.

event_execute	<i>Execute a parsed, authorized event</i>
---------------	---

Description

Unlike `cmd_execute()`, handlers perform their own Slack/Airtable API calls directly rather than returning a reply string for a workflow step to relay - there is no single "response destination" for a passive event the way there is for a command. The return value is a short status string for the GitHub Actions run log only.

Usage

```
event_execute(event)
```

Arguments

event	Parsed event list from <code>event_parse()</code> .
-------	---

Value

Character string describing the outcome, for logging.

event_list_chapter	<i>List events for a chapter</i>
--------------------	----------------------------------

Description

Queries Meetup Pro for recent events.

Usage

```
event_list_chapter(chapter, months = 3)
```

Arguments

chapter	Chapter identifier (e.g. "rladies-berlin").
months	Number of months of history.

Value

Data frame with columns: title, date, url, rsvp_count, source, chapter.

event_parse	<i>Parse a slack-event dispatch payload</i>
-------------	---

Description

The Worker's repository_dispatch payload is already structured JSON (unlike slash commands, which arrive as free text for `cmd_parse()` to split), so this is a thin validator rather than a parser: it confirms kind is one of the registered `jinx_events()` entries and normalizes the shape callers can rely on.

Usage

```
event_parse(payload)
```

Arguments

payload	A list with kind, team_id, response_url, event.
---------	---

Value

A list with kind, team_id, response_url, event; kind is "unknown" when the payload's kind isn't registered.

event_sync_chapters	<i>Sync events across all chapters</i>
---------------------	--

Description

Fetches events from all configured sources and writes a summary.

Usage

```
event_sync_chapters(
    org = "rladies",
    target_repo = "event-archive",
    months = 3,
    dry_run = TRUE,
    chapters = NULL
)
```

Arguments

org	GitHub organization.
target_repo	Repository for the event archive.
months	Number of months of history.
dry_run	If TRUE, print what would be synced without acting.
chapters	Meetup urlnames to sync. Read from the website data when NULL.

Details

The chapter list comes from the website's data/chapters entries via `chapter_meetup_groups()`, not from configuration: that data is already the source of truth for which chapters exist and which Meetup group each uses, and a hand-maintained list would drift immediately.

Value

Data frame of all events (invisibly).

gather_rag_source	<i>Dispatch a source spec to the appropriate gather function</i>
-------------------	--

Description

Looks up `gather_<type>()` where `<type>` has dashes replaced by underscores. Sources extend the indexer by defining a new `gather_<type>(src)` function.

Usage

```
gather_rag_source(src)
```

Arguments

src Source spec list (must have type).

Value

List of chunk records (with `chunk_idx` set per document).

gh_branch_upsert	<i>Create or reset a branch to match a base ref</i>
------------------	---

Description

Reads the base branch tip and either creates branch there or force-updates it. Used by helpers that need to (re-)open a clean working branch before pushing files.

Usage

```
gh_branch_upsert(org, repo, branch, base = "main", force = TRUE)
```

Arguments

org	Repository owner.
repo	Repository name.
branch	Branch to create or update.
base	Base branch whose tip is used as the new SHA. Defaults to "main".
force	Whether to force-update if the branch already exists. Defaults to TRUE.

Value

The SHA the branch now points to (invisibly).

gh_greet_contributor *Greet a new PR author*

Description

Thin wrapper around `gh_welcome_contributor()` that fixes `is_pr = TRUE` for use in PR-only workflows. Reusable across any repo.

Usage

```
gh_greet_contributor(
  owner,
  repo,
  number,
  author,
  org = "rladies",
  extra_message = NULL
)
```

Arguments

owner	Repository owner.
repo	Repository name.
number	PR number.
author	GitHub login of the author.
org	Organization name.
extra_message	Optional extra markdown to append after the standard welcome message.

Value

Comment URL or NULL if author is a team member (invisibly).

gh_open_or_update_pr	Open a PR, or return the existing open PR for a branch
----------------------	--

Description

If a PR is already open from branch into base, returns its URL; otherwise opens a new PR with the given title and body.

Usage

```
gh_open_or_update_pr(  
  org,  
  repo,  
  branch,  
  base = "main",  
  title,  
  body,  
  team_reviewers = NULL  
)
```

Arguments

- org Repository owner.
- repo Repository name.
- branch Head branch.
- base Base branch. Defaults to "main".
- title PR title.
- body PR body.
- team_reviewers Character vector of org team slugs to request review from, or NULL for none.

Value

PR URL.

gh_post_checklist	Post a content review checklist on a PR
-------------------	---

Description

Posts a review checklist comment tailored for content PRs (blog posts, news posts, or any markdown-driven content). Generic across content kinds — the only difference between blog and news today is the path filter the caller workflow applies.

Usage

```
gh_post_checklist(owner, repo, pr_number)
```

Arguments

owner	Repository owner.
repo	Repository name.
pr_number	PR number.

Value

Comment URL (invisibly).

gh_thank_contributor *Thank everyone who contributed to a merged PR*

Description

Posts a thank-you message tailored to whether this is the author's first merged PR, and credits everyone else who helped: co-authors (commit authors and Co-authored-by: trailers), reviewers, and people who commented on the PR.

Usage

```
gh_thank_contributor(owner, repo, pr_number, author, org = "rladies")
```

Arguments

owner	Repository owner.
repo	Repository name.
pr_number	PR number.
author	GitHub login of the PR author.
org	Organization name.

Value

Comment URL (invisibly).

`gh_thank_issue_contributors`*Thank everyone who took part in a closed issue*

Description

Posts a thank-you when an issue is closed, crediting the person who raised it (with a warmer message if it was their first here) plus everyone who was assigned to it or commented on it.

Usage

```
gh_thank_issue_contributors(owner, repo, number, author, completed = TRUE)
```

Arguments

<code>owner</code>	Repository owner.
<code>repo</code>	Repository name.
<code>number</code>	Issue number.
<code>author</code>	GitHub login of the person who opened the issue.
<code>completed</code>	Whether the issue was closed as completed. An issue closed as <code>not_planned</code> (spam, duplicate, invalid) is still thanked, but without the first-timer congratulation.

Value

Comment URL (invisibly), or NULL for bot authors.

`gh_welcome_contributor`*Welcome a contributor on a new PR or issue*

Description

Checks if the PR/issue author has previous contributions to the repo. If this is their first, posts a warm welcome message. If not, posts a shorter thank-you. No-op for bots and global team members.

Usage

```
gh_welcome_contributor(  
    owner,  
    repo,  
    number,  
    author,  
    is_pr = TRUE,  
    org = "rladies",  
    extra_message = NULL  
)
```

Arguments

owner	Repository owner.
repo	Repository name.
number	Issue or PR number.
author	GitHub login of the author.
is_pr	Whether this is a PR (TRUE) or issue (FALSE).
org	Organization name.
extra_message	Optional extra markdown to append after the standard welcome message (e.g. a project-specific reminder). Ignored when blank or NULL. The jinx signature is preserved as the final paragraph.

Value

Comment URL or NULL if author is a team member (invisibly).

gha_generate_dashboard

Generate GitHub Actions dashboard data

Description

Scans all non-meetup repositories in the organization, collects workflow information, and generates a JSON report.

Usage

```
gha_generate_dashboard(  
    org = "rladies",  
    exclude_pattern = "^meetup-",  
    output_path = NULL  
)
```

Arguments

org GitHub organization. Defaults to "rladies".
 exclude_pattern Regex pattern to exclude repos. Defaults to "^meetup-".
 output_path Path to write the JSON report. If NULL, returns the data without writing.

Value

List of workflow data (invisibly).

gt_check_invitations *Check pending global team invitations*

Description

Lists pending invitations and triggers finalization for accepted ones.

Usage

```
gt_check_invitations(org = "rladies")
```

Arguments

org GitHub organization. Defaults to "rladies".

Value

Character vector of usernames that have accepted their invitation (invisibly).

gt_create_offboarding *Create a global team offboarding issue*

Description

Create a global team offboarding issue

Usage

```
gt_create_offboarding(username, team, name = username, org = "rladies")
```

Arguments

username GitHub username.
 team Team slug.
 name Full name. Defaults to the username.
 org GitHub organization. Defaults to "rladies".

Value

The created issue URL (invisibly).

`gt_finalize_offboarding`

Finalize global team offboarding by removing user from teams

Description

Finalize global team offboarding by removing user from teams

Usage

```
gt_finalize_offboarding(username, team, org = "rladies")
```

Arguments

<code>username</code>	GitHub username.
<code>team</code>	Team slug.
<code>org</code>	GitHub organization. Defaults to "rladies".

Value

Invisibly returns NULL. Called for its side effect of removing the user from the specified teams.

`gt_finalize_onboarding`

Finalize global team onboarding for an accepted member

Description

Adds the user to their specific team and creates a tracking issue with the onboarding checklist.

Usage

```
gt_finalize_onboarding(username, team, name = username, org = "rladies")
```

Arguments

<code>username</code>	GitHub username.
<code>team</code>	Team slug (e.g. "website").
<code>name</code>	Full name.
<code>org</code>	GitHub organization. Defaults to "rladies".

Value

The created issue URL (invisibly).

gt_invite	<i>Invite a user to the RLadies+ global team</i>
-----------	--

Description

Sends an organization membership invitation and adds the user to the global team.

Usage

```
gt_invite(username, team, name = username, org = "rladies")
```

Arguments

username	GitHub username (without @).
team	Team slug (e.g. "website").
name	Full name (used in issue templates).
org	GitHub organization. Defaults to "rladies".

Value

Invisibly returns NULL. Called for its side effect of sending the invitation and adding the user to the specified team.

gt_remind_stale	<i>Send reminders on stale global team onboarding/offboarding issues</i>
-----------------	--

Description

Finds open issues with onboarding or offboarding labels that haven't been updated in days days, and posts a reminder comment.

Usage

```
gt_remind_stale(org = "rladies", days = 30, repo = "global-team")
```

Arguments

org	GitHub organization. Defaults to "rladies".
days	Number of days without activity before reminding. Defaults to 30.
repo	Repository to check. Defaults to "global-team".

Value

Invisibly returns NULL. Called for its side effect of posting reminder comments on stale issues.

gt_schedule_onboarding_meeting

Open the onboarding meeting poll and post it to the onboarding issue

Description

Part of global team onboarding: opens a "find a time" poll on samkoma so the new member and their team can settle on an onboarding meeting slot. The poll offers a two-week window of candidate dates starting one week after it is created, and the poll link is posted as a comment on the onboarding issue. The poll is created (and the comment posted) under Jinx's identity.

Usage

```
gt_schedule_onboarding_meeting(
    issue_number,
    name,
    team_name,
    org = "rladies",
    repo = "global-team",
    start = Sys.Date() + 7,
    from = "08:00",
    to = "20:00",
    slot = 30,
    tz = "UTC"
)
```

Arguments

issue_number	Onboarding issue number the poll link is posted to.
name	Full name of the new member (used in the poll title).
team_name	Human-readable team name (used in the poll title).
org	GitHub organization. Defaults to "rladies".
repo	Repository holding the onboarding issue. Defaults to "global-team".
start	First candidate date. Defaults to one week from today.
from, to	Daily availability window in 24h "HH:MM". Defaults to a broad UTC band that overlaps working hours across regions.
slot	Slot length in minutes. Defaults to 30.
tz	IANA timezone the poll window is painted in. Defaults to "UTC" since the global team spans timezones.

Value

The poll URL (invisibly).

guide_email_template *Fetch a canonical email template from the guide*

Description

The guide writes the templates that are sent as email with their subject on a leading SUBJECT: line, followed by BODY OF THE MESSAGE:. This splits the two apart so that callers send the guide's subject rather than inventing one, and errors if the template carries no subject at all.

Usage

```
guide_email_template(name, variables = list(), base_url = guide_base_url())
```

Arguments

name	Template name without extension, as it is filed in the guide (for example "meetup-group-description")
variables	Named list of <<KEY>> placeholder values. Keys are given without the angle brackets. Placeholders the guide leaves for a human to fill are passed through untouched.
base_url	Guide base URL, for testing against a local build.

Value

A list with name, subject and body.

See Also

[guide_template\(\)](#) for a template with no subject line.

Examples

```
## Not run:
notice <- guide_email_template("chapter-inactive-first-notice")
notice$subject

## End(Not run)
```

guide_template

Fetch a canonical communications template from the guide

Description

Organisational communications - what RLadies+ says to organisers and chapters - are written and edited in the guide, under `static/templates/`, and served as markdown from `https://guide.rladies.org/templates/<jinx fetches them instead of shipping copies, so a volunteer can fix the wording without waiting for a package release, and the text an organiser reads in the guide is the text jinx sends.`

Usage

```
guide_template(name, variables = list(), base_url = guide_base_url())
```

Arguments

name	Template name without extension, as it is filed in the guide (for example "meetup-group-description").
variables	Named list of <<KEY>> placeholder values. Keys are given without the angle brackets. Placeholders the guide leaves for a human to fill are passed through untouched.
base_url	Guide base URL, for testing against a local build.

Details

A failed fetch is an error. There is deliberately no vendored fallback, because a second copy is the drift this replaces.

Use `guide_email_template()` for a template that is sent as email and so carries its own subject line.

Value

The template text, as a single string.

Examples

```
## Not run:
guide_template("meetup-group-description")

guide_template(
  "chapter-onboarding-welcome",
  list(FIRST_NAME = "Ada", CITY = "Oslo")
)

## End(Not run)
```

i18n_check_coverage	<i>Check translation coverage across languages</i>
---------------------	--

Description

Check translation coverage across languages

Usage

i18n_check_coverage()

Value

Data frame with columns: language, total_templates, translated, coverage_pct.

i18n_get_chapter_language	<i>Get a chapter's preferred language</i>
---------------------------	---

Description

Looks up the chapter's language preference from its metadata. Falls back to English if not set.

Usage

i18n_get_chapter_language(chapter, org = "rladies")

Arguments

chapter	Chapter slug (e.g. "rladies-berlin").
org	GitHub organization.

Value

Language code string.

i18n_list_languages	<i>List supported languages</i>
---------------------	---------------------------------

Description

List supported languages

Usage

i18n_list_languages()

Value

Data frame with columns: code, name, native_name, direction.

i18n_translate_template	<i>Translate a template with language fallback</i>
-------------------------	--

Description

Looks up inst/translations/{language}/{template_name}.md first, falls back to inst/templates/{template_name} if the translation is missing.

Usage

i18n_translate_template(template_name, language = "en", variables = list())

Arguments

- template_name Template filename without path (e.g. "global-team-onboarding.md").
- language Language code (e.g. "es", "pt", "fr"). Defaults to "en".
- variables Named list of placeholder values for render_template().

Value

Rendered template as a single character string.

i18n_validate_translations	<i>Validate translations for placeholder consistency</i>
----------------------------	--

Description

Checks that all translated templates have the same <KEY> placeholders as the English baseline.

Usage

```
i18n_validate_translations(language = NULL)
```

Arguments

language Language code to validate. If NULL, validates all.

Value

Data frame with columns: template, language, status, missing_keys, extra_keys.

li_get_version	<i>Get LinkedIn API version string</i>
----------------	--

Description

Get LinkedIn API version string

Usage

```
li_get_version()
```

Value

Version string in YYYYMM format.

li_media_upload	<i>Upload media to LinkedIn</i>
-----------------	---------------------------------

Description

Upload media to LinkedIn

Usage

```
li_media_upload(author, media)
```

Arguments

author	LinkedIn URN of the owner.
media	File path to the image.

Value

LinkedIn image ID string.

li_oauth	<i>Perform LinkedIn OAuth authentication</i>
----------	--

Description

Interactive OAuth2 flow for obtaining a LinkedIn token.

Usage

```
li_oauth()
```

Value

OAuth token object.

li_post_write	<i>Post to LinkedIn</i>
---------------	-------------------------

Description

Post to LinkedIn

Usage

```
li_post_write(author, text, image = NULL, image_alt = "")
```

Arguments

author	LinkedIn URN of the author.
text	Post text.
image	Optional file path to an image.
image_alt	Alt text for the image.

Value

LinkedIn post ID (invisibly).

li_req	<i>Create a base LinkedIn API request</i>
--------	---

Description

Create a base LinkedIn API request

Usage

```
li_req(endpoint_version = "rest", ...)
```

Arguments

endpoint_version	API version path. Defaults to "rest".
...	Passed to li_req_auth().

Value

Configured httr2 request.

li_urn_me	<i>Get the LinkedIn URN for the authenticated user</i>
-----------	--

Description

Get the LinkedIn URN for the authenticated user

Usage

li_urn_me()

Value

URN string like "urn:li:person:abc123".

load_copilot_review_config	<i>Load the Copilot review bridge configuration</i>
----------------------------	---

Description

Locates the grimoire source skills and the per-repository file globs each review gate watches. See inst/config/copilot-review.yml.

Usage

load_copilot_review_config()

Value

Named list with grimoire and repos entries.

load_labels_config	<i>Load file-path to label mappings</i>
--------------------	---

Description

Load file-path to label mappings

Usage

load_labels_config()

Value

Named list with label mappings.

load_rag_sources	<i>Load the configured RAG source list</i>
------------------	--

Description

Load the configured RAG source list

Usage

load_rag_sources()

Value

List of source specs.

load_review_rules	<i>Load PR review rules</i>
-------------------	-----------------------------

Description

Load PR review rules

Usage

load_review_rules()

Value

Named list with defaults, rules, and naming conventions.

load_teams_config	<i>Load teams configuration</i>
-------------------	---------------------------------

Description

Load teams configuration

Usage

load_teams_config()

Value

Named list with organization, global_team_id, teams, and default_assignees.

mail_send	<i>Send an email as jinx</i>
-----------	------------------------------

Description

Sends plain text mail from jinx@rladies.org, cc'ing chapters@rladies.org by default so the onboarding inbox keeps the thread. jinx never sends *as* chapters@, which is why this needs only a single account's gmail.send token rather than domain-wide delegation.

Usage

```
mail_send(  
  to,  
  subject,  
  body,  
  cc = "chapters@rladies.org",  
  from = "jinx@rladies.org"  
)
```

Arguments

to	Recipient address, or a vector of them.
subject	Message subject.
body	Plain text body.
cc	Cc address. Defaults to "chapters@rladies.org"; pass NULL for none.
from	Sender address. Defaults to "jinx@rladies.org".

Value

The sent message id (invisibly).

meeting_poll_best	<i>Get the ranked best slots for a poll</i>
-------------------	---

Description

Get the ranked best slots for a poll

Usage

```
meeting_poll_best(id, edit_token = NULL, base_url = samkoma_base_url())
```

Arguments

id	Poll id.
edit_token	Optional host edit token, required to read a hidden poll's responses.
base_url	API base URL.

Value

A data frame with columns slot, count, and names (comma-separated), ordered best-first. The total number of respondents is attached as the total attribute.

meeting_poll_create	Create a meeting-scheduling poll on samkoma
---------------------	---

Description

Creates a "find a time" poll where participants paint their availability over a set of days and time slots. Polls are public by default so results can be read back without a stored secret; the returned edit_token is the host secret needed to lock the final slot or read a hidden poll.

Usage

```
meeting_poll_create(  
  title,  
  days,  
  from,  
  to,  
  slot,  
  tz = "UTC",  
  kind = c("dates", "weekdays"),  
  public = TRUE,  
  results_hidden = FALSE,  
  deadline = NULL,  
  base_url = samkoma_base_url()  
)
```

Arguments

title	Poll title (1-200 characters).
days	Character vector of day identifiers: ISO dates (YYYY-MM-DD) when kind = "dates", or weekday names (mon..sun) when kind = "weekdays". 1-60 items.
from, to	Start and end of the daily window, "HH:MM" (24h).
slot	Slot length in minutes.
tz	IANA timezone name (e.g. "Europe/Oslo"). Defaults to UTC.
kind	Either "dates" (default) or "weekdays".

public	Whether the poll is publicly readable. Defaults to TRUE.
results_hidden	Whether responses are hidden from voters.
deadline	Optional ISO 8601 datetime after which voting closes.
base_url	API base URL.

Value

A list with id, url, and edit_token.

meeting_poll_format_best	<i>Format ranked best slots as markdown</i>
--------------------------	---

Description

Format ranked best slots as markdown

Usage

```
meeting_poll_format_best(best, title = NULL, top = 3)
```

Arguments

best	A data frame from <code>meeting_poll_best()</code> .
title	Optional poll title for the heading.
top	Maximum number of slots to show.

Value

Character string of markdown.

meeting_poll_format_created	<i>Format a created poll as a markdown announcement</i>
-----------------------------	---

Description

Deliberately omits the edit token, which is a host secret.

Usage

```
meeting_poll_format_created(created, title)
```

Arguments

- created A list returned by `meeting_poll_create()`.
- title The poll title.

Value

Character string of markdown.

meeting_poll_get	<i>Fetch a poll and its aggregated responses</i>
------------------	--

Description

Fetch a poll and its aggregated responses

Usage

```
meeting_poll_get(id, edit_token = NULL, base_url = samkoma_base_url())
```

Arguments

- id Poll id.
- edit_token Optional host edit token, required to read a hidden poll's responses.
- base_url API base URL.

Value

Parsed poll object as a list.

meeting_poll_ics	<i>Export the locked slot of a poll as an iCalendar (.ics) string</i>
------------------	---

Description

Export the locked slot of a poll as an iCalendar (.ics) string

Usage

```
meeting_poll_ics(id, base_url = samkoma_base_url())
```

Arguments

- id Poll id.
- base_url API base URL.

Value

The .ics file contents as a character string.

meeting_poll_lock	<i>Lock in (or clear) the chosen slot for a poll</i>
-------------------	--

Description

Host-only: requires the poll's edit_token.

Usage

```
meeting_poll_lock(id, slot, edit_token, base_url = samkoma_base_url())
```

Arguments

id	Poll id.
slot	The chosen slot identifier (YYYY-MM-DDTHH:MM or [mon-sun]THH:MM), or NULL to clear a previously locked slot.
edit_token	Optional host edit token, required to read a hidden poll's responses.
base_url	API base URL.

Value

Invisibly, the parsed response.

meetup_urlname_status	<i>Whether a Meetup urlname looks taken</i>
-----------------------	---

Description

Advisory only. Meetup serves its single-page app with HTTP 200 for groups that do not exist, so the status code says nothing; the check looks for the "Group not found" copy in the body instead. That copy can change, so this reports "unknown" rather than guessing when it cannot tell, and a human should confirm before creating the group.

Usage

```
meetup_urlname_status(urlname)
```

Arguments

urlname	Meetup urlname to check.
---------	--------------------------

Value

One of "available", "taken", or "unknown".

pending_link_consume *Consume a pending chapter sign-up link for a newly joined email*

Description

Reads and deletes the KV pending_link:{email} record (same SLACK_TOKENS namespace and key format the Airtable invite flow writes to when marking an invite sent). R port of slack_pending_link_consume() from worker/src/slack-events.js.

Usage

```
pending_link_consume(
  email,
  namespace_id = slack_tokens_namespace_id(),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")
)
```

Arguments

email	Email address to look up.
namespace_id	KV namespace ID for SLACK_TOKENS.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.

Value

The parsed link record, or NULL if none was pending.

prepare_inactivity_emails *Prepare the guide's first inactivity notice for inactive chapters*

Description

Identifies inactive chapters and prepares the notice the guide documents for them. The wording and the subject line come from the guide, fetched at call time, so the message organisers receive is the one a volunteer can edit there.

Usage

```
prepare_inactivity_emails(
  chapters,
  template = "chapter-inactive-first-notice",
  dry_run = TRUE
)
```

Arguments

chapters	Chapter status data from <code>chapter_monitor_status()</code> .
template	Name of the guide template to send. Defaults to the first inactivity notice; the guide also documents a retirement notice ("chapter-retirement-scheduled") for chapters that do not reply.
dry_run	Controls the wording of the summary only. Either way the prepared emails are returned rather than sent: delivery goes through <code>mail_send()</code> , which a human still drives.

Value

Data frame of prepared emails, one row per inactive chapter, with chapter, email, subject and body (invisibly).

promo_blurb	<i>Draft a spotlight invitation for a channel in Jinx's voice</i>
-------------	---

Description

Calls Workers AI to write a short, warm invitation from the channel's name and description. Failures are swallowed and return NULL so the caller can fall back to a plain-description blurb rather than skip the promotion.

Usage

```
promo_blurb(  
  name,  
  description,  
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),  
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),  
  model = workers_ai_chat_model()  
)
```

Arguments

name	Channel name (without #).
description	Cleaned channel description.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
model	Workers AI chat model.

Value

Character scalar invitation, or NULL if the model call failed or returned nothing usable.

```
promo_eligible_channels
```

Select the community public channels eligible for promotion

Description

Keeps non-archived public channels that carry a description (Slack "purpose"), dropping the channel we post the spotlight to and any names in skip. The description is what the weekly spotlight is built from, so a channel with an empty purpose has nothing to promote and is left out.

Usage

```
promo_eligible_channels(channels, target_channel = NULL, skip = character())
```

Arguments

channels	List of Slack channel objects, as returned by <code>slack_conversations_list()</code> .
target_channel	Name of the channel the spotlight is posted to, excluded so it never promotes itself.
skip	Character vector of additional channel names to exclude.

Value

A data frame with one row per eligible channel and columns id, name, and description (cleaned of link markup and URLs).

```
promo_pick_channel
```

Pick the next channel to spotlight, round-robin

Description

Chooses the alphabetically-first eligible channel that has not been featured since the cycle began, so every channel gets a turn before any repeats. recent is pruned to currently-eligible ids first (channels that were archived or lost their description drop out), and the cycle resets once every eligible channel has been seen.

Usage

```
promo_pick_channel(eligible, recent = character())
```

Arguments

eligible	Data frame from <code>promo_eligible_channels()</code> .
recent	Character vector of channel ids featured earlier this cycle.

Value

NULL when nothing is eligible, otherwise a list with channel (the chosen single-row data frame) and recent (the updated vector to persist).

```
question_coding_declined_count
```

Count coding questions Jinx declined

Description

Count coding questions Jinx declined

Usage

```
question_coding_declined_count(rows)
```

Arguments

rows Data frame from [question_log_query\(\)](#).

Value

Integer count of outcome == "coding_declined" rows.

```
question_content_gaps
```

Rank content gaps for the weekly digest

Description

Narrower sibling of [question_gaps_rank\(\)](#): excludes "coding_declined" rows, since declining a coding question is working as designed, not a corpus gap. R port of content_gaps() from the deleted worker/src/question-digest.js.

Usage

```
question_content_gaps(rows, min_count = 1, limit = 10)
```

Arguments

rows Data frame from [question_log_query\(\)](#).
 min_count Minimum occurrence count to keep a gap.
 limit Maximum number of gaps to return.

Value

Data frame with columns question, outcome, count.

question_digest_build *Build the weekly question-gap digest text*

Description

Queries the anonymous question log, ranks content gaps and downvoted answers, drafts a proposed Guide answer for the top gaps, and formats the lot as Slack mrkdwn. R port of question_digest_build() from the deleted worker/src/question-digest.js.

Usage

```
question_digest_build(
  days = 7,
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  database_id = "4500d886-2593-44f9-9a01-d38cfa26e8dc",
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),
  model = workers_ai_chat_model()
)
```

Arguments

days	Number of days to cover. Defaults to 7.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
database_id	D1 database ID. Defaults to the provisioned jinx-question-log database.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
model	Workers AI chat model used for drafting.

Value

Character scalar digest text, or NULL when nothing was logged.

question_digest_format *Format the weekly question-gap digest as Slack mrkdwn*

Description

Distinct from [question_log_format\(\)](#), which renders a GitHub-comment markdown report for /jinx questions - this builds the Slack message text posted by [question_digest_post\(\)](#). R port of format_digest() from the deleted worker/src/question-digest.js.

Usage

```
question_digest_format(days, total, gaps, drafts, downvoted, coding_count)
```

Arguments

days	Number of days the report covers.
total	Total questions logged in the period.
gaps	Data frame from question_content_gaps() .
drafts	Data frame: the top-drafted subset of gaps, plus a draft character column (NA where drafting failed).
downvoted	Data frame from question_downvoted_rank() .
coding_count	Integer from question_coding_declined_count() .

Value

Character scalar Slack mrkdwn message.

question_digest_post *Build and post the weekly question-gap digest to Slack*

Description

Build and post the weekly question-gap digest to Slack

Usage

```
question_digest_post(
  days = 7,
  channel = env_default("SLACK_DIGEST_CHANNEL", "team-jinx"),
  slack_token = Sys.getenv("SLACK_TOKEN"),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  database_id = "4500d886-2593-44f9-9a01-d38cfa26e8dc",
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),
  model = workers_ai_chat_model()
)
```

Arguments

days	Number of days to cover. Defaults to 7.
channel	Slack channel to post to. Defaults to env SLACK_DIGEST_CHANNEL, falling back to "team-jinx".
slack_token	Slack bot token. Defaults to env SLACK_TOKEN.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
database_id	D1 database ID. Defaults to the provisioned jinx-question-log database.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
model	Workers AI chat model used for drafting.

Value

Invisibly, TRUE if a digest was posted, FALSE if there was nothing to report.

question_downvoted_rank

Rank questions where downvotes exceed upvotes

Description

R port of question_downvoted_rank() in worker/src/question-log.js.

Usage

```
question_downvoted_rank(rows, limit = 10)
```

Arguments

rows	Data frame from question_log_query() .
limit	Maximum number of rows to return.

Value

Subset of rows where down > up, ordered by down - up descending.

question_draft_guide_snippet

Draft a proposed Guide answer for a content-gap question

Description

Calls Workers AI to draft a short, human-reviewable answer for a question Jinx could not answer well. Failures are swallowed - a digest with an undrafted gap is better than a failed scheduled run. R port of draft_guide_snippet() from the deleted worker/src/question-digest.js.

Usage

```
question_draft_guide_snippet(
  question,
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),
  model = workers_ai_chat_model()
)
```

Arguments

question	The question Jinx could not answer well.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
model	Workers AI chat model.

Value

Character scalar draft, or NULL if the model call failed or returned nothing usable.

question_feedback_format

Format a reaction-feedback summary as a reply

Description

Format a reaction-feedback summary as a reply

Usage

```
question_feedback_format(summary)
```

Arguments

summary A list from [question_feedback_summary\(\)](#).

Value

Character scalar reply text.

question_feedback_summary

Summarize reaction-feedback tallies for a Slack workspace

Description

Aggregates the reaction_log:{team_id}:{day}:{reaction} KV counters [reaction_log_increment\(\)](#) writes, across the last days days. Backs the "feedback" command. R port of slash_feedback() from the deleted worker/src/slash-local.js.

Usage

```
question_feedback_summary(  
  team_id,  
  days = 7,  
  namespace_id = slack_tokens_namespace_id(),  
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),  
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")  
)
```

Arguments

team_id	Slack team id. Required (aborts if missing) - a blank prefix would otherwise scan every key in the namespace across both workspaces instead of just this one's.
days	Number of days to look back.
namespace_id	KV namespace ID for SLACK_TOKENS.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.

Value

A list with days, entries (integer count of KV entries scanned), and totals (a named integer vector, reaction name to summed count, sorted descending).

question_gaps_rank	<i>Rank content-gap questions by normalized-duplicate count</i>
--------------------	---

Description

R port of question_gaps_rank() in worker/src/question-log.js. Near-identical questions (case/whitespace-insensitive) are folded together so a popular unanswered question rises to the top.

Usage

```
question_gaps_rank(rows, limit = 10)
```

Arguments

rows	Data frame from <code>question_log_query()</code> .
limit	Maximum number of gaps to return.

Value

Data frame with columns question, outcome, count, ordered by count descending.

question_log_format	<i>Format a question-log report as markdown</i>
---------------------	---

Description

Format a question-log report as markdown

Usage

```
question_log_format(rows, gaps, downvoted, days)
```

Arguments

rows	Data frame from question_log_query() .
gaps	Data frame from question_gaps_rank() .
downvoted	Data frame from question_downvoted_rank() .
days	Number of days the report covers, for the header.

Value

Character string with markdown-formatted report.

question_log_purge	<i>Purge question-log rows past the retention window</i>
--------------------	--

Description

Deletes rows from the Cloudflare D1 jinx-question-log database older than `retention_days`, honouring the 180-day retention promise in PRIVACY.md. Formerly `question_log_purge()` in `worker/src/question-log.js`, ported here so the retention guarantee has a single implementation.

Usage

```
question_log_purge(  
  retention_days = 180,  
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),  
  database_id = "4500d886-2593-44f9-9a01-d38cfa26e8dc",  
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")  
)
```

Arguments

retention_days	Number of days to retain rows for. Defaults to 180.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
database_id	D1 database ID. Defaults to the provisioned jinx-question-log database.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.

Value

Integer number of rows deleted.

question_log_query	<i>Query the anonymous question log</i>
--------------------	---

Description

Reads rows from the Cloudflare D1 jinx-question-log database via `cloudflarer::cf_d1_query()` - the same table worker/src/question-log.js writes to from inside the Cloudflare Worker when @Jinx answers a question. No Slack user id, channel, or thread timestamp is stored, so a row cannot be traced to who asked.

Usage

```
question_log_query(
  since_day = as.character(Sys.Date() - 7),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  database_id = "4500d886-2593-44f9-9a01-d38cfa26e8dc",
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")
)
```

Arguments

since_day	Character YYYY-MM-DD. Only rows on or after this day are returned. Defaults to 7 days ago.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
database_id	D1 database ID. Defaults to the provisioned jinx-question-log database.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.

Value

Data frame with columns id, day, question, outcome, top_score, sources, up, down.

question_vote_apply	<i>Apply a reaction vote to the question it answered</i>
---------------------	--

Description

R port of question_vote_apply() from worker/src/slack-events.js (deleted as part of the reaction-handling migration). Looks up the D1 row an answer message links to via the answer_link:{team_id}:{channel}:{ts} KV key, and increments its up/down count.

Usage

```
question_vote_apply(  
  team_id,  
  item,  
  reaction,  
  namespace_id = slack_tokens_namespace_id(),  
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),  
  database_id = "4500d886-2593-44f9-9a01-d38cfa26e8dc",  
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")  
)
```

Arguments

- team_id Slack team id.
- item Reaction event's item list: type, channel, ts.
- reaction Raw reaction name.
- namespace_id KV namespace ID for SLACK_TOKENS.
- account_id Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
- database_id D1 database ID. Defaults to the provisioned jinx-question-log database.
- api_token Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.

Value

TRUE if a vote was applied, FALSE otherwise.

r_challenge_adversarial_check	<i>Adversarially review a candidate challenge</i>
-------------------------------	---

Description

Second, independent Workers AI pass that is prompted to reject the challenge unless it is confident it is correct, unambiguous, and fairly rated. Fails closed: any error or unparseable verdict counts as rejection.

Usage

```
r_challenge_adversarial_check(
  challenge,
  rerex_output = character(),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),
  model = workers_ai_chat_model()
)
```

Arguments

challenge	Parsed challenge from <code>r_challenge_parse()</code> .
rerex_output	Captured verification output from <code>r_challenge_rerex_check()</code> .
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
model	Workers AI chat model.

Value

List with ok (logical), verdict, and issues (character vector).

r_challenge_allowed_packages

Packages a generated challenge may rely on

Description

Constrains what the generator may use and what the container running the rerex verification must provide: base R plus a small tidyverse core.

Usage

```
r_challenge_allowed_packages()
```

Value

Character vector of package names.

`r_challenge_difficulties`*Difficulty levels for the weekly R challenge*

Description

Difficulty levels for the weekly R challenge

Usage

```
r_challenge_difficulties()
```

Value

Data frame with columns `level`, `label`, and `badge` (a Slack emoji), one row per difficulty.

`r_challenge_difficulty_badge`*Slack emoji badge for a difficulty level*

Description

Slack emoji badge for a difficulty level

Usage

```
r_challenge_difficulty_badge(difficulty)
```

Arguments

`difficulty` One of the levels in `r_challenge_difficulties()`.

Value

Character scalar emoji.

r_challenge_difficulty_label

Human-readable label for a difficulty level

Description

Human-readable label for a difficulty level

Usage

```
r_challenge_difficulty_label(difficulty)
```

Arguments

difficulty One of the levels in `r_challenge_difficulties()`.

Value

Character scalar label.

r_challenge_draft_build

Draft a fully verified weekly R challenge

Description

Orchestrates the full generator: draft with Workers AI, parse, verify the reference solution with replex, render the statement as a replex, then adversarially review. Regenerates on any failed gate, up to max_tries, and returns NULL if none passes, so a scheduled run never emits unverified output.

Usage

```
r_challenge_draft_build(
  difficulty = NULL,
  max_tries = 3L,
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),
  model = workers_ai_chat_model()
)
```

Arguments

difficulty	Optional difficulty to request; NULL lets the model choose.
max_tries	Maximum generate-and-verify attempts before giving up.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
model	Workers AI chat model.

Value

A list with challenge, statement (rendered statement reпреx), reпреx (solution verification output), critique, and attempts, or NULL if no verified challenge was produced.

r_challenge_draft_post

Draft, verify, and propose a weekly R challenge

Description

Top-level entry for the draft cron: builds a fully verified challenge with `r_challenge_draft_build()` and, if one passes every gate, opens it as a status: proposed issue via `r_challenge_open_issue()` for an organiser to review. Does nothing (and does not error) when no challenge survives verification, so a scheduled run never proposes unverified output.

Usage

```
r_challenge_draft_post(
  difficulty = NULL,
  repo = "rladies/cauldron",
  max_tries = 3L,
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),
  model = workers_ai_chat_model()
)
```

Arguments

difficulty	Optional difficulty to request; NULL lets the model choose.
repo	Target review repo as "owner/name".
max_tries	Maximum generate-and-verify attempts before giving up.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
model	Workers AI chat model.

Value

Invisibly, TRUE if a challenge was proposed, else FALSE.

r_challenge_format_issue

Format a drafted challenge as a GitHub review issue

Description

Builds the Markdown body an organiser reviews before approving: the rendered statement in the open, with the reference solution, reprex verification output, and adversarial notes in collapsed sections.

Usage

```
r_challenge_format_issue(build, source = "ai")
```

Arguments

build	Result of <code>r_challenge_draft_build()</code> .
source	Where the challenge came from, e.g. "ai" or "community".

Value

Character scalar Markdown issue body.

r_challenge_format_slack

Format a challenge as a Slack message

Description

Renders the difficulty badge, title, and the challenge's reprex statement for posting to the community channel. The title flows through `escape_markdown()` and the statement sits in a fenced code block (its backticks stripped), so model- or user-generated text cannot inject links or `<!channel>` mass-pings. The reference solution is never shown.

Usage

```
r_challenge_format_slack(title, difficulty, statement)
```

Arguments

title	Challenge title.
difficulty	Difficulty level (see <code>r_challenge_difficulties()</code>).
statement	Rendered reprex statement (character vector or scalar).

Value

Character scalar Slack mrkdwn message.

r_challenge_generate *Ask Workers AI to draft a weekly R challenge*

Description

Low-level generator: returns the model's raw completion, which `r_challenge_parse()` turns into a structured challenge. Most callers want `r_challenge_draft_build()`, which also verifies the result.

Usage

```
r_challenge_generate(
  difficulty = NULL,
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),
  model = workers_ai_chat_model()
)
```

Arguments

difficulty	Optional difficulty to request; NULL lets the model choose.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
model	Workers AI chat model.

Value

Character scalar with the model's raw response.

r_challenge_issue_title
GitHub issue title for a drafted challenge

Description

GitHub issue title for a drafted challenge

Usage

```
r_challenge_issue_title(challenge)
```

Arguments

challenge	Parsed challenge from <code>r_challenge_parse()</code> .
-----------	--

Value

Character scalar issue title.

r_challenge_open_issue

Open a proposed challenge issue in the review repo

Description

Creates a status: proposed issue in the challenge repo (default rladies/cauldron) from a verified draft, labelled with its source and its themed difficulty, ready for an organiser to review. Requires a GITHUB_PAT/GITHUB_TOKEN with issues: write on the target repo.

Usage

```
r_challenge_open_issue(build, repo = "rladies/cauldron", source = "ai")
```

Arguments

build	Result of r_challenge_draft_build() .
repo	Target review repo as "owner/name".
source	Where the challenge came from, e.g. "ai".

Value

The created issue's URL, invisibly.

r_challenge_parse

Parse a model completion into a structured challenge

Description

Tolerates prose or code fences around the JSON. Returns NULL for any output missing a required field, carrying an unknown difficulty, or without at least one machine-checkable test, so callers can regenerate.

Usage

```
r_challenge_parse(text)
```

Arguments

text	Raw model completion from r_challenge_generate() .
------	--

Value

A list with elements title, difficulty, statement, solution, and tests (character vector), or NULL if invalid.

r_challenge_render_statement

Render a challenge statement as a reprex

Description

Runs the statement (task comment + setup + expected output as literals) through the reprex runner, producing the rendered block shown to solvers. Because it runs the statement on its own, a statement that leaks the answer by calling the not-yet-written function errors out and is rejected.

Usage

```
r_challenge_render_statement(statement, runner = r_challenge_reprex_runner)
```

Arguments

statement	Character scalar of R source stating the challenge.
runner	Function taking code lines and returning rendered reprex output. Defaults to the isolated reprex runner.

Value

List with ok (logical) and rendered (character vector).

r_challenge_reprex_check

Verify a reference solution against its own tests

Description

Assembles the solution plus a base::stopifnot() for each test, runs it through runner, and treats an #> Error line (a failed test or a broken solution) as verification failure.

Usage

```
r_challenge_reprex_check(solution, tests, runner = r_challenge_reprex_runner)
```

Arguments

solution	Character scalar reference solution.
tests	Character vector of R expressions expected to be TRUE.
runner	Function taking a character vector of code lines and returning the rendered output. Defaults to the isolated reprex runner.

Value

List with ok (logical) and output (character vector).

rag_chunk_id	<i>Stable vector ID for a chunk</i>
--------------	-------------------------------------

Description

Matches the JS indexer scheme: first 32 hex chars of sha256("{repo}|{path}|{chunk_idx}").

Usage

```
rag_chunk_id(repo, path, chunk_idx)
```

Arguments

- repo Source repo (e.g. "rladies/rladiesguide").
- path Path within the source (e.g. /getting-started/).
- chunk_idx Zero-based chunk index within the document.

Value

32-character hex string.

rag_index_build	<i>Build the Jinx RAG index</i>
-----------------	---------------------------------

Description

Gathers chunks from every configured source, embeds them with Cloudflare Workers AI, and upserts the vectors into a Vectorize index. The source list is loaded from inst/config/rag-sources.yml unless sources is supplied directly.

Usage

```
rag_index_build(  
  sources = NULL,  
  account_id = NULL,  
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN"),  
  index_name = NULL,  
  batch_size = 50L,  
  model = "@cf/baai/bge-base-en-v1.5"  
)
```

Arguments

sources	Optional list of source specs (see <code>load_rag_sources()</code>).
account_id	Cloudflare account ID. Falls back to env <code>CLOUDFLARE_ACCOUNT_ID</code> , then <code>cloudflare_account_id()</code> .
api_token	Cloudflare API token. Defaults to env <code>CLOUDFLARE_API_TOKEN</code> .
index_name	Vectorize index. Defaults to env <code>VECTORIZE_INDEX</code> or <code>"rladies-content"</code> .
batch_size	Number of chunks per embed/upsert call.
model	Workers AI embedding model.

Value

Invisibly: list with chunks (total) and upsert (API response).

random_emoji	<i>Select a random emoji</i>
--------------	------------------------------

Description

Select a random emoji

Usage

```
random_emoji()
```

Value

Single emoji character.

reaction_direction	<i>Classify a Slack reaction as an up- or down-vote</i>
--------------------	---

Description

R port of `reaction_direction()` from `worker/src/question-log.js`. Strips a skin-tone modifier suffix (e.g. `"thumbsup:skin-tone-3"`) before matching.

Usage

```
reaction_direction(reaction)
```

Arguments

reaction	Raw reaction name.
----------	--------------------

Value

"up", "down", or NULL for a neutral reaction.

reaction_event_apply *Apply an incoming Slack reaction event*

Description

The reaction_added event handler registered in jinx_events(): increments the reaction-feedback tally and applies a vote to the question the reacted-to message answered, if any.

Usage

```
reaction_event_apply(team_id, event)
```

Arguments

team_id	Slack team id.
event	The event's event payload: reaction, item.

Value

Invisibly, NULL.

reaction_log_increment
Increment the daily reaction-feedback counter for a workspace

Description

R port of the KV counter logic in slack_event_handle_reaction() (worker/src/slack-events.js, deleted as part of the reaction-handling migration). Backs /jinx feedback.

Usage

```
reaction_log_increment(
  team_id,
  reaction,
  namespace_id = slack_tokens_namespace_id(),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")
)
```

Arguments

team_id	Slack team id.
reaction	Raw reaction name.
namespace_id	KV namespace ID for SLACK_TOKENS.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.

Value

Invisibly, the new count for today.

```
report_format_markdown
```

Format a report as markdown

Description

Format a report as markdown

Usage

```
report_format_markdown(report)
```

Arguments

report Report data from `report_generate()`.

Value

Character string with markdown-formatted report.

```
report_generate
```

Generate an organization activity report

Description

Collects stats across org repos: commits, PRs, issues.

Usage

```
report_generate(
  type = c("weekly", "monthly"),
  org = "rladies",
  exclude_pattern = "^meetup-"
)
```

Arguments

type Report type: "weekly" or "monthly".
 org GitHub organization. Defaults to "rladies".
 exclude_pattern Regex to exclude repos. Defaults to "^meetup-".

Value

A named list with report data (invisibly).

<code>review_check_pr_naming</code>	<i>Check file naming conventions in a PR</i>
-------------------------------------	--

Description

Validates that changed files follow repository naming conventions.

Usage

```
review_check_pr_naming(owner, repo, pr_number)
```

Arguments

<code>owner</code>	Repository owner.
<code>repo</code>	Repository name.
<code>pr_number</code>	PR number.

Value

Data frame with filename and valid columns.

<code>review_run</code>	<i>Run all PR review automation</i>
-------------------------	-------------------------------------

Description

Assigns reviewers, labels the PR, and posts a checklist comment.

Usage

```
review_run(owner, repo, pr_number)
```

Arguments

<code>owner</code>	Repository owner.
<code>repo</code>	Repository name.
<code>pr_number</code>	Pull request number.

Value

Invisibly returns NULL. Called for its side effects of assigning reviewers, labelling the PR, and posting a checklist comment.

`rum_collect_analytics` *Collect Cloudflare RUM (Web Analytics) traffic data*

Description

Queries the Cloudflare Web Analytics beacon data for a date range, via `cloudflarer::cf_rum_page_views()` and `cloudflarer::cf_rum_top()`. A sibling to `website_collect_analytics()` (which queries Plausible) — Cloudflare RUM is a separate, beacon-based data source.

Usage

```
rum_collect_analytics(
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  site_tag = Sys.getenv("CLOUDFLARE_RUM_SITE_TAG"),
  token = Sys.getenv("CLOUDFLARE_API_TOKEN"),
  since = Sys.Date() - 30,
  until = Sys.Date()
)
```

Arguments

<code>account_id</code>	Cloudflare account ID. Defaults to env <code>CLOUDFLARE_ACCOUNT_ID</code> .
<code>site_tag</code>	RUM site tag, as returned by <code>cloudflarer::cf_list_rum_sites()</code> . Defaults to env <code>CLOUDFLARE_RUM_SITE_TAG</code> .
<code>token</code>	Cloudflare API token. Defaults to env <code>CLOUDFLARE_API_TOKEN</code> .
<code>since, until</code>	Date range, half-open [<code>since</code> , <code>until</code>). Defaults to the last 30 days.

Value

Named list with `site_tag`, `since`, `until`, `pageviews`, `top_pages`, `top_sources`, and `top_countries`.

`rum_format_analytics` *Format Cloudflare RUM analytics as markdown*

Description

Format Cloudflare RUM analytics as markdown

Usage

```
rum_format_analytics(analytics)
```

Arguments

<code>analytics</code>	Data from <code>rum_collect_analytics()</code> .
------------------------	--

Value

Character string with markdown-formatted report.

<code>rum_generate_report</code>	<i>Generate a Cloudflare RUM analytics report</i>
----------------------------------	---

Description

Collects Cloudflare Web Analytics data and formats it as markdown. A sibling to [website_generate_report\(\)](#) (Plausible-backed).

Usage

```
rum_generate_report(  
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),  
  site_tag = Sys.getenv("CLOUDFLARE_RUM_SITE_TAG"),  
  token = Sys.getenv("CLOUDFLARE_API_TOKEN"),  
  since = Sys.Date() - 30,  
  until = Sys.Date(),  
  output_path = NULL  
)
```

Arguments

<code>account_id</code>	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
<code>site_tag</code>	RUM site tag, as returned by <code>cloudflarer::cf_list_rum_sites()</code> . Defaults to env CLOUDFLARE_RUM_SITE_TAG.
<code>token</code>	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.
<code>since, until</code>	Date range, half-open [<code>since</code> , <code>until</code>). Defaults to the last 30 days.
<code>output_path</code>	Optional path to write JSON data.

Value

Named list with analytics and markdown (invisibly).

setup_channel_process	<i>Set up a Slack channel with RLadies+ bookmarks</i>
-----------------------	---

Description

The "setup-channel" command handler: joins the channel if needed (public channels only - a private channel requires a human invite first), then adds any bookmarks from `inst/config/bookmarks.json` the channel doesn't already have. R port of `slash_setup_channel()` from the deleted `worker/src/slash-local.js`; reads the bookmarks config directly from this package's `inst/` instead of the Worker's redundant GitHub-raw fetch of the same file.

Usage

```
setup_channel_process(team_id, channel_id, channel_name)
```

Arguments

<code>team_id</code>	Slack team id.
<code>channel_id</code>	Channel to set up.
<code>channel_name</code>	Channel display name, for the reply text.

Value

Character scalar reply text.

short_url	<i>Shorten a URL using the Short.io API</i>
-----------	---

Description

Shorten a URL using the Short.io API

Usage

```
short_url(uri)
```

Arguments

<code>uri</code>	URL to shorten.
------------------	-----------------

Value

Shortened URL string.

slack_api_call	<i>Generic low-level Slack API caller shared by the event/command handlers that need more than <code>slack_post_message()</code>'s single <code>chat.postMessage</code> call - DM opening, channel lookup, bookmarks, and so on. Retries on a 429 or 5xx response, honouring the Retry-After header up to 60s. Slack's Retry-After for a Tier 2 method is routinely 30-60s, so a shorter cap guarantees failure under sustained rate limiting - which a bulk pass over a workspace's channels hits readily.</i>
----------------	---

Description

Generic low-level Slack API caller shared by the event/command handlers that need more than `slack_post_message()`'s single `chat.postMessage` call - DM opening, channel lookup, bookmarks, and so on. Retries on a 429 or 5xx response, honouring the Retry-After header up to 60s. Slack's Retry-After for a Tier 2 method is routinely 30-60s, so a shorter cap guarantees failure under sustained rate limiting - which a bulk pass over a workspace's channels hits readily.

Usage

```
slack_api_call(token, method, body = list(), encode = c("json", "form"))
```

Arguments

token	Slack bot token.
method	Slack Web API method name, e.g. "conversations.open".
body	Named list of request parameters.
encode	How to send body. "json" suits the <code>chat.*</code> methods that take structured blocks. "form" is required by the paginated read methods such as <code>conversations.list</code> , which ignore a JSON body outright - including <code>limit</code> and <code>cursor</code> , so a JSON-bodied call silently returns page one forever.

Value

The parsed JSON response (a list).

slack_bot_token	<i>Resolve the bot token for a jinx-operated Slack workspace</i>
-----------------	--

Description

R/GitHub Actions only ever operates in two known workspaces (organiser, community) via two static secrets, unlike the Worker's per-team OAuth-token KV store which supports arbitrary installs - jinx isn't installed anywhere else, so the simpler static lookup is enough.

Usage

```
slack_bot_token(workspace = c("organiser", "community"))
```

Arguments

workspace Either "organiser" or "community".

Value

The bot token string.

slack_channel_id_lookup

Look up a Slack channel's ID by name

Description

Caches the workspace's full channel list in KV (channel_index:{team_id}, same SLACK_TOKENS namespace and 1h TTL the Worker used) to avoid a paginated conversations.list call on every lookup. R port of slack_channel_id_lookup() from worker/src/slack-api.js.

Usage

```
slack_channel_id_lookup(
  team_id,
  name,
  workspace,
  namespace_id = slack_tokens_namespace_id(),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")
)
```

Arguments

team_id	Slack team id.
name	Channel name (without #).
workspace	"organiser" or "community".
namespace_id	KV namespace ID for SLACK_TOKENS.
account_id	Cloudflare account ID. Defaults to env CLOUDFLARE_ACCOUNT_ID.
api_token	Cloudflare API token. Defaults to env CLOUDFLARE_API_TOKEN.

Value

The channel ID, or NULL if not found.

slack_channel_index	<i>Index a workspace's public channels</i>
---------------------	--

Description

Shapes the paginated output of the internal `slack_conversations_list()` helper into the frame the copy plan compares against.

Usage

```
slack_channel_index(workspace = c("organiser", "community"), channels = NULL)
```

Arguments

workspace	Either "organiser" or "community".
channels	Optional pre-fetched channel list, for testing.

Value

A data frame with id, name, topic, purpose and is_member.

slack_channel_mention	<i>Resolve a channel name to a Slack <#id name> mention</i>
-----------------------	---

Description

Falls back to plain #name text if the lookup fails - matches the Worker's `channel_mention()` behaviour.

Usage

```
slack_channel_mention(team_id, name, workspace, channel_index = NULL)
```

Arguments

team_id	Slack team id.
name	Channel name (without #).
workspace	"organiser" or "community".
channel_index	Optional pre-loaded index from channel_index_load() , to avoid a redundant KV read when resolving several channel names in a row.

Value

Character string mention.

`slack_conversations_open`*Open a Slack direct message channel*

Description

Open a Slack direct message channel

Usage

```
slack_conversations_open(team_id, user_id, workspace)
```

Arguments

<code>team_id</code>	Slack team id.
<code>user_id</code>	Slack user id to DM.
<code>workspace</code>	"organiser" or "community".

Value

The DM channel ID, or NULL on failure.

`slack_interaction_process`*Process a Slack interaction from the invite-approval flow*

Description

The `slack_interaction` event handler registered in `jinx_events()`. Branches on `action_id` exactly like the three JS functions it replaces (`slack_invite_process_approval/_sent/_denial`, deleted from `worker/src/airtable-invite.js`), replying via `response_url` for each.

Usage

```
slack_interaction_process(action_id, action_data, admin_user, response_url)
```

Arguments

<code>action_id</code>	One of "invite_approve", "invite_deny", "invite_mark_sent".
<code>action_data</code>	Parsed button value payload: email, record_id, base_id, table_id, and (for approve replies) approver.
<code>admin_user</code>	Slack username who clicked the button.
<code>response_url</code>	The interaction's <code>response_url</code> .

Value

Invisibly, NULL.

slack_invite_approval_checklist_blocks

Build the post-approval checklist Block Kit card

Description

R port of slack_invite_approval_checklist_blocks() from the deleted worker/src/airtable-invite.js.

Usage

```
slack_invite_approval_checklist_blocks(
  email,
  approver,
  record_id,
  base_id,
  table_id
)
```

Arguments

email	Applicant email.
approver	Slack username who approved the request.
record_id	Airtable record ID.
base_id	Airtable base ID.
table_id	Airtable table ID.

Value

A list of Block Kit blocks.

slack_invite_batch *Send pending Slack invitations*

Description

Fetches pending invitees from Airtable and prepares invite emails with a Slack invitation link.

Usage

```
slack_invite_batch(
  invite_link,
  base_id = slack_invitees_base_id(),
  api_key = Sys.getenv("AIRTABLE_API_KEY"),
  sender_name = "RLadies+ Global",
  sender_email = "info@rladies.org",
  dry_run = TRUE
)
```

Arguments

invite_link	Active Slack invite URL (valid for 30 days).
base_id	Airtable base ID for Slack invitees.
api_key	Airtable API key.
sender_name	Name of the sender.
sender_email	Sender email address.
dry_run	If TRUE (default), only returns prepared emails.

Value

Data frame of prepared emails (invisibly).

slack_invite_request *Request a Slack invitation for someone not yet on the workspace*

Description

The bot cannot invite users directly (Slack admin scopes are not available to community apps). Instead, this function posts a message to an organisers' Slack channel with the email address and instructions for a workspace admin to action the invite manually, then flips the matching Airtable record's invited flag for audit.

Usage

```
slack_invite_request(
  email,
  channel = Sys.getenv("SLACK_INVITE_REQUEST_CHANNEL"),
  base_id = slack_invitees_base_id(),
  api_key = Sys.getenv("AIRTABLE_API_KEY"),
  token = Sys.getenv("SLACK_TOKEN")
)
```

Arguments

email	Email address to request an invite for.
channel	Slack channel where the request is posted. Defaults to SLACK_INVITE_REQUEST_CHANNEL env var, falling back to "global-team".
base_id	Airtable base ID for Slack invitees.
api_key	Airtable API key. If unset, the Airtable audit step is skipped.
token	Slack bot token (chat:write scope is enough).

Value

A single status string suitable for posting back as a PR comment.

slack_invite_request_blocks

Build the initial Slack invite-request Block Kit card

Description

R port of slack_invite_request_blocks() from the deleted worker/src/airtable-invite.js. chapter is shown but, matching the JS original, deliberately not carried in the button value payload.

Usage

```
slack_invite_request_blocks(email, name, chapter, record_id, base_id, table_id)
```

Arguments

email	Applicant email.
name	Applicant name, or NULL.
chapter	Chapter name, or NULL.
record_id	Airtable record ID.
base_id	Airtable base ID.
table_id	Airtable table ID.

Value

A list of Block Kit blocks.

slack_post_message *Post a message to a Slack channel*

Description

Sends a markdown-formatted message to the specified Slack channel using the Slack Web API.

Usage

```
slack_post_message(
  text,
  channel,
  token = Sys.getenv("SLACK_TOKEN"),
  unfurl = FALSE,
  blocks = NULL
)
```

Arguments

text	Message text (supports Slack mrkdwn formatting).
channel	Slack channel name (without #).
token	Slack API token. Defaults to <code>Sys.getenv("SLACK_TOKEN")</code> .
unfurl	Whether Slack may expand links into previews. Off by default: most of what Jinx posts links to an issue or a chapter page, where a preview card adds noise. Messages whose whole point is the linked page (community blog posts) turn it on.
blocks	Optional Slack Block Kit blocks. When given, text is still sent and serves as the notification and fallback text, which is what a push notification and an unsupported client show.

Value

API response (invisibly).

slack_response_url_post

Reply to a Slack interaction via its response_url

Description

Posts directly from R rather than relying on a GitHub Actions workflow step to curl the reply - keeps the reply logic testable and in one language. Slack's `response_url` accepts a message replacement for up to 30 minutes after the original interaction.

Usage

```
slack_response_url_post(
  response_url,
  text = NULL,
  blocks = NULL,
  replace_original = FALSE
)
```

Arguments

response_url	The response_url from a Slack interaction payload.
text	Fallback plain text (shown if blocks can't render).
blocks	Block Kit blocks list, or NULL.
replace_original	Whether to replace the original message.

Value

The raw response body (invisibly).

slack_subscribe_rss *Post a request to subscribe an RSS feed to a Slack channel*

Description

Slack only runs slash commands typed by a real user, so a bot cannot subscribe a feed directly. This posts an actionable request asking a human in the channel to run `/feed subscribe <url>`.

Usage

```
slack_subscribe_rss(
  rss_url,
  channel = "rladiesblogs",
  token = Sys.getenv("SLACK_TOKEN")
)
```

Arguments

rss_url	RSS feed URL to subscribe.
channel	Slack channel name (without #). Defaults to "rladiesblogs".
token	Slack API token.

Value

API response (invisibly).

slack_workspace_for_team
Map a Slack team id to its workspace name

Description

Map a Slack team id to its workspace name

Usage

```
slack_workspace_for_team(
  team_id,
  organiser_id = Sys.getenv("SLACK_ORGANIZER_TEAM_ID"),
  community_id = Sys.getenv("SLACK_COMMUNITY_TEAM_ID")
)
```

Arguments

team_id	Slack team id from an event/command payload.
organiser_id	Organiser workspace team id. Defaults to env SLACK_ORGANIZER_TEAM_ID.
community_id	Community workspace team id. Defaults to env SLACK_COMMUNITY_TEAM_ID.

Value

"organiser" or "community".

tags2hash	<i>Convert tags to hashtag string</i>
-----------	---------------------------------------

Description

Convert tags to hashtag string

Usage

```
tags2hash(tags)
```

Arguments

tags	Character vector of tags.
------	---------------------------

Value

Single string with hashtags.

validate_blog_entry	<i>Validate blog entry JSON files against schema</i>
---------------------	--

Description

Validate blog entry JSON files against schema

Usage

```
validate_blog_entry(  
  path,  
  schema = system.file("schemas", "blog-entry.json", package = "jinx")  
)
```

Arguments

path	Path to a directory containing blog JSON files, or a single JSON file path.
schema	Path to JSON schema file. Uses the bundled schema by default.

Value

A data frame with columns file, valid, and errors.

validate_directory_entries
<i>Validate directory entry JSON files against schema</i>

Description

Validate directory entry JSON files against schema

Usage

```
validate_directory_entries(  
  path,  
  schema = system.file("schemas", "directory-entry.json", package = "jinx")  
)
```

Arguments

- path Path to a directory containing JSON entry files, or a single JSON file path.
- schema Path to JSON schema file. Uses the bundled schema by default.

Value

A data frame with columns file, valid, and errors.

validate_directory_pr	<i>Post an automated directory review as a PR checklist comment</i>
-----------------------	---

Description

Runs the checks that used to be manual review-checklist items on the entry files changed in a PR (filenames, likely-duplicate slugs, contact method vs. social entry, stray contact info in free text) and posts a single consolidated checklist. Each entry lists clickable profile links so reviewers can confirm the social accounts resolve, rather than an unreliable automated check (the platforms block bot requests). Schema validity is covered separately by the directory repo's JSON validation.

Usage

```
validate_directory_pr(owner, repo, pr_number)
```

Arguments

owner	Repository owner.
repo	Repository name.
pr_number	PR number.

Value

Invisibly NULL.

```
website_collect_analytics
```

Collect website analytics from Cloudflare Web Analytics

Description

Queries the Cloudflare GraphQL Analytics API for visitor and pageview metrics over a date window. The website moved to Cloudflare, so Plausible has been removed; the Cloudflare collector is pending a token and site tag and is **not yet implemented**. The intended return shape matches what [website_format_analytics\(\)](#) consumes: aggregate, timeseries, top_pages, and top_sources.

Usage

```
website_collect_analytics(
  from = NULL,
  to = NULL,
  site_tag = Sys.getenv("CLOUDFLARE_SITE_TAG"),
  account_id = Sys.getenv("CLOUDFLARE_ACCOUNT_ID"),
  api_token = Sys.getenv("CLOUDFLARE_API_TOKEN")
)
```

Arguments

from, to	Date window (a Date or "YYYY-MM-DD" string). When both are NULL, defaults to the last 30 days.
site_tag	Cloudflare Web Analytics site tag. Defaults to Sys.getenv("CLOUDFLARE_SITE_TAG").
account_id	Cloudflare account tag. Defaults to Sys.getenv("CLOUDFLARE_ACCOUNT_ID").
api_token	Cloudflare API token with Account Analytics Read. Defaults to Sys.getenv("CLOUDFLARE_API_TOKEN").

Value

Named list with aggregate, timeseries, top_pages, and top_sources.

`website_format_analytics`*Format website analytics as markdown*

Description

Format website analytics as markdown

Usage

```
website_format_analytics(analytics)
```

Arguments

`analytics` Data from `website_collect_analytics()`.

Value

Formatted markdown string.

`website_generate_report`*Generate a website analytics report*

Description

Collects website analytics and formats as markdown.

Usage

```
website_generate_report(  
  from = NULL,  
  to = NULL,  
  period = NULL,  
  output_path = NULL  
)
```

Arguments

<code>from</code> , <code>to</code>	Date window (a Date or "YYYY-MM-DD" string). When both are NULL, defaults to the last 30 days.
<code>period</code>	Deprecated Plausible period string, accepted for backward compatibility with the command registry and ignored.
<code>output_path</code>	Optional path to write JSON data.

Value

Named list with analytics and markdown (invisibly).

website_merge_pending *Auto-merge PRs with pending label when date matches*

Description

Checks open PRs with "pending" label in the website repo. If a blog post's date matches today, merges the PR.

Usage

```
website_merge_pending(org = "rladies", repo = "rladies.github.io")
```

Arguments

org	GitHub organization.
repo	Repository name.

Value

Character vector of merged PR URLs (invisibly).

website_publish_report
Publish website analytics as a GitHub issue

Description

Publish website analytics as a GitHub issue

Usage

```
website_publish_report(  
  report_data,  
  org = "rladies",  
  target_repo = "global-team",  
  slack_channel = NULL  
)
```

Arguments

report_data	Data from website_generate_report() .
org	GitHub organization.
target_repo	Repository to publish to.
slack_channel	Optional Slack channel to post a summary to.

Value

Issue URL (invisibly).

welcome_message_render

Render a welcome DM for a newly joined Slack member

Description

Reads the welcome config and template directly from this package's inst/ (no GitHub-raw fetch needed - the content already ships here). Falls back to a hardcoded plain-text greeting if either read fails. R port of `welcome_message_render()` from `worker/src/slack-events.js`.

Usage

```
welcome_message_render(team_id, user_id, workspace, link = NULL)
```

Arguments

team_id	Slack team id.
user_id	Slack user id of the new member.
workspace	"organiser" or "community".
link	Optional pending chapter sign-up link from pending_link_consume() .

Value

Character string with the rendered welcome message.

welcome_send	<i>Send a welcome DM to a newly joined Slack member</i>
--------------	---

Description

The `team_join` event handler registered in `jinx_events()`: consumes any pending chapter sign-up link, opens a DM, renders the welcome message, and posts it. R port of `slack_event_handle_team_join()` from `worker/src/slack-events.js`.

Usage

```
welcome_send(team_id, user)
```

Arguments

<code>team_id</code>	Slack team id.
<code>user</code>	The event's Slack user object list: <code>id, profile\$email</code> .

Value

Invisibly, `NULL`.

Index

airtable_base_allowed, [7](#)
airtable_record_update, [8](#)
airtable_webhook_process, [8](#)
analytics_collect_chapter_activity, [9](#)
analytics_collect_chapter_activity(),
 [11](#)
analytics_collect_contributor_growth,
 [10](#)
analytics_collect_contributor_growth(),
 [11](#)
analytics_compute_trends, [10](#)
analytics_compute_trends(), [11](#)
analytics_format_markdown, [11](#)
analytics_generate_dashboard, [11](#)
announce_create_message, [12](#)
announce_post, [12](#)
announce_post_bluesky, [13](#)
announce_post_mastodon, [13](#)
announce_post_reply, [14](#)
announce_send_newsletter, [14](#)

blog_add_pr, [15](#)
blog_check_links, [15](#)
blog_check_links_repo, [16](#)
blog_create_entry, [16](#)
blog_feed_blocks, [17](#)
blog_feed_collect, [18](#)
blog_feed_collect(), [21](#), [22](#)
blog_feed_format, [18](#)
blog_feed_items, [19](#)
blog_feed_items(), [17](#), [19](#), [20](#)
blog_feed_new_items, [19](#)
blog_feed_og, [20](#)
blog_feed_og(), [17](#)
blog_feed_pending, [21](#)
blog_feed_pending(), [18](#)
blog_feed_post, [21](#)
blog_feed_run, [22](#)
blog_feed_run(), [22](#)
blog_feed_sources, [24](#)

blog_feed_sources(), [17](#), [19](#)

cf_ops_format_workers_report, [24](#)
cf_ops_get_kv_value, [25](#)
cf_ops_get_kv_value(), [26](#)
cf_ops_kv_delete, [25](#)
cf_ops_kv_put, [26](#)
cf_ops_kv_put(), [25](#)
cf_ops_list_kv_keys, [27](#)
cf_ops_list_kv_keys(), [25](#), [26](#)
cf_ops_list_workers_scripts, [27](#)
cf_ops_purge_cache, [28](#)
cf_ops_workers_invocations, [29](#)
cf_ops_workers_invocations(), [24](#)
cfp_check_deadlines, [29](#)
cfp_create_issue, [30](#)
cfp_list_open, [31](#)
cfp_recommend_speaker, [31](#)
channel_copy_apply, [32](#)
channel_copy_plan, [33](#)
channel_copy_plan(), [32](#)
channel_copy_proposals, [33](#)
channel_index_load, [34](#)
channel_index_load(), [142](#)
channel_promo_build, [35](#)
channel_promo_format, [36](#)
channel_promo_post, [36](#)
channel_rename_proposals, [37](#)
chapter_check_health, [38](#)
chapter_check_health(), [62](#)
chapter_checklist_tick, [38](#)
chapter_confirm_send, [39](#)
chapter_create, [40](#)
chapter_create_pr, [41](#)
chapter_create_setup, [42](#)
chapter_create_update, [42](#)
chapter_duplicate_check, [43](#)
chapter_duplicate_check(), [44](#), [45](#)
chapter_duplicate_comment, [44](#)
chapter_duplicate_report, [45](#)

- chapter_email_approvals, 46
- chapter_email_provision, 46
- chapter_find_issue, 47
- chapter_initiated_date, 48
- chapter_initiated_date(), 62
- chapter_mailbox_create, 48
- chapter_meetup_brief, 49
- chapter_meetup_groups, 50
- chapter_meetup_groups(), 88
- chapter_meetup_logo_upload, 51
- chapter_meetup_name, 51
- chapter_meetup_request, 52
- chapter_meetup_urlname, 52
- chapter_meta_fetch, 53
- chapter_monitor_status, 54
- chapter_monitor_status(), 113
- chapter_onboard_blocker, 54
- chapter_onboard_state, 55
- chapter_onboard_state(), 54, 55
- chapter_onboard_summary, 55
- chapter_onboard_website, 56
- chapter_organiser_submissions, 57
- chapter_remind_stale, 57
- chapter_repo_audit, 58
- chapter_repo_audit(), 59
- chapter_repo_audit_report, 59
- chapter_repo_create, 59
- chapter_repo_name, 60
- chapter_report_health, 60
- chapter_slack_prompt, 61
- chapter_slack_sent, 61
- chapter_status_reconcile, 62
- chapter_status_reconcile(), 63
- chapter_status_reconcile_report, 63
- chapter_status_report, 64
- chapter_team_audit, 64
- chapter_team_audit(), 65
- chapter_team_audit_report, 65
- chapter_team_create, 66
- chapter_team_description, 66
- chapter_team_name, 67
- chapter_team_slug, 68
- chapter_team_slug(), 59, 60, 66, 67
- chapter_validate_files, 68
- chapter_validate_files(), 69
- chapter_validate_report, 69
- chunk_markdown, 69
- cloudflare_account_id, 70
- cloudflare_embed, 70
- cloudflare_embed(), 71
- cloudflare_generate, 71
- cloudflare_vectorize_upsert, 72
- cloudflarer::cf_d1_query(), 122
- cloudflarer::cf_delete_kv_value(), 25
- cloudflarer::cf_list_workers_scripts(), 27
- cloudflarer::cf_purge_cache(), 28
- cloudflarer::cf_put_kv_value(), 26
- cloudflarer::cf_request(), 70–72
- cloudflarer::cf_resp(), 70–72
- cloudflarer::cf_rum_page_views(), 137
- cloudflarer::cf_rum_top(), 137
- cloudflarer::cf_workers_invocations(), 29
- cmd_attach_slack_context, 72
- cmd_authorize, 73
- cmd_execute, 74
- cmd_execute(), 86
- cmd_parse, 74
- cmd_parse(), 72–74, 87
- conference_generate_report, 75
- conference_list_speakers, 75
- contributor_format, 76
- contributor_list, 76
- contributor_list(), 76
- contributor_list_org, 77
- contributor_update, 77
- copilot_request_review, 78
- copilot_review_pr, 78
- copilot_sync_repo, 79
- copilot_sync_repo(), 78
- directory_crop_image, 80
- directory_mark_synced, 81
- directory_mark_synced(), 82
- directory_optimize_image, 82
- directory_purge_submissions, 82
- directory_sync_airtable, 83
- directory_synced_field(), 81
- directory_validate_filename, 84
- directory_verify_handles, 84
- event_authorize, 85
- event_create_summary, 85
- event_execute, 86
- event_list_chapter, 86
- event_list_chapter(), 85

- event_parse, [87](#)
- event_parse(), [72](#), [85](#), [86](#)
- event_sync_chapters, [87](#)
- event_sync_chapters(), [85](#)
- gather_rag_source, [88](#)
- gh_branch_upsert, [88](#)
- gh_greet_contributor, [89](#)
- gh_open_or_update_pr, [90](#)
- gh_post_checklist, [90](#)
- gh_thank_contributor, [91](#)
- gh_thank_issue_contributors, [92](#)
- gh_welcome_contributor, [92](#)
- gh_welcome_contributor(), [89](#)
- gha_generate_dashboard, [93](#)
- gt_check_invitations, [94](#)
- gt_create_offboarding, [94](#)
- gt_finalize_offboarding, [95](#)
- gt_finalize_onboarding, [95](#)
- gt_invite, [96](#)
- gt_remind_stale, [96](#)
- gt_schedule_onboarding_meeting, [97](#)
- guide_email_template, [98](#)
- guide_email_template(), [99](#)
- guide_template, [99](#)
- guide_template(), [98](#)
- i18n_check_coverage, [100](#)
- i18n_get_chapter_language, [100](#)
- i18n_list_languages, [101](#)
- i18n_translate_template, [101](#)
- i18n_validate_translations, [102](#)
- li_get_version, [102](#)
- li_media_upload, [103](#)
- li_oauth, [103](#)
- li_post_write, [104](#)
- li_req, [104](#)
- li_urn_me, [105](#)
- load_copilot_review_config, [105](#)
- load_copilot_review_config(), [79](#), [80](#)
- load_labels_config, [105](#)
- load_rag_sources, [106](#)
- load_review_rules, [106](#)
- load_teams_config, [106](#)
- mail_send, [107](#)
- mail_send(), [113](#)
- meeting_poll_best, [107](#)
- meeting_poll_best(), [109](#)
- meeting_poll_create, [108](#)
- meeting_poll_create(), [110](#)
- meeting_poll_format_best, [109](#)
- meeting_poll_format_created, [109](#)
- meeting_poll_get, [110](#)
- meeting_poll_ics, [110](#)
- meeting_poll_lock, [111](#)
- meetup_urlname_status, [111](#)
- meetup_urlname_status(), [50](#)
- pending_link_consume, [112](#)
- pending_link_consume(), [154](#)
- prepare_inactivity_emails, [112](#)
- promo_blurb, [113](#)
- promo_blurb(), [36](#)
- promo_eligible_channels, [114](#)
- promo_eligible_channels(), [114](#)
- promo_pick_channel, [114](#)
- question_coding_declined_count, [115](#)
- question_coding_declined_count(), [117](#)
- question_content_gaps, [115](#)
- question_content_gaps(), [117](#)
- question_digest_build, [116](#)
- question_digest_format, [116](#)
- question_digest_post, [117](#)
- question_digest_post(), [116](#)
- question_downvoted_rank, [118](#)
- question_downvoted_rank(), [117](#), [121](#)
- question_draft_guide_snippet, [118](#)
- question_feedback_format, [119](#)
- question_feedback_summary, [119](#)
- question_feedback_summary(), [119](#)
- question_gaps_rank, [120](#)
- question_gaps_rank(), [115](#), [121](#)
- question_log_format, [121](#)
- question_log_format(), [116](#)
- question_log_purge, [121](#)
- question_log_query, [122](#)
- question_log_query(), [115](#), [118](#), [120](#), [121](#)
- question_vote_apply, [123](#)
- r_challenge_adversarial_check, [123](#)
- r_challenge_allowed_packages, [124](#)
- r_challenge_difficulties, [125](#)
- r_challenge_difficulties(), [125](#), [126](#), [128](#)
- r_challenge_difficulty_badge, [125](#)

`r_challenge_difficulty_label`, 126
`r_challenge_draft_build`, 126
`r_challenge_draft_build()`, 127–130
`r_challenge_draft_post`, 127
`r_challenge_format_issue`, 128
`r_challenge_format_slack`, 128
`r_challenge_generate`, 129
`r_challenge_generate()`, 130
`r_challenge_issue_title`, 129
`r_challenge_open_issue`, 130
`r_challenge_open_issue()`, 127
`r_challenge_parse`, 130
`r_challenge_parse()`, 124, 129
`r_challenge_render_statement`, 131
`r_challenge_reprex_check`, 131
`r_challenge_reprex_check()`, 124
`rag_chunk_id`, 132
`rag_index_build`, 132
`random_emoji`, 133
`reaction_direction`, 133
`reaction_event_apply`, 134
`reaction_log_increment`, 134
`reaction_log_increment()`, 119
`report_format_markdown`, 135
`report_generate`, 135
`report_generate()`, 135
`review_check_pr_naming`, 136
`review_run`, 136
`rum_collect_analytics`, 137
`rum_collect_analytics()`, 137
`rum_format_analytics`, 137
`rum_generate_report`, 138

`setup_channel_process`, 139
`short_url`, 139
`slack_api_call`, 140
`slack_bot_token`, 140
`slack_bot_token()`, 37
`slack_channel_id_lookup`, 141
`slack_channel_id_lookup()`, 34
`slack_channel_index`, 142
`slack_channel_mention`, 142
`slack_conversations_open`, 143
`slack_interaction_process`, 143
`slack_invite_approval_checklist_blocks`, 144
`slack_invite_batch`, 144
`slack_invite_request`, 145
`slack_invite_request_blocks`, 146

`slack_post_message`, 146
`slack_post_message()`, 140
`slack_response_url_post`, 147
`slack_subscribe_rss`, 148
`slack_workspace_for_team`, 148

`tags2hash`, 149

`validate_blog_entry`, 149
`validate_directory_entries`, 150
`validate_directory_pr`, 150

`website_collect_analytics`, 151
`website_collect_analytics()`, 137, 152
`website_format_analytics`, 152
`website_format_analytics()`, 151
`website_generate_report`, 152
`website_generate_report()`, 138, 154
`website_merge_pending`, 153
`website_publish_report`, 153
`welcome_message_render`, 154
`welcome_message_render()`, 34
`welcome_send`, 155