Package: PsyLingLLM (via r-universe)

November 12, 2026

Type Package

Title Tools for Psycholinguistic Experiments with Large Language Models

Version 0.4.0

Description Provides tools for designing and running controlled psycholinguistic, psychological, cognitive, and educational experiments with large language models. Registry-based configuration separates models, providers, and application programming interfaces so that compatible services can share experimental designs. Functions support randomized and factorial trials, multi-turn conversations, response and reasoning capture, timing and token measurements, and structured result export.

Depends R (>= 4.0)

Imports curl, writexl, dplyr (>= 1.1.0), tibble (>= 3.2.0), jsonlite (>= 1.8.0), magrittr, yaml, readr, readxl, openxlsx, stringr, stats, utils, stringi

Suggests jsonvalidate, testthat (>= 3.0.0), withr

License MIT + file LICENSE

URL https://github.com/HanMingPsy/PsyLingLLM-R

BugReports https://github.com/HanMingPsy/PsyLingLLM-R/issues

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev libssl-dev libx11-dev zlib1g-dev

Repository https://hanmingpsy.r-universe.dev

Date/Publication 2026-10-07 14:44:09 UTC

RemoteUrl https://github.com/hanmingpsy/psylingllm-r

RemoteRef HEAD

RemoteSha 0e14fc4a6afd8c6777ec19c3a1300f4388152c42

Contents

adaptive_feedback_experiment . . . 3
build_body_pass2_structural . . . 6
build_registry_entry_from_analysis . . . 7
build_standardized_input . . . 8
cat_slowly . . . 9
classify_generation_interface . . . 9
common_error_blacklist . . . 10
compare_path_line . . . 11
complete_headers_for_pass2 . . . 11
conversation_experiment . . . 12
default_keyword_lexicon . . . 14
default_optional_keys . . . 14
detect_embedded_think_tag . . . 15
ensure_registry_header . . . 15
explain_alias . . . 16
extract_pass2_templates . . . 16
extract_usage_fields . . . 17
factorial_trial_experiment . . . 18
find_placeholder_path . . . 20
flatten_json_paths . . . 21
format_registration_preview . . . 21
generate_llm_experiment_list . . . 22
generate_llm_factorial_experiment_list . . . 22
get_model_config . . . 23
get_registry_entry . . . 24
get_registry_path . . . 25
has_nonempty_text . . . 25
infer_role_mapping_from_body . . . 26
infer_structure_from_placeholder . . . 27
inject_optional_params . . . 27
is_invalid_candidate . . . 28
is_key . . . 28
is_message_object . . . 29
list_get_by_path . . . 29
llm_caller . . . 30
llm_register . . . 32
load_registry . . . 35
make_alias . . . 35
make_pass2_probe_inputs . . . 36
merge_defaults_for_probe . . . 37
multi_model_experiment . . . 38
normalize_history_messages . . . 40
normalize_path_key_with_regex . . . 41
normalize_provider_label . . . 41
normalize_type_label . . . 42
probe_llm_streaming . . . 42

PsyLingLLM_Schema 44
register_endpoint_offline 45
register_endpoint_to_user_registry 46
render_pass2_path_consistency_report 47
save_experiment_results 47
score_candidates_ns 48
score_candidates_st 50
stream_reconstruct_text 51
strip_optionals_from_body 51
sub_placeholders 52
trial_experiment 53
update_progress_bar 54
validate_experiment_config 55
validate_registry_entry 56
validate_registry_schema 57

Index **58**

`adaptive_feedback_experiment`

Run an Adaptive Feedback Experiment

Description

Runs multi-turn conversations identified by `ConversationId` and ordered by `Turn`. After each successful response, `feedback_fn` may replace the next planned prompt or insert a bounded dynamic turn. Rolling history is passed to `llm_caller()` through `assistant_content`.

Usage

```
adaptive_feedback_experiment(
  model_key,
  generation_interface = "chat",
  api_key,
  api_url = NULL,
  data,
  trial_prompt = NULL,
  system_content = NULL,
  assistant_content = NULL,
  optionals = NULL,
  role_mapping = NULL,
  history_mode = c("all", "last"),
  max_history_turns = Inf,
  max_turns = NULL,
  stream = NULL,
  timeout = getOption("psylingllm.llm_timeout_sec", 120L),
  repeats = 1,
  random = FALSE,
```

```
  apply_mode = c("replace_next", "insert_dynamic"),
  feedback_fn,
  delay = 0,
  output_path = NULL,
  overwrite = TRUE,
  return_raw = FALSE
)

conversation_experiment_with_feedback(
  model_key,
  generation_interface = "chat",
  api_key,
  api_url = NULL,
  data,
  trial_prompt = NULL,
  system_content = NULL,
  assistant_content = NULL,
  optionals = NULL,
  role_mapping = NULL,
  history_mode = c("all", "last"),
  max_history_turns = Inf,
  max_turns = NULL,
  stream = NULL,
  timeout = getOption("psylingllm.llm_timeout_sec", 120L),
  repeats = 1,
  random = FALSE,
  apply_mode = c("replace_next", "insert_dynamic"),
  feedback_fn,
  delay = 0,
  output_path = NULL,
  overwrite = TRUE,
  return_raw = FALSE
)
```

Arguments

model_key	Character(1).
generation_interface	Character(1). Default "chat".
api_key	Character(1). Provider API key.
api_url	Optional character(1). Overrides registry default URL; required for non-official providers.
data	data.frame/tibble with ConversationId, Turn, Material; optional TrialPrompt.
trial_prompt	Optional character(1). Global prefix when row TrialPrompt is missing.
system_content	Optional character(1) or NULL. If NULL, uses registry ‘default_system‘ when available.

`assistant_content`
: Optional static few-shot seed: character vector or a list of message objects (‘list(role=..., content=...)‘). These appear before rolling history and are preserved as-is.

`optionals`
: Optional named list. If omitted, Registry defaults are used; explicit `NULL` suppresses them; a named list sends only the supplied provider-native parameters.

`role_mapping`
: Optional mapping for local labels; not forced into ‘llm_caller()‘ unless supplied.

`history_mode`
: "all" or "last". Default "all".

`max_history_turns`
: Integer(1) or Inf. Only for "all". Each turn contributes 2 messages.

`max_turns`
: Positive integer or `NULL`. Required as the per-conversation execution cap for `apply_mode = "insert_dynamic"`; ignored for `apply_mode = "replace_next"`.

`stream`
: Logical(1) or NULL. NULL uses registry default; otherwise force streaming/non-streaming.

`timeout`
: Integer(1). Per-request timeout seconds.

`repeats`
: Integer(1). Conversation block repetitions before randomization.

`random`
: Logical(1). Shuffle conversation order (Turn order preserved inside).

`apply_mode`
: "replace_next" or "insert_dynamic".

`feedback_fn`
: Function called as `feedback_fn(response, row, context)` after each successful response. It must return `NULL` or a named list containing only `name`, `next_prompt`, and `meta`. Text fields must be non-empty scalar strings when supplied, and `meta` must be JSON-serializable.

`delay`
: Numeric(1). Seconds between turns.

`output_path`
: Character or NULL. Where to save results/logs.

`overwrite`
: Logical(1). Overwrite outputs.

`return_raw`
: Logical(1). Include raw request/response.

Details

Feedback semantics: - apply_mode = "replace_next": DO NOT insert extra rows; simply overwrite the next planned row’s ‘TrialPrompt‘ in the same conversation. - apply_mode = "insert_dynamic": if under per-conversation cap, append a new executable row (next turn) using ‘next_prompt‘.

All request assembly (URL/headers/body/messages/defaults) is registry-driven by ‘llm_caller()‘; this function never mutates headers. Transport timeouts and provider HTTP errors both retain the caller’s public compatibility status `599`; provider errors are distinguished through their original `error$code` and recorded as experiment errors rather than timeouts.

`conversation_experiment_with_feedback()` is retained as a compatibility alias with the same arguments and behavior.

Value

Invisibly, a tibble with the PsyLingLLM result schema for each executed turn. Existing feedback fields are retained and the additional `FeedbackStatus`, `FeedbackApplied`, `FeedbackNextPrompt`, and `FeedbackMessage` fields make callback decisions auditable. `ResponseStatus` distinguishes empty final answers without changing `TrialStatus` compatibility.

`build_body_pass2_structural` *Build Pass-2 body*

Description

Normalize and simplify a raw Pass-1 request body into a structured Pass-2 form, retaining vendor fields and removing runtime-tunable optionals. The resulting body is schema-clean and suitable for registry serialization.

Usage

```
build_body_pass2_structural(pass1_body, optional_keys = NULL)
```

Arguments

`pass1_body`	list (raw). The raw Pass-1 input body from endpoint analysis.
`optional_keys`	Character vector of explicit optional parameter keys to remove. If `NULL`, defaults to `default_optional_keys()`.

Details

Rules

- **Messages style**: rebuild the messages *container* into a single message using the same field names inferred from the placeholder `"\${CONTENT}"` (no guessing); keep all other top-level fields from Pass-1 except optionals.
- **Single-prompt style**: keep the original key (e.g., `prompt`, `input`); remove optionals and retain other top-level fields.
- Always ensure `model` is present (prefer Pass-1; else use `model_guess`; otherwise `NA`).
- Append a clean merge point `"\${PARAMETER}: \${VALUE}"` at the top level to mark where optional parameters are injected.

Value

A list representing the normalized Pass-2 body, ready for YAML registry serialization.

See Also

strip_optionals_from_body, infer_structure_from_placeholder

`build_registry_entry_from_analysis` *Build a registry entry from analysis results*

Description

This function constructs a structured registry node from the results of an endpoint analysis, typically produced by ‘analyze_llm_endpoint()‘. It merges information from Pass-1 and Pass-2 analysis stages, preserving vendor-specific fields, normalizing optional parameters, and ensuring YAML-ready formatting for downstream usage.

Usage

```
build_registry_entry_from_analysis(
  model,
  analysis,
  provider = "official",
  generation_interface = NULL,
  url = NULL,
  headers_input = NULL,
  role_mapping_input = NULL,
  stream_param = NULL,
  optional_defaults = NULL
)
```

Arguments

`model`	Character(1). Model identifier, e.g. ‘"deepseek-chat"‘.
`analysis`	List. The result from ‘analyze_llm_endpoint()‘ containing ‘pass1‘ and ‘pass2‘ nodes.
`provider`	Character(1). Free-form provider tag, e.g. ‘"official"‘, ‘"chutes.ai"‘, or ‘"local"‘.
`generation_interface`	Character(1). Interface name (defaults to ‘"chat"‘). If ‘NULL‘, it will be inferred automatically via ‘classify_generation_interface()‘.
`url`	Character(1). Base URL; persisted as ‘default_url‘ only if ‘provider == "official"‘.
`headers_input`	Named list or ‘NULL‘. Custom HTTP headers. If ‘NULL‘, Pass-1 raw headers are used.
`role_mapping_input`	List or ‘NULL‘. Custom role mapping. If ‘NULL‘, the function infers mappings from Pass-1/Pass-2 bodies.
`stream_param`	Character scalar (optional).
`optional_defaults`	Named list. Default generation parameters (e.g. ‘list(stream = TRUE, max_tokens = 512)‘), stored under ‘input.optional_defaults‘.

Details

Changes vs previous version: - Omits ‘method‘ entirely when not provided (no ‘method: ~‘ in YAML). - In Pass-2 body, "\${PARAMETER}" fields are stored as ‘NULL‘ (‘~‘ in YAML) for cleaner output. - Accepts ‘optional_defaults‘ as a named list (e.g., ‘list(stream = TRUE, max_tokens = 512)‘) and stores it under ‘input.optional_defaults‘.

Value

A named list representing a single registry node, keyed by ‘"<model>@<provider>"‘. Each entry contains a nested structure describing model I/O, streaming behavior, token usage paths, and reasoning capabilities.

See Also

[analyze_llm_endpoint()], [classify_generation_interface()]

`build_standardized_input` *Build standardized Pass-2 headers & body from Pass-1 raw inputs*

Description

This function reconstructs a normalized set of request headers and body for Pass-2 processing, based purely on the structural layout of the original Pass-1 input. It uses structural inference around "\${CONTENT}" placeholders rather than heuristic keyword matching.

Usage

```
build_standardized_input(pass1_headers, pass1_body, optional_keys)
```

Arguments

`pass1_headers`	List (raw). The original request headers captured during Pass-1.
`pass1_body`	List (raw). The original request body captured during Pass-1.
`optional_keys`	Character vector or list. Optional field names to be explicitly retained in Pass-2 inference (e.g. ‘"stream"‘, ‘"max_tokens"‘). These serve as hints to preserve or normalize optional input elements.

Details

The goal is to generate a consistent and minimal representation of model input for registry or downstream YAML serialization.

Value

A named list containing three components:

headers_p2 Standardized headers for Pass-2 processing.

body_p2 Normalized body structure, inferred from ‘pass1_body‘.

diagnostics List of structural inference metadata, including style, container keys, message index, content keys, role mappings, default system prompt, and detected model ID.

See Also

[build_body_pass2_structural()], [complete_headers_for_pass2()]

---

cat_slowly *Print lines gradually with delay (console-safe)*

Description

Print lines gradually with delay (console-safe)

Usage

```
cat_slowly(lines, delay = 0.3, final_delay = 0, prefix = "")
```

Arguments

lines	Character vector of text lines.
delay	Numeric, seconds to wait between each line (default 0.3).
final_delay	Numeric, optional delay after the last line.
prefix	Optional string printed before each line (e.g. "»> ").

---

classify_generation_interface *Classify LLM generation interface from URL (robust and version-agnostic)*

Description

This unified function normalizes an API URL path, strips leading version prefixes (e.g. /api/v1/, /v2beta/), and classifies the interface type for LLM endpoints.

Usage

```
classify_generation_interface(url)
```

Arguments

`url`	character(1) Full or partial API URL

Details

Recognized endpoint patterns (case-insensitive, version-insensitive): - /chat, /chat/completions → "chat" - /completions → "completion" - /messages → "messages" - /conversation → "conversation" - /responses → "responses" - /generate or /text-generation → "generate" - /inference → "inference" Anything else → "unknown".

Value

character(1) interface type string

`common_error_blacklist` *Common provider error-field blacklist (path segments only)*

Description

Curated from OpenAI, Anthropic, Google/Vertex, Azure OpenAI, AWS Bedrock, Mistral, and Cohere docs. These are *field names* that typically occur in error/diagnostic payloads and should be excluded from answer/think scoring.

Usage

```
common_error_blacklist()
```

Details

Important: We intentionally DO NOT include "message" because many providers use '...message.content' for valid assistant responses.

Value

character vector of lowercase tokens (segments)

`compare_path_line` *Compare two optional scalar strings (paths). Return a structured line and a boolean flag.*

Description

Compare two optional scalar strings (paths). Return a structured line and a boolean flag.

Usage

```
compare_path_line(label, p1, p2, NS = FALSE)
```

Arguments

`label`	Human-readable label, e.g., "NS respond_path"
`p1`	Pass-1 selected path (character(1) or NULL)
`p2`	Pass-2 selected path (character(1) or NULL)
`NS`	Is NS answer or not.

Value

list(line=character(1), ok=logical(1))

`complete_headers_for_pass2` *Complete headers minimally for Pass-2 (keeps user values)*

Description

Complete headers minimally for Pass-2 (keeps user values)

Usage

```
complete_headers_for_pass2(pass1_headers, want_streaming = NA)
```

Arguments

`pass1_headers`	list
`want_streaming`	logical or NA

Value

list headers_p2

conversation_experiment

Run a Multi-turn Conversation Experiment

Description

Executes registry-driven multi-turn conversations identified by `ConversationId` and ordered by `Turn`. Running history is kept as structured role messages and passed to `llm_caller()` through `assistant_content`.

Usage

```
conversation_experiment(
  model_key,
  generation_interface = "chat",
  api_key,
  api_url = NULL,
  data,
  trial_prompt = NULL,
  system_content = NULL,
  assistant_content = NULL,
  optionals = NULL,
  role_mapping = NULL,
  history_mode = c("all", "last"),
  max_history_turns = Inf,
  stream = NULL,
  timeout = getOption("psylingllm.llm_timeout_sec", 120L),
  random = FALSE,
  repeats = 1,
  delay = 0,
  output_path = NULL,
  overwrite = TRUE,
  return_raw = FALSE
)
```

Arguments

`model_key`	Character(1). Registry key (e.g., "deepseek-chat" or "deepseek-chat@proxy").
`generation_interface`	Character(1). Interface name; default "chat".
`api_key`	Character(1). Provider API key.
`api_url`	Optional character(1). Overrides registry default URL; required for non-official providers.
`data`	data.frame/tibble with ConversationId, Turn, Material; optional TrialPrompt.
`trial_prompt`	Optional character(1). Global prompt prefix (if row TrialPrompt is missing).

`system_content` Optional character(1) or NULL. If NULL, registry ‘default_system‘ is used when available.

`assistant_content`
Optional static few-shot seed: character vector or a list of message objects (‘list(role=..., content=...)‘). These appear before rolling history and are preserved as-is.

`optionals` Optional named list. If omitted, Registry defaults are used; explicit `NULL` suppresses them; a named list sends only the supplied provider-native parameters.

`role_mapping` Optional mapping of roles. If absent, we only use registry mapping for **local labeling**, and we do **not** pass a role map to `llm_caller()` (no forcing).

`history_mode` One of ‘"all"‘, ‘"last"‘. Default ‘"all"‘.

`max_history_turns`
Integer(1) or Inf. Only for ‘history_mode="all"‘. Each turn = 2 messages (user+assistant).

`stream` Logical(1) or NULL. NULL uses registry default; otherwise force streaming/non-streaming.

`timeout` Integer(1). Per-request timeout in seconds.

`random` Logical(1). If TRUE, shuffle conversation order while preserving ascending Turn within each conversation.

`repeats` Integer(1). Number of repetitions per base conversation. Typically 1 in feedback loops; each run can dynamically add more trials.

`delay` Numeric(1). Delay (seconds) between turns.

`output_path` Character or NULL. Where to save results/logs.

`overwrite` Logical(1). Overwrite existing outputs.

`return_raw` Logical(1). Include raw request/response for debugging.

Details

Request assembly (URL, headers, body/messages, defaults) is driven entirely by the registry via `llm_caller()` — this function never mutates headers.

Required columns in ‘data‘: - ‘ConversationId‘ (character/factor) - ‘Turn‘ (integer/numeric) - ‘Material‘ (character)

Optional: - ‘TrialPrompt‘ (row-level); used when present, otherwise ‘trial_prompt‘.

History policy: - ‘history_mode = "all" | "last"‘ (default ‘"all"‘). If ‘"all"‘, you can cap with ‘max_history_turns‘ (number of past turns; each turn adds 2 messages).

Errors / timeouts: - Timeout inside `llm_caller()` produces ‘status = 599‘ and ‘TrialStatus = "TIMEOUT"‘. - A provider HTTP error retains public ‘status = 599‘, preserves its original status in ‘error$code‘, and produces ‘TrialStatus = "ERROR"‘.

Value

A data.frame/tibble with PsyLingLLM schema columns per turn: Response, Think, ModelName, TotalResponseTime, FirstTokenLatency (if present), PromptTokens, CompletionTokens, TrialStatus, ResponseStatus, Streaming, Timestamp, RequestID, plus HistoryMode and HistoryUsedMsgs.

`default_keyword_lexicon` *Default keyword lexicon for ns/st scoring*

Description

Default keyword lexicon for ns/st scoring

Usage

```
default_keyword_lexicon()
```

Value

A named list with strong_signal, ns, st, and blacklist entries.

`default_optional_keys` *Default optional keys for common LLM providers (explicit removal first)*

Description

This list aggregates widely-seen optional (runtime-tunable) fields across OpenAI, Anthropic, Google Gemini, Cohere, Mistral and Hugging Face style APIs. It is intentionally broad; tailor at call-site if needed.

Usage

```
default_optional_keys()
```

Value

character vector of key names

`detect_embedded_think_tag` *Detect embedded <think> tags in model output*

Description

This checks if any text fragment contains both <think> and </think>.

Usage

```
detect_embedded_think_tag(x)
```

Arguments

`x`	Character or list of characters.

Value

logical(1)

`ensure_registry_header` *Ensure registry directory exists and header is present*

Description

Creates the Registry directory and writes a file header if the Registry file does not exist yet or is empty.

Usage

```
ensure_registry_header(path = get_registry_path())
```

Arguments

`path`	character(1) registry yaml path

explain_alias *Explain how an alias was derived from a model id*

Description

Returns a tibble describing each token, whether it was kept/dropped, the reason category, and the final alias for auditing and debugging.

Usage

```
explain_alias(
  model_id,
  keep_version = c("none", "major", "major_minor", "full")
)
```

Arguments

model_id	Character scalar.
keep_version	One of c("none","major","major_minor","full").

Value

A tibble with columns: token, action, reason; attribute 'alias'.

Examples

```
explain_alias("meta-llama/llama-3.1-70b-instruct-128k", keep_version = "major")
```

extract_pass2_templates *Tolerant extraction of Pass-2 templates (headers and body)*

Description

This helper function extracts the standardized Pass-2 templates (headers and body) from a structured input object produced by [build_standardized_input()] or a compatible analysis result.

Usage

```
extract_pass2_templates(std)
```

Arguments

std	List. A standardized object containing Pass-2 fields such as 'headers_p2', 'body_p2', and optionally 'diagnostics'. Typically, this is the output of [build_standardized_input()].

Details

It is **tolerant** to missing or partial components — if ‘headers_p2‘ or ‘body_p2‘ are unavailable, the function gracefully falls back to their raw equivalents (‘headers‘ or ‘body‘). Diagnostics are returned if present.

Value

A named list with the following components:

headers_tmpl Standardized Pass-2 headers, or raw headers if missing.

body_tmpl Standardized Pass-2 body, or raw body if missing.

diag Diagnostics information if available, otherwise ‘NULL‘.

See Also

[build_standardized_input()]

---

extract_usage_fields *Extract usage field names from a parsed non-streaming response*

Description

Returns the names of fields inside the ‘usage‘ object if present. Safely handles absent/unnamed/atomic cases and always returns a character vector.

Usage

```
extract_usage_fields(ns_parsed)
```

Arguments

ns_parsed	A parsed JSON-like list (e.g., ‘resnon_streamparsed‘).

Value

A character vector of usage field names (possibly length 0).

Examples

```
# extract_usage_fields(list(usage = list(prompt_tokens=12, total_tokens=34)))
# -> c("prompt_tokens","total_tokens")
```

`factorial_trial_experiment` *Run Factorial Trial Experiment (registry-driven, assistant-native)*

Description

Builds a factorial trial list (carrier sentences × factor levels × optional CW), constructs a stimulus for each row (via 'fill_method' or sensible defaults), and calls `llm_caller()` for each trial using the package registry (no manual headers/body here).

Usage

```
factorial_trial_experiment(
  model_key,
  generation_interface = "chat",
  api_key,
  api_url = NULL,
  data,
  factors,
  CW = NULL,
  fill_method = NULL,
  trial_prompt = NULL,
  system_content = NULL,
  assistant_content = NULL,
  optionals = NULL,
  role_mapping = NULL,
  stream = NULL,
  timeout = getOption("psylingllm.llm_timeout_sec", 120L),
  repeats = 1,
  random = FALSE,
  delay = 0,
  output_path = NULL,
  overwrite = TRUE,
  return_raw = FALSE
)
```

Arguments

`model_key`	Character(1). Registry key (e.g., `"deepseek-chat"` or `"deepseek-chat@proxy"`).
`generation_interface`	Character(1). Interface name; default `"chat"`.
`api_key`	Character(1). Provider API key.
`api_url`	Optional character(1). Overrides registry default URL; required for non-official providers.
`data`	A `data.frame` or a file path accepted by `generate_llm_factorial_experiment_list()`. Must contain at least `Material` (carrier). If using CW, also provide `CW` or `Word`.

factors	A named list of factors, e.g., list(Congruity = c("Congruent", "Incongruent")).
CW	Optional data.frame with Item plus columns named by factor-level combinations in wide format.
fill_method	Optional function to build the per-row stimulus (see "Stimulus construction").
trial_prompt	Optional character(1). Global trial prefix; row-level TrialPrompt (if present) takes precedence.
system_content	Optional character(1) or NULL. If NULL, registry default_system is used when available.
assistant_content	Optional static few-shot seed: a character vector or a list of message objects (list(role = ..., content = ...)). These appear before rolling history and are preserved as-is.
optionals	Optional named list. The default NULL suppresses Registry defaults; a named list sends only the supplied provider-native parameters.
role_mapping	Optional mapping of roles (system/user/assistant). If absent, registry mapping applies.
stream	Logical(1) or NULL. If NULL, uses the registry default; otherwise forces streaming/non-streaming.
timeout	Integer(1). Per-request timeout in seconds (passed to llm_caller()).
repeats	Integer(1). Number of repetitions per factorial row.
random	Logical(1). Randomize trial order.
delay	Numeric(1). Delay (seconds) between trials.
output_path	Character or NULL. Path to save results/logs (same behavior as trial_experiment()).
overwrite	Logical(1). Overwrite existing outputs.
return_raw	Logical(1). Include raw request/response for debugging.

Details

The trial list is generated by generate_llm_factorial_experiment_list(), which yields columns such as Run, Item, Material (carrier), Word (CW), factor columns, and optionally TrialPrompt. This function then produces a per-row user message (stimulus) and executes the trials.

Stimulus construction - If fill_method is provided, it is called as:

```
fill_method(
  condition = list(<Factor> = <Level>, ...),
  material  = <carrier sentence>,
  word      = <CW string or NA>
)
```

- If fill_method is NULL, defaults to: 1) Replace the first occurrence of one of the placeholders "\{CW\}", "[CW]", "__CW__", or "____" in Material with Word (if non-NA); 2) Otherwise, paste(Material, Word) if a CW word exists; 3) Otherwise, just Material.

Error & timeout handling - A transport timeout uses public status = 599 and becomes TrialStatus = "TIMEOUT". - A provider HTTP error also retains public status = 599, preserves its original status in error$code, and becomes TrialStatus = "ERROR".

Value

A `data.frame` (or tibble) with PsyLingLLM schema columns per trial: `Response`, `Think`, `ModelName`, `TotalResponseTime`, `FirstTokenLatency`, `PromptTokens`, `CompletionTokens`, `TrialStatus`, `ResponseStatus`, `Streaming`, `Timestamp`, `RequestID`, plus `Stimulus` and `ConditionLabel`.

`find_placeholder_path` *Recursively find the first path to a placeholder string*

Description

This utility function traverses a nested R object (typically a list) to locate the first scalar string that exactly matches a given placeholder value (by default, `"\${CONTENT}"`). It performs a depth-first search and returns the traversal path as a list of keys and/or indices.

Usage

```
find_placeholder_path(x, placeholder = "${CONTENT}")
```

Arguments

`x`	A nested R object (usually a list or list-like structure) to be searched.
`placeholder`	Character(1). The exact scalar string value to look for. Defaults to `"\${CONTENT}"`.

Details

Path segments are expressed as character keys or integer indices, for example: “‘ list("messages", 2, "content") “‘ indicating that the placeholder was found at ‘x[["messages"]][[2]][["content"]]‘.

Value

A list of path segments (keys or indices) leading to the first occurrence of the placeholder, or ‘NULL‘ if not found.

See Also

[build_body_pass2_structural()]

`flatten_json_paths` *Flatten a JSON-like object into (path, key, value) rows*

Description

Flatten a JSON-like object into (path, key, value) rows

Usage

```
flatten_json_paths(x, keep_numeric = FALSE)
```

Arguments

`x`	list parsed from JSON
`keep_numeric`	logical; keep numeric leaves as strings

Value

data.frame(path, key, value, type)

`format_registration_preview` *Render a registry entry preview (verbatim from the registry node)*

Description

This function renders a human/CI readable preview **directly from the registry entry**, without any heuristics. It mirrors the exact structure that will be serialized into YAML (provider, input/body/fallback_body/optionals, output paths, streaming config).

Usage

```
format_registration_preview(entry)
```

Arguments

`entry`	A single-model registry entry (named list) as returned by 'build_registry_entry_from_analysis()', e.g.: list("<model_key>" = list(completion = <iface_node>, chat = <iface_node>, ...))

Details

The function supports entries with one or more interfaces (e.g., "chat", "completion"). Each interface is printed in order.

Value

Character vector of lines suitable for printing to console/CI.

`generate_llm_experiment_list` *Generate LLM Experiment List (CSV→XLSX conversion, UTF-8 safe)*

Description

Create an experiment trial table from a CSV/XLSX file or a data.frame. CSV files are temporarily converted to XLSX to prevent encoding issues. UTF-8 compatibility is ensured, and 'Item' and 'Run' are automatically generated if missing.

Usage

```
generate_llm_experiment_list(
  data,
  trial_prompt = "",
  repeats = 1,
  random = FALSE,
  force_base_cols = TRUE,
  save_path = NULL
)
```

Arguments

`data`	Path to CSV/XLSX file or a data.frame containing experiment materials.
`trial_prompt`	Character string applied globally if not overridden per trial.
`repeats`	Integer, number of times to repeat the dataset.
`random`	Logical, whether to randomize trial order.
`force_base_cols`	Logical, whether to force 'Run' and 'Item' to appear first.
`save_path`	Optional path to save the processed CSV/XLSX file.

Value

A data.frame containing 'Run', 'Item', 'Material', optional metadata, and 'TrialPrompt'.

`generate_llm_factorial_experiment_list` *Generate LLM Experiment List for Factorial Design (UTF-8 safe)*

Description

This function reads carrier sentences from a CSV/XLSX file or a data.frame, optionally expands a CW (critical word) wide table into long format, applies factors, repeats, randomizes order, and ensures UTF-8 safe text.

Usage

```
generate_llm_factorial_experiment_list(
  data,
  factors,
  CW = NULL,
  trial_prompt = "",
  repeats = 1,
  random = FALSE,
  force_base_cols = TRUE,
  save_path = NULL
)
```

Arguments

data	Path to CSV/XLSX file or a data.frame with columns 'Item' and 'Material'.
factors	List of factors, e.g. 'list(Congruity = c("Congruent","Incongruent"))'.
CW	data.frame with 'Item' and columns corresponding to factor-level combinations (wide format).
trial_prompt	Character string applied globally if not overridden per trial.
repeats	Integer, number of times to repeat the dataset.
random	Logical, whether to randomize trial order.
force_base_cols	Logical, whether to force 'Run'/'Item' to appear first.
save_path	Optional character, path to save the processed CSV/XLSX.

Value

A long-format data.frame with 'Run', 'Item', 'Material', 'Word', factor columns, 'TrialPrompt'.

---

get_model_config *Resolve a model name (id or alias) to a registry entry (new registry-aware)*

Description

Works with the new load_registry() bundle (list with $merged). Still supports passing a preloaded flat registry list via 'registry'.

Usage

```
get_model_config(model_name, registry = NULL)
```

Arguments

model_name	Character. Official model id pasted by user, or an alias.
registry	Optional. Either a flat list of models.

Value

A list (model config) or NULL if not found.

`get_registry_entry` *Retrieve a registry interface entry (inputs / outputs / streaming)*

Description

Look up a model first in the **user registry** (by default at ‘get_registry_path()‘), and if not found, fall back to the **system registry** located at ‘inst/registry/system_registry.yaml‘. Select an interface (e.g., ‘"chat"‘, ‘"completion"‘) and return a **normalized** node ready for request assembly.

Usage

```
get_registry_entry(
  model_key,
  generation_interface = NULL,
  path = get_registry_path()
)
```

Arguments

`model_key`	Character(1). Either ‘"<model>"‘ or ‘"<model>@<provider>"‘.
`generation_interface`	Character(1) or NULL. One of ‘"chat"‘, ‘"completion"‘, ‘"messages"‘, ‘"conversation"‘, ‘"responses"‘, ‘"generate"‘, ‘"inference"‘. If ‘NULL‘ and the model has exactly one interface, that interface is selected.
`path`	Character(1). User registry path. Defaults to ‘get_registry_path()‘.

Details

Normalization rules: - ‘input.role_mapping‘ may be partially specified or absent; we keep it as is (possibly ‘NULL‘ per key). Downstream callers decide whether to apply it. - Boolean-like values such as ‘"yes"/"no"‘, ‘"true"/"false"‘, ‘0/1‘ are coerced to logicals for ‘reasoning‘ and ‘streaming.enabled‘. - ‘optional_defaults‘ scalars are lightly normalized (numeric-like strings → numeric; boolean-like strings → logical) while preserving unknown vendor fields. - Output path fields (e.g., ‘respond_path‘, ‘delta_path‘) are **left as-is** (e.g., ‘list("choices..message.content")‘).

Model key conventions: - Official providers typically use keys **without** ‘@‘ (e.g., ‘"deepseek-chat"‘). - Non-official providers should use keys **with** ‘@provider‘ (e.g., ‘"deepseek-chat@proxy"‘).

Value

A list with fields:

model_key Resolved key (may include ‘@provider‘).

interface Selected interface name.

provider Provider label from the registry node (normalized to lower).

reasoning Logical. Whether the provider exposes reasoning fields.

input List with 'default_url', 'headers', 'body', 'fallback_body', 'optional_defaults', 'default_system', and 'role_mapping' (kept as provided).

output List with 'respond_path', 'thinking_path', 'id_path', 'object_path', and optional 'token_usage_path'.

streaming List with 'enabled', 'delta_path', 'thinking_delta_path'.

interfaces Character vector of available interfaces for this model.

Examples

```
## Not run:
  ent <- get_registry_entry("deepseek-chat", generation_interface = "chat")
  ent$input$role_mapping$user
  ent$output$respond_path

## End(Not run)
```

`get_registry_path` *Get user registry path (cross-platform, unified)*

Description

Uses the platform-specific user configuration directory returned by `tools::R_user_dir("PsyLingLLM", "config")`. The historical `~/.psylingllm/model_registry.yaml` location remains readable but is not rewritten automatically.

Usage

```
get_registry_path()
```

Value

Path to user registry YAML

`has_nonempty_text` *Does a value contain any non-empty textual content?*

Description

Robustly checks lists, vectors, raw, and scalars. Returns a single TRUE/FALSE.

Usage

```
has_nonempty_text(val)
```

Arguments

val	Any R object (e.g., result of json_get_by_path()).

Value

logical(1)

Examples

```
has_nonempty_text("hi")         # TRUE
has_nonempty_text(c("", "x"))   # TRUE
has_nonempty_text(character())  # FALSE
has_nonempty_text(NULL)         # FALSE
```

---

infer_role_mapping_from_body

Infer role mapping from a chat-style LLM request body

Description

This helper inspects a JSON-like request body used by chat-based LLM APIs and infers how conversational roles (e.g., *user*, *assistant*, *system*) are represented in provider-specific message structures.

Usage

```
infer_role_mapping_from_body(pass1_body)
```

Arguments

pass1_body	A list representing the request body (usually as captured during Pass-1 analysis). Typically this is the same structure passed to [build_standardized_input()] or [llm_register()].

Details

The function analyzes structural metadata (as produced by [infer_structure_from_placeholder()]) to identify the message container, role keys, and message indices. It returns a mapping that can later be used in the registry under ‘input$role_mapping‘ to ensure consistent message composition across providers.

Value

A named list with up to three fields describing provider-specific role labels:

user Label for the user role (e.g. ‘"user"‘ or ‘"human"‘).

assistant Label for the assistant role (e.g. ‘"assistant"‘, ‘"bot"‘, or ‘"model"‘).

system Label for the system/instruction role (e.g. ‘"system"‘ or ‘"context"‘). May include a companion field ‘system_content‘ with the detected instruction text.

Returns ‘NULL‘ if no mapping can be inferred.

See Also

[build_standardized_input()], [infer_structure_from_placeholder()]

`infer_structure_from_placeholder` *Infer structure from where "\${CONTENT}" lives*

Description

- If "\${CONTENT}" sits inside a small list which is inside a larger list, treat it as messages-style. * content_key = name of the element holding "\${CONTENT}" * role_key = the *immediately previous* element name in that small list * container_key = the named key that holds the sequence of messages * message_index = the numeric position in that sequence - If "\${CONTENT}" is directly under a top-level key, treat as single-prompt.

Usage

```
infer_structure_from_placeholder(body, placeholder = "${CONTENT}")
```

Arguments

body	A list or character vector representing the raw input structure in which the placeholder is to be searched.
placeholder	Character(1). The placeholder string to search for, typically "\${CONTENT}".

Value

list(style, container_key, message_index, content_key, role_key, default_system)

`inject_optional_params` *Inject optional params into a body using a whitelist*

Description

Inject optional params into a body using a whitelist

Usage

```
inject_optional_params(body, optional_params = NULL, whitelist = NULL)
```

Arguments

body	list
optional_params	named list of candidates to inject
whitelist	character vector of allowed keys

`is_invalid_candidate` *Hard filter: blacklist, nullish, booleans, numeric-only, too-short Segment-exact blacklist (compares path segments, not substrings).*

Description

Hard filter: blacklist, nullish, booleans, numeric-only, too-short Segment-exact blacklist (compares path segments, not substrings).

Usage

```
is_invalid_candidate(
  path,
  key,
  value,
  blacklist_tokens = default_keyword_lexicon()$blacklist
)
```

Arguments

path, key, value	character scalars
blacklist_tokens	character vector

Value

logical

`is_key` *Utility: check if a path segment is a named key*

Description

Determines whether a given path segment represents a valid named key (i.e., a non-empty single-length character string). Used internally by list/path traversal helpers such as [list_get_by_path()] and [find_placeholder_path()].

Usage

```
is_key(seg)
```

Arguments

seg	A single path segment to test (may be character, numeric, or other type).

Value

Logical scalar. ‘TRUE‘ if the segment is a non-empty character key, otherwise ‘FALSE‘.

---

is_message_object *Check if an object is a message object*

Description

Tests whether the input is a list that follows the message-object schema: it must contain fields ‘role‘ and ‘content‘, where ‘role‘ is a character vector of length 1, and ‘content‘ is either a character vector or a list.

Usage

```
is_message_object(x)
```

Arguments

x	Any R object.

Value

A logical scalar. Returns ‘TRUE‘ if ‘x‘ conforms to the message object structure; otherwise ‘FALSE‘.

---

list_get_by_path *Get a nested value by a mixed path (character keys and integer indices)*

Description

This helper function retrieves a nested value from a list-like object by following a *mixed path* consisting of character keys (for named lists) and/or integer indices (for unnamed elements).

Usage

```
list_get_by_path(x, path)
```

Arguments

x	A list or list-like R object from which to retrieve a nested value.
path	A vector or list of path segments. Each segment may be either: • A **character key**, e.g. '"messages"'; • An **integer index**, e.g. '2L'. The function follows these segments in order to descend into nested structures.

Details

It is tolerant to structural inconsistencies: if any intermediate component is missing, 'NULL', or not a list, the function returns 'NULL' rather than throwing an error.

Value

The nested value located at the specified path, or 'NULL' if the path does not exist or cannot be traversed.

See Also

[find_placeholder_path()]

---

llm_caller *Call an LLM via the registry*

Description

Builds a protocol request from the resolved model registry entry. Provider-native request parameters may be supplied through `optionals` or `...`; protocol-owned request fields remain controlled by the selected interface. The user message is composed from `trial_prompt` and `material`. Optional `system_content` and `assistant_content` are inserted before the user message *only* if the registry template actually supports `"\${ROLE}"` in the messages shape. Roles are not mapped by default; mapping is applied only when the caller supplies `role_mapping`.

Usage

```
llm_caller(
  model_key,
  generation_interface = NULL,
  api_url = NULL,
  trial_prompt = NULL,
  material = NULL,
  system_content = NULL,
  assistant_content = NULL,
  api_key = NULL,
  optionals,
  stream = NULL,
```

```
  role_mapping = NULL,
  timeout = 120,
  return_raw = FALSE,
  debug = FALSE,
  ...
)
```

Arguments

model_key	Character(1). Registry key, e.g., "deepseek-chat" (official, no "@") or "deepseek-chat@proxy" (non-official, with "@provider").
generation_interface	Character(1) or NULL. One of: "chat", "completion", "messages", "responses", "conversation", "generate", or "inference". If NULL and exactly one interface exists, it is selected automatically.
api_url	Character(1) or NULL. Optional only when registry provider is "official" (falls back to input.default_url). For non-official providers, this is required. If provided, it always overrides the registry default.
trial_prompt	Character(1) or NULL. Trial instruction text.
material	Character(1) or NULL. Stimulus/item content to combine with trial_prompt.
system_content	Character(1) or NULL. Optional system message to prepend. If NULL and registry input.default_system is set, that default is used. If both are missing and the template supports roles, a single warning is emitted and no system message is inserted.
assistant_content	Optional static few-shot seed: character vector or a list of message objects (list(role=..., content=...)). These appear before rolling history and are preserved as-is.
api_key	Character(1) or NULL. Injected into "\${API_KEY}" placeholders in headers/body.
optionals	Missing, NULL, or a named list. Missing → use registry defaults if present; NULL → none; named list → user keys only (no merge).
stream	Logical(1) or NULL. Overrides both optionals$stream and registry streaming.enabled.
role_mapping	Named list or NULL. Optional override to map abstract roles ("system", "assistant", "user") to provider labels. By default, no mapping is applied.
timeout	Numeric(1), default = 120. Request timeout in seconds.
return_raw	Logical(1), default = FALSE. If TRUE, include raw request/response in the result.
debug	Logical(1), default = FALSE. If TRUE, print diagnostic information.
...	Named provider-native request parameters. Names and values are sent unchanged. When the same name also appears in optionals, the value in ... wins with a warning.

Details

URL resolution

- User-provided api_url always takes precedence;

- When `provider == "official"`, `api_url` is optional (falls back to registry `input.default_url`);
- Otherwise (non-official), `api_url` is required.

Provider parameters use a three-state contract:

- Missing: use registry `input.optional_defaults` if present; otherwise inject nothing;
- `NULL`: inject nothing;
- Named list: use user keys only (do not merge defaults).

Native Registry v2 builders send provider parameter names and values unchanged. The legacy Registry v1 compatibility builder injects them through the `"\${PARAMETER}"` template anchor when present.

Response parsing is selected by the resolved registry interface. Native Registry v2 parsers interpret protocol-specific response and streaming event semantics; the Registry v1 compatibility parser retains the historical path-based extraction rules.

Value

A list with fields: `status`, `interface`, `model_key`, `streaming`, `usage`, `answer`, `thinking`, `response_status`, and optionally `raw` or `error`. `response_status` is one of `"OK"`, `"EMPTY_RESPONSE"`, `"TIMEOUT"`, or `"ERROR"`. Transport and provider failures retain the compatibility status `599`; the original HTTP status and provider evidence remain available in `error`.

`llm_register` *Register an LLM Endpoint with Pass-2 Validation*

Description

Orchestrates a two-pass analysis to register an arbitrary LLM endpoint into the PsyLingLLM registry. **Pass-1** probes the raw endpoint (non-stream and streaming/SSE) and scores likely extraction paths for ‘answer‘ and ‘thinking‘ fields. **Pass-2** builds a *standardized* request template (using the provider’s own field names), re-probes the endpoint, and verifies that the selected ports are still retrievable. Finally, it composes a registry entry and, if requested, writes/merges it into the user registry.

Usage

```
llm_register(
  url,
  provider = "official",
  headers,
  body,
  api_key,
  content_value = "Hello!",
  generation_interface = NULL,
  default_system_prompt = NULL,
  optional_defaults = NULL,
```

```
  role_mapping = NULL,
  timeout = 120,
  top_k = 5,
  ns_prob_thresh = list(answer = 0.6, think = 0.55),
  st_prob_thresh = list(answer = 0.7, think = 0.55),
  lexicon = default_keyword_lexicon(),
  stream_param = NULL,
  auto_register = FALSE
)
```

Arguments

`url`	Character(1). Endpoint URL.
`provider`	Character(1). Provider label stored in the registry. One of `"official"`, `"vllm"`, `"proxy"`, `"custom"`.
`headers`	Named list of HTTP headers. May include placeholders such as `\${API_KEY}`; `Accept: text/event-stream` is added automatically during probing if streaming requires it.
`body`	List template for the request payload. May include placeholders `\${ROLE}`, `\${CONTENT}`, and a parameter merge anchor `\${PARAMETER}`. The raw Pass-1 body is preserved as a fallback in the resulting registry entry.
`api_key`	Character(1). API/Bearer key used to substitute `\${API_KEY}` in headers/body during probes.
`content_value`	Character(1). Test content for the user message during probing (e.g., `"Hello!"`).
`generation_interface`	Optional character(1). Interface name to record in the entry (e.g., `"chat"`, `"messages"`, `"responses"`). If `NULL`, it may be auto-classified from `url`.
`default_system_prompt`	Optional character(1). Structural default system message to include in the entry (if applicable). Not forced at runtime.
`optional_defaults`	Named list of typed defaults (e.g., `list(stream = TRUE, max_tokens = 512)`). These are recorded in the entry and used by callers unless explicitly overridden.
`role_mapping`	Optional named list mapping roles (e.g., `list(system = "system", user = "user", assistant = "assistant")`). If `NULL`, the function may infer mapping from body structure for recording purposes; mapping is never *forced* unless a caller supplies it.
`timeout`	Numeric(1). Per-probe timeout in seconds.
`top_k`	Integer(1). Number of near-miss candidates to keep in the printed ranking tables.
`ns_prob_thresh`	Named list with probability thresholds for Pass-1/2 non-stream selection, e.g. `list(answer = 0.60, think = 0.55)`.
`st_prob_thresh`	Named list with probability thresholds for Pass-1/2 streaming selection, e.g. `list(answer = 0.70, think = 0.55)`.
`lexicon`	Keyword lexicon (list) for candidate scoring.

`stream_param` Character scalar (optional). Name of the request-body field used by the provider to enable streaming. For most OpenAI-compatible APIs this is ‘"stream"‘, but some endpoints use alternative keys such as ‘"streaming"‘, ‘"enable_sse"‘, or ‘"use_stream"‘. If ‘NULL‘, the function falls back to ‘"stream".

`auto_register` Logical(1). If `TRUE`, the composed entry is immediately written/merged into the user registry; otherwise only a preview is printed.

Details

This function prints a human-readable report (status, candidate paths, stream capability, usage fields, and Pass-1 vs Pass-2 consistency). It returns a structured object for programmatic inspection.

What “ports retrievability” means: the Pass-2 check verifies that the JSON paths chosen in Pass-1 (for answer/thinking and streaming deltas) can be rediscovered from the standardized request. It does not guarantee semantic equivalence of outputs, nor does it attempt to reconcile vendor-specific streaming event semantics beyond path retrievability.

Safety & side-effects: this function makes live requests to the given endpoint. Avoid running in examples/tests against production APIs unless guarded with `\dontrun{}` or environment checks.

Value

Invisibly returns a list with components:

- `report`: Character vector of human-readable log lines.
- `pass1`: List with raw/effective inputs, HTTP details, ranked candidates, and selected ports (`respond_path`, `thinking_path`, `delta_path`, `thinking_delta_path`).
- `pass2`: List with HTTP details and any Pass-2 warnings.

Workflow

1. **Pass-1 probe**: send non-streaming POST; attempt SSE streaming; parse and score candidate JSON paths for ‘answer‘/‘thinking‘ (and streaming deltas). 2. **Standardize**: construct Pass-2 headers/body that preserve vendor keys, with placeholders `\${ROLE}`, `\${CONTENT}`, and a parameter merge anchor `\${PARAMETER}`; apply typed defaults. 3. **Pass-2 probe**: re-issue the request; verify that the chosen ports are still discoverable (retrievability only; not a semantic equivalence test). 4. **Registry preview & upsert**: compose a registry entry (input/output/ streaming blocks) and optionally write it into the user registry.

See Also

`probe_llm_streaming`, `score_candidates_ns`, `score_candidates_st`, `build_standardized_input`, `make_pass2_probe_inputs`, `render_pass2_path_consistency_report`, `build_registry_entry_from_analysis`, `register_endpoint_to_user_registry`, `get_registry_entry`, `llm_caller`

```
# Example: OpenAI-compatible chat endpoint
llm_register(
  url = "https://api.openai.com/v1/chat/completions",
  provider = "official",
  headers = list(
    "Content-Type" = "application/json",
    "Authorization" = "Bearer \${API_KEY}"
  ),
  body = list(
    model = "gpt-4o-mini",
    messages = list(list(role = "user", content = "\${CONTENT}")),
    stream = TRUE
  ),
  api_key = Sys.getenv("OPENAI_API_KEY"),
  content_value = "Hello from PsyLingLLM!",
  generation_interface = "chat",
  optional_defaults = list(stream = TRUE, max_tokens = 512),
  auto_register = FALSE # preview only
)
```

```
# Example: DeepSeek-style endpoint (adjust URL/model to your provider) llm_register( url =
"https://api.deepseek.com/v1/chat/completions", provider = "proxy", headers = list(
"Content-Type" = "application/json", "Authorization" = "Bearer \${API_KEY}" ), body =
list( model = "deepseek-chat", messages = list(list(role = "user", content = "\${CONTENT}")),
stream = TRUE ), api_key = Sys.getenv("DEEPSEEK_API_KEY"), generation_interface = "chat",
auto_register = TRUE )
```

---

load_registry *Load the system LLM registry*

Description

This function loads the registry configuration from the internal YAML file shipped with the **PsyLingLLM** package. The registry contains model entries, interfaces, and defaults used by functions such as `llm_register()`, `get_model_config()`, and `trial_experiment()`.

Usage

```
load_registry()
```

Value

A named list corresponding to the parsed registry YAML. Each entry is a model node (see `build_registry_entry_from_ana`
Returns `NULL` if the file does not exist or cannot be parsed.

Examples

```
reg <- load_registry()
names(reg)
```

---

make_alias *Generate a compact alias for a model id*

Description

Convert provider/model-style identifiers (e.g., "unsloth/gemma-3-4b-it", "Alibaba-NLP/Tongyi-DeepResearch-30B-A3B", "llama-3.1-70b-instruct-128k") into short, stable aliases by: 1) taking the last path segment, lowercasing, tokenizing on non-alnum, 2) removing size/context/quantization/packaging tokens, 3) (optionally) keeping version tokens (major / major+minor / full), 4) keeping meaningful mode/intent tokens (it/instruct/chat/flash/turbo/...), 5) deduplicating and joining with "-".

Usage

```
make_alias(model_id, keep_version = c("none", "major", "major_minor", "full"))
```

Arguments

model_id	Character scalar. Provider/model or plain name.
keep_version	One of c("none","major","major_minor","full"). "none" (default) drops version tokens; "major" keeps first integer only (e.g., llama-3.1 -> 3); "major_minor" keeps e.g., "3-1"; "full" keeps all.

Value

Character scalar alias (lowercase kebab-case).

Examples

```
make_alias("unsloth/gemma-3-4b-it")
#> "gemma-it"
make_alias("meta-llama/llama-3.1-70b-instruct-128k", keep_version = "major")
#> "llama-3-instruct"
make_alias("mistralai/Mixtral-8x7B-Instruct-v0.1")
#> "mixtral-instruct"
make_alias("meituan-longcat/LongCat-Flash-Thinking-FP8")
#> "longcat-flash-thinking"
make_alias("deepseek-ai/DeepSeek-R1-Distill-Qwen-32B", keep_version = "none")
#> "deepseek-r1-distill-qwen"
```

---

make_pass2_probe_inputs

Build Pass-2 EFFECTIVE probe inputs (no forced system; structural role inference)

Description

Behavior: - Never add HTTP headers automatically (no implicit Accept). - If ‘role_mapping‘ is NULL, infer roles structurally via ‘infer_role_mapping_from_body()‘ (preferred when ‘pass1_body_for_roles‘ is provided); otherwise fall back to ‘std$diagnostics$role_mapping_inferred‘ when available. If still absent, only user role defaults to "user". - ‘include_system = TRUE‘ only injects a system message when BOTH: (a) diagnostics has a non-empty ‘default_system‘, and (b) a system role label exists (from ‘role_mapping‘ or inference). Otherwise system is skipped (no fallback/forcing). - For messages style, rebuild messages strictly by schema keys from diagnostics: [optional system], then [required user]. - For single-prompt, keep the prompt key shape; only merge probe defaults. - Probe defaults are merged into the body top-level and placeholders are substituted.

Usage

```
make_pass2_probe_inputs(
  std,
  api_key,
  content = "Hello!",
  role_mapping = NULL,
```

```
  defaults = list(stream = TRUE, max_tokens = 512, temperature = 0.7),
  include_system = FALSE,
  pass1_body_for_roles = NULL
)
```

Arguments

std	list from build_standardized_input()
api_key	character(1)
content	character(1) value for "\${CONTENT}"
role_mapping	NULL or list(user=..., system=...)
defaults	named list of probe-only params (e.g., list(stream=TRUE, max_tokens=512, temperature=0.7))
include_system	logical(1), default FALSE; add system only when role exists and default_system present
pass1_body_for_roles	optional list; when provided and role_mapping is NULL, roles are inferred from this body

Value

list(headers=list, body=list, role_used=list(user=..., system=...|NULL))

---

merge_defaults_for_probe

Merge default generation parameters into a request body (probe only)

Description

This helper function inserts a set of default parameters into a model request body for *probing* or lightweight inspection, without altering persistent templates.

Usage

```
merge_defaults_for_probe(
  body,
  defaults = list(stream = TRUE, max_tokens = 512, temperature = 0.7)
)
```

Arguments

body	A list representing the request body (e.g., a Pass-2 body structure).
defaults	A named list of default parameter values to inject into body. Defaults to: `list(stream = TRUE, max_tokens = 512, temperature = 0.7)`

Details

It removes the placeholder `"\${PARAMETER}"` if present and merges predefined defaults (e.g., `stream`, `max_tokens`, `temperature`) into the top-level body structure.

Commonly used during test or probe calls to ensure a minimal, fully specified request payload before sending to an API endpoint.

Value

A modified list identical to body but with:

- the placeholder field `"\${PARAMETER}"` removed (if present);
- each entry in `defaults` merged into the top level.

`multi_model_experiment` *Run Trials Across Multiple Models (registry-driven, assistant-native)*

Description

Executes the same trial table on multiple models. Each row of ‘models‘ describes one model configuration. For each row, this function delegates to `trial_experiment()` with the appropriate arguments and then row-binds the results into a single data frame.

Usage

```
multi_model_experiment(
  models,
  data,
  trial_prompt = NULL,
  repeats = 1,
  random = FALSE,
  delay = 0,
  timeout = getOption("psylingllm.llm_timeout_sec", 120L),
  overwrite = TRUE,
  return_raw = FALSE,
  system_content = NULL,
  assistant_content,
  role_mapping,
  optionals = NULL,
  stream = NULL,
  output_dir = NULL,
  combined_output_path = NULL
)
```

Arguments

models	data.frame/tibble describing models; see "models table".
data	Trial table for trial_experiment() (must contain 'Material'; 'TrialPrompt' optional).
trial_prompt	Optional character(1). Global trial prefix; row 'TrialPrompt' (if any) overrides.
repeats	Integer(1). Number of repetitions per trial for every model.
random	Logical(1). Randomize trial order for every model.
delay	Numeric(1). Delay (seconds) between trials for every model.
timeout	Integer(1). Per-request timeout in seconds.
overwrite	Logical(1). Overwrite per-model outputs.
return_raw	Logical(1). Include raw request/response in per-call results.
system_content	Optional character(1). Fallback system message if model row lacks one.
assistant_content	Optional static few-shot seed: character vector or a list of message objects ('list(role=..., content=...)'). These appear before rolling history and are preserved as-is.
role_mapping	Optional mapping of roles if model row lacks one.
optionals	Optional named list (or NULL) if model row lacks 'optionals'. *Omitting this argument* preserves the tri-state (registry defaults).
stream	Logical(1) or NULL. Default streaming policy if model row lacks 'stream'.
output_dir	Optional character(1). If given, used as a base directory for per-model outputs when the model row doesn't provide 'output_path'.
combined_output_path	Optional character(1). If provided, write the combined results CSV here. Parent directories are created; write failures raise an error.

Details

'models' table Required column: - 'model_key' (character): registry key, e.g. "deepseek-chat" or "deepseek-chat@proxy".

Optional columns (all per-model; if absent, corresponding top-level args are used): - 'generation_interface' (character, default "chat") - 'api_key' (character) - 'api_url' (character; required for non-official providers or to override official) - 'stream' (logical or NA; 'NA'/missing means follow registry default) - 'system_content' (character or NA) - 'assistant_content' (character vector or list-of-message objects; may be JSON string) - 'optionals' (list column or JSON string; tri-state honored if column missing) - 'role_mapping' (list column or JSON string; only passed if provided) - 'output_path' (character; per-model output path for trial_experiment())

Any JSON-like character column (e.g., 'optionals', 'role_mapping', 'assistant_content') will be JSON-decoded if it validates as JSON; otherwise passed through as-is.

Optionals tri-state - If 'models$optionals' exists for that row → pass it to downstream (user keys only). - Else if top-level 'optionals' is missing → do **not** pass the param (use registry defaults). - Else if top-level 'optionals' is provided (NULL or list) → pass that top-level value.

Errors & timeouts - Each model is run independently. Errors/timeouts inside trial_experiment() are recorded per-trial; the multi-model loop continues.

Value

A data.frame/tibble concatenating all per-model results. Each chunk retains the standard PsyLingLLM schema columns and is annotated with ‘ModelKey‘ (alias of ModelName).

`normalize_history_messages`

Normalize conversation history into message objects

Description

Converts various inputs (character vectors, lists of message objects, or mixed lists) into a standardized list of message objects. This is useful for constructing conversational contexts where heterogeneous sources must be coerced into a uniform schema.

Supported inputs:

- ‘NULL‘: returns an empty list.
- character vector: each element becomes a message from the "assistant" role.
- list of message objects: preserved as-is, except roles are mapped.
- mixed list: keeps message objects; coerces string elements into assistant messages.

Usage

```
normalize_history_messages(x, map_role)
```

Arguments

`x`	One of ‘NULL‘, a character vector, or a list.
`map_role`	A function that maps role names (e.g., ‘"assistant"‘, ‘"user"‘) to the target representation. Typically ‘tolower‘ or a custom mapper.

Value

A list of standardized message objects where each element is a list with fields:

role Character scalar indicating the message role.

content Character vector or list holding the message content.

See Also

[is_message_object]

`normalize_path_key_with_regex` *Normalize any path spec to a flat key AND, if needed, a regex with wildcard index*

Description

Normalize any path spec to a flat key AND, if needed, a regex with wildcard index

Usage

```
normalize_path_key_with_regex(path)
```

Arguments

`path`	mixed (list("a","b"), "a.b", 'list("a","b")')

Value

list(key=character(1), regex=character(1) or NA)

`normalize_provider_label` *Normalize a free-form provider label*

Description

This utility normalizes a provider label or tag string into a clean, canonical form. It removes a leading ''@'' if present and trims surrounding whitespace.

Usage

```
normalize_provider_label(provider)
```

Arguments

`provider`	A character string (or coercible value) representing the provider label. Only the first element is used.

Details

Used internally when constructing registry keys or interface metadata, ensuring that providers like '"@openai"', '" chutes.ai "', or '"local"' are standardized to consistent identifiers.

Value

A character scalar containing the normalized provider label. Returns ‘"unknown"‘ if the input is ‘NULL‘ or empty.

See Also

[build_registry_entry_from_analysis()]

`normalize_type_label` *Normalize a free-form type string (strip a leading '@' if present)*

Description

Normalize a free-form type string (strip a leading '@' if present)

Usage

```
normalize_type_label(type)
```

Arguments

`type`	character(1), e.g., "chutes.ai" or "@chutes.ai"

Value

character(1) normalized type, e.g., "chutes.ai"

`probe_llm_streaming` *Probe an LLM endpoint (non-stream + streaming, SSE-aware)*

Description

Sends two probes to a model endpoint using the given `url`, `headers`, and `body`:

1. **Non-streaming POST**: captures raw response text, HTTP status, a parsed JSON object (when parseable), and any usage block.
2. **Streaming attempt (SSE)**: first tries without an `Accept` header; if no SSE markers are detected, retries with `Accept: text/event-stream`. All SSE lines are returned along with a lightweight frame of deltas and the raw parsed JSON events.

Usage

```
probe_llm_streaming(url, headers, body, stream_param, timeout = 120)
```

Arguments

`url`	Character(1). Endpoint URL.
`headers`	Named list of HTTP headers (e.g., `Content-Type`, `Authorization`). Keys must be unique.
`body`	Either a list (will be JSON-encoded with `auto_unbox = TRUE`) or a character(1) JSON string. Non-stream probe forces `stream = FALSE`; streaming probe forces `stream = TRUE`.
`stream_param`	Character scalar (optional). Name of the request-body field used by the provider to enable stream.
`timeout`	Integer(1). Per-request timeout in seconds. Applies to both non-stream and stream attempts.

Details

Internally, the function toggles common streaming keys in `body` (e.g., `stream`, `streaming`) and parses SSE frames that begin with `data:`. It does not assume any vendor schema beyond standard SSE.

SSE detection. Streaming is considered honored if any line matches `^data:\s*\{` (i.e., a JSON object after `data:`). The function first attempts without an `Accept` header and retries with `Accept: text/event-stream` only if no SSE markers are observed.

Safety. This function does not log or redact secrets. Callers are responsible for redacting/omitting API keys in logs.

Errors. Network issues and HTTP errors are returned via the `non_stream$status_code` when possible; SSE parsing is best-effort and yields empty structures if no valid frames are detected.

Value

A list with three elements:

non_stream List with fields:

- `raw_text`: character, raw HTTP body.
- `status_code`: integer HTTP status.
- `parsed`: parsed JSON object (or `NULL` if parsing fails).
- `usage`: best-effort extraction of usage block (or `NULL`).

stream_attempt List with fields:

- `requested_streaming`: logical, always `TRUE`.
- `honored_streaming`: logical, whether SSE markers were seen.
- `accept_required`: logical, whether SSE appeared only after adding `Accept: text/event-stream`.
- `reason`: short diagnostic message.
- `raw_lines`: character vector of SSE lines as received.
- `raw_df`: data frame of parsed deltas (id, model, index, role, `content`, `reasoning_content`, `finish_reason`); may be empty.
- `raw_json`: list of parsed SSE `data: {...}` payloads.

input Echo of inputs: `list(url, headers, body)`.

Typical fields in `raw_df` (when present)

- `content`: token text deltas (chat-style).
- `reasoning_content`: reasoning/think deltas (if provider emits them).
- `finish_reason`: stream finish status per choice (if emitted).

See Also

`llm_register`, `get_registry_entry`

`PsyLingLLM_Schema` *PsyLingLLM Schema (v1)*

Description

Definition of the standardized experiment output schema.

Details

PsyLingLLM Schema (v1)

This defines the standardized output format for experiments in **PsyLingLLM**, including `trial_experiment()`, `conversation_experiment()`, and `factorial_trial_experiment()`.

Core Fields

Run Integer. Global sequential run index across all trials.

Item Integer. Original item/material index.

TrialPrompt Character. Prompt shown to the model.

Material Character. Experimental stimulus or content.

Response Character. Model's final output.

Think Character or NA. Model reasoning trace (if available).

ModelName Character. Model identifier (e.g., "deepseek-chat").

Timing Metrics

TotalResponseTime Numeric (seconds). Total time from request to completion.

FirstTokenLatency Numeric (seconds). Delay until the first token arrives (only present if `streaming = TRUE`).

Token Metrics

PromptTokens Integer. Number of input tokens (NA if API does not provide).

CompletionTokens Integer. Number of output tokens (NA if API does not provide).

Diagnostics

TrialStatus Character. Either "SUCCESS" or "ERROR".

ResponseStatus Character. Response-quality classification: "OK", "EMPTY_RESPONSE", "TIMEOUT", or "ERROR". This does not replace the backward-compatible transport-oriented `TrialStatus`.

Streaming Logical. Whether SSE streaming was used.

Timestamp Character. Completion time of the run (ISO 8601).

RequestID Character or NA. Provider request identifier if returned.

RequestMessages Character (JSON). Messages sent to API for this trial, including system + prior turns + current user. Distinct from ConversationHistory, which also includes the model's reply.

ConversationHistory Character (JSON). Full conversation history including assistant replies after this trial.

`register_endpoint_offline` *Register an LLM endpoint offline (supports reasoning / thinking path)*

Description

Builds a registry entry offline — no API key or network required. Accepts standard list-style paths and automatically serializes to PsyLingLLM YAML format. Supports reasoning models (thinking_path / thinking_delta_path).

Usage

```
register_endpoint_offline(
  model,
  provider = "official",
  generation_interface = "chat",
  url,
  headers,
  body,
  role_mapping = NULL,
 optional_defaults = list(stream = list(value = TRUE, type = "logical"), max_tokens =
   list(value = 1024, type = "numeric"), temperature = list(value = 0.7, type =
   "numeric")),
  output_spec = NULL,
  streaming_spec = NULL,
  thinking_path = NULL,
  thinking_delta_path = NULL,
  default_system = NULL,
  auto_register = TRUE
)
```

Arguments

model	Character. Model key, e.g. "deepseek-reasoner".
provider	Character. "official", "proxy", or "local".
generation_interface	Character. Interface type, e.g. "chat", "responses".
url	Character. Endpoint URL.
headers	Named list of HTTP headers.
body	Named list of body fields (with placeholders).
role_mapping	Named list mapping roles; NULL stored as ~.
optional_defaults	Named list (stream, max_tokens, temperature).
output_spec	Named list of output paths (each path may be a list of strings).
streaming_spec	Named list of streaming config.
thinking_path	Optional list specifying reasoning field path in response (e.g. list("choices","message","reasoning_content
thinking_delta_path	Optional list specifying reasoning delta path in streaming (e.g. list("choices","delta","reasoning_content")
default_system	Character or NULL. Default system message; if NULL, YAML shows ~.
auto_register	Logical. TRUE to skip confirmation prompt.

Value

Invisibly returns the registry entry.

---

register_endpoint_to_user_registry

Upsert endpoint entry into the user registry (safe merge)

Description

This Registry v1 writer inserts or updates an LLM endpoint entry in a YAML-based user registry. If an entry with the same key (e.g. "deepseek-chat@chutes.ai") already exists, new generation interface nodes (e.g. ‘chat‘, ‘completion‘) are merged instead of overwriting the whole model section. Native Registry v2 bundles are rejected rather than mixed with v1 entries.

Usage

```
register_endpoint_to_user_registry(entry, path = get_registry_path())
```

Arguments

entry	A named list, typically returned by ‘build_registry_entry_from_analysis()‘.
path	Optional Registry v1 YAML path (defaults to ‘get_registry_path()‘).

Details

Interface nodes are automatically detected by structure — any list containing fields like ‘input‘, ‘output‘, ‘streaming‘, or ‘provider‘ will be treated as an interface definition.

The complete result is validated and written to a temporary file before the original is replaced. Invalid existing YAML is never silently discarded.

Value

Invisibly returns the merged registry list.

---

render_pass2_path_consistency_report

Render a structured "[Pass-2 Verified Ports]" report section.

Description

This report ONLY checks path consistency between Pass-1 and Pass-2 best picks. If all paths are consistent, the section ends with a PASSED verdict. Otherwise, it ends with a FAILED verdict and actionable guidance.

Usage

```
render_pass2_path_consistency_report(pass1_paths, pass2_paths)
```

Arguments

pass1_paths	list with fields: - ns_answer, ns_think, st_answer, st_think (character(1) or NULL)
pass2_paths	list with same fields as pass1_paths

Value

character vector of lines (ready to cat/paste)

---

save_experiment_results

Save Experiment Results to File

Description

Saves experiment results to a CSV or Excel file, conforming to PsyLingLLM_Schema. If ‘output_path‘ is NULL, a default filename is generated as `{model}_{YYYYMMDD_HHMMSS}.csv` in the ‘results‘ subdirectory of the package user-data directory returned by `tools::R_user_dir("PsyLingLLM", "data")`.

Usage

```
save_experiment_results(
  data,
  output_path = NULL,
  model = NULL,
  overwrite = TRUE,
  auto_naming = TRUE
)
```

Arguments

data	data.frame Result dataset (following PsyLingLLM_Schema).
output_path	character File path (CSV or Excel). If NULL, auto-generate model name + timestamp in default folder.
model	character Model name (used for auto-naming).
overwrite	logical Whether to overwrite existing file. Default TRUE.
auto_naming	logical If TRUE (default), and no explicit filename is provided, auto-generate filename as model_timestamp.csv.

Details

Columns that are entirely NA (e.g., FirstTokenLatency when streaming = FALSE) will be removed automatically. Parent directories are created when needed. Data are written to a temporary file before the destination is replaced, and write failures raise an error rather than returning a path to a missing file.

Value

Invisible normalized file path

---

score_candidates_ns *Rank non-streaming JSON paths for answer/think (segment-aware, conflict-guarded)*

Description

Given a parsed **non-streaming** LLM JSON response, this function scores every leaf node and returns the most probable **answer** and **think** fields. The scorer:

- flattens the JSON into (path, key, value) rows;
- counts *segment-exact* matches against lexiconnsanswer and lexiconnsthink;
- applies path boosts (e.g., choices/message/content/text → answer; reasoning/thinking/analysis → think);
- penalizes answer picks found under reasoning-like segments;
- length-normalizes and clipped-length scores;
- softmax-normalizes to probabilities and selects tops above thresholds.

Usage

```
score_candidates_ns(
  obj,
  lexicon = default_keyword_lexicon(),
  prob_thresh = list(answer = 0.6, think = 0.55),
  top_k = 10
)
```

Arguments

obj	A parsed JSON object (list) representing a non-streaming response.
lexicon	A keyword lexicon list (see `default_keyword_lexicon()`), containing `$ns$answer`, `$ns$think`, and a global `$blacklist`.
prob_thresh	Named list with numeric thresholds in `[0, 1]` for selection: `list(answer = 0.60, think = 0.55)` by default. If the top probability for a side is below its threshold, that side is returned as `NULL`.
top_k	Integer. Number of top ranked candidate rows to keep in the `candidates` data frame (after sorting by the max of the two scores).

Details

The scoring uses segment-level token hits (no substring matching) plus path-based boosts and conflict penalties. Probabilities are produced via a softmax over the respective score vectors (`answer` and `think`) independently. Thresholds gate the final picks. See also `score_candidates_st()` for streaming (SSE) candidates ranking.

Value

A list with:

`best` A list with two entries `answer` and `think`, each either `NULL` or a list: `list(path, key, text, prob, score)`. If both winners point to the same path, the `think` winner is suppressed (set to `NULL`) to avoid duplication.

`candidates` A data frame of up to `top_k` rows with columns: `path`, `key`, `value`, `hits_answer`, `hits_think`, `len_norm`, `len_clip`, `path_boost_ans`, `path_boost_think`, `conflict_reason`, `score_ns_answer`, `score_ns_think`, `prob_ns_answer`, `prob_ns_think`, `excerpt`, `rank_key`.

Conflicts & thresholds

If both best indices resolve to the same `path`, `think` is suppressed. Any side whose top probability is below its threshold is returned as `NULL`.

See Also

`score_candidates_st`, `flatten_json_paths`, `default_keyword_lexicon`

score_candidates_st *Rank streaming (SSE) candidates for answer/think (temporal aggregation, delta-aware, conflict-guarded)*

Description

Given a sequence of **streaming** SSE payloads (parsed JSON objects or a pre-flattened table), this function aggregates token deltas across events and scores candidate JSON paths for the most probable **answer** and **think** fields.

Usage

```
score_candidates_st(
  raw_df,
  raw_json,
  lexicon = default_keyword_lexicon(),
  prob_thresh = list(answer = 0.7, think = 0.55),
  top_k = 10
)
```

Arguments

raw_df	A pre-flattened data.frame of event-level paths and values, typically containing columns such as event_index, path, key, and value. May be NULL if only raw_json is supplied.
raw_json	A list of parsed SSE JSON payloads, where each element corresponds to one event. Each should be a JSON-like list structure.
lexicon	A keyword lexicon list (see default_keyword_lexicon), containing entries stanswer, stthink, and a global $blacklist.
prob_thresh	Named list of acceptance thresholds in [0,1], controlling when top candidates for each side are accepted (defaults to list(answer = 0.70, think = 0.55)).
top_k	Integer. Keep at most this many top-ranked candidates in the returned candidates table.

Details

The algorithm aggregates streaming delta events, rewarding stable token updates across time and penalizing conflicts (e.g., answer-like text inside reasoning segments). It uses early/late-event cues, lexical matching, and conflict guards. See score_candidates_ns for the non-streaming variant.

Value

A list with two elements:

best List containing answer and think entries. Each entry is either NULL or a list with fields path, key, text, prob, and score.

candidates A data frame of up to top_k rows containing event-level metrics, normalized scores, and probabilities for each candidate path.

See Also

score_candidates_ns, flatten_json_paths, default_keyword_lexicon

---

`stream_reconstruct_text` *Reconstruct stream text from raw_json using exact or wildcard-index matching Falls back to regex (for ".." wildcard) if exact match finds nothing.*

Description

Reconstruct stream text from raw_json using exact or wildcard-index matching Falls back to regex (for ".." wildcard) if exact match finds nothing.

Usage

```
stream_reconstruct_text(raw_json, path)
```

Arguments

`raw_json`	list of event JSON objects
`path`	path spec (list/"a.b"/'list("a","b")')

Value

single character concatenation ("" if none)

`strip_optionals_from_body` *Strip optional runtime parameters from a request body*

Description

This function removes optional (runtime-tunable) parameters from a raw request body to yield a clean, structural Pass-2 body. It first removes fields listed in `optional_keys` (highest priority), then applies a heuristic to detect and drop non-standard but optional-like fields (e.g., vendor variants of temperature/top_p/max_tokens etc.).

Usage

```
strip_optionals_from_body(
  body,
  optional_keys = NULL,
  preserve_vendor_fields = TRUE,
  warn = TRUE
)
```

Arguments

body	list. Raw Pass-1-like request body.
optional_keys	character(). Explicit optional keys to remove first (highest priority). If NULL, default_optional_keys() is used.
preserve_vendor_fields	logical(1). If TRUE, keep fields that look like vendor-specific extensions (e.g., names starting with vendor_ or x_).
warn	logical(1). If TRUE, emit a message summarizing removed keys.

Details

Core structural fields (e.g., model, prompt, messages, "${PARAMETER}") are preserved. Optionally, vendor extension fields (e.g., prefixed by vendor_ or x_) can be preserved via preserve_vendor_fields = TRUE.

Value

list. The cleaned body.

---

sub_placeholders *Substitute placeholders "\${API_KEY}", "\${CONTENT}", "\${ROLE}" robustly*

Description

This function performs recursive placeholder substitution within nested R structures (lists or atomic vectors). Each placeholder string of the form "\${VARNAME}" is replaced by its corresponding value from the provided mapping list.

Usage

```
sub_placeholders(x, mapping)
```

Arguments

x	A list or atomic vector containing placeholder strings.
mapping	A named list providing substitution values, e.g., list(API_KEY = "...", CONTENT = "...", ROLE = "user").

Details

Features

- Works on nested lists and atomic vectors.
- Replacement values are coerced to length-1 character strings and escaped for Perl regular expressions.
- NULL, NA, or non-scalar replacements are converted to empty strings ("").

Value

The transformed object, with all placeholders replaced by their corresponding mapped values.

`trial_experiment` *Run LLM Trial Experiment*

Description

Executes repeated trials using a registry-defined LLM interface. All request assembly (URL, headers, body/messages, defaults) is driven by the registry and calls are delegated to `llm_caller()`.

Usage

```
trial_experiment(
  model_key,
  generation_interface = "chat",
  api_key,
  api_url = NULL,
  data,
  trial_prompt = NULL,
  system_content = NULL,
  assistant_content = NULL,
  optionals = NULL,
  role_mapping = NULL,
  stream = FALSE,
  timeout = 120,
  repeats = 1,
  random = FALSE,
  delay = 0,
  output_path = NULL,
  overwrite = TRUE,
  return_raw = FALSE
)
```

Arguments

`model_key`	Character(1). Registry key (e.g., "deepseek-chat" or "deepseek-chat@proxy").
`generation_interface`	Character(1). Interface name; defaults to "chat".
`api_key`	Character(1). Provider API key.
`api_url`	Optional character(1). Overrides registry default URL; required for non-official providers.
`data`	data.frame/tibble with ‘Material‘; may contain ‘TrialPrompt‘.
`trial_prompt`	Optional character(1). Global trial prompt (used when row ‘TrialPrompt‘ is missing).

system_content	Optional character(1) or NULL. If NULL, use registry 'default_system' when available.
assistant_content	Optional static few-shot seed: character vector or a list of message objects ('list(role=..., content=...)'). These appear before rolling history and are preserved as-is.
optionals	Optional named list. The default NULL suppresses Registry defaults; a named list sends only the supplied provider-native parameters.
role_mapping	Optional list. Override registry role names (effective only when the template uses \${ROLE}).
stream	Logical(1) or NULL. NULL = use registry default; otherwise force streaming/non-streaming.
timeout	Integer(1). Per-request HTTP/SSE timeout in seconds (passed to llm_caller()).
repeats	Integer(1). Number of repetitions per trial.
random	Logical(1). Randomize trial order.
delay	Numeric(1). Delay (seconds) between trials.
output_path	Character or NULL. Where to save results/logs (compatible with prior schema).
overwrite	Logical(1). Overwrite existing output files.
return_raw	Logical(1). Include raw request/response for debugging.

Details

Columns in 'data': - Required: 'Material' - Optional: 'TrialPrompt' (row-level), which overrides the global 'trial_prompt' when present.

Error/timeout handling: - Network/SSE timeout inside llm_caller() returns status = 599 and is marked as "TIMEOUT" without stopping the loop. - Provider HTTP errors retain public status = 599, preserve the original status in error$code, and are marked as "ERROR".

Value

A data.frame/tibble with PsyLingLLM schema columns: Response, Think, ModelName, TotalResponseTime, FirstTokenLatency (if stream), PromptTokens, CompletionTokens, TrialStatus, ResponseStatus, Streaming, Timestamp, RequestID.

---

update_progress_bar *Update progress bar for an experiment*

Description

Update progress bar for an experiment

Usage

```
update_progress_bar(
  current,
  total,
  start_time,
  bar_width = 40,
  model_name = ""
)
```

Arguments

current	Number of trials completed so far
total	Total number of trials
start_time	Experiment start time
bar_width	Width of the progress bar (default 40)
model_name	Name of the model, for display purposes

Value

NULL (prints the progress bar directly to the console)

---

validate_experiment_config

Validate experiment configuration and resolve API URL

Description

Applies PsyLingLLM's registry-first rules: - Official providers may omit `api_url` if the registry defines `input.default_url`. - Non-official providers (model key contains `"@"` or provider != "official") must supply `api_url`.

Usage

```
validate_experiment_config(
  api_key,
  api_url,
  model,
  data,
  required_cols = c("Material"),
  registry = NULL,
  generation_interface = "chat"
)
```

Arguments

api_key	Character(1). Provider API key.
api_url	Character(1) or NULL. Optional for official providers.
model	Character(1). Registry key (e.g., "deepseek-chat" or "deepseek-chat@proxy").
data	data.frame/tibble. Must contain columns in required_cols.
required_cols	Character vector. Columns that must be present in data.
registry	Ignored (kept for backward compatibility).
generation_interface	Character(1). Interface to validate, default "chat".

Details

Also checks required data columns and API key presence.

Value

Invisibly returns a list with resolved fields: list(api_key, api_url, model_key, provider, generation_interface).

---

validate_registry_entry

Validate structure of a registry entry

Description

Checks if key fields exist and have correct types. Designed for both online and offline entries.

Usage

```
validate_registry_entry(entry)
```

Arguments

entry	List. Single model entry from registry.

Value

Logical TRUE if valid; otherwise prints warnings.

validate_registry_schema *Validate a PsyLingLLM Registry Document*

Description

Detects the registry version and validates its structure. Existing Registry v1 documents are accepted through a compatibility-oriented structural check. Registry v2 documents are checked strictly, including allowed fields, value types, component identifiers, and cross-references.

Usage

```
validate_registry_schema(entry)
```

Arguments

entry	A complete registry document represented as a named list.

Details

This function only validates registry data. It does not read or write files, resolve credentials, check network access, or verify that a runtime component has already been implemented.

Value

'TRUE', invisibly, when the document is valid.

Index

∗ **llm**
    llm_register, 32
∗ **registry**
    llm_register, 32
∗ **sse**
    llm_register, 32
∗ **streaming**
    llm_register, 32

adaptive_feedback_experiment, 3

build_body_pass2_structural, 6
build_registry_entry_from_analysis, 7, *34*
build_standardized_input, 8, *34*

cat_slowly, 9
classify_generation_interface, 9
common_error_blacklist, 10
compare_path_line, 11
complete_headers_for_pass2, 11
conversation_experiment, 12
conversation_experiment_with_feedback *(*adaptive_feedback_experiment*)*, 3

default_keyword_lexicon, 14, *49–51*
default_optional_keys, 14
detect_embedded_think_tag, 15

ensure_registry_header, 15
explain_alias, 16
extract_pass2_templates, 16
extract_usage_fields, 17

factorial_trial_experiment, 18
find_placeholder_path, 20
flatten_json_paths, 21, *49*, *51*
format_registration_preview, 21

generate_llm_experiment_list, 22
generate_llm_factorial_experiment_list, 22
get_model_config, 23
get_registry_entry, 24, *34*, *44*
get_registry_path, 25

has_nonempty_text, 25

infer_role_mapping_from_body, 26
infer_structure_from_placeholder, *6*, 27
inject_optional_params, 27
is_invalid_candidate, 28
is_key, 28
is_message_object, 29

list_get_by_path, 29
llm_caller, 30, *34*
llm_register, 32, *44*
load_registry, 35

make_alias, 35
make_pass2_probe_inputs, *34*, 36
merge_defaults_for_probe, 37
multi_model_experiment, 38

normalize_history_messages, 40
normalize_path_key_with_regex, 41
normalize_provider_label, 41
normalize_type_label, 42

probe_llm_streaming, *34*, 42
PsyLingLLM_Schema, 44

register_endpoint_offline, 45
register_endpoint_to_user_registry, *34*, 46
render_pass2_path_consistency_report, *34*, 47

save_experiment_results, 47
score_candidates_ns, *34*, 48, *50*, *51*

score_candidates_st, *34*, *49*, 50
stream_reconstruct_text, 51
strip_optionals_from_body, *6*, 51
sub_placeholders, 52

trial_experiment, 53

update_progress_bar, 54

validate_experiment_config, 55
validate_registry_entry, 56
validate_registry_schema, 57