Package: nalanda (via r-universe)

May 18, 2026

Title R Toolbox to answer the question: Do books really change lives?

Version 0.0.1.4

Description Provides tools and utilities for analyzing research data related to books, reading, and prosocial behavior. Named after the historic Nalanda Mahavihara, a center of learning and scholarly collaboration in ancient India.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports dplyr, ellmer, ggplot2, purrr, progress, readr, rempsyc, rlang, stringr, tibble, forestplot, jsonlite, methods, tidyr

Suggests DT, ggimage, ggtext, gridtext, knitr, rmarkdown, Rmisc, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

URL https://centerconflictcooperation.github.io/nalanda/

BugReports https://github.com/centerconflictcooperation/nalanda/issues

Remotes rempsyc/rempsyc

Config/roxygen2/version 8.0.0

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

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

Date/Publication 2026-05-18 15:39:39 UTC

RemoteUrl https://github.com/centerconflictcooperation/nalanda

RemoteRef HEAD

RemoteSha 694011b9393ddf7590f383c8ea72c0e682ccc615

Contents

`aggregate_simulations` *Aggregate simulation runs*

Description

Computes the mean and standard deviation of an outcome across simulation replicates within each model-by-unit cell. This is the recommended first step before computing inter-model agreement: it collapses intra-model sampling noise so that downstream metrics reflect genuine model differences rather than Monte Carlo variance.

Usage

```
aggregate_simulations(
  data,
  outcome = "outcome",
  by = c("model", "book_id", "chapter_id", "group")
)
```

Arguments

`data`	A data frame with one row per simulation run, containing columns for model identity, unit identifiers, and the outcome variable.
`outcome`	Character string naming the outcome column to aggregate (default `"outcome"`).
`by`	Character vector of column names to group by. Must include a column identifying the model (typically `"model"`). Default `c("model", "book_id", "chapter_id", "group")`.

Value

A tibble with one row per unique combination of `by`, plus:

mean_{outcome} Mean of `outcome` across simulation runs.

sd_{outcome} Standard deviation across runs.

n_sims Number of simulation replicates in the cell.

Examples

```
sim_data <- data.frame(
  model = rep(c("gpt-4o", "gemini-2.5-flash"), each = 40),
  book_id = rep("BookA", 80),
  chapter_id = rep(paste0("ch", 1:4), each = 10, times = 2),
  group = rep(c("Democrat", "Republican"), 40),
  sim = rep(1:10, 8),
  rating = rnorm(80, 60, 10)
)
aggregate_simulations(sim_data, outcome = "rating",
  by = c("model", "book_id", "chapter_id", "group"))
```

---

combine_book_files *Combine chapter text files into one numbered file per book*

Description

Takes chapter files named like `1_howcanyou.txt` and `2_howcanyou.txt`, groups them by the shared title stem after the underscore, orders chapters by their numeric prefix, and writes one combined `.txt` file per group using filenames like `1_howcanyou.txt`.

Usage

```
combine_book_files(
  input_dir,
  output_dir = file.path(input_dir, "combined"),
  separator = "\n\n",
  overwrite = FALSE
)
```

Arguments

`input_dir`	Character scalar. Folder containing chapter `.txt` files.
`output_dir`	Character scalar. Folder where combined `.txt` files should be written. Defaults to a `combined/` subfolder inside `input_dir`.
`separator`	Character scalar. Text inserted between chapters when combining them. Defaults to two line breaks.
`overwrite`	Logical scalar. If `TRUE`, replace existing output files. Defaults to `FALSE`.

Value

A tibble with one row per combined book and columns describing the numeric output file, original title stem, and source chapter numbers.

`combine_split_chapter_files` *Combine split chapter chunk files*

Description

Combines text files named with page/range suffixes such as `3_part1-001-050.txt`, `3_part1-051-100.txt`, and `3_part1-101-138.txt` into a single `3_part1.txt` file. Files without a trailing `-start-end` range are copied to `output_dir` when needed.

Usage

```
combine_split_chapter_files(
  input_dir,
  output_dir = input_dir,
  extension = "txt",
  separator = "\n\n",
  overwrite = FALSE,
  remove_sources = FALSE
)
```

Arguments

`input_dir`	Character scalar. Folder containing chapter `.txt` files.
`output_dir`	Character scalar. Folder where consolidated files should be written. Defaults to `input_dir`.
`extension`	Character scalar file extension to match, without a leading dot by default. Defaults to `"txt"`.
`separator`	Character scalar. Text inserted between chunks when combining them. Defaults to two line breaks.
`overwrite`	Logical scalar. If `TRUE`, replace existing output files. Defaults to `FALSE`.
`remove_sources`	Logical scalar. If `TRUE`, remove split source chunk files after all outputs are written successfully. Defaults to `FALSE`.

Value

A tibble with one row per output file and columns describing the output path, source files, and action taken.

compute_run_ai_metrics *Compute derived pre/post effect metrics from raw turn-level output*

Description

Separates post-processing from model execution so users can re-compute metrics without re-running API calls.

Usage

```
compute_run_ai_metrics(x, per_group = NULL)
```

Arguments

x	A data frame or list-like object from `run_ai_on_chapters()` with turn-level rows including `chapter`, `sim`, `identity`, `turn_type`, and `rating`. If a list is supplied, the function will attempt to combine its data-frame elements with `dplyr::bind_rows()` before computing metrics. If cumulative output from `run_ai_cumulative_chapters()` is detected, this function delegates to `compute_run_ai_metrics_cum`
per_group	Optional logical. Whether the run used per-group mode (`{group}` in question template). If `NULL` (default), mode is inferred from `target_group`: • per-group if any non-missing `target_group` values exist; • single-question if `target_group` is entirely missing.

Value

A simulation-level tibble with derived metrics (for example `pre_outgroup`, `post_outgroup`, `delta_outgroup`, and in per-group mode also `pre_ingroup`, `post_ingroup`, `pre_gap`, `post_gap`, `delta_ingroup`, `delta_gap`).

Examples

```
metrics <- compute_run_ai_metrics(toy_run_ai_turns)
head(metrics)

# The processed output can be passed on to summary and plotting helpers.
summary_by_chapter <- summarize_chapter_scores(metrics)
head(summary_by_chapter)
```

compute_run_ai_metrics_cumulative

Compute cumulative chapter metrics against the original baseline

Description

Compute cumulative chapter metrics against the original baseline

Usage

```
compute_run_ai_metrics_cumulative(x, per_group = NULL)
```

Arguments

x	A data frame or list-like object from `run_ai_cumulative_chapters()` with turn-level rows including `book`, `chapter`, `chapter_index`, `sim`, `identity`, `turn_type`, and `rating`.
per_group	Optional logical. Whether the run used per-group mode. If `NULL`, mode is inferred from `target_group`.

Value

A chapter-level tibble comparing each post-chapter state against the baseline turn from the same `book × sim × identity` conversation.

compute_run_ai_metrics_one_turn

Compute one-turn ingroup/outgroup metrics from raw output

Description

Compute one-turn ingroup/outgroup metrics from raw output

Usage

```
compute_run_ai_metrics_one_turn(x, per_group = NULL)
```

Arguments

x	A data frame or list-like object from `run_ai_on_chapters_one_turn()` with single-turn rows including `chapter`, `sim`, `identity`, and `rating`.
per_group	Optional logical. Whether the run used per-group mode. If `NULL` (default), mode is inferred from `target_group`.

Value

A simulation-level tibble with one-turn metrics. In per-group mode this includes `ingroup_rating`, `outgroup_rating`, and `gap`. In single-question mode it includes `overall_rating` and `outgroup_rating`.

`evaluate_text_analysis` *Evaluate text-analysis outputs against reference labels*

Description

This helper computes common agreement metrics used in text-analysis papers, including accuracy, macro precision/recall/F1, Spearman correlation, and weighted Cohen's kappa.

Usage

```
evaluate_text_analysis(
  data,
  truth_col,
  estimate_col,
  by = NULL,
 metric = c("accuracy", "macro_precision", "macro_recall", "macro_f1", "spearman",
    "weighted_kappa"),
  kappa_weights = c("quadratic", "linear")
)
```

Arguments

`data`	A data frame containing reference and predicted columns.
`truth_col`	Name of the reference-label column.
`estimate_col`	Name of the model-estimate column.
`by`	Optional character vector of grouping columns.
`metric`	Character vector of metrics to compute. Supported values are `"accuracy"`, `"macro_precision"`, `"macro_recall"`, `"macro_f1"`, `"spearman"`, and `"weighted_kappa"`.
`kappa_weights`	Weighting scheme for Cohen's kappa. One of `"quadratic"` (default) or `"linear"`.

Value

A tibble with one row per requested group and one column per metric.

extract_pdf_text_with_llm *Extract text from a PDF with a multimodal LLM*

Description

This helper uploads a local PDF to a multimodal model through `ellmer` and asks the model to return clean running text. It is useful when ordinary OCR struggles with stamps, overlays, or poor scan quality but the target model can read PDFs directly.

Usage

```
extract_pdf_text_with_llm(
  pdf_path,
  prompt = paste("Transcribe the main body text from this PDF as plain UTF-8 text.",
    "Keep the wording faithful to the source.",
    "Ignore repeated stamps, watermarks, page numbers, headers, footers,",
    "and other obvious non-book overlays when they are not part of the book.",
    "Preserve paragraph breaks.", "Return only the extracted text.",
    "Do not add any introduction, explanation, summary, XML, markdown fences,",
   "or labels such as 'The following is the main body text from the PDF:'.", sep = " "),
  model = "gpt-5-mini",
  integration = getOption("nalanda.integration"),
  virtual_key = getOption("nalanda.virtual_key"),
  base_url = getOption("nalanda.base_url"),
  temperature = 1,
  seed = 42,
  output_path = NULL,
  timeout_s = getOption("ellmer_timeout_s", 120),
  max_tries = getOption("ellmer_max_tries", 5),
  retry_wait = 3,
  overwrite = FALSE
)
```

Arguments

`pdf_path`	Character scalar path to a local PDF file, a character vector of PDF paths, or a named/nested list of PDF paths such as the output of `list_book_chapters(extension = "pdf")`.
`prompt`	Character scalar instruction shown alongside the PDF. The default asks for faithful transcription while ignoring obvious non-book overlays such as repeated stamps, page numbers, and headers/footers.
`model`	Character. Model name for the chat backend.
`integration`	Optional Portkey/gateway route slug. If supplied and `model` is not fully-qualified, nalanda will build `"@{integration}/{model}"`.

virtual_key	Optional legacy virtual key. If supplied and model is not fully-qualified, nalanda will build "@{virtual_key}/{model}".
base_url	Character. Base URL for API calls.
temperature	Numeric. Sampling temperature passed to the backend.
seed	Integer. Random seed for reproducibility.
output_path	Optional output target. For a single PDF, this may be either an exact .txt file path or a directory path. For a character vector or nested list of PDFs, supply a directory-like path without a file extension; nalanda will write one .txt per PDF incrementally, preserving partial progress if a later file fails.
timeout_s	Numeric scalar request timeout in seconds. Applied via options(ellmer_timeout_s = ...) for the duration of the call.
max_tries	Integer scalar total number of request attempts. Applied via options(ellmer_max_tries = ...) for the duration of the call.
retry_wait	Numeric scalar seconds to wait between manual retries after a failed single-file attempt.
overwrite	Logical scalar. If TRUE, replace existing output files at output_path. Defaults to FALSE.

Details

In testing through the NYU Portkey/gateway path, PDF extraction was more reliable with gpt-5-mini than with Gemini routes. Gemini-family models may still work in other environments, but PDF handling through chat_portkey() was inconsistent in our tests.

Value

If pdf_path is a single file, a character scalar containing the extracted text. If pdf_path is a character vector or nested list, returns text with the same structure and names as the input. If output_path is supplied, text files are also written to disk.

---

fix_text_file *Fix text file encoding and normalize punctuation/whitespace*

Description

Read a file as raw bytes, drop NUL characters, guess or use a provided source encoding, convert to UTF-8 and normalize common punctuation and newlines.

Usage

```
fix_text_file(path, from = NULL)
```

Arguments

path	Character scalar. Path to the text file.
from	Optional character. Source encoding to feed to iconv. If NULL the function will try to guess and fall back to WINDOWS-1252.

Value

A character scalar with cleaned text (UTF-8).

`interpolate_spotify_audiobook_duration` *Interpolate Spotify audiobook duration from text size*

Description

Estimate Spotify audiobook durations for new chapters or books from a reference data set where Spotify duration is already known. The predictor can be either text file size in bytes or word count. File size is often the simplest option when all chapters are plain text files created by the same workflow.

Usage

```
interpolate_spotify_audiobook_duration(
  reference,
  target = NULL,
  duration_col,
  books_path = NULL,
  target_book = NULL,
  book_col = "book",
  extension = "txt",
  reference_book_col = NULL,
  size_col = NULL,
  words_col = NULL,
  file_col = NULL,
  text_col = NULL,
  measure = c("file_size", "word_count"),
  duration_unit = c("seconds", "minutes", "hours", "hms"),
  output_unit = c("minutes", "seconds", "hours", "hms"),
  method = c("ratio", "lm")
)
```

Arguments

`reference`	Data frame with known Spotify durations and, unless `books_path` is supplied, a text-size predictor.
`target`	Data frame with chapters or books to estimate. When `books_path` and `target_book` are supplied, this can be left as `NULL`.
`duration_col`	Character scalar. Column in `reference` containing known Spotify duration.
`books_path`	Character scalar or `NULL`. Optional folder containing one subfolder per book, with chapter text files inside each book folder.
`target_book`	Character vector or `NULL`. Book folder name(s) to estimate when `books_path` is supplied.

book_col — Character scalar. Book identifier column in reference.

extension — Character scalar. File extension to read from books_path.

reference_book_col — Character scalar or NULL. Optional book identifier in reference. When supplied, the predictor is summed within each book and duration_col must contain one unique duration per book. Use this when reference rows are chapter-level but Spotify durations are book-level.

size_col — Character scalar or NULL. Column containing file sizes in bytes. Use this when measure = "file_size".

words_col — Character scalar or NULL. Column containing word counts. Use this when measure = "word_count".

file_col — Character scalar or NULL. Column containing paths to text files. If supplied with measure = "file_size", file sizes are computed with file.info(). If supplied with measure = "word_count", words are counted from the files.

text_col — Character scalar or NULL. Column containing text strings to measure directly.

measure — Character scalar. Either "file_size" or "word_count".

duration_unit — Unit of duration_col: "seconds", "minutes", "hours", or "hms" for strings like "6:11:00".

output_unit — Unit for the returned estimate column. Use "hms" for spreadsheet-friendly strings like "5:56:00".

method — Estimation method. "ratio" fits a single seconds-per-unit rate through the origin. "lm" fits a linear model with an intercept.

Value

A tibble containing target plus .duration_seconds, an estimated_duration_* column in output_unit, .duration_measure, and .duration_method. The total estimated duration is also stored in the estimated_total_seconds and estimated_total_* attributes.

Examples

```
reference <- tibble::tibble(
  book = c("A", "B"),
  file_size_bytes = c(100000, 150000),
  spotify_duration_minutes = c(120, 180)
)

chapters <- tibble::tibble(
  chapter = c("chapter_1", "chapter_2"),
  file_size_bytes = c(25000, 50000)
)

interpolate_spotify_audiobook_duration(
  reference,
  chapters,
  duration_col = "spotify_duration_minutes",
  size_col = "file_size_bytes",
  duration_unit = "minutes"
)
```

list_book_chapters *List book chapter files inside a books directory*

Description

Given either a path to a directory of book folders or a single folder that directly contains chapter files, return a named list where each element is a character vector of chapter file paths (ordered by number or name).

Usage

```
list_book_chapters(books_path = "books", extension = "txt")
```

Arguments

books_path	Character scalar. Path containing subdirectories for each book (default "books").
extension	Character scalar file extension to match, without a leading dot by default. Defaults to "txt".

Value

A named list of character vectors of file paths.

make_annotation_prompt *Build a numeric-response prompt for text analysis*

Description

This helper creates prompts in the same style used by Rathje et al. (2024): a direct question, followed by a numeric response instruction, followed by the text placeholder. The returned prompt is a template and may include placeholders such as {text} or {language} that are expanded later by run_text_analysis().

Usage

```
make_annotation_prompt(
  question,
  labels = NULL,
  scale = NULL,
  anchors = NULL,
  text_label = "Here is the text:",
  text_placeholder = "{text}"
)
```

Arguments

question	Character scalar question shown before the response instructions.
labels	Optional character vector of class labels in numeric order. For example, c("positive", "neutral", "negative").
scale	Optional numeric vector of length 2 giving the response scale range, such as c(1, 7).
anchors	Optional character vector of length 2 giving the low and high anchor labels used with scale.
text_label	Character scalar introducing the text block.
text_placeholder	Character scalar placeholder to insert where the text should appear.

Value

A character scalar prompt template.

---

make_baseline_prompt *Build the baseline (Turn 1) prompt*

Description

Constructs the prompt: identity context + question(s). If the question template contains {group}, it is expanded once per group (ingroup first). Otherwise, the question is used as-is (single-question mode).

Usage

```
make_baseline_prompt(
  identity_context,
  question_template,
  groups,
  identity_label
)
```

Arguments

identity_context	Character scalar. The full context string for this identity.
question_template	Character scalar. Optionally contains {group} placeholder for per-group expansion.
groups	Character vector of all group labels.
identity_label	Character scalar. The group label assigned as identity (used to determine ingroup-first ordering).

Value

Character scalar prompt.

Examples

```
# Per-group mode (asks about each group, ingroup first):
make_baseline_prompt(
  identity_context = "You are simulating an American Democrat.",
  question_template = "How warmly (0-100) do you feel towards {group}s?",
  groups = c("Democrat", "Republican"),
  identity_label = "Democrat"
)

# Single-question mode (asks once, as-is):
make_baseline_prompt(
  identity_context = "You are simulating an American Democrat.",
  question_template = "How warmly (0-100) do you feel towards your outgroup?",
  groups = c("Democrat", "Republican"),
  identity_label = "Democrat"
)
```

---

make_post_prompt *Build the post-intervention (Turn 2) prompt*

Description

Constructs the prompt: material text + question(s). If the question template contains `{group}`, it is expanded once per group (ingroup first). Otherwise, the question is used as-is (single-question mode).

Usage

```
make_post_prompt(chapter_text, question_template, groups, identity_label)
```

Arguments

`chapter_text`	Character scalar. The full material text.
`question_template`	Character scalar. Optionally contains `{group}` placeholder for per-group expansion.
`groups`	Character vector of all group labels.
`identity_label`	Character scalar. The group label assigned as identity.

Value

Character scalar prompt.

Examples

```
# Per-group mode:
make_post_prompt(
  chapter_text = "This is a chapter about cooperation...",
  question_template = "How warmly (0-100) do you feel towards {group}s?",
  groups = c("Democrat", "Republican"),
  identity_label = "Democrat"
)

# Single-question mode:
make_post_prompt(
  chapter_text = "This is a chapter about cooperation...",
  question_template = "How warmly (0-100) do you feel towards your outgroup?",
  groups = c("Democrat", "Republican"),
  identity_label = "Democrat"
)
```

---

make_treatment_prompt *Build a concrete prompt for* simulate_treatment()

Description

This helper expands a single simulate_treatment() prompt template into a concrete prompt string. It is useful for inspecting prompt wording before launching a run, much like make_baseline_prompt() and make_post_prompt() are useful for run_ai_on_chapters().

Usage

```
make_treatment_prompt(
  prompt_template,
  intervention_text,
  identity_context = "",
  identity_label = NA_character_
)

build_simulate_treatment_prompt(
  prompt_template,
  intervention_text,
  identity_context = "",
  identity_label = NA_character_
)
```

Arguments

prompt_template
: Character scalar. A single prompt template that may include {intervention_text}, {identity}, and {group}.

`intervention_text`
: Character scalar. The intervention text to insert into `{intervention_text}`.

`identity_context`
: Character scalar. Optional identity context to prepend to the prompt.

`identity_label`
: Character scalar. Optional identity label used to expand `{identity}` and `{group}`.

Value

A character scalar containing the concrete prompt.

Examples

```
make_treatment_prompt(
  prompt_template = "{intervention_text}\n\nRate this as {identity}.",
  intervention_text = "A short climate message.",
  identity_context = "You are simulating an American adult.",
  identity_label = "American"
)
```

---

model_agreement *Compute inter-model agreement*

Description

Quantifies how consistently different AI models score the same units using ICC(2,1) (intraclass correlation, absolute agreement) and/or Kendall's W (coefficient of concordance). Models produce continuous scores on a 1–100 scale; this function operates on those raw scores (typically after aggregating simulation runs via `aggregate_simulations()`).

Usage

```
model_agreement(
  data,
  outcome = "mean_outcome",
  unit_by = c("book_id", "chapter_id", "group"),
  group_by = NULL,
  model_col = "model",
  metrics = c("icc", "kendall_w")
)
```

Arguments

`data`	A data frame with one row per model-by-unit combination.
`outcome`	Character string naming the score column (default `"mean_outcome"`).
`unit_by`	Character vector of columns that jointly identify a unit (default `c("book_id", "chapter_id", "group")`).

group_by	Optional character vector. If provided, agreement metrics are computed separately within each level of these columns (e.g., "group" to get separate estimates for Democrats and Republicans).
model_col	Character string naming the model column (default "model").
metrics	Character vector of metrics to compute. One or both of "icc" and "kendall_w" (default both).

Details

Each model is treated as a **rater** and each unique combination of unit_by columns as a **target**. ICC captures agreement in both level and rank order; Kendall's W converts the continuous scores to ranks internally and assesses rank-order concordance only.

Which metric to report?:

- **ICC(2,1)** is the primary recommendation for continuous scores. It penalises models that systematically differ in level **and** in rank ordering. Interpret with Cicchetti (1994) cut-offs: < .40 poor, .40–.59 fair, .60–.74 good, >= .75 excellent.
- **Kendall's W** converts the continuous 1–100 scores to ranks and asks only whether models rank the units the same way. Useful when the absolute scale is arbitrary or when the researcher cares about ordinal agreement (e.g., "which book scored highest?") rather than exact score match.

For a quick "single consistency score," report ICC. Add Kendall's W as a supplementary rank-agreement check.

Aggregation guidance:

Always aggregate simulation runs first via aggregate_simulations(). Failing to do so inflates *n* and distorts agreement estimates.

Units with missing scores for one or more models are excluded from ICC and Kendall's W because agreement metrics require the same units to be scored by all raters. The reported n_units is the number of complete units used in the calculation.

Value

A tibble with columns: any group_by columns, plus

metric "icc" or "kendall_w".

value The agreement statistic (0–1 scale).

interpretation Qualitative label (e.g., "good", "moderate").

n_models Number of models (raters).

n_units Number of units (targets).

p_value p-value for the statistic (F-test for ICC, chi-squared approximation for Kendall's W).

Examples

```
## Not run:
# After aggregating simulations
agg <- aggregate_simulations(sim_data, outcome = "rating",
  by = c("model", "book_id", "chapter_id", "group"))

# Overall agreement
model_agreement(agg, outcome = "mean_rating",
  unit_by = c("book_id", "chapter_id", "group"))

# Agreement by political group
model_agreement(agg, outcome = "mean_rating",
  unit_by = c("book_id", "chapter_id"),
  group_by = "group")

## End(Not run)
```

`model_agreement_sensitivity` *Summarize model agreement across analysis levels*

Description

Builds a slide-ready sensitivity table by running `model_agreement()` across several substantively useful unit definitions (for example book-level, chapter-level, and party-specific agreement). Lower-level rows are aggregated with an NA-safe mean before each agreement calculation, so skipped chapters do not turn an entire book mean into `NA`.

Usage

```
model_agreement_sensitivity(
  data,
  outcome = "mean_outcome",
  model_col = "model",
  analyses = NULL,
  metrics = c("icc", "kendall_w"),
  format = c("wide", "long"),
  digits = 2,
  drop_missing = TRUE
)
```

Arguments

`data`	A data frame with one row per model-by-unit combination.
`outcome`	Character string naming the score column (default `"mean_outcome"`).
`model_col`	Character string naming the model column (default `"model"`).

analyses	Optional named list defining analyses to run. Each element should be a list with unit_by and, optionally, group_by. If NULL, a default set is inferred from available book, chapter, and party/group columns.
metrics	Character vector of metrics to compute. One or both of "icc" and "kendall_w" (default both).
format	Character. "wide" returns one row per analysis/subgroup with ICC and Kendall's W side by side; "long" returns the stacked model_agreement() results with analysis labels.
digits	Integer. Number of decimal places used in the formatted wide table.
drop_missing	Logical. Whether to drop rows with missing model, unit, or grouping identifiers before computing each analysis (default TRUE).

Value

A tibble. With format = "wide", columns include Analysis level, Subgroup, N models, N units, ICC, and Kendall's W.

Examples

```
## Not run:
model_agreement_sensitivity(
  agg,
  outcome = "mean_delta_gap",
  model_col = "model"
)

## End(Not run)
```

---

model_pairwise_cor *Pairwise model correlations*

Description

Computes Pearson and/or Spearman correlations between every pair of models on a shared set of units. This is a diagnostic complement to the omnibus metrics in model_agreement(): it reveals *which* models diverge.

Usage

```
model_pairwise_cor(
  data,
  outcome = "mean_outcome",
  unit_by = c("book_id", "chapter_id", "group"),
  group_by = NULL,
  model_col = "model",
  methods = c("pearson", "spearman")
)
```

Arguments

data	A data frame with one row per model-by-unit combination.
outcome	Character string naming the score column (default "mean_outcome").
unit_by	Character vector of columns that jointly identify a unit (default c("book_id", "chapter_id", "group")).
group_by	Optional character vector. If provided, agreement metrics are computed separately within each level of these columns (e.g., "group" to get separate estimates for Democrats and Republicans).
model_col	Character string naming the model column (default "model").
methods	Character vector of correlation types. One or both of "pearson" and "spearman" (default both).

Value

A tibble with columns: any group_by columns, plus model_a, model_b, method, correlation, and n_units.

Examples

```
## Not run:
pw <- model_pairwise_cor(agg, outcome = "mean_rating",
  unit_by = c("book_id", "chapter_id", "group"))
plot_model_agreement(pw, type = "heatmap")

## End(Not run)
```

---

model_rank_consistency

Compare model-derived rankings

Description

Takes each model's continuous scores (1–100 scale) and derives rankings from them, then evaluates cross-model concordance via Kendall's W. The rankings are computed by the researcher from the raw scores — models themselves only produce continuous ratings, not ordinal ranks. Useful for answering "Do models rank books the same way?"

Usage

```
model_rank_consistency(
  data,
  outcome = "mean_outcome",
  unit_by = c("book_id", "chapter_id"),
  rank_within = NULL,
  model_col = "model"
)
```

Arguments

data	A data frame with one row per model-by-unit combination.
outcome	Character string naming the score column (default "mean_outcome").
unit_by	Character vector of columns that jointly identify a unit (default c("book_id", "chapter_id", "group")).
rank_within	Optional character vector of columns that define separate ranking contexts (e.g., "group"). Items are ranked independently within each combination of these columns.
model_col	Character string naming the model column (default "model").

Details

unit_by must identify exactly one row per model within each ranking context. If the data still contain lower-level rows (for example, chapters) and you want book-level ranks, aggregate those rows to the book level before calling this function.

Units with missing scores for one or more models are excluded from the concordance calculation. The reported n_items is the number of complete items used.

Value

A list with two elements:

ranks Tibble with the unit columns, model, score, and rank.

concordance Tibble with Kendall's W and associated statistics, one row per rank_within combination (or one row total).

Examples

```
## Not run:
rc <- model_rank_consistency(agg, outcome = "mean_rating",
  unit_by = c("book_id", "chapter_id"),
  rank_within = "group")
rc$concordance
rc$ranks

agg_book <- agg |>
  dplyr::group_by(model, book_id, group) |>
  dplyr::summarise(mean_rating = mean(mean_rating), .groups = "drop")

rc_book <- model_rank_consistency(agg_book, outcome = "mean_rating",
  unit_by = "book_id",
  rank_within = "group")

## End(Not run)
```

nalanda *A Random Historical Fact About Nalanda University*

Description

`nalanda()` returns a neutral, research-friendly fun fact about the ancient Nalanda University. The goal is simply to provide a small, informative piece of historical context with no evaluative or cultural interpretation.

Usage

```
nalanda()
```

Value

A character string containing one factual statement about Nalanda.

Examples

```
nalanda()
```

pairwise_for_level *Pairwise model correlations at a chosen analysis level*

Description

Aggregates lower-level rows to the requested unit level, then calls `model_pairwise_cor()`. This is a convenience wrapper for cases where data still contain chapter, party, or simulation-detail rows but the researcher wants correlations at a broader level, such as book-level correlations.

Usage

```
pairwise_for_level(
  data,
  outcome = "mean_outcome",
  unit_by = c("book_id", "chapter_id", "group"),
  group_by = NULL,
  model_col = "model",
  methods = c("pearson", "spearman"),
  drop_missing = TRUE
)
```

Arguments

data	A data frame with one row per model-by-unit combination.
outcome	Character string naming the score column (default "mean_outcome").
unit_by	Character vector of columns that jointly identify a unit (default c("book_id", "chapter_id", "group")).
group_by	Optional character vector. If provided, agreement metrics are computed separately within each level of these columns (e.g., "group" to get separate estimates for Democrats and Republicans).
model_col	Character string naming the model column (default "model").
methods	Character vector of correlation types. One or both of "pearson" and "spearman" (default both).
drop_missing	Logical. Whether to drop rows with missing model, unit, or grouping identifiers before aggregating (default TRUE).

Value

Output of model_pairwise_cor() for the requested level.

Examples

```
## Not run:
# Book-level pairwise correlations from chapter-party-level aggregated data
pw_book <- pairwise_for_level(
  agg,
  outcome = "mean_delta_gap",
  unit_by = "book",
  model_col = "model",
  methods = "pearson"
)

summarize_model_correlations(pw_book, method = "pearson")

## End(Not run)
```

---

plot_chapter_scores_faceted

Faceted plot of chapter scores

Description

Create a faceted plot (one facet per book) showing mean scores and error bars.

Usage

```
plot_chapter_scores_faceted(
  summary_df,
  dv = "post_outgroup",
  y_label = "Simulated scores"
)
```

Arguments

summary_df	Data frame produced by summarize_chapter_scores().
dv	Character. Column name prefix for mean and sd. For example, "post_outgroup" will plot mean_post_outgroup ± sd_post_outgroup.
y_label	Character string for y-axis label.

Value

A ggplot2 object.

Examples

```
chapter_summary <- summarize_chapter_scores(toy_sim_results)
plot_chapter_scores_faceted(
  chapter_summary,
  dv = "delta_outgroup",
  y_label = "Mean outgroup change"
)
```

---

plot_chapter_trajectories

Plot chapter trajectories by book

Description

Simple line plot of mean simulated outgroup rating across chapter order for each book.

Usage

```
plot_chapter_trajectories(
  summary_df,
  dv = "mean_post_outgroup",
  y_label = "Simulated scores"
)
```

Arguments

summary_df	A data frame produced by summarize_chapter_scores() with columns chapter_index, mean_post_outgroup, and book.
dv	Character. Column name to plot on the y-axis. Defaults to "mean_post_outgroup".
y_label	Character. Y-axis label.

Value

A ggplot2 object.

Examples

```
chapter_summary <- summarize_chapter_scores(toy_sim_results)
plot_chapter_trajectories(
  chapter_summary,
  dv = "mean_delta_gap",
  y_label = "Mean gap change"
)
```

---

plot_chapters_over_time

Plot chapters over time (multi-timepoint means)

Description

Create a plot showing means over chapter timepoints using rempsyc::plot_means_over_time for the wide-format response variables.

Usage

```
plot_chapters_over_time(
  chapters,
  dv = "delta_gap",
  group = "book",
  x_label = "Chapter",
  y_label = "Simulated scores",
  plot_title = NULL,
  plot_subtitle = "",
  append_model_info = TRUE,
  ci_type = "between",
  legend.position = "bottom",
  groups.order = "decreasing",
  text_size = 20,
  line_width = 3,
  point_size = 4,
  reverse_score = FALSE,
  error_bars = TRUE,
```

```
  neutrality_line = TRUE,
  point_images = NULL,
  image_size = 0.04,
  image_nudge_x = 0,
  image_nudge_y = 0,
  image_jitter_width = 0,
  image_jitter_height = 0,
  facet = NULL,
  facet_ncol = NULL,
  facets.order = "increasing"
)
```

Arguments

chapters	A data frame or list of processed simulation rows, typically returned by compute_run_ai_metrics(), containing columns book, chapter, and the desired dv. If a list is supplied, the function will attempt to combine its data-frame elements before plotting.
dv	Character. Name of the column to plot as the dependent variable (default: "pre_post_outgroup_difference").
group	The group by which to plot the variable
x_label	Character. X-axis label.
y_label	Character. Y-axis label.
plot_title	Optional character title. If NULL (default) or FALSE, no title is added.
plot_subtitle	Optional plot subtitle.
append_model_info	Logical. If TRUE (default), append model and temperature attributes to the subtitle (or create one if none is provided).
ci_type	Character. Type of confidence interval to pass to rempsyc::plot_means_over_time.
legend.position	Position for legend.
groups.order	Specifies the desired display order of the groups on the legend. Either provide the levels directly, or a string: "increasing" or "decreasing", to order based on the average value of the variable on the y axis, or "string.length", to order from the shortest to the longest string (useful when working with long string names). "Defaults to "decreasing".
text_size	Numeric. Base text size for axis/title text.
line_width	Numeric. Line thickness used in geom_line(). Defaults to 3. Can be reduced for publication figures or increased for presentation slides.
point_size	Numeric. Point size used in geom_point(). Defaults to 4. Adjust to improve readability depending on output format.
reverse_score	Logical. Whether to reverse score scale.
error_bars	Logical. Show error bars.
neutrality_line	Logical. Add a horizontal neutrality line at 50.

point_images	Optional named list mapping group levels to image file paths (PNG recommended). When supplied, the point markers are replaced with the corresponding images, and the legend labels are updated to show the matching image alongside the group name when ggtext is installed. Example: list(Democrat = "logos/dem.png", Republican = "logos/rep.png").
image_size	Numeric. Size of images when point_images is used. Passed to ggimage::geom_image(). Defaults to 0.04.
image_nudge_x	Numeric. Horizontal offset applied to point images only. Defaults to 0.
image_nudge_y	Numeric. Vertical offset applied to point images only. Defaults to 0.
image_jitter_width	Numeric. Horizontal jitter width applied to point images only. Defaults to 0.
image_jitter_height	Numeric. Vertical jitter height applied to point images only. Defaults to 0.
facet	The variable by which to facet grid.
facet_ncol	Optional numeric value passed to ggplot2::facet_wrap() as ncol when facet is supplied.
facets.order	Specifies the desired display order of facet panels. Either provide the levels directly, or a string: "increasing" or "decreasing", to order panels based on the average value of the y variable, or "string.length" to order panels by facet label length. Defaults to "increasing".

Value

A ggplot2 object.

Examples

```
plot_chapters_over_time(
  toy_sim_results,
  dv = "delta_outgroup",
  group = "party",
  facet = "book",
  y_label = "Outgroup change"
)
```

---

plot_chapters_over_time_one_turn

Plot chapter trajectories for one-turn simulations

Description

Plot chapter trajectories for one-turn simulations

Usage

```
plot_chapters_over_time_one_turn(chapters, dv = "outgroup_rating", ...)
```

Arguments

chapters	Raw output from run_ai_on_chapters_one_turn() or processed output from compute_run_ai_metrics_one_turn().
dv	Character. Name of the metric to plot. Defaults to "outgroup_rating".
...	Additional arguments passed to plot_chapters_over_time().

Value

A ggplot2 object.

---

plot_forest_books *Create a forest plot of book-level polarization reduction effects*

Description

Generates a forest plot displaying mean reduction in affective polarization across books, including 95% confidence intervals.

Usage

```
plot_forest_books(
  forest_df,
  dv = "delta_gap",
  add_ci_label = TRUE,
  digits = 2,
  label_cols = c("book"),
  show_ci_label = TRUE,
  ci_multiline = TRUE,
  ci_show_party = FALSE,
  show_legend = TRUE,
  ci_label_fontsize = NULL,
  ci_label_lineheight = 0.85,
  header = NULL,
  title = "",
  xlab = "",
  xticks = NULL,
  xticks.digits = NULL,
  zero = NA,
  show_overall = TRUE,
  ci.vertices = FALSE
)
```

Arguments

forest_df	Either: • A data frame produced by prepare_forest_books() containing book, mean, lower, and upper, or • A summary data frame (for example from summarize_chapter_scores(..., aggregate_level = "book")) with mean_{dv} and sd_{dv} columns.
dv	Character. Variable prefix used when forest_df is not already prepared. Defaults to "delta_gap".
add_ci_label	Logical. Passed to prepare_forest_books() when internal preparation is needed. Defaults to TRUE.
digits	Integer. Passed to prepare_forest_books() when internal preparation is needed. Defaults to 2.
label_cols	Character vector of left-side label columns. Defaults to "book".
show_ci_label	Logical. If TRUE, appends an internally generated ci label column to the right-side text.
ci_multiline	Logical. For grouped party output, print each party CI on its own line (using \n) when TRUE.
ci_show_party	Logical. Include party names in CI labels.
show_legend	Logical. Show party legend when grouped data are present.
ci_label_fontsize	Optional numeric size for the CI label column. Useful when grouped party CIs are shown on multiple lines.
ci_label_lineheight	Numeric line height for the CI label column when ci_label_fontsize is set.
header	Labels of the columns to be displayed and as specified in label_cols.
title	Plot title
xlab	X-axis label
xticks	Optional numeric vector of x-axis tick positions. Defaults to NULL, in which case readable pretty breaks are computed automatically.
xticks.digits	Integer number of digits for x-axis tick labels. Defaults to NULL, in which case it is inferred from the tick positions.
zero	Numeric scalar, NA, or NULL. Reference line position for forestplot. Defaults to NA (no zero/reference line). NULL is treated as NA for convenience.
show_overall	Logical. If TRUE (default), a vertical dashed line indicating the overall mean effect is added.
ci.vertices	Logical. Whether to draw CI vertices in the forest plot.

Details

Books are ordered from strongest to weakest mean effect.

The plot uses circular markers for point estimates and displays 95% confidence intervals.

The vertical dashed line (if enabled) represents the average effect across books.

The temporary mean/lower/upper columns required by forestplot are generated internally when needed.

If party is present, estimates are drawn as multiple CIs per book row (one row per book; one estimate per party).

Value

A forestplot grob object.

Examples

```
book_summary <- summarize_chapter_scores(
  toy_sim_results,
  aggregate_level = "book"
)
forest_df <- prepare_forest_books(book_summary, dv = "delta_gap")
plot_forest_books(
  forest_df,
  xlab = "Reduction in polarization gap",
  show_overall = FALSE
)
```

---

plot_model_agreement *Plot inter-model agreement*

Description

Creates diagnostic visualizations for model agreement or pairwise correlation results.

Usage

```
plot_model_agreement(data, type = c("metrics", "heatmap"), method = NULL)
```

Arguments

data	Output of model_agreement() (for type = "metrics") or model_pairwise_cor() (for type = "heatmap").
type	Character. "metrics" for a dot plot of agreement statistics; "heatmap" for a pairwise correlation tile plot.
method	Character. Correlation method to plot when type = "heatmap": "spearman" for rank correlations or "pearson" for linear correlations on the continuous scores. If NULL (default), Spearman is used when available, otherwise Pearson is used.

Value

A ggplot2 object.

Examples

```
## Not run:
plot_model_agreement(model_agreement(agg, outcome = "mean_rating"),
  type = "metrics")
plot_model_agreement(model_pairwise_cor(agg, outcome = "mean_rating"),
  type = "heatmap")
plot_model_agreement(model_pairwise_cor(agg, outcome = "mean_rating"),
  type = "heatmap", method = "pearson")

## End(Not run)
```

---

plot_top_unit_heatmap *Plot model-by-unit rank heatmap*

Description

Creates a heatmap from the `ranks` element returned by `summarize_top_units(..., include_ranks = TRUE)`. Rows are units, columns are models, and cells show each model's rank for that unit.

Usage

```
plot_top_unit_heatmap(
  data,
  item_col = NULL,
  model_col = "model",
  facet_by = NULL,
  top_n_items = NULL,
  item_labels = NULL,
  show_values = TRUE,
  title = "Unit ranks by model"
)
```

Arguments

`data`	The `ranks` data frame from `summarize_top_units()` with `include_ranks = TRUE`.
`item_col`	Character. Column identifying the ranked item. If `NULL`, the function tries to infer `"book"` or `"book_id"`.
`model_col`	Character. Column identifying the model (default `"model"`).
`facet_by`	Optional character vector of columns to facet by, e.g. `"party"`.
`top_n_items`	Optional integer. If supplied, keep only the best `top_n_items` per facet, based on average rank across models.
`item_labels`	Optional character vector for display labels. Use a named vector to map item IDs to labels, e.g. `c("1" = "Opening chapter")`. For chapter-like IDs that start with numbers, an unnamed vector is matched by chapter number.
`show_values`	Logical. If `TRUE` (default), print rank values in cells.
`title`	Optional plot title.

Value

A ggplot2 object.

Examples

```
## Not run:
top_books <- summarize_top_units(
  agg,
  outcome = "mean_delta_gap",
  item_by = "book",
  rank_within = "party",
  include_ranks = TRUE
)
plot_top_unit_heatmap(top_books$ranks, item_col = "book", facet_by = "party")

## End(Not run)
```

---

plot_top_unit_pairs *Plot paired subgroup ranks for top units*

Description

Creates a connected Cleveland dot plot from summarize_top_units() output when rankings were computed within a two-level subgroup, such as party. Each row is an item, dots show subgroup-specific mean ranks, and connecting lines show how much the ranking differs between subgroups.

Usage

```
plot_top_unit_pairs(
  data,
  item_col = NULL,
  subgroup_col = "party",
  top_n_items = NULL,
  item_labels = NULL,
  subgroup_order = NULL,
  title = "Paired subgroup ranks",
  x_breaks = NULL,
  x_limits = NULL
)
```

Arguments

data	Output of summarize_top_units() with a subgroup column such as "party".
item_col	Character. Column identifying the ranked item. If NULL, the function tries to infer "book" or "book_id".
subgroup_col	Character. Two-level subgroup column to connect, e.g. "party".

top_n_items	Optional integer. If supplied, keep only items with the best average mean_rank across subgroups.
item_labels	Optional character vector for display labels. Use a named vector to map item IDs to labels, e.g. c("1" = "Opening chapter"). For chapter-like IDs that start with numbers, an unnamed vector is matched by chapter number.
subgroup_order	Optional character vector giving the two subgroup levels in display order.
title	Optional plot title.
x_breaks	Optional numeric vector of x-axis breaks. If NULL, integer rank breaks are shown by default.
x_limits	Optional numeric vector of length 2. If NULL, limits are chosen from the displayed mean ranks.

Value

A ggplot2 object.

Examples

```
## Not run:
top_books_party <- summarize_top_units(
  agg,
  outcome = "mean_delta_gap",
  item_by = "book",
  rank_within = "party"
)
plot_top_unit_pairs(top_books_party, item_col = "book", subgroup_col = "party")

## End(Not run)
```

---

plot_top_units *Plot units that rank consistently high across models*

Description

Creates a dot plot from summarize_top_units() output. Units are ordered by average rank across models, point size shows the mean score, and text labels show how many models placed the unit in the top N.

Usage

```
plot_top_units(
  data,
  item_col = NULL,
  facet_by = NULL,
  top_n_items = NULL,
  item_labels = NULL,
  title = "Units most consistently ranked highest",
```

```
  x_breaks = NULL,
  x_limits = NULL,
  caption = NULL,
  show_top_n_label = TRUE
)
```

Arguments

data	Output of summarize_top_units().
item_col	Character. Column identifying the ranked item. If NULL, the function tries to infer "book" or "book_id".
facet_by	Optional character vector of columns to facet by, e.g. "party".
top_n_items	Optional integer. If supplied, keep only the best top_n_items per facet, based on mean_rank.
item_labels	Optional character vector for display labels. Use a named vector to map item IDs to labels, e.g. c("1" = "Opening chapter"). For chapter-like IDs that start with numbers, an unnamed vector is matched by chapter number.
title	Optional plot title.
x_breaks	Optional numeric vector of x-axis breaks. If NULL, integer rank breaks are shown by default.
x_limits	Optional numeric vector of length 2. If NULL, limits are chosen from the displayed mean ranks.
caption	Optional plot caption. If NULL, a caption explaining the point-size and top-N label encodings is generated when possible.
show_top_n_label	Logical. If TRUE (default), label points with the number of models placing the item in the top N.

Value

A ggplot2 object.

Examples

```
## Not run:
top_books <- summarize_top_units(
  agg,
  outcome = "mean_delta_gap",
  item_by = "book",
  rank_within = "party"
)
plot_top_units(top_books, item_col = "book", facet_by = "party")

## End(Not run)
```

`prepare_forest_books` *Prepare book-level data for forest plotting*

Description

Computes standard errors and 95% confidence intervals for book-level estimates. This function assumes the input is already aggregated at the book level (e.g., using `summarize_chapter_scores(..., aggregate_level = "book")`).

Usage

```
prepare_forest_books(
  summary_books,
  dv = "delta_gap",
  add_ci_label = TRUE,
  digits = 2
)
```

Arguments

`summary_books`	A data frame containing at least `book`, `sim`, and mean/sd columns matching the `dv` prefix pattern.
`dv`	Character. The variable prefix to use for the forest plot. Defaults to `"delta_gap"`. The function looks for `mean_{dv}` and `sd_{dv}` columns.
`add_ci_label`	Logical. Default is TRUE. If TRUE, a formatted character column `ci` is added containing the estimate and its 95% confidence interval in the format: `mean [lower, upper]`. If FALSE, only numeric columns (`mean`, `se`, `lower`, `upper`) are returned.
`digits`	Integer. Default is 2. Number of decimal places used when formatting the `ci` column. Ignored if `add_ci_label = FALSE`.

Details

Standard errors are computed as `sd / sqrt(sim)`. Confidence intervals are calculated using a normal approximation (`mean +/- 1.96 * SE`). When a summary row has one observation and the standard deviation is therefore missing, the CI is collapsed to the point estimate.

Value

A tibble with added columns:

mean Mean effect.

se Standard error of the mean.

lower Lower bound of the 95% CI.

upper Upper bound of the 95% CI.

Examples

```
book_summary <- summarize_chapter_scores(
  toy_sim_results,
  aggregate_level = "book"
)
prepare_forest_books(book_summary, dv = "delta_gap")
```

`rank_weighted` *Rank rows using a weighted rubric*

Description

Compute a weighted score across selected numeric columns and return the input data with a final `weighted_score`, sorted by score.

Usage

```
rank_weighted(
  data,
  weights,
  normalize = TRUE,
  decreasing = TRUE,
  na_rm = FALSE
)
```

Arguments

`data`	A data frame.
`weights`	Named numeric vector of weights. Names must match columns in `data`, and weights must sum to 1.
`normalize`	Logical. If `TRUE` (default), selected variables are scaled to `[0, 1]` using min-max normalization before weighting.
`decreasing`	Logical. If `TRUE` (default), rows are sorted from highest to lowest `weighted_score`.
`na_rm`	Logical. If `TRUE`, missing values in weighted columns are ignored and remaining weights are rescaled within each row. Rows where all weighted columns are missing receive `NA_real_`. Defaults to `FALSE`, which preserves missing values in `weighted_score`.

Value

A tibble containing all original columns plus `weighted_score`, sorted by score.

Examples

```
book_summary <- summarize_chapter_scores(
  toy_sim_results,
  aggregate_level = "book"
)
rank_weighted(
  book_summary,
  weights = c(mean_delta_outgroup = 0.6, mean_delta_gap = 0.4)
)
```

`read_book_texts` *Read book chapters into a nested list*

Description

Convert a list of chapter file paths (as produced by `list_book_chapters()`) into a nested list of chapter texts: list(book -> list(chapter_name -> text)).

Usage

```
read_book_texts(chapter_list)
```

Arguments

`chapter_list`	A named list of character vectors with file paths.

Value

A nested list of character scalars (texts) with chapter basenames as names. Each book element also stores its book name in a book attribute so selecting a single book with $ preserves enough metadata for simulation helpers.

`rename_chapters` *Rename chapter text files in a folder to a sequential order*

Description

Scans a folder for chapter files and renames them to `chapter1.ext`, `chapter2.ext`, ... using heuristics for ordering (intro, part 1/2, numeric chapter numbers, appendix, etc.).

Usage

```
rename_chapters(folder, extension = "txt")
```

Arguments

folder	Character scalar. Path to the folder containing chapter files.
extension	Character scalar file extension to match, without a leading dot by default. Defaults to "txt".

Value

A tibble with columns old_path, base, order_score, new_name, and new_path.

---

run_ai_cumulative_chapters

Run AI model on books with cumulative chapter context

Description

This function implements a cumulative multi-turn design where each simulation creates one persistent chat per book and identity. The chat first establishes a baseline, then processes chapters sequentially in order, one turn per chapter, preserving context across the full book.

Usage

```
run_ai_cumulative_chapters(
  book_texts,
  groups,
  context_text,
  question_text,
  output_mode = c("structured", "text"),
  n_simulations = 1,
  temperature = 0,
  seed = 42,
  model = "gemini-2.5-flash-lite",
  integration = getOption("nalanda.integration"),
  virtual_key = getOption("nalanda.virtual_key"),
  base_url = getOption("nalanda.base_url"),
  excerpt_chars = 200,
  checkpoint_dir = NULL,
  checkpoint_prefix = "run_ai_cumulative_chapters",
  save_dir = NULL,
  save_prefix = "results"
)
```

Arguments

book_texts	A nested list of books -> chapters as returned by read_book_texts().
groups	Character vector of group labels (length >= 2).

context_text	Character. Either a scalar template containing {identity} or a character vector of length length(groups).
question_text	Character scalar. A question template containing the placeholder {group}, which will be replaced with each group label.
output_mode	Character. "structured" (default) uses the backend's structured-output support. "text" is a compatibility mode for models that do not support structured outputs (for example some Anthropic models): nalanda appends strict JSON-only instructions to the prompt, calls the model as free text, then parses the JSON back into the same fields used by the rest of the pipeline. Text mode is best-effort and stores the original model reply in raw_response.
n_simulations	Integer. Number of repeated simulations per book per identity.
temperature	Numeric. Sampling temperature passed to the chat backend.
seed	Integer. Random seed for reproducibility (incremented for each simulation).
model	Character. Model name for the chat backend.
integration	Optional Portkey/gateway route slug. Use a route returned by ellmer::models_portkey(base_url = "https://ai-gateway.apps.cloud.rt.nyu.edu/v1/") when working with the NYU gateway.
virtual_key	Optional legacy virtual key.
base_url	Character. Base URL for API calls.
excerpt_chars	Integer. Number of chapter characters to retain in the stored prompt previews shown in results.
checkpoint_dir	Optional directory. If supplied, each completed book/identity/simulation conversation is saved as its own .Rds file as soon as it finishes. If the same call is rerun with the same checkpoint_dir, checkpoint_prefix, model, books, groups, and simulations, completed conversations are loaded from disk and skipped.
checkpoint_prefix	Character scalar used at the start of checkpoint filenames when checkpoint_dir is supplied.
save_dir	Optional directory. If supplied, each book is saved as one .Rds file as soon as all of its identities and simulations finish.
save_prefix	Character scalar used in book-level filenames when save_dir is supplied. Files are named {save_prefix}_{book}.Rds.

Value

A tibble or named list of tibbles with cumulative turn-level rows. The baseline turn is followed by one post turn per chapter, all within the same chat per book/identity/simulation.

Examples

```
## Not run:
raw_cumulative <- run_ai_cumulative_chapters(
  book_texts = list(
    "Book A" = list(
```

```
      chapter_1 = "A first chapter about cooperation.",
      chapter_2 = "A second chapter about conflict and repair."
    )
  ),
  groups = c("Democrat", "Republican"),
 context_text = "You are simulating an American adult who politically identifies as a {identity}.",
  question_text = "On a scale from 0 to 100, how warmly do you feel towards {group}s?",
  n_simulations = 1,
  temperature = 0,
  seed = 42
)

compute_run_ai_metrics_cumulative(raw_cumulative)

## End(Not run)
```

`run_ai_on_chapters` *Run AI model on book chapters and collect structured responses*

Description

This function implements a two-turn sequential chat design to measure the effect of reading book chapters on attitudes. For each simulation and each identity assignment, the function:

1. Establishes a baseline by assigning an identity, then asking for ratings of each group (ingroup first, outgroup second).
2. Shows the chapter and asks for post-intervention ratings in the same chat session (same ordering: ingroup first, outgroup second).

This design creates a within-agent pre-post comparison, with conversation memory maintained between turns. Ingroup and outgroup columns are computed post-hoc from the assigned identity and the group labels.

Usage

```
run_ai_on_chapters(
  book_texts,
  groups,
  context_text,
  question_text,
  output_mode = c("structured", "text"),
  n_simulations = 1,
  temperature = 0,
  seed = 42,
  model = "gemini-2.5-flash-lite",
  integration = getOption("nalanda.integration"),
  virtual_key = getOption("nalanda.virtual_key"),
  base_url = getOption("nalanda.base_url"),
```

```
  excerpt_chars = 200,
  checkpoint_dir = NULL,
  checkpoint_prefix = "run_ai_on_chapters",
  save_dir = NULL,
  save_prefix = "results"
)
```

Arguments

book_texts	A single character (one chapter) or a nested list of books -> chapters as returned by `read_book_texts()`.
groups	Character vector of group labels (length >= 2). These are the groups being compared. Example: `c("Democrat", "Republican")`.
context_text	Character. Either: • A scalar template containing `{identity}`, which will be expanded once for each group (e.g., `"You are simulating an American adult who politically identifies as a {identity}."`), or • A character vector of length equal to `length(groups)`, where each element is the full context for the corresponding group identity.
question_text	Character scalar. A question template containing the placeholder `{group}`, which will be replaced with each group label. Example: `"On a scale from 0 to 100, how warmly do you feel towards {group}s?"`
output_mode	Character. `"structured"` (default) uses the backend's structured-output support. `"text"` is a compatibility mode for models that do not support structured outputs (for example some Anthropic models): nalanda appends strict JSON-only instructions to the prompt, calls the model as free text, then parses the JSON back into the same fields used by the rest of the pipeline. Text mode is best-effort and stores the original model reply in `raw_response`.
n_simulations	Integer. Number of repeated simulations per chapter per identity (each simulation = 2 chat turns).
temperature	Numeric. Sampling temperature passed to the chat backend.
seed	Integer. Random seed for reproducibility (incremented for each simulation).
model	Character. Model name for the chat backend (for example, `"gemini-2.5-flash-lite"`). The value is passed directly to `ellmer::chat_portkey(model = ...)`.
integration	Optional Portkey/gateway route slug. Should look like `"vertexai"` or another route returned by `ellmer::models_portkey(base_url = "https://ai-gateway.apps.cloud.rt.nyu` If supplied and `model` is not fully-qualified (does not start with `"@"`), nalanda will build `"@{integration}/{model}"`. In some gateways this slug is not the upstream provider name. When available, a fully-qualified model string such as `"@gpt-5-mini/gpt-5-mini"` is the most reliable option. When both `nalanda.integration` and `nalanda.virtual_key` options are set and neither argument is supplied, `integration` is preferred.
virtual_key	Optional legacy virtual key. Should look like `"gemini-8c2498"` or similar. If supplied and `model` is not fully-qualified, nalanda will build `"@{virtual_key}/{model}"`. Use either `integration` or `virtual_key`, not both when explicitly supplying function arguments.

base_url	Character. Base URL for API calls.
excerpt_chars	Integer. Number of chapter characters to retain in the stored post-prompt preview shown in results.
checkpoint_dir	Optional directory. If supplied, each completed book/chapter/identity/simulation unit is saved as its own .Rds file as soon as it finishes. If the same call is rerun with the same checkpoint_dir, checkpoint_prefix, model, books, groups, and simulations, completed units are loaded from disk and skipped.
checkpoint_prefix	Character scalar used at the start of checkpoint filenames when checkpoint_dir is supplied.
save_dir	Optional directory. If supplied, each book is saved as one .Rds file as soon as all of its chapters, identities, and simulations finish.
save_prefix	Character scalar used in book-level filenames when save_dir is supplied. Files are named {save_prefix}_{book}.Rds.

Details

Authentication uses PORTKEY_API_KEY via ellmer::chat_portkey(). Set it persistently in .Renviron:

```
usethis::edit_r_environ()
# Add a line like:
# PORTKEY_API_KEY=your_api_key_here
```

Then restart your R session.

Value

A tibble of raw turn-level ratings, or a named list of tibbles (one per book). Each row is one rating observation and includes: chapter, sim, identity, turn_index, turn_type, target_group, and rating, plus prompt and metadata columns. Use compute_run_ai_metrics() to derive ingroup/outgroup summaries and gap/delta metrics. The object has class nalanda and model attributes.

Examples

```
# Per-group mode (asks about each group, ingroup first):
make_baseline_prompt(
  identity_context = "You are simulating an American Democrat.",
  question_template = "How warmly do you feel towards {group}s?",
  groups = c("Democrat", "Republican"),
  identity_label = "Democrat"
)

# Single-question mode (asks once, as-is):
make_baseline_prompt(
  identity_context = "You are simulating an American Democrat.",
  question_template = "How warmly do you feel towards your political outgroup?",
  groups = c("Democrat", "Republican"),
  identity_label = "Democrat"
)
```

`run_ai_on_chapters_one_turn` *Run AI model on book chapters with a single prompt per simulation*

Description

This function implements a one-turn design where identity context, chapter text, and the rating question are combined into a single prompt. Independent prompts are executed in parallel with `ellmer::parallel_chat_structured()`.

Usage

```
run_ai_on_chapters_one_turn(
  book_texts,
  groups,
  context_text,
  question_text,
  output_mode = c("structured", "text"),
  n_simulations = 1,
  temperature = 0,
  seed = 42,
  model = "gemini-2.5-flash-lite",
  integration = getOption("nalanda.integration"),
  virtual_key = getOption("nalanda.virtual_key"),
  base_url = getOption("nalanda.base_url"),
  excerpt_chars = 200,
  max_active = 10,
  rpm = 500
)
```

Arguments

`book_texts`	A single character (one chapter) or a nested list of books -> chapters as returned by `read_book_texts()`.
`groups`	Character vector of group labels (length >= 2). These are the groups being compared. Example: `c("Democrat", "Republican")`.
`context_text`	Character. Either: • A scalar template containing `{identity}`, which will be expanded once for each group. • A character vector of length equal to `length(groups)`, where each element is the full context for the corresponding group identity.
`question_text`	Character scalar. A question template containing the placeholder `{group}`, which will be replaced with each group label in per-group mode.
`output_mode`	Character. `"structured"` (default) uses the backend's structured-output support. `"text"` is a compatibility mode for models that do not support structured

	outputs (for example some Anthropic models): nalanda appends strict JSON-only instructions to the prompt, calls the model as free text, then parses the JSON back into the same fields used by the rest of the pipeline. Text mode is best-effort and stores the original model reply in `raw_response`.
`n_simulations`	Integer. Number of repeated simulations per chapter per identity.
`temperature`	Numeric. Sampling temperature passed to the chat backend.
`seed`	Integer. Random seed for reproducibility. As in `run_ai_on_chapters()`, the seed varies by simulation index only, so all chapters and identities within the same `sim` share the same seed.
`model`	Character. Model name for the chat backend.
`integration`	Optional Portkey/gateway route slug. If supplied and `model` is not fully-qualified, nalanda will build `"@{integration}/{model}"`. Use a route returned by `ellmer::models_portkey(bas = "https://ai-gateway.apps.cloud.rt.nyu.edu/v1/")` when working with the NYU gateway. When both `nalanda.integration` and `nalanda.virtual_key` options are set and neither argument is supplied, `integration` is preferred.
`virtual_key`	Optional legacy virtual key. If supplied and `model` is not fully-qualified, nalanda will build `"@{virtual_key}/{model}"`. Use either `integration` or `virtual_key`, not both when explicitly supplying function arguments.
`base_url`	Character. Base URL for API calls.
`excerpt_chars`	Integer. Number of chapter characters to retain in the stored prompt preview shown in results.
`max_active`	Integer. Maximum number of concurrent requests passed to `ellmer::parallel_chat_structured()` in structured mode. Text mode runs plain chat requests sequentially.
`rpm`	Integer. Requests-per-minute cap passed to `ellmer::parallel_chat_structured()` in structured mode. Text mode runs plain chat requests sequentially.

Value

A tibble of raw single-turn ratings, or a named list of tibbles (one per book). Each row is one rating observation and includes `chapter`, `sim`, `identity`, `turn_index`, `turn_type`, `target_group`, and `rating`, plus prompt and metadata columns. Use `compute_run_ai_metrics_one_turn()` to derive chapter-level one-turn summaries.

`run_text_analysis` *Run row-wise text analysis with a prompt template*

Description

This function applies a prompt template to each row of a text dataset and extracts structured responses with `ellmer`. It is designed for dataset-first workflows such as sentiment, emotion, offensiveness, or moral-foundation annotation across many short texts.

Usage

```
run_text_analysis(
  data,
  text_col = "text",
  prompt,
  response_type,
  output_mode = c("structured", "text"),
  id_col = NULL,
  n_simulations = 1,
  temperature = 0,
  seed = 42,
  model = "gemini-2.5-flash-lite",
  integration = getOption("nalanda.integration"),
  virtual_key = getOption("nalanda.virtual_key"),
  base_url = getOption("nalanda.base_url"),
  excerpt_chars = 200,
  max_active = 10,
  rpm = 500
)
```

Arguments

data	A data frame with at least one text column.
text_col	Name of the column containing the text to analyze.
prompt	Character scalar prompt template. It may reference any columns in data using {column_name} placeholders.
response_type	An ellmer structured type specification, for example ellmer::type_object(score = ellmer::type_number()).
output_mode	Character. "structured" (default) uses the backend's structured-output support. "text" is a compatibility mode for models that do not support structured outputs (for example some Anthropic models): nalanda appends strict JSON-only instructions to the prompt, calls the model as free text, then parses the JSON back into the same fields. Text mode is best-effort and stores the original model reply in raw_response.
id_col	Optional column name identifying each text row. When omitted, a sequential text_id is created.
n_simulations	Integer. Number of repeated runs per row.
temperature	Numeric. Sampling temperature passed to the backend.
seed	Integer. Random seed for reproducibility.
model	Character. Model name for the chat backend.
integration	Optional Portkey/gateway route slug. Use a route returned by ellmer::models_portkey(base_url = "https://ai-gateway.apps.cloud.rt.nyu.edu/v1/") when working with the NYU gateway.
virtual_key	Optional legacy virtual key.
base_url	Character. Base URL for API calls.

excerpt_chars	Integer. Number of text characters to retain in stored prompt previews.
max_active	Integer. Maximum number of concurrent requests passed to ellmer::parallel_chat_structured() in structured mode. Text mode runs plain chat requests sequentially.
rpm	Integer. Requests-per-minute cap passed to ellmer::parallel_chat_structured() in structured mode. Text mode runs plain chat requests sequentially.

Value

A tibble containing the original row metadata, simulation index, structured response fields, and stored prompt previews.

---

save_forest_plot *Save a forest plot to PNG and PDF formats*

Description

Exports a forestplot object to both PNG and PDF files using grid graphics devices.

Usage

```
save_forest_plot(
  plot_object,
  filename,
  width = 16/1.8,
  height = 9/1.8,
  res = 300
)
```

Arguments

plot_object	A forestplot grob object.
filename	Character string specifying the file path without extension.
width	Width of the output figure in inches.
height	Height of the output figure in inches.
res	Resolution in DPI for the PNG output (default = 300).

Details

Because forestplot uses grid graphics (not ggplot2), ggsave() is not compatible. This function opens graphics devices manually and prints the plot object.

Two files are created:

filename.png High-resolution raster image

filename.pdf Vector-based PDF

Examples

```
## Not run:
book_summary <- summarize_chapter_scores(
  toy_sim_results,
  aggregate_level = "book"
)
forest_plot <- plot_forest_books(book_summary, xlab = "Delta gap")
save_forest_plot(forest_plot, tempfile("nalanda-forest"))

## End(Not run)
```

`simulate_treatment` *Simulate a generic multi-turn treatment workflow*

Description

This function provides a simpler, prompt-first interface for running one or more turns against an intervention text. Each element of `prompt` defines one turn in the chat sequence. When `groups` is supplied, the same prompt sequence is repeated for each group identity; groups do not create additional turns.

Usage

```
simulate_treatment(
  intervention_text = "",
  prompt,
  response_type,
  output_mode = c("structured", "text"),
  groups = NULL,
  context_text = NULL,
  n_simulations = 1,
  temperature = 0,
  seed = 42,
  model = "gemini-2.5-flash-lite",
  integration = getOption("nalanda.integration"),
  virtual_key = getOption("nalanda.virtual_key"),
  base_url = getOption("nalanda.base_url"),
  excerpt_chars = 200,
  checkpoint_dir = NULL,
  checkpoint_prefix = "simulate_treatment",
  save_dir = NULL,
  save_prefix = "results"
)
```

Arguments

intervention_text	A single character string or a nested list of intervention texts. This is mapped internally onto the same job grid used by run_ai_on_chapters(). Defaults to "", which is useful when the full treatment is already encoded in prompt and/or context_text.
prompt	Character vector of prompt templates. Each element defines one turn. Prompt templates may include {intervention_text}, {identity}, and {group} placeholders.
response_type	An ellmer structured type specification applied to all turns (for example ellmer::type_object(score = ellmer::type_number())).
output_mode	Character. "structured" (default) uses the backend's structured-output support. "text" is a compatibility mode for models that do not support structured outputs (for example some Anthropic models): nalanda appends strict JSON-only instructions to the prompt, calls the model as free text, then parses the JSON back into the same tabular fields. Text mode is best-effort and stores the original model reply in raw_response.
groups	Optional character vector of group labels. If supplied, the full prompt sequence is rerun for each group identity.
context_text	Optional character scalar or vector. If provided, it is prepended to every turn for the corresponding group. Scalar values are recycled across groups, and {identity} is expanded when present.
n_simulations	Integer. Number of repeated simulations per intervention per identity.
temperature	Numeric. Sampling temperature passed to the chat backend.
seed	Integer. Random seed for reproducibility (incremented for each simulation index).
model	Character. Model name for the chat backend.
integration	Optional Portkey/gateway route slug. If supplied and model is not fully-qualified, nalanda will build "@{integration}/{model}". Use a route returned by ellmer::models_portkey(base_url = "https://ai-gateway.apps.cloud.rt.nyu.edu/v1/") when working with the NYU gateway.
virtual_key	Optional legacy virtual key. If supplied and model is not fully-qualified, nalanda will build "@{virtual_key}/{model}".
base_url	Character. Base URL for API calls.
excerpt_chars	Integer. Number of intervention-text characters to retain in stored prompt previews.
checkpoint_dir	Optional directory. If supplied, each completed treatment/identity/simulation unit is saved as its own .Rds file as soon as it finishes. If the same call is rerun with the same checkpoint_dir, checkpoint_prefix, model, treatments, groups, and simulations, completed units are loaded from disk and skipped.
checkpoint_prefix	Character scalar used at the start of checkpoint filenames when checkpoint_dir is supplied.

save_dir	Optional directory. If supplied, each intervention collection is saved as one .Rds file as soon as all of its treatments, identities, and simulations finish.
save_prefix	Character scalar used in book-level filenames when save_dir is supplied. Files are named {save_prefix}_{book}.Rds.

Value

A tibble of raw turn-level responses, or a named list of tibbles (one per book/intervention collection). Each row includes treatment, sim, identity, turn_index, turn_type, and one column per field returned by response_type, plus stored prompt previews and metadata columns.

Examples

```
## Not run:
simulate_treatment(
  intervention_text = "A short passage about people working together.",
  prompt = c(
   "Read the following text:\n\n{intervention_text}\n\nRate its readability from 0 to 100."
  ),
  response_type = ellmer::type_object(
    score = ellmer::type_number()
  ),
  n_simulations = 2,
  temperature = 0,
  seed = 42
)

simulate_treatment(
  groups = c("South African", "Danish"),
  context_text = "You are simulating an adult who identifies as {identity}.",
  prompt = c(
    climate_belief = paste(
      "Generally speaking, do you usually think of yourself as Danish or South African?",
      "On a scale from 0 to 100, how accurate do you think this statement is?",
      "Statement: Human activities are causing climate change"
    )
  ),
  response_type = ellmer::type_object(
    rating = ellmer::type_number()
  ),
  n_simulations = 2,
  temperature = 0,
  seed = 42
)

## End(Not run)
```

summarize_chapter_scores *Summarize simulated chapter scores*

Description

Aggregate simulation results by chapter (and book, if present) computing number of simulations, means and SDs for core model outputs. In the current schema, this includes ingroup/outgroup pre-post ratings plus delta and gap metrics (for example delta_outgroup, delta_ingroup, and delta_gap).

Usage

```
summarize_chapter_scores(
  x,
  aggregate_level = c("chapter", "book"),
  book_chapter_strategy = c("all", "last"),
  standardize = c("none", "z", "minmax", "max"),
  model_aggregation = c("none", "mean"),
  by_party = FALSE
)
```

Arguments

x	A data frame or list-like object containing simulation rows as produced by run_ai_on_chapters(). Expected columns include chapter, pre/post ingroup-outgroup fields, and the derived difference columns used in summaries (pre_gap, post_gap, delta_outgroup, delta_ingroup, delta_gap). If book and party are present, the summary will include those groupings.
aggregate_level	Character. One of "chapter" (default) or "book". When "book", results are aggregated to the book level.
book_chapter_strategy	Character. One of `"all"` (default) or `"last"`. Used only when `aggregate_level = "book"`. `"all"` averages across all chapter rows, while `"last"` first keeps the last non-missing cumulative chapter row per book, simulation, identity, model, and party. If `chapter_index` is unavailable, chapter order is inferred from the `chapter` labels.
standardize	Character. How to standardize metric columns before summarizing. `"none"` (default) keeps raw scores; `"z"` centers and scales scores within model; `"minmax"` rescales scores within model to 0–1; `"max"` divides scores within model by that model's maximum absolute score.
model_aggregation	Character. `"none"` (default) keeps separate rows per model when a `model` column is present. `"mean"` averages the per-model summary estimates into one consensus row per book/chapter/party grouping, adds `n_models`, and adds `sd_model_*` columns measuring disagreement across model-level mean estimates.
by_party	Logical. If TRUE, summaries are computed separately by party (if present).

Value

A tibble summarizing each chapter (and book if present). The returned object will have the original model attribute copied to it.

Examples

```
chapter_summary <- summarize_chapter_scores(toy_sim_results)
chapter_summary

book_summary <- summarize_chapter_scores(
  toy_sim_results,
  aggregate_level = "book"
)
book_summary

party_summary <- summarize_chapter_scores(
  toy_sim_results,
  by_party = TRUE
)
head(party_summary)
```

`summarize_identity_adherence` *Summarize whether the model adopts the requested identity*

Description

This helper turns raw simulation output into a tally table showing which reported `party` was returned for each requested `identity`. It is especially useful for first-turn checks from `run_ai_on_chapters_one_turn()`, where the main question is whether the model actually accepts the assigned identity.

Usage

```
summarize_identity_adherence(
  x,
  by = c("model", "book", "chapter", "identity"),
  compact = FALSE,
  expected_col = "identity",
  observed_col = "party"
)
```

Arguments

`x`	A data frame or list-like object containing raw rows from `run_ai_on_chapters()`, `run_ai_on_chapters_one_turn()`, or another workflow with `identity`, `party`, and `sim` columns.

by	Character vector of columns to group by before tallying. Defaults to c("model", "book", "chapter", "identity"). Missing columns are silently ignored. When compact = TRUE, the default drops identity so the result is one row per model/chapter grouping unless you explicitly include identity.
compact	Logical. If TRUE, return a wide compact summary with one row per grouping combination and one rate_* column per observed identity label.
expected_col	Character scalar naming the requested identity column. Defaults to "identity".
observed_col	Character scalar naming the model-reported identity column. Defaults to "party".

Details

Because raw chapter outputs can contain repeated rows per simulation (for example one row per target group, or one row per turn), this function first reduces the input to one identity-assignment row per simulated unit.

Value

A tibble with one row per grouping combination and reported identity, including counts (n), totals within group (total_n), proportions (prop), and a logical matches_requested.

Examples

```
x <- tibble::tibble(
  chapter = c("chapter_1", "chapter_1", "chapter_1", "chapter_1"),
  sim = c(1, 1, 2, 2),
  identity = c("Democrat", "Democrat", "Democrat", "Democrat"),
  party = c("Democrat", "Democrat", "Republican", "Republican"),
  target_group = c("Democrat", "Republican", "Democrat", "Republican"),
  rating = c(70, 40, 68, 35)
)

summarize_identity_adherence(x)
```

---

summarize_identity_match_rates

Summarize identity match rates by model

Description

This helper converts raw identity-adoption output into one row per grouping combination, reporting how often the model's reported identity matched the requested one.

Usage

```
summarize_identity_match_rates(
  x,
  by = c("model", "identity"),
  compact = FALSE,
  expected_col = "identity",
  observed_col = "party"
)
```

Arguments

x	A data frame or list-like object from a simulation workflow containing identity and party.
by	Character vector of columns to group by. Defaults to c("model", "identity").
compact	Logical. If TRUE, return a wide one-row-per-group summary with one rate column per requested identity plus a shared n column.
expected_col	Character scalar naming the requested identity column.
observed_col	Character scalar naming the model-reported identity column.

Value

A tibble with counts and match rates, including n_requested, n_match, adoption_rate, n_mismatch, and mismatch_rate.

---

summarize_model_correlations

Summarize pairwise model correlations

Description

Condenses the output of model_pairwise_cor() into one row per correlation method and subgroup. The summary includes the average pairwise correlation and the "most aligned" model, defined as the model with the highest average correlation with all other models. This is useful for adding a small headline annotation to correlation-matrix slides.

Usage

```
summarize_model_correlations(data, method = NULL, digits = 2)
```

Arguments

data	Output of model_pairwise_cor().
method	Optional character. If supplied, keep only one correlation method, e.g. "pearson" or "spearman".
digits	Integer. Number of decimal places used in the display label.

Value

A tibble with any subgroup columns from `data`, plus `method`, `mean_correlation`, `median_correlation`, `min_correlation`, `max_correlation`, `n_pairs`, `most_aligned_model`, `most_aligned_correlation`, and `label`.

Examples

```
## Not run:
pw <- model_pairwise_cor(agg, outcome = "mean_rating",
  unit_by = c("book_id", "chapter_id", "group"))
summarize_model_correlations(pw, method = "pearson")

## End(Not run)
```

---

summarize_simulation_stability

Summarize simulation stability across chapters

Description

This helper provides a compact bird's-eye view of where repeated simulation runs vary across chapters, books, parties, or models. It reuses the simulation-level SD columns from `summarize_chapter_scores()` and reports how often each metric showed non-zero variation within the requested grouping.

Usage

```
summarize_simulation_stability(x, by = "party", metrics = NULL, tol = 0)
```

Arguments

`x`	A data frame or list-like object. This can be raw simulation metrics (for example from `compute_run_ai_metrics()`) or chapter summaries from `summarize_chapter_scores()`.
`by`	Character vector of columns used for the compact summary. Defaults to `"party"`.
`metrics`	Optional character vector of metric names to inspect without the `sd_` prefix. Defaults to the four core ratings: `pre_ingroup`, `pre_outgroup`, `post_ingroup`, and `post_outgroup`.
`tol`	Numeric tolerance for treating an SD as zero. Defaults to `0`.

Details

If `x` is already output from `summarize_chapter_scores()`, the function uses it directly. Otherwise, it first computes chapter-level summaries with `by_party = TRUE`, because party-specific stability is the most common diagnostic use case.

Groups with only one simulation row have `NA` SD values from `stats::sd()`. Those groups are treated as not testable for variability and are excluded from the variation counts.

Value

A tibble with one row per grouping combination, including the number of assessed units (`n_units`), the proportion of units showing any pre-period variation, the proportion showing any post-period variation, and an overall `all_stable` flag.

Examples

```
stability <- summarize_simulation_stability(toy_sim_results)
stability

summarize_simulation_stability(
  toy_sim_results,
  by = c("model", "party")
)
```

---

summarize_top_units *Summarize units that rank consistently high across models*

Description

Aggregates lower-level rows to a chosen unit level, ranks units within each model, and summarizes which units most consistently appear near the top across models. This is useful for questions such as "Which books consistently have the strongest effects across models?"

Usage

```
summarize_top_units(
  data,
  outcome = "mean_outcome",
  item_by = "book_id",
  rank_within = NULL,
  model_col = "model",
  top_n = 3,
  higher_is_better = TRUE,
  standardize = c("z", "none", "minmax", "max"),
  include_ranks = FALSE,
  drop_missing = TRUE
)
```

Arguments

`data`	A data frame with one row per model-by-unit combination.
`outcome`	Character string naming the score column (default `"mean_outcome"`).
`item_by`	Character vector identifying the items to rank, e.g. `"book"` or `"book_id"`.
`rank_within`	Optional character vector defining separate ranking contexts, e.g. `"party"` to rank books separately within party.

`model_col` Character string naming the model column (default `"model"`).

`top_n` Integer. Number of top-ranked items to count for each model.

`higher_is_better` Logical. If `TRUE` (default), larger outcome values receive better ranks. If `FALSE`, smaller values receive better ranks.

`standardize` Character. How to standardize item scores within each model before computing cross-model mean scores. `"z"` (default) centers and scales scores within model; `"none"` keeps raw scores; `"minmax"` rescales scores within model to 0–1; `"max"` divides scores within model by that model's maximum absolute score. Ranks are unchanged by monotonic standardization, but `mean_score` and point sizes in `plot_top_units()` use the standardized scores.

`include_ranks` Logical. If `TRUE`, return a list with both the summary table and the model-level ranks. If `FALSE` (default), return only the summary table.

`drop_missing` Logical. Whether to drop rows with missing model, item, or ranking-context identifiers before aggregating (default `TRUE`).

Value

A tibble, or a list with `summary` and `ranks` when `include_ranks = TRUE`.

The summary table contains:

`rank_within` **columns** Optional grouping columns used to define separate ranking contexts, such as party.

`item_by` **columns** The ranked item identifiers, such as book.

`mean_score` Mean outcome score for the item across models.

`score_scale` The score standardization method used for `mean_score`.

`mean_rank` Average rank of the item across models. Lower values indicate more consistently high-ranked items when `higher_is_better = TRUE`.

`overall_mean_rank` When `rank_within` is supplied, the item's average rank computed without those ranking contexts. This preserves a common item order for subgroup displays.

`median_rank` Median rank of the item across models.

`top_n_models` Number of models that ranked the item within the top `top_n` items in its ranking context. For example, if `top_n = 3` and `top_n_models = 4`, then 4 models placed that item in their top 3.

`n_models` Number of models with non-missing ranks for the item.

`top_n` The top-N threshold used to compute `top_n_models`.

`top_n_label` Compact display label combining `top_n_models` and `n_models`, such as `"4/5"`.

When `include_ranks = TRUE`, the `ranks` table contains one row per model-by-item combination, including `score`, `rank`, and `top_n`.

Examples

```
## Not run:
summarize_top_units(
  agg,
  outcome = "mean_delta_gap",
  item_by = "book",
  rank_within = "party",
  model_col = "model",
  top_n = 3
)

## End(Not run)
```

`summarize_treatment_results` *Summarize generic treatment results*

Description

Aggregate raw outputs from `simulate_treatment()` by treatment, computing counts plus means and SDs for numeric response fields. This is useful for structured outputs such as readability, sentiment, or any other custom numeric scores returned by the model.

Usage

```
summarize_treatment_results(
  x,
  aggregate_level = c("treatment", "book"),
  by_identity = FALSE,
  by_turn = TRUE,
  fields = NULL
)
```

Arguments

`x`	A data frame or list-like object containing raw rows as produced by `simulate_treatment()`. If a list is supplied, nested data frames are flattened before summarising.
`aggregate_level`	Character. One of `"treatment"` (default) or `"book"`. `"treatment"` summarizes each treatment/intervention unit separately. `"book"` summarizes across all treatment rows within each book and requires a `book` column, usually produced by passing a nested book list to `simulate_treatment()`.
`by_identity`	Logical. If `TRUE`, summaries are computed separately by identity when an `identity` column is present.
`by_turn`	Logical. If `TRUE` (default), summaries are computed separately by turn using `turn_type` when available, otherwise `turn_index`.

`fields` Optional character vector of numeric columns to summarize. Defaults to all numeric columns except common bookkeeping fields such as `sim`, `turn_index`, and `treatment_index`.

Value

A tibble summarizing the requested aggregation level. Numeric fields are returned as `mean_*` and `sd_*` columns, alongside a `sim` count. Model metadata attributes are copied to the result.

Examples

```
readability <- tibble::tibble(
  treatment = c("treatment_1", "treatment_1", "treatment_2"),
  sim = c(1, 2, 1),
  turn_type = "turn_1",
  readability_score = c(6, 8, 7),
  readability_confidence = c(4, 5, 4)
)

summarize_treatment_results(readability)
```

---

toy_run_ai_turns *Toy raw turn-level AI simulation output*

Description

A synthetic turn-level dataset that mimics the raw output returned by `run_ai_on_chapters()` in per-group mode. It is intended for examples, documentation, and testing the full workflow from raw turns to derived metrics with `compute_run_ai_metrics()`.

Usage

```
toy_run_ai_turns
```

Format

A tibble with 128 rows and 11 variables:

book Book title.

chapter Chapter identifier.

sim Simulation index.

identity Simulated respondent identity.

party Political party grouping.

turn_index Conversation turn index.

turn_type Whether the row comes from the `baseline` or `post` turn.

target_group Group being rated on that row.

rating Synthetic rating on a 0-100 scale.

baseline_prompt Stored baseline prompt preview.

post_prompt Stored post prompt preview.

Source

Synthetic example created for package documentation.

`toy_sim_results` *Toy simulated chapter results*

Description

A small synthetic dataset that mimics the row-level structure returned by `run_ai_on_chapters()`. It is intended for examples, documentation, and testing plotting and summarization workflows without running live AI simulations. The object also includes `model` and `temperature` attributes so plotting helpers can display realistic metadata in subtitles.

Usage

```
toy_sim_results
```

Format

A tibble with 32 rows and 15 variables:

book Book title.

chapter Chapter identifier.

sim Simulation index.

identity Simulated respondent identity.

party Political party grouping.

pre_ingroup Pre-reading ingroup rating.

post_ingroup Post-reading ingroup rating.

pre_outgroup Pre-reading outgroup rating.

post_outgroup Post-reading outgroup rating.

pre_gap Pre-reading ingroup minus outgroup gap.

post_gap Post-reading ingroup minus outgroup gap.

delta_ingroup Change in ingroup rating.

delta_outgroup Change in outgroup rating.

delta_gap Change in affective polarization gap.

chapter_excerpt Short synthetic chapter excerpt.

Source

Synthetic example created for package documentation.

Index