

Package: saydo (via r-universe)

October 28, 2026

Title Link What People Say to What They Do with LLM-Coded Themes

Version 0.1.0

Description Measures the say-do gap in mixed-methods data: whether themes in what people say go with differences in what they do. Codes free text against a codebook with large language models through 'ellmer', or any coder function, and validates the codes against a human-coded sample with Cohen's kappa, precision and recall. Links the codes to participants' behavioural or outcome data, aggregating segment-level transcripts to participants where needed, and tests each theme against each metric with Welch's t-test and multiple-comparison adjustment. Developed by Neudata Consulting Ltd.

License MIT + file LICENSE

URL <https://neu-data.github.io/saydo/>,
<https://github.com/neu-data/saydo>

BugReports <https://github.com/neu-data/saydo/issues>

Depends R (>= 4.1)

Imports cli, ellmer (>= 0.3.0), ggplot2 (>= 3.5.0), rlang, stats,
tibble, withr

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Bernard Isekah Osangir [aut, cre], Neudata Consulting Ltd [cph,
fnd]

Maintainer Bernard Isekah Osangir <b.osangir@gmail.com>

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

Date/Publication 2026-10-28 12:00:02 UTC

RemoteUrl <https://github.com/cran/saydo>
RemoteRef HEAD
RemoteSha f7631e480f23e4df7e54d35ad1ccbb616f04cfd0

Contents

code_agreement	2
cross_tab_themes	3
llm_auto_code	4
mock_coder	6
mock_mixed_data	7
plot_theme_impact	7
Index	9

code_agreement	<i>Measure agreement between LLM codes and human codes</i>
----------------	--

Description

Validates automated coding against a human-coded subsample. For each theme it reports raw agreement, Cohen’s kappa with a confidence interval, and, treating the human codes as the reference, the LLM’s precision, recall and F1 score.

Usage

```
code_agreement(  
  llm_coded,  
  human_coded,  
  id_col = NULL,  
  themes = NULL,  
  conf_level = 0.95  
)
```

Arguments

- llm_coded A data frame returned by [llm_auto_code\(\)](#).
- human_coded A data frame with the same ID column and the human codes in columns named after the themes (logical, or 0/1).
- id_col Name of the ID column (a string). Defaults to the ID column used in [llm_auto_code\(\)](#).
- themes Character vector of theme columns. Defaults to the codebook stored by [llm_auto_code\(\)](#).
- conf_level Confidence level for the kappa interval.

Details

Units are matched by `id_col`, which must be unique in both data frames (use a segment ID when coding at segment level). Units missing from either data frame, or with an NA code from either coder, are left out of that theme's statistics.

The kappa interval uses the large-sample standard error $\sqrt{p_o(1-p_o)/(n(1-p_e)^2)}$, which is approximate for small samples. As a rough guide, kappa above 0.6 is commonly read as substantial agreement and above 0.8 as near-perfect.

Value

A tibble with one row per theme: `n` units compared, prevalence of the theme for each coder, percent agreement, kappa with `kappa_low` and `kappa_high`, and precision, recall and `f1`. Kappa is NA when it is undefined (both coders give every unit the same code).

Examples

```
df <- mock_mixed_data(60, seed = 1)
coded <- llm_auto_code(df, user_id, transcript_feedback,
                      codebook = c("UI_Confusion", "Positive_Experience"),
                      coder = mock_coder())

# Pretend a human coded a subsample, using the hidden profile:
human <- data.frame(
  user_id = df$user_id[1:30],
  UI_Confusion = df$profile[1:30] == "frustrated",
  Positive_Experience = df$profile[1:30] == "efficient"
)
code_agreement(coded, human)
```

cross_tab_themes

Cross-tabulate qualitative themes against a quantitative metric

Description

For each theme, compares a numeric metric between units where the theme is present and units where it is absent, using Welch's two-sample t-test. P-values are adjusted for multiple comparisons across themes.

Usage

```
cross_tab_themes(
  coded_data,
  metric_col,
  themes = NULL,
  metrics = NULL,
  id_col = NULL,
  p_adjust = "BH",
  alpha = 0.05
)
```

Arguments

coded_data	A data frame returned by <code>llm_auto_code()</code> .
metric_col	The numeric metric column (unquoted or a string). It is looked up in <code>metrics</code> if supplied, otherwise in <code>coded_data</code> .
themes	Character vector of theme columns. Defaults to the codebook stored by <code>llm_auto_code()</code> .
metrics	Optional participant-level data frame holding the metric.
id_col	Name of the ID column (a string) used to aggregate and join when <code>metrics</code> is supplied. Defaults to the ID column used in <code>llm_auto_code()</code> .
p_adjust	Multiple-comparison correction passed to <code>stats::p.adjust()</code> . Defaults to "BH" (false discovery rate).
alpha	Significance threshold applied to the adjusted p-value.

Details

When the text was coded at segment level (several rows per participant) and the metrics live in a separate participant-level table, pass that table as `metrics`: codes are first aggregated per participant (a theme counts as present if it appears in any of their segments) and then joined to the metrics by `id_col`.

Value

A tibble with one row per theme: group sizes and means, the mean difference (present minus absent) with its 95% confidence interval, Cohen’s d, raw and adjusted p-values, and a significant flag. Themes with fewer than two units in either group get NA statistics.

Examples

```
df <- mock_mixed_data(80, seed = 42)
coded <- llm_auto_code(df, user_id, transcript_feedback,
                      codebook = c("UI_Confusion", "Positive_Experience",
                                   "Feature_Request"),
                      coder = mock_coder())
cross_tab_themes(coded, session_duration_sec)
```

llm_auto_code	<i>Auto-code text with a codebook using structured LLM extraction</i>
---------------	---

Description

Sends each text to an LLM with a structured-output schema built from the codebook, and adds one logical column per code to data. Each text is coded in an independent conversation, so earlier texts never leak into the context of later ones.

Usage

```
llm_auto_code(
  data,
  id_col,
  text_col,
  codebook,
  chat = NULL,
  coder = NULL,
  max_active = 10
)
```

Arguments

data	A data frame with at least an ID column and a text column. Rows may be participants or individual transcript segments.
id_col	Column holding the participant/unit identifier (unquoted or a string).
text_col	Column holding the qualitative text (unquoted or a string).
codebook	Character vector of codes. Either unnamed labels (c("UI_Confusion", "Feature_Request")) or named definitions (c(UI_Confusion = "The user reports being lost or confused")). Explicit definitions give far more reliable coding.
chat	An ellmer::Chat object, e.g. <code>ellmer::chat_openai()</code> or <code>ellmer::chat_anthropic()</code> . Required unless coder is supplied.
coder	Optional function function(texts, codebook) returning a data frame of logical columns named after the codes, one row per text. Use this to plug in another backend, or mock_coder() for offline demos.
max_active	Maximum number of concurrent LLM requests.

Value

data with one logical column per code. Missing or empty texts get NA. The codebook is stored in the "saydo_codebook" attribute so [cross_tab_themes\(\)](#) can find the code columns automatically.

Examples

```
# Offline, with the keyword-based demo coder:
df <- mock_mixed_data(30, seed = 1)
coded <- llm_auto_code(df, user_id, transcript_feedback,
  codebook = c("UI_Confusion", "Positive_Experience"),
  coder = mock_coder())

head(coded)

## Not run:
# With a real LLM (needs an API key, e.g. OPENAI_API_KEY):
coded <- llm_auto_code(
  df, user_id, transcript_feedback,
  codebook = c(
    UI_Confusion = "The user reports being lost, confused or unable to find something",
    Feature_Request = "The user asks for a feature or capability that does not exist"
```

```

    ),
    chat = ellmer::chat_openai(model = "gpt-4.1-mini")
)

## End(Not run)

```

mock_coder

A keyword-based stand-in coder for demos and tests

Description

Returns a coder function for the coder argument of `llm_auto_code()` that tags texts by regular-expression matching instead of calling an LLM. It is meant for trying the workflow offline with `mock_mixed_data()`, not for real research coding.

Usage

```

mock_coder(
  patterns = c(UI_Confusion = "lost|confus|can't find|frustrat|error",
    Positive_Experience = "love|seamless|clean|effortless|efficient|fast",
    Feature_Request = "wish|please add|would be great")
)

```

Arguments

`patterns` Named character vector of case-insensitive regular expressions, one per code. Codes without a pattern are coded FALSE.

Value

A function `function(texts, codebook)`.

Examples

```

coder <- mock_coder()
coder(c("I got lost", "I love it"), c(UI_Confusion = "", Positive_Experience = ""))

```

mock_mixed_data	<i>Generate synthetic mixed-methods behavioural data</i>
-----------------	--

Description

Simulates users from three latent profiles (frustrated, efficient, casual) with session metrics and a short feedback text whose wording depends on the profile. Frustrated users have longer sessions and more errors; feature requests are added at random, independent of profile, so they should show no real effect. Useful for trying the package without an API key.

Usage

```
mock_mixed_data(n_users = 50, seed = NULL)
```

Arguments

n_users	Number of users to simulate.
seed	Optional random seed. The global random state is restored afterwards.

Value

A tibble with user_id, profile (the hidden ground truth), session_duration_sec, errors_encountered and transcript_feedback.

Examples

```
mock_mixed_data(5, seed = 1)
```

plot_theme_impact	<i>Plot how a metric shifts when each qualitative theme is present</i>
-------------------	--

Description

Draws one row per theme showing the difference in the metric mean (theme present minus absent) with its confidence interval. Point size shows how many units carry the theme; colour marks themes that remain significant after multiple-comparison adjustment.

Usage

```
plot_theme_impact(cross_tab_data, metric_label = NULL)
```

Arguments

cross_tab_data	A tibble returned by cross_tab_themes() .
metric_label	Axis label for the metric. Defaults to the metric name recorded by cross_tab_themes() .

Value

A ggplot object.

Examples

```
df <- mock_mixed_data(80, seed = 42)
coded <- llm_auto_code(df, user_id, transcript_feedback,
                      codebook = c("UI_Confusion", "Positive_Experience",
                                   "Feature_Request"),
                      coder = mock_coder())
plot_theme_impact(cross_tab_themes(coded, session_duration_sec))
```

Index

`code_agreement`, [2](#)
`cross_tab_themes`, [3](#)
`cross_tab_themes()`, [5](#), [7](#)

`ellmer:::Chat`, [5](#)

`llm_auto_code`, [4](#)
`llm_auto_code()`, [2](#), [4](#), [6](#)

`mock_coder`, [6](#)
`mock_coder()`, [5](#)
`mock_mixed_data`, [7](#)
`mock_mixed_data()`, [6](#)

`plot_theme_impact`, [7](#)

`stats::p.adjust()`, [4](#)