Package 'tidyILD'

March 11, 2026

Title Tidy Intensive Longitudinal Data Analysis

Version 0.3.0

Author Alex Litovchenko [aut, cre]

Maintainer Alex Litovchenko <al4877@columbia.edu>

Description A reproducible, tidyverse-style framework for intensive longitudinal data analysis in R, with built-in methodological safeguards, provenance tracking, and reporting tools. Encodes time structure, enforces within-between decomposition, provides spacing-aware lags, and integrates diagnostics and visualization. Use ild_prepare(), ild_center(), ild_lag(), and related functions for a unified pipeline from raw EMA/diary data to interpretable models.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0)

Imports tibble, dplyr, lubridate, rlang, lme4, nlme, ggplot2, mgcv

Suggests testthat (>= 3.0.0), roxygen2, knitr, broom.mixed, clubSandwich, jsonlite, yaml

VignetteBuilder knitr

Collate 'package.R' 'ild-class.R' 'ild_provenance.R' 'ild_methods.R' 'ild_compare_pipelines.R' 'utils.R' 'ild_prepare.R' 'ild_summary.R' 'ild_center.R' 'ild_decomposition.R' 'ild_lag.R' 'ild_spacing_class.R' 'ild_spacing.R' 'ild_design_check.R' 'ild_missing_pattern.R' 'ild_missing_bias.R' 'ild_missing_model.R' 'ild_ipw_weights.R' 'ild_ipw_refit.R' 'ild_tvem.R' 'ild_tvem_plot.R' 'ild_check_lags.R' 'ild_crosslag.R' 'ild_acf.R' 'ild_align.R' 'ild_lme.R' 'ild_person_model.R' 'ild_robust_se.R' 'ild_model_tidiers.R' 'ild_diagnostics.R' 'ild_manifest.R' 'ild_plot.R' 'ild_circadian.R' 'ild_simulate.R' 'ild_power.R' 'data.R' 'broom.R'

RoxygenNote 7.3.3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-03-11 08:40:02 UTC

Contents

`tidyILD-package` *tidyILD: Tidy Intensive Longitudinal Data Analysis*

Description

tidyILD is a reproducible, tidyverse-style framework for intensive longitudinal data (ILD) analysis in R, with built-in methodological safeguards, provenance tracking, and reporting tools. It supports ecological momentary assessment (EMA) and diary studies with a tidy pipeline from raw data to mixed-effects models: explicit time structure, within-between decomposition, spacing-aware lags, and diagnostics. Use it when you have repeated measures per person over time and want consistent handling of time, gaps, centering, and residual correlation (AR1/CAR1).

Details

All ILD structure (‘.ild_*‘ columns and ‘ild_*‘ metadata) is created only by `ild_prepare` (via the internal constructor). Downstream functions expect data prepared with `ild_prepare()`. For the full workflow and applications, see the vignettes.

Getting started

A minimal workflow: simulate or load data, prepare with `ild_prepare`, inspect with `ild_summary`, apply `ild_center` and `ild_lag`, fit with `ild_lme`, then `ild_diagnostics` or `ild_plot`. See the examples below.

Function index by topic

Setup and validation `ild_prepare`, `as_ild`, `is_ild`, `validate_ild`, `ild_meta`

Summaries and inspection `ild_summary`, `ild_spacing_class`, `ild_spacing`, `ild_design_check`, `ild_missing_pattern`, `ild_missing_bias`, `ild_missing_model`, `ild_ipw_weights`, `ild_ipw_refit`, `ild_plot` (types: trajectory, gaps, missingness)

Within-person and lags `ild_center`, `ild_center_plot`, `ild_decomposition`, `ild_lag`, `ild_check_lags`, `ild_crosslag`, `ild_align`

Modeling `ild_lme`, `ild_person_model`, `ild_tvem` (time-varying effects)

Diagnostics and visualization `ild_acf`, `ild_diagnostics`, `ild_plot` (types: fitted, residual_acf), `ild_heatmap`, `ild_spaghetti`, `ild_circadian`, `ild_tvem_plot`

Provenance and methods `ild_provenance`, `ild_history`, `ild_methods`, `ild_report`, `ild_compare_pipelines`, `ild_export_provenance`

Reproducibility `ild_manifest`, `ild_bundle`

Utilities and data `ild_simulate`, `ild_power`, `ema_example`

Person-level `ild_person_model`, `ild_person_distribution`

Model tidiers `augment_ild_model`, `tidy_ild_model` (model or robust SE via `se = "robust"`), `ild_robust_se`; `tidy.ild_lme`, `augment.ild_lme` (broom.mixed, see `broom_ild_lme`)

Vignettes

`browseVignettes("tidyILD")` lists all vignettes. Key entries:

- *From raw data to model with tidyILD* — Full pipeline: prepare, inspect, center, lag, fit, diagnose.
- *Short analysis report* — Fit, tidy fixed effects, fitted vs observed, residual ACF and Q-Q.
- *Within-between decomposition and irregular spacing* — Centering (BP/WP), gap-aware lags, spacing classification.
- *Reproducible ILD workflows with tidyILD provenance* — Inspect history, generate methods text, ild_report(), export and compare pipelines.
- *Glossary and quick-start checklist* — Table of main functions and a short checklist.

Key concepts

- **ILD:** Intensive longitudinal data; many repeated measurements per person over time (e.g. EMA).
- **Within-between decomposition:** `ild_center` adds `_bp` (person mean) and `_wp` (within-person deviation); use WP for within-person effects and BP for between-person or cross-level terms.
- **Spacing-aware lags:** `ild_lag` supports `index`, `gap_aware` (NA when `gap` > `max_gap`), and `time_window`; avoids misalignment from assuming equal spacing.
- **Residual correlation:** `ild_lme` can fit nlme with AR1 or CAR1 for residual autocorrelation; `ild_spacing_class` helps choose `regular-ish` vs `irregular-ish` spacing.
- **Person-level:** `ild_person_model` fits models separately per participant; `ild_person_distribution` plots the distribution of estimates across persons (N-of-1 / idiographic).

Author(s)

Alex Litovchenko <`al4877@columbia.edu`>

See Also

`browseVignettes` and `vignette(package = "tidyILD")` for vignettes. Core entry points: `ild_prepare`, `ild_lme`. Related packages: **lme4**, **nlme** (model backends), **broom.mixed** (tidiers).

Examples

```
library(tidyILD)
d <- ild_simulate(n_id = 10, n_obs_per = 12, irregular = TRUE, seed = 42)
x <- ild_prepare(d, id = "id", time = "time", gap_threshold = 7200)
ild_summary(x)
x <- ild_center(x, y)
x <- ild_lag(x, y, mode = "gap_aware", max_gap = 7200)
fit <- ild_lme(y ~ 1, data = x, ar1 = TRUE, correlation_class = "CAR1")
ild_diagnostics(fit, data = x)
ild_plot(fit, type = "fitted")
```

`as_ild` *Coerce to ILD object*

Description

If the object already has the required '.ild_*' columns and attributes, validates and returns it (with `tidyild_df` and `ild_tbl` class if missing). Otherwise errors.

Usage

```
as_ild(x)
```

Arguments

`x`	A data frame or tibble that may already be ILD-shaped.

Value

An ILD tibble with class `tidyild_df` and `ild_tbl`.

`augment_ild_model` *Augment an ILD model fit with fitted values and residuals*

Description

Returns a tibble with one row per observation: `.ild_id`, `.ild_time`, the response variable (column name from the model formula, e.g. `y`), `.fitted`, and `.resid`. This structure is used internally by [ild_diagnostics()] and [ild_plot()]. Requires `attr(fit, "ild_data")`; refit with [ild_lme()] if missing. Random effects predictions can be added later.

Usage

```
augment_ild_model(fit, ...)
```

Arguments

`fit`	A fitted model from [ild_lme()] (must have `attr(fit, "ild_data")`).
`...`	Unused.

Value

A tibble with columns `.ild_id`, `.ild_time`, the response (name from formula), `.fitted`, `.resid`.

`broom_ild_lme` *Tidy and augment ild_lme fits with broom.mixed*

Description

These S3 methods delegate to [broom.mixed::tidy()] and [broom.mixed::augment()] on the underlying model object so that ild_lme fits work in tidy workflows. Package **broom.mixed** must be attached (e.g. `library(broom.mixed)`).

Usage

```
tidy.ild_lme(x, ...)

augment.ild_lme(x, ...)
```

Arguments

`x`	A fitted model from [ild_lme()].
`...`	Passed to `broom.mixed::tidy()` or `broom.mixed::augment()`.

Value

Same as the corresponding broom.mixed method.

`ema_example` *Example EMA-style intensive longitudinal dataset*

Description

A small simulated dataset with 10 persons and 14 observations per person, irregular timing, and two variables (mood, stress). For use in examples and vignettes. Use [ild_prepare()] to convert to an ILD object.

Format

A data frame with 140 rows and 4 columns:

id Person identifier (1–10).

time POSIXct timestamp (irregular within person).

mood Simulated mood score.

stress Simulated stress score.

Source

Simulated with a fixed seed (12345) for reproducibility.

`ild_acf` *Autocorrelation function for ILD variables or model residuals*

Description

Computes ACF on a variable in ILD data or on residuals from an [ild_lme()] fit. Use this to check whether AR1 is appropriate before fitting models. ACF is computed over the ordered observation sequence (pooled or within person); it does not adjust for irregular time gaps.

Usage

```
ild_acf(x, ..., by_id = FALSE)
```

Arguments

`x`	Either an ILD object (see [is_ild()]) or a fitted model from [ild_lme()].
`...`	When `x` is ILD data, the variable(s) to compute ACF on (tidy-select; one variable). Ignored when `x` is a fit.
`by_id`	Logical. If `TRUE`, also return per-person ACF in `acf_by_id` (default `FALSE`).

Value

A list with `acf`: a tibble with columns `lag` and `acf` (pooled). If `by_id = TRUE`, `acf_by_id` is a named list of tibbles (one per person).

Examples

```
d <- ild_simulate(n_id = 5, n_obs_per = 10, seed = 1)
x <- ild_prepare(d, id = "id", time = "time")
ild_acf(x, "y")
fit <- ild_lme(y ~ 1 + (1 | id), data = x, ar1 = FALSE, warn_no_ar1 = FALSE)
ild_acf(fit)
```

ild_align *Align a secondary stream to primary ILD within a time window*

Description

For each row in the primary ILD, finds observations in the secondary data set (same id, time within `window` before the primary time) and attaches an aggregated value (e.g. mean, median, or closest). Use when combining self-report with wearables or other streams that have different timestamps.

Usage

```
ild_align(
  primary,
  secondary,
  value_var,
  window,
  time_secondary = "time",
  fun = c("mean", "median", "closest")
)
```

Arguments

`primary`	An ILD object (see [is_ild()]); the stream to keep as rows.
`secondary`	A data frame with id and time columns and the value variable(s) to align.
`value_var`	Character. Name of the column in `secondary` to align (e.g. `"heart_rate"`).
`window`	Numeric or lubridate duration. Time window (same units as `.ild_time_num`, e.g. seconds for POSIXct). Only secondary observations with time in `(primary_time - window, primary_time]` are used.
`time_secondary`	Character. Name of the time column in `secondary` (default `"time"`).
`fun`	Character. Aggregation for values in window: `"mean"`, `"median"`, or `"closest"` (most recent in window).

Value

The primary data with a new column `<value_var>_aligned` (numeric; NA where no secondary obs in window).

Examples

```
prim <- ild_prepare(
  data.frame(
    id = rep(1:2, each = 3),
    time = as.POSIXct(rep(c(0, 3600, 7200), 2), origin = "1970-01-01"),
    y = rnorm(6)
  ),
  id = "id", time = "time"
)
sec <- data.frame(
  id = rep(1:2, each = 4),
  time = as.POSIXct(rep(c(0, 1800, 3600, 5400), 2), origin = "1970-01-01"),
  heart_rate = 60 + rnorm(8, 0, 5)
)
ild_align(prim, sec, "heart_rate", window = 3600, fun = "mean")
```

`ild_bundle` *Bundle a result with a reproducibility manifest*

Description

Combines a result (e.g. a fit from [ild_lme()] or output from [ild_diagnostics()]) with a manifest and optional label for one-shot saving. Typical use: `saveRDS(ild_bundle(fit, label = "model_ar1"), "run.rds")`. You can build a manifest with [ild_manifest()] and pass `scenario` (e.g. from [ild_summary()]) and `seed` before bundling.

Usage

```
ild_bundle(result, manifest = NULL, label = NULL)
```

Arguments

`result`	Any object (e.g. fitted model, diagnostics list).
`manifest`	List. Reproducibility manifest from [ild_manifest()]. If `NULL`, [ild_manifest()] is called with default arguments.
`label`	Optional character. Short label for the run (e.g. `"model_ar1"` or `"diagnostics"`).

Value

A list with elements `result`, `manifest`, `label`, suitable for [saveRDS()].

Examples

```
dat <- ild_prepare(ild_simulate(seed = 1), "id", "time")
fit <- ild_lme(y ~ 1 + (1 | id), dat, ar1 = FALSE, warn_no_ar1 = FALSE)
b <- ild_bundle(fit, label = "ar1")
names(b)
b <- ild_bundle(fit, manifest = ild_manifest(seed = 1, scenario = list(n_obs = 50)), label = "run1")
```

`ild_center` *Within-person and between-person decomposition (centering)*

Description

For each selected variable, computes the person mean (between-person component) and the within-person deviation (variable minus person mean). Use ‘*_wp‘ at level-1 and ‘*_bp‘ at level-2 or in cross-level interactions to avoid ecological fallacy and conflation bias. Selected variables must be numeric.

Usage

```
ild_center(
  x,
  ...,
  type = c("person_mean", "grand_mean"),
  naming = c("suffix", "prefix")
)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`...`	Variables to center (tidy-select). Unquoted names or a single character vector of column names. Must be numeric.
`type`	Character. ‘"person_mean"‘ (default) for person-mean centering (x_bp, x_wp); ‘"grand_mean"‘ for grand-mean centering (x_gm, x_wp_gm).
`naming`	Character. ‘"suffix"‘ (default): new columns `var_bp`, `var_wp`; `"prefix"`: new columns `BP_var`, `WP_var`.

Value

The same ILD tibble with additional columns. ILD attributes are preserved.

`ild_center_plot` *Standalone WP/BP centering plot*

Description

Shows within-person deviation and between-person (person mean) distribution for selected variable(s). Uses the same plot as [ild_decomposition(..., plot = TRUE)]. Useful when you only want the visualization without the variance table.

Usage

```
ild_center_plot(x, ...)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`...`	Variables to plot (tidy-select). Must be numeric. Only the first is plotted.

Value

A ggplot object (WP vs BP density overlay for the first selected variable).

Examples

```
d <- ild_simulate(n_id = 10, n_obs_per = 8, seed = 1)
x <- ild_prepare(d, id = "id", time = "time")
ild_center_plot(x, y)
```

`ild_check_lags` *Check lag variable validity (gap-aware)*

Description

Given an ILD object and lag variable names, reports how many lagged values are valid vs invalid (NA because the time distance to the lagged row exceeded a threshold). Useful to audit lag columns before modeling without re-specifying `max_gap`.

Usage

```
ild_check_lags(x, lag_vars = NULL, max_gap = NULL)
```

Arguments

`x`	An ILD object (see [is_ild()]) that contains lag columns (e.g. from [ild_lag()] with `mode = "gap_aware"`).
`lag_vars`	Character vector of lag column names (e.g. `"y_lag1"`). If `NULL`, attempts to detect columns ending in `_lag{n}` or `_lag_window`.
`max_gap`	Numeric. Threshold used to define invalid (same units as `.ild_time_num`). If `NULL`, uses `ild_meta(x)$ild_gap_threshold`.

Value

A data frame with one row per lag variable: `var`, `lag` (parsed lag order or `"window"`), `n_valid`, `n_invalid`, `n_first`, `n_total`, `pct_valid`, `pct_invalid`.

`ild_circadian` *Time-of-day pattern plot for ILD (circadian-style)*

Description

Plots a variable by hour of day (or time-of-day) when `.ild_time` is POSIXct. Useful for EMA (e.g. mood or activity by hour). Does not add columns to the ILD object; hour is derived internally for plotting.

Usage

```
ild_circadian(x, var, type = c("boxplot", "line"))
```

Arguments

`x`	An ILD object (see [is_ild()]) with `.ild_time` as POSIXct.
`var`	Character or symbol. Variable to plot (e.g. mood, activity).
`type`	Character. `"boxplot"` (default) or `"line"` (mean per hour with optional SE).

Value

A ggplot object (variable by hour of day).

Examples

```
d <- ild_simulate(n_id = 5, n_obs_per = 12, seed = 1)
x <- ild_prepare(d, id = "id", time = "time")
ild_circadian(x, y)
```

`ild_compare_pipelines` *Compare provenance of two objects and report differences*

Description

Flattens provenance for each object (data steps plus analysis steps), then compares step sequences and key arguments. Useful to compare preprocessing pipelines, model settings, or to see what changed between two analyses.

Usage

```
ild_compare_pipelines(x1, x2)
```

Arguments

`x1`	First object (ILD data, model fit, or diagnostics with provenance).
`x2`	Second object (same types as x1).

Value

A list of class `ild_compare_pipelines` with `only_in_first` (step names only in x1), `only_in_second`, `differing` (list of per-step arg differences), and `summary` (character vector of human-readable differences). If either object has no provenance, returns a list with `message` and empty comparison components.

Examples

```
set.seed(1)
d <- ild_simulate(n_id = 5, n_obs_per = 6, seed = 1)
x1 <- ild_prepare(d, id = "id", time = "time")
x1 <- ild_center(x1, y)
x2 <- ild_prepare(d, id = "id", time = "time")
x2 <- ild_center(x2, y)
ild_compare_pipelines(x1, x2)
```

`ild_crosslag` *Cross-lag model: lag predictor then fit outcome ~ lag*

Description

One-call pipeline: [ild_lag()] the predictor, [ild_check_lags()] to audit, then [ild_lme()] to fit outcome on the lagged predictor. Returns the fit, the lag-term coefficient (estimate, CI, p), and lag validity check.

Usage

```
ild_crosslag(
  data,
  outcome,
  predictor,
  lag = 1L,
  mode = c("gap_aware", "index", "time_window"),
  max_gap = NULL,
  ar1 = FALSE,
  include_diagnostics = FALSE,
  ...
)
```

Arguments

`data`	An ILD object (see [is_ild()]).
`outcome`	Character or symbol. Name of the outcome variable (e.g. `"mood"`).
`predictor`	Character or symbol. Name of the predictor to lag (e.g. `"stress"`).
`lag`	Integer. Lag order (default 1).
`mode`	Character. Passed to [ild_lag()]: `"index"`, `"gap_aware"` (default), or `"time_window"`.

`max_gap`	Numeric or NULL. Passed to [ild_lag()] when `mode = "gap_aware"`.
`ar1`	Logical. If `TRUE`, fit with [ild_lme()] using AR1/CAR1 (default `FALSE`).
`include_diagnostics`	Logical. If `TRUE`, attach [ild_diagnostics()] to the return (default `FALSE`).
`...`	Passed to [ild_lme()].

Value

A list: `fit` (fitted model), `lag_term` (one-row tibble from [tidy_ild_model()] for the lag variable), `lag_check` (tibble from [ild_check_lags()]), `data` (ILD with lag column), and optionally `diagnostics`.

Examples

```
d <- ild_simulate(n_id = 10, n_obs_per = 8, seed = 1)
x <- ild_prepare(d, id = "id", time = "time")
out <- ild_crosslag(x, "y", "y", lag = 1, ar1 = FALSE, warn_no_ar1 = FALSE)
out$lag_term
out$lag_check
```

`ild_decomposition` *Within-person and between-person variance decomposition*

Description

Reports WP and BP variance and their ratio for selected variables. Use as a diagnostic and teaching tool: large ratio suggests within-person variance dominates; small ratio suggests between-person differences dominate. Helps avoid conflating WP and BP effects in modeling.

Usage

```
ild_decomposition(x, ..., plot = FALSE)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`...`	Variables to decompose (tidy-select). Must be numeric.
`plot`	Logical. If `TRUE`, return a list with `table` and `plot` (WP vs BP density overlay). Default `FALSE`.

Value

A tibble with columns `variable`, `wp_var`, `bp_var`, `ratio` (wp_var / bp_var; `Inf` if bp_var is 0). If `plot = TRUE`, a list with `table` and `plot` (ggplot).

Examples

```
d <- ild_simulate(n_id = 10, n_obs_per = 8, seed = 1)
x <- ild_prepare(d, id = "id", time = "time")
ild_decomposition(x, y)
```

`ild_design_check` *ILD design diagnostics: spacing, WP/BP, missingness, and recommendations*

Description

Aggregates [ild_summary()], [ild_spacing()], [ild_spacing_class()], and optionally [ild_decomposition()] and [ild_missing_pattern()] into one design summary. Use before modeling to see spacing class, correlation recommendation, within- vs between-person variance, and missingness.

Usage

```
ild_design_check(x, vars = NULL)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`vars`	Optional character vector of variable names for decomposition and missingness. If `NULL`, only spacing and summary are computed; `wp_bp` and `missingness` will be `NULL`.

Value

A list of class `ild_design_check`: `summary` (from ild_summary), `spacing_class` (regular-ish / irregular-ish), `spacing` (from ild_spacing), `recommendation` (AR1/CAR1 text), `wp_bp` (decomposition tibble or NULL), `missingness` (list with `summary` tibble and `pct_na` overall, or NULL). Use `print()` for a human-readable summary.

Examples

```
d <- ild_simulate(n_id = 10, n_obs_per = 8, irregular = TRUE, seed = 1)
x <- ild_prepare(d, id = "id", time = "time", gap_threshold = 7200)
ild_design_check(x, vars = "y")
```

ild_diagnostics *Residual diagnostics for an ILD model*

Description

Computes residual ACF (by person and/or pooled), residual vs fitted, residual vs time, and optional Q-Q. Use ‘type‘ to request only specific diagnostics. For ‘ild_lme‘ models with ‘ar1 = TRUE‘, the estimated AR/CAR parameter is reported when ‘type‘ includes ‘"residual_acf"‘.

Usage

```
ild_diagnostics(
  object,
  data = NULL,
  type = c("residual_acf", "residual_time", "qq"),
  by_id = TRUE,
  ...
)
```

Arguments

object	A fitted model from [ild_lme()] (or an object with ‘residuals()‘, and optional ‘fitted()‘; if not ‘ild_lme‘, pass ‘data‘ with ‘.ild_id‘ and ‘.ild_time_num‘ or ‘.ild_seq‘).
data	Optional. ILD data (required if ‘object‘ is not from [ild_lme()]).
type	Character vector. Which diagnostics to compute: ‘"residual_acf"‘, ‘"residual_time"‘ (residuals vs time and vs fitted), ‘"qq"‘. Default is all three.
by_id	Logical. If ‘TRUE‘, compute ACF within each person (default ‘TRUE‘).
...	Unused.

Details

The return value follows a stable schema: ‘meta‘ (engine, ar1, id/time columns, n_obs, n_id), ‘data$residuals‘ (tibble with ‘.ild_id‘, ‘.ild_time‘, response, ‘.resid‘, ‘.fitted‘), and ‘stats‘ (e.g. ‘acf‘, ‘ar1_param‘). Plots are not stored in the object; use [plot_ild_diagnostics()] to generate them from a diagnostics object. The column `.resid` is always filled; `.fitted` is filled when it can be computed without refitting, otherwise it is `NA` (same for both engines and all `type` values).

Residual ACF is computed over the ordered observation sequence within person; it does not adjust for irregular time gaps.

Value

A list of class ‘ild_diagnostics‘ with: ‘meta‘ (engine, ar1, id_col, time_col, n_obs, n_id, type, by_id), ‘data‘ (list with ‘residuals‘ = tibble of .ild_id, .ild_time, response (name from formula), .resid, .fitted; `data$residuals` always exists, `.resid` is always filled, `.fitted` is returned when it can be computed without refitting, otherwise NA), ‘stats‘ (list with ‘acf‘ = list(pooled = tibble, by_id

= list) when requested, ‘ar1_param‘ = numeric or NULL for lme). Use [plot_ild_diagnostics()] for plots.

Examples

```
x <- ild_prepare(ild_simulate(n_id = 3, n_obs_per = 6, seed = 1), id = "id", time = "time")
fit <- ild_lme(y ~ 1 + (1 | id), data = x, ar1 = FALSE, warn_no_ar1 = FALSE)
diag <- ild_diagnostics(fit, type = c("residual_acf", "qq"))
plot_ild_diagnostics(diag)
```

`ild_export_provenance` *Export provenance to a JSON or YAML file*

Description

Writes the full provenance structure (data or analysis) to a file for reproducibility supplements, preregistration appendices, or lab archiving. Requires the **jsonlite** package for JSON or **yaml** for YAML.

Usage

```
ild_export_provenance(x, path, format = c("auto", "json", "yaml"))
```

Arguments

`x`	An ILD object or an analysis object with [ild_provenance()].
`path`	Character. File path to write (e.g. `"analysis_provenance.json"`).
`format`	Character. `"auto"` (default) infers from file extension (`.json` -> JSON, `.yaml` or `.yml` -> YAML). Use `"json"` or `"yaml"` to force a format.

Value

The `path` invisibly, after writing the file.

`ild_heatmap` *ILD heatmap (alias for ild_plot with type = "heatmap")*

Description

Person x time heatmap of a variable. See [ild_plot()].

Usage

```
ild_heatmap(x, var = NULL, ...)
```

Arguments

`x`	ILD object or fitted model (for heatmap, data are taken from ild_data if model).
`var`	Variable to plot. If NULL, single data column is used.
`...`	Passed to [ild_plot()] (e.g. id_var, time_var).

Value

A ggplot object.

`ild_history` *Print a human-readable log of preprocessing or analysis steps*

Description

For ILD data, displays data provenance steps. For analysis objects (e.g. from [ild_lme()], [ild_diagnostics()]), displays source data provenance (if any) and analysis steps. Use [ild_provenance()] to get the raw structured object.

Usage

```
ild_history(x)
```

Arguments

`x`	An ILD object (see [is_ild()]) or an analysis object with `ild_provenance`.

Value

The provenance object (from [ild_provenance()]) invisibly, or a message if none.

`ild_ipw_refit` *Refit an ILD model with inverse-probability weights (sensitivity analysis)*

Description

Takes a fit from [ild_lme()] (or a formula and data) and refits the model using observation weights from [ild_ipw_weights()]. Only the lme4 (lmer) path is supported; nlme (`ar1 = TRUE`) is not supported for weighted refits. This is a sensitivity tool, not a full MNAR solution.

Usage

```
ild_ipw_refit(fit_or_formula, data, weights = ".ipw", ar1 = FALSE, ...)
```

Arguments

`fit_or_formula`	Either a fitted model from [ild_lme()] (lmerMod), or a formula. If a formula, `data` and `weights` are required.
`data`	An ILD object (see [is_ild()]) containing a `.ipw` column when refitting from a fit (or the data to fit when `fit_or_formula` is a formula).
`weights`	Character. Name of the weight column in `data` (default `".ipw"`).
`ar1`	Logical. Must be `FALSE` for IPW refit (nlme weighted refit not supported).
`...`	Passed to [ild_lme()] or `lme4::lmer()`.

Value

A fitted model (lmerMod) with `attr(..., "ild_data")` set so that [tidy_ild_model()] and [ild_diagnostics()] work.

Examples

```
set.seed(1)
d <- ild_simulate(n_id = 12, n_obs_per = 10, seed = 1)
d$stress <- rnorm(nrow(d))
d$mood <- d$y
d$mood[sample(nrow(d), 25)] <- NA
x <- ild_prepare(d, id = "id", time = "time")
x <- ild_center(x, mood)
mm <- ild_missing_model(x, "mood", "stress")
xw <- ild_ipw_weights(x, mm, stabilize = TRUE)
fit0 <- ild_lme(mood ~ mood_bp + mood_wp + stress + (1 | id), data = xw,
  ar1 = FALSE, warn_no_ar1 = FALSE)
fitw <- ild_ipw_refit(fit0, data = xw)
tidy_ild_model(fitw)
```

`ild_ipw_weights` *Compute inverse-probability-of-observation weights from a missingness model*

Description

Uses the fitted missingness model from [ild_missing_model()] to compute weights: unstabilized `w = 1 / p_obs` or stabilized `w = mean(p_obs) / p_obs`. Weights are trimmed to avoid extremes. Use with [ild_ipw_refit()] for sensitivity analysis. This is diagnostic and sensitivity tooling, not a full MNAR solution.

Usage

```
ild_ipw_weights(x, miss_fit, stabilize = TRUE, trim = c(0.01, 0.99))
```

Arguments

`x`	An ILD object (see [is_ild()]); rows must align to the data used to fit `miss_fit`.
`miss_fit`	The return value of [ild_missing_model()] (list with `fit`, `p_missing`, etc.).
`stabilize`	Logical. If `TRUE` (default), use stabilized weights `mean(p_obs) / p_obs`; otherwise `1 / p_obs`.
`trim`	Numeric of length 2. Quantiles to trim weights (default `c(0.01, 0.99)`). Weights below/above these quantiles are set to the quantile values.

Value

`x` with an added column `.ipw` (numeric). ILD attributes are preserved.

Examples

```
set.seed(1)
d <- ild_simulate(n_id = 15, n_obs_per = 8, seed = 1)
d$stress <- rnorm(nrow(d))
d$mood <- d$y
d$mood[sample(nrow(d), 20)] <- NA
x <- ild_prepare(d, id = "id", time = "time")
mm <- ild_missing_model(x, "mood", "stress")
xw <- ild_ipw_weights(x, mm, stabilize = TRUE)
summary(xw$.ipw)
```

`ild_lag` *Spacing-aware lag within person*

Description

Computes lagged values within each person. Use this instead of [dplyr::lag()], which assumes equal spacing and no gaps and is unsafe for irregular ILD.

Usage

```
ild_lag(
  x,
  ...,
  n = 1L,
  mode = c("index", "gap_aware", "time_window"),
  max_gap = NULL,
  window = NULL,
  resolution = c("closest_prior", "last_in_window", "mean_in_window")
)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`...`	Variables to lag (tidy-select). Unquoted names or selection.
`n`	Integer. Lag order (default 1 = previous observation).
`mode`	Character. `"index"`: row-based lag. `"gap_aware"`: same but NA when interval exceeds `max_gap`. `"time_window"`: value from (time - window, time] with `resolution`.
`max_gap`	Numeric. For `gap_aware` only. Same units as `.ild_time_num`. If `NULL`, uses `ild_meta(x)$ild_gap_threshold` (metadata-driven default).
`window`	Numeric or lubridate duration. For `time_window` only: time window width. Numeric is in same units as `.ild_time_num` (e.g. seconds for POSIXct). You can pass a lubridate period/duration (e.g. `lubridate::hours(2)`); it is converted to seconds for POSIXct.
`resolution`	Character. For `time_window`: `"closest_prior"` (default: most recent observation in window), `"last_in_window"`, or `"mean_in_window"`.

Value

The same ILD tibble with new lag columns. ILD attributes preserved.

`ild_lme` *Fit a linear mixed-effects model to ILD*

Description

When ‘ar1 = FALSE‘, fits with [lme4::lmer()] (no residual correlation). When ‘ar1 = TRUE‘, fits with [nlme::lme()] using a residual correlation structure: CAR1 (continuous-time) by default for irregular spacing, or AR1 when spacing is regular-ish. Use [ild_spacing_class()] to inform the choice; override with ‘correlation_class‘.

Usage

```
ild_lme(
  formula,
  data,
  ar1 = FALSE,
  correlation_class = c("auto", "AR1", "CAR1"),
  random = ~1 | .ild_id,
  warn_no_ar1 = TRUE,
  warn_uncentered = TRUE,
  ...
)
```

Arguments

`formula`	Fixed-effects formula. For ‘ar1 = TRUE‘, must be fixed-only (e.g. ‘y ~ x‘); random structure is set to ‘~ 1 | .ild_id‘ internally. For ‘ar1 = FALSE‘, formula may include random effects (e.g. ‘y ~ x + (1|id)‘).
`data`	An ILD object (see [is_ild()]).
`ar1`	Logical. If ‘TRUE‘, fit with nlme and residual AR1/CAR1 correlation; if ‘FALSE‘, fit with lme4 (no residual correlation).
`correlation_class`	Character. ‘"auto"‘ (default) uses [ild_spacing_class()] to choose CAR1 (irregular-ish) or AR1 (regular-ish). Use ‘"CAR1"‘ or ‘"AR1"‘ to override.
`random`	For ‘ar1 = TRUE‘, the random effects formula (default ‘~ 1 | .ild_id‘). Must use ‘.ild_id‘ as grouping for correlation to match.
`warn_no_ar1`	If ‘TRUE‘ (default), warn when ‘ar1 = FALSE‘ that temporal autocorrelation is not modeled.
`warn_uncentered`	If ‘TRUE‘ (default), warn when a predictor in the formula varies both within and between persons but is not decomposed (no _wp/_bp); suggests using [ild_center()].
`...`	Passed to [lme4::lmer()] or [nlme::lme()].

Value

A fitted model object (class ‘lmerMod‘ or ‘lme‘) with attribute ‘ild_data‘ (the ILD data) and ‘ild_ar1‘ (logical). When ‘ar1 = TRUE‘, the returned object has class ‘ild_lme‘ prepended and attribute ‘ild_random_resolved‘ (the formula actually passed to nlme, e.g. ‘~ 1 | M2ID‘). See [ild_diagnostics()] and [ild_plot()].

Examples

```
# lme4 path: formula includes random effects
set.seed(1)
dat <- ild_simulate(n_id = 5, n_obs_per = 6, seed = 1)
dat <- ild_prepare(dat, id = "id", time = "time")
dat <- ild_center(dat, y)
fit_lmer <- ild_lme(y ~ y_bp + y_wp + (1 | id), data = dat,
                    ar1 = FALSE, warn_no_ar1 = FALSE)
# nlme path (may not converge on all platforms; see ?nlme::lme)
## Not run:
fit_lme <- ild_lme(y ~ y_bp + y_wp, data = dat,
                   random = ~ 1 | id, ar1 = TRUE)

## End(Not run)
```

`ild_manifest` *Create a reproducibility manifest*

Description

Captures timestamp, optional seed, optional scenario fingerprint, session info, and optional git SHA for use when saving or serializing results (e.g. after [ild_lme()] or [ild_diagnostics()]). The return value is a serializable list suitable for [saveRDS()] or [ild_bundle()].

Usage

```
ild_manifest(
  seed = NULL,
  scenario = NULL,
  include_session = TRUE,
  include_git = FALSE,
  git_path = "."
)
```

Arguments

`seed`	Optional integer. Seed used for the run (e.g. from [ild_simulate()] or set before fitting). Not captured automatically; pass explicitly if you want it in the manifest.
`scenario`	Optional. Named list or character string describing the run (e.g. formula, n_obs, n_id, ar1). Build from [ild_summary()] or a short list when calling after [ild_lme()] / [ild_diagnostics()].
`include_session`	Logical. If ‘TRUE‘ (default), include [utils::sessionInfo()] in the manifest. Set to ‘FALSE‘ to reduce size.
`include_git`	Logical. If ‘TRUE‘, attempt to record the current git commit SHA from `git_path`. Default ‘FALSE‘.
`git_path`	Character. Path to the repository root (default `"."`). Used only when `include_git = TRUE`.

Value

A list with elements `timestamp` (POSIXct), `seed` (integer or NULL), `scenario` (as provided or NULL), `session_info` (list from sessionInfo() or NULL), `git_sha` (length-1 character or NA). All elements are serializable.

Examples

```
m <- ild_manifest()
names(m)
m <- ild_manifest(seed = 42, scenario = list(n_obs = 100, formula = "y ~ x"))
m$seed
m$scenario
```

`ild_meta` *Get ILD metadata attributes*

Description

Returns the metadata attributes set by [ild_prepare()]: user-facing id/time column names, gap threshold, n_units, n_obs, and spacing (descriptive stats only).

Usage

```
ild_meta(x)
```

Arguments

`x`	An ILD object (see [is_ild()]).

Value

A named list of metadata (ild_id, ild_time, ild_gap_threshold, ild_n_units, ild_n_obs, ild_spacing). `ild_spacing` includes overall stats and may contain `by_id`, a tibble of per-person spacing stats.

`ild_methods` *Generate methods-style narrative from provenance*

Description

Takes an ILD data object, a model fit, or a diagnostics object and produces a concise methods-style paragraph based on the recorded provenance (data preparation, centering, lagging, modeling, etc.).

Usage

```
ild_methods(x, robust_se = NULL, ...)
```

Arguments

`x`	An ILD object (see [is_ild()]), a model from [ild_lme()] or [ild_tvem()], a diagnostics object from [ild_diagnostics()], or another object with [ild_provenance()] (e.g. [ild_power()] result, [ild_missing_model()] result).
`robust_se`	Optional. If you reported fixed effects with cluster-robust SEs via [tidy_ild_model()] with `se = "robust"`, pass the type here (e.g. `"CR2"`) so the methods text can mention it.
`...`	Unused.

Value

A single character string (one or more sentences) suitable for a methods section. Use `cat()` or `print()` to display.

Examples

```
set.seed(1)
d <- ild_simulate(n_id = 5, n_obs_per = 6, seed = 1)
x <- ild_prepare(d, id = "id", time = "time")
x <- ild_center(x, y)
ild_methods(x)
```

`ild_missing_bias` *Test whether missingness is associated with a predictor (informative missingness)*

Description

Fits a logistic model of missingness (binary: is the outcome NA?) on a predictor variable. Use as a diagnostic: if the predictor is significant, missingness may be informative and results could be biased. This function does not correct for missingness; it flags the assumption for sensitivity analyses.

Usage

```
ild_missing_bias(x, outcome_var, predictor_var, random = FALSE)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`outcome_var`	Character. Name of the variable with missingness (e.g. `"mood"`).
`predictor_var`	Character. Name of the suspected predictor of missingness (e.g. `"stress"`).
`random`	Logical. If `TRUE`, fit a mixed-effects logistic model `is_missing ~ predictor + (1 | id)` via `lme4::glmer`; if `FALSE` (default), fit `glm(is_missing ~ predictor, family = binomial)`.

Value

A list with `predictor` (name), `estimate`, `std_error`, `p_value`, and `message` (short note about informative missingness).

Examples

```
set.seed(1)
d <- ild_simulate(n_id = 20, n_obs_per = 10, seed = 1)
d$stress <- rnorm(nrow(d))
d$mood <- d$y
d$mood[sample(nrow(d), 30)] <- NA  # some missing
x <- ild_prepare(d, id = "id", time = "time")
ild_missing_bias(x, "mood", "stress")
```

`ild_missing_model` *Fit a model for missingness (diagnostic / sensitivity)*

Description

Fits a logistic model predicting whether the outcome is missing from covariates. Use as a diagnostic; then [ild_ipw_weights()] to compute inverse-probability weights and [ild_ipw_refit()] for a sensitivity analysis. This is not a full MNAR solution—treat as diagnostic and sensitivity tooling.

Usage

```
ild_missing_model(
  x,
  outcome,
  predictors,
  random = FALSE,
  family = stats::binomial(),
  ...
)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`outcome`	Character. Name of the variable with missingness (e.g. `"mood"`).
`predictors`	Character vector. Names of covariates to predict missingness.
`random`	Logical. If `TRUE`, fit `glmer(is_missing ~ ... + (1|id))`; if `FALSE` (default), fit `glm(is_missing ~ ., family = binomial)`.
`family`	Passed to `glm` / `glmer` (default `binomial()`).
`...`	Passed to `glm()` or `lme4::glmer()`.

Value

A list with `fit` (the fitted model or `NULL` if no/all missing), `tidy` (tibble: term, estimate, std_error, p_value), `outcome`, `predictors`, and `message`. If `fit` is not `NULL`, `p_missing` is a numeric vector of predicted P(missing) per row (aligned to `x`).

Examples

```
set.seed(1)
d <- ild_simulate(n_id = 15, n_obs_per = 8, seed = 1)
d$stress <- rnorm(nrow(d))
d$mood <- d$y
d$mood[sample(nrow(d), 20)] <- NA
x <- ild_prepare(d, id = "id", time = "time")
mm <- ild_missing_model(x, "mood", "stress")
mm$tidy
```

`ild_missing_pattern` *Summarize missingness pattern in ILD*

Description

Returns a tabular summary of missingness by person and/or by variable, plus an optional heatmap plot. Complements [ild_summary()] and supports checking data before modeling. When `vars = NULL`, all non-internal data columns are used (observation presence across variables).

Usage

```
ild_missing_pattern(x, vars = NULL, max_ids = NULL, seed = NULL)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`vars`	Optional character vector of variable names to summarize. If `NULL`, all non-.ild_* data columns are used.
`max_ids`	Optional integer. If set, subset to this many persons (sampled) before computing `by_id`, `summary`, and `plot` to handle large N.
`seed`	Optional integer. Seed for sampling when `max_ids` is set.

Value

A list with: `summary` (tibble: one row per var, columns var, n_obs, n_na, pct_na), `plot` (ggplot2 object for missingness heatmap), `by_id`, `overall`, `n_complete`, `vars`.

`ild_person_distribution` *Plot distribution of person-level estimates from ild_person_model*

Description

Draws a histogram or density of the selected term's estimates across persons. Useful to visualize heterogeneity (e.g. distribution of slopes or intercepts).

Usage

```
ild_person_distribution(
  person_fit,
  term = NULL,
  type = c("histogram", "density")
)
```

Arguments

`person_fit`	Tibble returned by [ild_person_model()] (columns `term`, `estimate`, etc.).
`term`	Character. Which term to plot (e.g. `"(Intercept)"` or a covariate name). If `NULL`, the first term in the table is used.
`type`	Character. `"histogram"` (default) or `"density"`.

Value

A ggplot object.

`ild_person_model` *Fit a model separately per person (N-of-1 / idiographic)*

Description

Splits the ILD by person and fits the same formula (e.g. `lm`) within each. Returns a tibble of person-level estimates for teaching, N-of-1 analysis, or inspecting heterogeneity. Use [ild_person_distribution()] to visualize the distribution of estimates across persons.

Usage

```
ild_person_model(formula, data, method = c("lm"), min_obs = 2L)
```

Arguments

`formula`	A formula (e.g. `y ~ x`). Used for each person's `lm`.
`data`	An ILD object (see [is_ild()]).
`method`	Character. Currently only `"lm"` (default).
`min_obs`	Integer. Minimum observations per person to fit (default 2). Persons with fewer are omitted or get NA rows.

Value

A tibble with columns `.ild_id` (or the id column name from metadata), `term`, `estimate`, `std_error`, `p_value`, and optionally `sigma`, `n_obs`. One row per person per term (long format).

Examples

```
d <- ild_simulate(n_id = 5, n_obs_per = 8, seed = 1)
x <- ild_prepare(d, id = "id", time = "time")
pm <- ild_person_model(y ~ 1, x)
ild_person_distribution(pm, term = "(Intercept)")
```

`ild_plot` *ILD-specific plots*

Description

Produces trajectory (spaghetti), heatmap, gaps, and (if a fitted model is provided) fitted vs actual and residual ACF. Works for both lmerMod and lme (ild_lme with ar1 = TRUE).

Usage

```
ild_plot(
  x,
 type = c("trajectory", "heatmap", "gaps", "missingness", "fitted", "fitted_vs_actual",
   "residual_acf"),
  var = NULL,
  id_var = ".ild_id",
  time_var = c(".ild_time_num", ".ild_seq"),
  max_ids = 20L,
  seed = 42L,
  ...
)
```

Arguments

`x`	An ILD tibble or a fitted [ild_lme()] model.
`type`	Character (or vector). One or more of: ‘"trajectory"‘, ‘"heatmap"‘, ‘"gaps"‘, ‘"missingness"‘, ‘"fitted"‘ or ‘"fitted_vs_actual"‘ (requires fitted model), ‘"residual_acf"‘ (requires fitted model; ACF is over observation sequence, not adjusted for irregular time gaps). If length > 1, returns a named list of ggplots.
`var`	For ‘trajectory‘ or ‘heatmap‘, the variable to plot (optional; if missing and only one non-.ild_* column exists, it is used).
`id_var`	For trajectory, variable used for grouping (default ‘.ild_id‘).
`time_var`	For trajectory/gaps, x-axis: ‘.ild_time_num‘ or ‘.ild_seq‘.
`max_ids`	For trajectory, max number of persons to plot (sampled if larger; default 20). Set to ‘Inf‘ to plot all.
`seed`	Integer. Seed for sampling ids when ‘max_ids‘ is set (default 42).
`...`	Unused.

Value

A single ggplot when ‘length(type) == 1‘, or a named list of ggplots when ‘length(type) > 1‘.

Examples

```
x <- ild_prepare(ild_simulate(n_id = 3, n_obs_per = 6, seed = 1), id = "id", time = "time")
fit <- ild_lme(y ~ 1 + (1 | id), data = x, ar1 = FALSE, warn_no_ar1 = FALSE)
ild_plot(fit, type = "fitted_vs_actual")
ild_plot(fit, type = c("fitted_vs_actual", "residual_acf"))
```

`ild_power` *Simulation-based power analysis for a fixed effect in ILD models*

Description

Estimates empirical power by repeatedly simulating data with a known effect (via [ild_simulate()] plus one added predictor), fitting with [ild_lme()], and counting the proportion of runs where the target term is significant (Wald p < alpha). The workflow (simulate, fit, reject/retain) mirrors simulation-based power in packages like mixpower; `ild_power()` is focused on ILD and `ild_lme()`. For multi-parameter grids, LRT, or general LMMs, consider mixpower.

Usage

```
ild_power(
  formula,
  n_sim = 500L,
  n_id,
  n_obs_per,
  effect_size,
  test_term = NULL,
  alpha = 0.05,
  ar1 = FALSE,
  seed = 42L,
  return_sims = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`formula`	Fixed-effects formula including the predictor to power for and random effects, e.g. `y ~ x + (1 | id)`. For `ar1 = TRUE`, use a fixed-only formula (random passed internally).
`n_sim`	Integer. Number of simulation replications (default 500).
`n_id`	Integer. Number of persons per replication.
`n_obs_per`	Integer. Observations per person per replication.
`effect_size`	Numeric. True coefficient for `test_term` in the DGP.
`test_term`	Character or `NULL`. Which fixed-effect term to test. If `NULL`, taken as the first non-intercept fixed-effect term from the model (inferred from the formula).
`alpha`	Numeric. Significance level for rejection (default 0.05).
`ar1`	Logical. If `TRUE`, fit with nlme and residual AR1/CAR1 (default `FALSE`).
`seed`	Integer. Base random seed; replication `i` uses `seed + i`.
`return_sims`	Logical. If `TRUE`, include a tibble of per-run estimate, std_error, p_value, rejected in the result (default `FALSE`).
`verbose`	Logical. If `TRUE`, message progress (default `TRUE`).
`...`	Passed to [ild_simulate()] (e.g. `irregular`, `wp_effect`, `bp_effect`) and to [ild_lme()].

Details

The data-generating process adds one predictor (name from `test_term`) as standard normal and adds `effect_size * predictor` to the outcome on top of the base [ild_simulate()] DGP (id, time, y). No change to `ild_simulate()` is required.

For `ar1 = FALSE` (lmer), the lme4 backend does not report p-values; inference for the test term uses a Wald z-approximation (estimate / SE) so that power is still computed. For `ar1 = TRUE` (nlme), p-values come from the model summary.

Value

A list: `power` (proportion of converged runs with p < alpha), `n_sim`, `n_reject`, `n_converged`, `n_failed`, `alpha`, `test_term`. If `return_sims = TRUE`, also `sim_results` (tibble of per-run results).

Examples

```
set.seed(42)
res <- ild_power(
  formula = y ~ x + (1 | id),
  n_sim = 25L,
  n_id = 15L,
  n_obs_per = 10L,
  effect_size = 0.3,
  seed = 42L,
  verbose = FALSE
)
res$power
res$n_reject
```

`ild_prepare` *Prepare a data frame as an ILD (intensive longitudinal data) object*

Description

Validates and encodes longitudinal structure: parses time, sorts by id and time, handles duplicate timestamps, and adds internal columns (‘.ild_*‘) and metadata. All downstream functions assume the result of ‘ild_prepare()‘.

Usage

```
ild_prepare(
  data,
  id,
  time,
  gap_threshold = Inf,
  duplicate_handling = c("first", "last", "error", "collapse"),
  collapse_fn = NULL
)
```

Arguments

`data`	A data frame or tibble with at least an id and a time column.
`id`	Character. Name of the subject/unit identifier column.
`time`	Character. Name of the time column (Date, POSIXct, or numeric).
`gap_threshold`	Numeric. Time distance above which an interval is flagged as a gap (‘.ild_gap‘ TRUE). Same units as the numeric time (e.g. seconds if time is POSIXct). Use ‘Inf‘ to disable gap flagging.
`duplicate_handling`	Character. How to handle duplicate timestamps within the same id: ‘"first"‘ (keep first), ‘"last"‘ (keep last), ‘"error"‘ (stop with an error), ‘"collapse"‘ (aggregate with `collapse_fn`).

`collapse_fn`	Named list of functions, one per variable to collapse. Used only when `duplicate_handling = "collapse"`. E.g. `list(x = mean, y = function(z) z[1])`. Variables not in `collapse_fn` keep their first value within the duplicate group.

Value

An ILD tibble with '.ild_*' columns and metadata attributes. Spacing metadata (see [ild_meta()]) includes overall stats and a `by_id` tibble of per-person spacing stats (median_dt, iqr_dt, n_intervals, pct_gap). Use [ild_summary()] to inspect and check gap flags before modeling.

`ild_provenance` *Return the raw provenance object*

Description

For ILD data: returns `attr(x, "tidyILD")$provenance` (version + steps from preprocessing). For analysis objects (e.g. fits from [ild_lme()], [ild_diagnostics()], [ild_tvem()], [ild_power()], [ild_missing_model()]): returns `attr(x, "ild_provenance")`, which has `source_data_provenance` (snapshot of data provenance) and `analysis_steps` (list of analysis step records).

Usage

```
ild_provenance(x)
```

Arguments

`x`	An ILD object (see [is_ild()]) or an analysis object with `ild_provenance` attribute.

Value

For data: list with `version` and `steps`. For analysis: list with `version`, `source_data_provenance`, `analysis_steps`. `NULL` if none.

`ild_report` *Assemble a light report from a model fit*

Description

Builds a structured list with methods narrative (from [ild_methods()]), fixed-effects table (from [tidy_ild_model()]), a short diagnostics summary, and the raw provenance. Optionally exports provenance to a file.

Usage

```
ild_report(fit, export_provenance_path = NULL, robust_se = NULL, ...)
```

Arguments

`fit`	A fitted model from [ild_lme()] (or a list with a `fit` component, e.g. from [ild_crosslag()]).
`export_provenance_path`	Optional. If provided, [ild_export_provenance()] is called to write provenance to this path; the path is included in the returned list.
`robust_se`	Optional. Passed to [ild_methods()] when building the methods text (e.g. `"CR2"` if you used [tidy_ild_model()] with `se = "robust"`).
`...`	Unused.

Value

A list with a stable schema: `meta` (list with `n_obs`, `n_id`, `engine` when available), `methods`, `model_table`, `diagnostics_summary`, `provenance`, `provenance_export_path` (character or NULL).

Examples

```
set.seed(1)
x <- ild_prepare(ild_simulate(n_id = 5, n_obs_per = 6, seed = 1), id = "id", time = "time")
fit <- ild_lme(y ~ 1 + (1 | id), data = x, ar1 = FALSE, warn_no_ar1 = FALSE)
r <- ild_report(fit)
r$methods
r$model_table
```

`ild_robust_se` *Cluster-robust variance-covariance matrix for ILD model fits*

Description

Computes cluster-robust (sandwich) variance estimators with small-sample corrections via the **clubSandwich** package. Use with [tidy_ild_model()] via `se = "robust"` for fixed-effect inference. Requires `attr(fit, "ild_data")`; refit with [ild_lme()] if missing.

Usage

```
ild_robust_se(
  fit,
  type = c("CR2", "CR3", "CR0"),
  cluster = c("id", "data"),
  cluster_vec = NULL,
  ...
)
```

Arguments

`fit`	A fitted model from [ild_lme()] (lmerMod or lme).
`type`	Character. Correction type: `"CR2"` (recommended), `"CR3"`, or `"CR0"`.
`cluster`	Either `"id"` (default; cluster by the ILD id column from `ild_data`) or `"data"` to use a user-supplied vector via `cluster_vec`.
`cluster_vec`	When `cluster = "data"`, a vector of cluster IDs aligned to the model rows (same length and order as `attr(fit, "ild_data")`).
`...`	Passed to `clubSandwich::vcovCR()`.

Value

A list with `vcov` (matrix), `type`, `cluster_name`, `engine` (`"lmer"` or `"lme"`), and optionally `message` if a fallback was used (e.g. lme not fully supported on this build).

See Also

[tidy_ild_model()] with `se = "robust"`, `clubSandwich::vcovCR`.

Examples

```
if (requireNamespace("clubSandwich", quietly = TRUE)) {
  set.seed(1)
  dat <- ild_simulate(n_id = 8, n_obs_per = 6, seed = 1)
  dat <- ild_prepare(dat, id = "id", time = "time")
  dat <- ild_center(dat, y)
  fit <- ild_lme(y ~ y_bp + y_wp + (1 | id), data = dat, ar1 = FALSE, warn_no_ar1 = FALSE)
  rv <- ild_robust_se(fit, type = "CR2")
  rv$engine
  dim(rv$vcov)
}
```

`ild_simulate` *Simulate simple ILD for examples, tests, and power analysis*

Description

Generates a tibble with id, time, and outcome `y`. Optionally uses AR(1) within-person correlation and configurable WP/BP variance. Use [ild_prepare()] after to get a proper ILD object.

Usage

```
ild_simulate(
  n_id = 5L,
  n_obs_per = 10L,
  n_time = NULL,
  irregular = FALSE,
  ar1 = NULL,
  wp_effect = 0.5,
  bp_effect = 1,
  seed = 42L
)
```

Arguments

n_id	Integer. Number of persons (default 5).
n_obs_per	Integer. Observations per person (default 10).
n_time	Integer. Alias for n_obs_per (observations per person). If provided, overrides n_obs_per.
irregular	Logical. If TRUE, add random jitter to time (default FALSE).
ar1	Numeric or NULL. If numeric, within-person AR(1) correlation (e.g. 0.4). If NULL or 0, no AR (default NULL).
wp_effect	Numeric. Scale (SD) of within-person innovation (default 0.5).
bp_effect	Numeric. Scale (SD) of between-person random intercept (default 1).
seed	Integer. Random seed for reproducibility (default 42).

Value

A data frame with columns `id`, `time` (POSIXct), and `y`.

Examples

```
d <- ild_simulate(n_id = 3, n_obs_per = 5, seed = 1)
x <- ild_prepare(d, id = "id", time = "time")
d2 <- ild_simulate(n_id = 100, n_time = 50, ar1 = 0.4, wp_effect = 0.6,
  bp_effect = 0.3, irregular = TRUE, seed = 1)
```

---

ild_spacing   *Spacing diagnostics and correlation-structure recommendation*

Description

Reports observation intervals in human-friendly units (e.g. hours) and recommends AR1 vs CAR1 for use in [ild_lme()]. Surfaces the same logic that `ild_lme(..., ar1 = TRUE)` uses internally so users can see why a correlation structure was chosen.

Usage

```
ild_spacing(x, gap_large_hours = 12)
```

Arguments

x	An ILD object (see [is_ild()]).
gap_large_hours	Numeric. Intervals (in hours) above which to count as "large gaps" for `large_gaps_pct` (default 12). Ignored if time is not in seconds (e.g. numeric day indices).

Value

A list with `median_interval` (hours), `iqr` (hours), `large_gaps_pct` (percent of intervals > `gap_large_hours`), `coefficient_of_variation`, `recommendation` (character: use CAR1 or AR1), and `spacing_class` (`regular-ish` or `irregular-ish`).

Examples

```
d <- ild_simulate(n_id = 5, n_obs_per = 10, irregular = TRUE, seed = 1)
x <- ild_prepare(d, id = "id", time = "time", gap_threshold = 7200)
ild_spacing(x)
```

`ild_spacing_class` *Classify spacing as regular-ish vs irregular-ish*

Description

Returns a simple classification for use in documentation or when choosing correlation structure (e.g. AR1 vs CAR1 in [ild_lme()]). The rule is documented and overridable via arguments. Does not change core ILD behavior.

Usage

```
ild_spacing_class(x, cv_threshold = 0.2, pct_gap_threshold = 10)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`cv_threshold`	Numeric. Coefficient of variation of within-person intervals above which spacing is "irregular-ish" (default 0.2).
`pct_gap_threshold`	Numeric. Percent of intervals flagged as gaps above which spacing is "irregular-ish" (default 10).

Value

Character: ‘"regular-ish"‘ or ‘"irregular-ish"‘.

ild_spaghetti — *ILD spaghetti / person trajectories (alias for ild_plot with type = "trajectory")*

Description

Line plot of variable over time, one line per person. See [ild_plot()].

Usage

```
ild_spaghetti(x, var = NULL, ...)
```

Arguments

`x`	ILD object or fitted model.
`var`	Variable to plot. If NULL, single data column is used.
`...`	Passed to [ild_plot()] (e.g. max_ids, seed, id_var, time_var).

Value

A ggplot object.

ild_summary — *One-shot summary of an ILD object*

Description

Reports number of persons, number of observations, time range, descriptive spacing (median/IQR of intervals, percent gaps), and duplicate info. Uses [ild_meta()] and '.ild_*' columns only. No hard "regular"/"irregular" label; use [ild_spacing_class()] for that.

Usage

```
ild_summary(x)
```

Arguments

`x`	An ILD object (see [is_ild()]).

Value

A list with elements: `summary` (one-row tibble with n_id, n_obs, time_min, time_max, prop_gap, median_dt_sec, iqr_dt_sec), `n_units`, `n_obs`, `time_range`, `spacing`, `n_gaps`, `pct_gap`. The `summary` tibble is the primary contract for programmatic use.

ild_tvem *Fit a time-varying effects model (TVEM) for ILD*

Description

Fits a GAM with a smooth in time and a time-varying coefficient for a predictor using [mgcv::gam()]. Use [ild_tvem_plot()] to plot the time-varying effect. Requires `.ild_time_num` (or a numeric time column).

Usage

```
ild_tvem(
  data,
  outcome,
  predictor,
  time_var = ".ild_time_num",
  k = 10L,
  re_id = TRUE,
  ...
)
```

Arguments

`data`	An ILD object (see [is_ild()]).
`outcome`	Character. Name of the outcome variable.
`predictor`	Character. Name of the predictor with a time-varying effect.
`time_var`	Character. Name of the time variable (default `".ild_time_num"`).
`k`	Integer. Basis dimension for smooth terms (default 10).
`re_id`	Logical. If `TRUE` (default), include a random intercept by person (`s(.ild_id, bs="re")`).
`...`	Passed to [mgcv::gam()].

Value

A fitted `gam` object with class `c("tidyild_tvem", "gam", ...)` and attribute `ild_tvem_meta` (list with `outcome`, `predictor`, `time_var`, `k`, `re_id`).

Examples

```
set.seed(1)
d <- ild_simulate(n_id = 10, n_obs_per = 15, seed = 1)
d$x <- rnorm(nrow(d))
x <- ild_prepare(d, id = "id", time = "time")
tv <- ild_tvem(x, "y", "x", k = 5, re_id = TRUE)
ild_tvem_plot(tv)
```

ild_tvem_plot *Plot the time-varying coefficient from a TVEM fit*

Description

Builds a line plot of the smooth term for the time-varying effect of the predictor (with optional confidence band). Uses a grid over the time variable and [mgcv::predict.gam()] with `type = "terms"`.

Usage

```
ild_tvem_plot(tvem_fit, n_grid = 100L, level = 0.95)
```

Arguments

`tvem_fit`	A fitted object from [ild_tvem()] (class `tidyild_tvem`).
`n_grid`	Integer. Number of points over the time range for the curve (default 100).
`level`	Numeric. Confidence level for the band (default 0.95).

Value

A `ggplot` object (time on x-axis, estimated effect on y-axis).

Examples

```
set.seed(1)
d <- ild_simulate(n_id = 10, n_obs_per = 15, seed = 1)
d$x <- rnorm(nrow(d))
x <- ild_prepare(d, id = "id", time = "time")
tv <- ild_tvem(x, "y", "x", k = 5, re_id = TRUE)
ild_tvem_plot(tv)
```

is_ild *Check if an object is a valid ILD tibble*

Description

Returns TRUE if the object has all required ‘.ild_*‘ columns and ‘ild_*‘ metadata attributes (as set by [ild_prepare()]).

Usage

```
is_ild(x)
```

Arguments

`x`	Any object.

Value

Logical.

`plot_ild_diagnostics` *Plot diagnostics from an ild_diagnostics object*

Description

Generates ggplot objects for the requested diagnostic types. Plots are not stored in the diagnostics object; call this function to create them.

Usage

```
plot_ild_diagnostics(diag, type = NULL)
```

Arguments

`diag`	An object returned by [ild_diagnostics()].
`type`	Character vector. Which plots to build (default: the types stored in `diag$meta$type`).

Value

A named list of ggplot objects (e.g. `residual_acf`, `residuals_vs_fitted`, `residuals_vs_time`, `qq`).

`tidy_ild_model` *Tidy fixed effects from an ILD model fit*

Description

Returns a tibble of fixed-effect estimates with consistent columns for both lmer and lme engines: `term`, `estimate`, `std_error`, `ci_low`, `ci_high`, `p_value`. With `object` = `TRUE`, returns an object of class `tidyild_model` (meta + table) for use with `print.tidyild_model`.

Usage

```
tidy_ild_model(
  fit,
  conf_level = 0.95,
  object = FALSE,
  se = c("model", "robust"),
  robust_type = c("CR2", "CR3", "CR0"),
  ...
)
```

Arguments

`fit`	A fitted model from [ild_lme()] (lmerMod or lme).
`conf_level`	Numeric. Confidence level for intervals (default 0.95).
`object`	Logical. If `TRUE`, return a list with `meta` and `table` and class `tidyild_model` for polished printing (default `FALSE`).
`se`	Character. `"model"` (default) uses the model's standard errors; `"robust"` uses cluster-robust SEs from [ild_robust_se()] (requires **clubSandwich**).
`robust_type`	Character. When `se = "robust"`, the correction type: `"CR2"` (recommended), `"CR3"`, or `"CR0"`.
`...`	Passed to [ild_robust_se()] when `se = "robust"` (e.g. `cluster`, `cluster_vec`).

Value

A tibble, or when `object = TRUE` a list of class `tidyild_model`.

Model-based vs robust SE

With `se = "model"` (default), standard errors and CIs come from the fitted model. With `se = "robust"`, cluster-robust (sandwich) SEs are used; CIs and p-values are based on a Wald normal approximation. Install the **clubSandwich** package to use robust SEs.

`validate_ild` *Validate an ILD object and error if invalid*

Description

Checks presence and types of '.ild_*' columns and 'ild_*' attributes. Errors with a clear message if anything is missing or invalid. Calls `ild_normalize_internal()` so legacy objects get `attr(x, "tidyILD")` and class `tidyild_df`.

Usage

```
validate_ild(x)
```

Arguments

`x`	Object to validate (expected to be an ILD tibble).

Value

Invisibly returns `x` if valid.

Index