Package: tidyILD (via r-universe)

October 16, 2026

Title Tidy Intensive Longitudinal Data Analysis

Version 0.4.1

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.

URL https://github.com/alitovchenko/tidyILD

BugReports https://github.com/alitovchenko/tidyILD/issues

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

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

Suggests BH, testthat (>= 3.0.0), roxygen2, knitr, rmarkdown, rstan, RcppEigen, broom.mixed, clubSandwich, jsonlite, yaml, tsibble, brms, KFAS, ctsem

VignetteBuilder knitr

Collate 'package.R' 'ild-class.R' 'ild_provenance.R' 'ild_methods.R' 'ild_compare_pipelines.R' 'ild_compare_fits.R' 'utils.R' 'ild_schema_tidy_augment.R' 'ild_tidy_helpers.R' 'ild_augment_helpers.R' 'ild_tsibble.R' 'ild_prepare_wide.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_extensions.R' 'ild_missing_pattern.R' 'ild_missing_bias.R' 'ild_missing_model.R' 'ild_ipw_helpers.R' 'ild_ipw_weights.R' 'ild_iptw_weights.R' 'ild_iptw_msm_weights.R' 'ild_ipcw_weights.R' 'ild_joint_msm_weights.R' 'ild_ipw_refit.R' 'ild_msm_bootstrap.R' 'ild_msm_balance.R'

'ild_tvem.R' 'ild_tvem_plot.R' 'ild_check_lags.R'
'ild_panel_lag_prepare.R' 'ild_msm_history.R' 'ild_crosslag.R'
'ild_acf.R' 'ild_align.R' 'ild_lme.R' 'ild_person_model.R'
'ild_heterogeneity.R' 'ild_robust_se.R' 'ild_model_tidiers.R'
'ild_guardrail_registry.R' 'ild_missingness_report.R'
'ild_diagnostics_bundle.R' 'ild_diagnostics_utilities.R'
'ild_diagnostics.R' 'ild_manifest.R' 'ild_plot.R'
'ild_generics.R' 'ild_brms.R' 'ild_brms_s3.R'
'ild_brms_dynamics_formula.R' 'ild_ctsem.R' 'ild_ctsem_s3.R'
'ild_kfas_spec.R' 'ild_kfas_helpers.R' 'ild_kfas.R'
'ild_kfas_guardrails.R' 'ild_kfas_tidiers.R'
'ild_kfas_augment.R' 'ild_kfas_diagnose.R' 'ild_kfas_plot.R'
'ild_kfas_autoplot.R' 'ild_diagnose_fillers.R'
'ild_diagnose_methods.R' 'ild_autoplot_bundle.R'
'ild_circadian.R' 'ild_simulate.R' 'ild_msm_estimand.R'
'ild_msm_fit.R' 'ild_msm_contrast.R' 'ild_msm_simulation.R'
'ild_power.R' 'ild_recovery_metrics.R' 'data.R' 'broom.R'

RoxygenNote 7.3.3

Config/pak/sysreqs cmake make

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

Date/Publication 2026-06-12 18:48:00 UTC

RemoteUrl https://github.com/alitovchenko/tidyild

RemoteRef HEAD

RemoteSha 74c1166ed2e0d8dda8403c58948a15bd256f059c

Contents

tidyILD-package 4
as_ild 7
augment_ild_model 8
broom_ild_lme 8
ema_example 9
guardrail_registry 9
ild_acf 10
ild_align 10
ild_as_tsibble 12
ild_augment 12
ild_augment_schema 13
ild_augment_states 14
ild_autoplot 15
ild_brms 16
ild_brms_dynamics_formula 17
ild_build_msm_history 17
ild_bundle 18
ild_center 19
ild_center_plot 20

ild_check_lags . . . 20
ild_circadian . . . 21
ild_compare_fits . . . 22
ild_compare_pipelines . . . 22
ild_crosslag . . . 23
ild_ctsem . . . 24
ild_decomposition . . . 25
ild_design_check . . . 26
ild_diagnose . . . 27
ild_diagnostics . . . 30
ild_diagnostics_bundle . . . 31
ild_diagnostics_utilities . . . 33
ild_export_provenance . . . 34
ild_fit . . . 34
ild_heatmap . . . 35
ild_heterogeneity . . . 36
ild_history . . . 38
ild_ipcw_weights . . . 38
ild_iptw_msm_weights . . . 40
ild_iptw_weights . . . 42
ild_ipw_ess . . . 43
ild_ipw_refit . . . 44
ild_ipw_weights . . . 45
ild_joint_msm_weights . . . 46
ild_kfas . . . 47
ild_lag . . . 48
ild_lme . . . 49
ild_manifest . . . 51
ild_meta . . . 52
ild_methods . . . 52
ild_missing_bias . . . 53
ild_missing_cohort . . . 54
ild_missing_compliance . . . 55
ild_missing_hazard_first . . . 56
ild_missing_model . . . 57
ild_missing_pattern . . . 58
ild_missingness_report . . . 59
ild_msm_balance . . . 61
ild_msm_bootstrap . . . 62
ild_msm_contrast_over_time . . . 63
ild_msm_diagnose . . . 64
ild_msm_estimand . . . 65
ild_msm_fit . . . 66
ild_msm_history_spec . . . 68
ild_msm_overlap_plot . . . 69
ild_msm_recovery . . . 70
ild_msm_simulate_scenario . . . 71
ild_panel_lag_prepare . . . 72

ild_person_distribution . . . 73
ild_person_model . . . 73
ild_plot . . . 74
ild_plot_filtered_vs_smoothed . . . 75
ild_plot_forecast . . . 76
ild_plot_predicted_trajectory . . . 76
ild_plot_states . . . 77
ild_power . . . 77
ild_prepare . . . 79
ild_prior_ild . . . 81
ild_provenance . . . 82
ild_recovery_metrics . . . 82
ild_report . . . 83
ild_robust_se . . . 84
ild_simulate . . . 85
ild_spacing . . . 86
ild_spacing_class . . . 87
ild_spaghetti . . . 87
ild_summary . . . 88
ild_tidy . . . 89
ild_tidy_schema . . . 89
ild_tidy_states . . . 90
ild_tsibble_meta . . . 91
ild_tvem . . . 91
ild_tvem_plot . . . 92
is_ild . . . 93
plot_ild_diagnostics . . . 93
tidy_ild_model . . . 94
tidy_ild_msm_bootstrap . . . 95
validate_ild . . . 96

Index **97**

`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. Estimands that require **joint multivariate dynamics**, **penalized high-dimensional** longitudinal models, or full **latent-variable DSEM** are better handled by specialist packages after preprocessing; see `vignette("ild-specialist-backends", package = "tidyILD")`.

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` or `ild_fit` (same engines plus optional `backend = "brms"`), then `ild_diagnostics` or `ild_plot`. State-space: `ild_kfas`. See the examples below.

Function index by topic

Contracts (schemas) `ild_diagnostics_bundle`, `guardrail_registry`, `ild_tidy_schema`, `ild_augment_schema`

Setup and validation `ild_prepare`, `ild_as_tsibble`, `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_compliance`, `ild_missing_cohort`, `ild_missing_hazard_first`, `ild_missingness_report`, `ild_missing_bias`, `ild_missing_model`, `ild_ipw_weights`, `ild_ipw_refit`, `ild_plot` (types: trajectory, gaps, missingness, predicted_trajectory). These overlap with `ild_diagnostics_utilities` (bundle section providers).

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

Modeling `ild_fit` (unified: lme4, nlme, brms), `ild_lme`, `ild_brms` (Bayesian), `ild_kfas` (KFAS state-space; not via `ild_fit`), `ild_person_model`, `ild_heterogeneity`, `ild_heterogeneity_stratified`, `ild_tvem` (time-varying effects)

Diagnostics and visualization `ild_diagnose` (`ild_diagnostics_bundle`), `ild_diagnostics_utilities`, `ild_acf`, `ild_diagnostics`, `ild_plot` (types: fitted, predicted_trajectory, residual_acf; optional `facet_by`), `ild_plot_predicted_trajectory`, `ild_heatmap`, `ild_spaghetti`, `ild_circadian`, `ild_tvem_plot`

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

Reproducibility `ild_manifest`, `ild_bundle`

Package standards (developers) `vignette("developer-contracts", package = "tidyILD")`; normative spec: `system.file("dev", "DEVELOPER_CONTRACTS.md", package = "tidyILD")`

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

Person-level `ild_person_model`, `ild_person_distribution`

Model tidiers `ild_prior_ild`, `ild_tidy`, `ild_augment`, `ild_diagnose`, `ild_autoplot`, `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`)

Visualization

ild_plot, ild_spaghetti, and ild_heatmap cover trajectories, heatmaps, gaps, and missingness; optional facet_by adds panels (e.g. by cluster). For observed and fitted lines over time, use ild_plot_predicted_trajectory or ild_plot(..., type = "predicted_trajectory"). After ild_diagnose, use ild_autoplot on the bundle. Backend-specific plots include ild_tvem_plot, ild_plot_states, ild_plot_forecast, and ild_plot_filtered_vs_smoothed (KFAS). See vignette("visualization-in-tidyILD", package = "tidyILD") for a question-to-function map and recipes (including partial effects via **marginaleffects** / **ggeffects** on _wp / _bp columns).

Vignettes

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

- *From raw data to model with tidyILD* — Full pipeline: prepare, inspect, center, lag, fit, diagnose.
- *Visualization in tidyILD* — Index of plots, bundle sections, facet_by, predicted trajectories, and partial-effects templates.
- *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.
- *Temporal dynamics: choosing a model for ILD* — Maps estimands to lags, residual AR (ild_lme), time-varying effects (ild_tvem), and state-space backends (ild_kfas, ild_ctsem).
- *Specialist backends: when to move beyond the default stack* — Handoffs to dynamite, PGEE, DSEM, and multivariate workflows; export patterns after prepare/center/lag.
- *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.
- *Heterogeneity and person-specific effects* — BLUPs vs ild_person_model, ild_heterogeneity(), bundle plots.
- *Missingness in ILD: diagnostics and sensitivity routes* — ild_missingness_report(), MAR/MNAR context, IPW templates.

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). ild_heterogeneity summarizes partial-pooling person-specific effects from mixed models (contrasts with no pooling).

Author(s)

Alex Litovchenko <al4877@columbia.edu>

See Also

browseVignettes and vignette(package = "tidyILD") for vignettes. Core entry points: ild_prepare, ild_lme. vignette("ild-specialist-backends", package = "tidyILD") for using tidyILD as a preprocessing layer with external multivariate or high-dimensional estimators. 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 conforming to ild_augment_schema: .ild_id, .ild_time, .outcome, .fitted, .resid, .resid_std, engine, model_class, plus optional columns (see schema). .resid_std is Pearson-type when residuals(fit, type = "pearson") is available and length-matched; otherwise NA (principled but sparse). Used internally by [ild_diagnostics()] and [ild_plot()]. Requires attr(fit, "ild_data"); refit with [ild_lme()] if missing.

Usage

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

Arguments

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

Value

A tibble; see ild_augment_schema.

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.

guardrail_registry *Guardrail rule registry (analysis safety layer)*

Description

Returns the canonical catalog of **guardrail** rules tidyILD may use. Each rule has a stable `rule_id`, `section` (bundle slot), `severity`, and default `default_message` / `default_recommendation` text. When you call `ild_diagnose`, **triggered** rules appear as rows in `ild_diagnostics_bundle$guardrails` with `message` and `recommendation` possibly customized for the run.

Usage

```
guardrail_registry()
```

Value

A tibble with columns `rule_id`, `section`, `severity`, `default_message`, `default_recommendation`.

See Also

`ild_diagnose`, `ild_diagnostics_bundle`

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_as_tsibble *Convert an ILD object to a tsibble*

Description

Wraps [tsibble::as_tsibble()] using the subject and time column names from [ild_meta()] as key and index. When [ild_prepare()] recorded tsibble provenance (`ild_tsibble_meta()` is non-`NULL`), the `regular` argument is set from the stored `is_regular` flag so the reconstructed `interval` often matches the original for unchanged data. Otherwise defaults to `regular = TRUE`.

Usage

```
ild_as_tsibble(x, ...)
```

Arguments

`x`	An object that passes [validate_ild()].
`...`	Optional arguments passed to [tsibble::as_tsibble()] (e.g. override `regular`).

Value

A `tbl_ts` object (see **tsibble**).

See Also

```
vignette("tsibble-interoperability", package = "tidyILD")
```

Examples

```
d <- ild_simulate(n_id = 4, n_obs_per = 5, seed = 1)
x <- ild_prepare(d, id = "id", time = "time")
if (requireNamespace("tsibble", quietly = TRUE)) {
  ts <- ild_as_tsibble(x)
  class(ts)
}
```

ild_augment *Augment observations with fitted values and residuals (S3 generic)*

Description

Dispatches to [augment_ild_model()] for `lmerMod` and `lme` objects from [ild_lme()], or the `brmsfit` method for [ild_brms()]. All methods return tables conforming to `ild_augment_schema`.

Usage

```
ild_augment(x, ...)

## S3 method for class 'lmerMod'
ild_augment(x, ...)

## S3 method for class 'lme'
ild_augment(x, ...)

## Default S3 method:
ild_augment(x, ...)

## S3 method for class 'brmsfit'
ild_augment(x, summary = TRUE, ...)

## S3 method for class 'ild_fit_ctsem'
ild_augment(x, ...)

## S3 method for class 'ild_fit_kfas'
ild_augment(x, ...)
```

Arguments

x	A fitted model from [ild_lme()].
...	Passed to [augment_ild_model()].
summary	Logical. For brmsfit, use posterior predictive summary for fitted values (default TRUE).

---

ild_augment_schema *Column dictionary for* ild_augment() *outputs*

Description

Defines the minimum stable contract for augmented observation-level tables. augment_ild_model and ild_augment methods emit all required columns; see augment_ild_model for .resid_std semantics.

Usage

```
ild_augment_schema()
```

Details

Required columns

.ild_id	Person identifier (from ILD)

.ild_time	Time variable
.outcome	Observed outcome (numeric vector aligned to rows)
.fitted	Fitted / posterior mean fitted value
.resid	Raw residual
.resid_std	Pearson-type residual when residuals(fit, type = "pearson") is available and length-matched; otherwise
engine	Character engine id
model_class	Model class string

Optional: .fitted_lower, .fitted_upper, .influence, .state, .state_lower, .state_upper.

Value

A list with required and optional character vectors.

See Also

ild_tidy_schema, ild_diagnostics_bundle. Reserved prefixes and migration notes: vignette("developer-contracts package = "tidyILD").

---

ild_augment_states *Per-row latent states (long format) for complex multi-state models*

Description

For local_level only, use [ild_augment()] which already provides .state for the level. This function is reserved for future multi-state models.

Usage

```
ild_augment_states(x, ...)
```

Arguments

x	An object from [ild_kfas()].
...	Reserved.

Value

A tibble.

`ild_autoplot` *Autoplot for ILD model fits or diagnostics (S3 generic)*

Description

For `ild_diagnostics_bundle` from [ild_diagnose()], dispatches by `section` and `type` (section-first; see `?ild_autoplot` method for `ild_diagnostics_bundle`). For [ild_diagnostics()], calls [plot_ild_diagnostics()] directly. For `lmerMod` or `lme` fits from [ild_lme()], calls [ild_plot()] (default `type = "fitted_vs_actual"`).

Usage

```
ild_autoplot(x, ...)

## S3 method for class 'ild_diagnostics'
ild_autoplot(x, ...)

## S3 method for class 'lmerMod'
ild_autoplot(x, type = "fitted_vs_actual", ...)

## S3 method for class 'lme'
ild_autoplot(x, type = "fitted_vs_actual", ...)

## Default S3 method:
ild_autoplot(x, ...)

## S3 method for class 'brmsfit'
ild_autoplot(x, type = "pp_check", ...)

## S3 method for class 'ild_fit_ctsem'
ild_autoplot(x, type = c("fitted_vs_actual", "residual_time", "qq"), ...)

## S3 method for class 'ild_fit_kfas'
ild_autoplot(x, type = c("states", "innovations", "signal"), ...)

## S3 method for class 'ild_diagnostics_bundle'
ild_autoplot(x, section = "residual", type = NULL, ...)
```

Arguments

`x`	A bundle, `ild_diagnostics` object, or a fitted model with `ild_data`.
`...`	Passed to [plot_ild_diagnostics()] (`ild_diagnostics` objects, or bundle `section = "residual"` with `type = NULL` and legacy diagnostics), to [ild_plot()] (`lmerMod`/`lme`), or to `brms::pp_check()` (bundle `section = "predictive"`, `type = "ppc"`).
`type`	Character: `"states"` (smoothed level vs time), `"innovations"` (standardized), or `"signal"` (observation expectation vs outcome).

section	For ild_diagnostics_bundle only: "residual", "fit", "predictive", "data", "design", or "causal".

---

ild_brms *Fit a Bayesian mixed model to ILD data with* **brms**

Description

Wraps [brms::brm()] with ILD validation, default priors for common longitudinal mixed models, standardized posterior metadata in attr(fit, "ild_posterior"), and analysis provenance (priors, chains, iterations). Attach ild_data for use with [ild_tidy()], [ild_augment()], [ild_diagnose()], and [ild_methods()].

Usage

```
ild_brms(
  formula,
  data,
  prior = NULL,
  prior_template = c("default", "weakly_informative", "minimal_shrinkage"),
  warn_uncentered = TRUE,
  ...
)
```

Arguments

formula	A **brms / lme4**-style formula (e.g. y ~ x + (1|id)).
data	An ILD object from [ild_prepare()].
prior	Optional brmsprior object. If NULL, built from prior_template.
prior_template	Passed to [ild_prior_ild()] when prior is NULL.
warn_uncentered	If TRUE (default), warn when a numeric predictor varies within and between persons without _wp/_bp (same idea as [ild_lme()]).
...	Arguments passed to [brms::brm()] (e.g. chains, iter, warmup, control, backend, seed).

Details

Requires packages **brms** and a Stan backend (**rstan** or **cmdstanr**).

Value

A brmsfit object with attributes:

ild_data The ILD data frame.

ild_posterior List: sampler settings, prior summary, divergences, etc.

ild_provenance Standard tidyILD analysis provenance.

See Also

[ild_fit()] with `backend = "brms"` is equivalent to `ild_brms()`. For IPW/MSM workflows, full Bayesian joint MSM is not in tidyILD; see [ild_msm_inference].

`ild_brms_dynamics_formula` *Template brms formula for person-varying lag slope*

Description

Returns a suggested formula and notes for fitting with [ild_brms()]. This does **not** run the model. Identification, priors, and convergence depend on N, T, and scaling; see `vignette("brms-dynamics-recipes", package = "tidyILD")`.

Usage

```
ild_brms_dynamics_formula(outcome, lag_var, id_var = "id")
```

Arguments

`outcome`	Character name of the response.
`lag_var`	Character name of an existing lag column (e.g. from [ild_lag()]).
`id_var`	Character name of the grouping column in `data` (default `"id"` as stored after [ild_prepare()] — use the original id column name).

Value

A list: `formula` (`formula` object), `notes` (character).

See Also

[ild_brms()], [ild_panel_lag_prepare()].

`ild_build_msm_history` *Build MSM lagged history columns from a spec*

Description

Applies [ild_lag()] repeatedly according to an [ild_msm_history_spec()] and records a manifest under `attr(x, "ild_msm_history_manifest")`.

Usage

```
ild_build_msm_history(data, spec, mode = NULL, max_gap = NULL)
```

Arguments

data	An ILD object.
spec	Object from [ild_msm_history_spec()].
mode	Optional override for spec$mode.
max_gap	Optional override for spec$max_gap.

Value

ILD object with lagged columns added; attributes include ild_msm_history_manifest (tibble with variable, lag, column).

Examples

```
d <- ild_simulate(n_id = 6, n_obs_per = 5, seed = 1)
d$stress <- rnorm(nrow(d))
d$trt <- rbinom(nrow(d), 1, 0.4)
d <- ild_prepare(d, id = "id", time = "time")
hs <- ild_msm_history_spec(c("stress", "trt"), lags = 1:2)
d2 <- ild_build_msm_history(d, hs)
attr(d2, "ild_msm_history_manifest")
```

---

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_fits *Compare fitted models (AIC, BIC, nobs)*

Description

Builds a compact comparison table for a named list of fitted objects. Uses [stats::AIC()], [stats::BIC()], and [stats::nobs()] where those methods exist. This is **not** a likelihood-ratio test and does not establish that models are nested; interpret only when model comparison is scientifically appropriate for your estimand.

Usage

```
ild_compare_fits(fits, guardrail_bundles = NULL)
```

Arguments

`fits`	A named (or unnamed) list of fitted model objects.
`guardrail_bundles`	Optional named list of [ild_diagnostics_bundle] objects aligned with `fits` (same length and order). When provided, adds `n_guardrails` per row.

Value

A tibble with columns `model`, `engine` (from [ild_methods()] truncated to one line when long), `aic`, `bic`, `n_obs`, `converged` (best-effort), `n_guardrails` (if bundles given), and `notes` (fit-specific caveats).

See Also

`vignette("temporal-dynamics-model-choice", package = "tidyILD")`.

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_ctsem *Fit a continuous-time state-space model via ctsem*

Description

First-class tidyILD wrapper for **ctsem**. This backend is designed for irregularly spaced ILD where continuous-time dynamics are preferable to discrete-time approximations.

Usage

```
ild_ctsem(
  data,
  outcome,
  ct_model = NULL,
  model_type = c("stanct", "ctfit"),
  id_col = ".ild_id",
  time_col = ".ild_time_num",
  time_scale = 1,
  ...
)
```

Arguments

data	An object that passes [validate_ild()] (typically after [ild_prepare()]).
outcome	Character name of numeric outcome column.
ct_model	Optional ctsem model object. If NULL, a conservative default model is created: for model_type = "stanct", ctsem::ctModel(type = "ct", ...) with one manifest loading on one latent; for "ctfit", type = "omx" with Tpoints = max(rows per id, 2) from the prepared long data (custom ct_model may still be needed for irregular designs).
model_type	Character. "stanct" (default) or "ctfit".
id_col	Person-id column in data (default ".ild_id").
time_col	Continuous-time index column in data (default ".ild_time_num").
time_scale	Optional positive scalar to rescale time_col before fitting (default 1).
...	Passed to ctsem::ctStanFit() or ctsem::ctFit().

Details

v1 focuses on a conservative single-outcome pathway with explicit class, provenance, and diagnostics compatibility. Users may supply a custom ctsem model object; otherwise a simple 1-manifest local-level style template is used.

Value

An object of class c("ild_fit_ctsem", "ild_fit_model", "ild_analysis"). Attributes ild_data and ild_provenance are attached.

See Also

[ild_tidy()], [ild_augment()], [ild_diagnose()], [ild_autoplot()].

---

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.

Details

Also a **section provider** for [ild_diagnose()] (see [ild_diagnostics_utilities]).

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.

Bundle integration

[ild_diagnose()] embeds this object in `design$ild_design_check` on the `ild_diagnostics_bundle`.

See Also

[ild_diagnose()], [ild_diagnostics_bundle()]

Other ild_diagnostics_utilities: `ild_ipcw_weights()`, `ild_iptw_msm_weights()`, `ild_iptw_weights()`, `ild_ipw_weights()`, `ild_joint_msm_weights()`, `ild_missing_cohort()`, `ild_missing_compliance()`, `ild_missing_hazard_first()`, `ild_missing_model()`, `ild_missing_pattern()`, `ild_missingness_report()`

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_diagnose *Engine-agnostic diagnostics façade*

Description

Returns a single `ild_diagnostics_bundle` for `lmerMod`, `lme`, `brmsfit`, `ild_fit_kfas`, and `ild_fit_ctsem`. Sections are parallel across engines; see `?ild_diagnostics_bundle`. Residual ACF/Q-Q for frequentist models are stored in `residual$legacy_ild_diagnostics` for [plot_ild_diagnostics()]. For raw residual-only computation without the bundle, call [ild_diagnostics()] directly.

Usage

```
ild_diagnose(object, ...)

## S3 method for class 'ild_fit_kfas'
ild_diagnose(object, data = NULL, type = NULL, by_id = NULL, ...)

## S3 method for class 'lmerMod'
ild_diagnose(
  object,
  data = NULL,
  type = c("residual_acf", "residual_time", "qq"),
  by_id = TRUE,
  missing_model = FALSE,
  missing_model_predictors = NULL,
  causal_detail = FALSE,
  balance = FALSE,
  balance_treatment = NULL,
  balance_covariates = NULL,
  balance_weights_col = ".ipw_treat",
  balance_by_occasion = FALSE,
```

```
  ...
)

## S3 method for class 'lme'
ild_diagnose(
  object,
  data = NULL,
  type = c("residual_acf", "residual_time", "qq"),
  by_id = TRUE,
  missing_model = FALSE,
  missing_model_predictors = NULL,
  causal_detail = FALSE,
  balance = FALSE,
  balance_treatment = NULL,
  balance_covariates = NULL,
  balance_weights_col = ".ipw_treat",
  balance_by_occasion = FALSE,
  ...
)

## S3 method for class 'brmsfit'
ild_diagnose(
  object,
  data = NULL,
  type = c("all", "convergence", "sampler", "ppc"),
  by_id = TRUE,
  ppc_ndraws = 500L,
  missing_model = FALSE,
  missing_model_predictors = NULL,
  causal_detail = FALSE,
  balance = FALSE,
  balance_treatment = NULL,
  balance_covariates = NULL,
  balance_weights_col = ".ipw_treat",
  balance_by_occasion = FALSE,
  ...
)

## S3 method for class 'ild_fit_ctsem'
ild_diagnose(
  object,
  data = NULL,
  type = NULL,
  by_id = NULL,
  missing_model = FALSE,
  missing_model_predictors = NULL,
  causal_detail = FALSE,
  balance = FALSE,
```

```
  balance_treatment = NULL,
  balance_covariates = NULL,
  balance_weights_col = ".ipw_treat",
  balance_by_occasion = FALSE,
  ...
)

## Default S3 method:
ild_diagnose(object, ...)
```

Arguments

object	A fitted model.
...	Reserved.
data	Optional ILD data (defaults to attr(object, "ild_data")).
type	Residual diagnostic types for lmerMod/lme (passed to [ild_diagnostics()]). For brmsfit: "all" or any of "convergence", "sampler", "ppc".
by_id	For lmerMod/lme: ACF by person.
missing_model	If TRUE, fit ild_missing_model for the response (optional; can be slow). Requires at least one predictor.
missing_model_predictors	Character vector for ild_missing_model; default NULL uses formula predictors.
causal_detail	If TRUE, add extra quantile summaries for .ipw when present.
balance	If TRUE, add weighted covariate balance (SMD) and ESS to causal$balance when balance_covariates and balance_treatment are set.
balance_treatment	Character. Binary treatment column for balance (required if balance = TRUE).
balance_covariates	Character vector of covariate columns for SMDs (required if balance = TRUE).
balance_weights_col	Weights column for balance (default ".ipw_treat"; use ".ipw" for joint MSM).
balance_by_occasion	If TRUE, balance and ESS are stratified by .ild_seq.
ppc_ndraws	For brmsfit: posterior predictive draws.

Details

Building blocks: the same analyses are available as standalone functions ([ild_design_check()], [ild_missing_pattern()], etc.); see [ild_diagnostics_utilities].

Value

An ild_diagnostics_bundle.

See Also

ild_autoplot, ild_diagnostics_bundle, guardrail_registry, ild_diagnostics_utilities, ild_diagnostics

---

`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_diagnostics_bundle

Canonical ILD diagnostics bundle

Description

The bundle has identical top-level **slot names** for every estimation backend. Content differs by engine; unavailable sections are `NULL`. `warnings` and `guardrails` are always tibbles (possibly 0 rows). Guardrails use the canonical schema documented in `guardrail_registry`: `rule_id`, `section`, `severity`, `triggered`, `message`, `recommendation` (only triggered rules are stored as rows).

Usage

```
ild_diagnostics_bundle(
  meta = NULL,
  data = NULL,
  design = NULL,
  fit = NULL,
  residual = NULL,
  predictive = NULL,
  missingness = NULL,
  causal = NULL,
  warnings = NULL,
  guardrails = NULL,
  summary_text = character()
)

new_ild_diagnostics_bundle(
  meta = NULL,
  data = NULL,
  design = NULL,
  fit = NULL,
```

```
  residual = NULL,
  predictive = NULL,
  missingness = NULL,
  causal = NULL,
  warnings = NULL,
  guardrails = NULL,
  summary_text = character()
)

is_ild_diagnostics_bundle(x)
```

Arguments

meta, data, design, fit, residual, predictive, missingness, causal	Optional sections (list or structured object, or NULL).
warnings, guardrails	Tibbles (default empty).
summary_text	Character vector (default empty).
x	Object to test.

Details

Several standalone functions **feed** these slots when you use ild_diagnose; see ild_diagnostics_utilities (ild_design_check, ild_missing_pattern, ild_missing_model, ild_ipw_weights).

Bundles returned by ild_diagnose also set attr(bundle, "ild_fit") and attr(bundle, "ild_data") for ild_autoplot; re-run ild_diagnose if these are missing (e.g. after loading a saved bundle without the fitted object).

The print method lists each slot and prints a short **summary** line: number of warning rows, number of guardrail rows, highest guardrail severity (info < warning), and up to five triggered rule_id values.

Value

An object of class ild_diagnostics_bundle.

is_ild_diagnostics_bundle: logical.

Slots

meta List or NULL: run ids, engine, dimensions.

data List or NULL: spacing, gaps, compliance, missingness, distributions.

design List or NULL: within/between variation, time coverage, occasions, imbalance.

fit List or NULL: convergence, singularity, optimizer / MCMC diagnostics.

residual List or NULL: ACF, Q-Q, fitted vs observed, heteroskedasticity.

predictive List or NULL: CV, PPC, forecast error.

missingness List or NULL: IPW / imputation summaries when applicable.

causal List or NULL: weights, positivity, causal diagnostics when applicable.

`warnings` Tibble: structured warnings.

`guardrails` Tibble: methodological guardrails (see `guardrail_registry`).

`summary_text` Character: short narrative summary.

See Also

`ild_diagnose`, `guardrail_registry`, `ild_diagnostics_utilities`, `ild_tidy_schema`, `ild_augment_schema`. Per-section field semantics: `vignette("developer-contracts", package = "tidyILD")`.

`ild_diagnostics_utilities`

Standalone diagnostics utilities and bundle section providers

---

Description

These functions are **standalone utilities** you can run on prepared ILD data at any time (typically before or alongside modeling). The same logic also **feeds** `ild_diagnostics_bundle` sections when you call `ild_diagnose` on a fitted model, so the package presents one coherent diagnostics story: a top-level façade (`ild_diagnose`) plus reusable building blocks below.

Relationship to `ild_diagnose`

`ild_design_check` Populates the bundle **design** slot (embedded as `design$ild_design_check`) and contributes WP/BP and design-time missingness views.

`ild_missing_pattern` Drives **data** (`data$missing_pattern`, global columns) and **missingness** (variable-specific summaries when model predictors are known).

`ild_missing_model` Not called automatically by `ild_diagnose`. Use when you fit a missingness model for IPW or interpretation; it is the usual prerequisite for `ild_ipw_weights`.

`ild_ipw_weights` Adds `.ipw` (and related steps in the IPW workflow). When those columns are present on the ILD data attached to the fit, `ild_diagnose` fills the bundle **causal** slot with weight summaries.

`ild_iptw_weights`, `ild_iptw_msm_weights`, `ild_ipcw_weights`, `ild_joint_msm_weights` MSM-style inverse-probability weights for **treatment** (pooled or sequential A_t) and **monotone censoring**, combined into `.ipw`. Distinct from outcome missingness IPW above.

`ild_msm_bootstrap`, `tidy_ild_msm_bootstrap` Person-level bootstrap for weighted `lmer` after `ild_ipw_refit`; see `ild_msm_inference`.

`ild_msm_balance`, `ild_ipw_ess`, `ild_msm_overlap_plot` Weighted covariate balance (SMD), effective sample size, and propensity overlap plots for MSM / IPW workflows.

`ild_msm_estimand`, `ild_msm_fit`, `ild_msm_diagnose`, `ild_msm_contrast_over_time`, `ild_msm_history_spec`, `ild_b` Estimand-first MSM workflow, one-call diagnostics bridge, time-indexed contrasts, history construction, and simulation-based recovery checks.

See Also

`ild_diagnose`, `ild_diagnostics_bundle`, `ild_diagnostics`

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_fit *Fit a mixed model to ILD data (unified entry point)*

Description

Façade that selects the estimation backend explicitly. Equivalent to calling [ild_lme()] for "lme4" / "nlme", or [ild_brms()] for "brms". Named functions remain the primary APIs for full documentation and examples.

Usage

```
ild_fit(
  formula,
  data,
  backend = c("lme4", "nlme", "brms"),
  correlation_class = c("auto", "AR1", "CAR1"),
  random = ~1 | .ild_id,
  warn_no_ar1 = TRUE,
  warn_uncentered = TRUE,
  prior = NULL,
  prior_template = c("default", "weakly_informative", "minimal_shrinkage"),
  ...
)
```

Arguments

formula	Passed to [ild_lme()] or [ild_brms()].
data	An ILD object (see [is_ild()]).
backend	Character. "lme4" (default, [lme4::lmer()]), "nlme" ([nlme::lme()] with AR1/CAR1 residual structure), or "brms" ([brms::brm()]).
correlation_class, random, warn_no_ar1	Passed to [ild_lme()] when backend is "lme4" or "nlme". Ignored when backend = "brms", except correlation_class must be "auto" for "brms" (non-"auto" values are for residual correlation under nlme only and will error).
warn_uncentered	Passed to [ild_lme()] or [ild_brms()].
prior, prior_template	Passed to [ild_brms()] when backend = "brms". For "lme4" / "nlme", prior must be NULL and prior_template must remain the default ("default"); otherwise an error is raised so mistaken cross-backend arguments are not silently ignored.
...	Passed to [ild_lme()] / [lme4::lmer()] / [nlme::lme()] or [ild_brms()].

Details

State-space / latent-dynamics models are not mixed-model formulas: use [ild_kfas()] or [ild_ctsem()] directly; see `vignette("kfas-choosing-backend", package = "tidyILD")`.

To pass backend to `brms::brm()` (e.g. "rstan" vs "cmdstanr"), use [ild_brms()] directly: `ild_fit()` reserves backend for the tidyILD engine ("lme4", "nlme", "brms").

Value

A fitted model: lmerMod / lme from [ild_lme()], or brmsfit from [ild_brms()], each with the same attributes as those functions document.

See Also

[ild_kfas()] and [ild_ctsem()] for state-space / latent-dynamics backends (not available via `ild_fit()`).

---

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, facet_by = 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.
facet_by	Optional column in ILD for [ggplot2::facet_wrap()].
...	Passed to [ild_plot()] (e.g. id_var, time_var).

Value

A ggplot object.

---

ild_heterogeneity *Heterogeneity and person-specific effects from mixed models*

Description

Summarizes **conditional modes** (empirical Bayes / BLUP deviations) and **person-specific coefficients** (`fixef + ranef`) for hierarchical models fit with [ild_lme()] or [ild_brms()]. Outputs are labeled as **partial-pooling** estimates; they are not the same as no-pooling separate regressions (see [ild_person_model()]) or population-average fixed effects alone.

Usage

```
ild_heterogeneity(fit, ...)

## S3 method for class 'lmerMod'
ild_heterogeneity(
  fit,
  term = NULL,
  group = NULL,
  threshold = NULL,
  scale = c("raw", "sd_x", "sd_y"),
  conf_level = 0.95,
  ...
)

## S3 method for class 'lme'
ild_heterogeneity(
  fit,
  term = NULL,
  group = NULL,
  threshold = NULL,
  scale = c("raw", "sd_x", "sd_y"),
  conf_level = 0.95,
  ...
)
```

```
ild_heterogeneity_stratified(formula, data, subgroup, min_n_id = 5L, ...)

## S3 method for class 'ild_heterogeneity'
ild_tidy(x, ...)

## S3 method for class 'ild_heterogeneity'
ild_autoplot(x, type = c("caterpillar", "histogram"), term = NULL, ...)

## S3 method for class 'brmsfit'
ild_heterogeneity(
  fit,
  term = NULL,
  group = NULL,
  threshold = NULL,
  scale = c("raw", "sd_x", "sd_y"),
  n_draws = 500L,
  ...
)
```

Arguments

fit	A fitted lmerMod, lme, or brmsfit (from [ild_brms()]).
...	Passed to [ild_lme()].
term	Which random-effect term to plot (required if multiple).
group	Name of the grouping factor in ranef(fit) to use (default first).
threshold	Optional numeric; with scale, proportion of person-specific **total** coefficients exceeding this threshold is reported in summary.
scale	If threshold is set: "raw" (default), "sd_x" (multiply threshold by residual SD of a focal predictor from model frame), or "sd_y" (multiply by SD of response). Only used when a single numeric predictor is found for sd_x / sd_y heuristics.
conf_level	For lmerMod, Wald interval level for conditional modes (normal approx.).
formula, data	Passed to [ild_lme()] within each subgroup.
subgroup	Character name of a column in data defining subgroups.
min_n_id	Minimum number of distinct persons required in a subgroup.
x	An ild_heterogeneity object.
type	"caterpillar" (default) or "histogram".
n_draws	For brmsfit, number of posterior draws for summaries (capped).

Value

An object of class ild_heterogeneity: a list with meta, random_effects (long tibble), varcorr (tibble), summary (per-term metrics), and fit.

A tibble with one row per successful subgroup fit, including key columns from ild_heterogeneity()$summary flattened (subgroup_level, term, etc.). This is a **descriptive** comparison across refits, not a formal test of variance heterogeneity.

Methods (by generic)

- `ild_tidy(ild_heterogeneity)`: Tidy long table of person-level random effects (for `ild_heterogeneity` objects).
- `ild_autoplot(ild_heterogeneity)`: Caterpillar or histogram of person-specific effects.

See Also

[ild_tidy.ild_heterogeneity()], [ild_autoplot.ild_heterogeneity()], [ild_heterogeneity_stratified()], [ild_person_model()]

---

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_ipcw_weights *Inverse probability of censoring weights (IPCW) for monotone dropout*

Description

Builds **discrete-time** IPCW weights for **monotone loss-to-follow-up**: each person contributes a sequence of observed visits; after the last visit the person is censored. This differs from [ild_missing_model()] + [ild_ipw_weights()], which model **sporadic item missingness** on an outcome.

Usage

```
ild_ipcw_weights(x, predictors, stabilize = TRUE, trim = c(0.01, 0.99), ...)
```

Arguments

x	An ILD object (see [is_ild()]).
predictors	Character vector of covariate names (time-varying or baseline carried forward in x).
stabilize	Logical. If TRUE (default), multiply factors mean(1 - p_drop) / (1 - p_drop) at each row before cumulating; if FALSE, use 1 / (1 - p_drop).
trim	Numeric of length 2. Quantiles applied to **final** cumulative .ipw_censor (default c(0.01, 0.99)).
...	Passed to stats::glm().

Details

Internally, each person-occasion row is labeled with an indicator drop_next: whether the person drops before the next scheduled occasion (including the last observed row). A pooled logistic regression models drop_next given covariates; stabilized or unstabilized inverse probabilities are accumulated within person (product over follow-up) to yield .ipw_censor.

Assumptions: Monotone censoring, positivity, correct censoring model.

Value

x with added column .ipw_censor. Attribute ild_ipcw_fit holds the fitted glm.

Bundle integration

Adds .ipw_censor. Combine with [ild_iptw_weights()] or [ild_iptw_msm_weights()] via [ild_joint_msm_weights()].

See Also

[ild_iptw_weights()], [ild_joint_msm_weights()]

Other ild_diagnostics_utilities: ild_design_check(), ild_iptw_msm_weights(), ild_iptw_weights(), ild_ipw_weights(), ild_joint_msm_weights(), ild_missing_cohort(), ild_missing_compliance(), ild_missing_hazard_first(), ild_missing_model(), ild_missing_pattern(), ild_missingness_report()

Examples

```
set.seed(3)
d <- ild_simulate(n_id = 18, n_obs_per = 7, seed = 3)
d$stress <- rnorm(nrow(d))
x <- ild_prepare(d, id = "id", time = "time")
x <- ild_ipcw_weights(x, predictors = "stress")
summary(x$.ipw_censor)
```

`ild_iptw_msm_weights` *Sequential MSM inverse probability of treatment weights (IPTW for A_t)*

Description

For **time-varying binary treatment** A_t, fits a separate treatment propensity model at each occasion t (pooling persons at that time): $P(A_t \mid \bar{L}_t)$ where $\bar{L}_t$ is supplied by the user (e.g. lags from [ild_lag()], prior treatment, time-varying confounders). The stabilized factor at each person-occasion is multiplied **within person** over time to yield a **cumulative** MSM IPTW in `.ipw_treat`.

Usage

```
ild_iptw_msm_weights(
  x,
  treatment,
  history,
  time_var = ".ild_seq",
  stabilize = c("marginal", "prior_treatment", "none"),
  prior_treatment = NULL,
  trim = c(0.01, 0.99),
  ...
)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`treatment`	Character. Binary A_t column (`0`/`1` or two-level factor).
`history`	Predictors $\bar{L}_t$ for the **denominator** model: a `character` vector of column names, or a one-sided `formula` like `~ stress + trt_lag1`. Empty history fits an intercept-only model.
`time_var`	Character. Occasion index column (default `".ild_seq"`).
`stabilize`	Character: `"marginal"`, `"prior_treatment"`, or `"none"`.
`prior_treatment`	Character. Required when `stabilize = "prior_treatment"`: name of the A_{t-1} (or prior treatment) column for the numerator model.
`trim`	Numeric of length 2. Quantiles applied to the **final** cumulative `.ipw_treat` (default `c(0.01, 0.99)`). Use length-0 numeric to skip.
`...`	Passed to `stats::glm()` for each occasion (denominator model; numerator model when `prior_treatment` is used also uses `...`).

Details

This differs from [ild_iptw_weights()], which fits **one pooled** logistic over all rows (appropriate for a single treatment decision or descriptive pooling, not the standard sequential MSM estimand for $\bar{A}_K$).

Assumptions: Sequential exchangeability given $\bar{L}_t$, positivity at each t, and correct models. Variance (e.g. bootstrap by person) is not computed here.

Value

`x` with `.ipw_treat` set to the cumulative sequential weight. Attributes: `ild_iptw_msm_fits` (named list of denominator `glm` fits by occasion), `ild_iptw_msm_numerator_fits` (same for numerator when `prior_treatment`).

Stabilization

`marginal` Numerator at each occasion uses the sample marginal $P(A_t = 1)$ among rows at that time (same pattern as pooled stabilized IPTW).

`prior_treatment` Numerator uses $P(A_t \mid A_{t-1})$ from a logistic model `A_t ~ prior_treatment` fit at each occasion (pass `prior_treatment` column name). Denominator remains $P(A_t \mid \bar{L}_t)$.

`none` Unstabilized $1/P(A_t \mid \bar{L}_t)$ for the observed A_t.

Bundle integration

Sets `.ipw_treat` for use with [ild_ipcw_weights()] and [ild_joint_msm_weights()].

See Also

[ild_iptw_weights()], [ild_ipcw_weights()], [ild_joint_msm_weights()]

Other ild_diagnostics_utilities: `ild_design_check()`, `ild_ipcw_weights()`, `ild_iptw_weights()`, `ild_ipw_weights()`, `ild_joint_msm_weights()`, `ild_missing_cohort()`, `ild_missing_compliance()`, `ild_missing_hazard_first()`, `ild_missing_model()`, `ild_missing_pattern()`, `ild_missingness_report()`

Examples

```
set.seed(11)
d <- ild_simulate(n_id = 20, n_obs_per = 6, seed = 11)
d$stress <- rnorm(nrow(d))
d$trt <- as.integer(stats::rbinom(nrow(d), 1L, 0.4))
d <- ild_prepare(d, id = "id", time = "time")
d <- ild_lag(d, stress, mode = "gap_aware", max_gap = Inf)
d <- ild_lag(d, trt, mode = "gap_aware", max_gap = Inf)
x <- ild_iptw_msm_weights(d, treatment = "trt", history = ~ stress_lag1 + trt_lag1)
summary(x$.ipw_treat)
```

`ild_iptw_weights` *Inverse probability of treatment weights (IPTW)*

Description

Fits a **single pooled** treatment propensity model (logistic regression) over all person-occasions and computes stabilized or unstabilized IPTW weights (one factor per row). Use this for a **time-invariant** treatment or descriptive pooling. For **time-varying** A_t with a sequential MSM estimand, use [ild_iptw_msm_weights()] instead. Outcome missingness IPW is [ild_missing_model()] + [ild_ipw_weights()].

Usage

```
ild_iptw_weights(
  x,
  treatment,
  predictors,
  stabilize = TRUE,
  trim = c(0.01, 0.99),
  ...
)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`treatment`	Character. Name of a **binary** treatment column (numeric `0`/`1` or two-level factor).
`predictors`	Character vector of covariate names in `x` (main terms only).
`stabilize`	Logical. If `TRUE` (default), stabilized weights $P(A)/P(A|X)$ for the observed A; otherwise $1/P(A|X)$.
`trim`	Numeric of length 2. Quantiles for trimming (default `c(0.01, 0.99)`).
`...`	Passed to `stats::glm()`.

Details

Assumptions: Positivity ($0 < P(A|X) < 1$), correct specification of the treatment model. This is sensitivity / MSM-style tooling, not a substitute for careful causal design.

Value

`x` with added column `.ipw_treat`. Attributes include `ild_iptw_fit` (the fitted `glm` object).

Bundle integration

Adds `.ipw_treat`. Use [ild_joint_msm_weights()] to combine with `.ipw_censor` into `.ipw` for [ild_ipw_refit()] and [ild_diagnose()].

See Also

[ild_iptw_msm_weights()], [ild_ipcw_weights()], [ild_joint_msm_weights()], [ild_ipw_refit()]

Other ild_diagnostics_utilities: `ild_design_check()`, `ild_ipcw_weights()`, `ild_iptw_msm_weights()`, `ild_ipw_weights()`, `ild_joint_msm_weights()`, `ild_missing_cohort()`, `ild_missing_compliance()`, `ild_missing_hazard_first()`, `ild_missing_model()`, `ild_missing_pattern()`, `ild_missingness_report()`

Examples

```
set.seed(2)
d <- ild_simulate(n_id = 20, n_obs_per = 6, seed = 2)
d$stress <- rnorm(nrow(d))
d$trt <- as.integer(d$stress > 0)
x <- ild_prepare(d, id = "id", time = "time")
x <- ild_iptw_weights(x, treatment = "trt", predictors = "stress")
summary(x$.ipw_treat)
```

`ild_ipw_ess` *Effective sample size from weights*

Description

For nonnegative weights w, $\mathrm{ESS} = (\sum w)^2 / \sum w^2$.

Usage

```
ild_ipw_ess(
  data,
  weights_col = ".ipw",
  by_occasion = FALSE,
  time_var = ".ild_seq"
)
```

Arguments

`data`	An ILD object.
`weights_col`	Name of the nonnegative weight column (e.g. `".ipw"` or `".ipw_treat"`).
`by_occasion`	If `TRUE`, one ESS per level of `time_var`.
`time_var`	Column for occasion index (default `".ild_seq"`).

Value

A single numeric (pooled) or a `tibble` with `occasion` and `ess` when `by_occasion = TRUE`.

Examples

```
set.seed(1)
d <- ild_simulate(n_id = 10, n_obs_per = 5, seed = 1)
x <- ild_prepare(d, id = "id", time = "time")
x$.ipw <- runif(nrow(x), 0.5, 1.5)
ild_ipw_ess(x, ".ipw")
ild_ipw_ess(x, ".ipw", by_occasion = TRUE)
```

---

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 weight column (default `.ipw`) from [ild_ipw_weights()] or joint MSM weights from [ild_joint_msm_weights()], 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. For inference under estimated weights, see [ild_msm_bootstrap()] and [ild_msm_inference].

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.

Details

Also a **section provider** for [ild_diagnose()] when `.ipw` is present on the analysis data (see [ild_diagnostics_utilities]).

Value

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

Bundle integration

Adds `.ipw` to `x`. When that column is on the ILD data attached to the fit, [ild_diagnose()] fills `causal` (weight summaries) on `ild_diagnostics_bundle`.

See Also

[ild_diagnose()], [ild_diagnostics_bundle()], [ild_missing_model()], [ild_iptw_weights()], [ild_ipcw_weights()], [ild_joint_msm_weights()]

Other ild_diagnostics_utilities: `ild_design_check()`, `ild_ipcw_weights()`, `ild_iptw_msm_weights()`, `ild_iptw_weights()`, `ild_joint_msm_weights()`, `ild_missing_cohort()`, `ild_missing_compliance()`, `ild_missing_hazard_first()`, `ild_missing_model()`, `ild_missing_pattern()`, `ild_missingness_report()`

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_joint_msm_weights` *Joint MSM weights from IPTW and IPCW components*

Description

Multiplies `.ipw_treat` and `.ipw_censor` into a single analysis weight column (default `.ipw`) for use with [ild_ipw_refit()] and [ild_diagnose()]. Optional scaling sets the mean joint weight to 1.

Usage

```
ild_joint_msm_weights(
  x,
  stabilize = c("mean1", "none"),
  trim = c(0.01, 0.99),
  joint_name = ".ipw"
)
```

Arguments

`x`	An ILD object containing `.ipw_treat` and `.ipw_censor` (from [ild_iptw_weights()] or [ild_iptw_msm_weights()], and [ild_ipcw_weights()]).
`stabilize`	Character. `"none"`: `w_treat * w_censor` only. `"mean1"` (default): divide joint weights by their mean so the mean is 1 (helps comparability with stabilized components).
`trim`	Numeric of length 2. Quantiles for trimming the **joint** vector (default `c(0.01, 0.99)`).
`joint_name`	Character. Name of the combined weight column (default `".ipw"`).

Value

x with joint_name set; component columns unchanged.

Bundle integration

Sets .ipw (or joint_name) so existing causal diagnostics and guardrails apply to the **joint** weight.

See Also

[ild_iptw_weights()], [ild_iptw_msm_weights()], [ild_ipcw_weights()], [ild_ipw_refit()]

Other ild_diagnostics_utilities: ild_design_check(), ild_ipcw_weights(), ild_iptw_msm_weights(), ild_iptw_weights(), ild_ipw_weights(), ild_missing_cohort(), ild_missing_compliance(), ild_missing_hazard_first(), ild_missing_model(), ild_missing_pattern(), ild_missingness_report()

Examples

```
set.seed(4)
d <- ild_simulate(n_id = 15, n_obs_per = 8, seed = 4)
d$stress <- rnorm(nrow(d))
d$trt <- as.integer(d$stress > 0)
x <- ild_prepare(d, id = "id", time = "time")
x <- ild_iptw_weights(x, "trt", "stress")
x <- ild_ipcw_weights(x, "stress")
x <- ild_joint_msm_weights(x)
summary(x$.ipw)
```

---

ild_kfas *Fit a Gaussian state-space model via KFAS (single-subject ILD)*

Description

Opinionated entry point for **local level** models in v1. Requires the **KFAS** package. One distinct .ild_id per fit; subset your data first.

Usage

```
ild_kfas(
  data,
  outcome,
  state_spec = "local_level",
  observation_family = "gaussian",
  time_units,
  irregular_time = FALSE,
  smoother = TRUE,
  forecast_horizon = 0L,
  fit_context = c("single_series", "independent_series_per_id"),
  ...
)
```

Arguments

data	An object that passes [validate_ild()] (e.g. after [ild_prepare()]).
outcome	Character name of the numeric outcome column.
state_spec	Character; use "local_level" in v1. See inst/dev/KFAS_V1_BACKEND.md for the full controlled vocabulary.
observation_family	Currently only "gaussian".
time_units	Character describing the time column (e.g. "days", "hours"). Stored in provenance; required for clear reporting.
irregular_time	Logical; if TRUE, skips the irregular-spacing warning.
smoother	Logical; passed to KFS() smoothing (default TRUE).
forecast_horizon	Integer; when > 0, used by ild_plot_forecast() with predict on the fitted model and stored in mapping$forecast_horizon.
fit_context	"single_series" (default) or "independent_series_per_id" when you have fit the **same** state-space template separately for multiple persons and must not interpret results as a pooled multilevel latent model. The latter triggers guardrail GR_KFAS_UNMODELED_BETWEEN_PERSON_HETEROGENEITY.
...	Passed to KFAS::fitSSM() (e.g. method, inits).

Value

An object of class c("ild_fit_kfas", "ild_fit_model", "ild_analysis") with list slots kfas_model, kfs, spec, state_labels, mapping, schema_version, preprocessing. Attributes ild_data and ild_provenance are set.

See Also

inst/dev/KFAS_V1_BACKEND.md, [ild_tidy()], [ild_augment()], [ild_diagnose()].

Conceptual vignettes:

- vignette("kfas-state-space-modeling", package = "tidyILD")
- vignette("kfas-irregular-timing-spacing", package = "tidyILD")
- vignette("kfas-choosing-backend", package = "tidyILD")

---

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, bundle = 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.
bundle	Optional. An object from [ild_diagnose()] (class `ild_diagnostics_bundle`). When provided and `nrow(bundle$guardrails) > 0`, a short methodological-cautions sentence is appended after the provenance paragraph.
...	Unused.

Details

After fitting, call `diag <- ild_diagnose(fit)` and pass `bundle = diag` to surface triggered guardrails in the methods narrative without duplicating [ild_report()]'s structured `diagnostics_summary`.

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_cohort *Cohort-level fraction observed by occasion index*

Description

For each distinct '.ild_seq' value, computes how many rows exist and what fraction has a non-missing outcome. Optional line plot.

Usage

```
ild_missing_cohort(x, outcome, plot = TRUE)
```

Arguments

x	An ILD object.
outcome	Character; outcome column.
plot	Logical; if 'TRUE', return a 'ggplot2' object in '$plot'.

Details

Uses **within-study sequence** ('.ild_seq'), not calendar time. For irregular timing, interpret as ordinal wave position rather than equal time spacing.

Value

A list with ‘by_occasion‘ (tibble: ‘.ild_seq‘, ‘n_rows‘, ‘n_obs‘, ‘pct_observed‘) and if ‘plot = TRUE‘, ‘plot‘.

See Also

[ild_missing_pattern()], [ild_missing_compliance()]

Other ild_diagnostics_utilities: `ild_design_check()`, `ild_ipcw_weights()`, `ild_iptw_msm_weights()`, `ild_iptw_weights()`, `ild_ipw_weights()`, `ild_joint_msm_weights()`, `ild_missing_compliance()`, `ild_missing_hazard_first()`, `ild_missing_model()`, `ild_missing_pattern()`, `ild_missingness_report()`

`ild_missing_compliance` *Person-level missingness compliance for one outcome*

Description

Per person (after sorting by ‘.ild_seq‘), returns coverage on the outcome, longest streak of observed values, whether missingness is monotone (dropout pattern: once missing, stays missing), and optional comparison to an expected number of occasions.

Usage

```
ild_missing_compliance(x, outcome, expected_occasions = NULL)
```

Arguments

`x`	An ILD object (see [is_ild()]).
`outcome`	Character; column with possible ‘NA‘ (e.g. EMA outcome).
`expected_occasions`	Optional positive integer: planned waves per person for adherence (‘pct_of_expected‘). If persons have different row counts, this is only a rough benchmark; see Details.

Details

Monotone missing is defined only when the person has at least one missing outcome: all observations after the first missing are missing. Intermittent missing yields ‘FALSE‘. **Discrete hazard** for first missing is provided by [ild_missing_hazard_first()] (cohort-level by ‘.ild_seq‘).

Value

A tibble with columns ‘.ild_id‘, ‘n_rows‘, ‘n_obs_outcome‘, ‘pct_nonmissing_outcome‘, ‘longest_run_observed‘, ‘monotone_missing‘ (‘NA‘ if no missing values on the outcome for that person), and if ‘expected_occasions‘ is set, ‘pct_of_expected‘ and ‘meets_expected_rows‘.

See Also

[ild_missing_pattern()], [ild_missingness_report()]

Other ild_diagnostics_utilities: `ild_design_check()`, `ild_ipcw_weights()`, `ild_iptw_msm_weights()`, `ild_iptw_weights()`, `ild_ipw_weights()`, `ild_joint_msm_weights()`, `ild_missing_cohort()`, `ild_missing_hazard_first()`, `ild_missing_model()`, `ild_missing_pattern()`, `ild_missingness_report()`

`ild_missing_hazard_first` *Discrete hazard of first missing outcome on an ordinal schedule*

Description

For each '.ild_seq', estimates the fraction of person-occasions **at risk** (previous occasion observed, or first occasion) that are **missing** on the outcome. This is a coarse discrete-time hazard for **first** missing spell when missingness is intermittent; under **monotone dropout** it matches the usual discrete hazard of dropout.

Usage

```
ild_missing_hazard_first(x, outcome)
```

Arguments

`x`	An ILD object.
`outcome`	Character; outcome column.

Details

Assumptions: Rows are ordered within person by '.ild_seq'. The first row per person is always at risk. Later rows are at risk only if the previous row (same person) was non-missing—so a person who is already missing is not counted again. This targets a **first event** narrative; it is not a full repeated-events model.

Value

A tibble with '.ild_seq', 'n_at_risk', 'n_missing', 'hazard'.

See Also

[ild_missing_cohort()], [ild_missing_compliance()]

Other ild_diagnostics_utilities: `ild_design_check()`, `ild_ipcw_weights()`, `ild_iptw_msm_weights()`, `ild_iptw_weights()`, `ild_ipw_weights()`, `ild_joint_msm_weights()`, `ild_missing_cohort()`, `ild_missing_compliance()`, `ild_missing_model()`, `ild_missing_pattern()`, `ild_missingness_report()`

`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()`.

Details

Not run automatically by [ild_diagnose()]; it is the usual prerequisite for [ild_ipw_weights()] when you want IPW summaries in the bundle (see [ild_diagnostics_utilities]).

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`).

Bundle integration

There is no separate "missingness model" slot on the bundle. This function supports the IPW workflow; after [ild_ipw_weights()], [ild_diagnose()] can summarize weights in `causal` on `ild_diagnostics_bundle`.

See Also

[ild_diagnose()], [ild_diagnostics_bundle()], [ild_ipw_weights()]

Other ild_diagnostics_utilities: `ild_design_check()`, `ild_ipcw_weights()`, `ild_iptw_msm_weights()`, `ild_iptw_weights()`, `ild_ipw_weights()`, `ild_joint_msm_weights()`, `ild_missing_cohort()`, `ild_missing_compliance()`, `ild_missing_hazard_first()`, `ild_missing_pattern()`, `ild_missingness_report()`

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,
  outcome = NULL,
  expected_occasions = 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.

`outcome` Optional character; if set, `by_id` is left-joined with [ild_missing_compliance()] for that column (interpretable adherence metrics).

`expected_occasions` Passed to [ild_missing_compliance()] when `outcome` is set.

Details

Also a **section provider** for [ild_diagnose()] (see [ild_diagnostics_utilities]).

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`. When `outcome` is set, `by_id` includes compliance columns (see [ild_missing_compliance()]).

Bundle integration

[ild_diagnose()] passes summaries into `data$missing_pattern` (global) and `missingness` (model variables) on the `ild_diagnostics_bundle`.

See Also

[ild_diagnose()], [ild_diagnostics_bundle()]

Other ild_diagnostics_utilities: `ild_design_check()`, `ild_ipcw_weights()`, `ild_iptw_msm_weights()`, `ild_iptw_weights()`, `ild_ipw_weights()`, `ild_joint_msm_weights()`, `ild_missing_cohort()`, `ild_missing_compliance()`, `ild_missing_hazard_first()`, `ild_missing_model()`, `ild_missingness_report()`

`ild_missingness_report` *Missingness workflow report (orchestration)*

Description

Bundles person-level compliance, pattern summaries, cohort and hazard tables, optional [ild_missing_model()], heuristic flags, and short text snippets for methods sections. This is **diagnostic and reporting** tooling, not a substitute for formal sensitivity analysis or MNAR models.

Usage

```
ild_missingness_report(
  x,
  outcome,
  predictors = NULL,
  fit_missing_model = TRUE,
  random = FALSE,
  expected_occasions = NULL,
  max_ids = NULL,
```

```
  seed = NULL,
  cohort_plot = TRUE
)
```

Arguments

x	An ILD object (see [is_ild()]).
outcome	Character; outcome column with possible 'NA'.
predictors	Optional character vector passed to [ild_missing_model()] when 'fit_missing_model' is 'TRUE'. If empty, no missingness model is fit.
fit_missing_model	Logical; fit [ild_missing_model()] when predictors are non-empty.
random	Logical; passed to [ild_missing_model()] ('glmer' vs 'glm').
expected_occasions	Optional integer for [ild_missing_compliance()].
max_ids, seed	Passed to [ild_missing_pattern()] for large cohorts.
cohort_plot	Logical; include ggplot for [ild_missing_cohort()].

Value

A list (class 'ild_missingness_report') with:

compliance Tibble from [ild_missing_compliance()].

pattern Output of [ild_missing_pattern()] (includes compliance columns on 'by_id' if 'outcome' was passed through).

cohort List from [ild_missing_cohort()].

hazard Tibble from [ild_missing_hazard_first()].

flags Named list: 'dropout_late_pooled' (logical from same heuristic as guardrail 'GR_DROPOUT_LATE_CONCENTRAT

missing_model Result of [ild_missing_model()] or 'NULL'.

snippets Character vector of short paragraphs for copy-paste.

See Also

[ild_missing_pattern()], [ild_missing_bias()], [ild_ipw_weights()], 'vignette("ild-missingness-workflow", package = "tidyILD")'

Other ild_diagnostics_utilities: `ild_design_check()`, `ild_ipcw_weights()`, `ild_iptw_msm_weights()`, `ild_iptw_weights()`, `ild_ipw_weights()`, `ild_joint_msm_weights()`, `ild_missing_cohort()`, `ild_missing_compliance()`, `ild_missing_hazard_first()`, `ild_missing_model()`, `ild_missing_pattern()`

`ild_msm_balance` *Covariate balance (weighted SMD) for MSM / IPW*

Description

Computes **standardized mean differences** between weighted treated and control groups for each named covariate. When `by_occasion = TRUE`, balance is computed **within** each occasion (stratum); when `FALSE`, all rows are pooled into one pseudo-population.

Usage

```
ild_msm_balance(
  data,
  treatment,
  covariates,
  weights_col = ".ipw_treat",
  by_occasion = FALSE,
  time_var = ".ild_seq"
)
```

Arguments

`data`	An ILD object with `treatment`, `covariates`, and `weights_col`.
`treatment`	Character. Binary treatment column (`0`/`1` or two-level factor).
`covariates`	Character vector of numeric or binary covariate columns.
`weights_col`	Analysis weights (e.g. `".ipw_treat"` or `".ipw"`).
`by_occasion`	If `TRUE`, stratify by `time_var`.
`time_var`	Occasion index column (default `".ild_seq"`).

Value

A `tibble` with columns `stratum`, `covariate`, `smd`, `mean_treated`, `mean_control`, `ess_stratum`, `weight_col`.

Examples

```
set.seed(2)
d <- ild_simulate(n_id = 12, n_obs_per = 6, seed = 2)
d$stress <- rnorm(nrow(d))
d$trt <- as.integer(stats::rbinom(nrow(d), 1L, 0.45))
x <- ild_prepare(d, id = "id", time = "time")
x$.ipw_treat <- runif(nrow(x), 0.8, 1.2)
ild_msm_balance(x, treatment = "trt", covariates = "stress", weights_col = ".ipw_treat")
```

`ild_msm_bootstrap` *Cluster bootstrap inference for weighted* `lmer` *(MSM / IPW sensitivity)*

Description

Resamples **clusters** (persons) with replacement, subsets `attr(fit, "ild_data")` (or `data`), optionally re-estimates weights, and refits `lmer` with the same formula. Collects fixed effects across replicates for bootstrap standard errors and percentile confidence intervals.

Usage

```
ild_msm_bootstrap(
  fit = NULL,
  formula = NULL,
  data = NULL,
  weights_col = ".ipw",
  n_boot = 200L,
  weight_policy = c("fixed_weights", "reestimate_weights"),
  weights_fn = NULL,
  cluster = c("id", "data"),
  cluster_vec = NULL,
  coef_fun = function(f) lme4::fixef(f),
  seed = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`fit`	A `lmerMod` from [ild_ipw_refit()] or [ild_lme()] with `attr(fit, "ild_data")`. Ignored if `formula` and `data` are supplied.
`formula`	Optional. If `fit` is `NULL`, mixed-model formula for refitting.
`data`	Optional ILD object when `fit` is `NULL`; must contain `weights_col`.
`weights_col`	Name of the row-level weight column in each bootstrap dataset (default `".ipw"`).
`n_boot`	Number of bootstrap replicates.
`weight_policy`	`"fixed_weights"` (subset rows; weights unchanged within replicated rows) or `"reestimate_weights"` (apply `weights_fn` to each resampled dataset before fitting).
`weights_fn`	Function `function(x)` taking ILD `x` and returning ILD with `weights_col` present. Required when `weight_policy = "reestimate_weights"`. Ignored for `"fixed_weights"`.
`cluster`	`"id"` (default: cluster by `ild_meta(data)$ild_id`) or `"data"`.
`cluster_vec`	When `cluster = "data"`, length-`nrow(data)` vector of cluster IDs aligned to `ild_data` rows.
`coef_fun`	`function(fit)` returning named numeric vector of estimands (default `lme4::fixef`).

seed	Optional integer passed to set.seed() before resampling.
verbose	Logical. If TRUE, warn on failed refits.
...	Passed to lme4::lmer() on each successful replicate (e.g. control). Do not pass weights here; use weights_col on the ILD data.

Value

An object of class ild_msm_bootstrap: replicates (matrix n_boot x p), estimate (point estimates from fit), bootstrap_se, conf_low, conf_high, term_names, n_boot, n_success, n_cluster, metadata, and fit (original). Use [tidy_ild_msm_bootstrap()] for an ild_tidy_schema tibble.

See Also

[ild_msm_inference], [ild_ipw_refit()], [tidy_ild_msm_bootstrap()]

Examples

```
if (requireNamespace("lme4", quietly = TRUE)) {
  set.seed(5001)
  d <- ild_simulate(n_id = 12, n_obs_per = 6, seed = 5001)
  d$stress <- rnorm(nrow(d))
  d <- ild_prepare(d, id = "id", time = "time")
  d <- ild_center(d, y)
  d$.ipw <- runif(nrow(d), 0.8, 1.2)
  f0 <- ild_lme(y ~ y_bp + y_wp + stress + (1 | id), data = d, ar1 = FALSE,
    warn_no_ar1 = FALSE, warn_uncentered = FALSE)
  fw <- ild_ipw_refit(f0, data = d, weights = ".ipw")
  b <- ild_msm_bootstrap(fw, n_boot = 15L, weight_policy = "fixed_weights", seed = 2)
  print(b)
  tidy_ild_msm_bootstrap(b)
}
```

---

ild_msm_contrast_over_time

Compute MSM contrasts over time from a fitted weighted model

Description

Computes marginal treatment contrasts by occasion using model fixed effects. This helper is intentionally standalone: it accepts either an ild_msm_fit object or a weighted lmerMod with attached ild_data.

Usage

```
ild_msm_contrast_over_time(
  object,
  treatment = NULL,
  time_var = ".ild_seq",
```

```
  target_time = "all",
  conf_level = 0.95
)
```

Arguments

object	Output from [ild_msm_fit()] or a lmerMod.
treatment	Optional treatment column name. Defaults to object$estimand$treatment for ild_msm_fit, otherwise required.
time_var	Occasion column (default ".ild_seq").
target_time	Optional target subset: "all" (default), "final", or numeric vector of occasions.
conf_level	Confidence level for Wald intervals (default 0.95).

Value

Tibble with ‘ild_tidy_schema‘ columns plus method, terms, and target_time.

---

ild_msm_diagnose *Diagnose an* ild_msm_fit *result in one call*

Description

Convenience bridge to [ild_diagnose()] using object$fit and object$weights_data.

Usage

```
ild_msm_diagnose(object, ...)
```

Arguments

object	Output from [ild_msm_fit()].
...	Passed to [ild_diagnose()].

Value

An ild_diagnostics_bundle.

`ild_msm_estimand` *Define an MSM estimand specification*

Description

Creates a lightweight estimand object used by [ild_msm_fit()]. v1.1 preserves backward compatibility with v1 static ATE calls while adding explicit slots for regime specification, time targeting, and contrasts.

Usage

```
ild_msm_estimand(
  type = c("ate", "att"),
  regime = "static",
  treatment,
  time_var = ".ild_seq",
  contrast = NULL,
  target_time = "all",
  regime_value = 1,
  dynamic_rule = NULL
)
```

Arguments

`type`	Estimand type. Supports `"ate"` and placeholder `"att"`.
`regime`	Regime class. Supports `"static"` and scaffold `"dynamic"`.
`treatment`	Binary treatment column name.
`time_var`	Occasion index column (default `".ild_seq"`).
`contrast`	Optional contrast definition. Character labels are accepted. Lists may include `label`, `treated`, and `control`.
`target_time`	Optional target occasion: `"all"` (default), `"final"`, or numeric vector.
`regime_value`	For `regime = "static"`, assignment target (default 1).
`dynamic_rule`	For `regime = "dynamic"`, a deterministic rule: function taking `data` and returning binary assignment, or one of `"always_treat"`, `"never_treat"`, `"as_observed"`.

Value

Object of class `ild_msm_estimand`.

Examples

```
e <- ild_msm_estimand(treatment = "trt")
e
```

`ild_msm_fit` *Fit an MSM-style weighted mixed model from an estimand spec*

Description

High-level runner that takes an [ild_msm_estimand()] plus ILD data, builds treatment (and optional censoring) weights, refits a weighted `lmer`, and optionally computes inference via robust SEs or cluster bootstrap.

Usage

```
ild_msm_fit(
  estimand,
  data,
  outcome_formula,
  history,
  history_spec = NULL,
  treatment_engine = c("sequential_msm", "pooled"),
  predictors_censor = NULL,
  weights_col = ".ipw",
  stabilize_treat = c("marginal", "prior_treatment", "none"),
  prior_treatment = NULL,
  stabilize_joint = c("mean1", "none"),
  trim = c(0.01, 0.99),
  inference = c("none", "robust", "bootstrap"),
  robust_type = c("CR2", "CR3", "CR0"),
  n_boot = 200L,
  weight_policy = c("fixed_weights", "reestimate_weights"),
  weights_fn = NULL,
  seed = NULL,
  strict_inference = FALSE,
  ...
)
```

Arguments

`estimand`	Object from [ild_msm_estimand()].
`data`	ILD data.
`outcome_formula`	Outcome model formula for [ild_lme()] (typically includes treatment term).
`history`	Predictors for treatment process weights: one-sided formula (e.g. `~ stress_lag1 + trt_lag1`) or character vector.
`history_spec`	Optional [ild_msm_history_spec()] to build lagged history before weighting.
`treatment_engine`	`"sequential_msm"` (default) or `"pooled"`.

predictors_censor
: Optional character vector for IPCW model. If supplied, joint weights are built with [ild_joint_msm_weights()] and stored in `weights_col`.

`weights_col`
: Column used by [ild_ipw_refit()] (default `".ipw"`).

`stabilize_treat`
: Treatment-weight stabilization mode. For sequential MSM: `"marginal"`, `"prior_treatment"`, `"none"`. For pooled IPTW: mapped to logical `stabilize`.

`prior_treatment`
: Required when `stabilize_treat = "prior_treatment"` in sequential MSM.

`stabilize_joint`
: Joint-weight stabilization for [ild_joint_msm_weights()].

`trim`
: Quantile trimming applied during weight construction.

`inference`
: `"none"` (default), `"robust"`, or `"bootstrap"`.

`robust_type`
: Passed to [ild_robust_se()] and [tidy_ild_model()] when `inference = "robust"`.

`n_boot`
: Passed to [ild_msm_bootstrap()] when `inference = "bootstrap"`.

`weight_policy`
: Passed to [ild_msm_bootstrap()] when `inference = "bootstrap"`.

`weights_fn`
: Passed to [ild_msm_bootstrap()] when `inference = "bootstrap"` and `weight_policy = "reestimate_weights"`.

`seed`
: Optional seed for bootstrap inference.

`strict_inference`
: Logical. If `TRUE`, stop when requested inference or regime dispatch cannot be delivered as requested. If `FALSE` (default), degrade with warning and record machine-readable status.

`...`
: Passed to [ild_lme()].

Details

v1.1 keeps the v1 workflow and adds explicit capability signaling for inference/regime paths that are degraded or unsupported.

Value

Object of class `ild_msm_fit` with elements: `estimand`, `history_spec`, `weights_data`, `fit`, `inference`, `treatment_engine`, `weights_col`. Analysis provenance is attached.

Examples

```
set.seed(31)
d <- ild_simulate(n_id = 12, n_obs_per = 6, seed = 31)
d$stress <- rnorm(nrow(d))
d$trt <- as.integer(stats::rbinom(nrow(d), 1L, 0.45))
d <- ild_prepare(d, id = "id", time = "time")
d <- ild_center(d, y)
d <- ild_lag(d, stress, max_gap = Inf)
d <- ild_lag(d, trt, max_gap = Inf)
est <- ild_msm_estimand(treatment = "trt")
res <- ild_msm_fit(
```

```
  estimand = est,
  data = d,
  outcome_formula = y ~ y_bp + y_wp + stress + trt + (1 | id),
  history = ~ stress_lag1 + trt_lag1,
  predictors_censor = "stress",
  inference = "none",
  warn_no_ar1 = FALSE,
  warn_uncentered = FALSE
)
res$fit
```

`ild_msm_history_spec` *Build a declarative MSM history specification*

Description

Creates a lightweight spec object used by [ild_build_msm_history()] to generate lagged confounder/treatment history columns with deterministic names (e.g. `stress_lag1`, `trt_lag2`).

Usage

```
ild_msm_history_spec(
  vars,
  lags = 1L,
  mode = c("gap_aware", "index", "time_window"),
  max_gap = Inf
)
```

Arguments

`vars`	Character vector of variable names to lag.
`lags`	Integer vector of lag orders. Recycled across `vars`. Defaults to `1L`.
`mode`	Passed to [ild_lag()] (default `"gap_aware"`).
`max_gap`	Passed to [ild_lag()] for `mode = "gap_aware"`.

Value

Object of class `ild_msm_history_spec`.

Examples

```
s <- ild_msm_history_spec(vars = c("stress", "trt"), lags = 1:2)
s
```

`ild_msm_overlap_plot` *Propensity overlap plot (pooled or sequential MSM IPTW)*

Description

For **pooled** [ild_iptw_weights()], uses `attr(data, "ild_iptw_fit")` and plots the distribution of fitted $P(A = 1 \mid L)$ by treatment level.

Usage

```
ild_msm_overlap_plot(data, treatment, source = c("auto", "pooled", "msm"))
```

Arguments

`data`	ILD with treatment column and relevant `glm` attributes.
`treatment`	Binary treatment column name.
`source`	Character: `"auto"` (prefer MSM fits if present), `"pooled"`, or `"msm"`.

Details

For **sequential** [ild_iptw_msm_weights()], uses `attr(data, "ild_iptw_msm_fits")` and facets by occasion: fitted denominator propensity at each t.

Value

A `ggplot` object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(3)
  d <- ild_simulate(n_id = 15, n_obs_per = 5, seed = 3)
  d$stress <- rnorm(nrow(d))
  d$trt <- as.integer(stats::rbinom(nrow(d), 1L, 0.45))
  x <- ild_prepare(d, id = "id", time = "time")
  x <- ild_iptw_weights(x, treatment = "trt", predictors = "stress")
  ild_msm_overlap_plot(x, treatment = "trt", source = "pooled")
}
```

`ild_msm_recovery` *MSM recovery simulation harness*

Description

Repeats an MSM analysis over simulated datasets with known `true_ate` and summarizes estimation bias, RMSE, CI coverage, and positivity stress summaries.

Usage

```
ild_msm_recovery(
  n_sim = 100L,
  n_id = 80L,
  n_obs_per = 10L,
  true_ate = 0.5,
  n_boot = 200L,
  inference = c("bootstrap", "robust", "none"),
  seed = 1001L,
  censoring = TRUE,
  scenario_grid = NULL
)
```

Arguments

`n_sim`	Number of simulations.
`n_id`	Number of persons per simulation.
`n_obs_per`	Planned observations per person.
`true_ate`	True additive treatment effect.
`n_boot`	Bootstrap replicates when `inference = "bootstrap"`.
`inference`	`"bootstrap"` (default), `"robust"`, or `"none"`.
`seed`	Base seed.
`censoring`	If `TRUE`, include monotone dropout in DGP and fit IPCW+joint pipeline.
`scenario_grid`	Optional data frame/list of scenario settings. Supported columns: `scenario_id`, `n_id`, `n_obs_per`, `true_ate`, `censoring`, `positivity_stress`,`treatment_strength`, `censoring_strength`, `misspec_treatment_model`.

Value

List with `summary`, `summary_by_scenario`, and `sim_results`.

Examples

```
## Not run:
out <- ild_msm_recovery(n_sim = 10, n_id = 40, n_obs_per = 10, true_ate = 0.5, n_boot = 50)
out$summary

## End(Not run)
```

`ild_msm_simulate_scenario` *Simulate a simple longitudinal MSM scenario*

Description

Generates ILD with time-varying treatment, confounding, and optional monotone censoring/dropout under a known treatment effect (`true_ate`).

Usage

```
ild_msm_simulate_scenario(
  n_id = 60L,
  n_obs_per = 10L,
  true_ate = 0.5,
  censoring = TRUE,
  positivity_stress = 1,
  treatment_strength = 0.8,
  censoring_strength = 0.6,
  seed = 42L
)
```

Arguments

`n_id`	Number of persons.
`n_obs_per`	Planned observations per person.
`true_ate`	True additive treatment effect on outcome.
`censoring`	If `TRUE`, apply monotone dropout and return observed rows only.
`positivity_stress`	Multiplier for treatment/censoring logits (>1 worsens overlap; <1 improves overlap).
`treatment_strength`	Base strength of treatment dependence on confounders/history.
`censoring_strength`	Base strength of dropout dependence on stress/treatment.
`seed`	Integer seed.

Value

ILD object with columns `stress` and binary `trt`; attributes include `true_ate`.

Examples

```
d <- ild_msm_simulate_scenario(n_id = 20, n_obs_per = 8, true_ate = 0.5, seed = 12)
ild_summary(d)$summary
```

`ild_panel_lag_prepare` *Prepare several lag columns and audit them*

Description

Sequentially applies [ild_lag()] to each named variable with the same `mode`, `max_gap`, `window`, and `resolution`, then runs [ild_check_lags()] once on all created lag columns. Use this when building multivariate lag specifications (e.g. several predictors or lag orders) with consistent gap or time-window semantics.

Usage

```
ild_panel_lag_prepare(
  data,
  variables,
  n = 1L,
  mode = c("gap_aware", "index", "time_window"),
  max_gap = NULL,
  window = NULL,
  resolution = c("closest_prior", "last_in_window", "mean_in_window")
)
```

Arguments

`data`	An ILD object (see [is_ild()]).
`variables`	Character vector of column names to lag.
`n`	Integer vector of lag orders, recycled to `length(variables)`. Ignored for `mode = "time_window"` (see [ild_lag()]).
`mode`, `max_gap`, `window`, `resolution`	Passed to each [ild_lag()] call.

Value

A list with `data` (ILD with lag columns), `lag_vars` (names of created columns), `check` (tibble from [ild_check_lags()]), and `spec` (list of arguments for provenance).

See Also

`vignette("temporal-dynamics-model-choice", package = "tidyILD")`, [ild_crosslag()], [ild_lag()].

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",
    "predicted_trajectory", "residual_acf"),
  var = NULL,
  id_var = ".ild_id",
  time_var = c(".ild_time_num", ".ild_seq"),
  max_ids = 20L,
  seed = 42L,
  facet_by = NULL,
  ...
)
```

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), ‘"predicted_trajectory"‘ (observed and fitted lines vs time; 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/‘predicted_trajectory‘, x-axis: ‘.ild_time_num‘ or ‘.ild_seq‘.
max_ids	For trajectory and ‘predicted_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).
facet_by	Optional character: name of a column in the ILD (e.g. cluster or site) to pass to [ggplot2::facet_wrap()] for ‘trajectory‘, ‘heatmap‘, ‘gaps‘, and ‘predicted_trajectory‘.
...	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_plot_filtered_vs_smoothed

Plot filtered vs smoothed state (first state)

Description

Compares one-step filtered state (`att`) with smoothed (`alphahat`) when available.

Usage

```
ild_plot_filtered_vs_smoothed(x)
```

Arguments

x	An object from [ild_kfas()].

Value

A ggplot object.

`ild_plot_forecast` *Plot forecast or future simulation (stub when no horizon)*

Description

When `forecast_horizon > 0` was used in [ild_kfas()], attempts a short ahead forecast via **KFAS**; otherwise returns an informative empty panel.

Usage

```
ild_plot_forecast(x, ...)
```

Arguments

`x`	An object from [ild_kfas()].
`...`	Passed to `predict` for the fitted **KFAS** model when forecasting is available.

Value

A ggplot object.

`ild_plot_predicted_trajectory`
Observed and fitted values vs time (trajectory overlay)

Description

Uses [augment_ild_model()] or [ild_augment()] on the fitted object and plots two lines per person (observed vs fitted) against ‘time_var‘. For a scatter of observed vs fitted, use [ild_plot()] with ‘type = "fitted"‘.

Usage

```
ild_plot_predicted_trajectory(
  fit,
  time_var = c(".ild_time_num", ".ild_seq"),
  max_ids = 20L,
  seed = 42L,
  facet_by = NULL
)
```

Arguments

fit	Model from [ild_lme()] or [ild_brms()] (must carry ‘ild_data‘).
time_var	‘.ild_time_num‘ or ‘.ild_seq‘ (default first of these in [match.arg()]).
max_ids, seed	Passed through for subsampling persons.
facet_by	Optional column name in ‘ild_data‘ for [ggplot2::facet_wrap()].

Value

A ‘ggplot‘ object.

---

ild_plot_states *Plot smoothed latent states (first state by default)*

Description

Plot smoothed latent states (first state by default)

Usage

```
ild_plot_states(x, state_index = 1L, ...)
```

Arguments

x	An object from [ild_kfas()].
state_index	Integer; which column of alphahat to plot (default 1L).
...	Passed to [ggplot2::labs()].

Value

A ggplot object.

---

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).

See Also

[ild_recovery_metrics()] to summarize bias, RMSE, and coverage from return_sims. vignette("benchmark-simulation-
package = "tidyILD") (worked example).

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 = NULL,
  time = NULL,
  gap_threshold = Inf,
  duplicate_handling = c("first", "last", "error", "collapse"),
  collapse_fn = NULL,
  input_format = c("long", "wide"),
  wide_cols = NULL,
  wide_names_pattern = "^(.+)_t(.+)$",
  wide_time_parser = c("numeric", "date", "datetime"),
  wide_time_format = NULL,
```

```
  wide_keep_cols = NULL
)
```

Arguments

data	A data frame or tibble with at least an id and a time column. If id and time are both omitted, data must be a tbl_ts from **tsibble** (requires a single key column and an index); key and index names are inferred (see [ild_as_tsibble()] for the reverse). **Tsibble interoperability:** Accepting a tbl_ts is an input convenience; the result is always a plain ILD tibble (the tbl_ts class is dropped). Metadata from the source tsibble (key, index, interval summary, regularity) is stored in attr(x, "tidyILD")$tsibble when available; use [ild_tsibble_meta()] to read it. Use [ild_as_tsibble()] for a best-effort round-trip. **Conceptual choice (keys):** ILD uses one subject identifier per row; tbl_ts must have exactly one key column. Compound keys are not supported—combine levels into a single id column before calling ild_prepare().
id	Character. Name of the subject/unit identifier column. Omit both id and time when data is a tsibble (see above).
time	Character. Name of the time column (Date, POSIXct, or numeric).
gap_threshold	Numeric. Time distance above which an interval is flagged as a gap (‘.ild_gap‘ TRUE). Same units as the numeric time (e.g. seconds if time is POSIXct). Use ‘Inf‘ to disable gap flagging.
duplicate_handling	Character. How to handle duplicate timestamps within the same id: ‘"first"‘ (keep first), ‘"last"‘ (keep last), ‘"error"‘ (stop with an error), ‘"collapse"‘ (aggregate with collapse_fn).
collapse_fn	Named list of functions, one per variable to collapse. Used only when duplicate_handling = "collapse". E.g. list(x = mean, y = function(z) z[1]). Variables not in collapse_fn keep their first value within the duplicate group.
input_format	Character. Input data layout: "long" (default) or "wide". Wide inputs are converted to canonical long form before validation; downstream architecture is unchanged.
wide_cols	Character vector of wide measurement columns to convert when input_format = "wide". If NULL, all non-id/non-keep columns must match wide_names_pattern.
wide_names_pattern	Regular expression with exactly two capture groups: measure and time token. Default "^(.+)_t(.+)$" parses names such as mood_t1 and stress_t2024-01-01.
wide_time_parser	Character. How to parse extracted time tokens: "numeric" (default), "date", or "datetime".
wide_time_format	Optional format string passed to as.Date() or as.POSIXct() when parsing wide time tokens.
wide_keep_cols	Optional character vector of baseline/static columns to carry across all generated long rows when input_format = "wide".

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.

See Also

`vignette("tsibble-interoperability", package = "tidyILD")` (tsibble ingestion, provenance, `ild_as_tsibble()`).

`ild_prior_ild` *Prior specifications for common ILD mixed models*

Description

Returns a `brmsprior` object suitable for the `prior` argument of [ild_brms()] / [brms::brm()]. Templates follow weakly informative defaults: Student-t on intercepts and group SDs, normal on regression coefficients, and exponential on residual SD where applicable.

Usage

```
ild_prior_ild(
  template = c("default", "weakly_informative", "minimal_shrinkage")
)
```

Arguments

`template`	Character selecting the template:
	`default` Student-t intercept; Normal(0, 5) fixed effects; Student-t group SDs; exponential residual SD (Gaussian models).
	`weakly_informative` Wider scales (more diffuse) on intercept and `b` coefficients.
	`minimal_shrinkage` Tighter Normal(0, 2.5) on `b` (similar to Gelman et al. scaled logistic guidance; use with scaled predictors).

Value

A `brmsprior` data frame (class `brmsprior`).

See Also

[ild_brms()]

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_recovery_metrics *Recovery metrics from simulation replications*

Description

Summarizes **bias**, **RMSE**, and **nominal Wald interval coverage** for a known coefficient using the per-replication table from [ild_power()] when `return_sims = TRUE`. Purely descriptive; does not run new simulations.

Usage

```
ild_recovery_metrics(sim_results, truth, level = 0.95, converged_only = TRUE)
```

Arguments

`sim_results`	A data frame or tibble with columns `estimate`, `std_error`, and `converged` (as in `ild_power(..., return_sims = TRUE)$sim_results`).
`truth`	Known value of the estimand (e.g. `effect_size` in [ild_power()]).
`level`	Nominal coverage level for Wald intervals (default `0.95`).
`converged_only`	If `TRUE` (default), only rows with `converged == TRUE` are used.

Value

A one-row tibble with truth, n (rows used), n_total (rows in sim_results if converged_only), mean_estimate, bias, rmse, coverage, level.

See Also

[ild_power()]

`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 (includes guardrails_narrative and guardrails when [ild_diagnose()] succeeds), provenance, provenance_export_path (character or NULL). When guardrails are triggered, methods_with_guardrails repeats the methods paragraph with guardrails appended (same pattern as ild_methods(fit, bundle = d)).

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()`.

Details

For fits from [ild_ipw_refit()] with **estimated** IPW/IPCW weights, these SEs apply to the **weighted outcome (mixed) model** only; they do **not** incorporate first-stage uncertainty from the propensity/censoring models unless you treat weights as fixed. For MSM-style inference, prefer [ild_msm_bootstrap()] with an appropriate `weight_policy`; see [ild_msm_inference].

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`, [ild_msm_bootstrap()], [ild_msm_inference].

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`.

See Also

`vignette("benchmark-simulation-recovery", package = "tidyILD")` (simulation benchmarks and DGP description).

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"‘.

See Also

`vignette("kfas-irregular-timing-spacing", package = "tidyILD")` (KFAS and irregular timing). `vignette("ild-decomposition-and-spacing", package = "tidyILD")` (within-between and spacing).

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, facet_by = NULL, ...)
```

Arguments

x	ILD object or fitted model.
var	Variable to plot. If NULL, single data column is used.
facet_by	Optional column in ILD for [ggplot2::facet_wrap()] (e.g. cluster).
...	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`. If `ild_prepare()` was run on a `tbl_ts`, `tsibble` is also present (same as [ild_tsibble_meta()]). The `summary` tibble is the primary contract for programmatic use.

ild_tidy *Tidy fixed effects from an ILD model fit (S3 generic)*

Description

Dispatches to [tidy_ild_model()] for `lmerMod` and `lme` objects from [ild_lme()], or the `brmsfit` method for [ild_brms()]. All methods return tables conforming to `ild_tidy_schema`.

Usage

```
ild_tidy(x, ...)

## S3 method for class 'lmerMod'
ild_tidy(x, ...)

## S3 method for class 'lme'
ild_tidy(x, ...)

## Default S3 method:
ild_tidy(x, ...)

## S3 method for class 'brmsfit'
ild_tidy(x, intervals = TRUE, object = FALSE, ...)

## S3 method for class 'ild_fit_ctsem'
ild_tidy(x, ...)

## S3 method for class 'ild_fit_kfas'
ild_tidy(x, ...)
```

Arguments

`x`	A fitted model from [ild_lme()].
`...`	Passed to [tidy_ild_model()].
`intervals`	Logical. For `brmsfit` from [ild_brms()], include posterior `rhat`, `ess_bulk`, `ess_tail` (default `TRUE`).
`object`	Logical. Passed through for non-brms fits; ignored for `brmsfit`.

ild_tidy_schema *Column dictionary for ild_tidy() outputs*

Description

Minimum stable contract for parameter-level tables returned by `ild_tidy`. Implementations (`tidy_ild_model`, `ild_tidy.brmsfit`) emit all required columns; `conf_low` / `conf_high` replace legacy `ci_low` / `ci_high`.

Usage

```
ild_tidy_schema()
```

Details

Required columns: `term`, `component`, `effect_level`, `estimate`, `std_error`, `conf_low`, `conf_high`, `statistic`, `p_value`, `interval_type`, `engine`, `model_class`.

Optional: `rhat`, `ess_bulk`, `ess_tail`, `pd`, `rope_low`, `rope_high`.

`interval_type`: Frequentist rows are typically `Wald` (model or robust; see `tidy_ild_model`). Bootstrap output from `tidy_ild_msm_bootstrap` uses `bootstrap_percentile` (equal-tailed over replicate coefficients).

Value

A list with `required` and `optional` character vectors.

See Also

`ild_augment_schema`, `ild_diagnostics_bundle`. Full semantics for `component`, `effect_level`, and `interval_type`: `vignette("developer-contracts", package = "tidyILD")`.

---

ild_tidy_states *Tidy per-time state summaries for ild_kfas*

Description

Returns a tibble with one row per time index and columns for the smoothed level (and optional variance). Separate from [ild_tidy()] which tidies **parameters**.

Usage

```
ild_tidy_states(x, ...)
```

Arguments

`x`	An object from [ild_kfas()].
`...`	Reserved.

Value

A `tibble`.

ild_tsibble_meta *Tsibble provenance from an ILD object*

Description

If [ild_prepare()] was called on a `tbl_ts`, metadata from the source tsibble (key, index, interval summary, regularity) is stored in `attr(x, "tidyILD")$tsibble`. Returns `NULL` if the object was not prepared from a tsibble.

Usage

```
ild_tsibble_meta(x)
```

Arguments

`x`	An object that passes [validate_ild()].

Value

A named list or `NULL`.

See Also

[ild_prepare()], [ild_as_tsibble()], `vignette("tsibble-interoperability", package = "tidyILD")`

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 conforming to `ild_tidy_schema`: required columns include `term`, `component`, `effect_level`, `estimate`, `std_error`, `conf_low`, `conf_high`, `statistic`, `p_value`, `interval_type`, `engine`, `model_class`; optional Bayesian columns are `NA` for these engines. `component` is `"fixed"` for coefficient rows; `effect_level` is inferred conservatively from term names (e.g. `_wp`/`_bp` suffixes) or `"population"` for the intercept, `"unknown"` when ambiguous.

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`).

Details

With `object = TRUE`, returns an object of class `tidyild_model` (meta + table) for use with `print.tidyild_model`.

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. For **IPW / MSM** weighted `lmer` fits ([ild_ipw_refit()]), robust SEs do not account for estimated weights; use [ild_msm_bootstrap()] and [tidy_ild_msm_bootstrap()] when you need bootstrap CIs; see [ild_msm_inference].

`tidy_ild_msm_bootstrap`

Tidy fixed effects from `ild_msm_bootstrap`

Description

Returns a tibble matching `ild_tidy_schema` with `interval_type` `bootstrap_percentile` (equal-tailed over replicate coefficients).

Usage

```
tidy_ild_msm_bootstrap(x, conf_level = 0.95, ...)
```

Arguments

`x`	Object from [ild_msm_bootstrap()].
`conf_level`	Numeric. Used for implied tail quantiles (default 0.95).
`...`	Unused.

Value

A tibble; see `ild_tidy_schema`.

See Also

[ild_msm_bootstrap()], [ild_msm_inference]

`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

∗ **datasets**
ema_example, 9
∗ **ild_diagnostics_utilities**
ild_design_check, 26
ild_ipcw_weights, 38
ild_iptw_msm_weights, 40
ild_iptw_weights, 42
ild_ipw_weights, 45
ild_joint_msm_weights, 46
ild_missing_cohort, 54
ild_missing_compliance, 55
ild_missing_hazard_first, 56
ild_missing_model, 57
ild_missing_pattern, 58
ild_missingness_report, 59

as_ild, *5*, 7
augment.ild_lme, *5*
augment.ild_lme *(*broom_ild_lme*)*, 8
augment_ild_model, *5*, 8, *13*

broom_ild_lme, *5*, 8
browseVignettes, *7*

ema_example, *5*, 9

guardrail_registry, *5*, 9, *30, 31*, *33*

ild_acf, *5*, 10
ild_align, *5*, 10
ild_as_tsibble, *5*, 12
ild_augment, *5*, 12, *13*
ild_augment_schema, *5*, *8*, *12*, 13, *33*, *90*
ild_augment_states, 14
ild_autoplot, *5, 6*, 15, *30*, *32*
ild_autoplot.ild_heterogeneity *(*ild_heterogeneity*)*, 36
ild_brms, *5*, 16
ild_brms_dynamics_formula, 17
ild_build_msm_history, 17, *33*
ild_bundle, *5*, 18
ild_center, *5, 6*, 19
ild_center_plot, *5*, 20
ild_check_lags, *5*, 20
ild_circadian, *5*, 21
ild_compare_fits, *5*, 22
ild_compare_pipelines, *5*, 22
ild_crosslag, *5*, 23
ild_ctsem, 24
ild_decomposition, *5*, 25
ild_design_check, *5*, 26, *32, 33*, *39*, *41*, *43*, *46, 47*, *55, 56*, *58–60*
ild_diagnose, *5, 6*, *9*, 27, *32, 33*
ild_diagnostics, *5*, 30, *30*, *33*
ild_diagnostics_bundle, *5*, *9*, *14*, *16*, *27*, *29, 30*, 31, *33*, *45*, *53*, *57*, *59*, *64*, *90*
ild_diagnostics_utilities, *5*, *30*, *32*, 33, *33*
ild_export_provenance, *5*, 34
ild_fit, *5*, 34
ild_heatmap, *5, 6*, 35
ild_heterogeneity, *5, 6*, 36
ild_heterogeneity.brmsfit *(*ild_heterogeneity*)*, 36
ild_heterogeneity.lme *(*ild_heterogeneity*)*, 36
ild_heterogeneity.lmerMod *(*ild_heterogeneity*)*, 36
ild_heterogeneity_stratified, *5*
ild_heterogeneity_stratified *(*ild_heterogeneity*)*, 36
ild_history, *5*, 38
ild_ipcw_weights, *27*, *33*, 38, *41*, *43*, *46, 47*, *55, 56*, *58–60*
ild_iptw_msm_weights, *27*, *33*, *39*, 40, *43*, *46, 47*, *55, 56*, *58–60*
ild_iptw_weights, *27*, *33*, *39*, *41*, 42, *46, 47*, *55, 56*, *58–60*
ild_ipw_ess, *33*, 43
ild_ipw_refit, *5*, *33*, 44

ild_ipw_weights, *5*, *27*, *32, 33*, *39*, *41*, *43*, 45, *47*, *55, 56*, *58–60*
ild_joint_msm_weights, *27*, *33*, *39*, *41*, *43*, 46, *46*, *55, 56*, *58–60*
ild_kfas, *5*, 47
ild_lag, *5, 6*, 48
ild_lme, *5–7*, 49
ild_manifest, *5*, 51
ild_meta, *5*, 52
ild_methods, *5*, 52, *83*
ild_missing_bias, *5*, 53
ild_missing_cohort, *5*, *27*, *39*, *41*, *43*, *46, 47*, 54, *56*, *58–60*
ild_missing_compliance, *5*, *27*, *39*, *41*, *43*, *46, 47*, 55, *55, 56*, *58–60*
ild_missing_hazard_first, *5*, *27*, *39*, *41*, *43*, *46, 47*, *55*, 56, *56*, *58–60*
ild_missing_model, *5*, *27*, *29*, *32, 33*, *39*, *41*, *43*, *46, 47*, *55, 56*, 57, *59, 60*
ild_missing_pattern, *5*, *27*, *32, 33*, *39*, *41*, *43*, *46, 47*, *55, 56*, 58, *58*, *60*
ild_missingness_report, *5*, *27*, *39*, *41*, *43*, *46, 47*, *55, 56*, *58*, 59, *59*
ild_msm_balance, *33*, 61
ild_msm_bootstrap, *33*, 62
ild_msm_contrast_over_time, *33*, 63
ild_msm_diagnose, *33*, 64
ild_msm_estimand, *33*, 65
ild_msm_fit, *33*, 66
ild_msm_history_spec, *33*, 68
ild_msm_inference, *33*
ild_msm_overlap_plot, *33*, 69
ild_msm_recovery, *33*, 70
ild_msm_simulate_scenario, 71
ild_panel_lag_prepare, *5*, 72
ild_person_distribution, *5, 6*, 73
ild_person_model, *5, 6*, 73
ild_plot, *5, 6*, 74
ild_plot_filtered_vs_smoothed, *6*, 75
ild_plot_forecast, *6*, *48*, 76
ild_plot_predicted_trajectory, *5, 6*, 76
ild_plot_states, *6*, 77
ild_power, *5*, 77
ild_prepare, *5*, *7*, 79
ild_prior_ild, *5*, 81
ild_provenance, *5*, 82
ild_recovery_metrics, 82
ild_report, *5*, 83
ild_robust_se, *5*, 84
ild_simulate, *5*, 85
ild_spacing, *5*, 86
ild_spacing_class, *5, 6*, 87
ild_spaghetti, *5, 6*, 87
ild_summary, *5*, 88
ild_tidy, *5*, 89, *89*
ild_tidy.brmsfit, *89*
ild_tidy.ild_heterogeneity *(*ild_heterogeneity*)*, 36
ild_tidy_schema, *5*, *14*, *33*, *63*, 89, *89*, *94, 95*
ild_tidy_states, 90
ild_tsibble_meta, 91
ild_tvem, *5*, 91
ild_tvem_plot, *5, 6*, 92
is_ild, *5*, 93
is_ild_diagnostics_bundle *(*ild_diagnostics_bundle*)*, 31

lmer, *62*

new_ild_diagnostics_bundle *(*ild_diagnostics_bundle*)*, 31

plot_ild_diagnostics, 93
predict, *48*, *76*
print, *32*

tibble, *14*, *83*, *90*
tidy.ild_lme, *5*
tidy.ild_lme *(*broom_ild_lme*)*, 8
tidy_ild_model, *5*, *89, 90*, 94
tidy_ild_msm_bootstrap, *33*, *90*, 95
tidyILD *(*tidyILD-package*)*, 4
tidyILD-package, 4

validate_ild, *5*, 96