Package: fixes (via r-universe)

October 21, 2026

Type Package

Title Event Studies and Inference for Difference-in-Differences

Version 1.0.0

Date 2026-10-21

Description Provides tools for event-study analysis and inference in difference-in-differences (DiD) designs with panel data, from the classic two-way fixed-effects (TWFE) event study to estimators for staggered adoption. Includes event_study() for event-study curves (dynamic effects by relative time), att() for aggregated ATT estimation (overall, by cohort, by calendar time), did() for basic two-way fixed-effects DiD, and honest_sensitivity() for sensitivity analysis under violations of parallel trends (Rambachan and Roth, 2023), all running on an internal C++ fixed-effects OLS engine. Supports multiple modern estimators: Callaway-Sant'Anna (2021), Sun-Abraham (2021), Borusyak-Jaravel-Spiess (2024), Wooldridge TWM, and Deb et al. FLEX. Results plot via base plot() methods.

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, broom, tibble, rlang, lifecycle (>= 1.0.2), Rcpp, scales, parallel

License MIT + file LICENSE

Encoding UTF-8

Suggests fixest, knitr, rmarkdown, haven, testthat, plotly, tidyr, did, didimputation, modelsummary, tinytable, lpSolveAPI, Rglpk, TruncatedNormal, Matrix, pracma, HonestDiD

VignetteBuilder knitr

URL https://github.com/yo5uke/fixes, https://yo5uke.com/fixes/

BugReports https://github.com/yo5uke/fixes/issues

Config/roxygen2/version 8.1.0

Config/roxygen2/markdown TRUE

LinkingTo Rcpp, RcppArmadillo

Config/pak/sysreqs libicu-dev

Repository https://yo5uke.r-universe.dev

Date/Publication 2026-10-21 08:53:14 UTC

RemoteUrl https://github.com/yo5uke/fixes

RemoteRef HEAD

RemoteSha 0a1e1cbba31bd344b7599273f973f9c850247c92

Contents

att 2
att_gt 5
autoplot.es_result 6
contamination_weights 7
did 8
event_study 11
glance.att_result 16
glance.did_result 17
glance.es_result 18
honest_sensitivity 18
plot.att_gt_result 20
plot.att_result 22
plot.es_result 23
plot.honest_result 25
plot.sa_contamination_weights 25
tidy.att_result 27
tidy.did_result 28
tidy.es_result 29

Index **30**

`att` *ATT aggregation for staggered adoption designs*

Description

Estimates average treatment effects on the treated (ATT) using either the Callaway-Sant'Anna (2021) or Borusyak-Jaravel-Spiess (2024) estimator, aggregated to a single summary effect (`"simple"`), per-cohort effects (`"by_cohort"`), or per calendar-time effects (`"by_time"`).

Usage

```
att(
  formula,
  data,
  time,
  timing = NULL,
  unit = NULL,
  estimator = c("cs", "bjs"),
  aggregation = c("simple", "by_cohort", "by_time"),
  control_group = c("nevertreated", "notyettreated"),
  anticipation = 0L,
  conf_level = 0.95,
  interval = 1,
  time_transform = FALSE,
  outcome
)
```

Arguments

formula	A model formula `outcome ~ treatment | fixed effects` (see `event_study()`). The fixed effects only serve to name the unit: both estimators absorb unit and time effects themselves, and neither takes covariates.
data	A data.frame containing panel data. `formula` and `data` may come in either order.
time	Unquoted calendar time variable (numeric).
timing	Treatment timing: an unquoted column giving each unit's first treatment period (`NA` = never treated), or a single treatment date. Omit it when the formula holds a D column. A timing column must have the same value on every row of a unit; otherwise `att()` stops with an error naming the offending units.
unit	Unquoted unit identifier; taken from the fixed effects when they are `time` plus one other column.
estimator	Estimation strategy: `"cs"` (Callaway-Sant'Anna 2021, default) or `"bjs"` (Borusyak-Jaravel-Spiess 2024).
aggregation	Aggregation type: `"simple"` (overall ATT, default), `"by_cohort"` (one ATT per treatment cohort), or `"by_time"` (one ATT per calendar time period).
control_group	For `estimator = "cs"`: comparison group, `"nevertreated"` (default) or `"notyettreated"`.
anticipation	For `estimator = "cs"`: number of anticipation periods before treatment (non-negative integer, default `0L`).
conf_level	Numeric confidence level(s) (default `0.95`). Multiple levels are supported, e.g., `c(0.90, 0.95)`.
interval	Numeric time spacing (default `1`; informational only).
time_transform	Logical; if `TRUE`, creates consecutive integer time within unit via `dplyr::dense_rank()`. A timing column or value is then read as an index; a D column needs no adjustment.

`outcome` **[Deprecated]** The unquoted outcome of the older interface. Still works in fixes 1.x, with a deprecation warning that shows the equivalent formula call; fixes 2.0.0 is planned to accept only `formula`. Name it when you use it: the second argument is now `data`.

Details

This function complements `event_study()`: use `event_study()` when you want a full event-study curve (dynamic effects by relative time), and `att()` when you want aggregated ATT estimates that collapse the time dimension. Both take the same formula, and treatment is given in the same three ways (see `event_study()`, **Specifying treatment**):

```
att(y ~ D | id + year, data = df, time = year)                 # D column
att(y ~ 1 | id + year, data = df, time = year, timing = g)     # timing column
att(y ~ treated | id + year, data = df, time = year, timing = 2005)
```

The result has `broom::tidy()` and `broom::glance()` methods, so it can be passed to `modelsummary::modelsummary()` (rendered with `tinytable`) to build a publication table — including stacking several estimators (e.g. `did()` TWFE, CS, and BJS) side by side in one table.

`att()` supersedes `calc_att()` as of fixes 1.0.0.

Value

A `data.frame` of class `"att_result"` with columns:

`group` Cohort or calendar time (`NA` for `"simple"`).

`estimate` ATT point estimate.

`std.error` Standard error.

`statistic` t-statistic (`estimate / std.error`).

`p.value` Two-sided p-value (normal approximation).

`conf_low_XX`, `conf_high_XX` CI bounds for each `conf.level`.

Attributes: `aggregation`, `estimator`, `conf.level`, `N`, `N_units`, `N_treated`, `N_nevertreated`, `unit_var`, `time_var`, `control_group` (CS only), `att_gt` (CS raw ATT(g,t) table), `tau_it` (BJS unit-time effects table).

Aggregation formulas (CS estimator)

- **simple**: $\theta = \sum_g (n_g / n_{treated}) \cdot \overline{ATT(g,\cdot)}$ where $\overline{ATT(g,\cdot)}$ is the mean over post-treatment periods.
- **by_cohort**: $\theta(g) = \overline{ATT(g,\cdot)}$ per cohort.
- **by_time**: $\theta(t) = \sum_{g \le t} w(g,t) \cdot ATT(g,t)$ with $w(g,t) = n_g / \sum_{g' \le t} n_{g'}$.

Rows with a missing or non-finite outcome, time, or unit are dropped and the panel is balanced by dropping units not observed in every period, as `did::att_gt()` does by default; both steps warn, and `N`, `N_units`, `N_treated` and `N_nevertreated` describe the balanced sample.

Standard errors

For "cs", the SE of each aggregate is computed from its unit-level influence function, which carries the covariance across the ATT(g,t) it averages and the estimation of the cohort weights (the same SEs as did::aggte()).

For "bjs", the SE is the conservative cluster-robust SE of Borusyak et al. (2024, Theorem 3), clustered by unit, including the uncertainty from the imputation regression. It matches didimputation::did_imputation().

See Also

event_study() for event-study (dynamic) estimates, and calc_att() for the deprecated verb-style predecessor.

Examples

```
## Not run:
att(y ~ 1 | id + year, data = df, time = year, timing = g)
att(y ~ D | id + year, data = df, time = year,
    estimator = "bjs", aggregation = "by_cohort")

## End(Not run)
```

---

att_gt *Extract the ATT(g,t) table from a Callaway-Sant'Anna result*

Description

Returns the cohort-by-period ATT(g,t) table computed by event_study(estimator = "cs") or att(estimator = "cs") as a real att_gt_result object with its own plot() and autoplot() methods. Bootstrap metadata (when the originating call used bootstrap = TRUE) is carried along so simultaneous-inference overlays keep working.

Usage

```
att_gt(x)
```

Arguments

x	An es_result or att_result object estimated with estimator = "cs".

Value

A data frame of class c("att_gt_result", "data.frame") with columns g (cohort), t (calendar time), estimate, and std_error.

See Also

plot.att_gt_result(), event_study(), att()

Examples

```
## Not run:
res <- event_study(y ~ 1 | id + year, data = df, time = year, timing = g,
                   estimator = "cs")
gt <- att_gt(res)
plot(gt)                  # heatmap
plot(gt, type = "facet")  # one panel per cohort

## End(Not run)
```

`autoplot.es_result` *Autoplot for event-study results*

Description

S3 method that plots an `es_result` (from `event_study()`). It forwards arguments to `plot.es_result()`.

Usage

```
## S3 method for class 'es_result'
autoplot(
  object,
  ci_level = 0.95,
  type = c("errorbar", "ribbon"),
  time_axis = c("relative", "calendar"),
  ...
)
```

Arguments

`object`	An `es_result` returned by `event_study()`.
`ci_level`	Confidence level (numeric, e.g., 0.95).
`type`	Plot type: `"errorbar"` (default) or `"ribbon"`.
`time_axis`	`"relative"` (default) for event time, or `"calendar"` for the original time values; see `plot.es_result()`.
`...`	Additional styling arguments; see `plot.es_result()`.

Value

A `ggplot` object.

Examples

```
# res <- event_study(...)
# ggplot2::autoplot(res, ci_level = 0.95, type = "ribbon")
```

contamination_weights *Contamination Weights for TWFE Event-Study Coefficients (Sun-Abraham 2021)*

Description

Estimates the contamination weights $\omega_{e,\ell}^{\ell''}$ that decompose each TWFE event-study coefficient $\hat{\mu}_{\ell''}$ into a linear combination of cohort-specific ATTs (CATTs):

$$\hat{\mu}_{\ell''} = \sum_{(e,\ell)} \omega_{e,\ell}^{\ell''} \cdot \widehat{CATT}_{e,\ell}$$

(Sun and Abraham 2021, Equation 20).

The weights are obtained via the OVB auxiliary regression (Eq. 12): for each cohort-period CATT cell (e, ℓ), regress the cohort-specific indicator $1\{E_i = e\} \cdot 1\{t - E_i = \ell\}$ on all cohort-aggregated relative-time indicators $D_{i,t}^{\ell''}$ and two-way fixed effects. The resulting regression coefficient on $D_{i,t}^{\ell''}$ is $\omega_{e,\ell}^{\ell''}$.

Usage

```
contamination_weights(data, time, timing, unit, fe = NULL, baseline = -1L)
```

Arguments

data	A data.frame with one row per unit-period (balanced panel).
time	Unquoted name of the calendar time variable (numeric).
timing	Unquoted name of the first-treatment-period variable; NA marks never-treated units. The value must be the same on every row of a unit.
unit	Unquoted name of the unit identifier.
fe	One-sided fixed-effects formula, e.g. ~ id + year. When NULL (default), falls back to ~ <unit> + <time> with a warning.
baseline	Integer reference (baseline) period excluded from the TWFE specification (default -1L). Must match the baseline argument used in event_study().

Value

An object of class c("sa_contamination_weights", "data.frame") with one row per (catt_cohort, catt_period, tw triple, and columns:

catt_cohort Cohort e (first treatment period).

catt_period Relative event time ℓ of the CATT.

twfe_period Relative event time ℓ'' of the TWFE coefficient being decomposed.

weight Contamination weight $\omega_{e,\ell}^{\ell''}$.

is_own Logical; TRUE when catt_period == twfe_period.

Attributes: baseline, cohorts, cohort_sizes, incl_periods.

Interpretation

- **Own-period cell** (`catt_period == twfe_period`): $\omega^{\ell}_{e,\ell}$ represents the weight the TWFE estimator places on $CATT_{e,\ell}$. Under the SA IW estimator these equal the cohort-size weights $n_e / \sum n_{e'}$.
- **Cross-period cell** (`catt_period != twfe_period`): Any non-zero weight indicates contamination: the TWFE coefficient $\hat{\mu}_{\ell''}$ also picks up treatment effects from period $\ell \neq \ell''$.
- **Verification**: the OVB identity (property iii) holds exactly, so $\hat{\mu}_{\ell''} = \sum_{(e,\ell)} \omega^{\ell''}_{e,\ell} \cdot \widehat{CATT}_{e,\ell}$ up to floating-point precision.

References

Sun, L. and Abraham, S. (2021). Estimating dynamic treatment effects in event studies with heterogeneous treatment effects. *Journal of Econometrics*, 225(2), 175–199.

See Also

`plot.sa_contamination_weights()`, `event_study()`, and `compute_contamination_weights()` for the deprecated predecessor.

Examples

```
## Not run:
# Estimate contamination weights
cw <- contamination_weights(
  data     = panel_data,
  time     = year,
  timing   = first_treat,
  unit     = id,
  fe       = ~ id + year,
  baseline = -1L
)
print(cw)
plot(cw)

## End(Not run)
```

---

did *Basic two-way fixed-effects DiD estimation*

Description

Estimates a classic difference-in-differences model of the form `outcome ~ D + covariates | fixed effects` on the package's internal fixed-effects OLS engine (numerically equivalent to `fixest::feols()`). The formula is the one used by `event_study()` and `att()`: swap the function to get effects by event time or aggregated staggered ATTs.

`did()` supersedes `run_did()` as of fixes 1.0.0.

Treatment is given in one of three ways, as in `event_study()`:

```
# D: 1 for treated units in post-treatment periods
did(y ~ D | id + year, data = df)

# each unit's first treated period (NA = never treated)
did(y ~ 1 | id + year, data = df, time = year, timing = g)

# a treated-group dummy and one treatment date
did(y ~ treated | id + year, data = df, time = year, timing = 2006)
```

With `timing`, `D = treatment * (time >= timing)` is built internally. For staggered-adoption settings see `att()`; for dynamic effects see `event_study()`.

Usage

```
did(
  formula,
  data,
  time = NULL,
  timing = NULL,
  unit = NULL,
  cluster = NULL,
  weights = NULL,
  conf_level = 0.95,
  vcov = "HC1",
  vcov_args = list(),
  outcome,
  treatment,
  fe = NULL,
  covariates = NULL
)
```

Arguments

`formula`	A model formula `outcome ~ treatment + covariates | fixed effects`. The left-hand side may be an expression (e.g. `log(y)`); without `|` the model has no fixed effects. The first right-hand-side term is D without `timing`, the treated-group dummy with a single `timing` date, and absent with a timing column (see **Description**). All variables must be columns of `data`.
`data`	A data.frame (panel format). `formula` and `data` may come in either order.
`time`	Unquoted time variable. Required with `timing`; with plain fixed effects such as `| id + year` it also lets `did()` report the unit (see `event_study()`, **Panel identifiers**).
`timing`	Treatment timing: an unquoted column giving each unit's first treatment period (`NA` = never treated), or a single numeric treatment date. Default `NULL`: the formula holds D.
`unit`	Unquoted unit identifier column (for metadata). Taken from the fixed effects when possible.

`cluster` Clustering specification: a one-sided formula (`~ id`), a single character column name, or a numeric vector of length `nrow(data)`. When `cluster` is specified and `vcov` is the default `"HC1"`, cluster-robust standard errors are used automatically.

`weights` Observation weights (formula or numeric vector).

`conf_level` Confidence level(s) for CIs. Scalar or vector (e.g., `c(0.90, 0.95)`). Default `0.95`.

`vcov` VCOV type string passed to `fixest::vcov()`. Default `"HC1"`. Ignored in favour of cluster-robust SE when `cluster` is supplied and `vcov` is left at its default `"HC1"`.

`vcov_args` Named list of additional arguments forwarded to `fixest::vcov()`.

`outcome`, `treatment`, `fe`, `covariates`
[Deprecated] The model arguments of the older interface: the unquoted outcome, the treatment column (D, or the treated-group dummy when `timing` is given), a one-sided fixed-effects formula (`NULL` with both `unit` and `time` gives `~ unit + time`), and a one-sided covariate formula. They still work in fixes 1.x, with a deprecation warning that shows the equivalent formula call; fixes 2.0.0 is planned to accept only `formula`. Name them when you use them: the second argument is now `data`.

Value

A `did_result` object (named list) with elements:

`estimates` Data frame with the treatment coefficient: `term`, `estimate`, `std.error`, `statistic`, `p.value`, and `conf_low_XX`/`conf_high_XX` for each `conf.level` entry.

`coeftable` Data frame with all kept coefficients (treatment and covariates): `term`, `estimate`, `std.error`, `statistic`, `p.value`.

`vcov` Full coefficient covariance matrix.

`fit_stats` Model-level statistics (nobs, R-squareds, sigma, logLik, AIC, BIC) used by `broom::glance()`.

`df.t` Degrees of freedom of the t inference.

Attributes: `call`, `formula_str`, `outcome`, `treatment`, `timing`, `fe`, `vcov_type`, `cluster_vars`, `conf.level`, `N`, `N_units`, `N_treated`, `unit`, `time` (and `timing_var` with a timing column). `N`, `N_units`, and `N_treated` describe the *estimation* sample (after rows with missing values are dropped), matching `broom::glance()`, not `nrow(data)`.

See Also

`event_study()` for dynamic effects, `att()` for staggered aggregation, and `run_did()` for the deprecated verb-style predecessor.

Examples

```
## Not run:
res <- did(y ~ D | id + year, data = df)
```

```
# Built from a treated-group dummy and the treatment date
res <- did(y ~ treated | id + year, data = df, time = year, timing = 2006)

# Covariates and cluster-robust SE
res <- did(y ~ D + x1 | id + year, data = df, cluster = ~ id)

print(res)
broom::tidy(res)
broom::glance(res)
# modelsummary::modelsummary(res)

## End(Not run)
```

`event_study` *Event study estimation for panel data*

Description

Estimates dynamic treatment effects by relative time (an event-study curve) on panel data. Supports the classic TWFE design (universal or staggered timing) plus modern staggered-robust estimators — Callaway-Sant'Anna, Sun-Abraham, Borusyak-Jaravel-Spiess, Wooldridge two-way Mundlak, and Deb et al. FLEX — selected via `estimator=`. Estimation runs on the package's internal fixed-effects OLS engine; results are returned as a tidy table with rich metadata.

Usage

```
event_study(
  formula,
  data,
  time,
  timing = NULL,
  unit = NULL,
  estimator = c("twfe", "cs", "sa", "bjs", "twm", "flex"),
  cluster = NULL,
  weights = NULL,
  baseline = -1L,
  lead_range = NULL,
  lag_range = NULL,
  conf_level = 0.95,
  vcov = "HC1",
  vcov_args = list(),
  control_group = c("nevertreated", "notyettreated"),
  anticipation = 0L,
  bootstrap = FALSE,
  boot_reps = 999L,
  boot_alpha = 0.05,
  boot_seed = NULL,
```

```
  group = NULL,
  trends = FALSE,
  interval = 1,
  time_transform = FALSE,
  rel_time = NULL,
  staggered = NULL,
  outcome,
  treatment = NULL,
  fe = NULL,
  covariates = NULL
)
```

Arguments

formula	A model formula `outcome ~ treatment + covariates | fixed effects`. The left-hand side may be an expression (e.g. `log(y)`); the part after `|` is optional. What the first right-hand-side term means depends on `timing`; see **Specifying treatment**. Fixed effects such as `id^year` require the optional {fixest} package. All variables must be columns of `data`.
data	A data.frame containing panel data. `formula` and `data` may come in either order, so `df |> event_study(y ~ D | id + year, time = year)` works.
time	Unquoted time variable (numeric or Date).
timing	Treatment timing; see **Specifying treatment**. Either an unquoted column giving each unit's first treatment period (`NA` = never treated) or a single treatment date (numeric or Date). Omit it when the formula holds a D column. A timing column must have the same value on every row of a unit; `"cs"`, `"sa"`, `"bjs"` and `"twm"` stop with an error naming the offending units otherwise. `NA` **means never treated:** this follows `did::att_gt()`, `fixest::sunab()`, and `didimputation`. If `NA` instead represents *missing* treatment timing for an otherwise treated unit, those observations are silently absorbed into the control group, which is almost certainly wrong. A D column avoids the coding question.
unit	Unquoted unit identifier. Taken from the fixed effects when possible; see **Panel identifiers**.
estimator	Estimation strategy: `"twfe"` (default) for the classic TWFE event study, `"cs"` (Callaway-Sant'Anna 2021), `"sa"` (Sun-Abraham 2021 interaction-weighted), `"bjs"` (Borusyak-Jaravel-Spiess 2024 imputation), `"twm"` (Wooldridge two-way Mundlak), or `"flex"` (Deb et al. 2024, repeated cross-sections).
cluster	Cluster specification (one-sided formula like `~ id + year`, a single character column name, or a vector of length `nrow(data)`).
weights	Observation weights (a name/one-sided formula or a numeric vector of length `nrow(data)`).
baseline	Integer baseline period (default `-1`); the reference period excluded from the estimation results.
lead_range, lag_range	Integers for pre/post windows. If `NULL`, determined automatically.
conf_level	Numeric vector of confidence levels (default `0.95`).

vcov — VCOV type: "HC1" (default), "iid", or "hetero" on the internal engine; other types are delegated to fixest::vcov() when the optional {fixest} package is installed.

vcov_args — List of additional arguments forwarded to fixest::vcov() (requires {fixest}).

control_group — For estimator = "cs": comparison group, either "nevertreated" (default) or "notyettreated".

anticipation — For estimator = "cs": number of anticipation periods before treatment (non-negative integer, default 0L).

bootstrap — Logical; if TRUE and estimator = "cs", compute simultaneous confidence bands via the multiplier bootstrap (Algorithm 1, Callaway and Sant'Anna 2021). Adds conf_low_sim and conf_high_sim columns to the result and stores the full (g,t)-level bootstrap object as attr(result, "bootstrap"). Default FALSE.

boot_reps — Integer number of bootstrap draws (default 999). Used only when bootstrap = TRUE and estimator = "cs".

boot_alpha — Significance level for the simultaneous band (default 0.05). Note: this is independent of conf_level, which governs the pointwise delta-method CIs.

boot_seed — Integer seed for the bootstrap RNG; NULL (default) does not set a seed. Pass an integer for reproducible results.

group — Unquoted group identifier for estimator = "flex" only. Identifies which treatment group (cohort) each observation belongs to in a repeated cross-section design (R_{ig} in Deb et al. 2024). Each group must map to exactly one value of timing (or NA for never-treated groups). With a D column, each group's start is the first period with D = 1. Not used by other estimators.

trends — Logical; for estimator = "twm" only. When TRUE, adds cohort-specific linear trend regressors $d_g \cdot t$ to the Procedure 5.1 regression (Wooldridge 2025, Section 8), allowing each cohort's counterfactual trend to deviate linearly from the common time trend. Requires at least 2 pre-treatment periods per cohort. Default FALSE.

interval — Numeric spacing of the time variable (default 1; ignored internally for Dates).

time_transform — Logical; if TRUE, creates consecutive integer time within unit. A timing column or value is then read as an index; a D column needs no adjustment.

rel_time — Optional unquoted column holding a **pre-built event time** (time relative to treatment), e.g. the Time_to_Treatment produced by paneltools/fect's get.cohort(). When supplied, event_study() uses this column verbatim as the event-study factor — i(rel_time, treatment, ref = baseline) — instead of computing relative time internally from time and timing; the first right-hand-side term is then the treated-group dummy. Never-treated / control rows should be NA. Requires estimator = "twfe"; cannot be combined with timing, staggered = TRUE or time_transform = TRUE. See **Details** for how this differs from the calendar-difference and time_transform conventions.

staggered — Logical or NULL (default). Only consulted by estimator = "twfe": TRUE means a staggered design, FALSE a single universal treatment date. With the default NULL the design follows from how treatment is given (a timing column is staggered, a single date is universal, a D column is universal when every unit starts in the same period). Pass an explicit value to override the inference.

outcome, treatment, fe, covariates

[Deprecated] The model arguments of the older interface: the unquoted outcome, the treated-group dummy (used only by `estimator = "twfe"`), a one-sided fixed-effects formula, and a one-sided covariate formula. They still work in fixes 1.x, with a deprecation warning that shows the equivalent formula call; fixes 2.0.0 is planned to accept only `formula`. Name them when you use them: the second argument is now `data`.

Details

The model is a formula, `outcome ~ treatment + covariates | fixed effects`, as in `fixest::feols()`; the same formula works in `did()` and `att()`.

`event_study()` supersedes `run_es()` as of fixes 1.0.0.

How relative (event) time is defined. `event_study()` can build the event-time axis in three different ways, and in sparse or unbalanced panels (e.g. monthly surveys observed only a few times a year) they do **not** coincide:

Method	Definition	How to invoke
Calendar difference (default)	`(time - timing) / interval`	a timing column, a scalar `timing`, or a D column
Within-unit rank	`dense_rank(time)` within each unit	`time_transform = TRUE`
Pre-built event time	supplied column, used verbatim	`rel_time = <col>`

In a balanced panel all three agree. When observation spacing is irregular they diverge: for a unit observed at periods `{1, 5, 9, 20}` treated at 20, the calendar difference puts period 9 at `9 - 20 = -11`, whereas a global observed-period grid (as used by `paneltools/fect`'s `get.cohort()`) places it two observed waves before treatment (`-2`). To reproduce an analysis built on such a grid, pass that grid's event-time column via `rel_time` so the design matches `fixest::feols(y ~ i(rel_time, treatment, ref) | fe)` exactly. `att()` uses the same calendar-difference convention as the default here. With a D column, a unit's start is the first period in which D = 1 is observed.

Missing values and unbalanced panels. With `estimator = "cs"`, rows with a missing or non-finite outcome, time, or unit are dropped, and the panel is then balanced by dropping every unit not observed in all periods, as `did::att_gt()` does by default (`allow_unbalanced_panel = FALSE`). Both steps warn, and `N`, `N_units`, `N_treated` and `N_nevertreated` describe the balanced sample. The regression-based estimators drop rows with missing values from the fit as `fixest::feols()` does; with `estimator = "sa"` or `"twm"` the cohort shares used to aggregate cohort-specific effects count only units that remain in the estimation sample (Sun and Abraham 2021, Section 4.1).

Value

A `data.frame` of class `"es_result"` with columns:

- `term`, `estimate`, `std.error`, `statistic`, `p.value`
- `conf_low_XX`, `conf_high_XX` (for each requested `conf.level`)
- `relative_time` (integer; 0 = event), `is_baseline` (logical; marks the reference period)

Attributes include: `lead_range`, `lag_range`, `baseline`, `interval`, `call`, `model_formula`, `conf.level`, `N`, `N_units`, `N_treated`, `N_nevertreated`, `fe`, `vcov_type`, `cluster_vars`, `staggered`, `sunab_used`, `unit_var`, `time_var`.

Specifying treatment

Describe treatment in whichever way matches your data:

Your data has	Write
D (1 for treated units in post-treatment periods)	`y ~ D + x | id + year, time = year`
each unit's first treated period (`NA` = never treated)	`y ~ x | id + year, time = year, timing = g`
a treated-group dummy and one treatment date	`y ~ treated + x | id + year, time = year, timing = 2005`

- **D column** (no `timing`): the first right-hand-side term is D. A unit's treatment starts in the first period with `D = 1` and must stay on from then on (staggered adoption). Rows with a missing D are dropped. A single start date across units gives the classic universal design, several give the staggered one. With `time_transform = TRUE` the start is read on the time index, so no index arithmetic is needed.
- **Timing column** (`timing = g`): `g` defines treatment, so the right-hand side holds only covariates (write `y ~ 1 | id + year` when there are none). D or the treated-group dummy must not appear in the formula.
- **Treated-group dummy and a date** (`timing = 2005`): the first term is the treated-group dummy (constant within units), and treatment starts at `timing` for every treated unit.

A pre-built event-time column goes in `rel_time`, with the treated-group dummy as the first term: `y ~ treated | id + year, time = year, rel_time = ttt`.

Panel identifiers

`time` is required. `unit` is taken from the fixed effects when they are plain columns made of `time` and exactly one other column (`| id + year` with `time = year`, in either order); a message then reports the unit and time used. Otherwise pass `unit`: it is needed for a D column and for `estimator = "cs"`, `"sa"`, `"bjs"` and `"twm"`. `print()` shows the identifiers of every result.

Fixed effects and covariates by estimator

`"twfe"`, `"sa"` and `"twm"` use the fixed effects as written. `"cs"`, `"bjs"` and `"flex"` absorb the unit (group) and time effects themselves: those fixed effects are accepted silently, any others are ignored with a warning. Covariates are supported by `"twfe"`, `"twm"` and `"flex"`, and may be expressions such as `log(x)` or `factor(region)`; `"cs"`, `"sa"` and `"bjs"` stop with an error when the formula contains covariates.

Key Features

- One formula for the model (outcome, treatment, covariates, fixed effects), shared with `did()` and `att()`.
- Six estimators behind one interface: `"twfe"`, `"cs"`, `"sa"`, `"bjs"`, `"twm"`, `"flex"`.
- Flexible clustering, weights, and VCOV choices (e.g., `vcov = "HC1" | "iid" | "hetero" ...`).
- Automatic lead/lag window detection and customizable baseline period.
- Returns an `"es_result"` object compatible with `plot()`, `print()`, and `autoplot()`.

See Also

plot.es_result() for visualization, att() for aggregated ATT estimation, and run_es() for the deprecated verb-style predecessor.

Examples

```
## Not run:
# D column: 1 for treated units in post-treatment periods
res <- event_study(y ~ D | id + year, data = df, time = year)
plot(res)

# Timing column (NA = never treated), Callaway-Sant'Anna
res <- event_study(y ~ 1 | id + year, data = df, time = year, timing = g,
                   estimator = "cs")

# Treated-group dummy and a single treatment date, clustered by unit
res <- event_study(y ~ treated + x1 | id + year, data = df, time = year,
                   timing = 2005, cluster = ~ id)

## End(Not run)
```

---

glance.att_result *Glance at an* att_result *object*

Description

Returns a one-row summary of an att() result for the goodness-of-fit block of a modelsummary::modelsummary() table.

Usage

```
glance.att_result(x, ...)
```

Arguments

x	An att_result object returned by att().
...	Unused; for S3 generic compatibility.

Value

A one-row data frame with nobs, n.units, n.treated, estimator, and aggregation.

Examples

```
## Not run:
res <- att(y ~ D | id + year, data = df, time = year)
broom::glance(res)

## End(Not run)
```

`glance.did_result` *Glance at a* `did_result` *object*

Description

Returns a single-row summary of model-level statistics from a `did()` result: `r.squared`, `adj.r.squared`, `within.r.squared`, `sigma`, `nobs`, `AIC`, `BIC`, and `logLik`, in the column layout of `broom::glance()` on a `fixest` model.

Usage

```
glance.did_result(x, ...)
```

Arguments

`x`	A `did_result` object returned by `did()`.
`...`	Unused; kept for compatibility with the `broom::glance()` generic.

Value

A one-row tibble of model-level statistics.

Examples

```
## Not run:
res <- did(y ~ D | id + year, data = df)
broom::glance(res)

## End(Not run)
```

glance.es_result *Glance at an* es_result *object*

Description

Returns a one-row summary of an `event_study()` result: sample sizes, estimator, and VCOV type.

Usage

```
glance.es_result(x, ...)
```

Arguments

x	An es_result object.
...	Unused; kept for compatibility with the `broom::glance()` generic.

Value

A one-row tibble with columns `nobs`, `n_units`, `n_treated`, `n_nevertreated`, `estimator`, and `vcov_type`.

Examples

```
## Not run:
res <- event_study(y ~ 1 | id + year, data = df, time = year, timing = g,
                   estimator = "cs")
broom::glance(res)

## End(Not run)
```

honest_sensitivity *Honest sensitivity analysis for parallel-trends violations*

Description

Robust inference and sensitivity analysis for event-study / DiD designs following Rambachan and Roth (2023). Instead of assuming parallel trends holds exactly, it asks how large a violation of parallel trends would have to be before the causal conclusion changes, returning confidence sets for a post-treatment effect under a sequence of restrictions on the possible difference in trends.

Two restriction families are supported:

- `"relative_magnitude"` ($\Delta^{RM}(\bar{M})$): post-treatment violations are at most $\bar{M}$ times the largest pre-treatment violation (Rambachan & Roth 2023, Section 2.4.1).
- `"smoothness"` ($\Delta^{SD}(M)$): the difference in trends deviates from linearity by at most M per period (Section 2.4.3).

Inference uses the Andrews-Roth-Pakes (ARP) conditional moment-inequality test (Section 3.2.1), which is uniformly valid and recommended for general restriction sets.

Usage

```
honest_sensitivity(
  object = NULL,
  type = c("relative_magnitude", "smoothness"),
  m_grid = NULL,
  l_vec = NULL,
  alpha = 0.05,
  grid_points = 1000L,
  betahat = NULL,
  sigma = NULL,
  n_pre_periods = NULL,
  n_post_periods = NULL,
  Mvec = lifecycle::deprecated(),
  gridPoints = lifecycle::deprecated(),
  numPrePeriods = lifecycle::deprecated(),
  numPostPeriods = lifecycle::deprecated()
)
```

Arguments

`object`	An `es_result` from `event_study()` with `estimator = "twfe"` (which carries the event-study coefficient covariance). Alternatively, supply `betahat` and `sigma` directly (see below) for any estimator.
`type`	Restriction family: `"relative_magnitude"` (default) or `"smoothness"`.
`m_grid`	Numeric vector of restriction parameters. For `"relative_magnitude"` these are $\bar{M}$ values (default `seq(0, 2, by = 0.5)`); for `"smoothness"` these are M values (default a data-driven sequence from 0 to the largest pre-period SD).
`l_vec`	Numeric weight vector over post-treatment periods defining the target $\theta = l'\tau_{post}$. Defaults to the first post-treatment period.
`alpha`	Significance level (default `0.05` for 95% confidence sets).
`grid_points`	Number of grid points for test inversion (default `1000`).
`betahat`	Optional event-study coefficient vector (pre then post, excluding the reference period), ordered by relative time. Required when `object` does not carry an `es_vcov` attribute.
`sigma`	Optional covariance matrix of `betahat`.
`n_pre_periods`, `n_post_periods`	Optional integer counts; inferred from `object` or from `betahat`/`l_vec` when omitted.
`Mvec`	**[Deprecated]** Renamed to `m_grid`.
`gridPoints`	**[Deprecated]** Renamed to `grid_points`.
`numPrePeriods`, `numPostPeriods`	**[Deprecated]** Renamed to `n_pre_periods` / `n_post_periods`.

Value

A `data.frame` of class `"honest_result"` with one row per restriction value plus the original (parallel-trends) confidence interval, with columns `M`, `lb`, `ub`, `method`, and `type`. The breakdown value (largest restriction at which the robust CI still excludes 0) is stored in `attr(., "breakdown")`.

References

Rambachan, A. and Roth, J. (2023). A More Credible Approach to Parallel Trends. *Review of Economic Studies*, 90(5), 2555-2591.

See Also

`event_study()`, `plot.honest_result()`

`plot.att_gt_result` *Plot the ATT(g,t) matrix from a Callaway-Sant'Anna event study*

Description

Visualises the full cohort-by-period ATT(g,t) matrix stored in the `att_gt` attribute of an `es_result` object produced by `run_es(estimator = "cs")`. Two display styles are available:

- `"heatmap"`: a tile plot with calendar time t on the x-axis and cohort g on the y-axis, colour-filled by the point estimate. Cells whose pointwise confidence interval excludes zero are marked with a filled dot; cells that are simultaneously significant (when bootstrap data are available) are additionally marked with an open diamond.
- `"facet"`: one panel per cohort showing ATT(g,t) over calendar time t with a pointwise confidence ribbon, mirroring the style of `plot_es()`. A lighter simultaneous CI ribbon is overlaid when bootstrap data are available.

Both types draw a vertical dashed line at $t = g$ (treatment onset) for each cohort.

Usage

```
## S3 method for class 'att_gt_result'
plot(
  x,
  type = c("heatmap", "facet"),
  ci_level = 0.95,
  zero_line = TRUE,
  theme = c("bw", "minimal", "classic"),
  color = NULL,
  fill = NULL,
  alpha = NULL,
  ...
)

## S3 method for class 'att_gt_result'
autoplot(object, ...)
```

Arguments

x	An es_result object returned by run_es(estimator = "cs"), or an att_gt_result data frame produced by extracting attr(x, "att_gt") and giving it class "att_gt_result".
type	"heatmap" (default) or "facet".
ci_level	Confidence level for pointwise intervals (default 0.95).
zero_line	Logical; draw a horizontal reference line at zero in the "facet" display (default TRUE).
theme	One of "bw" (default), "minimal", or "classic".
color	Line and point colour used in the "facet" display (defaults to the package's steel blue, matching plot.es_result()).
fill	Ribbon fill colour in the "facet" display (defaults to the package's lighter ribbon tint).
alpha	Ribbon transparency in the "facet" display (default 0.35).
...	Passed on to the plotting engine.
object	An att_gt_result object; see att_gt().

Value

A ggplot2::ggplot() object.

Bootstrap annotation

When attr(x, "bootstrap") is present (i.e., run_es() was called with bootstrap = TRUE), both plot types add simultaneous inference overlays sourced from the (g,t)-level bootstrap object.

See Also

plot_es(), run_es()

Examples

```
## Not run:
cs_result <- event_study(y ~ 1 | id + year, data = mydata, time = year,
                         timing = g, estimator = "cs")
plot(att_gt(cs_result))
plot(att_gt(cs_result), type = "facet")

## End(Not run)
```

`plot.att_result` *Plot an ATT aggregation result*

Description

Base `plot()` method for `att_result` objects returned by `att()` (or the deprecated `calc_att()`). Draws point estimates with confidence-interval bars: a single point for `aggregation = "simple"`, one point per cohort for `"by_cohort"`, and one per calendar period for `"by_time"`.

Usage

```
## S3 method for class 'att_result'
plot(
  x,
  ci_level = 0.95,
  color = NULL,
  zero_line = TRUE,
  theme_style = c("bw", "minimal", "classic"),
  ...
)

## S3 method for class 'att_result'
autoplot(object, ...)
```

Arguments

`x`	An `att_result` object.
`ci_level`	Confidence level to display (default `0.95`; must be one of the levels requested in the originating call).
`color`	Point and interval colour (default black).
`zero_line`	Logical; draw a dashed reference line at zero (default `TRUE`).
`theme_style`	One of `"bw"` (default), `"minimal"`, or `"classic"`.
`...`	Unused.
`object`	An `att_result` object.

Value

A `ggplot` object.

See Also

`att()`

Examples

```
## Not run:
res <- att(y ~ 1 | id + year, data = df, time = year, timing = g,
           aggregation = "by_cohort")
plot(res)

## End(Not run)
```

`plot.es_result` *Plot an event-study result*

Description

Base `plot()` method for `es_result` objects returned by `event_study()` (or the deprecated `run_es()`). Draws the event-study curve with pointwise confidence bands as a static ggplot, or — with `interactive = TRUE` — an interactive plotly chart with hover tooltips (requires the suggested {plotly} package).

Usage

```
## S3 method for class 'es_result'
plot(
  x,
  ci_level = 0.95,
  type = c("errorbar", "ribbon"),
  time_axis = c("relative", "calendar"),
  interactive = FALSE,
  show_simultaneous = FALSE,
  ...
)
```

Arguments

`x`	An `es_result` object.
`ci_level`	Confidence level to display (default `0.95`). Must be one of the levels the result was estimated at (`conf_level` in `event_study()`).
`type`	`"errorbar"` (default) or `"ribbon"`. Static plots only.
`time_axis`	`"relative"` (default) for event time, or `"calendar"` for the original time values. See Details.
`interactive`	Logical; if `TRUE`, return an interactive plotly chart instead of a ggplot. Default `FALSE`.
`show_simultaneous`	Logical; overlay the simultaneous bootstrap CI band (requires `bootstrap = TRUE` in the originating `event_study()` call). Default `FALSE`.

`...` Further styling arguments: for static plots `vline_val`, `hline_val`, `vline_color`, `hline_color`, `linewidth`, `pointsize`, `alpha`, `barwidth`, `color`, `fill`, `errorbar_color`, `theme_style` (`"bw"`, `"minimal"`, or `"classic"`); for interactive plots `markersize`, `show_ribbon`, `height`, `width`, and the shared color arguments. `vline_val` defaults to relative time -1, the last period before treatment, on whichever axis is in use. `errorbar_color` colours the bars alone and follows `color` unless set.

Details

The default display is black error bars. `type = "ribbon"` switches to a line with a shaded band and, unless `color`/`fill` say otherwise, to the package's muted steel-blue palette. In either display `color` sets the points (and the line), while `errorbar_color` can recolour the bars on their own.

`time_axis` controls what the horizontal axis shows. The default `"relative"` plots event time (periods since treatment) and labels the axis "Relative Time to Treatment". `"calendar"` instead plots the original time values (e.g. `2010`, or a `Date`), labels the axis with the `time` column's name, and moves the dashed vertical reference line to the treatment period. It is available whenever every treated unit adopts in the same period — universal timing, or a staggered design that happens to have a single cohort — and errors otherwise, because with several cohorts one relative period corresponds to several calendar periods.

Value

A `ggplot` object, or a `plotly` object when `interactive = TRUE`.

See Also

`event_study()`, `autoplot.es_result()`

Examples

```
## Not run:
res <- event_study(y ~ 1 | id + year, data = df, time = year, timing = g,
                   estimator = "cs")
plot(res)
plot(res, type = "ribbon", ci_level = 0.9)
plot(res, color = "#1B4965", errorbar_color = "grey50")
plot(res, interactive = TRUE)

# Universal timing: label the axis with calendar years instead
uni <- event_study(y ~ d | id + year, data = df, time = year,
                   timing = 2010)
plot(uni, time_axis = "calendar")

## End(Not run)
```

plot.honest_result *Plot a honest sensitivity analysis*

Description

Visualises the output of `honest_sensitivity()`: robust confidence intervals for the target effect under progressively weaker parallel-trends restrictions (increasing M or $\bar{M}$), alongside the original confidence interval that assumes parallel trends holds exactly. This is the "top-down" sensitivity plot of Rambachan and Roth (2023).

Usage

```
## S3 method for class 'honest_result'
plot(x, ...)

## S3 method for class 'honest_result'
autoplot(object, ...)
```

Arguments

`x`	A `honest_result` object from `honest_sensitivity()`.
`...`	Unused.
`object`	A `honest_result` object.

Value

A `ggplot` object.

See Also

`honest_sensitivity()`

plot.sa_contamination_weights *Plot contamination weights as a tile heatmap*

Description

Base `plot()` method for the contamination weights returned by `contamination_weights()`: a ggplot2 tile heatmap.

Each cell at position (`twfe_period`, `catt_label`) shows the weight $\omega_{e,\ell}^{\ell''}$: how much of $CATT_{e,\ell}$ leaks into the TWFE coefficient $\hat{\mu}_{\ell''}$.

- **Diagonal cells** (`catt_period == twfe_period`): own-period weights (sum across cohorts should be ≈ 1).
- **Off-diagonal cells**: cross-period contamination (ideally close to zero under treatment effect homogeneity).

Usage

```
## S3 method for class 'sa_contamination_weights'
plot(
  x,
  limit_abs = NULL,
  midpoint = 0,
  low = "#2166AC",
  mid = "white",
  high = "#B2182B",
  theme = c("bw", "minimal", "classic"),
  show_values = FALSE,
  value_digits = 2L,
  ...
)
```

Arguments

x	An sa_contamination_weights object from contamination_weights().
limit_abs	Numeric; symmetric colour scale limit [-limit, limit]. Defaults to the maximum absolute weight (rounded up to one decimal).
midpoint	Numeric; midpoint of the diverging colour scale (default 0).
low	Colour for negative weights (default "#2166AC").
mid	Colour for zero weight (default "white").
high	Colour for positive weights (default "#B2182B").
theme	Character; "bw" (default), "minimal", or "classic".
show_values	Logical; overlay weight values in each tile (default FALSE).
value_digits	Integer; decimal digits when show_values = TRUE (default 2L).
...	Unused.

Value

A ggplot2::ggplot() object.

See Also

contamination_weights()

`tidy.att_result` *Tidy an* `att_result` *object*

Description

Returns a tidy data frame of the aggregated ATT estimates from a `att()` result, in the column format expected by `modelsummary::modelsummary()` (and therefore renderable with `tinytable::tt()`). This makes it easy to put overall, by-cohort, or by-time ATTs into a publication table, and to stack several estimators (e.g. `did()` TWFE, CS, and BJS) side by side in one table.

The `term` column is derived from the aggregation type so that rows align across models:

- `aggregation = "simple"` -> `"ATT"`
- `aggregation = "by_cohort"` -> `"Cohort <g>"`
- `aggregation = "by_time"` -> `"Time <t>"`

Usage

```
## S3 method for class 'att_result'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An `att_result` object returned by `att()`.
`conf.int`	Logical; add `conf.low`/`conf.high` columns? Default `FALSE`.
`conf.level`	Confidence level for `conf.int` (normal approximation). Default `0.95`.
`...`	Unused; for S3 generic compatibility.

Value

A data frame with columns `term`, `estimate`, `std.error`, `statistic`, `p.value` (and optionally `conf.low`, `conf.high`).

Examples

```
## Not run:
res <- att(y ~ D | id + year, data = df, time = year,
           estimator = "cs", aggregation = "by_cohort")
broom::tidy(res)

# Compare estimators in one table:
twfe <- did(y ~ D | id + year, data = df)
cs   <- att(y ~ D | id + year, data = df, time = year, estimator = "cs")
bjs  <- att(y ~ D | id + year, data = df, time = year, estimator = "bjs")
modelsummary::modelsummary(list(TWFE = twfe, CS = cs, BJS = bjs))

## End(Not run)
```

`tidy.did_result` *Tidy a* `did_result` *object*

Description

Returns a tidy data frame of model coefficients from a `did()` result: all regressors (treatment and covariates), in the format expected by `modelsummary::modelsummary()`. The table is read from the fit stored at estimation time; the VCOV type chosen in `did()` (`vcov` / `cluster`) is already reflected in the standard errors.

Usage

```
## S3 method for class 'did_result'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	A `did_result` object returned by `did()`.
`conf.int`	Logical; add `conf.low`/`conf.high` columns? Default `FALSE`.
`conf.level`	Confidence level for `conf.int`. Default `0.95`. Intervals use Student-t quantiles with the fit's degrees of freedom, matching `fixest`/`broom` conventions.
`...`	Unused; kept for compatibility with the `broom::tidy()` generic.

Value

A tibble with columns `term`, `estimate`, `std.error`, `statistic`, `p.value` (and optionally `conf.low`, `conf.high`).

Examples

```
## Not run:
res <- did(y ~ D | id + year, data = df)
broom::tidy(res)
broom::tidy(res, conf.int = TRUE)

## End(Not run)
```

`tidy.es_result` *Tidy an* `es_result` *object*

Description

Returns the event-study coefficient table of an `event_study()` result as a plain tibble — the format expected by `modelsummary::modelsummary()` and other broom-aware tooling. The baseline row (zero by construction) is included; drop it with `x[!x$is_baseline, ]` beforehand if undesired.

Usage

```
## S3 method for class 'es_result'
tidy(x, ...)
```

Arguments

`x`	An `es_result` object.
`...`	Unused; kept for compatibility with the `broom::tidy()` generic.

Value

A tibble with columns `term`, `estimate`, `std.error`, `statistic`, `p.value`, `relative_time`, `is_baseline`, and the `conf_low_XX`/`conf_high_XX` pairs of the originating call.

Examples

```
## Not run:
res <- event_study(y ~ 1 | id + year, data = df, time = year, timing = g,
                   estimator = "cs")
broom::tidy(res)

## End(Not run)
```

Index

att, 2
att(), *5*, *8–10*, *14–16*, *22*, *27*
att_gt, 5
att_gt(), *21*
autoplot.att_gt_result
        *(*plot.att_gt_result*)*, 20
autoplot.att_result *(*plot.att_result*)*,
        22
autoplot.es_result, 6
autoplot.es_result(), *24*
autoplot.honest_result
        *(*plot.honest_result*)*, 25

broom::glance(), *10*, *17, 18*
broom::tidy(), *28, 29*

calc_att(), *4, 5*, *22*
compute_contamination_weights(), *8*
contamination_weights, 7
contamination_weights(), *25, 26*

did, 8
did(), *4*, *14, 15*, *17*, *27, 28*
did::aggte(), *5*
did::att_gt(), *4*, *14*

event_study, 11
event_study(), *3–10*, *18–20*, *23, 24*, *29*

ggplot2::ggplot(), *21*, *26*
glance.att_result, 16
glance.did_result, 17
glance.es_result, 18

honest_sensitivity, 18
honest_sensitivity(), *25*

plot(), *5*
plot.att_gt_result, 20
plot.att_gt_result(), *5*
plot.att_result, 22
plot.es_result, 23
plot.es_result(), *6*, *16*, *21*
plot.honest_result, 25
plot.honest_result(), *20*
plot.sa_contamination_weights, 25
plot.sa_contamination_weights(), *8*
plot_es(), *20, 21*

run_did(), *8*, *10*
run_es(), *14*, *16*, *21*, *23*

tidy.att_result, 27
tidy.did_result, 28
tidy.es_result, 29