# Package: didgpu (via r-universe)

October 31, 2026

**Title** GPU-Accelerated, Checkpointed Difference-in-Differences Estimator

**Version** 0.1.2

**Description** A clean-room reimplementation of the heterogeneity-robust dynamic difference-in-differences estimator of de Chaisemartin and D'Haultfoeuille (2024) <doi:10.1162/rest_a_01414>, designed for long-running econometric work: per-cell checkpointing to disk, resumable runs after crash or out-of-memory, parallel cluster bootstrap, configurable backends (pure R, optional 'Rcpp' and 'CUDA'), and bit-for-bit numerical equivalence with the reference 'DIDmultiplegtDYN' package across binary, multivalued, and continuous treatments.

**License** MIT + file LICENSE

**URL** https://github.com/JoshuaAmmons/didgpu

**BugReports** https://github.com/JoshuaAmmons/didgpu/issues

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Depends** R (>= 4.1)

**Imports** data.table, jsonlite, MASS, Rcpp, sandwich, stats, utils, tools

**Suggests** broom, contdid, cowplot, did, DIDmultiplegt, DIDmultiplegtDYN, DRDID, eventstudyr, fect, fixest, ggplot2, HonestDiD, openxlsx, polars, quadprog, splines2, testthat (>= 3.0.0)

**LinkingTo** Rcpp, RcppEigen

**SystemRequirements** GNU make; optional NVIDIA CUDA Toolkit (>= 12.0) with cuBLAS and cuSOLVER for the GPU backend

**Config/testthat/edition** 3

**Config/pak/sysreqs** nvidia-cuda-dev make

**Repository** https://joshuaammons.r-universe.dev

**Date/Publication** 2026-10-30 22:06:23 UTC

**RemoteUrl** https://github.com/JoshuaAmmons/didgpu

**RemoteRef** HEAD

**RemoteSha** adf4955f4510a9b420d854c7a0cdc452b0978543

# Contents

didgpu-package . . . 3
coef.didgpu_result . . . 4
confint.didgpu_result . . . 4
didgpu . . . 5
didgpu_aggregate_cells . . . 10
didgpu_backend_info . . . 11
didgpu_bacon . . . 11
didgpu_bootstrap_more . . . 12
didgpu_by . . . 13
didgpu_by_path . . . 15
didgpu_compare . . . 16
didgpu_compute_paths . . . 17
didgpu_cpu_hello . . . 18
didgpu_cs . . . 19
didgpu_cs_aggregate . . . 21
didgpu_cs_continuous . . . 22
didgpu_did_continuous . . . 23
didgpu_did_static . . . 25
didgpu_equivalence . . . 26
didgpu_estimate_runtime . . . 27
didgpu_event_study_data . . . 29
didgpu_fect . . . 30
didgpu_fect_equivalence . . . 32
didgpu_fect_placebo . . . 33
didgpu_freyaldenhoven . . . 34
didgpu_glance . . . 35
didgpu_has_cuda_support . . . 36
didgpu_honest_did . . . 36
didgpu_init_checkpoint . . . 38
didgpu_joint_placebo . . . 38
didgpu_lb_frac_affected . . . 39
didgpu_load_checkpoint . . . 41
didgpu_loo . . . 42
didgpu_resume . . . 43
didgpu_run_saxpy . . . 44
didgpu_simulate_panel . . . 44
didgpu_simulate_panel_bidir . . . 46

didgpu_summarize_panel . . . 47
didgpu_test_sharp_null . . . 47
didgpu_tidy . . . 49
didgpu_twfe . . . 50
glance.didgpu_result . . . 51
plot.didgpu_loo_result . . . 51
plot.didgpu_result . . . 52
print.didgpu_bacon . . . 53
print.didgpu_by_result . . . 53
print.didgpu_cs_continuous_result . . . 54
print.didgpu_cs_result . . . 54
print.didgpu_did_continuous_result . . . 55
print.didgpu_did_static_result . . . 55
print.didgpu_equivalence . . . 56
print.didgpu_fect_placebo . . . 56
print.didgpu_freyaldenhoven_result . . . 57
print.didgpu_honest_did_result . . . 57
print.didgpu_joint_placebo . . . 58
print.didgpu_loo_result . . . 58
print.didgpu_result . . . 59
print.didgpu_twfe_result . . . 59
summary.didgpu_result . . . 60
tidy.didgpu_result . . . 60
vcov.didgpu_result . . . 61

**Index** **62**

---

`didgpu-package` *didgpu: GPU-Accelerated, Checkpointed Difference-in-Differences*

---

## Description

A clean-room reimplementation of the de Chaisemartin and D'Haultfoeuille dynamic difference-in-differences estimator, designed for long-running econometric work. Three architectural commitments distinguish it from the reference 'DIDmultiplegtDYN':

## Details

1. **Per-cell checkpointing.** Each bootstrap iteration is saved to disk as it completes. A crash at hour N preserves N-worth of work.
2. **Resumable runs.** Re-invoking with the same `checkpoint_dir` skips completed cells. Identical seed plus identical config produces identical aggregated output.
3. **Pluggable backend.** Pure R for correctness, Rcpp+Eigen for CPU throughput, optional CUDA (cuBLAS + cuSOLVER) for the bootstrap inner loop. Same numerics across backends.

## Status

This package is in early development. Currently supported: binary, non-absorbing treatment; the core estimator in pure R; checkpointing layer; CPU backend. Not yet supported: multivalued/continuous treatment, controls, weights, the by_path framework, the GPU backend (scaffolded only).

## Author(s)

**Maintainer**: Joshua Ammons <jdammons89@gmail.com> [copyright holder]

---

`coef.didgpu_result` *Coefficient extractor for didgpu_result*

---

## Description

Returns a named numeric vector of point estimates. Names are "Effect_1", ..., "Effect_n_effects" optionally followed by "Placebo_1", ..., "Placebo_n_placebos" and (if available) "ATE".

## Usage

```
## S3 method for class 'didgpu_result'
coef(object, which = "all", ...)
```

## Arguments

| | |
|---|---|
| `object` | A `didgpu_result` object. |
| `which` | One of `"effects"`, `"placebos"`, `"ate"`, or `"all"` (default). Controls which subset is returned. |
| `...` | Unused. |

## Value

Named numeric vector.

---

`confint.didgpu_result` *Confidence intervals for didgpu_result*

---

## Description

Returns the percentile-based CI matrix already computed during aggregation (from the bootstrap distribution). Re-computing at a different level requires a fresh run because the cell-level percentiles are not stored on disk.

### Usage

```
## S3 method for class 'didgpu_result'
confint(object, parm = NULL, level = NULL, ...)
```

### Arguments

| | |
|---|---|
| object | A didgpu_result object. |
| parm | Optional character vector of coefficient names to subset. |
| level | Confidence level. Must match the level used at fit time; otherwise a warning is issued and the stored CI is returned anyway. |
| ... | Unused. |

### Value

A 2-column matrix with the lower and upper bounds.

---

## didgpu *Run a checkpointed, GPU-capable dynamic DiD estimation*

---

### Description

Reimplementation entry point. See vignette("reference_internals") for the design spec and didgpu_backend_info() for what's available on this machine.

### Usage

```
didgpu(
  df,
  outcome,
  group,
  time,
  treatment,
  effects = 1L,
  placebo = 0L,
  cluster = NULL,
  controls = NULL,
  weight = NULL,
  continuous = NULL,
  trends_nonparam = NULL,
  trends_lin = FALSE,
  predict_het = NULL,
  only_never_switchers = FALSE,
  same_switchers = FALSE,
  same_switchers_pl = FALSE,
  dont_drop_larger_lower = FALSE,
  switchers = "",
```

```
  normalized = FALSE,
  bootstrap_reps = 0L,
  ci_level = 95,
  seed = 1L,
  checkpoint_dir = NULL,
  resume = TRUE,
  backend = "auto",
  n_workers = 1L,
  verbose = FALSE,
  on_iter = NULL,
  bootstrap = NULL,
  graph_off = FALSE,
  ggplot_args = NULL,
  less_conservative_se = FALSE,
  more_granular_demeaning = FALSE,
  drop_if_d_miss_before_first_switch = FALSE,
  effects_equal = FALSE,
  predict_het_hc2bm = FALSE,
  normalized_weights = FALSE,
  save_results = NULL,
  save_sample = FALSE,
  design = NULL,
  date_first_switch = NULL,
  by = NULL,
  by_path = NULL,
  reset = 0,
  avg_time_periods = FALSE
)
```

## Arguments

| | |
|---|---|
| `df` | A data.frame (or data.table) panel. Must contain the columns named by `outcome`, `group`, `time`, `treatment`. |
| `outcome`, `group`, `time`, `treatment` | Character. Column names. |
| `effects` | Integer. Number of post-treatment event-times to estimate (e = 1..effects). Must be >= 1. |
| `placebo` | Integer. Number of pre-treatment placebos. 0 to skip. |
| `cluster` | Character or NULL. Column name for cluster bootstrap; default NULL clusters by `group`. |
| `controls` | Character vector or NULL. Names of covariate columns to control for. The point estimate adjusts diff_y by the FWL projection on these covariates' first differences, restricted to never-switcher rows within each baseline-treatment cohort. |
| `weight` | Character or NULL. Name of a per-row weight column. If NULL, every observation gets weight 1 (the binary case). |
| `continuous` | Integer or NULL. If set (typically `1` for linear continuous treatment), collapses cohorts to a single one (`d_sq = 0` for all units) and adds polynomial features |

(baseline_D)^1, ..., (baseline_D)^continuous as auto-controls in the FWL projection. Use when the treatment column is genuinely continuous (e.g., dosage, tax rate) so that no two units share an identical baseline value.

trends_nonparam
: Character or NULL. Name of a categorical column to extend the cohort grouping by. With this set, cohort averages are computed per (time, d_sq, trends_nonparam) instead of per (time, d_sq), allowing time trends to vary by the extra grouping (e.g., industry).

trends_lin
: Logical. If TRUE, allow group-specific linear trends: the estimator runs on the first-difference of the outcome (and any user controls), then for each k = 1..effects returns the cumulative sum of per-event-time first-difference DIDs (mapping back to a level effect). Forces same_switchers = TRUE and suppresses ATE. Reference: Section 1.3 of the Web Appendix of de Chaisemartin and D'Haultfoeuille (2024).

predict_het
: Optional list of length 2: list(covariates, event_times), where covariates is a character vector of time-invariant column names and event_times is an integer vector of which event-times to do the heterogeneity regression for (use -1 for all). Regresses the per-group ATE contribution at each event-time on the covariates (with cohort interaction dummies and HC1 robust SEs), to characterise effect heterogeneity. Returns the regression in result$results$predict_het. Not compatible with normalized = TRUE (the regression target is the unnormalised DID).

only_never_switchers
: Logical. If TRUE, restrict controls to strictly never-switched units (drop pre-switch rows of units that eventually switch).

same_switchers
: Logical. If TRUE, restrict switcher rows to units that have valid controls at every event-time q in 1..effects. Mirrors the reference's same_switchers option.

same_switchers_pl
: Logical. If TRUE, additionally restrict the placebo computations to units that have valid pre-period diff_y at every placebo horizon q in 1..placebo. Mirrors the reference's same_switchers_pl option.

dont_drop_larger_lower
: Logical. By default (FALSE), drop the post-non-monotone rows of any unit whose treatment both increases strictly above and decreases strictly below the baseline. Set TRUE to keep those rows.

switchers
: One of "" (default — both directions), "in" (only switcher-in units), or "out" (only switcher-out units). Mirrors the switchers arg in DIDmultiplegtDYN.

normalized
: Logical. If TRUE, divide each per-event-time DID estimate by its pooled cumulative treatment-change magnitude delta_D_k, so the reported number is a per-unit-of-treatment effect. For binary on/off treatment this divides the k-th effect by k (so cumulative ATTs become per-period). For continuous or multivalued treatment, divides by the average per-switcher cumulative change in actual treatment magnitude over event-times 1..k. Mirrors normalized in DIDmultiplegtDYN.

bootstrap_reps
: Integer. Number of bootstrap iterations, default 0. Standard errors, confidence intervals and the joint nullity tests are computed analytically from the estimator's

asymptotic linear representation, exactly as DIDmultiplegtDYN computes them, so no resampling is needed for them. Set this above zero only if you specifically want bootstrap quantities; it does not change the reported SE column.

ci_level — Numeric in (0, 100). Confidence level for CIs.

seed — Integer. RNG seed for bootstrap iter 1 onward (iter 0 is the deterministic point estimate). Per-iter seed is seed + iter.

checkpoint_dir — Character or NULL. If non-NULL, every cell is saved here and the run is resumable. If NULL, all work is in-memory and is lost on crash.

resume — Logical. If TRUE and the manifest exists, skip cells already in it. If FALSE, error rather than overwrite.

backend — One of "auto", "reference", "r", "cpu", "cuda". See didgpu_backend_info().

n_workers — Integer >= 1. Number of parallel worker processes for the bootstrap loop. 1 = sequential (default). With n_workers > 1, uses parallel::makeCluster() to distribute bootstrap iters across cores. Each cell is saved to disk atomically so resume still works. The point estimate (cell 0) always runs sequentially first.

verbose — Logical. Print one line per completed cell. Default FALSE, so that a fit prints what DIDmultiplegtDYN prints and nothing else.

on_iter — Optional function called as on_iter(iter, value) after each cell save. Useful for emailing/logging.

bootstrap — DIDmultiplegtDYN's way of asking for a bootstrap: the number of replications, c(reps, seed), c(reps = , seed = ) or list(reps, seed). When given it sets bootstrap_reps (and seed).

graph_off — Logical. As in DIDmultiplegtDYN, the event-study graph is drawn after the fit unless this is TRUE; either way it is returned in $plot (a ggplot, when ggplot2 and cowplot are installed).

ggplot_args — A list of ggplot2 layers added to that graph.

less_conservative_se — Logical. As in DIDmultiplegtDYN: demean each switcher's outcome change within switchers sharing its full treatment path, rather than its (baseline, switch date, dose) cell, which gives smaller standard errors.

more_granular_demeaning — Logical. Turns less_conservative_se on, as in DIDmultiplegtDYN.

drop_if_d_miss_before_first_switch — Logical. Drop a group's outcomes from the first period, before its first switch, at which its treatment is missing.

effects_equal — TRUE, "all" or "lb, ub": test that all effects, or effects lb to ub, are equal. Reported in $results$p_equality_effects.

predict_het_hc2bm — Logical. With predict_het, cluster its HC2 standard errors (by cluster, or by group) with the Bell-McCaffrey small-sample adjustment.

normalized_weights — Logical, requires normalized = TRUE. Report the weight each normalized effect puts on the current treatment and on each of its lags, in $normalized_weights.

`save_results` Path of a CSV file to which the estimates, standard errors, confidence intervals and counts are written.

`save_sample` Logical. Return, in `$save_sample`, the data with `did_sample` (control, switcher-in or switcher-out) and `did_effect` (the effect a switcher's cell is used for).

`design` `c(p, path)`: tabulate the switchers' treatment paths over the effects' window, the most common paths that together cover share `p` of switchers, in `$design`; `path` is `"console"` or an Excel file.

`date_first_switch` `c(option, path)`: tabulate the switchers' first switch dates, overall (`option = ""`) or by baseline treatment (`"by_baseline_treat"`), in `$date_first_switch`; `path` is `"console"` or an Excel file.

`by` Name of a group-level, time-invariant variable: the estimation is run separately for each of its levels, in `$by_level_1`, `$by_level_2`, ..., with a combined plot.

`by_path` A positive integer, `-1` or `"all"`: the estimation is run separately for the switchers following each of the most common `by_path` treatment paths (all of them with `-1` or `"all"`), with the not-yet-switched groups of the same baseline as controls.

`reset` Non-negative integer, as in `DIDmultiplegtDYN` 2.4.0. With `reset = k > 0`, the panel is restricted to the groups with the most non-missing treatment observations, and a group is split into a new sub-group whenever its treatment has stayed unchanged for `k` periods after a change; standard errors are then clustered by the original groups unless `cluster` is given.

`avg_time_periods` Logical, as in `DIDmultiplegtDYN` 2.4.0: report the average number of periods over which the effect of a dose is accumulated, in `$avg_time_periods` and `$avg_cumul`.

## Details

`didgpu()` behaves as `DIDmultiplegtDYN::did_multiplegt_dyn()` does around the estimates too: it stops with the reference's error when no effect can be estimated, prints the reference's messages (horizons cut to what the data allow, effects or placebos that cannot be estimated, fixest's note on rows dropped from the control regressions), records the joint tests' caveats in `$results$vcov_warnings`, and prints the same tables.

## Value

An object of class `didgpu_result` (see `print.didgpu_result()`). Its `$results` holds three matrices, each laid out exactly as `DIDmultiplegtDYN::did_multiplegt_dyn()` lays out its own:

`Effects` one row per event-time `k`: the cumulative DID estimate, its SE, CI bounds and switcher / observation counts.

`ATE` a single row, the reference's `Av_tot_eff`: the average total effect **per unit of treatment**, namely `sum_k (N_k * DID_k) / sum_k (N_k * delta_k)`, where `N_k` is the switcher mass at event-time `k` and `delta_k` the average treatment change among those switchers. That denominator is what makes it a per-unit quantity, so the ATE is **not** in general the switcher-weighted mean of the `Effects` column: the two coincide only when treatment is binary and absorbing, where every `delta_k` is 1. `normalized` does not affect it, and it is suppressed (`NA`) under `trends_lin`.

`Placebos` one row per placebo, same columns as `Effects`.

## Examples

```
# Simulate a small panel with a known event-time profile.
p <- didgpu_simulate_panel(
  n_units = 40L, n_periods = 10L,
  frac_treated = 0.6,
  tau_profile = c(0.5, 1.0, 1.2),
  sigma = 0.4, seed = 17L
)

# Point estimate only (no bootstrap), pure-R backend.
fit <- didgpu(p, "Y", "unit", "period", "D",
               effects = 3L, placebo = 1L,
               bootstrap_reps = 0L, backend = "r",
               verbose = FALSE)
coef(fit)

# With a small bootstrap for SEs / CIs.

fit_boot <- didgpu(p, "Y", "unit", "period", "D",
                   effects = 3L, placebo = 1L,
                   bootstrap_reps = 20L, seed = 1L,
                   backend = "r", verbose = FALSE)
print(fit_boot)
confint(fit_boot)
```

---

`didgpu_aggregate_cells` *Read all committed cells back into memory*

---

## Description

Uses the manifest as source of truth: skips any .rds not in the manifest, and warns about manifest rows whose .rds is missing.

## Usage

```
didgpu_aggregate_cells(checkpoint_dir)
```

## Arguments

`checkpoint_dir` Path to an existing checkpoint directory.

## Value

Named list: cell `b` is at position `as.character(b)`.

## Examples

```
p <- didgpu_simulate_panel(n_units = 30L, n_periods = 8L, seed = 1L)
cdir <- tempfile("agg_demo_")
didgpu(p, "Y", "unit", "period", "D",
        effects = 1L, bootstrap_reps = 2L, seed = 1L,
        checkpoint_dir = cdir, backend = "r", verbose = FALSE)
cells <- didgpu_aggregate_cells(cdir)
length(cells)              # 3 (point estimate + 2 bootstrap reps)
unlink(cdir, recursive = TRUE)
```

---

# didgpu_backend_info *Report installed backends and their availability*

---

## Description

Report installed backends and their availability

## Usage

```
didgpu_backend_info()
```

## Value

A data.frame with columns `backend`, `available`, `notes`.

## Examples

```
didgpu_backend_info()
```

---

# didgpu_bacon *Goodman-Bacon decomposition of the TWFE DiD estimator*

---

## Description

Decomposes the static two-way fixed-effects DiD coefficient into the weighted average of all 2x2 timing-group comparisons (Goodman-Bacon 2021). The headline diagnostic is the total weight placed on "forbidden" comparisons that use already-treated units as controls – the source of TWFE bias under heterogeneous, dynamic treatment effects. Pair it with `didgpu_twfe()` (the estimate) and `didgpu()` / `didgpu_cs()` (robust alternatives).

## Usage

```
didgpu_bacon(df, outcome, group, time, treatment)
```

### Arguments

| | |
|---|---|
| df | A data.frame / data.table panel. |
| outcome, group, time, treatment | Character column names: outcome, unit id, time id, and the binary 0/1 absorbing treatment indicator. |

### Details

Scope: a **balanced** panel with a **binary, absorbing** (staggered-adoption) treatment – the canonical Goodman-Bacon case. Always-treated units (treated in the first period, no pre-period) are dropped with a note, as in the reference implementation. For unbalanced panels, treatment that switches off, or continuous treatment, use didgpu().

### Value

A didgpu_bacon object: a list with beta_twfe (the static TWFE DiD coefficient on the decomposition sample), comparisons (a data.frame: type, treated, control timing values, estimate, weight), summary (per-type total weight and weighted-average 2x2 estimate), forbidden_weight (total weight on already-treated-control comparisons), and n_always_treated (units dropped).

### See Also

didgpu_twfe(), didgpu(), didgpu_cs().

### Examples

```
p <- didgpu_simulate_panel(n_units = 80L, n_periods = 12L,
                           tau_profile = c(0.5, 1.0), seed = 7L)
p$D <- as.integer(p$D >= 0.5)            # binary staggered treatment
bd <- didgpu_bacon(p, "Y", "unit", "period", "D")
bd
```

---

## didgpu_bootstrap_more *Add more bootstrap reps to an existing checkpoint*

---

### Description

Use this when you already have, say, 100 reps committed and you realise you want 200. Rather than throwing away the 100 you have and starting over, this function patches the checkpoint's recorded bootstrap_reps upward and re-invokes didgpu() with resume = TRUE, so only the new (extra_reps) cells are computed.

### Usage

```
didgpu_bootstrap_more(checkpoint_dir, df, extra_reps, ...)
```

### Arguments

| | |
|---|---|
| checkpoint_dir | Path to existing checkpoint. |
| df | The same panel originally passed (validated by hash). |
| extra_reps | Integer >= 1: how many MORE reps to add. |
| ... | Additional arguments forwarded to didgpu() (e.g. n_workers, verbose). Overriding identity-relevant arguments like outcome, effects, seed will fail compatibility check. |

### Details

The new cells use seeds seed + (current_reps + 1)..(current_reps + extra_reps), which is exactly what didgpu() would have done on a fresh run with the larger bootstrap count. The result is therefore identical to starting from scratch with the larger count.

### Value

The aggregated result, same shape as didgpu().

### Examples

```
p <- didgpu_simulate_panel(n_units = 40L, n_periods = 10L,
                           tau_profile = c(0.5, 1.0), seed = 17L)
cdir <- tempfile("didgpu_more_demo_")
didgpu(p, "Y", "unit", "period", "D",
        effects = 2L, bootstrap_reps = 3L, seed = 1L,
        checkpoint_dir = cdir, backend = "r", verbose = FALSE)
# Add 2 more bootstrap reps without redoing the first 3.
fit <- didgpu_bootstrap_more(cdir, df = p, extra_reps = 2L,
                             verbose = FALSE)
fit$args$bootstrap_reps   # now 5
unlink(cdir, recursive = TRUE)
```

---

## didgpu_by *Estimate didgpu separately for each level of a grouping variable*

---

### Description

Wrapper that splits the panel by the unique values of by_var and runs didgpu() on each subset, returning a named list of didgpu_result objects. All other arguments are forwarded identically to each per-level call.

## Usage

```
didgpu_by(
  df,
  by_var,
  outcome,
  group,
  time,
  treatment,
  ...,
  checkpoint_dir = NULL,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| df | A panel data.frame. |
| by_var | Character. Name of the grouping column. Each row in df must have exactly one value of this column. Levels must be coercible to character (used as subgroup labels and as subdirectory names under checkpoint_dir, if set). |
| outcome, group, time, treatment | Forwarded to didgpu(). |
| ... | Additional arguments forwarded to didgpu() (e.g. effects, placebo, bootstrap_reps, normalized). |
| checkpoint_dir | If non-NULL, each subgroup writes into a subdirectory of this path named after the level. |
| verbose | Logical. Print one line per subgroup as it starts. |

## Details

If checkpoint_dir is supplied, each subgroup writes into checkpoint_dir/<level>/. Resume works the same way as a normal didgpu() call inside each subgroup directory.

## Value

An object of class didgpu_by_result — a named list of didgpu_result objects, one per subgroup, plus an attribute by_var recording the grouping column name.

## Examples

```
p <- didgpu_simulate_panel(n_units = 40L, n_periods = 10L,
                           tau_profile = c(0.5, 1.0),
                           seed = 17L)
p$region <- ifelse(p$unit %% 2L == 0L, "north", "south")
fit_by <- didgpu_by(p, "region",
                    outcome = "Y", group = "unit",
                    time = "period", treatment = "D",
                    effects = 2L, bootstrap_reps = 0L,
                    backend = "r", verbose = FALSE)
print(fit_by)
```

didgpu_by_path *Run didgpu separately per treatment trajectory*

## Description

Wrapper: calls `didgpu_compute_paths()` to add a `path` column to the panel, then runs `didgpu_by(df, by_var = "path", ...)` on the result. Subgroups (rows with `path = NA` because they're not in `top_n`) are dropped before estimation.

## Usage

```
didgpu_by_path(
  df,
  outcome,
  group,
  time,
  treatment,
  effects = 1L,
  top_n = NULL,
  ...,
  checkpoint_dir = NULL,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `df` | A panel data.frame. |
| `outcome`, `group`, `time`, `treatment` | Column names (we only need them to compute F_g consistently with `didgpu()`). |
| `effects` | Integer. Number of post-switch periods to encode in the path (i.e. the path string has length `effects + 1`). |
| `top_n` | Integer or NULL. If non-NULL, keep only the `top_n` most-common paths; groups with rarer paths are tagged with `NA_character_` for the path column. NULL = keep every distinct path. |
| `...` | Additional arguments forwarded to `didgpu()` (e.g. `placebo`, `bootstrap_reps`, `normalized`). |
| `checkpoint_dir` | If non-NULL, each subgroup writes into a subdirectory of this path named after the path string. |
| `verbose` | Logical. Print one line per subgroup as it starts. |

## Value

A `didgpu_by_result`: named list of `didgpu_result` objects, one per path.

## Examples

```
p <- didgpu_simulate_panel(n_units = 100L, n_periods = 12L,
                           tau_profile = c(0.5, 1.0),
                           seed = 17L)
fit_by_path <- didgpu_by_path(
  p, outcome = "Y", group = "unit", time = "period", treatment = "D",
  effects = 2L, top_n = 3L,
  bootstrap_reps = 0L, backend = "r", verbose = FALSE
)
print(fit_by_path)
```

---

`didgpu_compare` *Compare didgpu r-backend to the DIDmultiplegtDYN reference*

---

## Description

Runs both backends on the same panel and same call args, then reports the max absolute disagreement per output column. Useful for verifying didgpu produces output identical to the reference on your own data before using it in production.

## Usage

```
didgpu_compare(
  df,
  outcome,
  group,
  time,
  treatment,
  effects = 1L,
  placebo = 0L,
  cluster = NULL,
  switchers = "",
  tolerance = 1e-10,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `df` | A data.frame (or data.table) panel. Must contain the columns named by `outcome`, `group`, `time`, `treatment`. |
| `outcome, group, time, treatment` | Character. Column names. |
| `effects` | Integer. Number of post-treatment event-times to estimate (e = 1..effects). Must be >= 1. |
| `placebo` | Integer. Number of pre-treatment placebos. 0 to skip. |

| | |
|---|---|
| cluster | Character or NULL. Column name for cluster bootstrap; default NULL clusters by group. |
| switchers | One of "" (default — both directions), "in" (only switcher-in units), or "out" (only switcher-out units). Mirrors the switchers arg in DIDmultiplegtDYN. |
| tolerance | Numeric. Threshold above which a disagreement is flagged as a failure. Default 1e-10 (effectively machine epsilon for our pipeline). |
| verbose | Logical. Print a per-column table. |

## Value

Invisibly a list with pass (logical scalar), report (data.frame of per-column max diffs), fit_r and fit_ref (the two full fits). If the reference is not installed, returns immediately with pass = NA and a warning.

## Examples

```
## Not run:
# Requires the DIDmultiplegtDYN package to be installed.
p <- didgpu_simulate_panel(n_units = 40L, n_periods = 10L, seed = 17L)
report <- didgpu_compare(p, "Y", "unit", "period", "D",
                         effects = 2L, placebo = 1L)
report$pass
head(report$report)

## End(Not run)
```

---

# didgpu_compute_paths *Add a per-group treatment-trajectory column to a panel*

---

## Description

For each group, builds a comma-separated string of the treatment values observed at (F_g - 1, F_g, F_g + 1, ..., F_g + effects - 1): the baseline period plus the first effects post-switch periods. Groups with no switch (F_g > T_max) get a "no-switch" path string.

## Usage

```
didgpu_compute_paths(
  df,
  outcome,
  group,
  time,
  treatment,
  effects = 1L,
  top_n = NULL
)
```

### Arguments

| | |
|---|---|
| df | A panel data.frame. |
| outcome, group, time, treatment | Column names (we only need them to compute F_g consistently with didgpu()). |
| effects | Integer. Number of post-switch periods to encode in the path (i.e. the path string has length effects + 1). |
| top_n | Integer or NULL. If non-NULL, keep only the top_n most-common paths; groups with rarer paths are tagged with NA_character_ for the path column. NULL = keep every distinct path. |

### Details

Use the resulting path column as a by_var to estimate treatment effects separately per trajectory (e.g. via didgpu_by()).

### Value

The input data.frame with a new column path of type character. Order of rows preserved. Groups with no observed treatment trajectory get NA.

### Examples

```
p <- didgpu_simulate_panel(n_units = 60L, n_periods = 12L,
                           tau_profile = c(0.5, 1.0),
                           seed = 17L)
aug <- didgpu_compute_paths(p, "Y", "unit", "period", "D",
                            effects = 2L)
table(aug$path, useNA = "ifany")
```

---

## didgpu_cpu_hello *Smoke target for the CPU build path*

---

### Description

Returns a fixed value (42). Used to verify the CPU half of the build system works in isolation from any GPU machinery.

### Value

Numeric scalar.

didgpu_cs *Estimate Callaway-Sant'Anna (2021) staggered-treatment DiD*

## Description

Estimates group-time average treatment effects ATT(g, t) for each cohort g (defined by first-treatment period) and each post-treatment time t >= g, then aggregates into the requested summary.

## Usage

```
didgpu_cs(
  df,
  outcome,
  group,
  time,
  treatment,
  control_group = c("never", "notyet"),
  est_method = c("OR", "IPW", "DR"),
  aggregation = c("event", "group", "calendar", "overall"),
  covariates = NULL,
  bootstrap_reps = 0L,
  bootstrap_kind = c("cluster", "multiplier"),
  ci_level = 95,
  seed = 1L,
  backend = "auto",
  verbose = TRUE,
  base_period = c("varying", "universal"),
  allow_unbalanced_panel = FALSE,
  first_treat = NULL
)
```

## Arguments

| | |
|---|---|
| df | A panel data.frame. |
| outcome, group, time, treatment | Column names. treatment must be binary (0/1). |
| control_group | Either "never" (never-treated; default) or "notyet" (not-yet-treated). |
| est_method | One of "OR", "IPW", "DR". Default "OR". |
| aggregation | One of "event", "group", "calendar", "overall". Default "event". Use didgpu_cs_aggregate() to compute additional aggregations from a single fit. |
| covariates | Optional character vector of time-invariant covariate column names for OR / IPW / DR adjustment. |
| bootstrap_reps | Integer. Number of bootstrap reps for SE estimation. Default 0L (point estimate only). |

bootstrap_kind One of "cluster" (resample units, refit per rep) or "multiplier" (multiplier wild bootstrap on the influence functions; much faster for large B). Default "cluster".

ci_level Numeric in (0, 100). Default 95.

seed Integer.

backend One of "auto", "r", "cuda".

verbose Logical. Print progress per (g, t).

base_period "varying" (the default, as in did::att_gt) or "universal". Under "varying" a pre-treatment cell compares period t with the period before it; under "universal" every cell is compared with the last period before g, and that period's own cell is reported as exactly 0. Post-treatment cells, and so the overall ATT, are identical under both.

allow_unbalanced_panel
Logical, as in did::att_gt. On an unbalanced panel, FALSE (the default) first drops every unit not observed in all periods; TRUE keeps them and uses the repeated-cross-section estimators of Sant'Anna and Zhao (2020), as did does. Ignored on a balanced panel.

first_treat Optional name of a column holding the period each unit is first treated, 0 if never – did::att_gt's gname. If omitted, the cohort is inferred as the first period in which the unit is observed with treatment == 1. That is identical on a balanced panel, but under allow_unbalanced_panel = TRUE a unit whose adoption period is unobserved would land in a later cohort, so pass it there (a warning says so when it matters).

## Details

Three inner estimators (est_method):

- "OR" (outcome regression, default): fit a linear model of the outcome change on covariates among controls only; predict counterfactual for treated; ATT = mean of (observed - predicted) over treated cohort cells.
- "IPW" (inverse-probability weighting): weight treated and control cells by inverse propensity score.
- "DR" (doubly robust): combines OR and IPW; consistent if either model is correct. Preferred in practice.

Four aggregation schemes (aggregation):

- "event" (default): event-study, indexed by event-time e = t - g.
- "group": per-cohort average effect over post-treatment periods.
- "calendar": per-calendar-time average over cohorts already treated.
- "overall": single scalar summary, the weighted average of all post-treatment (g, t) cells.

## Value

An object of class `didgpu_cs_result`:

- `att_gt`: long-form data.frame of ATT(g, t) estimates.
- `aggregation`: the chosen aggregation summary (event-study / group / calendar / overall).
- `args`: the canonical args bundle (for re-aggregation).

## References

Callaway, B. and Sant'Anna, P. (2021). "Difference-in-Differences with multiple time periods." *Journal of Econometrics* 225(2): 200-230.

## Examples

```
p <- didgpu_simulate_panel(n_units = 100L, n_periods = 12L,
                           tau_profile = c(0.5, 1.0), seed = 17L)
fit <- didgpu_cs(p, "Y", "unit", "period", "D",
                 est_method = "OR", aggregation = "event",
                 bootstrap_reps = 0L, backend = "r", verbose = FALSE)
print(fit)
```

---

# didgpu_cs_aggregate *Re-aggregate a fitted didgpu_cs result with a different scheme*

---

## Description

Re-aggregate a fitted didgpu_cs result with a different scheme

## Usage

```
didgpu_cs_aggregate(
  fit,
  aggregation = c("event", "group", "calendar", "overall")
)
```

## Arguments

| | |
|---|---|
| `fit` | A `didgpu_cs_result` object. |
| `aggregation` | One of `"event"`, `"group"`, `"calendar"`, `"overall"`. |

## Value

The same `didgpu_cs_result` with a new `aggregation` slot.

`didgpu_cs_continuous` *Callaway-Goodman-Bacon-Sant'Anna (2024) continuous-treatment DiD*

## Description

Estimates the continuous-treatment dose-response: the level effect $ATT(d)$ and the causal response (slope) $ACRT(d) = ATT'(d)$ of a continuous dose, comparing dose-$d$ units to a never-treated (dose 0) comparison group via a within-unit before/after change. The dose-response is fit by a B-spline regression; ATT(d) subtracts the comparison mean change and ACRT(d) is the analytic spline derivative. Standard errors come from a unit-level multiplier bootstrap.

## Usage

```
didgpu_cs_continuous(df, yname, dname, gname, tname, idname, dvals = NULL,
  degree = 3L, num_knots = 0L, control_group = "nevertreated",
  bootstrap_reps = 200L, ci_level = 95, seed = 1L, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `df` | A data.frame / data.table panel. |
| `yname`, `dname`, `gname`, `tname`, `idname` | Character column names: outcome, continuous dose, cohort (0 = never-treated; positive = adoption period), time, and unit id. |
| `dvals` | Numeric dose values at which to report the curve. Default: quantiles 0.1..0.99 of the positive doses. |
| `degree` | B-spline degree (default 3, cubic). |
| `num_knots` | Number of interior knots (default 0). Knots are placed at interior quantiles of the positive dose. |
| `control_group` | `"nevertreated"` (default). |
| `bootstrap_reps` | Multiplier-bootstrap replicates for SEs (default 200; 0 to skip). |
| `ci_level` | Confidence level in percent (default 95). |
| `seed` | Bootstrap RNG seed. |
| `verbose` | Logical. |

## Details

Native reimplementation: the spline basis uses **splines2** (the same library **contdid** uses, so the basis matches bit-for-bit), but the DiD logic, dose-response and ATT/ACRT extraction are didgpu's own. Scope: a single treated cohort vs a never-treated comparison (the canonical continuous-DiD design). Point estimates match `contdid::cont_did` exactly.

## Value

A `didgpu_cs_continuous_result`: a list with `dose` (the dvals), `att.d`, `acrt.d` (and their SEs / CIs when bootstrapped), `overall_att`, `overall_acrt`, and `args`.

## References

Callaway, B., Goodman-Bacon, A. & Sant'Anna, P.H.C. (2024). Difference-in-Differences with a Continuous Treatment. NBER WP 32117.

## See Also

`didgpu_cs()` for the binary staggered estimator.

## Examples

```
if (requireNamespace("contdid", quietly = TRUE) &&
    requireNamespace("splines2", quietly = TRUE)) {
  d <- contdid::simulate_contdid_data(n = 400, num_time_periods = 2)
  didgpu_cs_continuous(d, "Y", "D", "G", "time_period", "id",
                       degree = 3, num_knots = 2, bootstrap_reps = 0)
}
```

---

# didgpu_did_continuous *Continuous-treatment DiD with no stayers (de Chaisemartin-D'Haultfoeuille 2024)*

---

## Description

First-difference difference-in-differences for a continuous treatment that changes for (almost) all units, so there are no pure stayers. Estimates the level effect of a dose change, $effect(d) = E[dY|dD = d] - E[dY|dD = 0]$, and the average causal response $ACR(d) = d/dd\, E[dY|dD = d]$, where $dY, dD$ are within-unit first differences and $E[dY|dD = 0]$ is the common trend identified from quasi-stayers (units with $dD \approx 0$).

## Usage

```
didgpu_did_continuous(df, outcome, treatment, id, time,
  estimator = c("parametric", "nonparametric"), degree = 2L, dvals = NULL,
  bandwidth = NULL, bootstrap_reps = 200L, ci_level = 95, seed = 1L,
  verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `df` | A data.frame / data.table panel. |
| `outcome`, `treatment`, `id`, `time` | Character column names: outcome, the continuous treatment, unit id, and time. |
| `estimator` | `"parametric"` (degree-degree polynomial in dD; sqrt(n)) or `"nonparametric"` (local-linear in dD; n^(2/5); EXPERIMENTAL). |
| `degree` | Polynomial degree for the parametric estimator (default 2). |
| `dvals` | Dose-change values at which to report effect(d)/ACR(d). Default: quantiles 0.1..0.9 of the nonzero dD. |

| | |
|---|---|
| bandwidth | Local-linear bandwidth (nonparametric). Default: a Silverman-type rule, `1.06 * sd(dD) * n^(-1/5)`. |
| bootstrap_reps | Multiplier-bootstrap replicates for SEs (default 200). |
| ci_level | Confidence level in percent (default 95). |
| seed | Bootstrap RNG seed. |
| verbose | Logical. |

## Details

Two estimators of $E[dY|dD = d]$: `"parametric"` fits a degree-degree polynomial in dD by OLS (sqrt(n) rate; the authors' parametric alternative), while `"nonparametric"` uses a local-linear (Gaussian-kernel) regression evaluated at each dose with quasi-stayers near it (n^(2/5) rate; slower because it estimates a derivative). Both are validated by simulation (recovering a known dose-response); there is no maintained R reference package to cross-check bit-for-bit, so the nonparametric path in particular is flagged EXPERIMENTAL.

## Value

A `didgpu_did_continuous_result`: a list with `dose` (dvals), `effect.d`, `acr.d` (+ SEs/CIs), `overall_acr`, `estimator`, and `args`.

## References

de Chaisemartin, C., D'Haultfoeuille, X., Pasquier, F. & Vazquez-Bare, G. (2024). Difference-in-Differences Estimators for Treatments Continuously Distributed at Every Period.

## See Also

`didgpu_cs_continuous()` (Callaway et al. 2024 dose-response); `didgpu()` with `continuous=` for the DIDmultiplegtDYN-equivalent parametric continuous estimator in the dynamic framework.

## Examples

```
set.seed(1); nU <- 800L
dD <- stats::rnorm(nU, 0, 1)
dY <- 0.3 + 2 * dD - 0.5 * dD^2 + stats::rnorm(nU, 0, 0.5)
df <- data.frame(id = rep(seq_len(nU), each = 2L),
                 t = rep(1:2, nU),
                 D = as.numeric(rbind(0, dD)),
                 Y = as.numeric(rbind(0, dY)))
didgpu_did_continuous(df, "Y", "D", "id", "t", estimator = "parametric",
                      degree = 2, bootstrap_reps = 0)
```

`didgpu_did_static` *de Chaisemartin-D'Haultfoeuille (2020) DID_M instantaneous estimator*

## Description

The DID_M estimator for a binary treatment that may switch on AND off (non-absorbing). It compares each switching unit's period-over-period outcome change to that of stayers with the same prior-period treatment, and averages over all switch events. Consistent for the ATT among switchers under common trends + no anticipation, even with heterogeneous and dynamic treatment effects (where TWFE is biased). Standard errors are from a cluster (by default unit) bootstrap, matching the package's other estimators.

## Usage

```
didgpu_did_static(df, outcome, group, time, treatment, weight = NULL,
  bootstrap_reps = 100L, cluster = NULL, ci_level = 95, seed = 1L,
  verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `df` | A data.frame / data.table panel. |
| `outcome`, `group`, `time`, `treatment` | Character column names: outcome, unit id, time id, and a binary 0/1 (possibly non-absorbing) treatment. |
| `weight` | Optional character column name of observation weights. |
| `bootstrap_reps` | Integer; cluster-bootstrap replicates for the SE (0 to skip and return only the point estimate). Default 100. |
| `cluster` | Optional character column name to resample in the bootstrap. Defaults to `group`. |
| `ci_level` | Confidence level in percent (default 95). |
| `seed` | RNG seed for the bootstrap. |
| `verbose` | Logical. |

## Value

A `didgpu_did_static_result`: a list with `did` (the DID_M point estimate), `se`, `ci`, `n_switchers`, `per_period` (per-period switcher counts and directional DiDs), and `args`.

## References

de Chaisemartin, C. & D'Haultfoeuille, X. (2020). Two-Way Fixed Effects Estimators with Heterogeneous Treatment Effects. *American Economic Review* 110(9): 2964-2996.

## See Also

`didgpu()` for the dynamic generalization, `didgpu_twfe()` / `didgpu_bacon()` for the biased TWFE baseline and its decomposition.

## Examples

```
p <- didgpu_simulate_panel(n_units = 80L, n_periods = 8L,
                           tau_profile = c(0.5, 1.0), seed = 5L)
p$D <- as.integer(p$D >= 0.5)
didgpu_did_static(p, "Y", "unit", "period", "D", bootstrap_reps = 0L)
```

---

`didgpu_equivalence` *Pre-trends equivalence (TOST) test for a didgpu event study*

---

## Description

Runs a two-one-sided-tests (TOST) equivalence test on the placebo (pre-treatment) estimates of a `didgpu()` fit. For a user-supplied margin `delta`, each placebo horizon tests $H_0 : |\theta| \geq \delta$ against $H_1 : |\theta| < \delta$; rejecting $H_0$ is positive evidence the pre-trend is within $\pm$`delta`. The joint claim "all placebos lie within `delta`" follows by the intersection-union principle: it holds iff every horizon individually rejects.

## Usage

```
didgpu_equivalence(x, delta, alpha = 0.05)
```

## Arguments

| | |
|---|---|
| `x` | A `didgpu_result` from `didgpu()` run with `placebo` > `0` and `bootstrap_reps` > `0` (equivalence testing needs standard errors). |
| `delta` | Positive numeric equivalence margin on the outcome scale – the largest pre-trend you would consider economically negligible. |
| `alpha` | One-sided significance level (default 0.05). |

## Details

Unlike a conventional placebo test (where a *large* p-value is the hoped-for result but only weakly informative), here a *small* `equivalence_p` is the hoped-for result and is a genuine rejection.

## Value

A `didgpu_equivalence` object (a data.frame with one row per placebo horizon: `event_time`, `estimate`, `std.error`, `equivalence_p`, `passes_at_delta`) carrying attributes `delta`, `alpha`, `joint_pass` (TRUE iff every horizon passes; NA if any SE is missing) and `breakdown_delta` (the smallest margin at which the joint equivalence would hold at level `alpha`).

## See Also

`didgpu()`, `didgpu_fect_equivalence()` for the fect-family analogue, `didgpu_honest_did()` for HonestDiD sensitivity bounds.

## Examples

```
p <- didgpu_simulate_panel(n_units = 80L, n_periods = 12L,
                           tau_profile = c(0.5, 1.0), seed = 7L)
p$D <- as.integer(p$D >= 0.5)
fit <- didgpu(p, "Y", "unit", "period", "D", effects = 3L, placebo = 3L,
              bootstrap_reps = 200L, verbose = FALSE)
didgpu_equivalence(fit, delta = 0.5)
```

---

# didgpu_estimate_runtime

*Estimate the runtime of a planned didgpu() call*

---

## Description

Times a couple of single-iter fits and extrapolates to the bootstrap_reps total, accounting for parallel workers and any already-completed cells in the checkpoint dir. Useful for deciding whether to grab coffee or to scale down `bootstrap_reps`.

## Usage

```
didgpu_estimate_runtime(
  df,
  outcome,
  group,
  time,
  treatment,
  effects = 1L,
  placebo = 0L,
  cluster = NULL,
  controls = NULL,
  weight = NULL,
  trends_nonparam = NULL,
  only_never_switchers = FALSE,
  same_switchers = FALSE,
  dont_drop_larger_lower = FALSE,
  switchers = "",
  bootstrap_reps = 100L,
  checkpoint_dir = NULL,
  backend = "auto",
  n_workers = 1L,
  probes = 2L
)
```

## Arguments

| | |
|---|---|
| df | A data.frame (or data.table) panel. Must contain the columns named by outcome, group, time, treatment. |
| outcome, group, time, treatment | Character. Column names. |
| effects | Integer. Number of post-treatment event-times to estimate (e = 1..effects). Must be >= 1. |
| placebo | Integer. Number of pre-treatment placebos. 0 to skip. |
| cluster | Character or NULL. Column name for cluster bootstrap; default NULL clusters by group. |
| controls | Character vector or NULL. Names of covariate columns to control for. The point estimate adjusts diff_y by the FWL projection on these covariates' first differences, restricted to never-switcher rows within each baseline-treatment cohort. |
| weight | Character or NULL. Name of a per-row weight column. If NULL, every observation gets weight 1 (the binary case). |
| trends_nonparam | Character or NULL. Name of a categorical column to extend the cohort grouping by. With this set, cohort averages are computed per (time, d_sq, trends_nonparam) instead of per (time, d_sq), allowing time trends to vary by the extra grouping (e.g., industry). |
| only_never_switchers | Logical. If TRUE, restrict controls to strictly never-switched units (drop pre-switch rows of units that eventually switch). |
| same_switchers | Logical. If TRUE, restrict switcher rows to units that have valid controls at every event-time q in 1..effects. Mirrors the reference's same_switchers option. |
| dont_drop_larger_lower | Logical. By default (FALSE), drop the post-non-monotone rows of any unit whose treatment both increases strictly above and decreases strictly below the baseline. Set TRUE to keep those rows. |
| switchers | One of "" (default — both directions), "in" (only switcher-in units), or "out" (only switcher-out units). Mirrors the switchers arg in DIDmultiplegtDYN. |
| bootstrap_reps | Integer. Number of bootstrap iterations. |
| checkpoint_dir | Character or NULL. If non-NULL, every cell is saved here and the run is resumable. If NULL, all work is in-memory and is lost on crash. |
| backend | One of "auto", "reference", "r", "cpu", "cuda". See didgpu_backend_info(). |
| n_workers | Integer >= 1. Number of parallel worker processes for the bootstrap loop. 1 = sequential (default). With n_workers > 1, uses parallel::makeCluster() to distribute bootstrap iters across cores. Each cell is saved to disk atomically so resume still works. The point estimate (cell 0) always runs sequentially first. |
| probes | Integer. Number of timing probes to average (default 2). |

## Value

Invisibly a list with wall_per_iter (median seconds per point-estimate fit), n_remaining (cells still to do), n_workers, total_seconds, total_human (formatted string).

## Examples

```
p <- didgpu_simulate_panel(n_units = 40L, n_periods = 10L,
                           tau_profile = c(0.5, 1.0),
                           seed = 17L)
info <- didgpu_estimate_runtime(p, "Y", "unit", "period", "D",
                                effects = 2L, bootstrap_reps = 100L,
                                backend = "r", probes = 1L)
info$total_human
```

---

`didgpu_event_study_data` *Extract long-format event-study data from a didgpu_result*

---

## Description

Returns one row per event-time (placebos at negative k, effects at k = 1..effects), with columns `event_time`, `estimate`, `std.error`, `conf.low`, `conf.high`, `kind`. Suitable for direct plotting with ggplot2 — e.g.:

## Usage

```
didgpu_event_study_data(x)
```

## Arguments

| | |
|---|---|
| `x` | A `didgpu_result`. |

## Details

```
library(ggplot2)
ggplot(didgpu_event_study_data(fit),
       aes(event_time, estimate)) +
  geom_pointrange(aes(ymin = conf.low, ymax = conf.high)) +
  geom_hline(yintercept = 0, linetype = "dashed")
```

Note: placebos are conventionally plotted at "negative" event time (`-1, -2, ...`) even though they are computed at horizon `1, 2, ...` from the switch. The `event_time` column follows the convention.

## Value

A data.frame.

## Examples

```
p <- didgpu_simulate_panel(n_units = 40L, n_periods = 10L, seed = 17L)
fit <- didgpu(p, "Y", "unit", "period", "D",
              effects = 2L, placebo = 1L,
              bootstrap_reps = 0L, backend = "r", verbose = FALSE)
didgpu_event_study_data(fit)
```

`didgpu_fect` *Counterfactual-prediction DiD estimators (fect family)*

## Description

Fits one of three counterfactual-prediction estimators (Liu, Wang, Xu 2024) on a panel and returns an estimate of the average treatment effect on the treated (ATT) along with placebo-style robustness diagnostics. The three methods differ in how they model the counterfactual outcome Y(0):

## Usage

```
didgpu_fect(
  df,
  outcome,
  group,
  time,
  treatment,
  method = c("fe", "ife", "mc"),
  min_T0 = NULL,
  effects = 1L,
  r = 2L,
  lambda = NULL,
  tol = 1e-05,
  max_iter = 500L,
  bootstrap_reps = 100L,
  seed = 1L,
  checkpoint_dir = NULL,
  backend = "auto",
  n_workers = 1L,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `df` | A data.frame (or data.table) panel. Must contain the columns named by `outcome`, `group`, `time`, `treatment`. |
| `outcome, group, time, treatment` | Character. Column names. |
| `method` | One of `"fe"`, `"ife"`, `"mc"`. Default `"fe"`. |
| `min_T0` | Integer. Minimum number of UNTREATED periods a unit must have to be kept. A unit's counterfactual is identified only from its untreated observations: the unit fixed effect needs at least one, an r-factor loading needs several. Units below the threshold are dropped with a warning and the count is returned as `$n_units_dropped`. `NULL` (the default) follows fect's rule: `1` for `method = "fe"`, `5` for `"ife"` and `"mc"`. Because the defaults differ by method, `method = "ife"`, `r = 0` does NOT generally equal `method = "fe"`; they fit different samples. Pass `min_T0 = 1` to make them agree. Mirrors `min.T0` in fect. |

| | |
|---|---|
| effects | Integer. Number of post-treatment event-times to estimate (e = 1..effects). Must be >= 1. |
| r | Integer. For method "ife", the number of latent factors. Default 2L. Ignored for "fe" and "mc". |
| lambda | Numeric or NULL. For method "mc", the nuclear-norm penalty parameter. If NULL (default), chosen by cross-validation over a default grid. Ignored for "fe" and "ife". |
| tol | Numeric. Convergence tolerance for the iterative methods ("fe", "ife", "mc"). Default 1e-5. |
| max_iter | Integer. Maximum iterations for the iterative methods. Default 500L. |
| bootstrap_reps | Integer. Number of bootstrap iterations. |
| seed | Integer. RNG seed for bootstrap iter 1 onward (iter 0 is the deterministic point estimate). Per-iter seed is seed + iter. |
| checkpoint_dir | Character or NULL. If non-NULL, every cell is saved here and the run is resumable. If NULL, all work is in-memory and is lost on crash. |
| backend | One of "auto", "reference", "r", "cpu", "cuda". See didgpu_backend_info(). |
| n_workers | Integer >= 1. Number of parallel worker processes for the bootstrap loop. 1 = sequential (default). With n_workers > 1, uses parallel::makeCluster() to distribute bootstrap iters across cores. Each cell is saved to disk atomically so resume still works. The point estimate (cell 0) always runs sequentially first. |
| verbose | Logical. Print one line per completed cell. |

## Details

- "fe" — two-way fixed effects (unit + time FE), no factor loadings. Fast; requires the strict parallel-trends assumption.
- "ife" — interactive fixed effects (Bai 2009): unit FE + time FE + r latent factors lambda_i' F_t. Relaxes parallel trends to allow unit-specific time-varying confounders.
- "mc" — matrix completion (Athey et al. 2021): low-rank recovery of the control-only outcome matrix via nuclear-norm soft-thresholding. No need to choose r in advance.

All three share the per-cell checkpoint + resume + parallel bootstrap infrastructure of didgpu().

**Status: SCAFFOLDED.** The R-side public API is stable; the actual estimation logic is not yet implemented. Each method currently raises a NotImplementedError. The CUDA + Rcpp+Eigen backends for these will land in subsequent releases.

## Value

An object of class didgpu_fect_result — same general shape as didgpu_result (effects table, placebo table, ATE, S3 methods work the same way).

## References


- Liu, L., Wang, Y., and Xu, Y. (2024). "A practical guide to counterfactual estimators for causal inference with time-series cross-sectional data." *American Journal of Political Science*.

- Bai, J. (2009). "Panel data models with interactive fixed effects." *Econometrica* 77 (4): 1229-1279.
- Athey, S., Bayati, M., Doudchenko, N., Imbens, G., and Khosravi, K. (2021). "Matrix completion methods for causal panel data models." *JASA* 116 (536): 1716-1730.


## Examples

```
## Not run:
# NOT YET IMPLEMENTED -- the example shows the planned interface only.
p <- didgpu_simulate_panel(n_units = 100L, n_periods = 20L, seed = 17L)
fit <- didgpu_fect(p, outcome = "Y", group = "unit",
                   time = "period", treatment = "D",
                   method = "ife", r = 2L,
                   bootstrap_reps = 100L,
                   checkpoint_dir = "checkpoints/fect_run1")
print(fit)
plot(fit)

## End(Not run)
```

---

# didgpu_fect_equivalence

*Equivalence test on fect placebos*

---

## Description

Tests H0: |effect at placebo horizon h| > delta for a user-supplied tolerance delta. REJECTING this null lets us conclude the placebo deviation is below delta. Implemented as a two-one-sided-tests (TOST) procedure on the bootstrap distribution.

## Usage

```
didgpu_fect_equivalence(placebo_result, delta)
```

## Arguments

| | |
|---|---|
| `placebo_result` | A data.frame returned by `didgpu_fect_placebo()` with non-NA `se` (i.e., from a run with `bootstrap_reps > 0`). |
| `delta` | Numeric. The equivalence margin. |

## Value

The input data.frame augmented with `equivalence_p` (the TOST p-value: small means the placebo deviation IS below delta) and `passes_at_delta` (logical, TRUE if equivalence_p < 0.05).

## Examples

```
p <- didgpu_simulate_panel(n_units = 60L, n_periods = 12L,
                           tau_profile = c(0.5, 1.0), seed = 17L)
pl <- didgpu_fect_placebo(p, "Y", "unit", "period", "D",
                          method = "fe", n_placebos = 2L,
                          bootstrap_reps = 30L, seed = 1L)
eq <- didgpu_fect_equivalence(pl, delta = 0.5)
eq
```

---

`didgpu_fect_placebo` *Run placebo (pre-treatment) tests on a fitted fect model*

---

## Description

For each unit g with first-switch period F_g, "treat" the M pre-treatment periods immediately before F_g (i.e., periods F_g - 1, F_g - 2, ..., F_g - M) and refit the fect estimator on the remaining truly-pre-treatment cells. Effects at those placebo cells should be statistically indistinguishable from zero if the identifying assumption holds.

## Usage

```
didgpu_fect_placebo(
  df,
  outcome,
  group,
  time,
  treatment,
  method = c("fe", "ife", "mc"),
  n_placebos = 3L,
  r = 2L,
  lambda = NULL,
  tol = 1e-05,
  max_iter = 500L,
  bootstrap_reps = 0L,
  seed = 1L,
  backend = "auto"
)
```

## Arguments

| | |
|---|---|
| `df` | A panel data.frame (the same one used to fit). |
| `outcome`, `group`, `time`, `treatment` | Column names. |
| `method` | One of `"fe"`, `"ife"`, `"mc"`. |
| `n_placebos` | Integer. Number of pre-treatment placebos to test per unit. Default `3L`. Equivalent to the `placebo` option in the fect package. |

`r`, `lambda`, `tol`, `max_iter`
: Method-specific args forwarded to the underlying fit (see `didgpu_fect()`).

`bootstrap_reps`
: Integer. Number of bootstrap replicates for SE estimation. Default `0L` = no bootstrap, SEs are NA.

`seed`
: Integer. Bootstrap seed.

`backend`
: Backend for the underlying fits.

## Value

A data.frame with one row per placebo horizon: `horizon` (negative, -1 = period immediately before F_g), `estimate`, `se`, `p_value` (test of H0: estimate = 0), plus a `joint_p` attribute giving the joint p-value across all placebos.

## Examples

```
p <- didgpu_simulate_panel(n_units = 60L, n_periods = 12L,
                           tau_profile = c(0.5, 1.0), seed = 17L)
pl <- didgpu_fect_placebo(p, "Y", "unit", "period", "D",
                          method = "fe", n_placebos = 2L)
pl
```

---

# didgpu_freyaldenhoven *Freyaldenhoven-Hansen-Shapiro (2019) pre-event proxy event study*

---

## Description

Panel event-study estimator that uses an auxiliary covariate (a "proxy" affected by the confound but not the policy) to correct for confounding pre-trends. `estimator = "OLS"` is the plain two-way FE event study; `estimator = "FHS"` adds the proxy as an endogenous regressor and instruments it (2SLS) with a far policy lead, purging the confound. Reimplements `eventstudyr::EventStudy`'s first-difference parameterization, so the event-study coefficients match that reference.

## Usage

```
didgpu_freyaldenhoven(df, outcome, policy, id, time,
  estimator = c("OLS", "FHS"), proxy = NULL, proxyIV = NULL, pre = 0L,
  post = 1L, overidpre = pre + post, overidpost = 1L,
  normalize = -(pre + 1L), cluster = NULL, tol = 1e-10, max_iter = 1000L,
  verbose = TRUE)
```

## Arguments

`df`
: A data.frame / data.table panel.

`outcome`, `policy`, `id`, `time`
: Character column names: outcome, the (binary or continuous) policy variable, unit id, and integer time.

| | |
|---|---|
| estimator | "OLS" (default) or "FHS". |
| proxy | For "FHS", the character name of the proxy covariate. |
| proxyIV | For "FHS", the instrument column. Default: the first-differenced policy lead with the strongest first-stage F. |
| pre, post | Non-negative integers: anticipation leads (pre) and dynamic lags (post). |
| overidpre, overidpost | Extra leads/lags (over-identification / endpoints). Defaults mirror eventstudyr (overidpost = 1, overidpre = pre + post). |
| normalize | Event-time coefficient to omit (normalized to 0). Default -(pre + 1). |
| cluster | Optional unit-level cluster column for SEs. Defaults to id. |
| tol, max_iter | Two-way demeaning controls. |
| verbose | Logical. |

## Value

A didgpu_freyaldenhoven_result: a list with coefficients (a matrix: Estimate / SE / LB.CI / UB.CI per event-study term, plus the proxy for FHS), estimator, proxyIV, and args.

## References

Freyaldenhoven, S., Hansen, C. & Shapiro, J.M. (2019). Pre-event Trends in the Panel Event-Study Design. *American Economic Review* 109(9): 3307-3338.

## Examples

```
if (requireNamespace("eventstudyr", quietly = TRUE)) {
  d <- eventstudyr::example_data
  didgpu_freyaldenhoven(d, "y_base", "z", "id", "t",
                        estimator = "FHS", proxy = "x_r", pre = 0, post = 3)
}
```

---

# didgpu_glance *One-row glance summary of a didgpu_result*

---

## Description

Columns: n_effects, n_placebos, n_switchers, n_obs_effect_1, p_jointeffects, p_jointplacebo, n_boot, backend, seed.

## Usage

```
didgpu_glance(x, ...)
```

### Arguments

| | |
|---|---|
| x | A didgpu_result. |
| ... | Unused. |

### Value

A one-row data.frame.

### Examples

```
p <- didgpu_simulate_panel(n_units = 40L, n_periods = 10L, seed = 17L)
fit <- didgpu(p, "Y", "unit", "period", "D",
              effects = 2L, bootstrap_reps = 0L,
              backend = "r", verbose = FALSE)
didgpu_glance(fit)
```

---

## didgpu_has_cuda_support *Report whether didgpu was compiled with CUDA support*

---

### Description

Returns TRUE if the package was built with the NVIDIA CUDA Toolkit available (so src/cuda_*.cu was compiled and linked into the package DLL), FALSE otherwise. See didgpu_backend_info() for the user-facing way to inspect backend availability.

### Value

Logical.

---

## didgpu_honest_did *Sensitivity analysis for a fitted event-study DiD*

---

### Description

Given a fitted didgpu / didgpu_cs result with both pre- and post- treatment event-study coefficients, runs the Rambachan-Roth (2023) sensitivity analysis: bound the post-treatment estimate under a user-specified restriction on the magnitude of pre-trend violations, and report the "breakdown" parameter – the smallest violation that would flip the substantive conclusion.

## Usage

```
didgpu_honest_did(
  fit,
  event_post = 1L,
  method = c("RM", "M"),
  Mbar = c(0, 0.5, 1, 1.5, 2),
  alpha = 0.05,
  ci_level = 100 * (1 - alpha)
)
```

## Arguments

| | |
|---|---|
| fit | A didgpu_result (with placebos > 0) or didgpu_cs_result (which always includes pre-treatment placebos automatically). |
| event_post | Integer. Which post-treatment event-time to bound. Default 1L. |
| method | One of "M" (smoothness; bound consecutive-period violations) or "RM" (relative magnitudes; bound by Mbar times the largest pre-trend deviation). Default "RM". |
| Mbar | Numeric vector. Grid of restriction values to test. For "M", this is the smoothness bound in outcome units; for "RM", the multiplier on the largest pre-trend. Default c(0, 0.5, 1, 1.5, 2). |
| alpha | Numeric. Significance level. Default 0.05. |
| ci_level | Numeric. Same as (1 - alpha) * 100. Convenience. |

## Value

A didgpu_honest_did_result: data.frame with one row per Mbar value: Mbar, lb (lower bound of robust CI), ub, crosses_zero (TRUE if CI includes 0; the conclusion is fragile). Plus a $breakdown attribute giving the smallest Mbar at which the CI includes zero.

## References

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

## Examples

```
p <- didgpu_simulate_panel(n_units = 100L, n_periods = 12L,
                           tau_profile = c(0.5, 1.0),
                           seed = 17L)
fit <- didgpu_cs(p, "Y", "unit", "period", "D",
                 est_method = "OR", bootstrap_reps = 30L,
                 backend = "r", verbose = FALSE)
sens <- didgpu_honest_did(fit, event_post = 1L,
                          method = "RM",
                          Mbar = c(0, 0.5, 1.0))
print(sens)
```

## didgpu_init_checkpoint *Initialise a checkpoint directory*

### Description

Creates `checkpoint_dir`, writes `meta.json` recording the run configuration, and seeds an empty `manifest.csv`. Refuses to overwrite an existing manifest unless `force = TRUE`.

### Usage

```
didgpu_init_checkpoint(checkpoint_dir, meta, force = FALSE)
```

### Arguments

| | |
|---|---|
| `checkpoint_dir` | Path to checkpoint directory (created if missing). |
| `meta` | Named list. Must include at minimum: `panel_hash`, `seed`, `bootstrap_reps`, `effects`, `placebo`, `outcome`, `group`, `time`, `treatment`, `package_version`. Anything else is recorded as-is. |
| `force` | If TRUE, wipe any existing cells/ and manifest.csv before initialising. Default FALSE. |

### Value

Invisibly, the normalised `checkpoint_dir` path.

### Examples

```
cdir <- tempfile("init_demo_")
meta <- list(panel_hash = "abc123", seed = 1L, bootstrap_reps = 5L,
             effects = 2L, placebo = 0L, outcome = "Y", group = "g",
             time = "t", treatment = "D",
             package_version = "0.1.0")
didgpu_init_checkpoint(cdir, meta)
list.files(cdir)
unlink(cdir, recursive = TRUE)
```

## didgpu_joint_placebo *Windowed / subset joint test of pre-treatment placebos*

### Description

Runs the same chi-square joint Wald test as `fit$results$p_jointplacebo` but on a chosen subset of placebo horizons, using the bootstrap covariance `didgpu()` already stored. Useful when parallel trends only need to hold over a specific pre-treatment window.

## Usage

```
didgpu_joint_placebo(x, horizons = NULL)
```

## Arguments

| | |
|---|---|
| x | A didgpu_result from didgpu() run with placebo > 0 and bootstrap_reps > 0. |
| horizons | Integer vector of placebo horizons to include, as positive distances before treatment (1 = the period immediately pre-treatment, i.e. event time -1, matching Placebo_1). Negative values are accepted and used by magnitude. NULL (default) uses all placebo horizons, reproducing fit$results$p_jointplacebo. |

## Value

A didgpu_joint_placebo object: a list with statistic (chi-square), df, p_value, horizons (the included event times, negative), estimates (the included placebo point estimates), and n_boot.

## See Also

didgpu_equivalence() for an equivalence (TOST) framing of the same placebos.

## Examples

```
p <- didgpu_simulate_panel(n_units = 80L, n_periods = 12L,
                           tau_profile = c(0.5, 1.0), seed = 7L)
p$D <- as.integer(p$D >= 0.5)
fit <- didgpu(p, "Y", "unit", "period", "D", effects = 3L, placebo = 3L,
              bootstrap_reps = 200L, verbose = FALSE)
didgpu_joint_placebo(fit, horizons = 1:2)   # only the two nearest leads
```

---

# didgpu_lb_frac_affected

*Sharp lower bound on the fraction of always-takers affected*

---

## Description

Given a binary treatment D, a discrete mediator M, and an outcome Y, computes a sharp lower bound on the fraction of always-takers (units with M(1) = M(0)) whose outcome is moved by the treatment. The bound is identified from the total-variation distance between treated and control conditional distributions of Y given M.

## Usage

```
didgpu_lb_frac_affected(
  df,
  d,
  m,
  y,
  B = 500L,
  at_group = NULL,
  num_Ybins = 5L,
  cluster = NULL,
  reg_formula = NULL,
  alpha = 0.05,
  seed = 1L,
  backend = "auto"
)
```

## Arguments

| | |
|---|---|
| df | A data.frame. Cross-sectional; one row per observation. |
| d | Character. Column name of the binary treatment (0/1). |
| m | Character. Column name of the discrete mediator. |
| y | Character. Column name of the outcome (continuous outcomes are binned to num_Ybins quantile bins). |
| B | Number of bootstrap resamples for the CI. Default 500L. |
| at_group | Optional character: name of an always-taker subgroup column. NULL pools over all groups. |
| num_Ybins | Integer or NULL. For continuous y, the number of quantile bins to use. Default 5L. Ignored when y is already discrete. |
| cluster | Character or NULL. Column name for cluster bootstrap. |
| reg_formula | Optional one-sided formula for regression-adjusted partial densities. |
| alpha | Numeric. Test level. Default 0.05. |
| seed | Integer. RNG seed. |
| backend | One of "auto", "r", "cuda". "auto" picks "cuda" when available. |

## Value

A list with lb (lower bound), ci_low, ci_high, method = "TV".

## Examples

```
## Not run:
# NOT YET IMPLEMENTED. Planned API:
df <- data.frame(D = sample(0:1, 1000, TRUE),
                 M = sample(1:3, 1000, TRUE),
                 Y = rnorm(1000))
lb <- didgpu_lb_frac_affected(df, "D", "M", "Y", B = 500L)
```

```
lb$lb; lb$ci_low; lb$ci_high

## End(Not run)
```

---

didgpu_load_checkpoint *Load checkpoint metadata and manifest*

---

## Description

Load checkpoint metadata and manifest

## Usage

```
didgpu_load_checkpoint(checkpoint_dir)
```

## Arguments

| | |
|---|---|
| checkpoint_dir | Path to an existing checkpoint directory. |

## Value

A list with `meta` (parsed from meta.json), `manifest` (data.frame; possibly zero rows), and `checkpoint_dir` (normalised).

## Examples

```
p <- didgpu_simulate_panel(n_units = 30L, n_periods = 8L, seed = 1L)
cdir <- tempfile("load_demo_")
didgpu(p, "Y", "unit", "period", "D",
        effects = 1L, bootstrap_reps = 0L,
        checkpoint_dir = cdir, backend = "r", verbose = FALSE)
chk <- didgpu_load_checkpoint(cdir)
chk$meta$effects
nrow(chk$manifest)
unlink(cdir, recursive = TRUE)
```

didgpu_loo *Leave-one-out (LOO) robustness analysis*

## Description

For each entity (cohort / unit / cluster / level of a column), re-fits the estimator dropping that entity and reports the headline estimate. The headline depends on family:

- didgpu_result: the ATE.
- didgpu_cs_result: the requested aggregation's first row (or the overall value if aggregation = "overall").
- didgpu_fect_result: the ATE.

## Usage

```
didgpu_loo(fit, by = "cohort", df = NULL, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| fit | A fitted didgpu_result, didgpu_cs_result, or didgpu_fect_result. |
| by | Either "cohort" (drop each treatment cohort in turn), "unit" (drop each unit), "cluster" (drop each cluster as defined by the fit's cluster arg if set, otherwise group), or a character giving a column name to use for the drop key. |
| df | Optional original panel. If NULL, the function looks for an attached panel hash + path in the fit; if not found, you must supply df. Standard pattern: pass the same df you fit with. |
| verbose | Logical. Print one line per leave-out fit. |

## Details

Useful for detecting single-cohort or single-unit influence on headline estimates.

## Value

An object of class didgpu_loo_result: a data.frame with columns leave_out (the entity dropped), estimate (the headline under that drop), delta (estimate - full-sample estimate), delta_pct (delta as % of full-sample). Sorted by abs(delta) descending. Plus $full attribute (the full-sample estimate) and $by attribute.

## Examples

```
p <- didgpu_simulate_panel(n_units = 80L, n_periods = 12L,
                           tau_profile = c(0.5, 1.0), seed = 17L)
fit <- didgpu_cs(p, "Y", "unit", "period", "D",
                 est_method = "OR", aggregation = "overall",
                 bootstrap_reps = 0L, backend = "r", verbose = FALSE)
loo <- didgpu_loo(fit, by = "cohort", df = p, verbose = FALSE)
```

```
print(loo)
```

---

didgpu_resume *Quick resume helper*

---

## Description

Equivalent to calling `didgpu(...)` with the same arguments and `checkpoint_dir = checkpoint_dir, resume = TRUE`, but pulls the arguments from `meta.json` so the caller does not have to repeat them. The panel must still be supplied (panels are not stored on disk; only the panel hash, for integrity checks).

## Usage

```
didgpu_resume(checkpoint_dir, df, ...)
```

## Arguments

| | |
|---|---|
| `checkpoint_dir` | Path to existing checkpoint directory. |
| `df` | The same panel originally passed (validated by hash). |
| `...` | Optional overrides forwarded to `didgpu()`. |

## Details

All optional arguments may be overridden by passing them through `...`; the override takes precedence over the checkpointed value. Use this to (for example) bump `bootstrap_reps` higher or switch `backend` mid-run — but note that mismatches on identity-relevant fields (outcome, group, time, treatment, effects, placebo, seed) will be rejected by `.check_meta_compatibility()` inside `didgpu()`.

## Value

The aggregated result, same shape as `didgpu()`.

## Examples

```
p <- didgpu_simulate_panel(n_units = 40L, n_periods = 10L,
                           tau_profile = c(0.5, 1.0), seed = 17L)
cdir <- tempfile("didgpu_resume_demo_")
fit1 <- didgpu(p, "Y", "unit", "period", "D",
               effects = 2L, bootstrap_reps = 3L, seed = 1L,
               checkpoint_dir = cdir, backend = "r", verbose = FALSE)
# Resume with no extra work — produces the identical result.
fit2 <- didgpu_resume(cdir, df = p)
identical(coef(fit1), coef(fit2))
unlink(cdir, recursive = TRUE)
```

## didgpu_run_saxpy *Run a SAXPY (y = a*x + y) on the GPU*

### Description

Smoke-test for the CUDA toolchain. Computes y = a*x + y on the GPU and returns the result. Only available when the package was built with CUDA support; otherwise errors.

### Arguments

| | |
|---|---|
| a | Numeric scalar. |
| x, y | Numeric vectors of the same length. |

### Value

Updated y vector.

## didgpu_simulate_panel *Generate a simulated DiD panel with known event-time profile*

### Description

Generate a simulated DiD panel with known event-time profile

### Usage

```
didgpu_simulate_panel(
  n_units = 100L,
  n_periods = 20L,
  frac_treated = 0.5,
  min_treat_period = NULL,
  max_treat_period = NULL,
  tau_profile = c(0.2, 0.4, 0.6, 0.5, 0.4),
  sigma = 0.5,
  unit_fe_sd = 1,
  time_fe_sd = 0.3,
  seed = 1L,
  late_entry_frac = 0,
  max_entry_period = NULL
)
```

## Arguments

| | |
|---|---|
| n_units | integer. Number of units. |
| n_periods | integer. Number of time periods (1..n_periods). |
| frac_treated | numeric between 0 and 1. Share of units that ever get treated. The rest are never-treated controls. |
| min_treat_period | integer. Earliest treatment-on period. |
| max_treat_period | integer. Latest treatment-on period. Must be <= n_periods. Treated units' F_g is drawn uniformly between min_treat_period and max_treat_period. |
| tau_profile | numeric vector. Event-time effects: tau_profile[k+1] is the effect at event time k = 0, 1, .... Length length(tau_profile) - 1 upper bound on k. |
| sigma | numeric. Idiosyncratic noise SD. |
| unit_fe_sd | numeric. Unit fixed-effect SD. |
| time_fe_sd | numeric. Time fixed-effect SD. |
| seed | integer. RNG seed. |
| late_entry_frac | numeric in [0, 1]. Share of units that enter the panel after period 1, producing an UNBALANCED panel: their rows before entry are absent entirely (not NA). Default 0 (balanced), which leaves the RNG stream untouched so every existing seeded fixture stays bit-identical. |
| max_entry_period | integer. Latest entry period for late entrants, drawn uniformly from 2:max_entry_period. Defaults to max(2, floor(n_periods * 0.4)), which keeps most late entrants with a usable pre-period before they switch. |

## Value

A data.frame with columns: unit (int), period (int), D (binary treatment indicator), Y (outcome). Sorted by (unit, period). Also has an attribute "truth": a list with F_g (named numeric per unit; Inf = never-treated), tau_profile, unit_fe, time_fe, and entry_period (named integer per late-entering unit, or NULL when balanced).

## Examples

```
p <- didgpu_simulate_panel(n_units = 40L, n_periods = 10L,
                           tau_profile = c(0.5, 1.0, 1.2),
                           seed = 17L)
head(p)
# Inspect the underlying DGP:
truth <- attr(p, "truth")
table(truth$F_g)            # cohort sizes (Inf = never-treated)
truth$tau_profile
```

`didgpu_simulate_panel_bidir` *Generate a panel with BOTH switcher-in and switcher-out units*

## Description

Useful for testing the cross-direction Neyman pooling. The first `frac_in` of treated units start at d=0 and turn on (switcher-in); the remaining `1 - frac_in` start at d=1 and turn off (switcher-out). Never-treated units stay at d=0.

## Usage

```
didgpu_simulate_panel_bidir(
  n_units = 100L,
  n_periods = 20L,
  frac_treated = 0.6,
  frac_in = 0.5,
  min_treat_period = NULL,
  max_treat_period = NULL,
  tau_in = c(0.5, 1, 1.2),
  tau_out = c(-0.5, -0.8, -1),
  sigma = 0.4,
  unit_fe_sd = 1,
  time_fe_sd = 0.3,
  seed = 1L
)
```

## Arguments

| | |
|---|---|
| `n_units` | integer. Number of units. |
| `n_periods` | integer. Number of time periods. |
| `frac_treated` | numeric between 0 and 1. Share that ever switch. |
| `frac_in` | numeric between 0 and 1. Of switchers, share going in (rest go out). 0 = all out-switchers, 1 = all in-switchers. |
| `min_treat_period`, `max_treat_period` | Treatment window. |
| `tau_in`, `tau_out` | numeric vectors. Event-time profiles for in-switchers and out-switchers respectively. |
| `sigma`, `unit_fe_sd`, `time_fe_sd`, `seed` | As in `didgpu_simulate_panel`. |

## Value

Data.frame with (unit, period, D, Y) plus a `truth` attribute.

## didgpu_summarize_panel

*Summarize a panel before fitting*

### Description

Computes summary statistics that let you sanity-check a panel and know what the estimator can recover. Prints a human-readable summary and returns the underlying numbers invisibly.

### Usage

```
didgpu_summarize_panel(df, outcome, group, time, treatment, verbose = TRUE)
```

### Arguments

| | |
|---|---|
| df | A panel. |
| outcome, group, time, treatment | Column names. |
| verbose | Print the summary. Default TRUE. |

### Value

Invisibly, a list with n_units, n_periods, n_rows, is_balanced, d_sq_dist (table), n_never_change, n_switchers_in, n_switchers_out, f_g_dist (table), max_effects, max_placebo, na_outcome, na_treatment.

### Examples

```
p <- didgpu_simulate_panel(n_units = 40L, n_periods = 10L,
                           tau_profile = c(0.5, 1.0),
                           seed = 17L)
s <- didgpu_summarize_panel(p, "Y", "unit", "period", "D")
s$n_switchers_in
s$max_effects
```

## didgpu_test_sharp_null

*Sharp test of full mediation*

### Description

Tests whether a binary treatment D affects an outcome Y only through a discrete mediator M. The null hypothesis is sharp full mediation: Y(1, m) = Y(0, m) for every level m. Implemented via three moment-inequality testing procedures.

## Usage

```
didgpu_test_sharp_null(
  df,
  d,
  m,
  y,
  method = c("CS", "ARP", "FSST"),
  B = 500L,
  num_Ybins = 5L,
  cluster = NULL,
  reg_formula = NULL,
  alpha = 0.05,
  seed = 1L,
  backend = "auto"
)
```

## Arguments

| | |
|---|---|
| df | A data.frame. Cross-sectional; one row per observation. |
| d | Character. Column name of the binary treatment (0/1). |
| m | Character. Column name of the discrete mediator. |
| y | Character. Column name of the outcome (continuous outcomes are binned to num_Ybins quantile bins). |
| method | One of "CS" (Cox-Shi), "ARP" (Andrews-Roth-Pakes), or "FSST" (Fang-Santos-Shaikh-Torgovitsky). Default "CS". |
| B | Integer. Number of bootstrap resamples for variance estimation. Default 500L. |
| num_Ybins | Integer or NULL. For continuous y, the number of quantile bins to use. Default 5L. Ignored when y is already discrete. |
| cluster | Character or NULL. Column name for cluster bootstrap. |
| reg_formula | Optional one-sided formula for regression-adjusted partial densities. |
| alpha | Numeric. Test level. Default 0.05. |
| seed | Integer. RNG seed. |
| backend | One of "auto", "r", "cuda". "auto" picks "cuda" when available. |

## Value

A list with reject (logical), test_stat (numeric), cv (critical value), pval (numeric), method (character).

## References

Kwon, S. and Roth, J. (2026). "(Empirical) Bayes approaches to parallel trends." *Review of Economic Studies*, forthcoming.

## Examples

```
## Not run:
# NOT YET IMPLEMENTED. The example below shows the planned API.
df <- data.frame(D = sample(0:1, 1000, TRUE),
                 M = sample(1:3, 1000, TRUE),
                 Y = rnorm(1000))
res <- didgpu_test_sharp_null(df, "D", "M", "Y", method = "CS",
                              B = 500L, seed = 1L)
res$pval

## End(Not run)
```

---

# didgpu_tidy *Tidy a didgpu_result into a one-row-per-coefficient data.frame*

---

## Description

Return shape matches the broom convention: columns `term`, `estimate`, `std.error`, `statistic`, `p.value`, `conf.low`, `conf.high`. The `term` column distinguishes effects (`Effect_1`, `Effect_2`, ..., `ATE`) from placebos (`Placebo_1`, ...).

## Usage

```
didgpu_tidy(x, conf.int = TRUE, conf.level = NULL, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `didgpu_result` or `didgpu_cs_result` object. For a `didgpu_cs_result` the return has one row per entry of the `aggregation` table (`kind = "aggregate"`) followed by one row per ATT(g, t) cell (`kind = "att_gt"`). `.cs_aggregate()` propagates point estimates only, so `std.error` and the CI columns are `NA` on the aggregate rows; the cell rows carry the real SEs and CIs (themselves `NA` unless `bootstrap_reps > 0`). |
| `conf.int` | Logical. Include confidence interval columns. Default TRUE. |
| `conf.level` | Confidence level. Defaults to whatever the fit used. |
| `...` | Unused. |

## Value

A data.frame.

## Examples

```
p <- didgpu_simulate_panel(n_units = 40L, n_periods = 10L, seed = 17L)
fit <- didgpu(p, "Y", "unit", "period", "D",
              effects = 2L, placebo = 1L,
              bootstrap_reps = 0L, backend = "r", verbose = FALSE)
didgpu_tidy(fit)
```

didgpu_twfe *Naive two-way fixed-effects (TWFE) dynamic event study*

## Description

Fits the distributed-lag TWFE specification with unit and time fixed effects and cluster-robust standard errors. Intended as the "naive" baseline to report next to the bias-robust estimators `didgpu()` and `didgpu_cs()`.

## Usage

```
didgpu_twfe(df, outcome, group, time, treatment, effects = 1L,
  placebo = 0L, cluster = NULL, tol = 1e-10, max_iter = 1000L,
  verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `df` | A data.frame / data.table panel. |
| `outcome`, `group`, `time`, `treatment` | Character column names: the outcome, unit id, time id, and (binary, possibly non-absorbing) treatment indicator. |
| `effects` | Integer >= 1. Number of post/contemporaneous lag terms (Effect_1 = contemporaneous, Effect_k = lag k-1). |
| `placebo` | Integer >= 0. Number of lead terms (Placebo_j = lead j), the pre-trend / anticipation checks. |
| `cluster` | Optional character column name to cluster SEs on. Defaults to `group`. |
| `tol` | Convergence tolerance for the iterative two-way fixed-effect demeaning. Default `1e-10`. |
| `max_iter` | Max demeaning iterations. Default `1000`. |
| `verbose` | Logical. |

## Details

The fixed effects are absorbed by iterative two-way demeaning (cost scales with observations, not units), so the estimator stays fast on large panels where building unit/time dummy matrices would be impractical. Point estimates are bit-exact to `lm()` on the dummy specification; the `Effect_k` / `Placebo_j` output mirrors `didgpu()` so the bias-prone TWFE estimate sits next to the robust one. TWFE is biased under heterogeneous/dynamic effects; see `didgpu_bacon()` for the Goodman-Bacon decomposition of that bias.

## Value

An object of class `didgpu_twfe_result`: a list with `coef` (named vector), `results` (with `Effects` and `Placebos` matrices: Estimate / SE / LB.CI / UB.CI / N), and `args`.

## See Also

didgpu() and didgpu_cs() for the heterogeneity-robust estimators; didgpu_bacon() for the bias decomposition.

## Examples

```
p <- didgpu_simulate_panel(n_units = 60L, n_periods = 10L,
                           tau_profile = c(0.5, 1.0), seed = 17L)
fit <- didgpu_twfe(p, "Y", "unit", "period", "D",
                   effects = 3L, placebo = 2L, verbose = FALSE)
print(fit)
```

---

# glance.didgpu_result *broom::glance method for didgpu_result*

---

## Description

broom::glance method for didgpu_result

## Usage

```
glance.didgpu_result(x, ...)
```

## Arguments

| | |
|---|---|
| x | A didgpu_result. |
| ... | Unused. |

## Value

A one-row data.frame.

---

# plot.didgpu_loo_result *Plot method for didgpu_loo_result (tornado plot)*

---

## Description

Plot method for didgpu_loo_result (tornado plot)

## Usage

```
## S3 method for class 'didgpu_loo_result'
plot(x, n_show = 20L, ...)
```

### Arguments

| | |
|---|---|
| x | A didgpu_loo_result. |
| n_show | Integer. Number of top rows to show. Default 20L. |
| ... | Extra args for plot(). |

### Value

The input invisibly.

---

## plot.didgpu_result *Event-study plot of a didgpu result*

---

### Description

Plots estimates against event-time horizon: pre-treatment placebos at negative horizons, post-treatment effects at positive horizons, with the stored CIs as vertical error bars and a horizontal dashed line at 0 for reference. Uses base R graphics — no ggplot2 dependency. Returns the input invisibly so calls can be chained.

### Usage

```
## S3 method for class 'didgpu_result'
plot(x, ..., show_zero_line = TRUE, show_zero_horizon = TRUE, ci = TRUE)
```

### Arguments

| | |
|---|---|
| x | A didgpu_result object. |
| ... | Extra graphical parameters passed to the underlying plot() call (e.g. main, xlab, ylab, col, pch, lwd, xlim, ylim). |
| show_zero_line | Logical. Draw a dashed line at y = 0 (default TRUE). |
| show_zero_horizon | Logical. Mark the boundary between placebo and effect horizons with a vertical dashed line (default TRUE). |
| ci | Logical. Draw error bars at the stored CI level (default TRUE; suppressed when bootstrap_reps = 0 because the CIs are NA). |

### Value

The input invisibly.

## print.didgpu_bacon *Print method for didgpu_bacon*

### Description

Print method for didgpu_bacon

### Usage

```
## S3 method for class 'didgpu_bacon'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A didgpu_bacon object from didgpu_bacon(). |
| ... | Unused. |

### Value

x, invisibly.

## print.didgpu_by_result *Print method for didgpu_by_result*

### Description

Shows a compact table of per-subgroup point estimates.

### Usage

```
## S3 method for class 'didgpu_by_result'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A didgpu_by_result object. |
| ... | Unused. |

### Value

The input invisibly.

---

`print.didgpu_cs_continuous_result` *Print method for didgpu_cs_continuous_result*

---

## Description

Print method for didgpu_cs_continuous_result

## Usage

```
## S3 method for class 'didgpu_cs_continuous_result'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `didgpu_cs_continuous_result`. |
| `...` | Unused. |

## Value

`x`, invisibly.

---

`print.didgpu_cs_result` *Print method for didgpu_cs_result*

---

## Description

Print method for didgpu_cs_result

## Usage

```
## S3 method for class 'didgpu_cs_result'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `didgpu_cs_result`. |
| `...` | Unused. |

## Value

The input invisibly.

## print.didgpu_did_continuous_result

*Print method for didgpu_did_continuous_result*

### Description

Print method for didgpu_did_continuous_result

### Usage

```
## S3 method for class 'didgpu_did_continuous_result'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A didgpu_did_continuous_result. |
| ... | Unused. |

### Value

x, invisibly.

## print.didgpu_did_static_result

*Print method for didgpu_did_static_result*

### Description

Print method for didgpu_did_static_result

### Usage

```
## S3 method for class 'didgpu_did_static_result'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A didgpu_did_static_result. |
| ... | Unused. |

### Value

x, invisibly.

---

`print.didgpu_equivalence` *Print method for didgpu_equivalence*

---

## Description

Print method for didgpu_equivalence

## Usage

```
## S3 method for class 'didgpu_equivalence'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `didgpu_equivalence` object from `didgpu_equivalence()`. |
| `...` | Unused. |

## Value

`x`, invisibly.

---

`print.didgpu_fect_placebo` *Print method for didgpu_fect_placebo*

---

## Description

Print method for didgpu_fect_placebo

## Usage

```
## S3 method for class 'didgpu_fect_placebo'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A placebo-test result. |
| `...` | Unused. |

## Value

The input invisibly.

## print.didgpu_freyaldenhoven_result *Print method for didgpu_freyaldenhoven_result*

### Description

Print method for didgpu_freyaldenhoven_result

### Usage

```
## S3 method for class 'didgpu_freyaldenhoven_result'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A didgpu_freyaldenhoven_result. |
| ... | Unused. |

### Value

x, invisibly.

## print.didgpu_honest_did_result *Print method for didgpu_honest_did_result*

### Description

Print method for didgpu_honest_did_result

### Usage

```
## S3 method for class 'didgpu_honest_did_result'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A didgpu_honest_did_result. |
| ... | Unused. |

### Value

The input invisibly.

## print.didgpu_joint_placebo

*Print method for didgpu_joint_placebo*

### Description

Print method for didgpu_joint_placebo

### Usage

```
## S3 method for class 'didgpu_joint_placebo'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A didgpu_joint_placebo object from didgpu_joint_placebo(). |
| ... | Unused. |

### Value

x, invisibly.

## print.didgpu_loo_result

*Print method for didgpu_loo_result*

### Description

Print method for didgpu_loo_result

### Usage

```
## S3 method for class 'didgpu_loo_result'
print(x, n = 10L, ...)
```

### Arguments

| | |
|---|---|
| x | A didgpu_loo_result. |
| n | Integer. Print top-n most-influential rows. Default 10L. |
| ... | Unused. |

### Value

The input invisibly.

## print.didgpu_result *Print method for didgpu_result*

### Description

Prints the estimation tables exactly as `DIDmultiplegtDYN::did_multiplegt_dyn()` prints its own: the event-study effects and their joint test, the average total effect, and the placebos and their joint test.

### Usage

```
## S3 method for class 'didgpu_result'
print(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | A `didgpu_result` object. |
| `...` | Unused (for S3 method compatibility). |

### Value

The input invisibly.

## print.didgpu_twfe_result *Print method for didgpu_twfe_result*

### Description

Print method for didgpu_twfe_result

### Usage

```
## S3 method for class 'didgpu_twfe_result'
print(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | A `didgpu_twfe_result`. |
| `...` | Unused. |

### Value

`x`, invisibly.

## summary.didgpu_result *Summary method for didgpu_result*

### Description

The same display as `print.didgpu_result()`, as in DIDmultiplegtDYN.

### Usage

```
## S3 method for class 'didgpu_result'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | A `didgpu_result` object. |
| `...` | Unused (for S3 method compatibility). |

### Value

The input invisibly.

## tidy.didgpu_result *broom::tidy method for didgpu_result*

### Description

broom::tidy method for didgpu_result

### Usage

```
tidy.didgpu_result(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | A `didgpu_result`. |
| `...` | Passed to `didgpu_tidy()`. |

### Value

A data.frame.

`vcov.didgpu_result` *Variance-covariance matrix of estimates*

## Description

Returns the covariance matrix of `(Effects, Placebos)`, computed at fit time and stored on the result. It is analytic – built from the same influence functions as the reported SEs, so its diagonal equals `SE^2` exactly and it needs no bootstrap. Where analytic SEs are not available (`controls`, `continuous`, `trends_lin`) it is the empirical covariance of the bootstrap replicates instead, and a square NA matrix if `bootstrap_reps` is 0 or 1.

## Usage

```
## S3 method for class 'didgpu_result'
vcov(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | A `didgpu_result` object. |
| `...` | Unused. |

## Details

The ordering matches `coef(object)`'s default (effects first, then placebos). The ATE row/column is NOT included — it is a linear combination of the per-event-time effects, so its variance can be recovered as `t(w) %*% vcov(object) %*% w` where `w` is the incidence-weighted vector.

## Value

A square matrix.

# Index


coef.didgpu_result, 4
confint.didgpu_result, 4

didgpu, 5
didgpu(), *11–14*, *24–26*, *31*, *38, 39*, *50, 51*
didgpu-package, 3
didgpu_aggregate_cells, 10
didgpu_backend_info, 11
didgpu_backend_info(), *36*
didgpu_bacon, 11
didgpu_bacon(), *25*, *50, 51*, *53*
didgpu_bootstrap_more, 12
didgpu_by, 13
didgpu_by_path, 15
didgpu_compare, 16
didgpu_compute_paths, 17
didgpu_cpu_hello, 18
didgpu_cs, 19
didgpu_cs(), *11, 12*, *23*, *50, 51*
didgpu_cs_aggregate, 21
didgpu_cs_aggregate(), *19*
didgpu_cs_continuous, 22
didgpu_cs_continuous(), *24*
didgpu_did_continuous, 23
didgpu_did_static, 25
didgpu_equivalence, 26
didgpu_equivalence(), *39*, *56*
didgpu_estimate_runtime, 27
didgpu_event_study_data, 29
didgpu_fect, 30
didgpu_fect(), *34*
didgpu_fect_equivalence, 32
didgpu_fect_equivalence(), *26*
didgpu_fect_placebo, 33
didgpu_fect_placebo(), *32*
didgpu_freyaldenhoven, 34
didgpu_glance, 35
didgpu_has_cuda_support, 36
didgpu_honest_did, 36
didgpu_honest_did(), *26*
didgpu_init_checkpoint, 38
didgpu_joint_placebo, 38
didgpu_joint_placebo(), *58*
didgpu_lb_frac_affected, 39
didgpu_load_checkpoint, 41
didgpu_loo, 42
didgpu_resume, 43
didgpu_run_saxpy, 44
didgpu_simulate_panel, 44
didgpu_simulate_panel_bidir, 46
didgpu_summarize_panel, 47
didgpu_test_sharp_null, 47
didgpu_tidy, 49
didgpu_tidy(), *60*
didgpu_twfe, 50
didgpu_twfe(), *11, 12*, *25*

glance.didgpu_result, 51

plot.didgpu_loo_result, 51
plot.didgpu_result, 52
print.didgpu_bacon, 53
print.didgpu_by_result, 53
print.didgpu_cs_continuous_result, 54
print.didgpu_cs_result, 54
print.didgpu_did_continuous_result, 55
print.didgpu_did_static_result, 55
print.didgpu_equivalence, 56
print.didgpu_fect_placebo, 56
print.didgpu_freyaldenhoven_result, 57
print.didgpu_honest_did_result, 57
print.didgpu_joint_placebo, 58
print.didgpu_loo_result, 58
print.didgpu_result, 59
print.didgpu_result(), *9*, *60*
print.didgpu_twfe_result, 59

summary.didgpu_result, 60

tidy.didgpu_result, 60

vcov.didgpu_result, 61