Package 'shapleydecomp'

November 3, 2026

Title Shapley Value Decomposition for Discrete-Time Simulation Models

Version 1.0.0

Description Attributes changes in the outcomes of a discrete-time simulation model to its input factors using Shapley values (Shapley 1953) <doi:10.1515/9781400881970-018>, which for binary factors coincide with the symmetric standardization of Das Gupta (1993) <https://www2.census.gov/library/publications/1993/demographics/p23-186.pdf>. Any calibrated model can be decomposed by implementing four S3 methods that run the model, summarize its output, and construct counterfactual inputs. Counterfactuals either remove a factor entirely or hold it at its value in a baseline year, and a sequential decomposition attributes each year's policy conditional on the past.

License MIT + file LICENSE

URL https://github.com/aucarter/shapleydecomp, https://aucarter.github.io/shapleydecomp/

BugReports https://github.com/aucarter/shapleydecomp/issues

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.0.0)

Imports data.table, parallel, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Austin Carter [aut, cre]

Maintainer Austin Carter <aucarter@uw.edu>

Repository CRAN

Date/Publication 2026-11-03 16:00:02 UTC

Contents

build_counterfactual . . . 2
cf_types . . . 3
decomp_algebra . . . 4
decomp_pair . . . 4
get_baseline_value . . . 5
make_run_table . . . 6
prepare_decomp . . . 6
run_decomp . . . 7
run_simulation . . . 8
sequential_totals . . . 9
shapley_decompose . . . 9
summarize_results . . . 12
validate_model_interface . . . 12
warm_start . . . 13

Index **16**

`build_counterfactual` *Construct counterfactual parameters for a factor*

Description

Given the observed parameters, produce a modified parameter set where the specified factor is set to its counterfactual value from `decomp_year` onward.

Usage

```
build_counterfactual(
  model,
  params,
  factor_name,
  decomp_year,
  cf_type,
  baseline_values,
  ...
)
```

Arguments

`model`	An S3 model object
`params`	Observed model parameters
`factor_name`	Character, name of the factor to modify
`decomp_year`	Integer, the year from which the counterfactual applies
`cf_type`	Character, one of `CF_CHANGE_SINCE` or `CF_TOTAL_IMPACT`

baseline_values
: Named list of baseline values for each factor (pre-computed by get_baseline_value())

...
: Additional arguments

Value

Modified model parameters with the factor set to its counterfactual value.

See Also

Other model interface: get_baseline_value(), prepare_decomp(), run_simulation(), summarize_results(), validate_model_interface(), warm_start

---

cf_types *Counterfactual type constants*

Description

Two types of counterfactual analysis are supported:

- CF_CHANGE_SINCE: The counterfactual value is the factor's value at the baseline year (typically the first decomposition year). This measures the impact of changes in the factor *since* that year.
- CF_TOTAL_IMPACT: The counterfactual value is zero (or the model's notion of "no intervention"). This measures the *total* impact of the factor, not just changes since a reference point.

Usage

```
CF_CHANGE_SINCE

CF_TOTAL_IMPACT
```

Format

An object of class character of length 1.

An object of class character of length 1.

Value

A length-one character string: "change_since" or "total_impact", the values accepted by the cf_type argument of shapley_decompose.

Examples

```
CF_CHANGE_SINCE
CF_TOTAL_IMPACT
```

decomp_algebra *Decompose all summary variables for a single group*

Description

Decompose all summary variables for a single group

Usage

```
decomp_algebra(data, params)
```

Arguments

data	A data.table for a single group × decomp-year with one row per on/off combination of the factors, in any order.
params	Parameter list (see run_decomp).

Value

A data.table with one row per summary variable: a measure column and one column per factor.

Examples

```
# Outcomes for every on/off combination of two factors, where the
# factors interact: y = 10 + 5a + 3b + 2ab
rt <- make_run_table(c("a", "b"))
rt[, y := 10 + 5 * a + 3 * b + 2 * a * b]
# The interaction term is split evenly: a gets 5 + 1, b gets 3 + 1
decomp_algebra(rt, list(summary_vars = "y", decomp_vars = c("a", "b"),
                        decomp_method = "shapley"))
```

decomp_pair *Decompose a single summary variable using Shapley value weights*

Description

For each factor, computes the weighted average marginal effect of switching the factor from 0 (counterfactual) to 1 (observed), averaging over all combinations of the other factors.

Usage

```
decomp_pair(dt, params, summary_var)
```

Arguments

dt	A data.table with one row per on/off combination of the factors, in any order.
params	Parameter list with decomp_vars and decomp_method.
summary_var	Character, the name of the summary variable column.

Value

A one-row data.table with a measure column and one column per factor.

Examples

```
rt <- make_run_table(c("a", "b"))
rt[, y := 10 + 5 * a + 3 * b + 2 * a * b]
decomp_pair(rt, list(decomp_vars = c("a", "b"), decomp_method = "shapley"),
            "y")
```

---

get_baseline_value *Get the baseline value for a factor*

Description

Returns the value of a factor at the baseline year (for CF_CHANGE_SINCE) or the zero/null value (for CF_TOTAL_IMPACT). Called once per factor before the decomposition loop begins.

Usage

```
get_baseline_value(model, params, factor_name, cf_type, baseline_year, ...)
```

Arguments

model	An S3 model object
params	Observed model parameters
factor_name	Character, name of the factor
cf_type	Character, CF_CHANGE_SINCE or CF_TOTAL_IMPACT
baseline_year	Integer, the reference year for CF_CHANGE_SINCE
...	Additional arguments

Value

The baseline value for the factor (format is model-specific).

See Also

Other model interface: build_counterfactual(), prepare_decomp(), run_simulation(), summarize_results(), validate_model_interface(), warm_start

make_run_table *Generate the factorial run table for decomposition*

Description

Creates a `data.table` with one row per counterfactual combination. Each factor column contains 0 (counterfactual) or 1 (observed), encoding whether that factor is "on" or "off" for that run.

Usage

```
make_run_table(factors)
```

Arguments

`factors`	Character vector of factor names.

Value

A `data.table` with 2^n rows: `run_id` (integer), and one column per factor with values 0 or 1.

Examples

```
make_run_table(c("treatment", "prevention", "demography"))
```

prepare_decomp *Prepare the model for decomposition*

Description

One-time preparation step called before the decomposition loop. Use this for expensive setup operations like running an initial simulation to extract denominators, converting counts to percentages, etc.

Usage

```
prepare_decomp(model, params, ...)
```

Arguments

`model`	An S3 model object
`params`	Model parameters
`...`	Additional arguments

Details

The default method returns the model and params unchanged.

Value

A list with components:

`model`	Possibly modified model object
`params`	Possibly modified parameters
`full_sim`	Optional: output of a full (observed) simulation, used for warm starts and as the "all-on" reference
`full_summary`	Optional: summary of the full simulation

See Also

Other model interface: `build_counterfactual()`, `get_baseline_value()`, `run_simulation()`, `summarize_results()`, `validate_model_interface()`, `warm_start`

`run_decomp` *Run decomposition across groups and decomp years*

Description

Applies the Shapley decomposition to each group × decomp-year in the data.

Usage

```
run_decomp(data, params)
```

Arguments

`data`	A `data.table` with one column per decomp variable (0/1 indicator), one column per summary variable, and optionally a `decomp_year` column and the `group_vars` columns. Each group must have exactly one row per on/off combination; row order does not matter.
`params`	A list with at least: **summary_vars** Character vector of summary variable column names **decomp_vars** Character vector of factor column names (0/1 indicators) **decomp_method** Character, `"shapley"` (`"das_gupta"` accepted as a deprecated alias) **group_vars** Optional character vector of grouping columns, such as a location identifier

Value

A `data.table` with one row per group × decomp-year × measure, and one column per factor giving its decomposition effect.

Examples

```
# Outcomes for every on/off combination of two factors, where the
# factors interact: y = 10 + 5a + 3b + 2ab
rt <- make_run_table(c("a", "b"))
rt[, y := 10 + 5 * a + 3 * b + 2 * a * b]
rt[, decomp_year := 2000L]
run_decomp(rt, list(summary_vars = "y", decomp_vars = c("a", "b"),
                    decomp_method = "shapley"))
```

`run_simulation` *Run a simulation with the given parameters*

Description

Execute the discrete-time simulation model and return raw simulation output. This is called once for each counterfactual combination at each decomposition year.

Usage

```
run_simulation(model, params, ...)
```

Arguments

`model`	An S3 model object (class determines dispatch)
`params`	Model parameters (possibly modified for counterfactual)
`...`	Additional arguments passed to the method

Value

Raw simulation output (model-specific format). Will be passed to `summarize_results()`.

See Also

Other model interface: `build_counterfactual()`, `get_baseline_value()`, `prepare_decomp()`, `summarize_results()`, `validate_model_interface()`, `warm_start`

sequential_totals *Total impact per factor at every year from a sequential decomposition*

Description

Sums the step attributions of a `shapley_decompose()` run across decomposition steps: the total impact of factor i at year T is $\sum_{y \leq T} \phi_i^{(y)}(T)$, where $\phi_i^{(y)}(T)$ is the Shapley value at year T of the year-y policy (`extended_decomp`). Summed over factors this telescopes to (observed) - (all policies off from the first decomp year) at every T, so it is only meaningful when `decomp_years` is contiguous.

Usage

```
sequential_totals(result)
```

Arguments

`result`	The list returned by `shapley_decompose()` with `extended = TRUE`.

Value

data.table with one row per group x year x measure and one column per factor.

See Also

`shapley_decompose`, whose examples call this function.

shapley_decompose *Run a full decomposition analysis*

Description

This is the main user-facing function. Given a model object that implements the shapleydecomp S3 interface, it performs a complete factorial decomposition of the specified factors across all decomposition years.

Usage

```
shapley_decompose(
  model,
  params,
  factors,
  summary_vars,
  decomp_years,
  method = "shapley",
  cf_type = CF_CHANGE_SINCE,
```

```
  baseline_year = min(decomp_years),
  extended = TRUE,
  step_scenarios = TRUE,
  verbose = TRUE,
  n_cores = 1L,
  group_vars = character(),
  ...
)
```

Arguments

model	An S3 model object. Must have methods for `run_simulation()`, `summarize_results()`, `build_counterfactual()`, and `get_baseline_value()`.
params	Model parameters (model-specific list/object).
factors	Character vector of factor names to decompose.
summary_vars	Character vector of summary variable names (columns in the output of `summarize_results()`).
decomp_years	Integer vector of years at which to perform decomposition (e.g., `2000:2023`).
method	Character, decomposition method. Only `"shapley"` (default) is supported. `"das_gupta"` is accepted as a deprecated alias: Das Gupta's symmetric standardization is the special case of the Shapley value for binary on/off factors, so the weights are identical.
cf_type	Character, counterfactual type: `"change_since"` (default) — counterfactual is the value at the baseline year; or `"total_impact"` — counterfactual is zero / no intervention.
baseline_year	Integer, the reference year for `"change_since"` counterfactuals. Defaults to `min(decomp_years)`.
extended	Logical, whether to compute the extended factor decomposition (effects at all output years >= decomp year). Default TRUE.
step_scenarios	Logical, whether to compute step scenarios (all-on time series per decomp year). Default TRUE.
verbose	Logical, print progress messages. Default TRUE.
n_cores	Integer, number of cores for parallel counterfactual simulation runs. Default 1 (sequential). Values > 1 run every counterfactual simulation, across all decomp years, in one `parallel::mclapply()` map, so that more cores than the 2^n runs of a single year can be used. Windows does not support forking, so there it falls back to sequential with a warning.
group_vars	Character vector of columns in the output of `summarize_results()` that identify independent groups, such as locations or draws, each decomposed separately. Default none.
...	Additional arguments passed to model methods.

Details

The decomposition is sequential (Carter 2025, *Quantifying averted burden from infectious disease control policies*, PhD dissertation, University of Washington, https://hdl.handle.net/1773/

53310). At each decomp year y, every policy before y is at its observed value, every policy after y is at its counterfactual, and only the year-y policy varies across the 2^n coalitions. The attribution at step y is therefore the marginal impact of the year-y policy, conditional on the past and undiluted by the future. `extended_decomp` records that impact at every later year, and `sequential_totals()` sums the steps into each factor's total impact at every year.

Value

A list with components:

decomp data.table — standard year-by-year decomposition (one row per decomp year × measure, factor effects in columns)

step_scenarios data.table — all-on time series per decomp year (columns: step_year, year, plus summary vars). NULL if `step_scenarios = FALSE`.

extended_decomp data.table — extended decomposition (factor effects at all years >= decomp year). NULL if `extended = FALSE`.

run_table data.table — the factorial design table used.

run_summary data.table — per decomp year: the number of coalition runs, how many were simulated, how many were reused from the previous step, and the factors that were null players (their counterfactual left the params unchanged, so their effect is 0).

params List of decomposition parameters used.

See Also

`vignette("shapleydecomp")` for a worked example, and `validate_model_interface` to check a model first.

Examples

```
# A two-factor SIR model shipped with the package
sir <- new.env()
sys.source(system.file("examples", "sir_model.R", package = "shapleydecomp"),
           envir = sir)
model  <- sir$create_sir_model(years = 1990:2010)
params <- sir$create_sir_params(years = 1990:2010)

res <- shapley_decompose(
  model, params,
  factors      = c("beta_reduction", "treatment_effect"),
  summary_vars = c("infections", "deaths"),
  decomp_years = 2000:2010,
  cf_type      = "total_impact",
  verbose      = FALSE
)
res$decomp

# Each factor's total impact at every year
sequential_totals(res)
```

summarize_results *Summarize simulation output into a data.table*

Description

Extract summary measures (e.g., deaths, infections) from raw simulation output. Must return a `data.table` with at minimum a `year` column and one column per summary variable specified in the decomposition.

Usage

```
summarize_results(model, sim_output, ...)
```

Arguments

`model`	An S3 model object
`sim_output`	Raw simulation output from `run_simulation()`
`...`	Additional arguments

Value

A `data.table` with columns: `year`, plus one column per summary variable (e.g., `deaths`, `infections`).

See Also

Other model interface: `build_counterfactual()`, `get_baseline_value()`, `prepare_decomp()`, `run_simulation()`, `validate_model_interface()`, `warm_start`

validate_model_interface *Validate that a model implements the shapleydecomp interface*

Description

Checks that the required S3 methods exist for the given model class and runs a minimal smoke test if parameters and factors are provided.

Usage

```
validate_model_interface(
  model,
  params = NULL,
  factors = NULL,
  summary_vars = NULL,
  baseline_year = NULL,
  verbose = TRUE
)
```

Arguments

model	An S3 model object
params	Optional: model parameters for smoke test
factors	Optional: character vector of factor names
summary_vars	Optional: character vector of summary variable names
baseline_year	Optional: integer baseline year
verbose	Logical, print diagnostic messages

Value

Invisible TRUE if all checks pass; stops with an error otherwise.

See Also

Other model interface: build_counterfactual(), get_baseline_value(), prepare_decomp(), run_simulation(), summarize_results(), warm_start

Examples

```
sir <- new.env()
sys.source(system.file("examples", "sir_model.R", package = "shapleydecomp"),
           envir = sir)
validate_model_interface(
  sir$create_sir_model(),
  sir$create_sir_params(),
  factors       = c("beta_reduction", "treatment_effect"),
  summary_vars  = c("infections", "deaths"),
  baseline_year = 2000
)
```

---

warm_start *Warm starts*

Description

Optional methods that let shapley_decompose() skip re-simulating the years before each decomposition year. Up to year $y - 1$ every coalition run is identical to the observed run, so a model that can restart from saved state only needs to simulate from y onward.

Usage

```
supports_warm_start(model, ...)

get_warm_start_state(model, sim_output, year, ...)

run_from_warm_start(model, warm_state, params, output_years, ...)
```

Arguments

model	An S3 model object
...	Additional arguments. shapley_decompose() passes warm_start_year to run_from_warm_start() here.
sim_output	Observed simulation output from run_simulation()
year	Integer, the year whose end-of-year state to extract
warm_state	Observed simulation output to restart from (the full_sim element returned by prepare_decomp())
params	Model parameters (possibly counterfactual)
output_years	Integer vector of years the output must cover

Details

When supports_warm_start() returns TRUE, shapley_decompose() calls run_from_warm_start(model, warm_state, params, output_years,warm_start_year = y - 1, ...) for every counterfactual run instead of run_simulation(). warm_state is the observed simulation output (the full_sim returned by prepare_decomp()). run_from_warm_start() extracts the state at the end of warm_start_year from it, typically by calling get_warm_start_state(model, warm_state, warm_start_year), and simulates forward with the counterfactual params. shapley_decompose() itself never calls get_warm_start_state(). It is the conventional helper for that extraction step.

The default supports_warm_start() returns FALSE. The default get_warm_start_state() and run_from_warm_start() methods throw an error.

Value

supports_warm_start(): logical. get_warm_start_state(): a model state object in a model-specific format. run_from_warm_start(): raw simulation output in the same format as run_simulation(), covering at least output_years.

See Also

Other model interface: build_counterfactual(), get_baseline_value(), prepare_decomp(), run_simulation(), summarize_results(), validate_model_interface()

Examples

```
# A stock that decays by 20% a year and receives input x:
# s_t = 0.8 * s_{t-1} + x_t
run_simulation.stock_model <- function(model, params, ...) {
  s <- as.numeric(stats::filter(params$x, 0.8, method = "recursive"))
  data.table::data.table(year = params$years, stock = s)
}
summarize_results.stock_model <- function(model, sim_output, ...) sim_output
get_baseline_value.stock_model <- function(model, params, factor_name,
                                           cf_type, baseline_year, ...) {
  list(type = "zero")
}
build_counterfactual.stock_model <- function(model, params, factor_name,
```

```
                                            decomp_year, cf_type,
                                            baseline_values, ...) {
  params$x[params$years >= decomp_year] <- 0
  params
}

# Warm start: restart from the observed stock at the end of
# warm_start_year and simulate only the remaining years
supports_warm_start.stock_model <- function(model, ...) TRUE
get_warm_start_state.stock_model <- function(model, sim_output, year, ...) {
  s <- sim_output$stock[sim_output$year == year]
  # Before the first simulated year there is no stock yet
  if (length(s) == 0) 0 else s
}
run_from_warm_start.stock_model <- function(model, warm_state, params,
                                            output_years, warm_start_year,
                                            ...) {
  s0   <- get_warm_start_state(model, warm_state, warm_start_year)
  keep <- params$years > warm_start_year
  s <- as.numeric(stats::filter(params$x[keep], 0.8, method = "recursive",
                                init = s0))
  data.table::data.table(year = params$years[keep], stock = s)
}

for (g in c("run_simulation", "summarize_results", "get_baseline_value",
            "build_counterfactual", "supports_warm_start",
            "get_warm_start_state", "run_from_warm_start")) {
  .S3method(g, "stock_model", get(paste0(g, ".stock_model")))
}

params <- list(years = 2000:2010, x = rep(1, 11))
warm <- shapley_decompose(structure(list(), class = "stock_model"), params,
                          factors = "x", summary_vars = "stock",
                          decomp_years = 2000:2010,
                          cf_type = "total_impact", verbose = FALSE)

# The same model without warm starts gives the same answer
.S3method("supports_warm_start", "stock_cold", function(model, ...) FALSE)
cold <- shapley_decompose(structure(list(), class = c("stock_cold",
                                                      "stock_model")),
                          params, factors = "x", summary_vars = "stock",
                          decomp_years = 2000:2010,
                          cf_type = "total_impact", verbose = FALSE)
all.equal(warm$decomp, cold$decomp)
```

Index

∗ **datasets**
    cf_types, 3
∗ **model interface**
    build_counterfactual, 2
    get_baseline_value, 5
    prepare_decomp, 6
    run_simulation, 8
    summarize_results, 12
    validate_model_interface, 12
    warm_start, 13

build_counterfactual, 2, *5*, *7, 8*, *12–14*

CF_CHANGE_SINCE *(*cf_types*)*, 3
CF_TOTAL_IMPACT *(*cf_types*)*, 3
cf_types, 3

decomp_algebra, 4
decomp_pair, 4

get_baseline_value, *3*, 5, *7, 8*, *12–14*
get_warm_start_state *(*warm_start*)*, 13

make_run_table, 6

prepare_decomp, *3*, *5*, 6, *8*, *12–14*

run_decomp, 7
run_from_warm_start *(*warm_start*)*, 13
run_simulation, *3*, *5*, *7*, 8, *12–14*

sequential_totals, 9
shapley_decompose, *3*, *9*, 9
summarize_results, *3*, *5*, *7, 8*, 12, *13, 14*
supports_warm_start *(*warm_start*)*, 13

validate_model_interface, *3*, *5*, *7, 8*, *11, 12*, 12, *14*

warm_start, *3*, *5*, *7, 8*, *12, 13*, 13