Package 'DAGassist'

November 12, 2026

Title Align Regressions with Target Estimands

Version 0.3.1

Maintainer Graham Goff <goffgrahamc@gmail.com>

Description Provides tools to align estimands with the identification that they require. Given a 'dagitty' object and a model specification, 'DAGassist' classifies variables by causal roles, recovers a target estimand, and generates a report comparing the original model with DAG-derived adjustment sets. Supports exporting publication-grade reports in 'LaTeX', 'Word', 'Excel', 'dotwhisker', or plain text/'markdown'. 'DAGassist' is built on 'dagitty', an 'R' package that uses the 'DAGitty' web tool (<https://dagitty.net/>) for creating and analyzing DAGs. Methods draw on Pearl (2009) <doi:10.1017/CBO9780511803161> and Textor et al. (2016) <doi:10.1093/ije/dyw341>.

License GPL (>= 2)

URL https://github.com/grahamgoff/DAGassist, https://grahamgoff.com/DAGassist/

BugReports https://github.com/grahamgoff/DAGassist/issues

Depends R (>= 3.5)

Imports broom, cli, crayon, dagitty, magrittr, stats, tools, utils, writexl, dplyr, dotwhisker

Suggests fixest, ggplot2, ggdag, knitr, modelsummary, rmarkdown, testthat (>= 3.0.0), marginaleffects, WeightIt, DirectEffects, lme4, readxl

VignetteBuilder knitr

Config/Needs/website pkgdown, rmarkdown, htmltools

Config/testthat/edition 3

Encoding UTF-8

Config/roxygen2/version 8.0.0

LazyData true

NeedsCompilation no

Author Graham Goff [aut, cre] (ORCID: <https://orcid.org/0000-0002-0717-6995>),
Michael Denly [aut] (ORCID: <https://orcid.org/0000-0002-7074-5011>)

Repository CRAN

Date/Publication 2026-11-12 15:30:51 UTC

Contents

add_edges_robustness 2
bad_controls_in 3
balance_models 4
classify_nodes 5
DAGassist 6
pdag_robustness 11
print.DAGassist_addedge_summary 12
print.DAGassist_pdag_summary 13
print.DAGassist_report 13
print.DAGassist_roles 14
print.DAGassist_validation 14
toy_dag 15
toy_data 15
turnout_dag 16
turnout_data 17

Index **18**

`add_edges_robustness` *Diagnose robustness to added ("missing") edges*

Description

Tests how the adjustment set and identification respond to edges the root DAG assumes are absent (DAGWOOD exclusion branches). Each added edge is evaluated as its own branch DAG (root + that edge).

Usage

```
add_edges_robustness(dag, exposure, outcome, add_edges, formula = NULL)
```

Arguments

`dag`	A `dagitty` DAG.
`exposure`, `outcome`	Optional; inferred from the DAG when omitted.
`add_edges`	Character vector like `c("Z -> Y", "X <-> Y")`. Directed (`->`, `<-`) and bidirected (`<->`, latent common cause) edges are supported.
`formula`	Optional model formula or engine call; used only to decide whether a role-flipping covariate is in your specification.

Value

A DAGassist_addedge_summary object.

Examples

```
# What if the DAG is missing an arrow, or has unmeasured confounding?
add_edges_robustness(toy_dag, add_edges = c("A -> X", "X <-> Y"))
```

---

bad_controls_in *flag bad controls (mediator/collider/desc of Y) among a candidate set*

Description

flag bad controls (mediator/collider/desc of Y) among a candidate set

Usage

```
bad_controls_in(dag, controls, exposure, outcome)
```

Arguments

dag	A dagitty DAG object.
controls	Character vector of variable names.
exposure	Character; exposure node name (X).
outcome	Character; outcome node name (Y).

Value

A character vector (possibly empty) containing the elements of controls that are identified as "bad controls".

This is essentially the inverse of pick_minimal_controls(), as it returns bad controls, rather than the minimal/canonical set of good controls

Examples

```
# Which variables in a formula are bad controls, given the DAG?
bad_controls_in(toy_dag, controls = c("M", "C", "Z"))
```

`balance_models` *Balance diagnostics across a list of fitted models*

Description

Compares the complete-case analytic sample of a reference model against each other model in a named list, flagging covariates whose sample composition shifts (`|(S)MD| > threshold`). Engine-agnostic: it reads variable names from each model and recomputes samples on `data`, so it works for `lm`, `glm`, `feols`, `coxph`, weighted fits, etc. Models are never refit.

Usage

```
balance_models(
  models,
  data,
  reference = 1,
  covariates = NULL,
  threshold = 0.1,
  extra_vars = NULL,
  include_outcome = FALSE
)
```

Arguments

`models`	Named list of fitted models. The names label the comparisons.
`data`	The shared source data frame (a superset of every model's rows).
`reference`	Name or index of the model that defines the baseline sample.
`covariates`	Optional character vector of covariates to assess. Default: the union of all models' predictors present in `data`.
`threshold`	Flag covariates with `|(S)MD|` above this (default 0.1).
`extra_vars`	Additional variables that drive listwise deletion (e.g. a weight or cluster column not detected from the call).
`include_outcome`	Assess the outcome variable too (default FALSE).

Value

An object of class `DAGassist_balance`: a list with `$reference`, `$comparisons` (per-comparison (S)MD tables) and `$summary` (one row per comparison: n_ref, n_cmp, n_covariates, n_flagged, pct_flagged, any_flagged).

Examples

```
# Rows with missing C drop out of the Original model only
d <- toy_data
d$C[d$Z > 1] <- NA
balance_models(
  list(Original = lm(Y ~ X + M + C + Z, data = d),
       Minimal  = lm(Y ~ X + Z, data = d)),
  data = d
)
```

`classify_nodes` *Classify DAG nodes*

Description

Labels each node by causal role in a console tabular grid. This function is mostly used as an internal helper, but can be used on its own. Users are encouraged to alternatively use `DAGassist::DAGassist(show=roles)` for role table specific output.

Usage

```
classify_nodes(dag, exposure, outcome)
```

Arguments

`dag`	A `dagitty` DAG object.
`exposure`	Optional– inferred from DAG if not set; character; exposure node name (Exp.).
`outcome`	Optional– inferred from DAG if not set; character; outcome node name (Out.).

Value

A data.frame with one row per node and columns:

- `variable` (node name)
- logical flags: `is_exposure`, `is_outcome`, `is_confounder`, `is_mediator`, `is_collider`, `is_neutral_on_treatment`, `is_neutral_on_outcome`, `is_descendant_of_mediator`, `is_descendant_of_collider`, `is_descendant_of_conf` `is_descendant_of_confounder_off_bdp`
- `role` (a single primary label)

Note

Roles legend: `Exp.` = exposure `Out.` = outcome `CON` = confounder `MED` = mediator `COL` = collider `dOut` = descendant of Out. `dMed` = descendant of any mediator, `dCol` = descendant of any collider `dConfOn` = descendant of a confounder on a back-door path `dConfOff` = descendant of a confounder off a back-door path `NCT` = neutral control on treatment `NCO` = neutral control on outcome

Examples

```
classify_nodes(toy_dag)
```

`DAGassist` *Generate and/or export report that classifies nodes, compares models, and (optionally) target causal estimands.*

Description

`DAGassist()` validates a DAG + model specification, classifies node roles, builds minimal and canonical adjustment sets, fits comparable models, and renders a compact report in several formats (console, LaTeX fragment, DOCX, XLSX, plain text). It can also target sample-average estimands via weighting (e.g., total) and recover sample average controlled direct effects via sequential g-estimation.

Usage

```
DAGassist(
  dag,
  formula = NULL,
  data = NULL,
  exposure,
  outcome,
  engine = stats::lm,
  labels = NULL,
  verbose = TRUE,
  type = c("console", "latex", "word", "docx", "excel", "xlsx", "text", "txt", "dwplot",
    "dotwhisker"),
  show = c("all", "roles", "models"),
  out = NULL,
  imply = FALSE,
  eval_all = FALSE,
  exclude = NULL,
  omit_intercept = TRUE,
  omit_factors = TRUE,
  bivariate = FALSE,
  estimand = c("raw", "none", "total", "direct"),
  engine_args = list(),
  weights_args = list(),
  wts_omit = NULL,
  auto_acde = TRUE,
  acde = list(),
  directeffects_args = list(),
  uncertain_edges = NULL,
  pdag = NULL,
  add_edges = NULL
)
```

Arguments

dag	A **dagitty** object (see dagitty::dagitty()).
formula	Either (a) a standard model formula Y ~ X + ..., or (b) a single **engine call** such as feols(Y ~ X + Z | fe, data = df, ...). When an engine call is provided, engine, data, and extra arguments are automatically extracted from the call.
data	A data.frame (or compatible, e.g. tibble). Optional if supplied via the engine call in formula.
exposure	Optional character scalar; if missing/empty, inferred from the DAG (must be unique).
outcome	Optional character scalar; if missing/empty, inferred from the DAG (must be unique).
engine	Modeling function, default stats::lm. Ignored if formula is a single engine call (in that case the function is taken from the call).
labels	list; optional variable labels (named character vector or data.frame).
verbose	logical (default TRUE). Controls verbosity in the console printer (formulas + notes) and in type = "text" output.
type	output type. One of "console" (default), "latex"/"docx"/"word", "excel"/"xlsx", "text"/"txt", or the plotting types "dwplot"/"dotwhisker". For type = "latex", if no out= is supplied, a LaTeX fragment is printed to the console instead of being written to disk.
show	character vector or list; specify which sections to include in the output. One of "all" (default), "roles" (roles grid only), or "models" (model comparison only.
out	output file path for the non-console types: • type="latex": a **LaTeX fragment** written to out (usually .tex); when omitted, the fragment is printed to the console. • type="text"/"txt": a **plain-text** file written to out; when omitted, the report is printed to console. • type="dotwhisker"/"dwplot": a **image (.png)** file written to out; when omitted, the plot is rendered within RStudio. • type="docx"/"word": a **Word (.docx)** file written to out. • type="excel"/"xlsx": an **Excel (.xlsx)** file written to out. Ignored for type="console".
imply	logical; default FALSE. Controls whether roles/sets are computed on a **pruned DAG** or the **full DAG**. • If FALSE (default): restrict DAG evaluation to exposure, outcome, and terms named in the model (prune the DAG to what appears in the specification). • If TRUE: evaluate on the full DAG and allow DAG-implied controls in the minimal/canonical sets; the roles table includes all DAG nodes.
eval_all	logical; default FALSE. When TRUE, retain original RHS terms that are not DAG nodes (e.g., fixed effects, interactions, splines) in derived minimal/canonical formulas. When FALSE, non-DAG RHS terms are dropped from derived formulas.

`exclude` character vector or list; remove neutral controls from the canonical set. Recognized values are `"nct"` (drop *neutral-on-treatment* controls) and `"nco"` (drop *neutral-on-outcome* controls). Users can supply one or both, e.g. `exclude = c("nco", "nct")`; each requested variant is fitted and shown as a separate "Canon. (-...)" column in the console/model exports.

`omit_intercept` logical; drop intercept rows from the model comparison display (default `TRUE`).

`omit_factors` logical; drop factor-level rows from the model comparison display (default `TRUE`). This parameter only suppresses factor **output**; factor terms still enter the regression.

`bivariate` logical; if `TRUE`, include a bivariate (exposure-only) specification in the comparison table in addition to the user's original and DAG-derived models (default `FALSE`).

`estimand` character vector; causal estimand(s) for reported columns. Any of: `"raw"` (default), `"total"`, `"direct"`, or `"none"`.

- `"raw"`: naive regression fits implied by the supplied engine/formulas.
- `"total"`: inverse-probability weighted versions of each comparison model (via **WeightIt**) to target sample total effect.
- `"direct"`: for DAGs with mediator(s), adds sequential g-estimation columns: (i) unweighted sequential-g and (ii) IPW-weighted sequential-g (weights estimated without conditioning on mediators) to target the **sample average controlled direct effect**.

`engine_args` Named list of extra arguments forwarded to `engine(...)`. If `formula` is an engine call, arguments from the call are merged with `engine_args` (call values take precedence).

`weights_args` list; arguments forwarded to **WeightIt** when computing IPW weights for `"total"` and for the weighted direct effect refit. If `trim_at` is supplied, weights are winsorized at the requested quantile before refitting.

`wts_omit` character vector; terms to omit from the weighting (treatment) model even when `eval_all = TRUE`. Useful for keeping non-DAG fixed effects in the outcome model while preventing them from entering the propensity/weight model.

`auto_acde` logical; if `TRUE` (default), automates handling conflicts between specifications and estimand arguments. Fails gracefully with a helpful error when users specify ACDE estimand for a model without mediators.

`acde` list; options for the controlled direct effect workflow (estimand `"direct"`). Users can override parts of the sequential g-estimation specification with named elements: `m` (mediators), `x` (baseline covariates), `z` (intermediate covariates), `fe` (fixed-effects variables), `fe_as_factor` (wrap `fe` as `factor()`), and `include_descendants` (treat descendants of mediators as mediators).

`directeffects_args` Named list of arguments forwarded to `DirectEffects::sequential_g()` when `estimand` includes `"direct"` (e.g., simulation/bootstrap controls, variance estimator options).

`uncertain_edges` Character vector of edges with unknown direction, e.g. `c("A -- B")`. Triggers a PDAG robustness summary. See `pdag_robustness()`.

pdag — Optional dagitty PDAG; its undirected (--) edges are treated as uncertain, equivalent to listing them in uncertain_edges.

add_edges — Character vector of hypothesized edges absent from the DAG, e.g. c("Z -> Y", "X <-> Y"). Each is tested as a separate exclusion-branch DAG: DAGassist reports whether adding it changes the adjustment set or breaks identification. Directed (->, <-) and bidirected (<->) edges are supported. See add_edges_robustness().

Details

Engine-call parsing. If formula is a call (e.g., feols(Y ~ X | fe, data=df)), DAGassist extracts the engine function, formula, data argument, and any additional engine arguments directly from that call; these are merged with engine/engine_args you pass explicitly (call arguments win).

fixest tails. For engines like **fixest** that use | to denote FE/IV parts, DAGassist preserves any | ... tail when constructing minimal/canonical formulas (e.g., Y ~ X + controls | fe | iv(...)).

Roles grid. The roles table displays short headers:

- Exp. (exposure),
- Out. (outcome),
- CON (confounder),
- MED (mediator),
- COL (collider),
- dOut (descendant of Y),
- dMed (descendant of any mediator),
- dCol (descendant of any collider),
- dConfOn (descendant of a confounder **on** a back-door path),
- dConfOff (descendant of a confounder **off** a back-door path),
- NCT (neutral control on treatment),
- NCO (neutral control on outcome). These extra flags are used to (i) warn about bad controls, and (ii) build filtered canonical sets such as "Canonical (–NCO)" for export.

Bad controls. For total-effect estimation, DAGassist flags as bad controls any variables that are MED, COL, dOut, dMed, or dCol. These are warned in the console and omitted from the model-comparison table. Valid confounders (pre-treatment) are eligible for minimal/canonical adjustment sets.

Output types.

- console prints roles, adjustment sets, formulas (if verbose), and a compact model comparison (using {modelsummary} if available, falling back gracefully otherwise).
- latex writes or prints a **LaTeX fragment** you can \\input{} into a paper — it uses tabularray long tables and will include any requested Canon. (-NCO / -NCT) variants.
- docx/word writes a **Word** doc (respects options(DAGassist.ref_docx=...) if set).
- excel/xlsx writes an **Excel** workbook with tidy tables.
- text/txt writes a **plain-text** report for logs/notes.
- dwplot/dotwhisker produces a dot-whisker visualization of the fitted models.

Dependencies. Core requires {dagitty}. Optional enhancements: {modelsummary} (pretty tables), {broom} (fallback tidying), {rmarkdown} + **pandoc** (DOCX), {writexl} (XLSX), {dotwhisker}/{ggplot2} for plotting.

Raw vs Weighted Direct Effect The unweighted sequential-g estimator in **DirectEffects** uses linear regression in its second stage. By the Frisch–Waugh–Lovell theorem, this implies an estimand that is weighted by the conditional variance of the (residualized) exposure given controls—i.e., a regression-weighted average of unit-level effects, not a sample-average controlled direct effect. DAGassist therefore reports both the raw sequential-g result and a weighted sequential-g refit (using **WeightIt** IPW weights estimated without mediators) to target the *sample average* controlled direct effect.

Value

A DAGassist_report object (a named list) returned invisibly for file/plot outputs and printed for type = "console".

The object contains:

validation List. Output of validate_spec(): DAG validity + exposure/outcome checks.

roles data.frame. Raw node-role flags from classify_nodes().

roles_display data.frame. Roles table formatted for printing/export.

labels_map Named character vector. Variable → display label map used in tables/plots.

controls_minimal Character vector. (Legacy) One minimal adjustment set.

controls_minimal_all List of character vectors. All minimal adjustment sets.

controls_canonical Character vector. Canonical adjustment set (possibly empty).

controls_canonical_excl Named list. Filtered canonical sets created by exclude.

conditions List. Parsed conditional statements from the DAG (if any).

formulas List. User formula plus DAG-derived formula variants (minimal/canonical/etc.).

models List. Fitted models for each formula variant (including minimal-list fits).

bad_in_user Character vector. RHS terms classified as mediator/collider/etc.

unevaluated Character vector. Terms carried through but not evaluated by the engine.

unevaluated_str Character scalar. Pretty-printed version of unevaluated.

settings List. Print/export settings, including coef_omit and show.

.__data data.frame or NULL. The data used to fit models (stored for downstream helpers).

For file outputs (type = "latex", "docx", "xlsx", "txt", "dotwhisker"), the returned object includes attribute file, the normalized output path.

See Also

print.DAGassist_report() and vignette("DAGassist", package = "DAGassist").

Examples

```
# toy_dag and toy_data ship with the package; the true total effect
# of X on Y is recovered by adjusting for Z alone.

# 1) Core: DAG-derived specs + engine-call parsing
DAGassist(toy_dag, lm(Y ~ X + Z + M, data = toy_data))

# 2) Roles grid only
DAGassist(toy_dag, lm(Y ~ X + Z + M, data = toy_data), show = "roles")

# 3) Target sample-average estimands via weighting

DAGassist(toy_dag, lm(Y ~ X + Z + M, data = toy_data), estimand = "total")

# 4) File export (LaTeX fragment)

  out <- file.path(tempdir(), "dagassist_report.tex")
  DAGassist(toy_dag, lm(Y ~ X + Z + M, data = toy_data), type = "latex", out = out)
```

---

pdag_robustness *Diagnose adjustment-set and role robustness to uncertain edge directions*

Description

Given a DAG plus one or more edges whose direction is uncertain (a PDAG), enumerate every acyclic orientation and report whether the minimal/canonical adjustment sets or covariate roles change – i.e., whether your estimand is robust to that structural uncertainty.

Usage

```
pdag_robustness(
  dag,
  exposure,
  outcome,
  uncertain_edges = NULL,
  pdag = NULL,
  formula = NULL,
  max_uncertain = 10L
)
```

Arguments

dag	A dagitty DAG (the "root" model).
exposure, outcome	Optional; inferred from the DAG when omitted.

uncertain_edges	Character vector like c("A -- B") naming edges whose direction is unknown.
pdag	Optional dagitty PDAG; its -- edges are treated as uncertain.
formula	Optional model formula; used to decide whether an ambiguous covariate is actually in your specification (affects re-estimation advice).
max_uncertain	Integer guard on the number of uncertain edges (default 10 -> up to 1024 worlds).

Value

A DAGassist_pdag_summary object (printed as a bullet summary).

Examples

```
# What if we're unsure which way two arrows point?
pdag_robustness(toy_dag, uncertain_edges = c("Z -- X", "M -- Y"))
```

---

print.DAGassist_addedge_summary

Print an edge-addition (exclusion) robustness summary

Description

Print an edge-addition (exclusion) robustness summary

Usage

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

Arguments

x	A DAGassist_addedge_summary from add_edges_robustness().
...	Ignored.

Value

Invisibly returns x.

print.DAGassist_pdag_summary

Print a PDAG robustness summary

Description

Print a PDAG robustness summary

Usage

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

Arguments

x	A DAGassist_pdag_summary from pdag_robustness().
...	Ignored.

Value

Invisibly returns x.

print.DAGassist_report

Print method for DAGassist reports

Description

Nicely prints the roles table, highlights potential bad controls, shows minimal/canonical adjustment sets, optionally shows formulas, and renders a compact model comparison (using {modelsummary} if available, falling back to {broom} or basic coef() preview).

Usage

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

Arguments

x	A "DAGassist_report" object returned by DAGassist().
...	Additional arguments (currently unused; present for S3 compatibility).

Details

The printer respects the verbose flag in the report: when TRUE, it includes formulas and a brief note on variables added by DAG logic (minimal and canonical sets). Fitting errors are shown inline per model column and do not abort printing.

Value

Invisibly returns `x`.

`print.DAGassist_roles` *Print node classifications (aligned)*

Description

Print node classifications (aligned)

Usage

```
## S3 method for class 'DAGassist_roles'
print(x, n = Inf, ...)
```

Arguments

`x`	Output of classify_nodes() (class "DAGassist_roles")
`n`	Max rows to print (default all)
`...`	(ignored)

Value

Invisibly returns x

`print.DAGassist_validation` *Minimal, clean printout for validation results with color coding*

Description

Minimal, clean printout for validation results with color coding

Usage

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

Arguments

`x`	the list (class `out`) from validate_spec
`n`	Max number of issues to show (default 10).
`...`	Ignored.

Value

Invisibly returns `x`.

toy_dag *DAG behind toy_data*

Description

The graph used to simulate toy_data, with X marked as the exposure and Y as the outcome. Pass it directly to DAGassist().

Usage

```
toy_dag
```

Format

A dagitty object with 7 nodes.

Source

Defined in data-raw/make_data.R.

See Also

toy_data

toy_data *Toy data for demonstrating causal roles*

Description

A small simulated dataset in which each causal role that DAGassist() classifies appears at least once. Generated from toy_dag, so the correct adjustment set is known by construction: the total effect of X on Y requires adjusting for Z alone.

Usage

```
toy_data
```

Format

A data frame with 2,000 rows and 7 columns:

X Exposure. Caused by Z.

Y Outcome. Caused by X, M, Z, A, and B.

Z Confounder: a common cause of X and Y.

M Mediator: lies on the path X -> M -> Y.

C Collider: a common descendant of X and Y.

A Neutral control on the outcome; affects Y only.

B Neutral control on the outcome; affects Y only.

Details

The data-generating process is linear and Gaussian with `set.seed(42)`. True coefficients are `X -> Y = 0.7`, `M -> Y = 0.6`, `Z -> Y = 0.3`, `A -> Y = 0.2`, `B -> Y = -0.1`.

Source

Simulated by `data-raw/make_data.R`.

See Also

toy_dag for the generating graph.

---

turnout_dag *DAG behind turnout_data*

Description

The graph used to simulate turnout_data, with `income` marked as the exposure and `turnout` as the outcome. Node coordinates and display labels are attached, so it plots directly with `ggdag::ggdag_status()`. Pass it to `DAGassist()` to classify covariates and re-estimate against DAG-derived adjustment sets.

Usage

```
turnout_dag
```

Format

A `dagitty` object with 7 nodes and 10 edges.

Details

Minimal sufficient adjustment sets, as returned by `dagitty::adjustmentSets()`:

- Total effect: `{age, parental_ses}`
- Direct effect: `{age, polint, parental_ses}`
- Canonical (total): `{age, elect_comp, industry, parental_ses}`

Source

Defined in `data-raw/make_data.R`.

See Also

turnout_data

turnout_data *Simulated data on income and voter turnout*

Description

A simulated cross-section used to demonstrate the `DAGassist()` workflow on an applied question: does higher income increase voter turnout? Because the data-generating process is known, the correct answer is known too, and the cost of a misspecified control set can be read directly off the output.

Usage

```
turnout_data
```

Format

A data frame with 5,000 rows and 7 columns:

turnout Outcome. Voter turnout propensity.

income Exposure. Caused by `parental_ses`, `age`, and `industry`.

parental_ses Confounder: a common cause of `income` and `turnout`.

age Confounder: affects `income`, `turnout`, and `industry`.

polint Political interest. A mediator on the path `income -> polint -> turnout`. Adjusting for it removes the indirect effect and is a bad control for the total effect.

industry A cause of `income` lying on the back-door path `income <- industry <- age -> turnout`. Safe to adjust for, but not required once `age` is included.

elect_comp Election competitiveness. A neutral control on the outcome: it affects `turnout` only, so adjusting for it leaves the estimate unbiased and improves precision.

Details

Generated from turnout_dag with `set.seed(42)`. Income raises turnout directly (`0.30`) and indirectly by raising political interest (`0.50 * 0.40 = 0.20`), so the **true total effect is** `0.50` and the **true direct effect is** `0.30`.

Adjusting for `{age, parental_ses}` recovers the total effect; adjusting for `{age, polint, parental_ses}` recovers the direct effect. Regressing `turnout` on every available covariate returns roughly `0.28` — close to the direct effect, because conditioning on the mediator `polint` silently changes the estimand rather than producing an obviously wrong number.

Source

Simulated by `data-raw/make_data.R`.

See Also

turnout_dag for the generating graph; toy_data for a smaller example covering every causal role.

Index

∗ **datasets**
toy_dag, 15
toy_data, 15
turnout_dag, 16
turnout_data, 17

add_edges_robustness, 2
add_edges_robustness(), *9*, *12*

bad_controls_in, 3
balance_models, 4

classify_nodes, 5

DAGassist, 6
DAGassist(), *13*, *15–17*
dagitty::dagitty(), *7*
DirectEffects::sequential_g(), *8*

pdag_robustness, 11
pdag_robustness(), *8*, *13*
print.DAGassist_addedge_summary, 12
print.DAGassist_pdag_summary, 13
print.DAGassist_report, 13
print.DAGassist_report(), *10*
print.DAGassist_roles, 14
print.DAGassist_validation, 14

stats::lm, *7*

toy_dag, *15*, 15, *16*
toy_data, *15*, 15, *17*
turnout_dag, 16, *17*
turnout_data, *16*, 17