

Package: DAGassist (via r-universe)

May 21, 2026

Title Test Robustness with Directed Acyclic Graphs

Version 0.2.8

Maintainer Graham Goff <goffgrahamc@gmail.com>

Description Provides robustness checks 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. Exports 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](https://doi.org/10.1017/CBO9780511803161)> and Textor et al. (2016) <[doi:10.1093/ije/dyw341](https://doi.org/10.1093/ije/dyw341)>.

License GPL (>= 2)

URL <https://github.com/grahamgoff/DAGassist>,
<https://grahamgoff.github.io/DAGassist/>

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

Depends R (>= 3.5)

Imports broom, cli, crayon, dagitty, magrittr, stats, tools, utils,
writexl, dplyr, ggplot2, dotwhisker

Suggests devtools, fixest, ggdag, knitr, modelsummary, rmarkdown,
testthat (>= 3.0.0), tidyverse, DiagrammeR, marginalesffects,
WeightIt, DirectEffects, lme4

VignetteBuilder knitr

Config/Needs/website pkgdown, rmarkdown, DiagrammeR, htmltools

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libicu-dev libssl-dev libnode-dev zlib1g-dev

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

Date/Publication 2026-02-20 00:13:17 UTC

RemoteUrl https://github.com/grahamgoff/dagassist

RemoteRef HEAD

RemoteSha 962c6c2dbdd9b91daa7ba937c689eff1fe775aea

Contents

bad_controls_in	2
classify_nodes	3
DAGassist	4
print.DAGassist_report	9
print.DAGassist_roles	10
print.DAGassist_validation	10
Index	12

bad_controls_in	<i>flag bad controls (mediator/collider/desc of Y) among a candidate set</i>
-----------------	--

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

```
d <- ggdag::dagify(
  Y ~ X + M + Z,
  M ~ X + Z,
  C ~ X + Y,
  exposure = "X",
  outcome = "Y")
# M: mediator / Z: confounder / C: collider

# hypothetical candidate controls
controls <- c("Z", "M", "C")

# Flag controls that would bias the total effect of X on Y:
bad_controls_in(d, controls = c("Z", "M", "C"), exposure = "X", outcome = "Y")

# expected: c("M", "C") # mediator & collider are "bad controls"; Z is OK
```

 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_confounder`, `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

```
d1 <- dagitty::dagitty("dag {X[exposure];Y[outcome] Z -> X; Z -> Y; X -> Y}")
classify_nodes(d1)
```

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., SATE/SATT) and recover sample average controlled direct effects via sequential g-estimation (e.g., SACDE).

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", "SATE", "SATT", "SACDE", "SCDE"),
  engine_args = list(),
```

```

weights_args = list(),
wts_omit = NULL,
auto_acde = TRUE,
acde = list(),
directeffects_args = list()
)

```

Arguments

dag	A dagitty object (see <code>dagitty::dagitty()</code>).
formula	Either (a) a standard model formula $Y \sim X + \dots$, or (b) a single engine call such as <code>feols(Y ~ X + Z fe, data = df, ...)</code> . 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 <code>stats::lm</code> . 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).
type	output type. One of "console" (default), "latex"/"docx"/"word", "excel"/"xlsx", "text"/"txt", or the plotting types "dwpplot"/"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: <ul style="list-style-type: none"> • 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"/"dwpplot": 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 . <ul style="list-style-type: none"> • If FALSE (default): restrict DAG evaluation to exposure, outcome, and terms named in the model (prune the DAG to what appears in the specification).

	<ul style="list-style-type: none"> • 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 <i>neutral-on-treatment</i> controls) and "nco" (drop <i>neutral-on-outcome</i> 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	<p>character vector; causal estimand(s) for reported columns. Any of: "raw" (default), "SATE", "SATT", "SACDE" (alias "SCDE"), or "none".</p> <ul style="list-style-type: none"> • "raw": naive regression fits implied by the supplied engine/formulas. • "SATE"/"SATT": inverse-probability weighted versions of each comparison model (via WeightIt) to target sample ATE/ATT. • "SACDE"/"SCDE": 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 "SATE"/"SATT" and for the weighted SACDE refit. If trim_at is supplied, weights are win-sorized 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 (estimands "SACDE"/"SCDE"). 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 "SACDE" (e.g., simulation/bootstrap controls, variance estimator options).

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),
- dConFO (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 SACDE. 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 \rightarrow 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

```

if (requireNamespace("dagitty", quietly = TRUE)) {
  g <- dagitty::dagitty("dag { Z -> X; X -> M; X -> Y; M -> Y; Z -> Y }")
  dagitty::exposures(g) <- "X"; dagitty::outcomes(g) <- "Y"
  n <- 300
  Z <- rnorm(n); X <- 0.8*Z + rnorm(n)
  M <- 0.9*X + rnorm(n)
  Y <- 0.7*X + 0.6*M + 0.3*Z + rnorm(n)
  df <- data.frame(Z, X, M, Y)

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

  # 2) Target sample-average estimands via weighting (requires WeightIt)
  if (requireNamespace("WeightIt", quietly = TRUE)) {
    r2 <- DAGassist(g, lm(Y ~ X + Z + M, data = df), estimand = "SATE")
  }

  # 3) Mediator case: sequential g-estimation (requires DirectEffects)
  if (requireNamespace("DirectEffects", quietly = TRUE)) {
    r3 <- DAGassist(g, lm(Y ~ X + Z + M, data = df), estimand = "SACDE")
  }

  # 4) File export (LaTeX fragment)

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

```

```
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.

Index

`bad_controls_in`, [2](#)

`classify_nodes`, [3](#)

`DAGassist`, [4](#)

`DAGassist()`, [9](#)

`dagitty::dagitty()`, [5](#)

`DirectEffects::sequential_g()`, [7](#)

`print.DAGassist_report`, [9](#)

`print.DAGassist_report()`, [8](#)

`print.DAGassist_roles`, [10](#)

`print.DAGassist_validation`, [10](#)

`stats::lm`, [5](#)