Package 'DAGassist'

November 11, 2025

Title Test Robustness with Directed Acyclic Graphs

Version 0.2.7

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, flags problematic controls, and generates a report comparing the original model with minimal and canonical 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> and Textor et al. (2016) <doi: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

VignetteBuilder knitr

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

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

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 2025-11-11 15:50:07 UTC

Contents

`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

```
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_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

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

`DAGassist` *Generate a (console/LaTeX/word/excel/txt) report classifying nodes and comparing models*

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 also supports passing a **single engine call** (e.g. `feols(Y ~ X + Z | fe, data = df)`) instead of a plain formula.

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,
  engine_args = list()
)
```

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	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 "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	Which sections to include in the output. One of "all" (default), "roles" (only the roles grid), or "models" (only the model comparison table/plot). This makes it possible to generate and export just roles or just comparisons.
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. Specifies **evaluation scope.** • If FALSE (default): restrict DAG evaluation to variables **named in the formula** (prune the DAG to exposure, outcome, and RHS terms). Roles/sets/bad-controls are computed on this pruned graph, and the roles table **only** shows those variables. Essentially, it fits the DAG to the formula. • If TRUE: evaluate on the **full DAG** and allow DAG-implied controls in the minimal/canonical sets. The roles table shows all DAG nodes, and the printout notes any variables added beyond your RHS. Essentially, it fits the formula to the DAG.

`eval_all`	Logical; default `FALSE`. When `TRUE`, keep **all original RHS terms** that are not in the DAG (e.g., fixed effects, interactions, splines, convenience covariates) in the minimal and canonical formulas. When `FALSE` (default), RHS terms not present as DAG nodes are dropped from those derived formulas.
`exclude`	Optional character vector to remove neutral controls from the canonical set. Recognized values are `"nct"` (drop *neutral-on-treatment* controls) and `"nco"` (drop *neutral-on-outcome* controls). You 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**–they are still included in 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.
`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).

Details

In addition to tabular export formats, you can create a dot-whisker plot (via `type = "dwplot"` or `type = "dotwhisker"`) for the model comparison.

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.

Value

An object of class `"DAGassist_report"`, invisibly for file and plot outputs, and printed for `type="console"`. The list contains:

- `validation` - result from `validate_spec(...)` which verifies acyclicity and X/Y declarations.
- `roles` - raw roles data.frame from `classify_nodes(...)` (logic columns).
- `roles_display` - roles grid after labeling/renaming for exporters.
- `bad_in_user` - variables in the user's RHS that are `MED`/`COL`/`dOut`/`dMed`/`dCol`.
- `controls_minimal` - (legacy) one minimal set (character vector).
- `controls_minimal_all` - list of all minimal sets (character vectors).
- `controls_canonical` - canonical set (character vector; may be empty).
- `controls_canonical_excl` - named list of filtered canonical sets (e.g. `$nco`, `$nct`) when `exclude` is used.
- `formulas` - list with `original`, `minimal`, `minimal_list`, `canonical`, and any filtered canonical formulas.
- `models` - list with fitted models `original`, `minimal`, `minimal_list`, `canonical`, and any filtered canonical models.
- `verbose`, `imply` - flags as provided.

Interpreting the output

See the vignette articles for worked examples on generating roles-only, models-only, and LaTeX/Word/Excel reports.

Model Comparison:

- **Minimal** - the smallest adjustment set that blocks all back-door paths (confounders only).
- **Canonical** - the largest permissible set: includes all controls that are not `MED`, `COL`, `dOut`, `dMed`, or `dCol`.

Errors and edge cases

- If exposure/outcome cannot be inferred uniquely, the function stops with a clear message.
- Fitting errors (e.g., FE collinearity) are captured and displayed in comparisons without aborting the whole pipeline.

See Also

`print.DAGassist_report()` for the console printer, and the helper exporters in `report_*` modules.

Examples

```
# generate a console DAGassist report
DAGassist(dag = g,
          formula = lm(Y ~ X + Z + C + M, data = df))

# generate a LaTeX DAGassist report in console
DAGassist(dag = g,
          formula = lm(Y ~ X + Z + C + M, data = df),
          type = "latex")

# generate just the roles table in the console
DAGassist(dag = g,
          show = "roles")
```

`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