Package 'dagwood'

October 13, 2022

Title DAGs with Omitted Objects Displayed (DAGWOOD)

Version 0.1.4

Description DAGs With Omitted Objects Displayed (DAGWOOD) is a framework to help reveal key hidden assumptions in a causal DAG. This package provides an implementation of the DAGWOOD algorithm. Further description can be found in Haber et al (2022) <DOI:10.1016/j.annepidem.2022.01.001>.

License MIT + file LICENSE

Encoding UTF-8

Imports dagitty

Suggests ggdag

RoxygenNote 7.1.2

NeedsCompilation no

Author Noah Haber [aut, cre] (<https://orcid.org/0000-0002-5672-1769>)

Maintainer Noah Haber <noahhaber@stanford.edu>

Repository CRAN

Date/Publication 2022-03-22 22:00:02 UTC

R topics documented:

---

dagwood *DAGs with Omitted Objects Displayed (DAGWOOD)*

Description

A framework to help reveal key hidden assumptions in a causal directed acyclic graph (DAG). Details on how DAGs with Omitted Objects Displayed (DAGWOOD) works, can be used, and should be interpreted are available from our paper: https://doi.org/10.1016/j.annepidem.2022.01.001

DAGWOODS take a root DAG and generates DAGWOOD branch DAGs from it. At present, there are two types of branch DAGs:

1. Exclusion branch DAGs represent nodes which, if exist, violate key causal model assumptions. The most common example are confounders. There are two types of exclusion branch DAGs at present: known exclusion branch DAGs (KEBDs) representing nodes that an analyst knows exist, but did not include them in the analytical model (for example, due to lack of data), and unknown exclusion branch DAGs (UEBDs) representing unknown or otherwise undeclared nodes which violate key model assumptions.
2. Misdirection branch DAGs represent DAGs in which a minimal number of DAG arrows might be flipped in order to violate key model assumptions. Common examples might be the case when something that is adjusted-for as a confounder is a collider, or if missing time nodes exist such that there is "reverse" causality between the exposure and outcome.

This package is implemented based on the DAGITTY package by Johannes Textor. More information is available here: http://dagitty.net/learn/index.html

The dagwood function generates an object of class "dagwood" which organizes relevant outputs for use, using the $ operator. At this time, this includes $DAGs.branch (the main output) and $DAGs.tested (used mainly for diagnostics).

DAGWOOD objects currently output all branch DAGs in $DAGs.branch from the DAGWOOD object, which is a data frame which includes #' details of why the branch DAG was tested, the type of branch DAG, what was changed from the original root #' DAG, and the branch DAG itself as a dagitty object. The branch DAGs can be viewed, manipulated, etc as any other dagitty-based DAG, using the dagitty syntax.

In addition, the DAGWOOD object contains $DAGs.tested, which stores each potential branch DAG that the algorithm tests for validity, and the results of those tests. This is mainly used for diagnostic purposes (i.e. determining what DAGs were tested and why they were accepted or rejected).

Current limitations to this package: Instrumental variables support is experimental (conditional IVs not yet supported) Does not currently support non-minimal adjustment sets from root DAGs Does not currently match KEBD branch DAG matching with the UEBD branch DAGs Does not check for issues with the root DAG (i.e. assumes root DAG is correctly specified)

Future features: improved graphical outputs, additional branch DAG types, possibly a GUI Improve DAG import features and data checking (e.g. make sure IV is valid on entry)

Abbreviations used: BD: branch DAG DAG: directed acyclic graph DAGWOOD: directed acyclic graph with omitted objects displayed EBD: exclusion branch DAG IV: instrumental variable KEBD: known exclusion branch DAG MBD: misdirection branch DAG UEBD: unknown exclusion branch DAG

Usage

```
dagwood(
  DAG.root,
```

```
  exposure = NA,
  outcome = NA,
  KEBDs = NA,
  instrument = NA,
  fixed.arrows = NA
)
```

Arguments

DAG.root	A string formula describing the DAG, in the format style of DAGitty
exposure	A character string identifying the exposure of interest (must match the formula above)
outcome	A character string identifying the outcome of interest (must match the formula above)
KEBDs	(not yet implemented)
instrument	The character string identifying which node is the instrumental variable of interest
fixed.arrows	(experimental) These arrows are prevented from flipping direction in the misdirection branch DAG algorithm. Nodes and arrows should be entered in the form of DAGitty.

Value

Does not return a data or object; all outputs are stored in the $ operators (e.g. $DAGs.branch)

References

https://doi.org/10.1016/j.annepidem.2022.01.001

Examples

```
# Generate a DAGWOOD from an example root DAG:
DAG.root <-"Chocolate -> Alzheimers
Chocolate <- Education -> Alzheimers
Chocolate -> CV
CV -> Alzheimers"

# Identify the exposure and outcome of interest
exposure <- "Chocolate"
outcome <- "Alzheimers"

# Run the DAGWOOD algorithm and store results
choc.alz.dagwood <- dagwood(DAG.root,exposure,outcome)

# Get the branch DAGs from the DAGWOOD object
branch.DAGs <- choc.alz.dagwood$DAGs.branch

# Display the first branch DAG in the list
```

```
library(ggdag)
ggdag(branch.DAGs$DAG.branch.candidate[1]) + theme_dag()
```

Index