DAGitty User Manual

Johannes Textor

March 24, 2011

Abstract

DAGitty is a software for drawing and analyzing causal diagrams, known in epidemiology as directed acyclic graphs (DAGs). Its main tasks include the identification of minimal sufficient adjustment sets for estimating the total causal effect from exposure to outcome, and the identification of insufficient adjustment via biasing paths.

DAGitty should run directly and without installation in any web browser that supports modern HTML and JavaScript.

Contents

1 Introduction

1.1 Causal diagrams

To convey an idea of the purpose of DAGitty, this introduction gives some tiny examples on confounding and adjustment sets; for a more detailed discussion of these subjects, we recommend the book *Causality* by Judea Pearl [6], or the corresponding chapter *Causal diagrams* in the textbook of Rothman, Greenland, and Lash [7].

Causal diagrams are also called *Bayesian networks* (in computer science) or even *DAGs* (in epidemiology).[1] Simply put, a DAG is a formal model about causal relationships between certain entities of interest in a specific scenario. For example, the sentence "smoking causes cancer" could be translated into the following simple causal diagram:

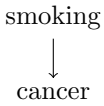


Figure 1: A very simple causal diagram.

An important application for causal diagram, which is also the focus of DAGitty, is to isolate the *causal* effects of a variable of interest, called *exposure* onto another, called *outcome*, from the *biasing* effects between the two variables. For example, consider the following, slightly more complex causal diagram:

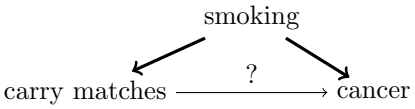


Figure 2: A classical confounding triangle.

If we were to perform a study on the relationship between carrying matches in one's pocket and developing lung cancer, we would probably find a correlation between these two variables. However, as the above diagram indicates, this correlation would not imply that carrying matches in your pocket causes lung cancer: Smokers are more likely to carry matches in their pockets, and also more likely to develop lung cancer. This is a classical example of a *confounded* association between two variables, which is mediated via the *biasing path* (bold). In this example, would we adjust for smoking, e.g. by averaging separate effect estimates for smokers and non-smokers, we would probably no longer find a correlation between carrying matches and lung cancer as such adjustment would *close the biasing path*.

Loosely speaking[2], any set of variables in a causal diagram that blocks all biasing paths between an exposition and an outcome, but leaves the causal paths open, is called a *sufficient adjustment set*. If the causal diagram is accurate, then adjustment, stratification, or selection (e.g. by restriction or matching) for this set of variables in an epidemiological study minimizes bias when estimating the *total effect* from exposition to outcome in an epidemiological study. Adjustment sets will be explained in more detail in Section 4.

[1] The term "DAG" is somewhat confusing to computer scientists and mathematicians, for whom a DAG is simply an abstract mathematical structure without specific semantics attached to it.

[2] See Shpitser et al.[10] for a more precise definition

The purpose of DAGitty is to aid study design through the identification of suitable, small sufficient adjustment sets in complex causal diagrams and, more generally, through the identification of causal and biasing paths in a diagram.

There are two ways to run DAGitty: either from the internet or from your own computer.

1.2 Running DAGitty online

To run DAGitty online, simply open its URL in your favourite browser:

`http://www.dagitty.net`

DAGitty should run in every modern browser. If it doesn't, please send me an E-Mail so I can fix the problem; see contact information at the end of this manual.

1.3 Installing DAGitty on your own computer

DAGitty can be "installed" on your computer for use without an internet connection. To do this, download the file

`http://www.dagitty.net/dagitty.zip`

which is a ZIP archive containing DAGitty's source. Unpack this ZIP file anywhere in your file system. To run DAGitty, just open the file `dags.html` in the unpacked folder.

2 Loading and saving diagrams

This section covers the three basic steps of working with DAGitty: (1) Loading a diagram; (2) manipulating the graphical layout of the diagram; and (3) saving the diagram. First of all, any causal diagram consists of vertices (variables) and edges (relationships between variables). You can either create the diagram directly using DAGitty's graphical user interface (explained in the next section), or prepare a textual diagram description in a word processor such as Microsoft Word ® and then import this description into DAGitty. In addition, DAGitty contains some pre-defined examples that you can use to become familiar with the program. To do so, just select one of the pre-define examples from the "Examples" menu.

2.1 DAGitty's textual syntax for causal diagrams

DAGitty's textual syntax for causal diagrams is similar to the one used by the DAG program by Sven Knüppel [4]. A diagram description *(model text data)* consists of two parts:

1. A list of the variables in the diagram
2. A list of connections between the variables

The list of variables is simply one variable per line (blank lines are ignored by DAGitty). By convention, the variable in the first line is the exposure and the variable in the second line is the outcome of your model. Variable names must not contain spaces or colons; please use dashes or underscores instead (i.e., write `fitness_level` instead of `fitness level`). After each variable name follows a character that indicates the status of the variable, which can be either "1" (normal variable), "A" (adjusted for), or "U" (latent/unobserved).

The list of connection consists of several lines each starting with a start variable name, followed by one or more other target variables that the start variable is connected to. Figure 3 contains a working example of a textual model description. When you modify a diagram within DAGitty, the vertex labels will be augmented by additional information, to help DAGitty remember the layout of the vertices and for other purposes (see rightmost column in Figure 3).

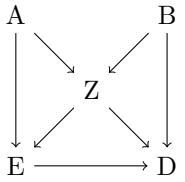
(a) *vertex labels*

E 1
D 1
A 1
B 1
Z 1

(b) *adjacency list*

E D
A E Z
B D Z
Z E D

(c) *resulting graph*



(d) *augmented vertex labels*

E 1 @-2.2,1.6
D 1 @1.4,1.6
A 1 @-2.2,-1.5
B 1 @1.4,-1.5
Z 1 @-0.3,-0.1

Figure 3: Example for a textual model definition with DAGitty (a,b: model text data; c: resulting diagram). When the diagram is edited within DAGitty (d), the vertex labels and adjustment status are augmented with additional information that DAGitty uses to layout the vertices on the canvas (rightmost column): the layout coordinates of each variable are indicated behind the @ sign.

2.2 Loading a textually defined diagram into DAGitty

To load a textually defined diagram into DAGitty, simply copy&paste the variable list, followed by a blank line, followed by the list of connections into the *"Model text data"* text box. Then click on *"Update DAG"*. DAGitty will now generate a preliminary graphical layout for your diagram on the canvas, which may not yet look the way you intended it to look, but can be freely modified.

2.3 Modifying the graphical layout of a diagram

To layout the vertices and edges of your diagram more clearly than DAGitty did, simply drag the vertices with your mouse on the canvas. You may notice that DAGitty modifies the information in the *"Model text data"* field on the fly, and augments it with additional position information for each vertex. In general, all changes you make to your diagram within DAGitty are immediately reflected in the model text data.

2.4 Saving the diagram

To save your diagram locally, just copy&paste the contents of the *"Model text data"* field to a text file, e.g. a Microsoft Word ® document, and save that file locally to your computer[3].

When you wish to continue working on the diagram, copy the model text data back into DAGitty as explained above.

3 Editing models using the graphical user interface

As explained above, you are free to make changes directly to the textual description of your model, which will be reflected on the canvas next time you click on *"Update DAG"*. However, you can also create, modify, and delete vertices and connections directly on the canvas. The list of minimal sufficient adjustment sets (see next section) will be updated automatically if necessary.

3.1 Creating a new model

To create a new model, select "New Model" from the "Model" menu. You will be asked for the names of the exposure and the outcome variable, and an initial model containing just those variables and an arrow from exposition to outcome will be drawn. Then you can add variables and connections to the model as explained below.

[3]This is most easily done by clicking in the text field, pressing "CTRL + A" to select the entire content of the text field, then pressing "CTRL + C" to copy the selected content. You can then paste the content into Microsoft Word using "CTRL + V"

3.2 Adding new variables

To add a new variable to the model, double-click on a free space in the canvas (i.e., not on an existing variable) or press the *"n"* key. A small dialog will pop up asking you for the name of the new variable. Enter the name into the dialog and press the enter key or click *"OK"*. If you click *"Cancel"*, no new variable will be created.

3.3 Adding new connections

To add a new connection, double-click first on the source vertex (which will become highlighted) and then on the target vertex. The connection will be inserted. If a connection existed before in the opposite direction, that connection will be deleted, because otherwise there would now be a cycle in the model.

Instead of double-clicking on a vertex, you can also move the mouse pointer over the vertex and press the key *"c"*.

3.4 Deleting variables

To delete a variable, move the mouse pointer over that variable and hit the *del* key on your keyboard. All connections to that variable will be deleted along with the variable. DAGitty will refuse to delete the exposition or the outcome variable from the model; if you wish to do so, you must previously select a new exposition/outcome (see below).

3.5 Deleting connections

A connection is deleted just like it has been inserted, i.e., by double-clicking first on the start variable and then on the target variable. A connection is also deleted automatically if a new one is inserted in the opposite direction (see above).

3.6 Setting exposition and outcome

As explained above, per default the exposition is the variable in the first line of the variable list and the outcome is the one on the second line. To turn a different variable into the exposition, move the mouse pointer over that variable and hit the *e* key; for the outcome, hit the *o* key instead. Doing so will change the colors of the vertices on the canvas to reflect the new structure of the graph.

3.7 Displaying the moral graph

To identify minimal sufficient adjustment sets, DAGitty uses the so-called "moral graph", which results from a transformation of the model to an undirected, typically smaller, graph. This procedure is also highly recommended if you wish to verify the calculation by hand. See the nice explanation by Shrier and Platt [11] for details on this procedure.

In DAGitty, you can switch between display of the model and its moral graph by pressing the *m* key.

3.8 Workarounds for functions that are still missing

Some functions are not yet here in DAGitty, but would be nice to have and shall be implemented in future versions. In the meantime, the following workarounds can be used.

- Renaming variables: This is not yet conveniently possible. However, you can copy&paste the vertex labels and adjacency list to a word processor of your choice and then replace every occurence of the variable name of choice with the new version ussing the word processor's *search and replace* functions. Afterwards, copy the model description back into DAGitty.

4 Adjustment sets

Finding sufficient adjustment sets is one main purpose of DAGitty. In a nutshell, a sufficient adjustment set is a set S of covariates such that adjustment, stratification, or selection (e.g. by restriction or matching) will minimize bias when estimating the causal effect of the exposure on the outcome. You can read more about controlling bias and counfounding in Pearl's textbook, chapter 3.3 and epilogue [6]. Moreover, Shrier and Platt [11] give a nice step-by-step tutorial on how to test if a set of covariates is a sufficient adjustment set.

Briefly, a sufficient adjustment set S blocks all non-causal paths between exposure and outcome, but leaves open all causal paths (i.e., paths of the form $e \rightarrow x_1 \rightarrow \ldots \rightarrow x_k \rightarrow o$). A path p is blocked (or *d-separated*) by a set Z if at least one of the following properties holds [6]:

- The path p contains a chain $x \rightarrow m \rightarrow y$ or a fork $x \leftarrow m \rightarrow y$ such that m is in Z.
- The path p contains a collider $x \rightarrow c \leftarrow y$ such that c is not in Z and furthermore, Z does not contain any successor of c in the graph.

A path that is not blocked is called *open*. An open non-causal path is called a *biasing path*. As proved by Lauritzen et al. ([5], see also Tian et al. [14]), it suffices to restrict our attention to the part of the model that consists of exposure, outcome, and their ancestors for identifying sufficient adjustment sets. This is indicated by DAGitty by coloring irrelevant nodes in gray. The relevant variables are colored according to which node they are ancestors of (exposure, outcome, or both) – see the legend on the left-hand side of the screen. The highlighting may be turned on and off by toggling the "highlight ancestors" checkbox.

Furhermore, DAGitty can optionally highlight (all) biasing and/or causal paths in different colours. This is controlled via the "highlight causal paths" and "highlight biasing paths" checkboxes.

4.1 Minimal sufficient adjustment sets

A *minimal* sufficient adjustment set (MSAS) is a sufficient adjustment set of which no proper subset is itself sufficient. For example, consider again the causal diagram in Figure 3. In this example, the following three sets are sufficient adjustment sets:

$$\{A, B, Z\}$$

$$\{A, Z\}$$

$$\{B, Z\}$$

Of these three sets, $\{A, Z\}$ and $\{B, Z\}$ are minimal sufficient adjustment sets while the set $\{A, B, Z\}$ is sufficient, but not minimal.

Note that adjusting for $\{Z\}$ is *not* sufficient, since this would "open" the path $E \leftarrow A \rightarrow Z \leftarrow B \leftarrow D$: Since both E and D depend on Z, adjusting for Z will induce additional correlation between E and D.

Pearl [6] gives the following sufficient criterion for a sufficient adjustment set S. Although this criterion is not necessary in general, it is necessary for a *minimal* adjustment set.

- S does not contain any descendant of the exposure. Thus, it is never necessary to adjust for a descendant when the *total effect* from exposure to outcome is to be estimated. Further versions of DAGitty may allow to calculate adjustment sets for effects other than the total effect (e.g., the direct effect), which in general requires adjustment for a descendant of the exposure.
- S contains all variables that are direct parents of both exposure and outcome.

4.2 Finding minimal sufficient adjustment sets

Whenever you create a new causal model or make changes to it, DAGitty will calculate all minimal sufficient adjustment sets and display them in the *"Adjustment sets"* field.

4.3 Forcing adjustment for specific covariates

You can also tell DAGitty that you wish a specific covariate to be included into every adjustment set. To do this, move the mouse over the vertex of that covariate and press the *a* key. DAGitty will then update the list of minimal sufficient adjustment sets accordingly – every set displayed is now minimal in the sense that removing any vertex *except those you specified* will render that set insufficient. However, DAGitty will refuse to adjust for a variable that is a successor of the exposure (see above).

4.4 Avoiding adjustment for latent covariates

You can tell DAGitty that a certain variable is latent (unobserved) by moving the mouse over that covariate and pressing the *u* key. DAGitty will only calculates adjustment sets that do not contain latent variables. However, if too many or some important variables are latent, then it may be impossible to close all biasing paths.

5 Acknowledgements

The author wishes to thank Michael Elberfeld, Juliane Hardt, Sven Knüppel, and Sabine Schipf (in alphabetical order) for enlightening discussions about DAGs that made this program possible.

6 Legal notice

7 Bundled libraries

DAGitty ships along with the following JavaScript libraries:

- *Raphaël*, a library for smooth cross-browser vector graphics in SVG and VML, developed by Dmitry Baranovskiy and licensed under the MIT license [2].
- *Prototype.js*, a framework that makes life with JavaScript much easier. Only some parts of Prototype (mainly those focusing on data structures) are included to keep the code small. Developed by the Prototype Core Team and licensed under the MIT license [13].

Furthermore, DAGitty uses some modified code from the *Dracula Graph Library* by Philipp Strathausen, which is also licensed under the MIT license [12].

I am grateful to all authors of these libraries for their valuable work.

8 Bundled examples

DAGitty contains some builtin examples for didactic and illustrative purposes. Some of these examples are taken from published papers or talks given at scientific meetings. These are, in inverse chronological order:

- Polzer et al., 2010 [3]
- Schipf et al., 2010 [8]
- Shrier & Pratt, 2008 [11]
- Sebastiani et al.[4], 2005 [9]
- Aicd & Campos, 1996 [1]

9 Author contact

The author of DAGitty, i.e. me, would be glad to receive feedback from those who use DAGitty in their research or for educational purpose. Also, you are welcome to send me your with suggestions or requests for features that you miss in DAGitty:

Johannes Textor
Institut für Theoretische Informatik
University of Lübeck, Germany

textor@tcs.uni-luebeck.de
www.tcs.uni-luebeck.de/mitarbeiter/textor

References

[1] Silvia Acid and Luis M. De Campos. An algorithm for finding minimum d-separating sets in belief networks. In *Proceedings of the twelfth Conference of Uncertainty in Artificial Intelligence*, pages 3–10, 1996.

[2] Dmitry Baranovskiy. Raphael–javascript library. `http://raphaeljs.com`, 2010.

[3] Ines Polzer et al., 2010. personal communication.

[4] Sven Knüppel and Andreas Stang. DAG program: identifying minimal sufficient adjustment sets. *Epidemiology (Cambridge, Mass.)*, 21(1):159, 2010.

[5] S. L. Laurizen, A. P. Dawid, B. N. Larsen, and H.-G. Leimer. Independence properties of directed markov fields. *Networks*, 20(5):491–505, 1990.

[6] Judea Pearl. *Causality: models, reasoning, and inference*. Cambridge University Press, 2000.

[7] Kenneth J. Rothman, Sander Greenland, and Timothy L. Lash. *Modern Epidemiology*. Wolters Kluwer, 2008.

[8] Sabine Schipf, Robin Haring, Nele Friedrich, Matthias Nauck, Katharina Lau, Dietrich Alte, Andreas Stang, Henry Völzke, and Henri Wallaschofski. Low total testosterone is associated with increased risk of incident type 2 diabetes mellitus in men: Results from the study of health in pomerania (SHIP). *The Aging Male*, 2010. in press.

[4] The example actually shows only a small part of their DAG.

[9] P. Sebastiani, M. F. Ramoni, V. Nolan, C. T. Baldwin, and M. H. Steinberg. Genetic dissection and prognostic modeling of overt stroke in sickle cell anemia. *Nat. Genet.*, 37:435–440, Apr 2005.

[10] Ilya Shpitser, Tyler VanderWeele, and James Robins. On the validity of covariate adjustment for estimating causal effects. In *Proceedings of UAI 2010*, pages 527–536. AUAI Press, 2010.

[11] Ian Shrier and Robert W. Platt. Reducing bias through directed acyclic graphs. *BMC Medical Research Methodology*, 8(70), 2008.

[12] Philipp Strathausen. Dracula graph layout and drawing framework. `http://www.graphdracula.net`, 2010.

[13] Prototype Core Team. Prototype. `http://www.prototypejs.org`, 2010.

[14] Jin Tian, Azaria Paz, and Judea Pearl. Finding minimal d-separators. Technical Report R-254, 1998.