Causal Diagrams Cheat Sheet

Nick Huntington-Klein

April 6, 2019

Causal Diagrams

- Causal diagrams are a set of *variables* and a set of assumptions about *how those variables cause each other*
- Causal diagrams are our way of representing how we think the *data was generated*. They are our *model of the world*
- Once we have our diagram written down, it will tell us how we can *identify* the effect we're interested in
- Writing a causal diagram requires us to make some assumptions - but without assumptions you literally can't get anywhere! Just hopefully make them reasonable assumptions
- "`X` causes `Y`" means that if we could reach in and change the value of `X`, then the *distribution* of `Y` would change as a result. It doesn't necessarily mean that *all* `Y` is because of `X`, or that changing `X` will *always* change `Y`. It just means that a change in `X` will cause the distribution of `Y` to change.

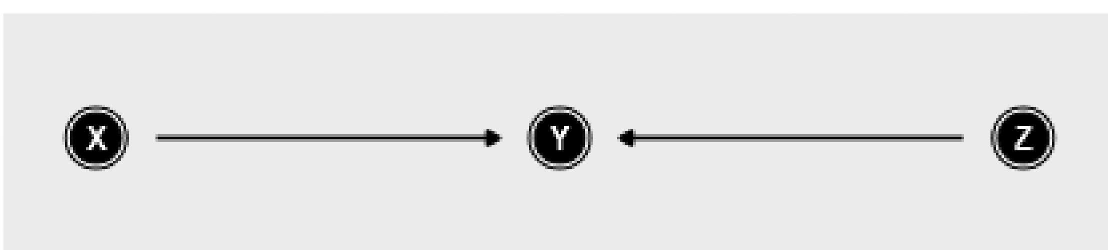
Components of a Causal Diagram

A causal diagram consists of (a) a set of variables, and (b) causal arrows between those variables.

An arrow from `X` to `Y` means that `X` causes `Y`. We can represent this in text as `X -> Y`. Arrows always go one direction. You can't have `X -> Y` and `X <- Y`. See "Tricky Situations" below if it feels like both variables cause each other.

If a variable is *not in* the causal diagram, that means that we're assuming that the variable is not important in the system, or at least not relevant to the effect we're trying to identify.

If an arrow is *not in* the causal diagram, that means that we're assuming that neither variable causes the other. This is not a trivial assumption. In the below diagram, we assume that `X` does not cause `Z`, and that `Z` does not cause `X`.



Building a Diagram

Steps to building a diagram:

1. Consider all the variables that are likely to be important in the data generating process (this includes variables you can't observe)
2. For simplicity, combine them together or prune the ones least likely to be important
3. Consider which variables are likely to affect which other variables and draw arrows from one to the other

Front and Back Doors

Once you have a diagram and you know that you want to identify the effect of `X` on `Y`, it is helpful to make a list of all the front and back doors from `X` to `Y`, and determine whether they are open or closed.

To do this:

1. Write down the list of *all paths* you can follow on your diagram to get from `X` to `Y`
2. If that path only contains arrows that point *from* `X` *towards* `Y`, like `X -> Y` or `X -> A -> Y`, that is a *front door path*
3. If that path contains any arrows that point *towards* `X`, like `X <- W -> Y` or `X -> Z <- A -> Y`, that is a *back door path*

Open and Closed Paths and Colliders

1. If a path from `X` to `Y` contains any *colliders* where two arrows point at the same variable, like how both arrows point to `Z` in `X -> Z <- A -> Y`, then that path is *closed* by default since `Z` is a collider on that path
2. Otherwise, that path is *open* by default
3. If you control/adjust for a variable, then that will *close* any open paths it is on where that variable **isn't** a collider. So, for example, if you control for `W`, the open path `X <- W -> Y` will close
4. If you control/adjust for a variable, then that will *open* any closed back door paths it is on where that variable **is** a collider. So, for example, if you control for `Z`, the closed path `X -> Z <- A -> Y` will open back up. You can close it again by controlling for another non-collider on that path, like `A` in this case

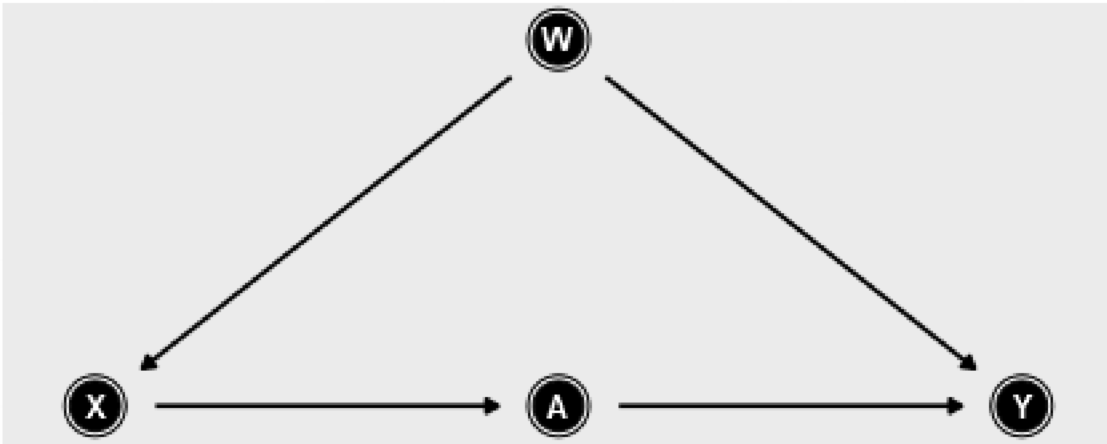
Identifying with the Front Door Method

Sometimes, you can identify the effect of `X` on `Y` by focusing on front doors.

For example, in the below diagram, there are no open back doors from `X` to `A` (there's a back door `X <- W -> Y <- A`, but it is closed since `Y` is a collider on this path), and so you can identify `X -> A`.

Similarly, the only back door from `A` to `Y` is `A <- X <- W -> Y`, and so if you control for `X`, you can identify `A -> Y`.

By combining `X -> A` and `A -> Y` you can identify `X -> Y`.

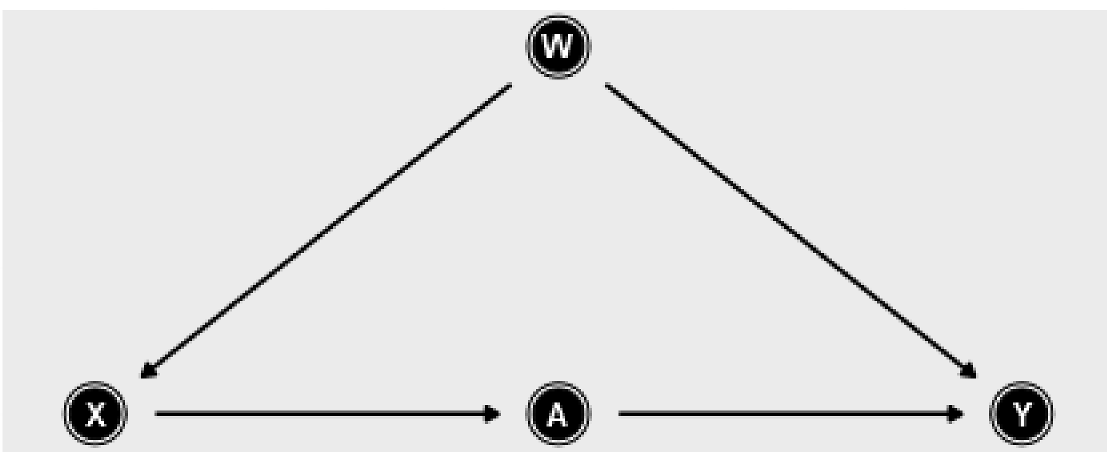


A common issue with this method is finding places where you can apply it convincingly. Are there really no open back door paths from `X` to `A`, or from `A` to `Y` in the true data-generating process?

Identifying with the Back Door Method

Once you've identified the full list of front-door paths and back-door paths, if you control/adjust for a set of variables that *fully blocks* all back door paths (without blocking your front door paths), then you have identified the effect of `X` on `Y`.

For example, in the below diagram, we have the front door path `X -> A -> Y` and the back door path `X <- W -> Y`. If we control for `W` but not `A`, then we have identified the effect of `X` on `Y`.



A common issue with this method is that often you will need to control for variables that you can't actually observe. If you don't have data on `W`, then you can't identify the effect of `X` on `Y` using the back door method.

Testing Implications

Causal diagrams require assumptions to build, and they also imply certain relationships. Sometimes you can test these relationships, and if they're not right, that suggests that you might not have the right model.

For example, in the diagram used in the previous two sections, `W` and `A` should be unrelated if you control for `X`. Also, `X` and `Y` should be unrelated if you control for `W` and `A`. You could test these predictions yourself.

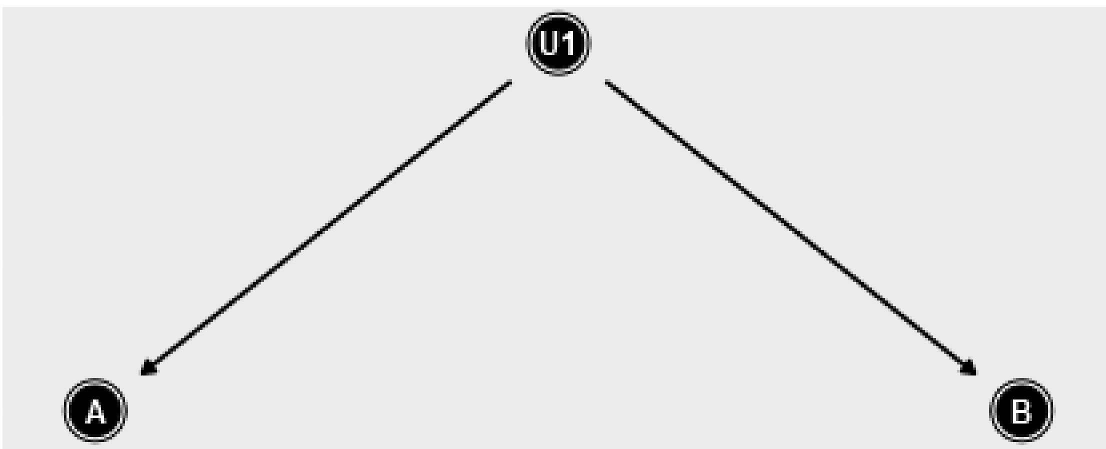
If you control for `X` and still find a strong relationship between `W` and `A`, or if you control for `W` and `A` and still find a strong relationship between `X` and `Y`, then your causal diagram is wrong.

Tricky Situations

Some situations make it difficult to think about how to construct a causal diagram. We'll cover those here.

(1) Lots of variables. In the social sciences, the true data-generating process may have lots and lots of variables, perhaps hundreds! It's important to remember to (a) Remove any vairables that are likely to only be of trivial importance, and (b) if you have multiple variables that lie on identical-looking paths, you can combine them. For example, you can often combine "race", "gender", "parental income", "birthplace" etc. into the single variable "background"

(2) Variables that are related but don't cause each other. Often we have two variables that are know are *correlated*, but we don't think either causes the other. For example, people buy more ice cream on days they wear shorts. In these cases, we include a "common cause" on the graph. We don't necessarily need to be specific about what it is, or if it's measured or unmeasured. I usually label these "U1" "U2" etc.



Note that some people will use a double headed arrow here instead, like `A <-> B`. It means the same thing. We won't be doing this in our class though.

(3) Variables that cause each other. A variable should never be able to cause itself. So, a causal diagram should never have a *loop* in it. The simplest example of a loop is if `A -> B` and `B -> A`. If we mix these together we have `A -> B -> A`. `A` causes itself! That doesn't make sense. This is why causal diagrams are technically called "directed *acyclic* graphs".

But surely in the real world there are plenty of variables with self-feedback loops, or variables which cause each other. So what do we do?

In these cases, you want to think about *time* carefully. Instead of `A -> B` and `B -> A`, split `A` and `B` up into two variables each: `ANow` and `ALater`, and `BNow` and `BLater`. Obviously, a causal effect needs time to occur. Now we have `ANow -> BLater` and `BNow -> ALater` and the loop is gone.