DAGs and Causal Inference for Human-Wildlife Conflict in TZA

Introduction to Causal Inference

this is Brendan's crash course primer on Causal Inference (CI) and DAG's as it related to multiple regression and the TZA Wildlife conflict paper. We'll do a primer and then draw some DAGS. I advise installing the `dagitty` package.

Background

A primary aim of the scientific enterprise is to infer causal effects of predictors on outcome variables of inference, so we can understand how systems function. This also can help folks working in applied contexts make informed interventions, such as mitigating human-wildlife conflict. Well-designed experiments are one typical approach to understand causality, but in many cases, like the study presented in this paper, experiments would be infeasible or unethical. Many common approaches in statistical inference, such as multivariate regression, do not make any claims about causality, and statistical information flows directly between outcome variable and predictors. Researchers are often concerned about the effect of predictor, X, on an outcome variable, Y. However, X may be correlated with another covariate(s) of interest, Z, which can confound the relationship between X and Y. To infer the relationship between X and Y, researchers will often add covariates like Z (and often times many others) to control for potential covariates. A common term in ecology papers is to ''control for seasonality" or ''control for environmental effects."

Confounding factors are a real, and valid concern, but whether or not to include a variable in a multivariate regression depends on the causal relationships between measureable variables of interest, and any potential unobserved variables. In some cases, including covariate Z can reduce the precision of an estimate of the effect of X on Y or render it entirely unreliable if Z is a collider (where X and Y both cause Z).

What's a DAG

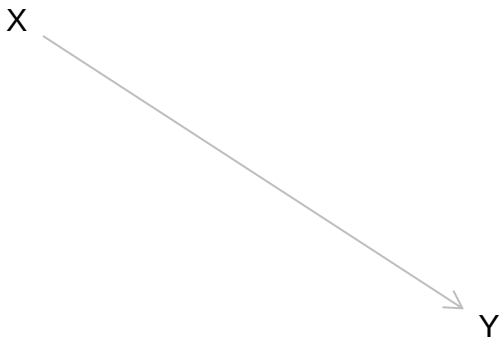
DAG stands for directed acyclical graph. DAGs and CI are a topic seperate from, but related to statistical inference. When it comes to regression, these models do not imply causality. Information flows freely between variables of interst. However, if we aim to make inference about causal relationships, we need to close any backdoor paths through which information may flow in our DAG to assess the true effect of X on Y.

Drawing a DAG

To draw a DAG, we first consider all of the variable of interest in the system. We typically want to known the effect of a treatment/predictor/exposure on an outcome variable. If we think X, our predictor, directly causes Y, we draw an arrow from X to Y like so:

```
playdag <-
  plot(dagitty('dag {X -> Y}'))
```

```
## Plot coordinates for graph not supplied! Generating coordinates, see ?coordinates for how to set you
```



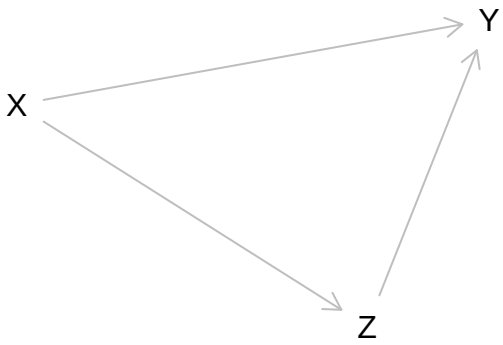
This arrow implies a direct causal relationship between X and Y. What that means is the natural process determining Y is *directly influenced* by the status of X. Altering X via external intervention would also alter Y. However, an arrow X → Y only represents that part of the causal effect which is not mediated by any of the other variables in the diagram. If you are sure X does not direcly mediate Y, you can exclude an arrow. We must also ensure that X preceeds Y, as causes must come before effects. In instances where this is not the case, and there are bidirectional arrows between X and Y we violate this assumption and need an experiment or time series of treatments on outcomes. CI folks also think the omission of an arrow is a stronger claim that the inclusion of an arrow.

Pipes as confounds

Variables can also indirectly influence inference. Lets posit the following DAG:

```
playdag <-
  plot(dagitty('dag {
  X->Y
  X->Z
  Z->Y
                    }'))
```

```
## Plot coordinates for graph not supplied! Generating coordinates, see ?coordinates for how to set your
```



In this case there is a direct path of X -> Y. However, there is also an indirect, aka "backdoor" path of X -> Z -> Y. This indirect path is called a "pipe." If we wish to make inference about X causing Y, we must condition on Z in our regression to close the backdoor path. This means include it as a predictor. In `lm` syntax this means `Y ~ X + Z`.

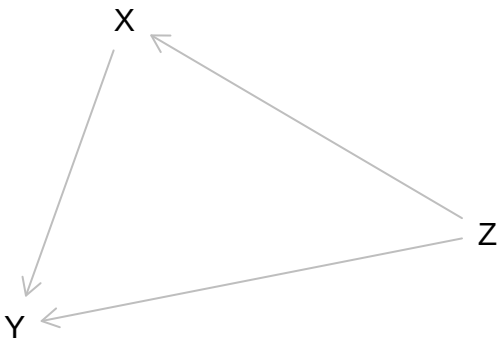
Forks as confounds

There exists another confound, commonly encountered called a fork. This is the classic confound, researchers think the mean about when they say control for something. It means that some variable Z directly causes both the predictor X and the outcome Y. Z might be some varaible like, occuring in the the same location. Draw the DAG, Clarice.

```
playdag <-
  plot(dagitty('dag {
```

```
  X->Y
  X<-Z
  Z->Y
                    }'))
```

```
## Plot coordinates for graph not supplied! Generating coordinates, see ?coordinates for how to set you
```



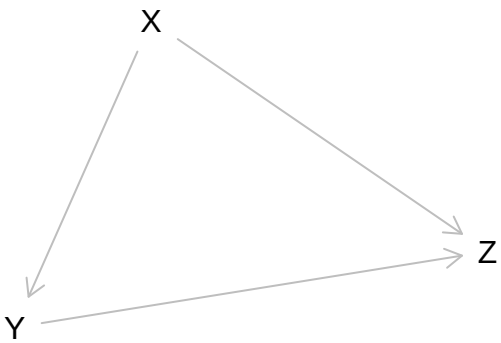
One concern we did not address, is **unobserved variables**. For example, if we did not observe or measure Z, and the above DAG is true that means we cannot make a reliable inference about the direct causal effect of X on Y. Thus it is imoportant to include important causal data you did not collect in your DAG.

Colliders

Lastly we have colliders. This means X and Y both directly cause Z. DAG me.

```
playdag <-
  plot(dagitty('dag {
  X->Y
  X->Z
  Z<-Y
                    }'))
```

```
## Plot coordinates for graph not supplied! Generating coordinates, see ?coordinates for how to set you
```




```
DAGGGG <- dagitty('dag {
  X->Y
  X->Z
  Z<-Y
                    }')
isCollider(DAGGGG , "X" , "Z","Y")
```

```
## [1] TRUE
```

On any causal pathway, anything where 2 arrows enter is a collider. Unlike pipes and forks, we do not ever want to condition on colliders, as this opens the back door path, and allows information to flow, thus altering

inference. ### more There is a bit more we could cover, like descendents, but we will postpone that for now. Our goal for our question is to DAG out all related variables of interest in our system. If we want to see how X causes Y, or livestock head causes conflict, we then trace the direct and backdoor paths from X to Y, and condition on the necessary variables to close backdoor paths. Sometimes ew can, and in some causal diagrams we can't. N

Now lets draw DAGs for our paper. I think it is important we all agree on a DAG or 2 before we rerun analyses.

DAGS for livestock

First we will do one where we do not have the guards double arrow. As that shoul be simpler. You can also do this on daggity.com to save and work with tis in a GUI or expor LaTeX code.

First we can draw causal relationships between each variable, and write down a justification.

One on the advantages of `dagitty` package ia that we can use it to identify the adjustment sets. That is the minimum things we need to adjust for to make an infrence about the relationship between X (i.e. building distance) and Y (livestock conflict).

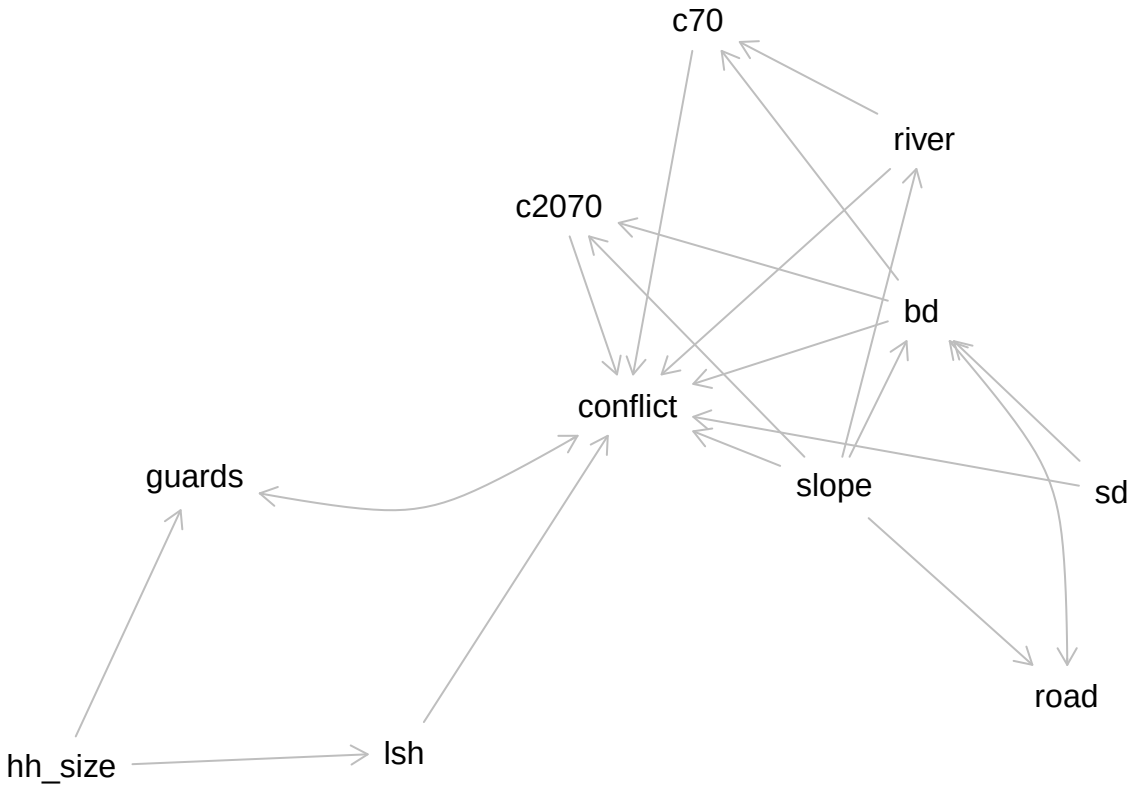
First DAG, livestock conflict.

This is the one that we are going to use for the paper Please check to make sure all justifications make sense, and see if we missed anything.

```
ls_conf_yes_guard <-
  dagitty('dag {
  c2070 -> conflict
  bd -> conflict
  bd <-> road
  c70 -> conflict
  hh_size -> guards
  hh_size -> lsh
  lsh -> conflict
  river -> c70
  river -> conflict
  sd -> bd
  sd -> conflict
  bd -> c70
  bd -> c2070
  guards <-> conflict
  slope -> bd
  slope -> conflict
  slope -> c2070
  slope -> river
  slope -> road
}')

plot(ls_conf_yes_guard)
```

```
## Plot coordinates for graph not supplied! Generating coordinates, see ?coordinates for how to set you
```



1) c2070 -> conflict: c2070 is refuge habitat for carnivores who live there more and use c2070 as refuge to avoid detection and forage at night.
2) bd -> conflict: building density signifies human presence, and carnivores avoiddense areas.
3) bd <-> road: People will build settlements along roads due to access. It is less certain that building density also causes roads, but possible tertiary roads and smaller roads get built to connect dense places.
4) c70 -> conflict: c70 is refuge habitat for predators, i.e. riparian zone and dense thickets. Loss of c70, causes loss of habitat and can increase conflict.
5) hh_size -> lsh: Larger households are often mutli-generational, which means they have more capital which is often invested in cattle
6) hh_size -> guards: The more people in the house, the more there are avaiable to act as guards.
7) lsh -> conflict: The greater number of cattle there are to eat, the more likely they will be predated.
8) river -> c70: Rivers are habitat for vegataion communites calssified as c2070.
9) river -> conflict: Animals need water to live, may raid more when less water available.
10) sd -> bd: laws in place to prevent people from buiding within a 500m buffer of reserve, so they build elsewhere. People prefer to build away from PA, and really only do so out of necessity
11) sd -> conflict: some species prefer to be in PA, and are only likely raid farms close to boundaries. Others venture out to raid farms.
12) bd -> c70: construction of buildings causes loss in c70 and changes classification probability
13) bd -> c2070: construction of buildings causes loss in c2070 and changes classification probability
14) guards <-> conflict: Guards in theory reduce conflict if effective. That is their point. However, due to conflict, farmers may be more inclined to hire guards. To break this bidirectional arrow, one could randomly apply numbers of guards to people's herds, prevenrnt them from changing it, and measure conflict. However, this is unethical. Instead, one would need to measure conflict levels, or amount of crop loss, as a function of the number of guards introduced, or used at each timestep.
15) slope -> bd: More houses are built on less hilly land for ease of construction and material transport.
16) slope -> conflict: Some predators travel in hillier areas
17) slope -> c2070: c2070 is more likely on hillsides either dur to the difficulty require in cutting it down, or ecological conditions.
18) slope -> rivers: water flows down hills and is likely to be in places with decreasing slopes.

19) slope -> road: slope influences where roads are built, the are preferntially built in easier, less hilly places and lower mountain passes.

Now we can run the adjustment sets.

###this will tell us what are the minimal things we need to condition on to make inferences about the relationship between items

```
adjustmentSets( ls_conf_yes_guard , exposure="c2070" , outcome="conflict" , type="canonical") #independ
```

```
## { bd, c70, hh_size, lsh, river, sd, slope }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="c2070" , outcome="conflict" , type="minimal") #independen
```

```
## { bd, slope }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="c70" , outcome="conflict" ) #account for river
```

```
## { bd, river }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="c70" , outcome="conflict", type="canonical" ) #account fo
```

```
## { bd, c2070, hh_size, lsh, river, sd, slope }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="lsh" , outcome="conflict" ) #account for hh_size
```

```
##  {}
```

```
adjustmentSets( ls_conf_yes_guard , exposure="lsh" , outcome="conflict" , type="canonical") #account fo
```

```
## { bd, c2070, c70, hh_size, river, sd, slope }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="river" , outcome="conflict" )
```

```
## { slope }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="river" , outcome="conflict" , type="canonical")
```

```
## { bd, c2070, hh_size, lsh, sd, slope }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="sd" , outcome="conflict" )
```

```
##  {}
```

```
adjustmentSets( ls_conf_yes_guard , exposure="sd" , outcome="conflict" , type="canonical")
```

```
## { hh_size, lsh, river, slope }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="bd" , outcome="conflict" )
```

```
## { sd, slope }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="bd" , outcome="conflict" , type="canonical")
```

```
## { hh_size, lsh, river, sd, slope }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="slope" , outcome="conflict" )
```

```
##  {}
```

```
adjustmentSets( ls_conf_yes_guard , exposure="slope" , outcome="conflict", type="canonical" )
```

```
## { hh_size, lsh, sd }
```

```
adjustmentSets( ls_conf_yes_guard , exposure="guards" , outcome="conflict" )
adjustmentSets( ls_conf_yes_guard , exposure="guards" , outcome="conflict", type="canonical" )
```

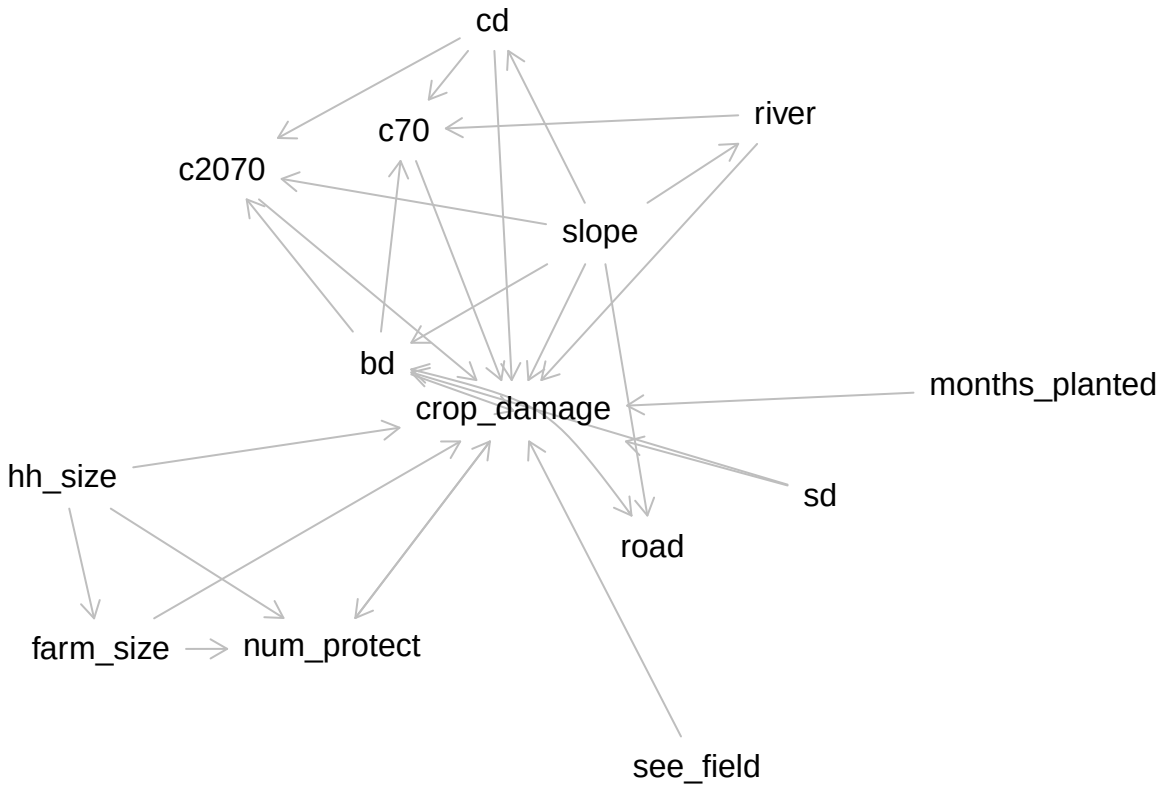
Note the last adjustment set. There is no output. That is because we can't make any reliable inference about guards affecting conflict given our DAG. We need to design data collection or a study where this is a single arrow, or a different dag is implied. Double arrows typically mean we need to break apart the timescale. Guards cause conflict in that they in theory reduce it. Conflict causes guards because people may get more guards if they have conflict. We need data that separates these two.

```
## Second DAG: Crop damage


```r
###crop damage
crop_damage_dag <-
  dagitty('dag {
  c2070 -> crop_damage
  c70 -> crop_damage
  river -> c70
  river -> crop_damage
  months_planted -> crop_damage
  farm_size -> crop_damage
  farm_size -> num_protect
  num_protect -> crop_damage
  crop_damage -> num_protect
  hh_size -> num_protect
  hh_size -> farm_size
  hh_size -> crop_damage
  see_field -> crop_damage
  road <-> bd
  bd -> crop_damage
  bd -> c2070
  bd -> c70
  sd -> bd
  sd -> crop_damage
  cd -> c70
  cd -> c2070
  cd -> crop_damage
  slope -> bd
  slope -> crop_damage
  slope -> c2070
  slope -> river
  slope -> road
  slope -> cd
  }')

plot(crop_damage_dag)
```

```
## Plot coordinates for graph not supplied! Generating coordinates, see ?coordinates for how to set you
```
```



1) c2070 -> crop_damage: c2070 is habitat refuge for crop raiders. More habitat could mean there are more places to hide, or less habitat could mean that they are forced to raid crops more.
2) c70 -> crop_damage: c70 is habitat refuge for crop raiders. More habitat could mean there are more places to hide, or less habitat could mean that they are forced to raid crops more.
3) river -> c70: the preence of water in rivers creates conditions where c70 classified lands grow.
4) river -> crop_damage: Animals dwell near rivers, and are likely to cause conflict at places near them as a consequence.
5) months_planted -> crop_damage : The more time there are crops in the field, the more likely conflict will be observed,
6) farm_size -> crop_damage: That larger the farm, the more available crops are, and the more likely they will get damaged.
7) farm_size -> num_protect #farm size influences the type of protection, which influences the number of strategies used. this is really indirect.
8) num_protect -> crop_damage: Using a range of strategies may reduce crop raiding.
9) crop_damage -> num_protect: desperate farmers with crop damage may try lots of new crop strategies out of desperation.
10) hh_size -> num_protect: larger households engage more effor in protection
11) hh_size -> farm_size:more avialble people to work fields, makes it possible to have a larger farm
12) hh_size -> crop_damage: Animals avoid field with more human activity
13) farm_size -> num_protect: larger farms employ more protection strategies, particularly things like fences etc. that do not require folks standing there
14) see_field -> crop_damage: farmers that see their field can react quickly and minimize damage
15) road <-> bd: People will build settlements along roads due to access. It is less certain that building density also causes roads, but possible tertiary roads and smaller roads get built to connect dense places.
16) bd -> crop_damage: building density attracts and deters different crop raiders (i.e. vervets vs. elephants)
17) bd -> c2070: construction of buildings causes loss in c2070 and changes classification probability
18) bd -> c70: construction of buildings causes loss in c70 and changes classification probability
19) sd -> bd: Cities expand toawrd protected areas, less dense at edges. 500m buffer & preservation mean that density is lower right next to PA.

20) sd -> crop_damage: Different animals have different risk tolerances, some venture far from PA.
21) cd -> c70: increased crop density and land conversion means there is less likely to be c70.
22) cd -> c2070: increased crop density and land conversion means there is less likely to be c2070.
23) cd -> crop_damage: more beneficial to raid areas with a higher density of crops.
24) slope -> bd: More houses are built on less hilly land for ease of construction and material transport.
25) slope -> crop_damage: elephants don't like traveling on hills, so less likely to damage farms on slopes.
26) slope -> c2070: 2070 is more likely on hillsides either dur to the difficulty require in cutting it down, or ecological conditions.
27) slope -> river: water flows down hills and is likely to be in places with decreasing slopes.
28) slope -> road: slope influences where roads are built, the are preferntially built in easier, less hilly places and lower mountain passes.
29) slope -> cd: crops are more densley planted in flat areas (less runoff, easier to plant things close together)

Now lets look at the adjustment sets.

```
adjustmentSets( crop_damage_dag , exposure="c2070" , outcome="crop_damage" , type="canonical") #indepen
```

```
## { bd, c70, cd, farm_size, hh_size, months_planted, river, sd,
##   see_field, slope }
```

```
adjustmentSets( crop_damage_dag , exposure="c2070" , outcome="crop_damage" , type="minimal") #independe
```

```
## { bd, cd, slope }
```

```
adjustmentSets( crop_damage_dag , exposure="c70" , outcome="crop_damage" ) #account for river
```

```
## { bd, cd, river }
```

```
adjustmentSets( crop_damage_dag , exposure="c70" , outcome="crop_damage", type="canonical" ) #account f
```

```
## { bd, c2070, cd, farm_size, hh_size, months_planted, river, sd,
##   see_field, slope }
```

```
adjustmentSets( crop_damage_dag , exposure="cd" , outcome="crop_damage" ) #account for hh_size
```

```
## { slope }
```

```
adjustmentSets( crop_damage_dag , exposure="cd" , outcome="crop_damage" , type="canonical") #account fo
```

```
## { bd, farm_size, hh_size, months_planted, river, sd, see_field, slope }
```

```
adjustmentSets( crop_damage_dag , exposure="river" , outcome="crop_damage" )
```

```
## { slope }
```

```
adjustmentSets( crop_damage_dag , exposure="river" , outcome="crop_damage" , type="canonical")
```

```
## { bd, c2070, cd, farm_size, hh_size, months_planted, sd, see_field,
##   slope }
```

```
adjustmentSets( crop_damage_dag , exposure="sd" , outcome="crop_damage" )
```

```
##  {}
```

```
adjustmentSets( crop_damage_dag , exposure="sd" , outcome="crop_damage" , type="canonical")
```

```
## { cd, farm_size, hh_size, months_planted, river, see_field, slope }
```

```
adjustmentSets( crop_damage_dag , exposure="bd" , outcome="crop_damage" )
```

```
## { sd, slope }
```

```
adjustmentSets( crop_damage_dag , exposure="bd" , outcome="crop_damage" , type="canonical")
```

```
## { cd, farm_size, hh_size, months_planted, river, sd, see_field, slope }
```

```
adjustmentSets( crop_damage_dag , exposure="months_planted" , outcome="crop_damage" )
```

```
##  {}
```

```
adjustmentSets( crop_damage_dag , exposure="months_planted" , outcome="crop_damage" , type="canonical")
```

```
## { bd, c2070, c70, cd, farm_size, hh_size, river, sd, see_field, slope }
```

```
adjustmentSets( crop_damage_dag , exposure="see_field" , outcome="crop_damage" )
```

```
##  {}
```

```
adjustmentSets( crop_damage_dag , exposure="see_field" , outcome="crop_damage" , type="canonical")
```

```
## { bd, c2070, c70, cd, farm_size, hh_size, months_planted, river, sd,
##   slope }
```

```
adjustmentSets( crop_damage_dag , exposure="num_protect" , outcome="crop_damage" )
adjustmentSets( crop_damage_dag , exposure="num_protect" , outcome="crop_damage" , type="canonical")

adjustmentSets( crop_damage_dag , exposure="hh_size" , outcome="crop_damage" )
```

```
##  {}
```

```
adjustmentSets( crop_damage_dag , exposure="hh_size" , outcome="crop_damage" , type="canonical")
```

```
## { bd, c2070, c70, cd, months_planted, river, sd, see_field, slope }
```

```
adjustmentSets( crop_damage_dag , exposure="farm_size" , outcome="crop_damage" )
```

```
## { hh_size }
```

```
adjustmentSets( crop_damage_dag , exposure="farm_size" , outcome="crop_damage" , type="canonical")
```

```
## { bd, c2070, c70, cd, hh_size, months_planted, river, sd, see_field,
##   slope }
```

```
adjustmentSets( crop_damage_dag , exposure="slope" , outcome="crop_damage" )
```

```
##  {}
```

```
adjustmentSets( crop_damage_dag , exposure="slope" , outcome="crop_damage" , type="canonical")
```

```
## { farm_size, hh_size, months_planted, sd, see_field }
```

Now we can look at all of the direct arrows to estimate the effect of X on Y, and determine which covariates to include in the models relevant to the predictor of interest.

For c2070 the minimal model `m_c2070_min` includes:

```
 adjustmentSets( crop_damage_dag , exposure="c2070" , outcome="crop_damage" , type="minimal") #independ
```

```
## { bd, cd, slope }
```

while the canonical model, `m_c70_c2070_can` includes:

```
 adjustmentSets( crop_damage_dag , exposure="c2070" , outcome="crop_damage" , type="canonical") #indepe
```

```
## { bd, c70, cd, farm_size, hh_size, months_planted, river, sd,
##   see_field, slope }
```

For c70 the minimal model `m_c70_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="c70" , outcome="crop_damage" , type="minimal")
```

```
## { bd, cd, river }
```

For c70 the canonical model `m_c70_c2070_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="c70" , outcome="crop_damage", type="canonical" )
```

```
## { bd, c2070, cd, farm_size, hh_size, months_planted, river, sd,
##   see_field, slope }
```

For cd the minimal model `m_cd_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="cd" , outcome="crop_damage" , type="minimal")
```

```
## { slope }
```

For cd the canonical model `m_cd_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="cd" , outcome="crop_damage", type="canonical" )
```

```
## { bd, farm_size, hh_size, months_planted, river, sd, see_field, slope }
```

For river the minimal model `m_riv_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="river" , outcome="crop_damage" )
```

```
## { slope }
```

For river the minimal model `m_riv_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="river" , outcome="crop_damage" , type="canonical")
```

```
## { bd, c2070, cd, farm_size, hh_size, months_planted, sd, see_field,
##   slope }
```

For settlement distance the minimal model `m_sd_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="sd" , outcome="crop_damage" )
```

```
##  {}
```

It requires no other covariates.

For settlement distance the canonical model `m_sd_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="sd" , outcome="crop_damage" , type="canonical")
```

```
## { cd, farm_size, hh_size, months_planted, river, see_field, slope }
```

For building density the minimal model `m_bd_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="bd" , outcome="crop_damage" )
```

```
## { sd, slope }
```

For building density the canonical model `m_sd_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="bd" , outcome="crop_damage" , type="canonical")
```

```
## { cd, farm_size, hh_size, months_planted, river, sd, see_field, slope }
```

For months planted the minimal model `m_mp_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="months_planted" , outcome="crop_damage" )
```

```
##  {}
```

For months planted the canonical model `m_fs_mp_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="months_planted" , outcome="crop_damage" , type="canonical")
```

```
## { bd, c2070, c70, cd, farm_size, hh_size, river, sd, see_field, slope }
```

For see field the minimal model `m_see_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="see_field" , outcome="crop_damage" )
```

```
##  {}
```

For see field the canonical model `m_see_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="see_field" , outcome="crop_damage" , type="canonical")
```

```
## { bd, c2070, c70, cd, farm_size, hh_size, months_planted, river, sd,
##   slope }
```

For number of protection strategies the minimal model `m_np_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="num_protect" , outcome="crop_damage" )
```

For number of protection strategies the canonical model `m_np_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="num_protect" , outcome="crop_damage" , type="canonical")
```

For household size the minimal model `m_hhs_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="hh_size" , outcome="crop_damage" )
```

```
##  {}
```

For household size the canonical model `m_hhs_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="hh_size" , outcome="crop_damage" , type="canonical")
```

```
## { bd, c2070, c70, cd, months_planted, river, sd, see_field, slope }
```

For farm size the minimal model `m_fs_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="farm_size" , outcome="crop_damage" )
```

```
## { hh_size }
```

For farm size the canonical model `m_fs_mp_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="farm_size" , outcome="crop_damage" , type="canonical")
```

```
## { bd, c2070, c70, cd, hh_size, months_planted, river, sd, see_field,
##   slope }
```

For slope the minimal model `m_slope_min` includes:

```
adjustmentSets( crop_damage_dag , exposure="slope" , outcome="crop_damage" )
```

```
##  {}
```

For slope the canonical model `m_slope_can` includes:

```
adjustmentSets( crop_damage_dag , exposure="slope" , outcome="crop_damage" , type="canonical")
```

```
## { farm_size, hh_size, months_planted, sd, see_field }
```

Note that c2070 and c70 have the same canonical model. Moths planted and farm size have the same canonical model. We cannot estimate the effectivenes of the number of protection strategies on crop damae given our curent data. We need time series.