DAGs and Causal Inference for Wuman-Wildlife Conflict in TZA

Introduction to Causal Inference

this is Brendan's crash course primer on Causal Inference 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.

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

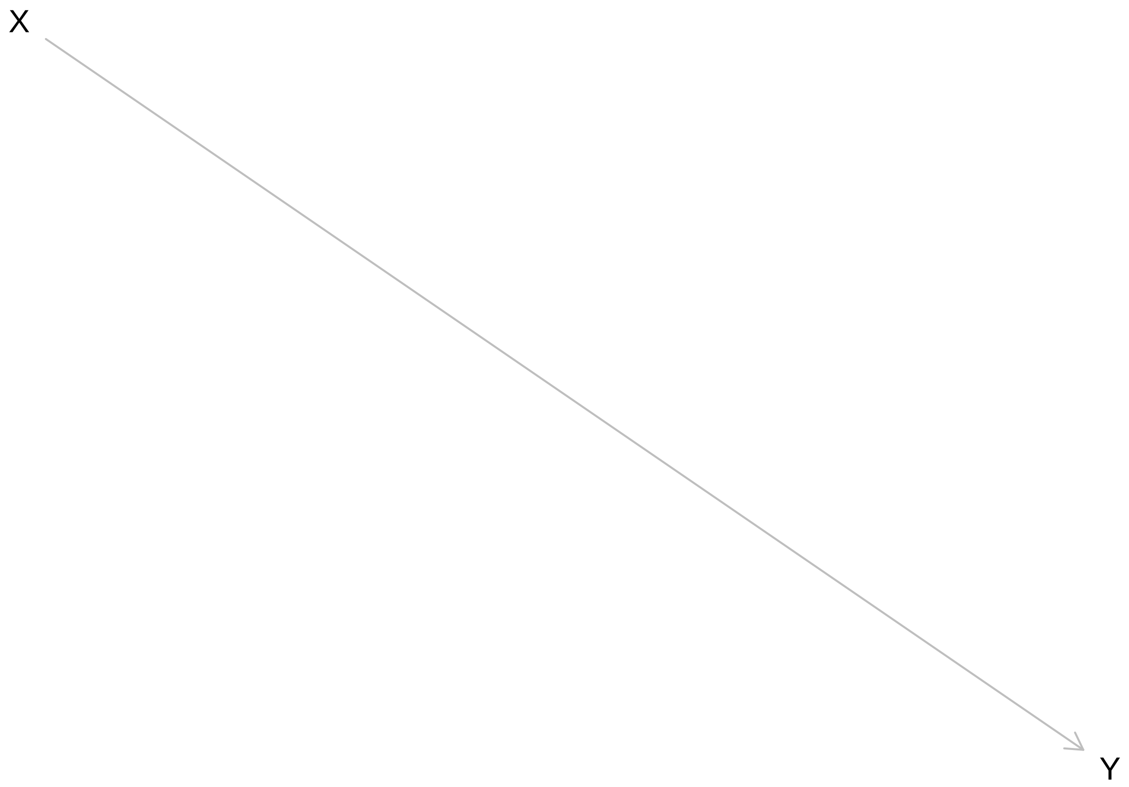
DAGs and CI are a topic seperate from, but related to statistical inference. When it comes to regression, these models do not imply causality. Information flows freely between variables of interst. However, if we aim to make inference about causal relationships, we need to close any backdoor paths through 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
```



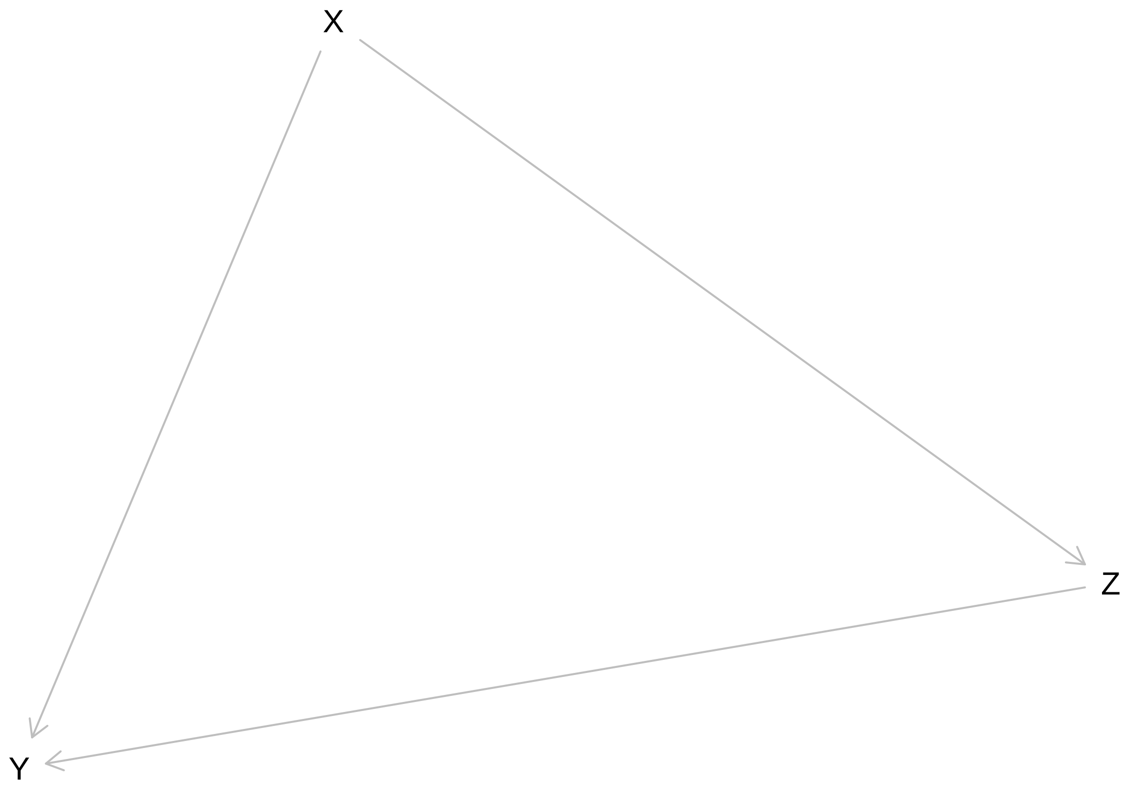
This arrow implies a direct causal relationship between X and Y. What that means is the natural process determining Y is *directly influenced* by the status of X. Altering X via external intervention would also alter Y. However, an arrow X → Y only represents that part of the causal effect which is not 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:

```r
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
```



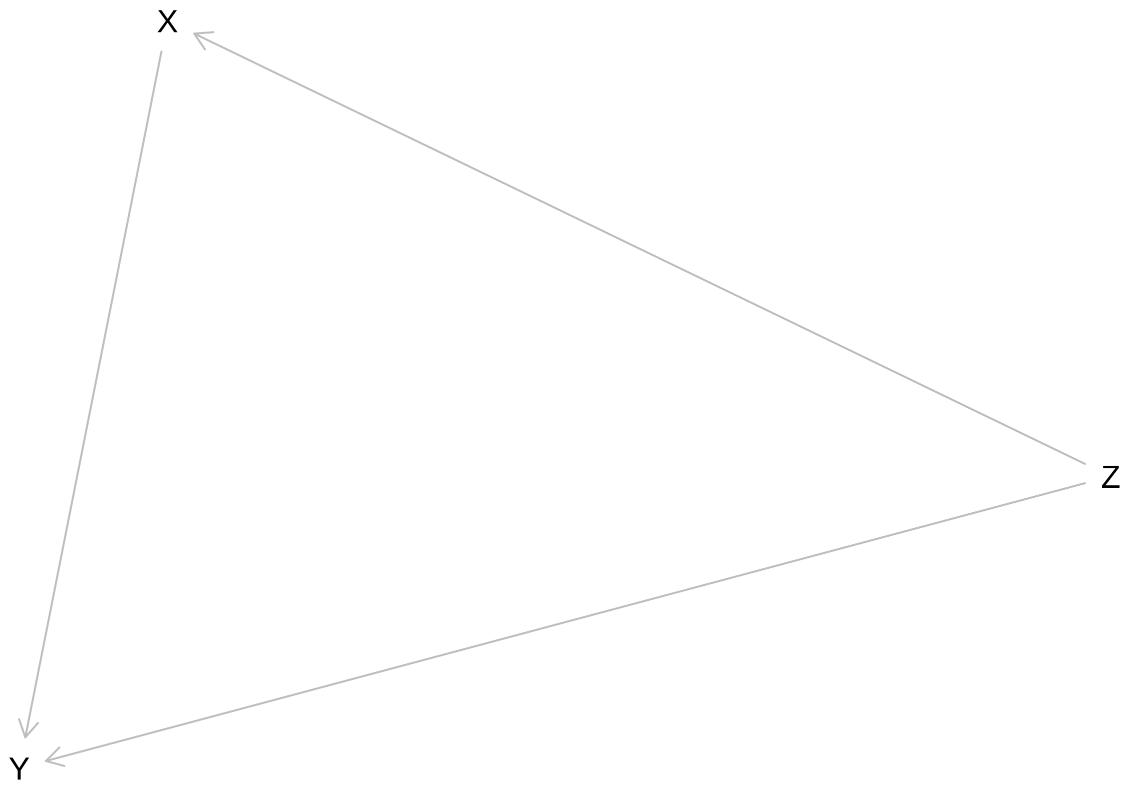
In this case there is a direct path of X -> Y. However, there is also an indirect, aka "backdoor" path of X -> Z -> Y. This indirect path is called a "pipe." If we wish to make inference about X causing Y, we must condition on Z in our regression to close the backdoor path. This means include it as a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



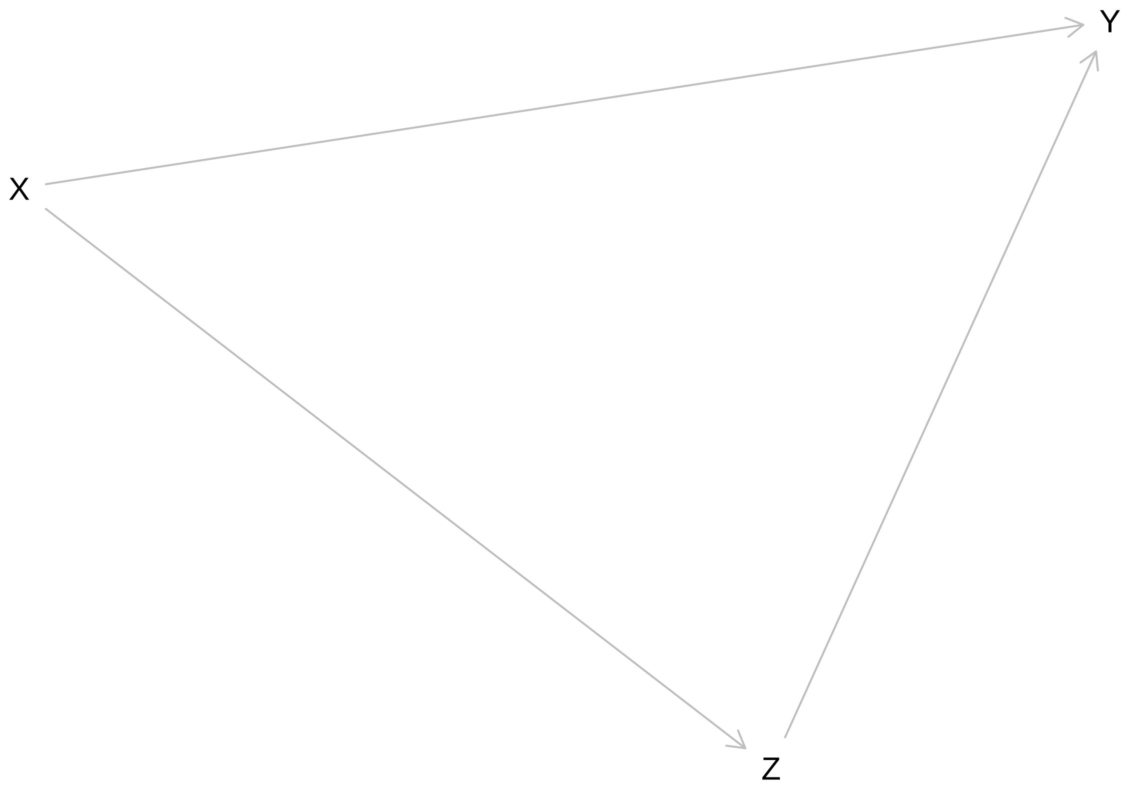
One concern we did not address, is unobserved variable. For example, if we did not observe or measure Z, and the above DAG is true that means we cannot make a reliable inference about the direct causal effect of X on Y. Thus it is imoportant to include important causal data you did not collect in your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

Colliders

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

```r
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
```



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.

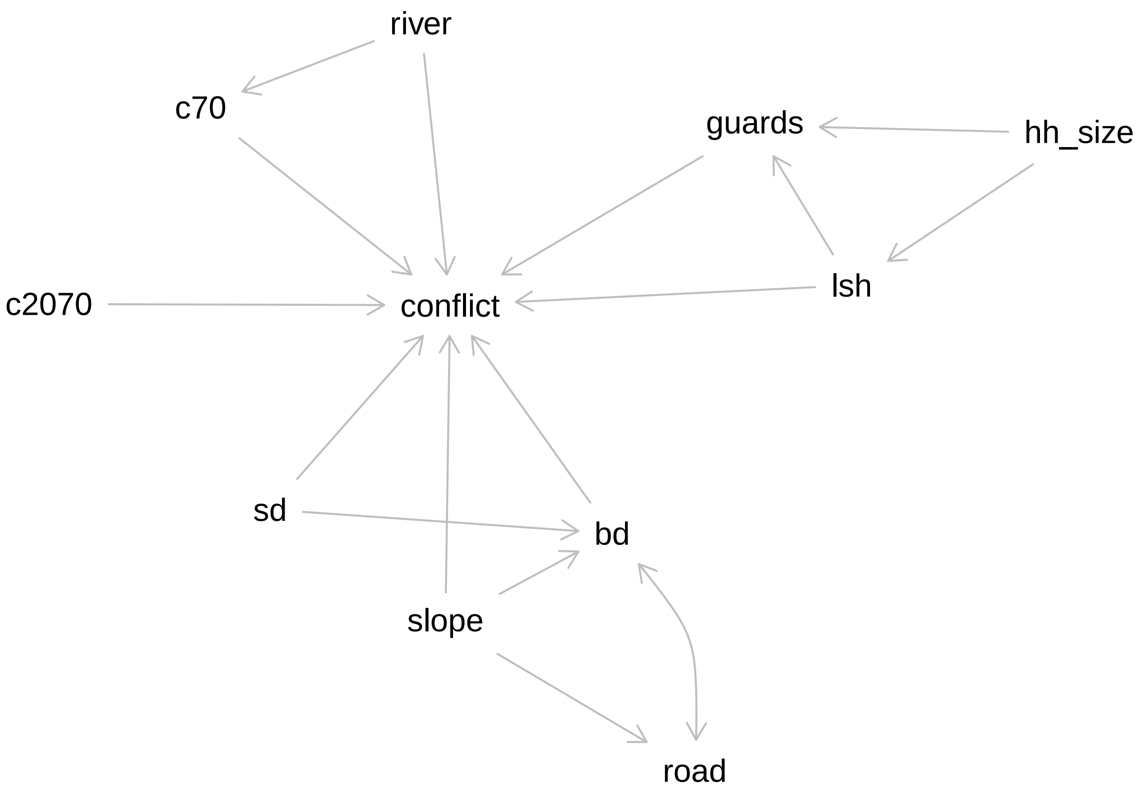
DAGS for livestock

First we will do one where we do not have the guards double arrow.

```
####here is relevant code to DAGs and grumeti
##imply all relationships
ls_conf_no_guard <-
  dagitty('dag {
  c2070 -> conflict
  bd -> conflict
  bd <-> road
  c70 -> conflict
  hh_size -> lsh
  hh_size -> guards
  lsh -> conflict
  lsh -> guards
  river -> c70
  river -> conflict
  sd -> bd
  sd -> conflict
  guards -> conflict
  slope -> bd
  slope -> conflict
  slope -> road
}'
          )
```

Now lets plot the DAG.

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



###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_no_guard , exposure="c2070" , outcome="conflict" , type="canonical") #independe
```

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

```
adjustmentSets( ls_conf_no_guard , exposure="c2070" , outcome="conflict" , type="minimal") #independent
```

```
##  {}
```

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

```
## { river }
```

```
adjustmentSets( ls_conf_no_guard , exposure="c70" , outcome="conflict", type="canonical" ) #account for
```

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

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

```
## { hh_size }
```

```
adjustmentSets( ls_conf_no_guard , exposure="lsh" , outcome="conflict" , type="canonical") #account for
```

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

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

```
##  {}
```

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

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

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

```
##  {}
```

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

```
## { c2070, c70, guards, hh_size, lsh, river, slope }
```

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

```
##  {}
```

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

```
## { c2070, c70, guards, hh_size, lsh, river, sd }
```

```
adjustmentSets( ls_conf_no_guard , exposure="guards" , outcome="conflict" )
```

```
## { lsh }
```

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

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

```
##identifing colliders
isCollider(ls_conf_no_guard , "hh_size" , "guards","conflict") #shows not a collider on path from hh_si
```

```
## [1] FALSE
```

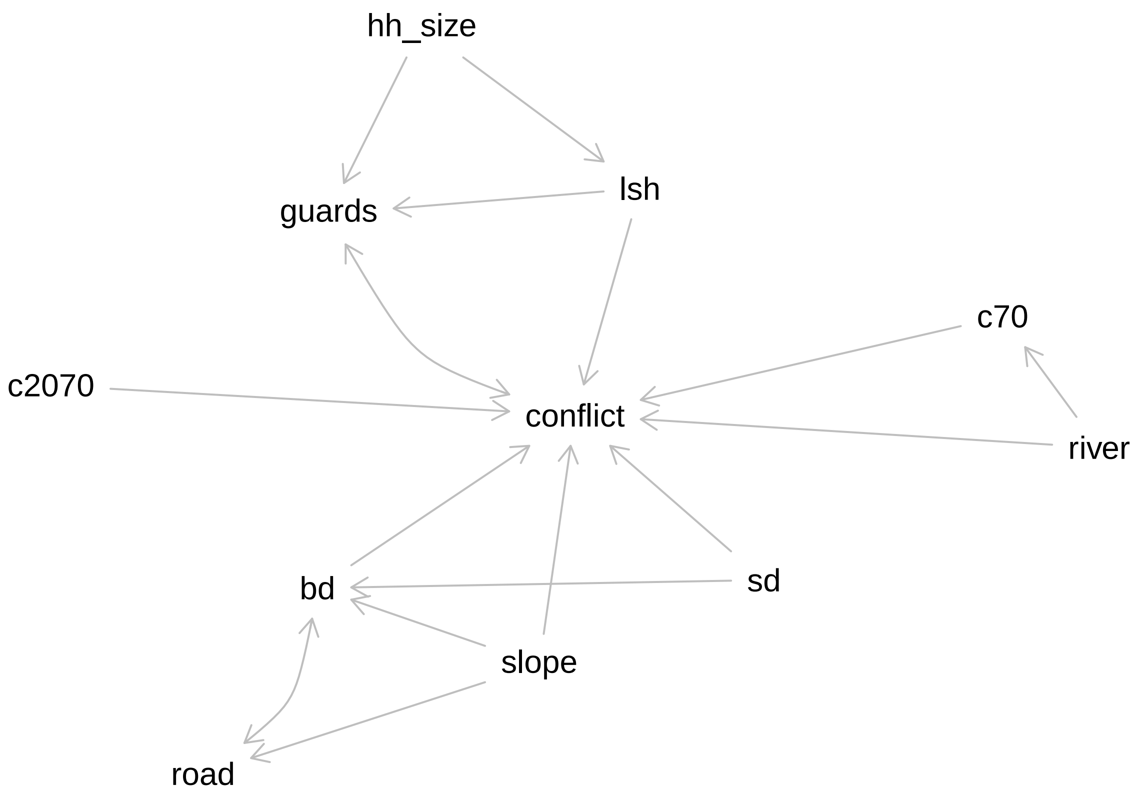
```
isCollider(ls_conf_no_guard , "hh_size" , "guards","lsh") #shows guards is collider on path from hh_siz
```

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

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

plot(ls_conf_yes_guard)
```

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



```
###this will tell us what are the minimal things we need to condition on to make inferences about the r
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
```

```
##  {}
```

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

```
## { 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" )
```

```
##  {}
```

```
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")
```

```
## { c2070, c70, hh_size, lsh, river, slope }
```

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

```
##  {}
```

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

```
## { c2070, c70, hh_size, lsh, river, sd }
```

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

##identifing colliders
isCollider(ls_conf_yes_guard , "hh_size" , "guards","conflict") #shows not a collider on path from hh_s
```

```
## [1] FALSE
```

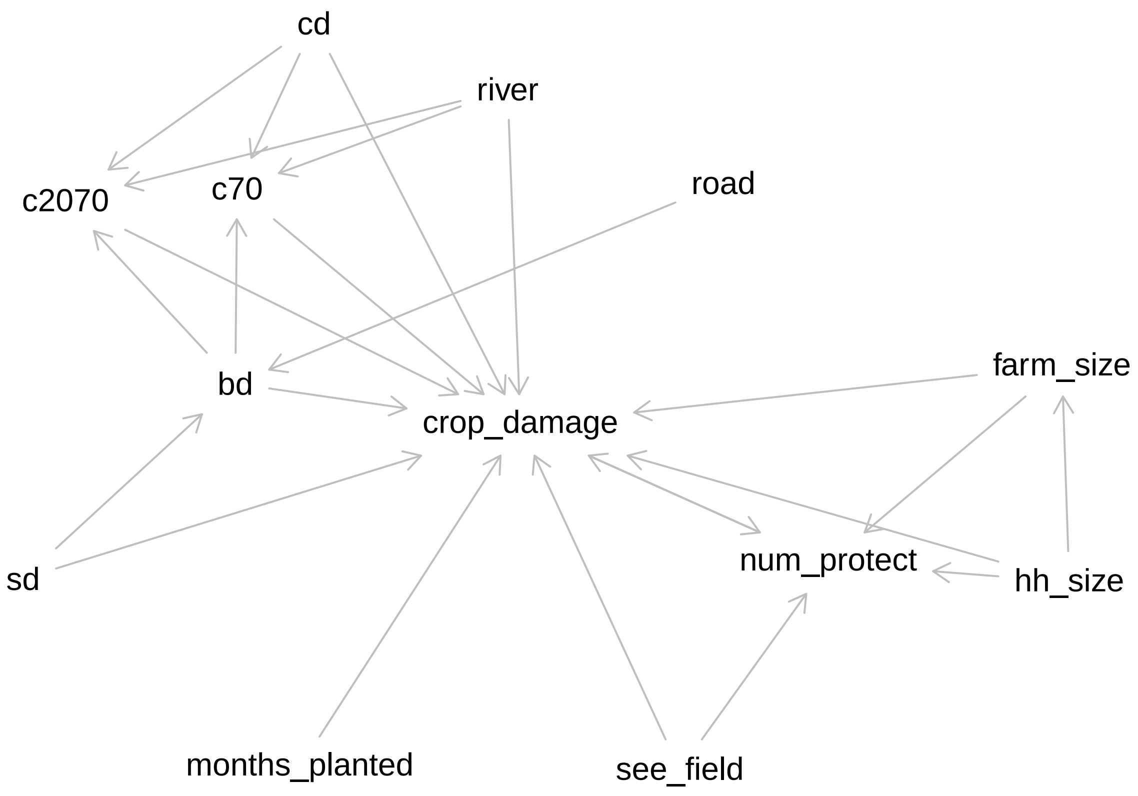
```
isCollider(ls_conf_yes_guard , "hh_size" , "guards","lsh") #shows guards is collider on path from hh_si
```

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

```
###crop damage
crop_damage_dag <-
  dagitty('dag {
  c2070 -> crop_damage
  c70 -> crop_damage
  river -> c70
  river -> c2070
  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 -> num_protect
  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
  }')

plot(crop_damage_dag)
```

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



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

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

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

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

```
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, road, sd,
##   see_field }
```

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

```
##  {}
```

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

## { bd, farm_size, hh_size, months_planted, river, road, sd, see_field }
adjustmentSets( crop_damage_dag , exposure="river" , outcome="crop_damage" )

##  {}
adjustmentSets( crop_damage_dag , exposure="river" , outcome="crop_damage" , type="canonical")

## { bd, cd, farm_size, hh_size, months_planted, road, sd, see_field }
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, road, see_field }
adjustmentSets( crop_damage_dag , exposure="bd" , outcome="crop_damage" )

## { sd }
adjustmentSets( crop_damage_dag , exposure="bd" , outcome="crop_damage" , type="canonical")

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