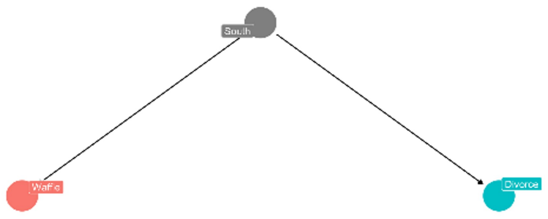
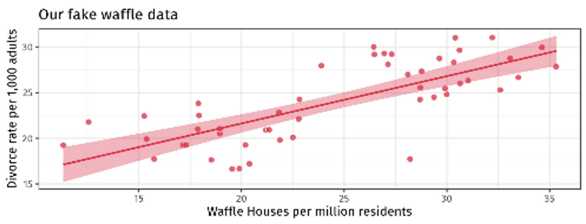
5 December 2021 Sunday 20:34

This chapter and the next are both about terrible things that can happen when we simply add variables to a regression, without a clear idea of a causal model.
MAC Lereath

A totally spurious relationship

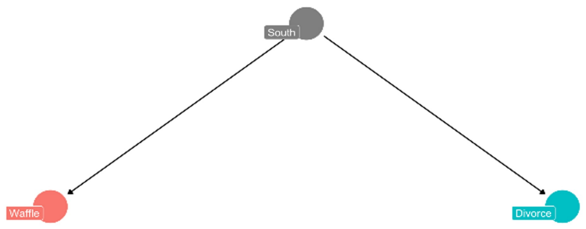
The DAG on the left will produce an association like the one on the right





Even if Waffles and Divorce are **causally** unrelated

What to do: condition on S ,ie. Make
Model which includes South
Mu=a+bw w



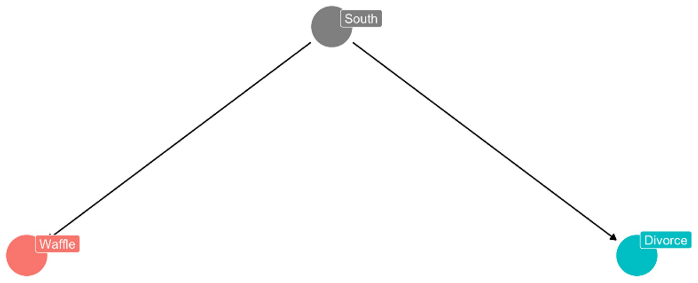
A DAG like this will produce a *correlation* between Waffles and Divorce, even when there is no **causal** relationship

This is called **confounding**: the South is **confounding** the relationship between Waffles and Divorce

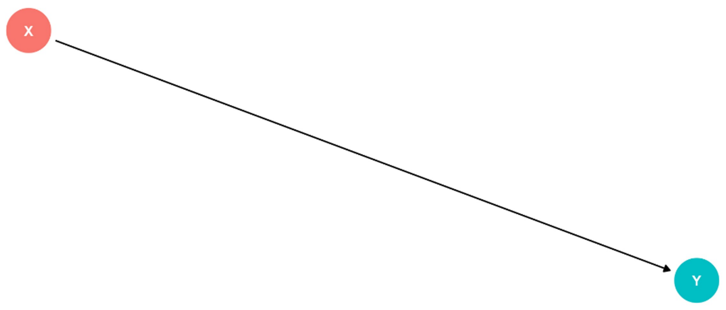
What's going on here?

To **identify** the effect of Waffles on Divorce we need to **account**, or **control for**, the indirect flow resulting from South

Otherwise we will be *confounded*

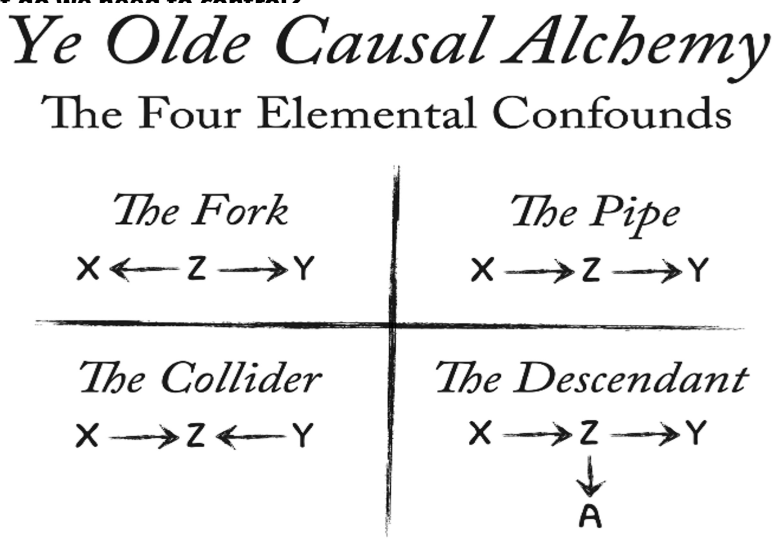


What do we need to control?



Nothing

What do we need to control?



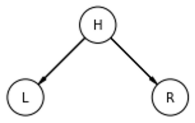
1) Fork:

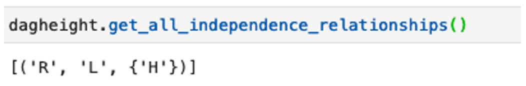
(1) The first type of relation is the one we worked with just above, a fork: X ⟵ Z ⟶ Y. This is the classic confounder. In a fork, some variable Z is a common cause of X and Y, generating a correlation between them.
If we condition on Z, then learning
X tells us nothing about Y.
X and Y are independent, conditional on Z.(if you include in the they should be around 0)

For instance our height model is created as

```
N = 100  # number of individuals
height = np.random.normal(10, 2, N)  # sim total height of each
leg_prop = np.random.uniform(0.4, 0.5, N)  # leg as proportion of height
leg_left = leg_prop * height + np.random.normal(0, 0.02, N)  # sim left leg as proportion + error
leg_right = leg_prop * height + np.random.normal(0, 0.02, N)  # sim right leg as proportion +
error
```

So height causes left and right leg heights:





```
dagheight.get_all_independence_relationships()
```

```
[('R', 'L', {'H'})]
```

ie :
mu = a + bl * d.leg_left + br * d.leg_right
Bl~0
Br ~0

Multicolinear milk:
Model the total energy content of milk (kcal.per.g.) using the variables perc.fat (percent fat) and perc.lactose

Start by modeling kcal.per.g as a function of perc.fat and perc.lactose, but in two bivariate regressions.

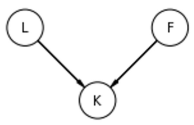
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_mean	ess_sd	ess_bulk	ess_tail	r_hat
a	0.00	0.09	-0.16	0.17	0.0	0.0	991.05	459.91	997.28	611.25	1.0
bF	0.86	0.09	0.68	1.04	0.0	0.0	1021.55	1013.71	1039.49	738.05	1.0
sigma	0.49	0.07	0.37	0.62	0.0	0.0	968.58	934.07	1005.26	714.56	1.0

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_mean	ess_sd	ess_bulk	ess_tail	r_hat
a	-0.00	0.07	-0.14	0.12	0.0	0.0	1225.79	514.64	1190.82	837.05	1.0
bL	-0.90	0.08	-1.03	-0.74	0.0	0.0	955.65	955.65	952.09	767.34	1.0
sigma	0.41	0.06	0.31	0.52	0.0	0.0	1005.57	974.42	1040.92	768.10	1.0

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_mean	ess_sd	ess_bulk	ess_tail	r_hat
a	0.00	0.07	-0.13	0.14	0.00	0.00	730.56	382.37	732.03	448.25	1.00
bF	0.24	0.19	-0.12	0.62	0.01	0.01	359.15	359.15	354.94	510.90	1.00
bL	-0.68	0.19	-1.04	-0.31	0.01	0.01	373.54	372.86	370.03	476.70	1.01
sigma	0.41	0.06	0.31	0.52	0.00	0.00	667.35	667.35	679.83	643.70	1.00

We assumed :

```
dagMiLK = CausalGraphicalModel(
   nodes=["K", "L", "F"], edges=[("L", "K"), ("F", "K")]
)
pgm = daft.PGM()
coordinates = {"K": (2, 2), "L": (1, 1), "F": (3, 1)}
for node in dagMiLK.dag.nodes:
   pgm.add_node(node, node, *coordinates[node])
for edge in dagMiLK.dag.edges:
   pgm.add_edge(*edge)
pgm.render()
plt.gca().invert_yaxis()
```



Mutivariate reg. Told us : Theres no gain in knowing F (

-0.12	0.62

)when we know L (

-1.04	-0.31

)
But when we look at the bivariates we
See a positive effect of F (

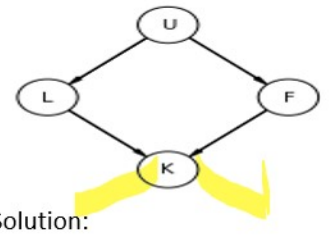
0.68	1.04

)
And negative effect of L(

-1.03	-0.74

)

So what is going on here:
We know that F and L are negatively correlated
Hence probably :



Solution:
If we could measure D, or had an evolutionary
and economic model to predict it based upon other aspects of a species, that would be better than stumbling through regressions. We'd just regression K on D, **ignoring the mediating L and F**, to estimate the causal influence of density on energy

(2) Pipe:

The second type of relation is a pipe: X → Z → Y.
If we condition on Z now, we also block the path
from X to Y.
So in both a fork and a pipe, conditioning of the middle variable
blocks the path.
Example (Post treatment bias):

Let's create a model :

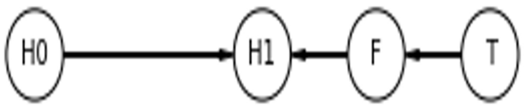
```
# number of plants
N = 100
# simulate initial heights
h0 = np.random.normal(10, 2, N)
# assign treatments and simulate fungus and growth
treatment = np.repeat([0, 1], N / 2)
fungus = np.random.binomial(n=1, p=0.5 -treatment * 0.4,
size=N)
h1 = h0 + np.random.normal(5 - 3 * fungus, size=N)
# compose a clean data frame
d = pd.DataFrame.from_dict({"h0": h0, "h1": h1, "treatment":
treatment, "fungus": fungus})

az.summary(d.to_dict(orient="list"), kind="stats", round_to=
2)
```

So we create heights from the initila heigts such that
if no treatment then we have 50 % probability that plant will have fungus else 10 percent prob. Of catching fungus.

İf no fungus The final height should be first height + 5cm (+-)
Else firtst height + 2 (+-)

So real dag of the simulation is:



Which implies if include Fungus in the model the effect of T should seem to be 0
(What to do : do not include post treatment effect ie fungus):
OR if look at the independence relationships:

```
plant_dag.get_all_independence_relationships(),
[('F', 'H0', set()),
 ('F', 'H0', {'T'}),
 ('H0', 'T', set()),
 ('H0', 'T', {'F'}),
 ('H0', 'T', {'F', 'H1'}),
 ('H1', 'T', {'F'}),
 ('H1', 'T', {'F', 'H0'})]
```

Lets forget how we created the model assume we try to model the height:
$H=H_0 * p$

```
with pm.Model() as m_6_6:
   p = pm.Lognormal("p", 0, 0.25)

   mu = p * d.h0
   sigma = pm.Exponential("sigma", 1)

   h1 = pm.Normal("h1", mu=mu, sigma=sigma, observed=d.h1)

   m_6_6_trace = pm.sample()

az.summary(m_6_6_trace, round_to=2)
```

3) Collider:

The third type of relation is a collider: $X \longrightarrow Z \longleftarrow Y$.
Unlike the other two types of relations, in a collider there is no association between X and Y unless you condition on Z.

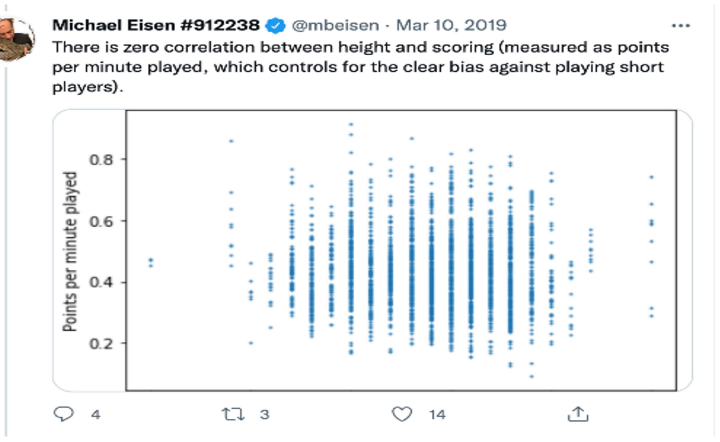
Conditioning on Z, the collider variable, opens the path. Once the path is open, information flows between
X and Y.

Example: Let's make simulation
(1) Each year, 20 people are born with uniformly distributed happiness values.
(2) Each year, each person ages one year. Happiness does not change.
(3) At age 18, individuals can become married. The odds of marriage each year are proportional to an individual’s happiness.
(4) Once married, an individual remains married.
(5) After age 65, individuals leave the sample. (They move to Spain.)

So in our model : Age effects marriage, and Happiness effects Marriage but

So in our model : Age effects marriage, and Happiness effects Marriage but there is no relation between H and A
But since it is a collider if we include M in regression we should see a spurious correlation:
H->M<-A
H= a+bmM+baA
Another example

Example: the NBA


Michael Eisen #912238 @mbeisen · Mar 10, 2019
There is zero correlation between height and scoring (measured as points per minute played, which controls for the clear bias against playing short players).

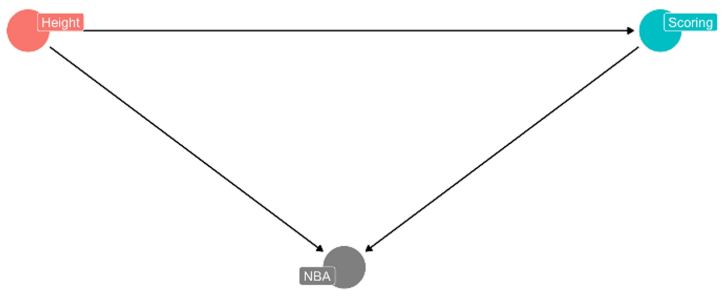
Should the NBA stop worrying about height when drafting players?

No!

Obviously, height helps in basketball

But **among NBA players** there might be no relationship between height and scoring, because shorter players have other advantages

Among NBA players = adjusting, or controlling for, being in the NBA



4

(4) The fourth bit of knowledge you need is that conditioning on a descendent variable
is like conditioning on the variable itself, but weaker. A descendent is a variable influenced by another variable

Shutting the backdoor. Blocking all confounding paths between some predictor X and some outcome Y is known as shutting the backdoor. The metaphor in play is that we don't want any spurious correlation sneaking in through a non-causal path, which is one that enters the back of the predictor X.

No matter how complicated a causal DAG appears, it is always built out of these four types of relations. And since you know how to open and close each, you (or your computer) can figure out which variables you need to control—or not—in order to shut the backdoor. Here's the recipe:

(1) List all of the paths connecting X (the potential cause of interest) and Y (the outcome).

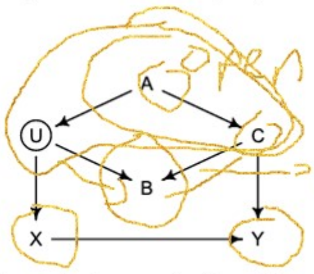
(2) Classify each path by whether it is open or closed. A path is open unless it contains a collider.

M= a+bx X +bc*c

(3) Classify each path by whether it is a backdoor path. A backdoor path has an arrow entering X.

(4) If there are any backdoor paths that are also open, decide which variable(s) to condition on to close it.

Example:



→ close open path = include a feature

We are interested in the blue path, the causal effect of X on Y. Which of the observed covariates do we need to add to the model, in order to correctly infer it? To figure this out, look for backdoor paths. Aside from the direct path, there are two paths from X to Y:

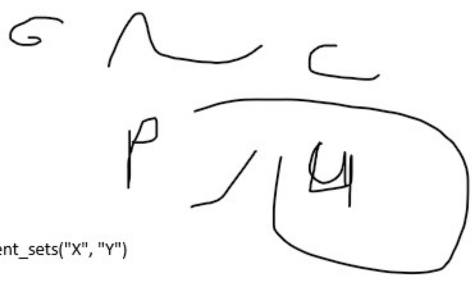
(1) $X \leftarrow U \leftarrow A \rightarrow C \rightarrow Y$

(2) $X \leftarrow U \rightarrow B \leftarrow C \rightarrow Y$

Consider the first path, passing through A. This path is open, because there is no collider within it. Information will flow through this path, confounding X → Y. It is a backdoor. To shut this backdoor, we need to condition on one of its variables. We can't condition on U, since it is unobserved. That leaves A or C. Either will shut the backdoor.

Or use dag library to list it for you:

```
ag_6_1 = CausalGraphicalModel(
   nodes=["X", "Y", "C", "U", "B", "A"],
   edges=[
      ("X", "Y"),
      ("U", "X"),
      ("A", "U"),
      ("A", "C"),
      ("C", "Y"),
      ("U", "B"),
      ("C", "B"),
   ],
)
all_adjustment_sets = dag_6_1.get_all_backdoor_adjustment_sets("X", "Y")
for s in all_adjustment_sets:
   if all(not t.issubset(s) for t in all_adjustment_sets if t != s):
      if s != {"U"}:
         print(s)
```



GRAND CHİLD