Methods 4 – Portfolio Assignment 2

```
pacman::p_load(tidyverse, rethinking, dagitty, GGally)
```

- *Type:* Group assignment
- *Due:* 3 April 2022, 23:59

Hello CogSci's :)w

In this portfolio, you are asked to do four tasks:

- Make a DAG for something

- Simulate data that fits the DAG

- Use linear models to confirm that the DAG fits the data

- Mess it up.

Each of the four tasks have some sub-steps.
Report briefly what you find, for example in a markdown document, for example called report.md so that the poor TA can easily get an overview before looking in your code :)

Then you can also make a (brief!) explanation of the phenomenon you are DAGGIN, simulating and modelling.

Looking forward !

Task 1: The DAG

- **Come up with an** incredibly interesting and scientifically important made-up **example** for a phenomenon to investigate. Decide on two variables (an outcome and a predictor) that you would like to investigate the relation between. If in doubt, you **can be inspired by Peter's amazing example** on the next page.

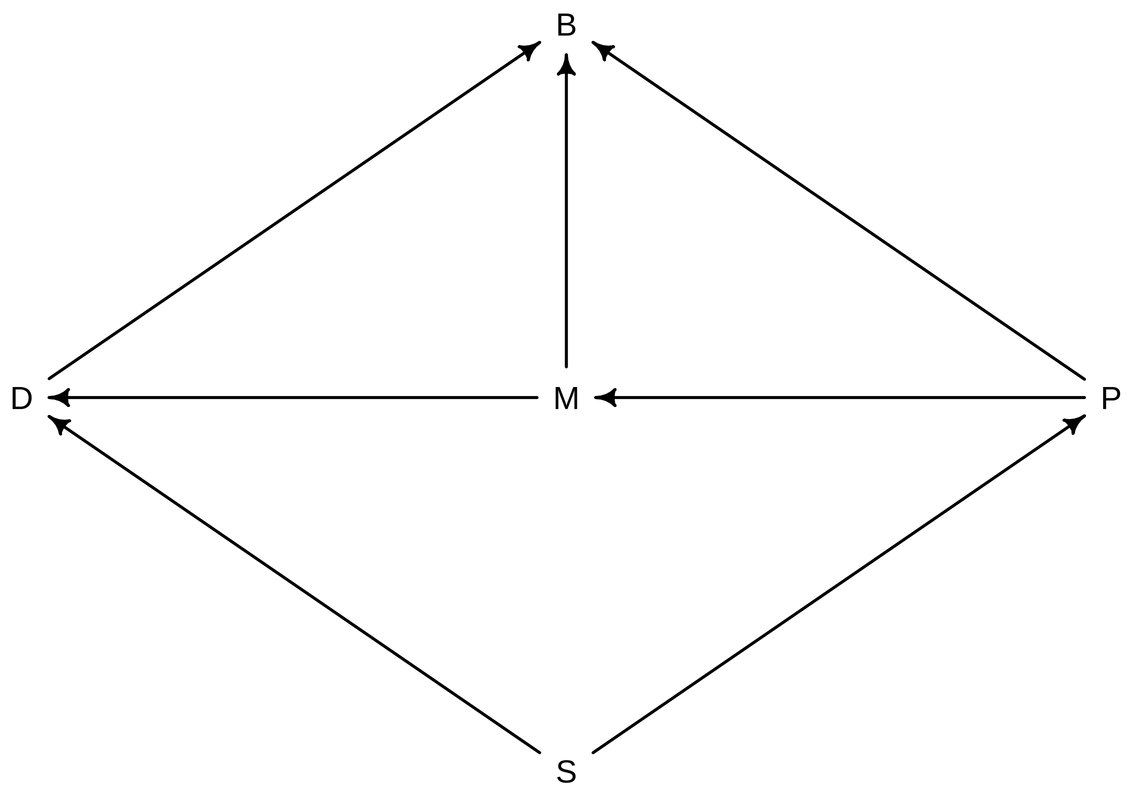
- **Make a DAG** for the phenomenon. Make it medium complicated: that means, make sure there are some different kinds of relations (see next step). Change it if you don't get anything interesting for the next steps.
Draw it somehow (on paper, in R, laser engraved in diamond).
Code it in dagitty (this is a nice tool: http://dagitty.net/dags.html)

We want to investigate the effect of having live music (M) at a friday-bar on beer purchase (B).

M: Whether there is live music playing at the bar or not (binary) P: The price of a beer. D: How many people are dancing. B: The amount of beers sold. S: The size of the bar in m^2.

Our DAG is based on these assumptions: Amount of beers sold (B) decreases with the price of a beer (P), increases with how many people are dancing (D) and increases if there is live music present (M). It is more likely that there is live music (M) if the beer prices are higher (P), and more people will be dancing (D) is there is live music (M). By increasing the size of the bar (S), we will expect more people to dance and the beer prices to increase (P).

```
DAG <- dagitty( "dag {
D <- S -> P
P -> M -> D -> B
P -> B <- M
}")
coordinates(DAG) <- list(x= c(B=0,S=0,M=0, D=-1, P=1), y=c(B=1, M=2, D=2, P=2, S=3))
drawdag(DAG)
```



$$\begin{tikzcd} & B & \\ D \arrow[ur] & M \arrow[l] \arrow[u] & P \arrow[l] \arrow[ul] \\ & S \arrow[ul] \arrow[ur] & \end{tikzcd}$$

- Find **elemental forms of variable relations** in the DAG (i.e., forks, pipes, colliders, and their descendants).

lavmig**

- Find out **what variables to include (and not include)** in a multiple linear regression to avoid 'back door' (AKA non-causal) paths. Do this first with your eyes and your mind. Then you can use dagitty's function `adjustmentSets()`.

```
adjustmentSets(DAG, "M", "B")
```

```
## { P }
```

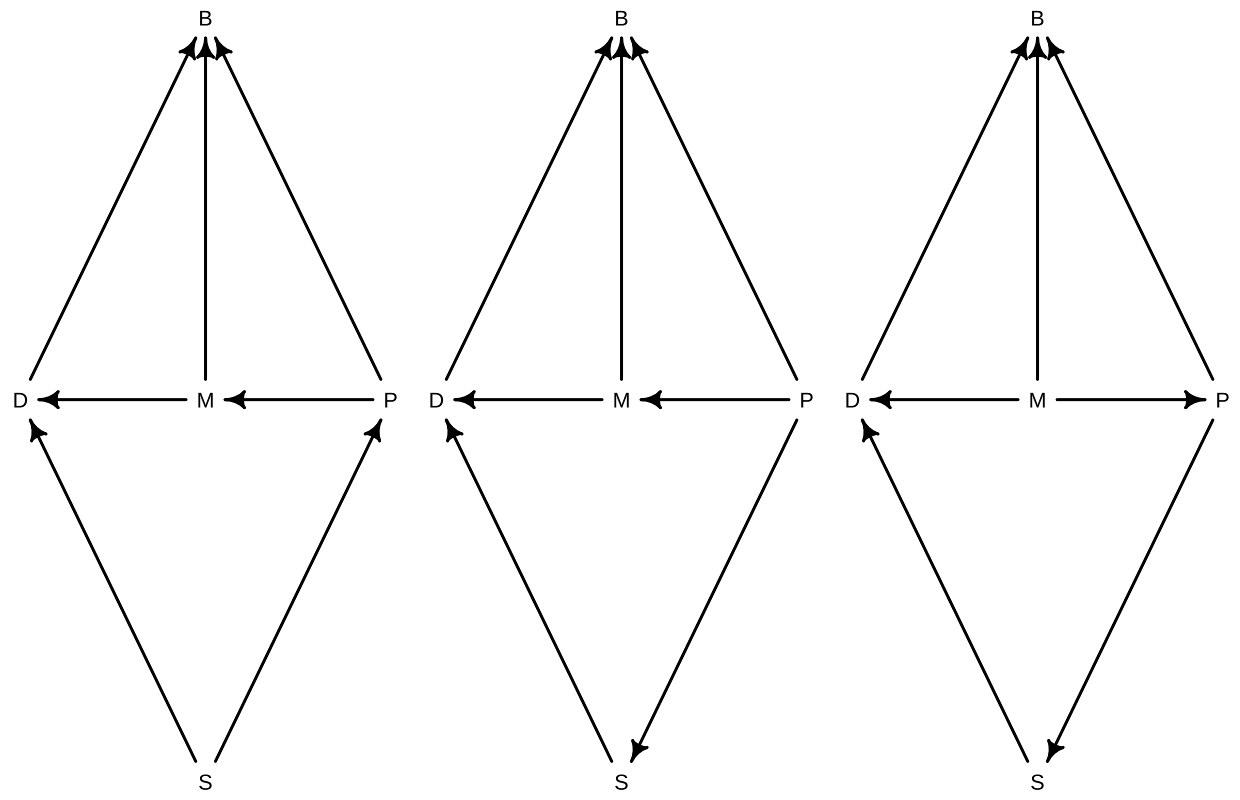
- Find out which **conditional independencies** the DAG implies. First with the mind, then with daggity's function `impliedConditionalIndependencies()`.

```r
impliedConditionalIndependencies(DAG)
```

```
## B _||_ S | D, M, P
## D _||_ P | M, S
## M _||_ S | P
```

- Find the full list of **Markov equivalent** DAGS. Use daggity's function `equivalentGraphs()`.

```r
MElist <- equivalentDAGs(DAG)
drawdag(MElist)
```



Task 2: The data

- **Simulate some data that fits the DAG.** There are many ways to do this. A simple way is just to sample one variable from a normal distribution which has another variable as mean. McElreath does this in the book a few times, and you can use this as inspiration.

Simulate data

```r
set.seed(3)
```

```r
# In the simulation area of the venue chosen for Broca's Bodega is normally distributed with a mean of 
S = rnorm(300, 120, 30)

# Beer price (P) is normally distributed with a mean=25 in a "larger-than-average-sized bar" and mean=1
P = rnorm(300, ifelse(S>=mean(S), 25, 15), 3)

# So we need to simulate M, which is, in turn, influenced by P - beer price.
M = rnorm(300, ifelse(P >= mean(P), 75, 45), 15)

# Now that we have M, we can simulate D, which is influenced by both S and M.
# The number of people dancing D is influenced by music quality
D = rnorm(300, ifelse(M>=mean(M), 70, 40), 10)

# Additionally, if the venue of the friday bar is more than 120 squares
# Around 10 people with an SD of 2 will join the dance floor
# If the area is less than 120, then 10 +-2 will leave the dancefloor.
D = D + rnorm(300, ifelse(S<= 120, -20, +20), 2)

### Now we are ready to simulate the outcome variable - beers purchased B.

# In order to simulate B, which is dependant on M, D and P, we have to simulate betas that will encode 

bM = rnorm(300, 0.1, 0.01) # For one unit the music gets better people by 0.1 more beer
bP = rnorm(300, -0.5, 0.02) # If the price of the beer increases, people will buy less beer
bD = rnorm(300, 0.1, 0.01) # The more people are dancing, the more beers will be purchased

# Now we finally simulate the outcome variable B.
B = rpois(300, 6 + bM*M + bP*P + bD*D)

d_sim = tibble(
  M_unstd = M,
  S_unstd = S,
  B_unstd = B,
  P_unstd = P,
  D_unstd = D,
  M = scale(M),
  S = scale(S),
  B = scale(B),
  P = scale(P),
  D = scale(D)
)

d_sim <- d_sim[complete.cases(d_sim), ] # removing NAs
```

Task 3: Statistics

- Run **multiple linear regression**s to **test the conditional independencies implied by your DAG**. Make sure to avoid backdoor paths. See that the linear model shows the conditional independencies implied by your DAG, implying that the data and the DAG are compatible (if the linear model doesn't show the conditional independencies implied by the DAG, the data and the DAG doesn't fit).

We got the following three conditional independencies:

B // S | D, M, P D // P | M, S M // S | P

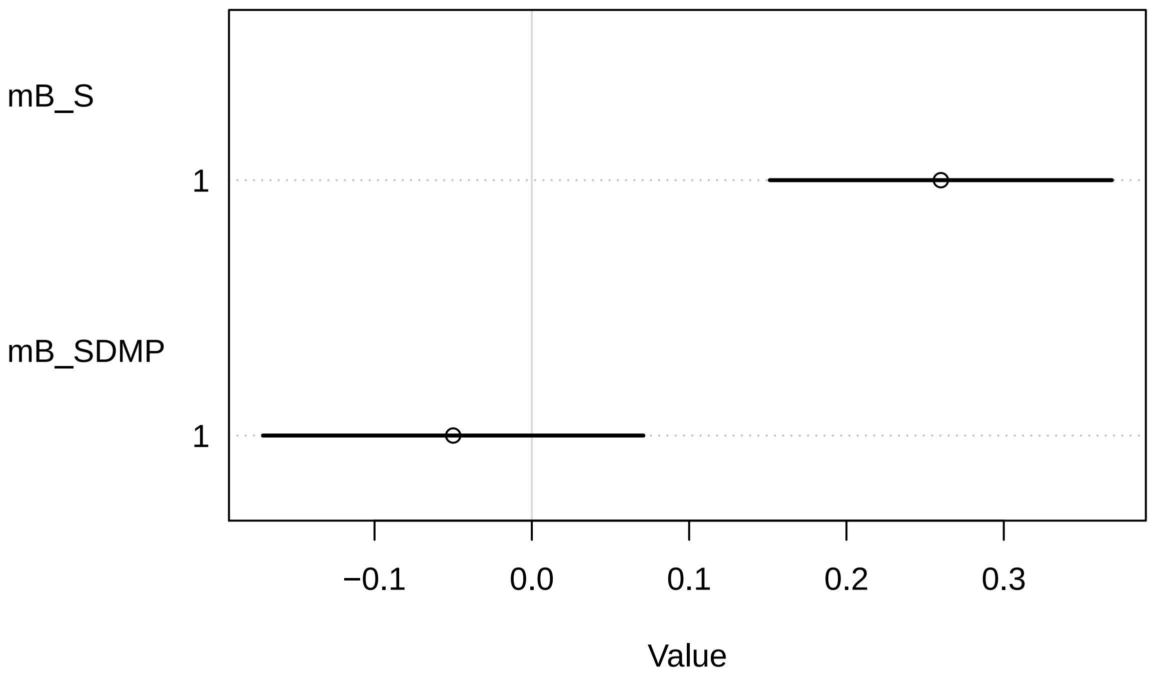
```r
# testing B _||_ S | D, M, P

# B ~ S
mB_S <- quap(
  alist(
    B ~ dnorm(mu, sigma),
    mu <- a + bS*S,
    a ~ dnorm(0, 0.2),
    bS ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)

# B ~ S + D + M + P
mB_SDMP <- quap(
  alist(
    B ~ dnorm(mu, sigma),
    mu <- a + bS*S + bD*D + bM*M + bP*P,
    a ~ dnorm(0, 0.2),
    bS ~ dnorm(0, 0.5),
    bD ~ dnorm(0, 0.5),
    bM ~ dnorm(0, 0.5),
    bP ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)

precis(mB_S)
```

```
##               mean         sd       5.5%      94.5%
## a     -0.001092938 0.05360616 -0.08676593 0.08458006
## bS     0.258348852 0.05547173  0.16969431 0.34700340
## sigma  0.962131640 0.03925018  0.89940227 1.02486101
```

```r
precis(mB_SDMP)
```

```
##               mean         sd      5.5%       94.5%
## a     -0.005748119 0.03907994 -0.0682054  0.05670917
## bS    -0.052912111 0.06170213 -0.1515240  0.04569981
## bD     0.832279817 0.08226465  0.7008050  0.96375461
## bM     0.339240798 0.06050764  0.2425379  0.43594369
## bP    -0.603666969 0.06958684 -0.7148802 -0.49245375
## sigma  0.688988849 0.02813813  0.6440187  0.73395902
```

```r
coeftab_plot(coeftab(mB_S, mB_SDMP) , pars=c("bS"))
```



The plot shows that the effect of S on the outcome B becomes basically 0, when we include the other predictors (D,M,P). This indicates that the conditional independency is indeed reflected in our data.

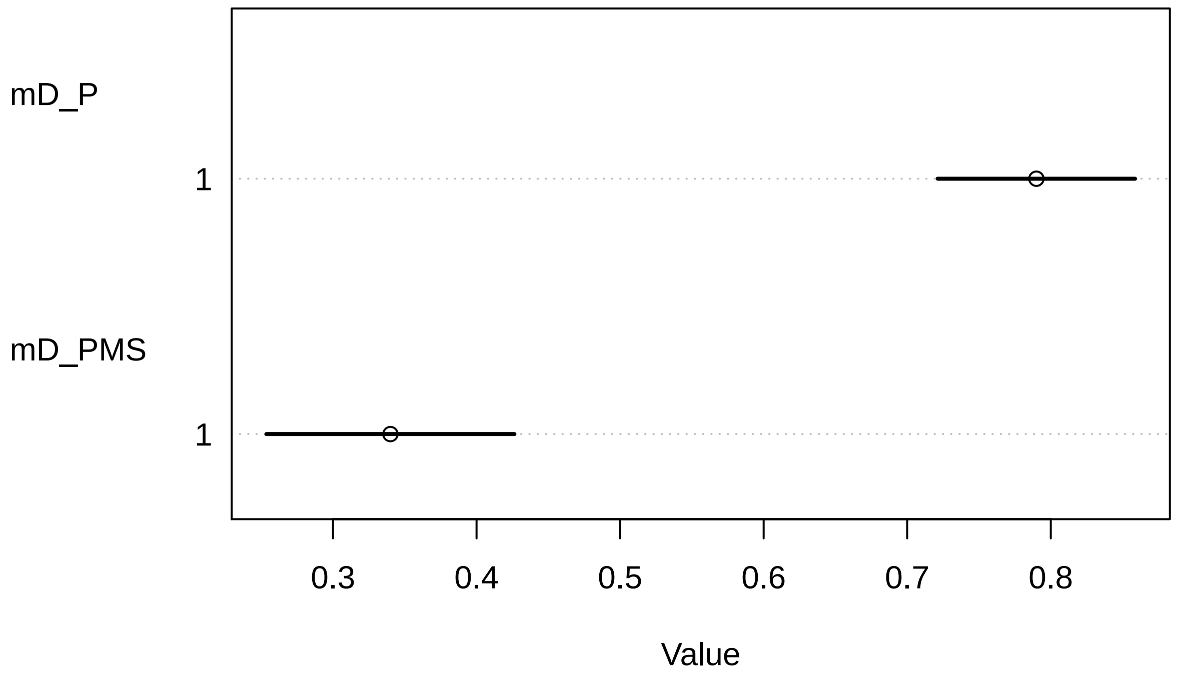
```
# testing D _||_ P | M, S

# D ~ P
mD_P <- quap(
  alist(
    D ~ dnorm(mu, sigma),
    mu <- a + bP*P,
    a ~ dnorm(0, 0.2),
    bP ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)

# D ~ P + M + S
mD_PMS <- quap(
  alist(
    D ~ dnorm(mu, sigma),
    mu <- a + bP*P + bM*M + bS*S,
    a ~ dnorm(0, 0.2),
    bP ~ dnorm(0, 0.5),
    bM ~ dnorm(0, 0.5),
    bS ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)

precis(mD_P)
```

```
##              mean         sd        5.5%      94.5%
## a     0.00312511 0.03460787 -0.05218495 0.05843517
## bP    0.78574344 0.03508376  0.72967283 0.84181406
## sigma 0.60758979 0.02481709  0.56792730 0.64725229
```

```
precis(mD_PMS)
```

```
##                 mean         sd        5.5%      94.5%
## a     -0.0009928542 0.02722439 -0.04450268 0.04251697
## bP     0.3448972755 0.04408730  0.27443725 0.41535730
## bM     0.4038310286 0.03504282  0.34782583 0.45983623
## bS     0.2804432494 0.03967182  0.21704002 0.34384648
## sigma  0.4751439167 0.01940784  0.44412644 0.50616140
```

```
coeftab_plot(coeftab(mD_P, mD_PMS) , pars=c("bP"))
```



The plot shows that the effect of P on the outcome D becomes close to0, when we include the other predictors (M, S). This indicates that the conditional independency is indeed reflected in our data. However, not as strongly as one could wish for.

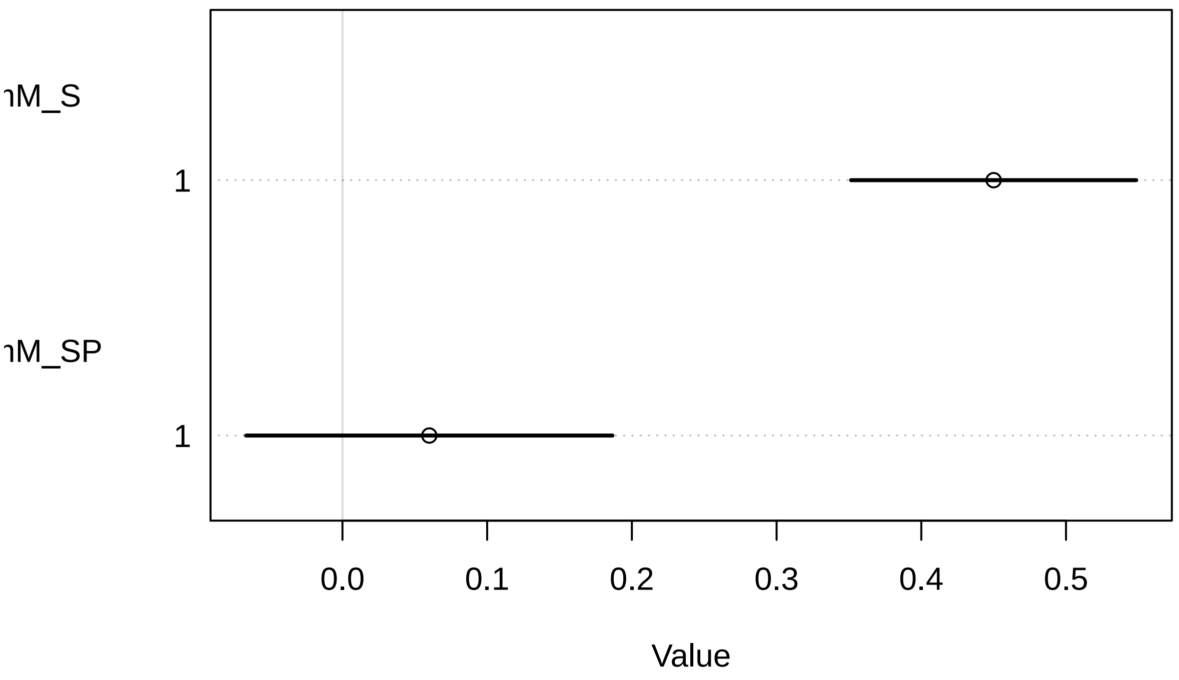
```
# testing M _||_ S | P

# M ~ S
mM_S <- quap(
  alist(
    M ~ dnorm(mu, sigma),
    mu <- a + bS*S,
    a ~ dnorm(0, 0.2),
    bS ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)

# M ~ S + P
mM_SP <- quap(
  alist(
    M ~ dnorm(mu, sigma),
    mu <- a + bS*S + bP*P,
    a ~ dnorm(0, 0.2),
    bS ~ dnorm(0, 0.5),
    bP ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)

precis(mM_S)
```

```
##              mean         sd        5.5%      94.5%
## a     0.007467325 0.04885104 -0.07060607 0.08554072
## bS    0.453912631 0.05027933  0.37355656 0.53426871
## sigma 0.871088821 0.03554586  0.81427967 0.92789797
```

```
precis(mM_SP)
```

```
##              mean         sd        5.5%     94.5%
## a     0.007997319 0.04403647 -0.06238147 0.0783761
## bS    0.059299280 0.06457997 -0.04391199 0.1625105
## bP    0.551384016 0.06443891  0.44839819 0.6543698
## sigma 0.780610274 0.03186118  0.72968996 0.8315306
```

```
coeftab_plot(coeftab(mM_S, mM_SP) , pars=c("bS"))
```



The plot shows that the effect of S on the outcome M becomes basically 0, when we include the other predictor (P). This indicates that the conditional independency is indeed reflected in our data.

Task 4: Messing it up

- Try and **deliberately have an open back door path** and see if you can get wrong inference.

Our DAG tells us (as tested with the "adjustmentSets"-function) that we must stratify by P when modelling the effect of M on B to avoid a backdoor path. We tested this with two models; one including P and one without P.

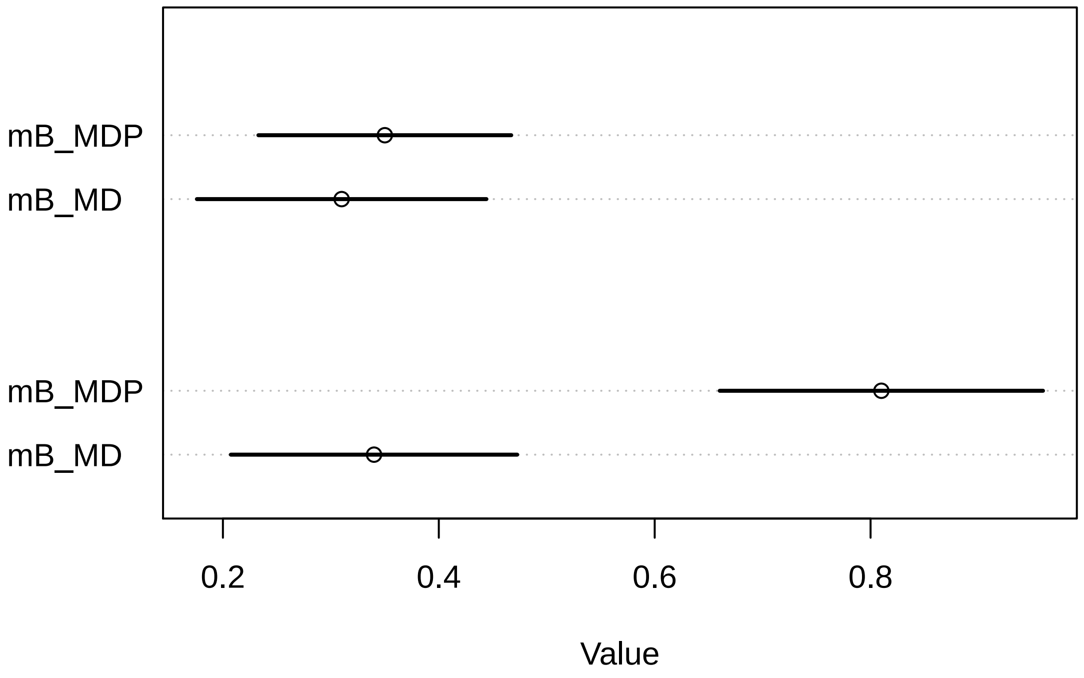
```
# B ~ M + D (open backdoor)
mB_MD <- quap(
  alist(
    B ~ dnorm(mu, sigma),
    mu <- a + bM*M + bD*D,
    a ~ dnorm(0, 0.2),
    bM ~ dnorm(0, 0.5),
    bD ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)

# B ~ M + D + P (closed backdoor)
mB_MDP <- quap(
  alist(
    B ~ dnorm(mu, sigma),
    mu <- a + bM*M + bP*P + bD*D,
    a ~ dnorm(0, 0.2),
    bM ~ dnorm(0, 0.5),
    bP ~ dnorm(0, 0.5),
    bD ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)


precis(mB_MD)
```

```
##               mean         sd       5.5%      94.5%
## a     -0.004626439 0.04476396 -0.0761679 0.06691502
## bM     0.314211893 0.06843285  0.2048430 0.42358081
## bD     0.335525672 0.06767230  0.2273723 0.44367907
## sigma  0.794144101 0.03240819  0.7423496 0.84593865
```

```
precis(mB_MDP)
```

```
##              mean         sd        5.5%       94.5%
## a     -0.00588029 0.03912874 -0.06841557  0.05665499
## bM     0.34778835 0.05975805  0.25228344  0.44329326
## bP    -0.62588248 0.06464699 -0.72920085 -0.52256412
## bD     0.80587238 0.07640425  0.68376364  0.92798113
## sigma  0.68988901 0.02817442  0.64486085  0.73491717
```

```
coeftab_plot(coeftab(mB_MD, mB_MDP), pars = c("bM", "bD"))
```

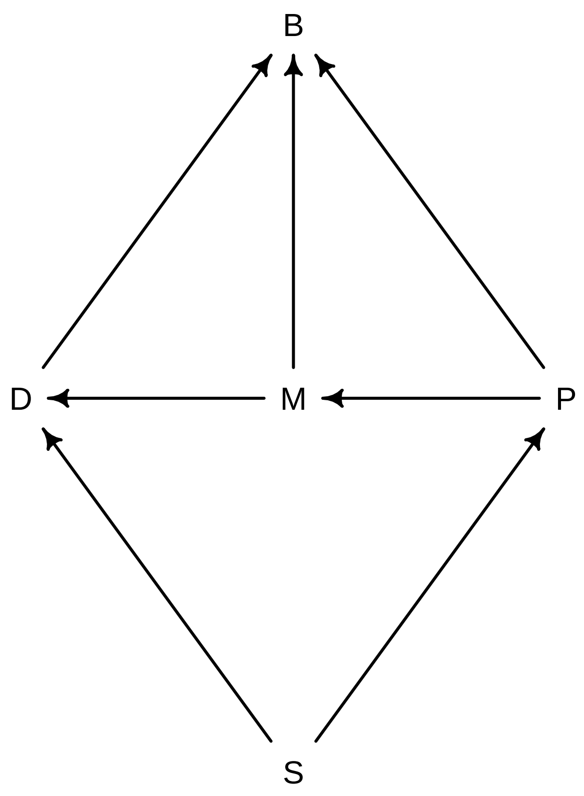


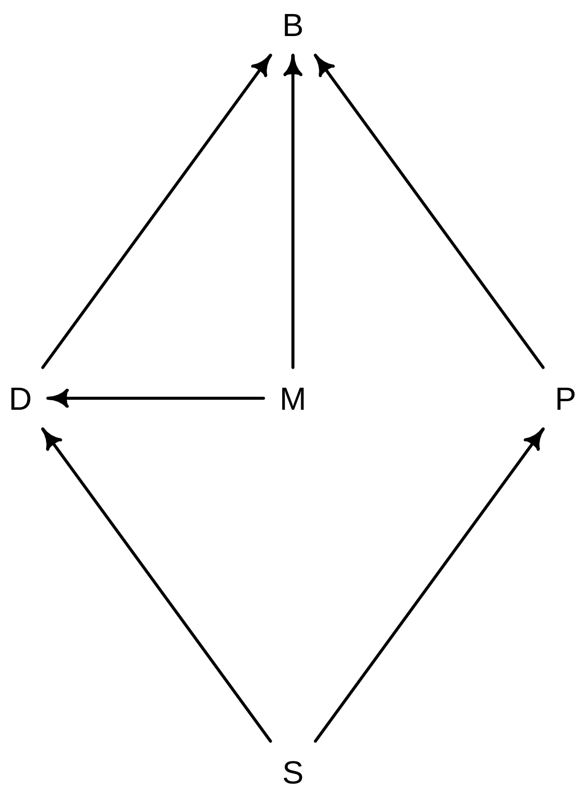
Answer: We see that the effect of D on B is underestimated unless P is included in the model. :) This makes sense as the effect of M on B is both direct but also indirect via D.

- Try and deliberately **simulate some data that doesn't fit the DAG**, or **create a new DAG that doesn't fit the data**.

We've chosen to create a new DAG and test this on our data. The new, fake DAG is identical to the real one except that we've removed the arrow from P to M. The real and fake DAGs are visualised below:

```
# real DAG (left)
DAG_real <- dagitty("dag {
D <- S -> P
P -> M -> D -> B
P -> B <- M
}")
# fake DAG (right)
DAG_fake <- dagitty("dag {
D <- S -> P
M -> D -> B
P -> B <- M
}")

coordinates(DAG_real) <- list(x= c(B=0,S=0,M=0, D=-1, P=1), y=c(B=1, M=2, D=2, P=2, S=3))
coordinates(DAG_fake) <- list(x= c(B=0,S=0,M=0, D=-1, P=1), y=c(B=1, M=2, D=2, P=2, S=3))

DAGs <- list(DAG_real, DAG_fake)
drawdag(DAGs)
```





These are the implied conditional independencies we get from our two DAGs:

```
# Real DAG
print(impliedConditionalIndependencies(DAG_real))
```

```
## B _||_ S | D, M, P
## D _||_ P | M, S
## M _||_ S | P
```

```
# Fake DAG
impliedConditionalIndependencies(DAG_fake)
```

```
## B _||_ S | D, M, P
## D _||_ P | S
## M _||_ P
## M _||_ S
```

- Use the same approach as above to **show that the DAG is wrong** (by showing that conditional independencies don't exist in the data, for example).

We get the following three conditional independencies from our FAKE DAG that are NOT a part of the REAL DAG that we used for simulating our data. D // P | S M // P M // S

When we test these three, we would suspect them not to show results.

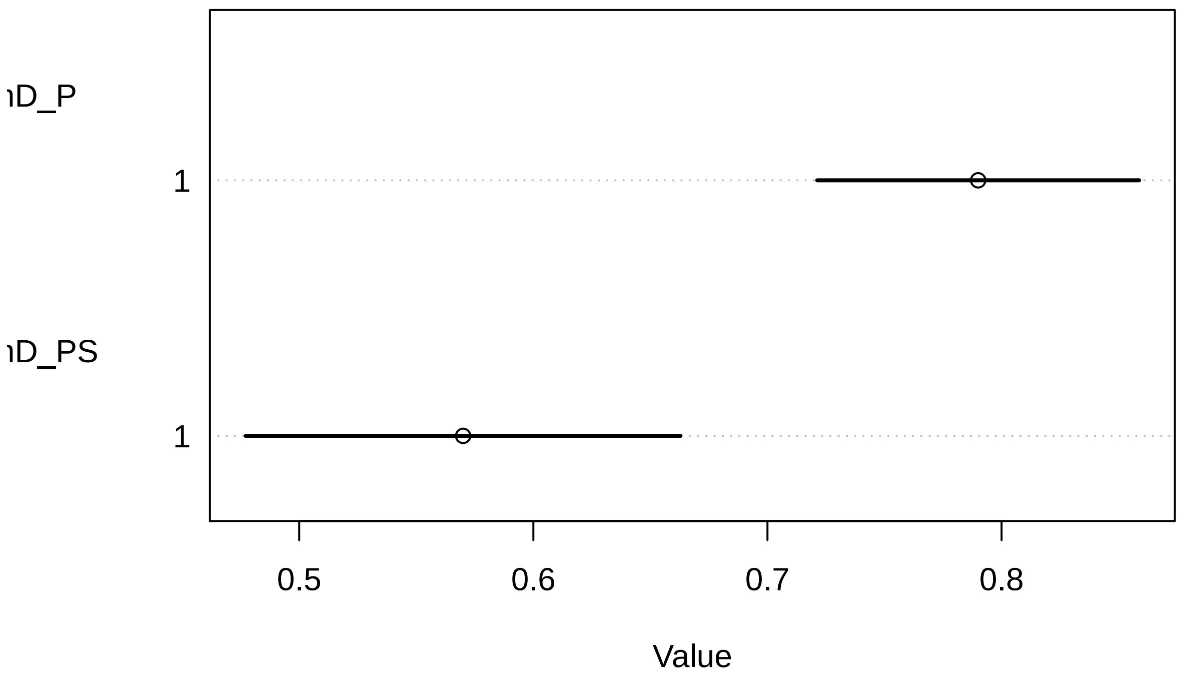
```
# testing D _||_ P | S

# D ~ P
mD_P <- quap(
  alist(
    D ~ dnorm(mu, sigma),
    mu <- a + bP*P,
    a ~ dnorm(0, 0.2),
    bP ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)

# D ~ P + S
mD_PS <- quap(
  alist(
    D ~ dnorm(mu, sigma),
    mu <- a + bP*P + bS*S,
    a ~ dnorm(0, 0.2),
    bP ~ dnorm(0, 0.5),
    bS ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)

precis(mD_P)
```

```
##              mean         sd        5.5%      94.5%
## a     0.003146043 0.03460070 -0.05215256 0.05844465
## bP    0.785667666 0.03507636  0.72960887 0.84172646
## sigma 0.607460049 0.02480386  0.56781870 0.64710140
```

```
precis(mD_PS)
```

```
##              mean         sd        5.5%      94.5%
## a     0.002325317 0.03255939 -0.04971088 0.05436151
## bP    0.568719872 0.04738022  0.49299713 0.64444261
## bS    0.303279793 0.04748524  0.22738921 0.37917037
## sigma 0.570610361 0.02330090  0.53337102 0.60784970
```

```
coeftab_plot(coeftab(mD_P, mD_PS) , pars=c("bP"))
```



```
# testing M _||_ P
mM_P <- quap(
  alist(
    M ~ dnorm(mu, sigma),
    mu <- a + bP*P,
    a ~ dnorm(0, 0.2),
    bP ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)
precis(mM_P)
```

```
##              mean         sd       5.5%      94.5%
## a     0.008153167 0.04408245 -0.0622991 0.07860544
## bP    0.593697587 0.04505169  0.5216963 0.66569889
## sigma 0.781473312 0.03189571  0.7304978 0.83244882
```

```
# testing M _||_ S
mM_S <- quap(
  alist(
    M ~ dnorm(mu, sigma),
    mu <- a + bS*S,
    a ~ dnorm(0, 0.2),
    bS ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)
precis(mM_S)
```

```
##              mean         sd        5.5%      94.5%
## a     0.007468357 0.04884986 -0.07060316 0.08553987
## bS    0.453913088 0.05027806  0.37355904 0.53426713
## sigma 0.871066593 0.03554359  0.81426107 0.92787212
```

Results for: D // P | S: The fake DAG suggests that there's no relationship between D and P once we stratify by S. This is partly reflected in these results, which is kinda weird.

Results for: M // P and M // S: If this was reflected in our data then there should be no relationship between M and P or M and S, however from the "precis" outputs there do seem to be a pretty strong relationship between the two couples of variables.