Assignment portfolio

Methods 4: The General Linear Model

Cognitive Science BSc at Aarhus University

Aleksander Moeslund Wael

Student no. 202005192

202005192@post.au.dk

19/05/2022

Methods 4 – Portfolio 1

- *Type:* Group assignment
- *Due:* 6 March 2022, 23:59

Okay here is a re-skinned version of some of McElreath's Exercises.

Have fun :)

Trigger alert for anyone who has recently experienced a pandemic.

– Peter and Chris

Pandemic Exercises

1) Testing Efficiency

Imagine there was a global pandemic.

It's a bit difficult, I know.

Maybe a new version of the old SARS-CoV turns out to be really infectious, or something like that.

A test is developed that is cheap and quick to use, and the government asks you to determine its efficiency.

To do this, they find X people that they know for sure are infected, and X people that they know for sure are not infected. *NB: This is not always possible. For example, there is an ongoing global pandemic in the real world - maybe you heard of it -where a 100% sure test doesn't exist, as far as I know. But let's ignore that. The government finds a wizard who can tell for sure, but he wants a lot of money and he's really slow too.*

Okay, so X infected people take the test, and X uninfected people take the test. See the results below. P means positive, N means negative.

- Infected:

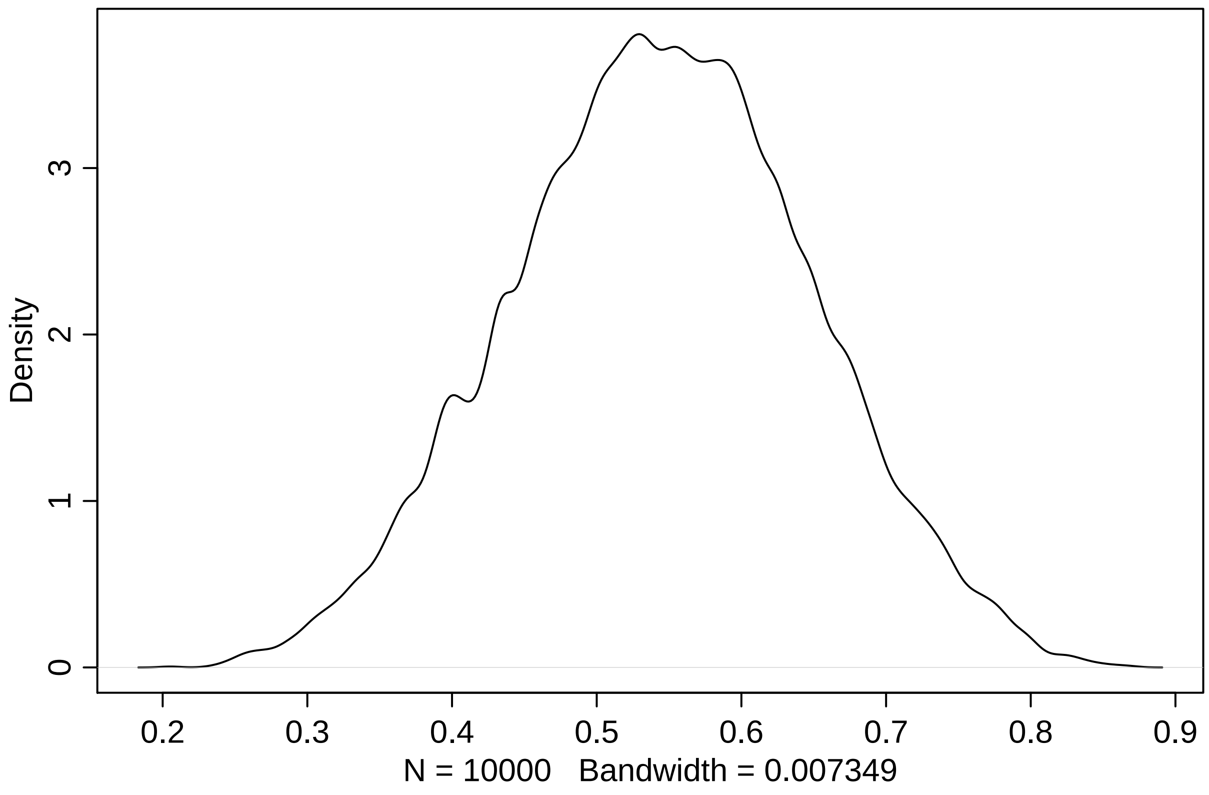
$$P, N, P, P, N, P, P, N, N, N, P, P, N, P, P, N, N, P, N, P$$

- Uninfected:

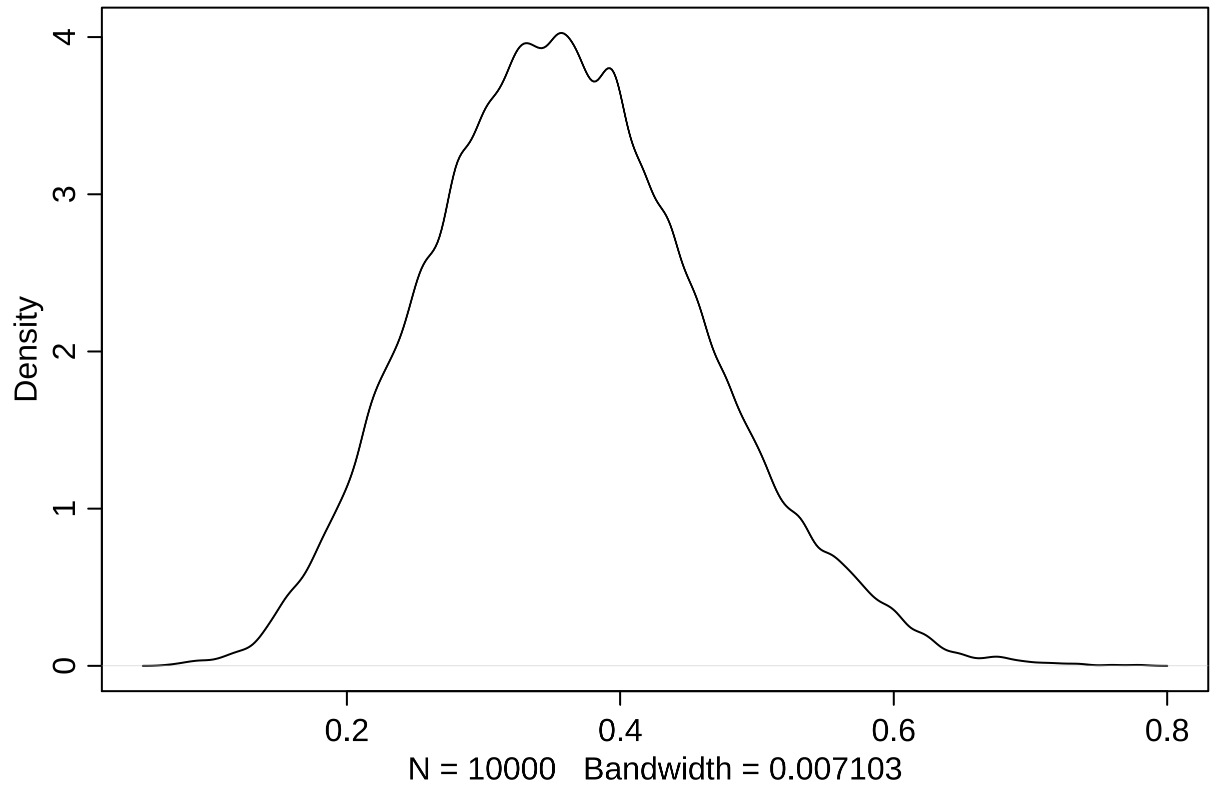
$$P, N, N, P, N, P, P, N, N, N, P, N, N, N, N, P, P, N, N, N$$

A) Estimate the probabilities of testing positive given that you're infected, and given that you're not infected. Use the grid approximation method as in the book. Use a prior you can defend using. Report the full posterior probability distribution for each case (we can do better than just a single value!).

```r
# Infected:
p_grid <- seq(from=0 , to=1 , length.out=1000 )
prior <- rep(1 , 1000 ) # a flat prior as we assume it's equally likely to be infected (it's a very wic
likelihood <- dbinom(11 , size=20 , prob=p_grid) # 11 out of 20
posterior <- likelihood * prior
posterior <- posterior / sum(posterior)

set.seed(100)
samples_infected <- sample(p_grid , prob=posterior , size=1e4 , replace=TRUE)
dens(samples_infected)
```



```r
# Uninfected:
p_grid <- seq(from=0 , to=1 , length.out=1000 )
prior <- rep(1 , 1000 ) # a flat prior as we assume it's equally likely to be infected (it's a very wic
likelihood <- dbinom(7 , size=20 , prob=p_grid) # 7 out of 20
posterior <- likelihood * prior
posterior <- posterior / sum(posterior)

set.seed(100)
samples_uninfected <- sample(p_grid , prob=posterior , size=1e4 , replace=TRUE)
dens(samples_uninfected)
```



B) The government says that they find probability distributions difficult to use. They ask you to provide them with a confidence interval of 95% within which the 'real' probability can be found. Do it.

```
HPDI(samples_infected, prob = 0.95) # 0.35 - 0.75
```

```
##     |0.95     0.95|
## 0.3483483 0.7497497
```

```
HPDI(samples_uninfected, prob = 0.95) # 0.17 - 0.56
```

```
##     |0.95     0.95|
## 0.1711712 0.5585586
```

C) The government says that their voters find confidence intervals difficult to read. In addition, they are so wide that it looks like the government doesn't know what they're doing. They want a point estimate instead. Give them one.

```
chainmode(samples_infected) # 0.53
```

```
## [1] 0.53483
```

```
chainmode(samples_uninfected) # 0.35
```

```
## [1] 0.350261
```

Conclusion: Summary of posterior predictive distributions:

For people infected:

By examining the two posterior probability distributions, we get the impression that there is a higher probability of testing positive if you are actually infected than if you are not infected (luckily enough). The confidence interval (HDPI interval) tells us that there is a 95% chance that the true probability of testing positive given that you are infected is between 35% and 75%. This interval is pretty wide, so therefor we also looked at the point estimate, which is the maximum value of the posterior predictive distribution (the mode). Using the point estimate, we can (carefully) conclude that there's around a 53% probability of being infected given a positive test.

For people NOT infected:

For the people not infected, the HDPI interval tells us that there is a 95% chance that the true probability of testing positive is between 17% and 56%. According to the point estimate, there is more precisely a 35% probability of testing positive given that you are not infected. The point estimate is useful, because it is easy to interpret, but nevertheless it is less informative than reporting a full distribution.
We can indeed conclude that there is a higher probability of getting a positive test result, given that you are actually infected.

2) Dark Cellars

Months pass. Thousands of people are tested by the wizards of the world governments. A fancy company analyses the data, and determine, with very high confidence they say, the probability of testing positive with the current test. They give the following point estimates:

- A 53% chance of testing positive if you are infected.
- A 45% chance of testing positive if you are not infected.

NB: These numbers also happen to be real estimates for the efficiency of the COVID kviktest[^1]. Remember that the actual Danish government doesn't have any wizards, though.

A) You are sitting in your dark cellar room, trying to come up with an apology to the Danish government, when you receive a positive test result on your phone. Oh, that party last weekend. In order to fight the boredom of isolation life, you start doing statistical inference. Estimate the probability that you are infected, given that it is *a priori* equally likely to be infected or not to be.

```
Pr_pos_inf <- 0.53 # probability of getting a positive test given you're infected
Pr_pos_uninf <- 0.45 # probability of getting a positive test given you're NOT infected
Pr_inf <- 0.5 # general probability in population (prior)

Pr_pos <- Pr_pos_inf * Pr_inf + Pr_pos_uninf * (1 - Pr_inf) # general prob of getting a positive result

# the probability of being infected given a positive test (this is what we want)
Pr_inf_pos <- Pr_pos_inf*Pr_inf / Pr_pos
Pr_inf_pos # 0.54
```

```
## [1] 0.5408163
```

There's a 54% probability of being infected given a positive test result

B) A quick Google search tells you that about 546.000[^2] people in Denmark are infected right now. Use this for a prior instead.

```
Pr_inf_new <- 546000/5.8e6
Pr_inf_new # 0.0941 (9% of people are infected)
```

```
## [1] 0.09413793
```

```
Pr_pos <- Pr_pos_inf * Pr_inf_new + Pr_pos_uninf * (1 - Pr_inf_new)

Pr_inf_pos <- Pr_pos_inf*Pr_inf_new / Pr_pos
Pr_inf_pos
```

```
## [1] 0.1090486
```

After updating our model, so that the prior is informed with the *actual* amount of infected people in the population: We see there is only 11% probability of actually being infected given that you get a positive test result.

C) A friend calls and says that they have been determined by a wizard to be infected. You and your friend danced tango together at the party last weekend. It has been estimated that dancing tango with an infected person leads to an infection 32% of the time[^3]. Incorporate this information in your estimate of your probability of being infected.

```
Pr_inf_tango <- 0.32

Pr_pos <- Pr_pos_inf * Pr_inf_tango + Pr_pos_uninf * (1 - Pr_inf_tango)

Pr_inf_pos_tango <- Pr_pos_inf*Pr_inf_tango / Pr_pos
Pr_inf_pos_tango # 0.3566
```

```
## [1] 0.3566022
```

We assume that we disregard the information that only 9% of the population is actually infected. We then apply the 32% prob of being infected given tango dance as our new prior. This gives a 35.66% probability of being infected given a positive test and a tango dance.

D) You quickly run and get two more tests. One is negative, the other positive. Update your estimate.

- A 53% chance of testing positive if you are infected.
- A 45% chance of testing positive if you are not infected.

```
# taking our previous posterior making it a prior, but IN ODDS
Odds_inf_pos_tango <- Pr_inf_pos_tango/(1-Pr_inf_pos_tango) # turning the prior in probabilities into a

# Bayes factor given a positive test result
Bayes_fact_pos <- 0.53/0.45

# Bayes factor given a negative test result
Bayes_fact_neg <- (1-0.53)/(1-0.45)

# now we can multiply the prior in odds with Bayes factor, first given a positive result and then multi
Updated_odds <- Odds_inf_pos_tango*Bayes_fact_pos*Bayes_fact_neg

# turning back into probabilities
Pr_inf_pos_tango_tests <- Updated_odds/(1+Updated_odds)
Pr_inf_pos_tango_tests
```

```
## [1] 0.358082
```

Given another positive test result and another negative test result the probability of being infected is now 35.80%. This is slightly above over estimate from before, but it seems as if the two tests cancel each other.

E) In a questionnaire someone sent out for their exam project, you have to answer if you think you are infected. You can only answer yes or no (a bit like making a point estimate). What do you answer?

Answer (E):

In this calculation we've worked with the calculations in *odds*. By exploiting that updating can be done by multiplying a prior with bayes factor. We assume that there's a 53% chance of testing positive if you are infected and a 45% chance of testing positive if you are not infected.

Assuming we've been dancing tango with an infected individual and have gotten 1 positive and 1 negative test, we calculate that there's only 35.8% probability of actually being infected. Therefore we say NO, we don't not believe that we are infected.

F) You are invited to a party. They ask if you are infected. They also say that they would prefer if you used an asymmetric loss function when making your decision: it is three times worse to falsely answer not infected, than to falsely answer infected. What do you answer?

```r
prob_inf <- Pr_inf_pos_tango_tests
prob_uninf <- 1-Pr_inf_pos_tango_tests

prob_inf*3 # cost of being infected
```

```
## [1] 1.074246
```

```r
prob_uninf*1 # cost of not being infected
```

```
## [1] 0.641918
```

In part "D" we calculated the probability of being infected after dancing tango and having the two test, this is "Pr_inf_pos_tango_tests". The probability of being infected is deemeed 3 times as bad as not being infected, hence comparing the loss in the two situations gives us a relative measure of cost.

As the cost associated with actually being infected after the two test is larger than the cost of not being infected, we decide to stay at home.

3) Causal Models

A problem from our textbook *Statistical Rethinking (2nd ed.)* (p. 160):

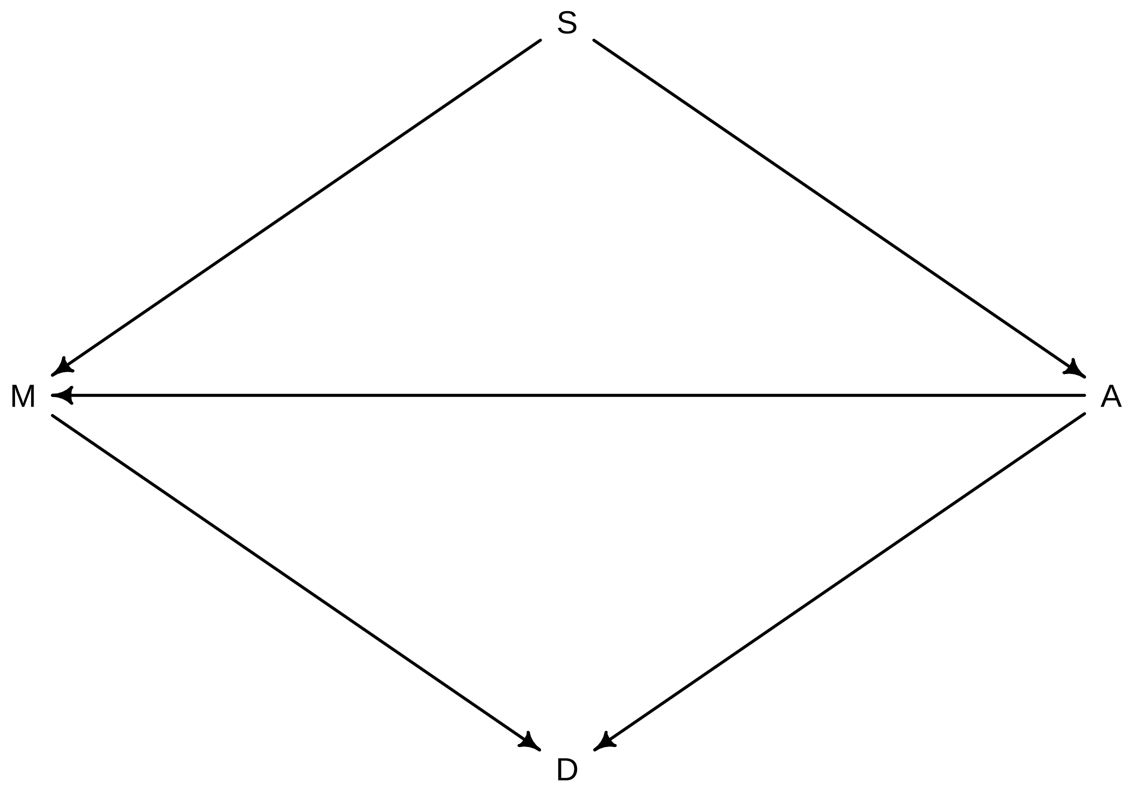
> **5H4.** Here is an open practice problem to engage your imagination. In the divorce data, states in the southern United States have many of the highest divorce rates. Add the `South` indicator variable to the analysis. First, draw one or more DAGs that represent your ideas for how Southern American culture might influence any of the other three variables (D, M, or A). Then list the testable implications of your DAGs, if there are any, and fit one or more models to evaluate the implications. What do you think the influence of "Southernness" is?

5H4.1 - Drawing the DAGS

```r
# install.packages("dagitty")
library(dagitty)
```

```
## Warning: package 'dagitty' was built under R version 4.0.5
```

```r
dag1 <- dagitty( "dag { M <- S -> A -> M -> D <- A}")
coordinates(dag1) <- list( x=c(D=0,S=0, M=-1, A=1), y=c(D=3, M=2, A=2, S=1))
drawdag( dag1 )
```



Considerations behind our choice of DAG:

S -> A: We believe that Southerness directly influences age of marriage due to several reasons. People from Southern states are known to be more religious and conservative. We believe that their prevalent positive view on the traditional marriage and their negative views on e.g. sex before marriage will cause people to marry younger. It is simply a social norm. Another practical issue is that there in many Southern states are more strict abortion rules, but as we all well know that does not stop people from having sex. So it is possible to imagine that if a woman gets pregnant, it is more likely that she will keep the child than a woman from a Nothern State, and this might force the young couple into marrying (again also because of social norms and concern for reputation).

S -> M: We believe that Southerness directly influences marriage rate because of the religious incentive to get married, as it is the "ideal" life according to Christianity.

A -> D: We believe that age influences divorce rate negatively, because of the assumption that the decision to marry was less well thought through by young people than older people, meaning that young people are more impulsive and driven by emotion, and therefor more relationships break. Also, young people simply have longer time to change and grow apart.

A -> M: We believe that age influences marriage rate negatively, because if people marry from a younger age there are simply more people to get married. So as the median age goes up, the marriage rate goes down.

M -> D: We believe that marriage rate influences divorce rate for the simple reason that more marriages increase the likelihood of more divorces.

These are the links that we thought were reasonable to assume, but you can of course make other DAGs. Some might argue that Southerness could have a direct impact on divorce rate, so it is not merely mediated by the links to age and marriage rate. Nevertheless, this is the DAG that we went with.

Testing Conditional Independencies:

```
impliedConditionalIndependencies(dag1)
```

```
## D _||_ S | A, M
```

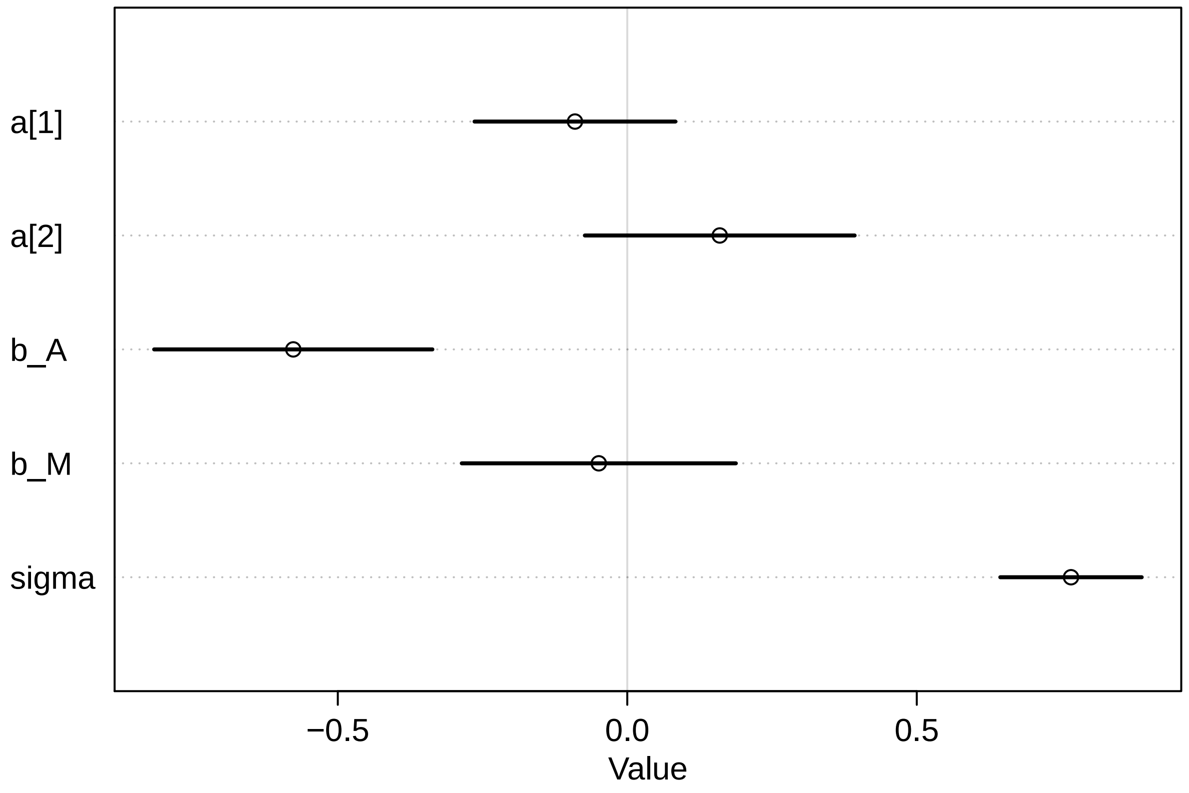
In terms of correlations in the model, we see that D is independent from S, conditional on A and M. Let's test this in our data:

Loading data:

```
data(WaffleDivorce)

d <- list()
d$A <- standardize( WaffleDivorce$MedianAgeMarriage )
d$D <- standardize( WaffleDivorce$Divorce )
d$M <- standardize( WaffleDivorce$Marriage )
d$S <- as.integer(WaffleDivorce$South+1)
```

Testing Conditional Independencies:

```
mD_SAM <- quap(
  alist(
   D ~ dnorm(mu, sigma),
   mu <- a[S] + b_A*A + b_M*M,
   a[S] ~ dnorm(0, 0.2),
   b_A ~ dnorm(0, 0.5),
   b_M ~ dnorm(0, 0.5),
   sigma ~ dexp(1)), data = d)

precis_plot(precis(mD_SAM, depth=2))
```



We wanted to see that S and D are independent (i.e. the effect of S on D is overlapping 0) when we stratify by A and M (i.e. include them in our model). This is exactly what we can see from the plot above.

5H4.2 - Making models to test implications of our causal reasoning:

Making model:

```
# this model runs two regressions
m_D_A_M_S <- quap(
  alist(
    ## S -> A
    A ~ dnorm(mu_A, sigma_A),
    mu_A <- aA[S],
    aA[S] ~ dnorm(0,0.2),
    sigma_A ~ dexp(1),

    ## A -> D <- M
    D ~ dnorm( mu , sigma ) ,
    mu <- a + bM[S]*M + bA[S]*A ,
    a ~ dnorm( 0 , 0.2 ) ,
    bM[S] ~ dnorm( 0 , 0.5 ) ,
    bA[S] ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 ),

    ## A -> M <- S
    M ~ dnorm( mu_M , sigma_M ),
    mu_M <- aM[S] + bAM[S]*A,
    aM[S] ~ dnorm( 0 , 0.2 ),
    bAM[S] ~ dnorm( 0 , 0.5 ),
    sigma_M ~ dexp( 1 )
    ) , data = d )

precis(m_D_A_M_S, depth=2)
```

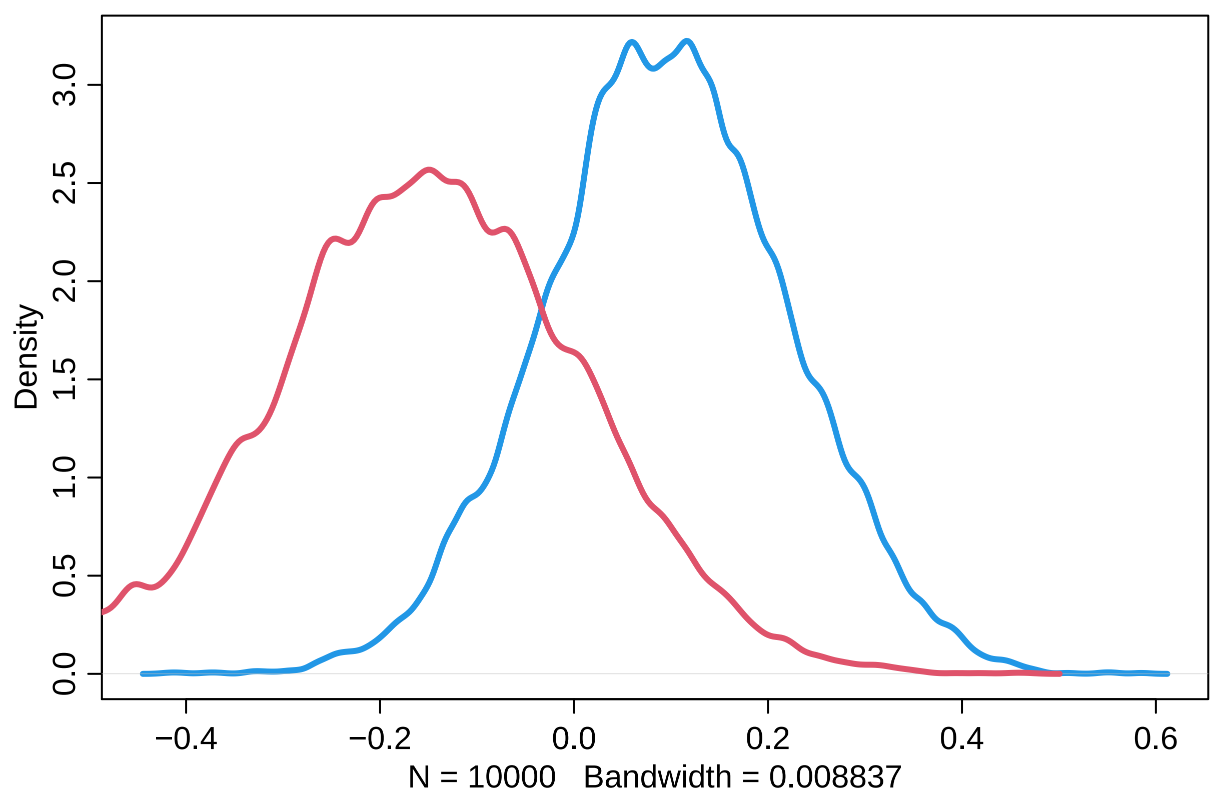
```
##                mean         sd        5.5%      94.5%
## aA[1]    0.09338565 0.12514004 -0.10661231  0.2933836
## aA[2]   -0.14885848 0.15880166 -0.40265420  0.1049372
## sigma_A  0.95988800 0.09543649  0.80736206  1.1124139
## a       -0.04127333 0.09571905 -0.19425086  0.1117042
## bM[1]   -0.10531849 0.14866137 -0.34290807  0.1322711
## bM[2]    0.42396924 0.31818011 -0.08454403  0.9324825
## bA[1]   -0.53877157 0.15047099 -0.77925327 -0.2982899
## bA[2]   -0.83838169 0.31903953 -1.34826847 -0.3284949
## sigma    0.73027697 0.07369326  0.61250091  0.8480530
## aM[1]    0.04320631 0.09936658 -0.11560067  0.2020133
## aM[2]   -0.04535941 0.14730020 -0.28077358  0.1900548
## bAM[1]  -0.70253800 0.10196694 -0.86550085 -0.5395751
## bAM[2]  -0.54009160 0.27516963 -0.97986580 -0.1003174
## sigma_M  0.68019448 0.06807878  0.57139144  0.7889975
```

```
# M and A are strongly negatively associated. If we interpret this causally, it indicates that manipula
```

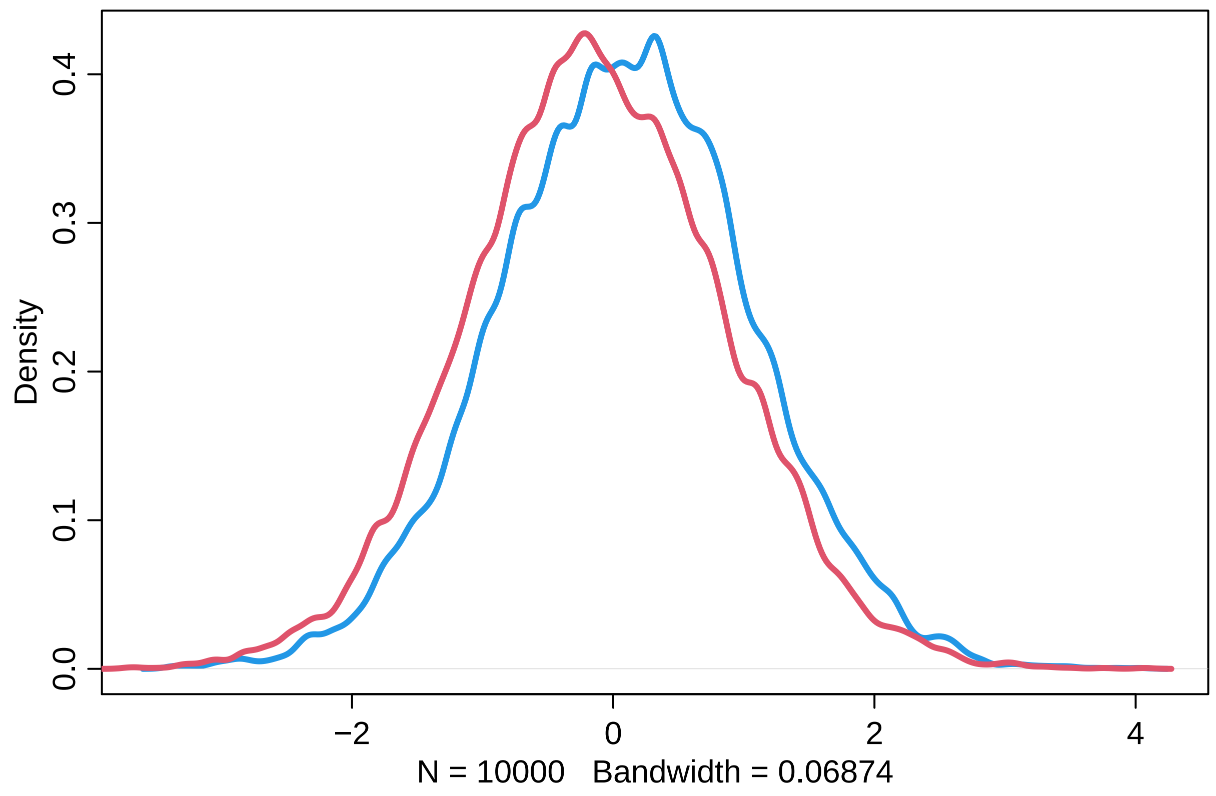
Now we have a bunch of parameters. In order to conclude on these, we are going to simulate from them and do contrasting. By looking at the parameters, we can get an idea about what could be interesting to dive into. As a rule of thumb, if you can multiply the standard deviation with two and add/subtract it from the mean, without the mean going to zero, then you might have an interesting effect. As we see, this is in most cases not possible with our model, but the two slopes for median age do have this trade. Therefor, we will start by investigating our claim that Southerness influences median marriage age.

1. investigation: How does Southerness influence median-marriage-age?

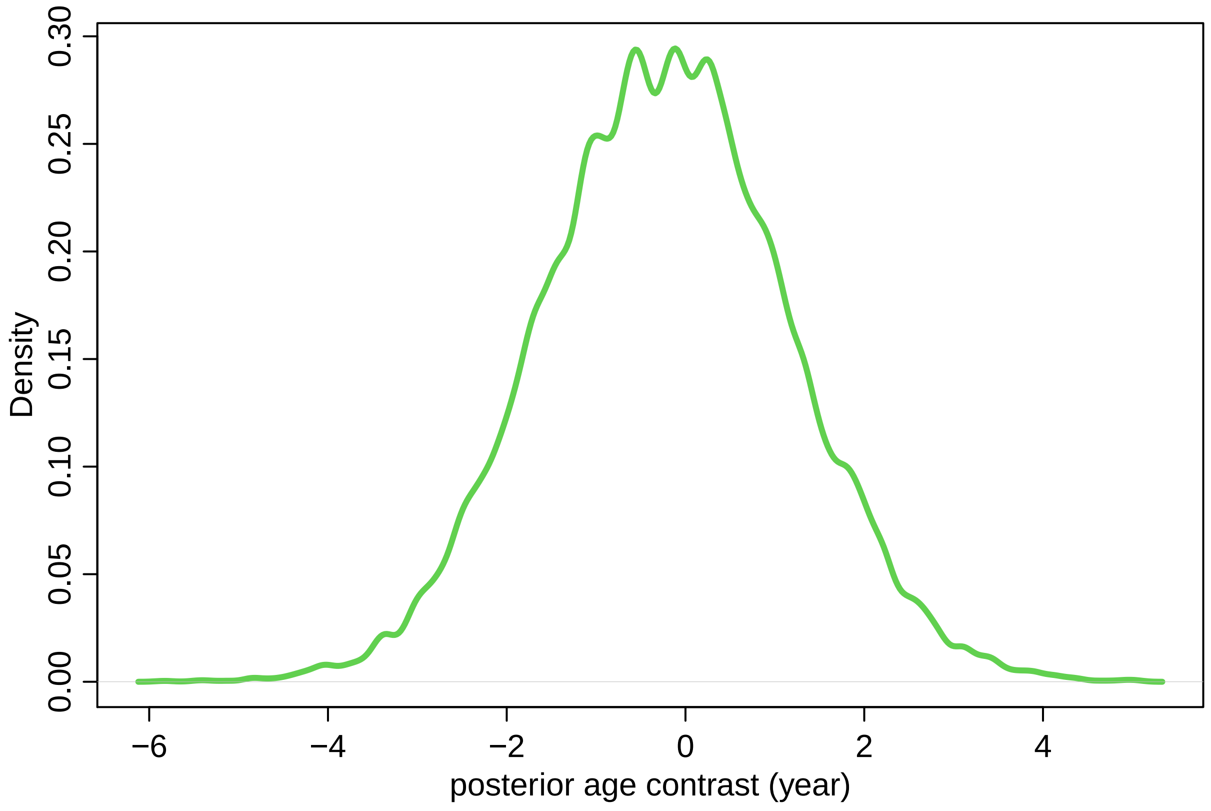
```
# simulating
post <- extract.samples(m_D_A_M_S)

dens(post$aA[,1], lwd=3, col=4) # blue = non-southern
dens(post$aA[,2], lwd=3, col=2, add=TRUE) # red = south
```



```r
A1 <- rnorm(1e4, post$aA[,1], post$sigma_A) #simulating 10.000 observations and using the parameters fr
A2 <- rnorm(1e4, post$aA[,2], post$sigma_A) #same for Southerns
dens(A1, col=4, lwd=3) # non-south = blue
dens(A2, col=2, lwd=3, add=TRUE) # south = red
```



```
# contrast
A_contrast <- A2 - A1
dens(A_contrast,
     col=3, lwd=3,
     xlab="posterior age contrast (year)")
```



```
# proportion above zero
sum(A_contrast > 0)/1e4 # 42.64%
```

```
## [1] 0.4345
```

```
sum(A_contrast < 0)/1e4 # 57.36%
```

```
## [1] 0.5655
```

Answer: We see from the posterior predictive that there is a lot of overlap, but in order to be able to conclude anything, it is important to do the contrasting. From the contrast plot we see that the distribution is centered pretty close around zero, but it is skewed a little bit. More accurately, 57% of times you randomly pick a south-state, they have younger marriage age than non-south-state, based on our simulations. This means that Southerness is *not* a very good predictor of median-age, as this is very close to chance level.

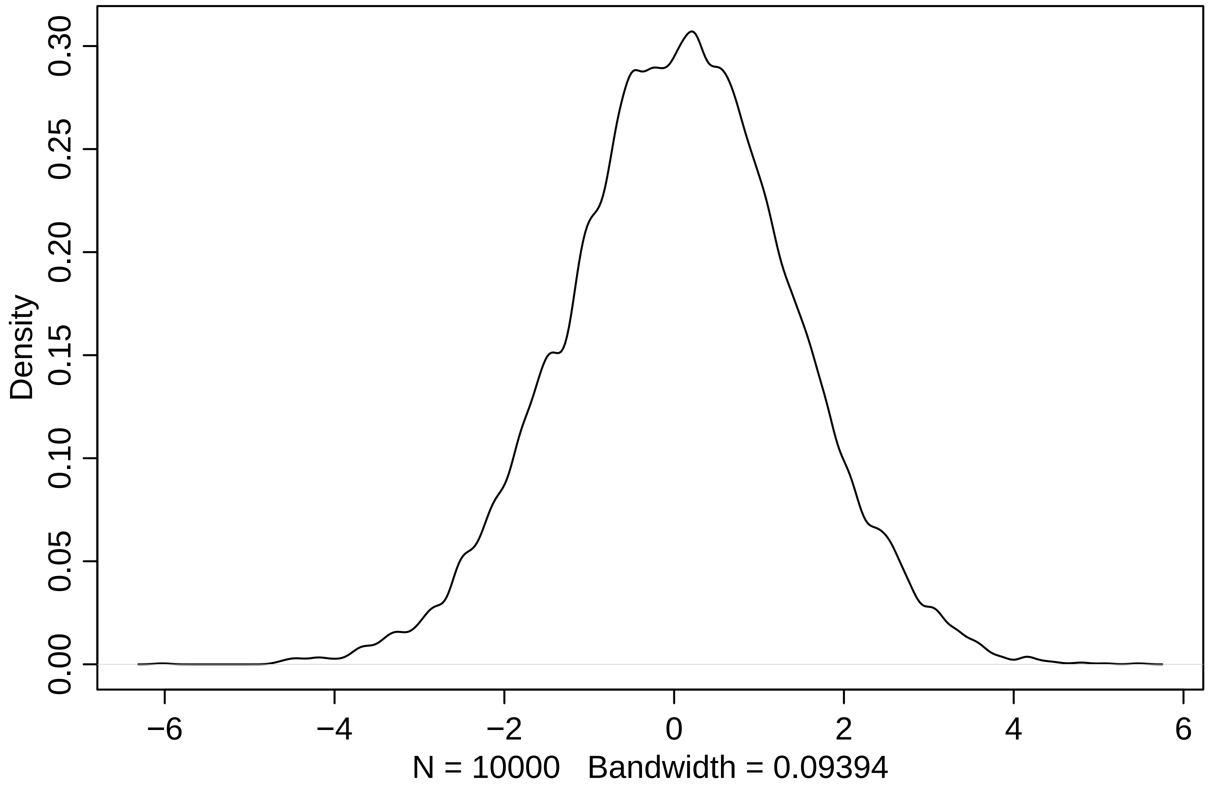
Next, we will investigate the other direct link from Southerness that we assumed in our DAG.

2. investigation: How does Southerners influence marriage rate?

```
m_D_A_M_S_sim <- sim(m_D_A_M_S,
             n = 1e4,
             data = list(S=c(1,2)),
             vars=c("A","M", "D"))
```

```
## Warning in if (left == var) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (left == var) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (left == var) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (left == var) {: the condition has length > 1 and only the first
## element will be used
```

```
M_contrast <- m_D_A_M_S_sim$M[,2]-m_D_A_M_S_sim$M[,1]
dens(M_contrast)
```



```
sum(M_contrast > 0)/1e4 # 51.91%
```

```
## [1] 0.5241
```

```
sum(M_contrast < 0)/1e4 # 48.09%
```

```
## [1] 0.4759
```

Answer: Based on our contrasting, we estimate that 51.91% of the times you randomly pick a South state, their marriage rate will be higher than that of a non-southern state. You literally cannot get much closer to chance level, so we can conclude that Southerness is an extremely poor predictor of marriage rate.

Since we in our DAG assume that there is no direct causal link between Southerness and divorce rate, we should have encapsulated all the effect from by investigating the effect from Southerness on age and marriage rate. Since both of our analyses showed that Southerness was a very poor predictor, we conclude that Southerness does not have a big impact on a state's divorce rate.

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.

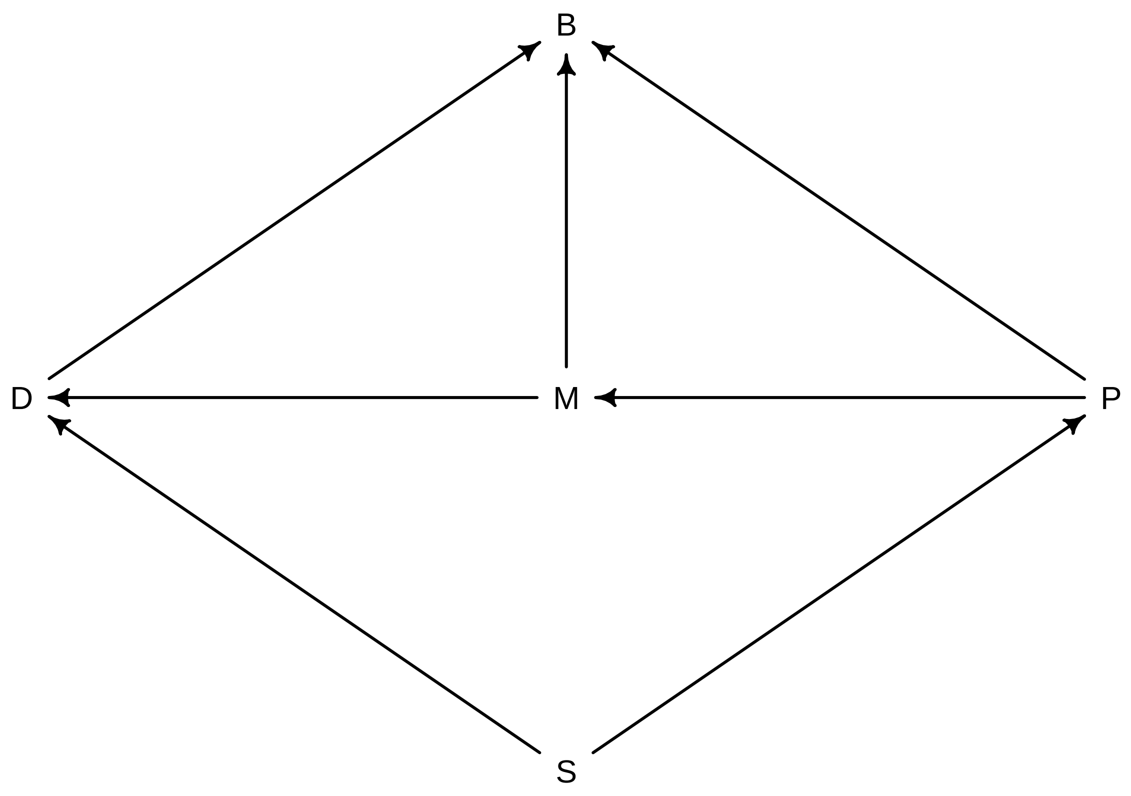
- Make a DAG for the phenomenon. Make it medium complicated: that means, make sure there are some different kinds of relations (see next step). Change it if you don't get anything interesting for the next steps.
Draw it somehow (on paper, in R, laser engraved in diamond).
Code it in dagitty (this is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



- 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 }
```

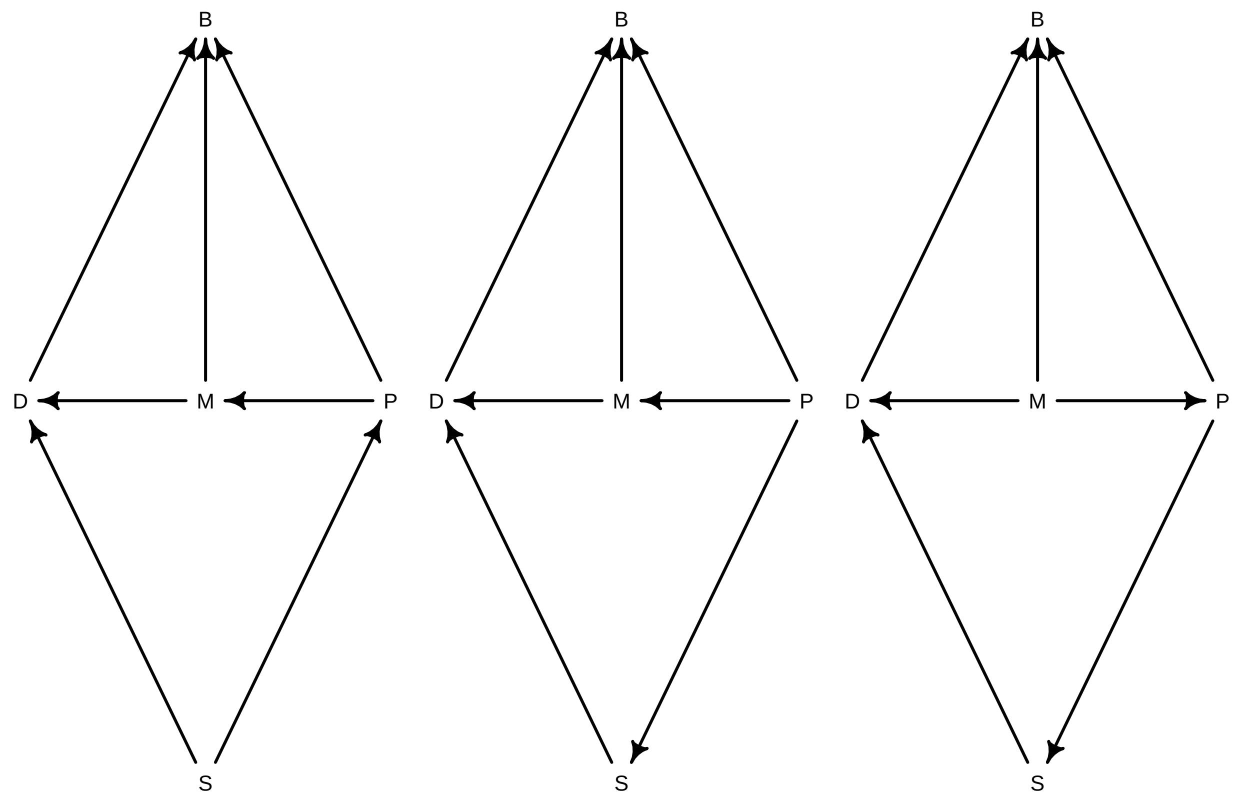
- Find out which **conditional independencies** the DAG implies. First with the mind, then with daggity's function `impliedConditionalIndependencies()`.

```
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()`.

```
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

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

```
# 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

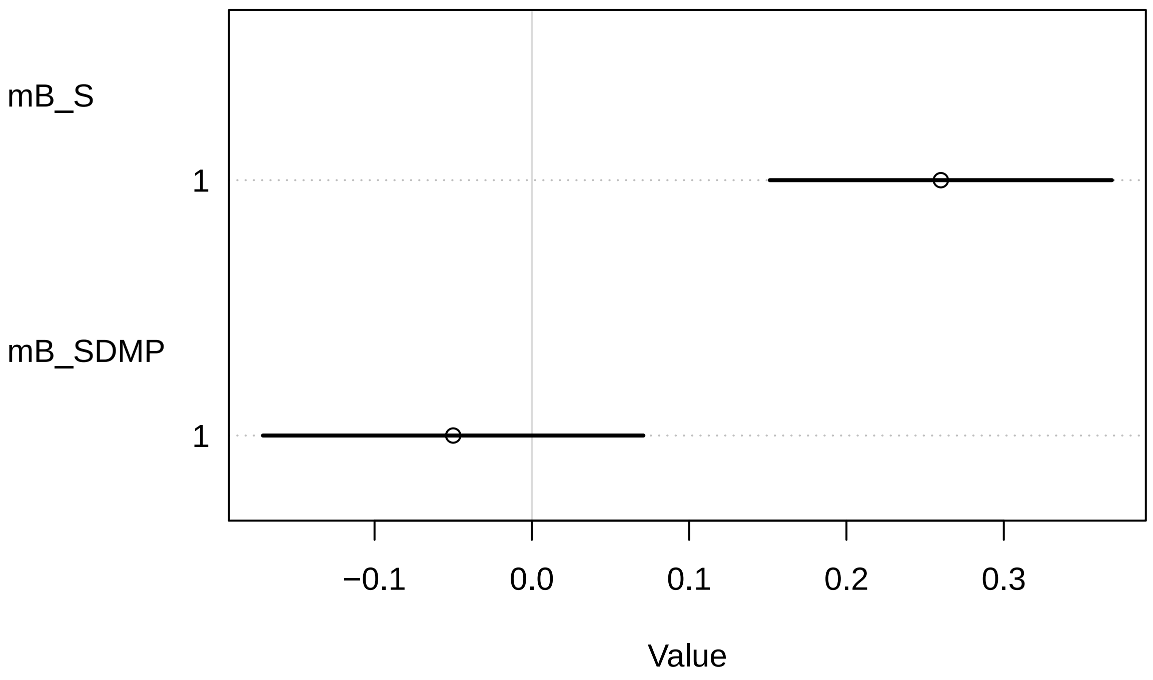
```
# testing B _||_ S | D, M, P

# 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
```

```
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
```

```
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)
```

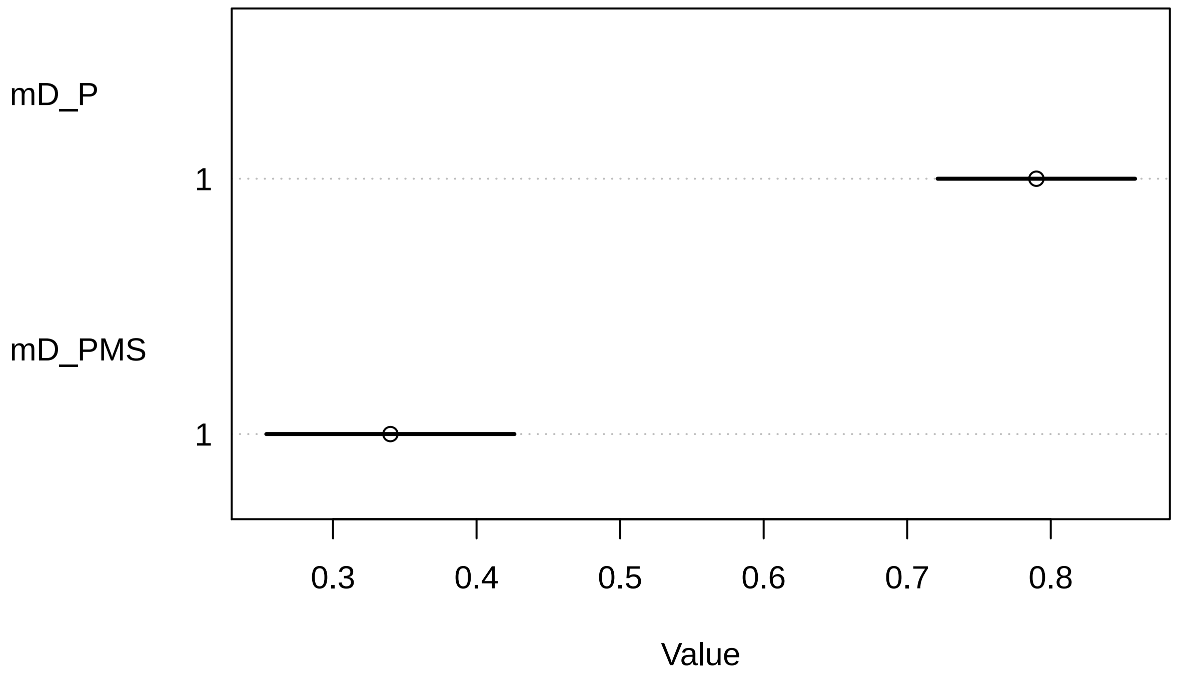
```r
  ), data = d_sim
)

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
```

```r
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
```

```r
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.

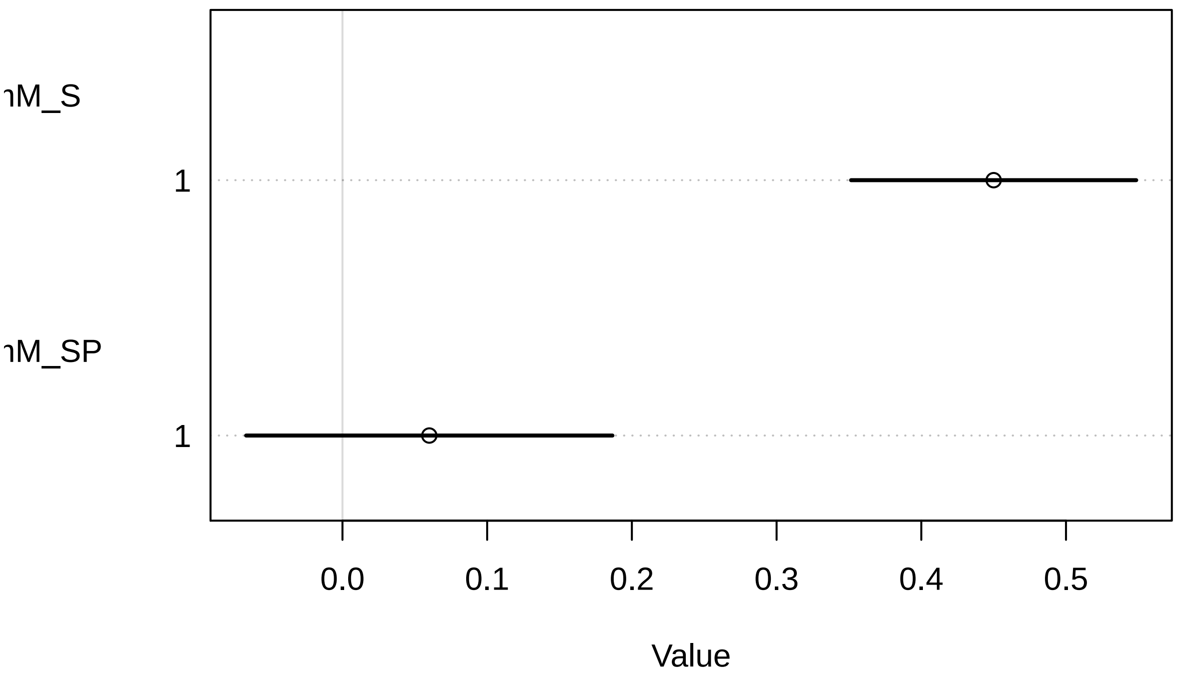
```
# testing M _||_ S | P

# 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.

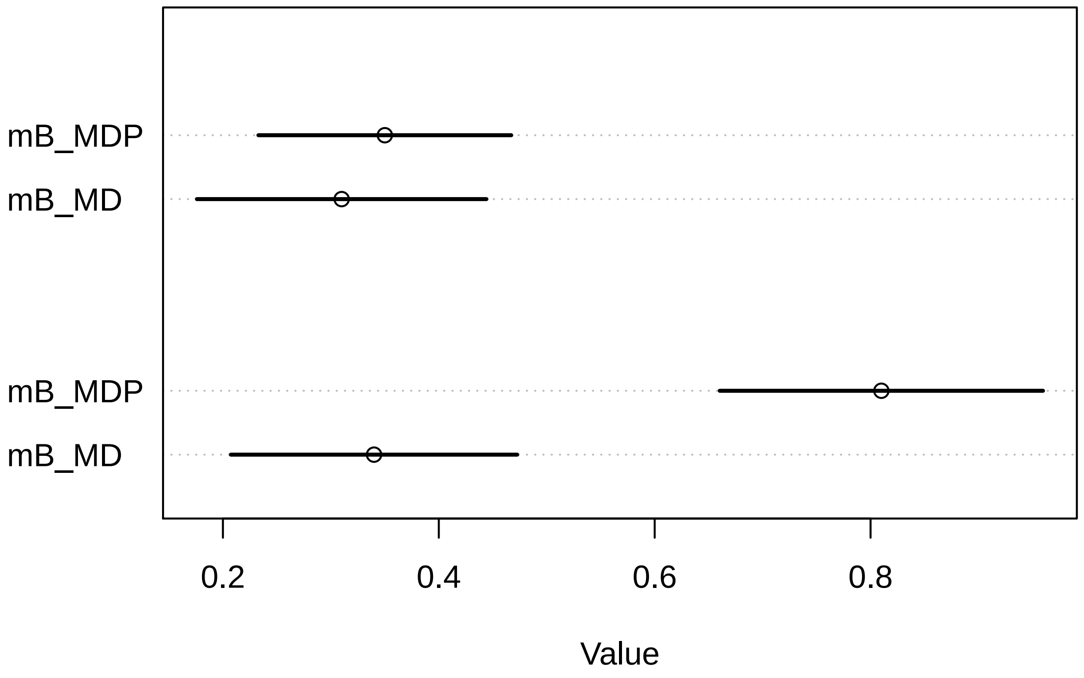
```
# B ~ M + D (open backdoor)
mB_MD <- quap(
  alist(
    B ~ dnorm(mu, sigma),
    mu <- a + bM*M + bD*D,
    a ~ dnorm(0, 0.2),
    bM ~ dnorm(0, 0.5),
    bD ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = d_sim
)

# 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"))
```

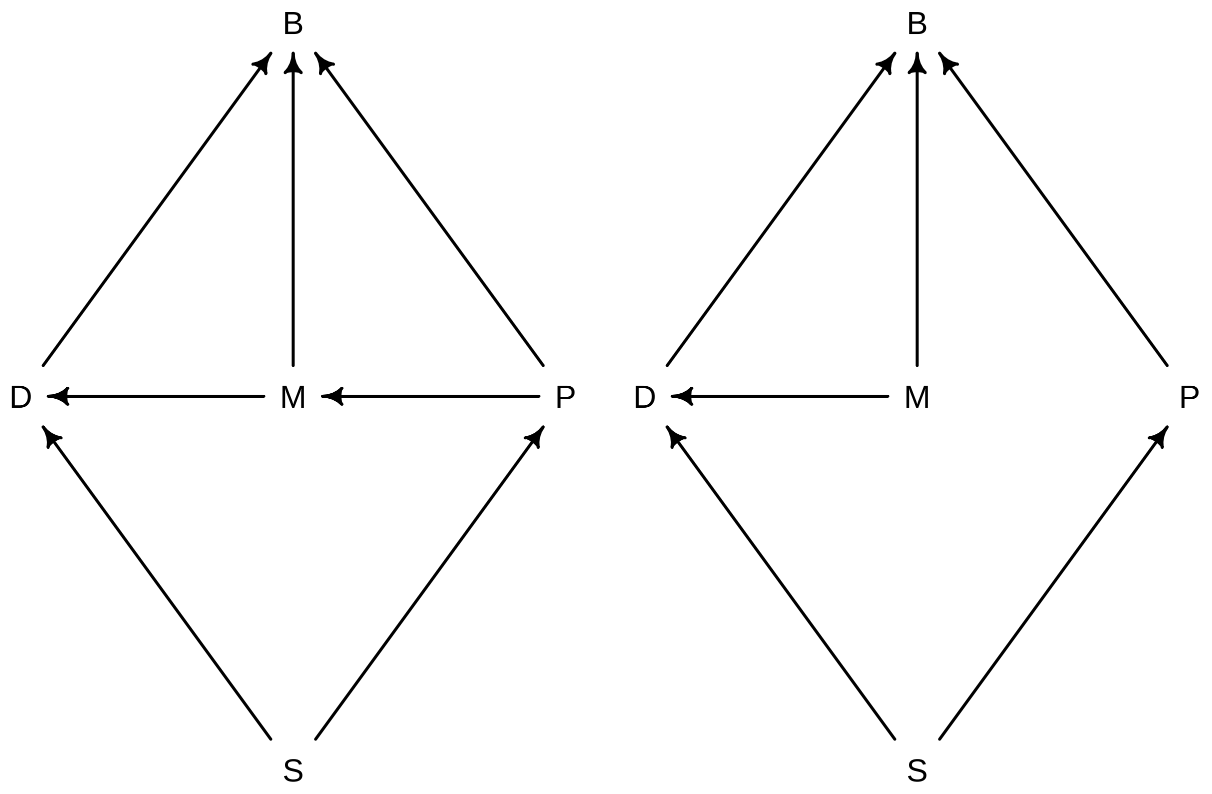


Answer: We see that the effect of D on B is underestimated unless P is included in the model. :) This makes sense as the effect of M on B is both direct but also indirect via D.

- 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:

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

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

```r
# 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.

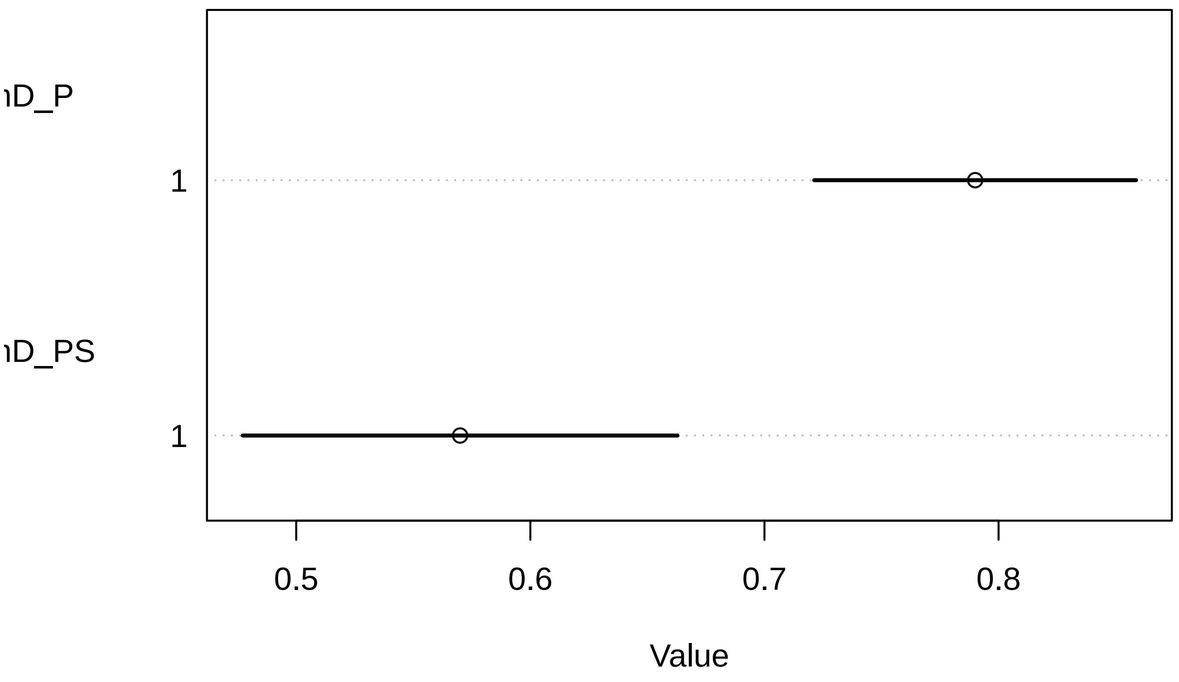
```r
# testing D _||_ P | S

# 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
```

```r
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
```

```r
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.

Methods 4 – Portfolio Assignment 3

- *Type:* Individual assignment
- *Due:* 1 May 2022, 23:59

Hey again CogSci's :)

So now for the last of the three portfolios :)

This time it's an individual one. We will build a workflow and use it to analyze a new dataset.

There are seven tasks below. As usual, handing in as a markdown is nice :)

1. Get familiar with the data

This dataset contains information about passengers aboard the Titanic.

```r
df_train <- read_csv("data/titanic_train.csv")
```

```
## Rows: 891 Columns: 12

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): Name, Sex, Ticket, Cabin, Embarked
## dbl (7): PassengerId, Survived, Pclass, Age, SibSp, Parch, Fare

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
df_test <- read_csv("data/titanic_test.csv")
```

```
## Rows: 418 Columns: 11

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): Name, Sex, Ticket, Cabin, Embarked
## dbl (6): PassengerId, Pclass, Age, SibSp, Parch, Fare

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

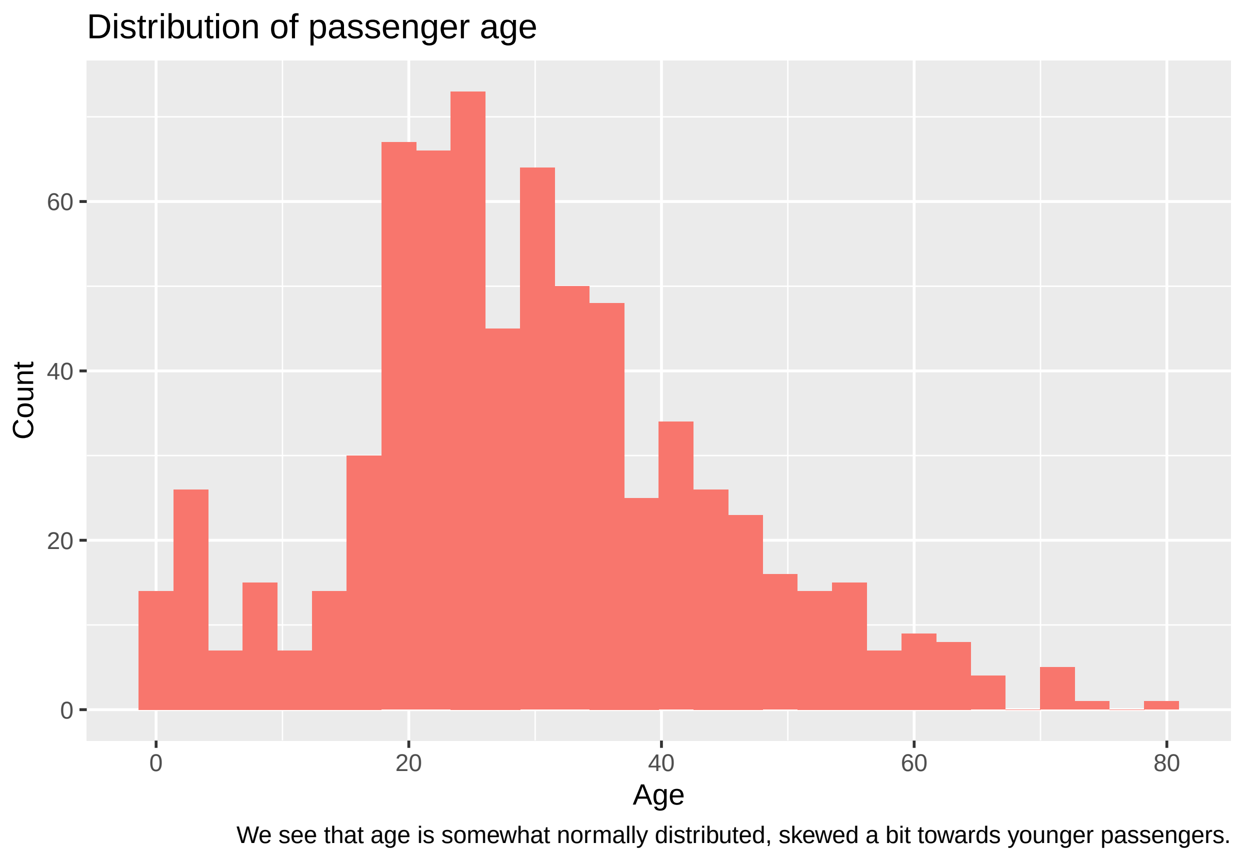
The dataset includes 12 variables (as described in the Kaggle entry):

Survival: Survival (0 = No; 1 = Yes) Pclass: Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd) Name: Name of passenger Sex: Sex of passenger Age: Age of passenger Sibsp: Number of Siblings/Spouses Aboard for passenger Parch: Number of Parents/Children Aboard for passenger Ticket: Ticket number Fare: Passenger fare (price paid by passenger, can include multiple tickets) Cabin: Cabin no. (a lot of NA's in this variable) Embarked: Port of embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

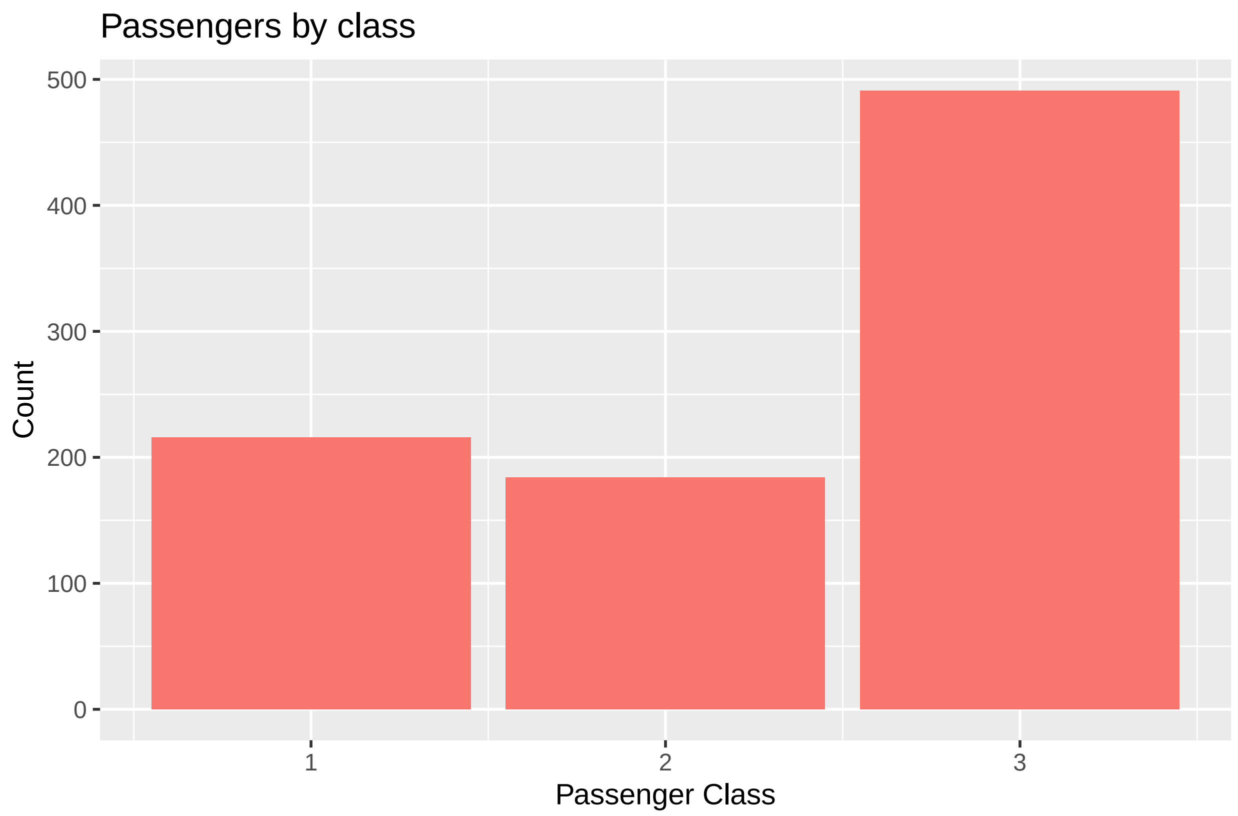
Plotting some variables of interest

```
df_train %>%
  ggplot()+
  geom_histogram(aes(Age, fill = "red"))+
  xlab("Age")+
  ylab("Count")+
  labs(title = "Distribution of passenger age", caption = "We see that age is somewhat normally distribu
  theme(legend.position = "none")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 177 rows containing non-finite values (stat_bin).
```

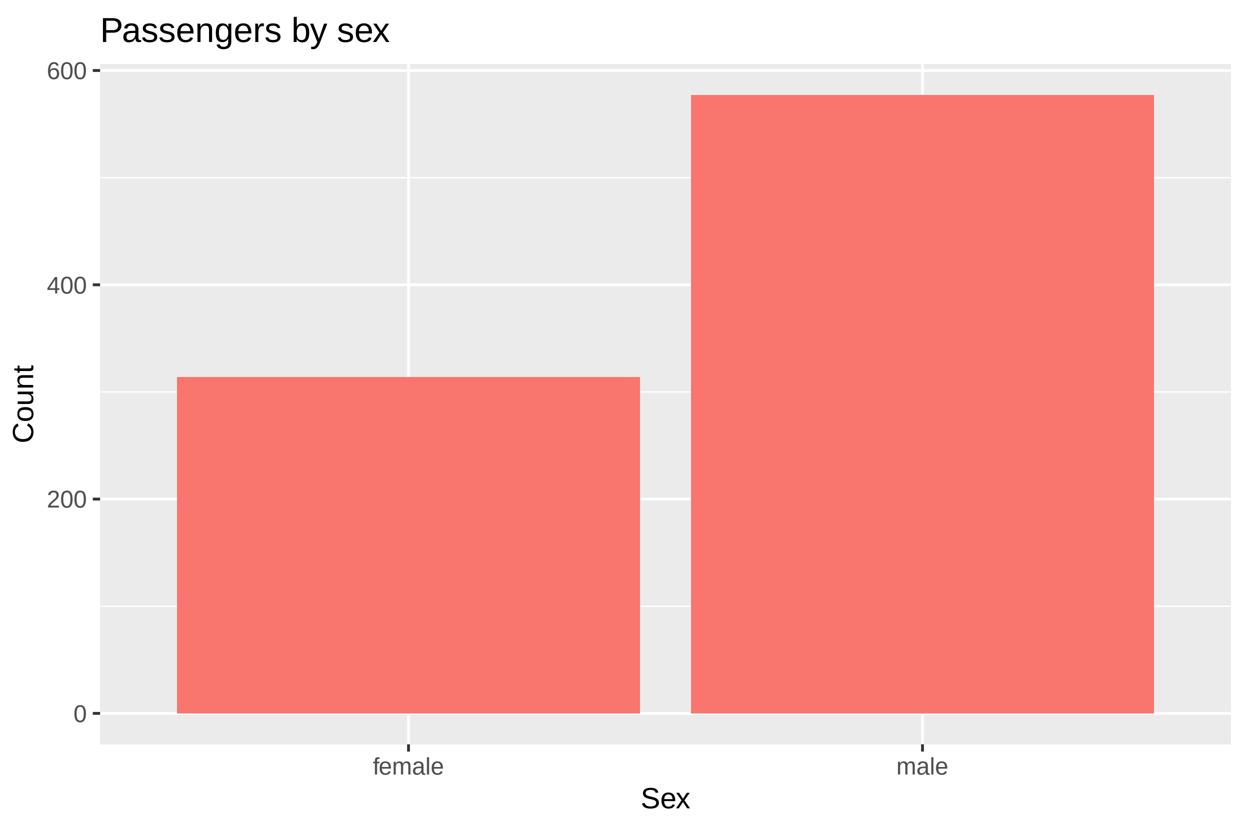


```
df_train %>%
  ggplot()+
  geom_bar(aes(Pclass, fill = "red"))+
  xlab("Passenger Class")+
  ylab("Count")+
  labs(title = "Passengers by class", caption = "We see that there are more on 3rd class than 1st and 2n
  theme(legend.position = "none")
```



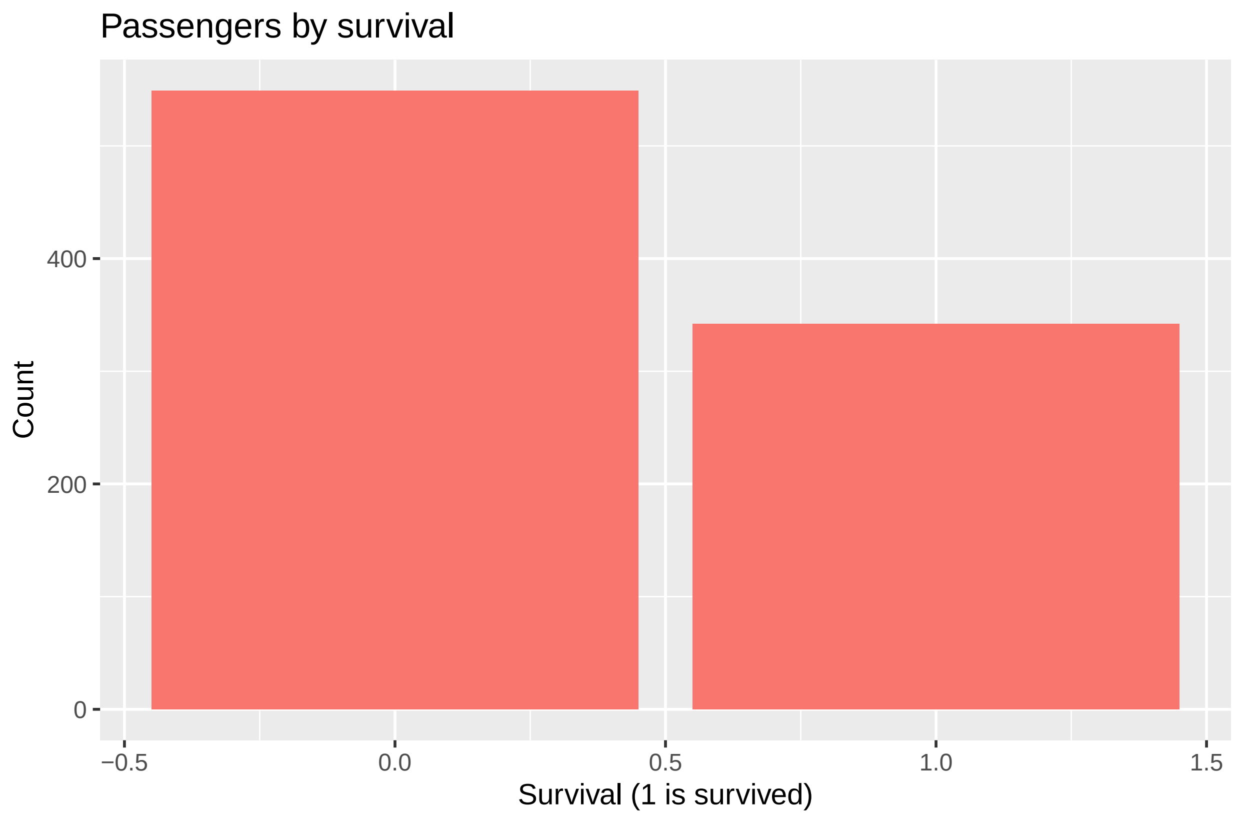
We see that there are more on 3rd class than 1st and 2nd combined.

```
df_train %>%
  ggplot()+
  geom_bar(aes(Sex, fill = "red"))+
  xlab("Sex")+
  ylab("Count")+
  labs(title = "Passengers by sex", caption = "Generally more men were aboard the Titanic than women, al
  theme(legend.position = "none")
```



Generally more men were aboard the Titanic than women, almost double the amount.

```
df_train %>%
  ggplot()+
  geom_bar(aes(Survived, fill = "red"))+
  xlab("Survival (1 is survived)")+
  ylab("Count")+
  labs(title = "Passengers by survival", caption = "Unfortunately, fewer people survived than did not.")
  theme(legend.position = "none")
```



Unfortunately, fewer people survived than did not.

2. Choose an estimand / outcome

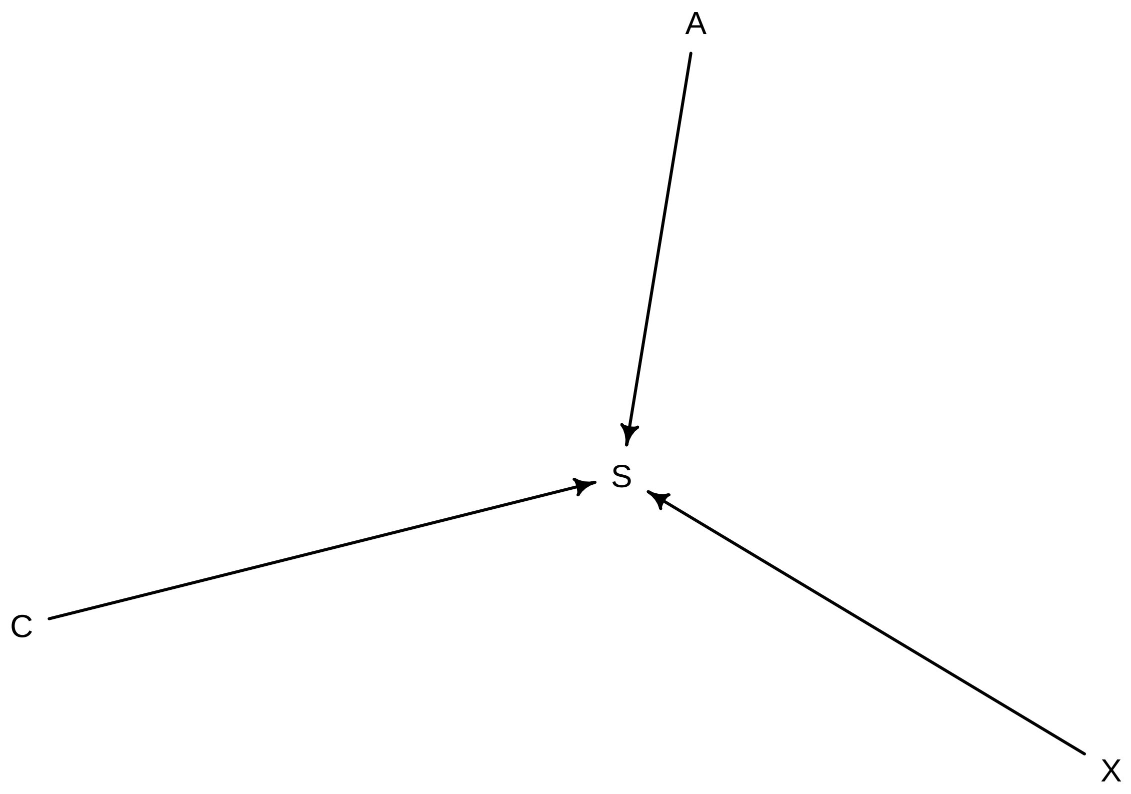
- **I will do as recommended and choose survival as my outcome variable. :)**

3. Make a scientific model (i.e., a DAG)

Make a DAG that seems theoretically reasonable, and that includes some of the variables in the dataset. It can include unobserved variables too, if you want, but then you have to come up with them.

You might have to return to this point later on ;)

```
dag <- dagitty( "dag {
    A -> S
    C -> S
    X -> S
}")

drawdag(dag)
```



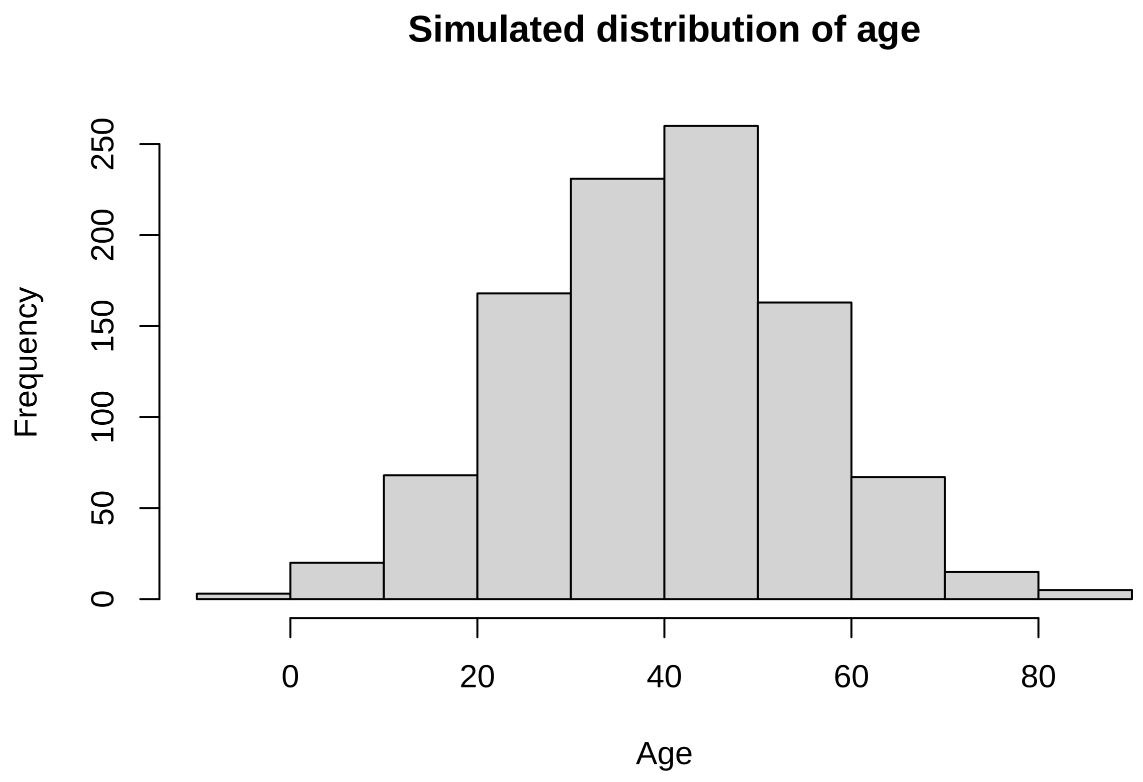
$$\begin{tikzcd} & & A \arrow[d] & \\ C \arrow[rr] & & S & X \arrow[l] \end{tikzcd}$$

Considering the variables available, i would expect survival to be predicted by sex, age and class. This is a very simple DAG, but it is based on my ignorance of the famous ship

4. Simulate data from the DAG

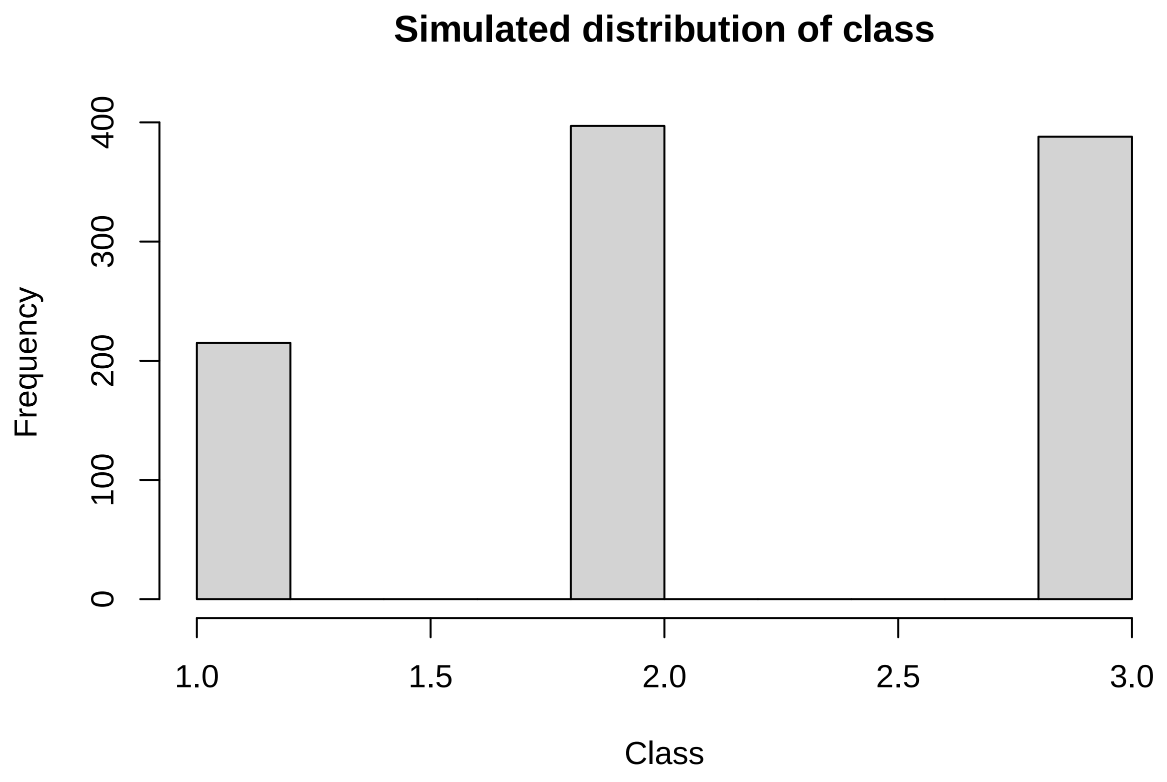
Age: I would expect a mean age of 40 and a std of 15. When sampling, some values might dip below 0, which of course isn't interpretable when dealing with age, but when simulating it should be okay. There were roughly 2200 people aboard the Titanic, but for simplicity I will sample 1000.

```
A = rnorm(1000, 40, 15)
hist(A, main = paste("Simulated distribution of age"), xlab = "Age")
```



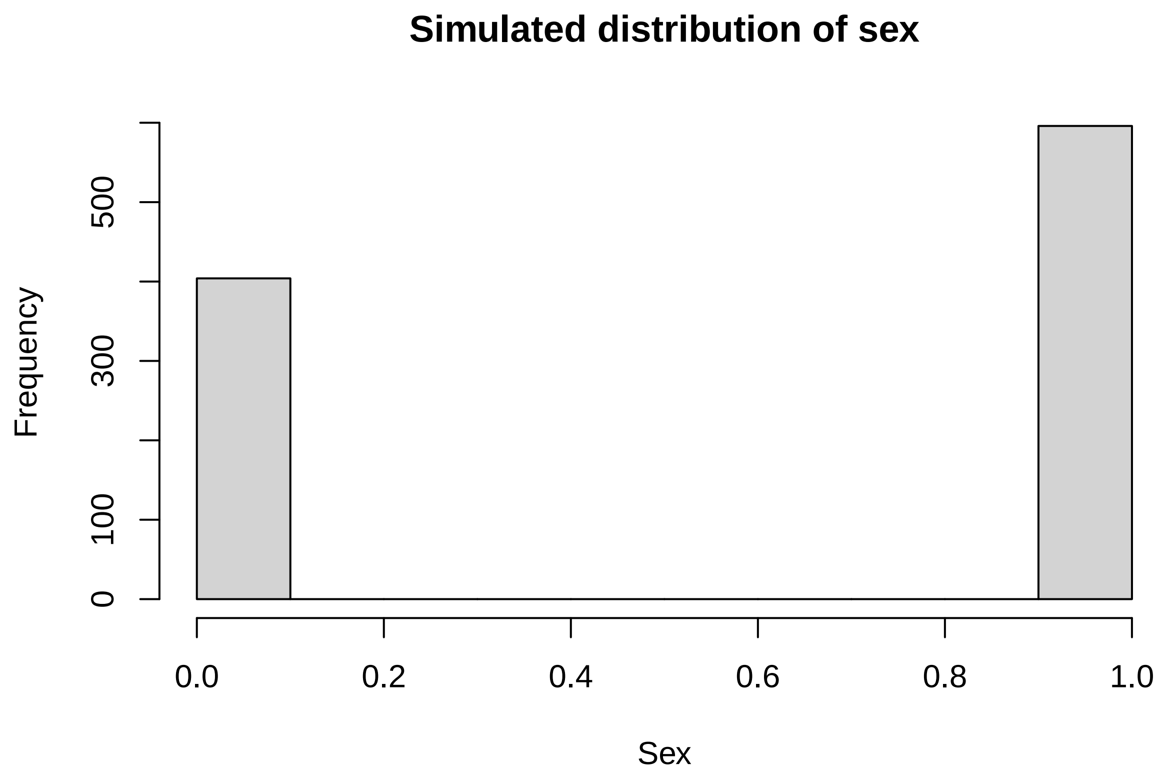
Class: I would expect less 1st class than 2nd and 3rd, as 1st class probably requires more space and requires more service.

```
C = sample(1:3, 1000, prob = c(0.2, 0.4, 0.4), replace=TRUE)
hist(C, main = paste("Simulated distribution of class"), xlab = "Class")
```



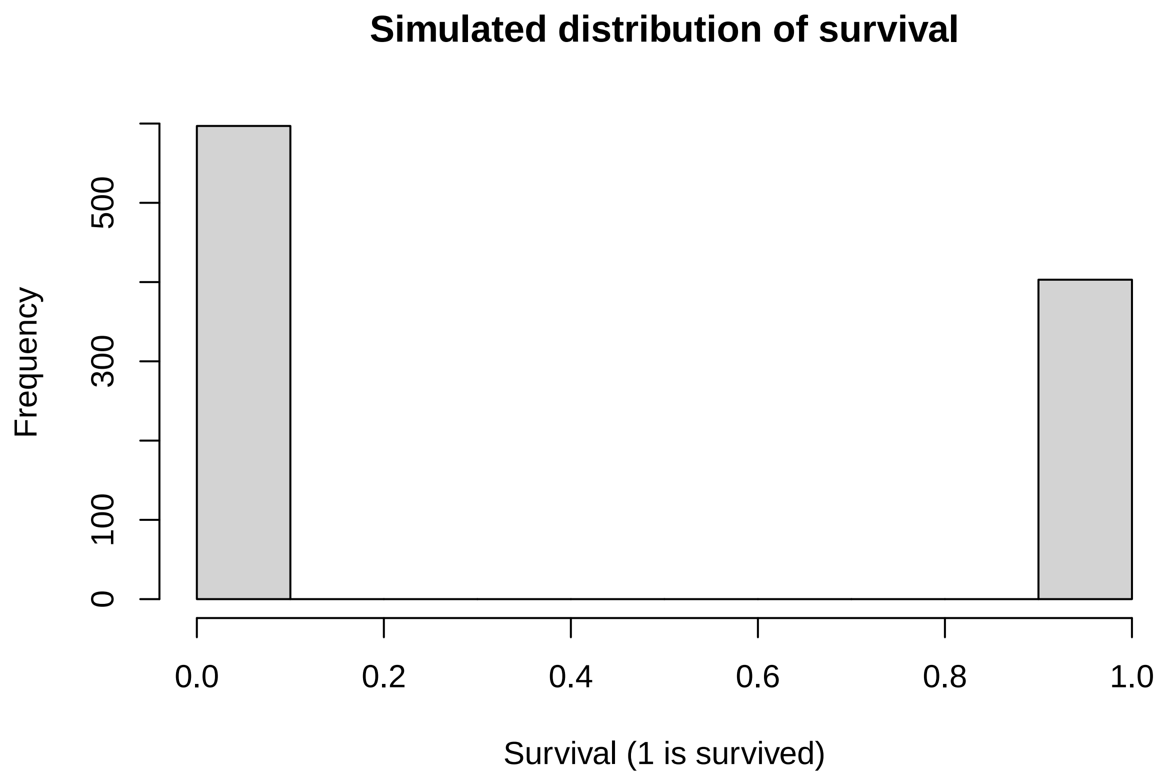
Sex: Based on the movie and stereotypes, I'd expect more men aboard than women due to the expensive tickets, and assuming that crew members are largely men. 1 = mean, 0 = women

```
X = rbinom(1000, 1, 0.6)
hist(X, main = paste("Simulated distribution of sex"), xlab = "Sex")
```



Survived: The “base” survival rate is sort of arbitrary, but i’d imagine less people to survive than not. Then again, i don’t know how likely people are to survive a sinking ship. The weightings are based on the following: Age is scaled and weighted so that younger people are more likely to survive, higher class are more likely to survive and women are more likely to survive.

```r
S = rbinom(1000, 1, prob = 0.4 + (-0.05)*scale(A) + (-0.05)*scale(C) + (-0.1)*scale(X))
hist(S, main = paste("Simulated distribution of survival"), xlab = "Survival (1 is survived)")
```



```
table(is.na(S))
```

```
##
## FALSE
##  1000
```

```
df_sim <- data.frame(A, C, X, S)
```

Based on the plots above, i'd say my simulated data isn't that far off from the real data.

5. Make a statistical model

The relevant predictors for predicting survival, as implied by the DAG, are age, sex and class, therefore S ~ A + X + C. In the model, survived is a binary variable (although treated as an integer), so the posterior is a binomial distribution. There are random slopes for sex and for class, and age is a fixed effect. Priors are chosen based on prior predictive checks. Interactions will be explored when modeling the real data later on.

6. Test the statistical model on the simulated data

```
# Standardizing
dat_sim = list(
  A = scale(df_sim$A),
  C = as.factor(df_sim$C),
  X = as.factor(df_sim$X),
  S = as.integer(df_sim$S)
)

# Using a modest prior
m1_1 <- ulam(
  alist(

    ## S ~ A + X + C

    S ~ dbinom(1, p),
    logit(p) <- aX[X] + aC[C] + bA*A,
    aX[X] ~ dnorm(0, 0.5),
    aC[C] ~ dnorm(0, 0.5),
    bA ~ dnorm(0, 0.5)

  ), data = dat_sim, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 5.7 seconds.
```

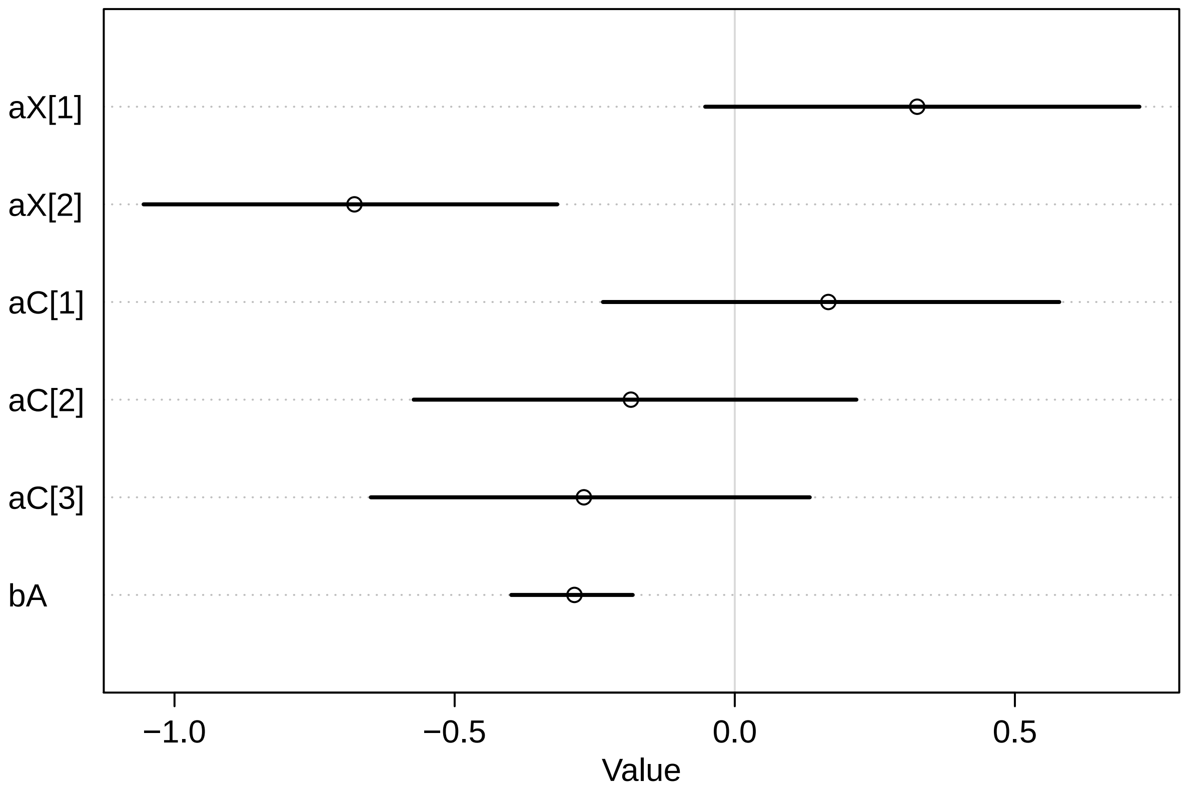
```
# Very narrow priors
m1_2 <- ulam(
  alist(

    ## S ~ A + X + C

    S ~ dbinom(1, p),
    logit(p) <- aX[X] + aC[C] + bA*A,
    aX[X] ~ dnorm(0, 0.0001),
    aC[C] ~ dnorm(0, 0.0001),
    bA ~ dnorm(0, 0.0001)

  ), data = dat_sim, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 3.7 seconds.
```

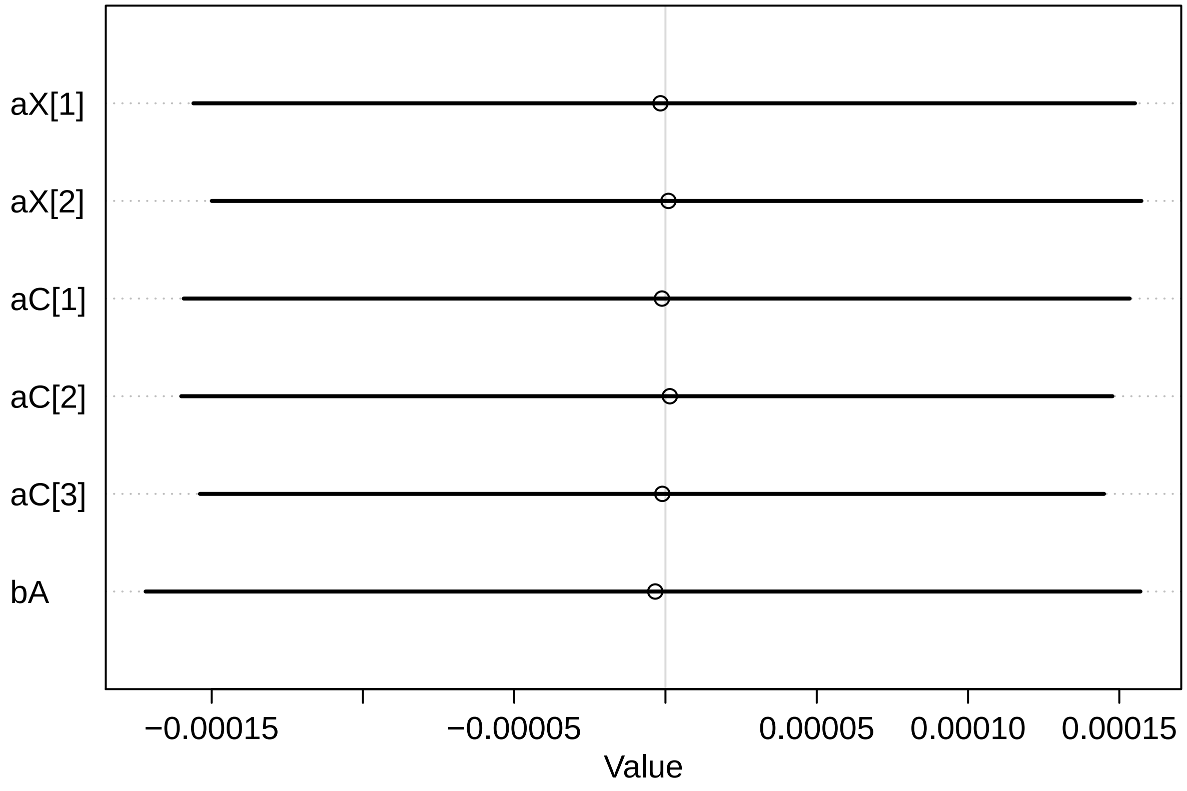
```
# Very large priors
m1_3 <- ulam(
  alist(

    ## S ~ A + X + C

    S ~ dbinom(1, p),
    logit(p) <- aX[X] + aC[C] + bA*A,
    aX[X] ~ dnorm(0, 10),
    aC[C] ~ dnorm(0, 10),
    bA ~ dnorm(0, 10)

  ), data = dat_sim, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 25.1 seconds.
```

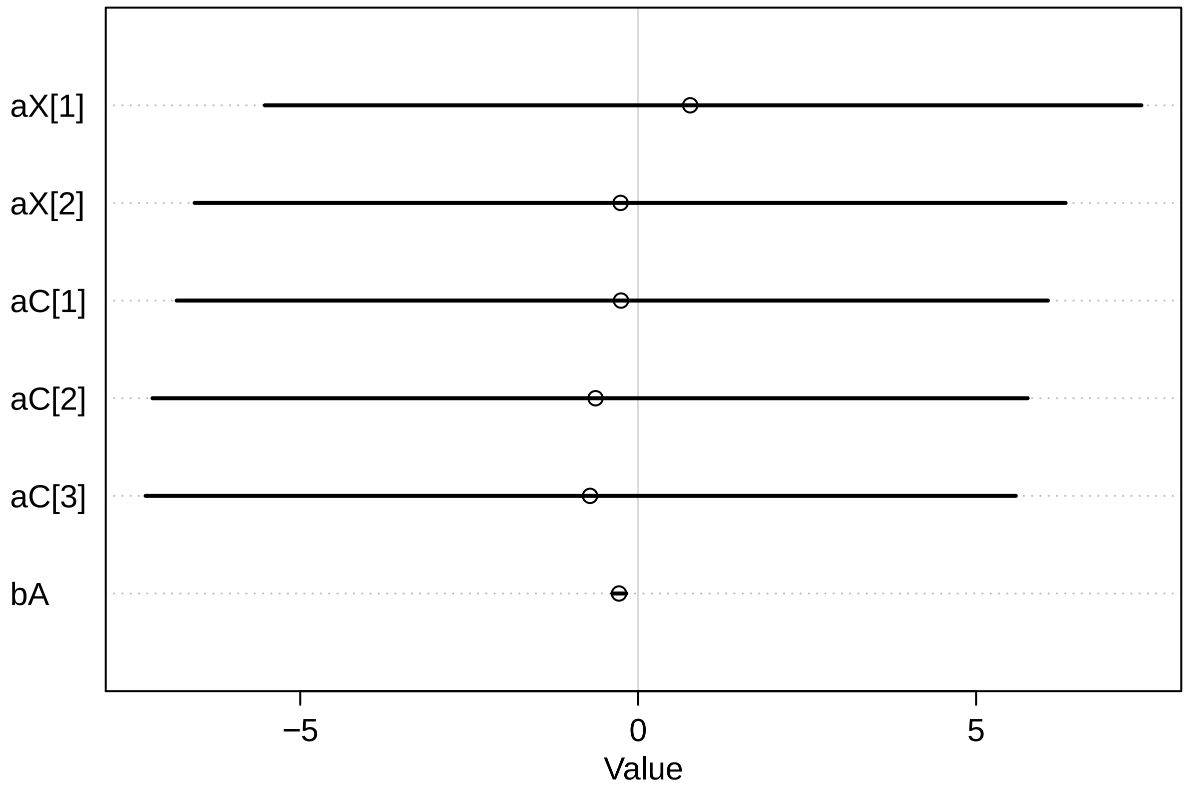
```
precis_plot(precis(m1_1, depth = 2))
```



```
precis_plot(precis(m1_2, depth = 2))
```



```
precis_plot(precis(m1_3, depth = 2))
```



This is the statistical model including all variables, congruent with the DAG. This shows indeed that sex has an effect on survival with men dying more than women. Class also affects whether passengers survive or not, with 1st class being most safe, then 2nd and then 3rd. The slope parameter for age shows that older passengers are less likely to survive. The model reflects the intuitions which the data was simulated on. The priors I've chosen for all three parameters are (0, 0.5), since these look most reasonable in regards to my expectations of the values. The too narrow priors make the effects crazy small, and increases uncertainty wildly. Too broad priors have the same effect but on a larger scale. I will stick with priors of mean 0 and std 0.5.

A note on priors

```
# Extracting priors
prior1 <- extract.prior(m1_1)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 2.8 seconds.
```

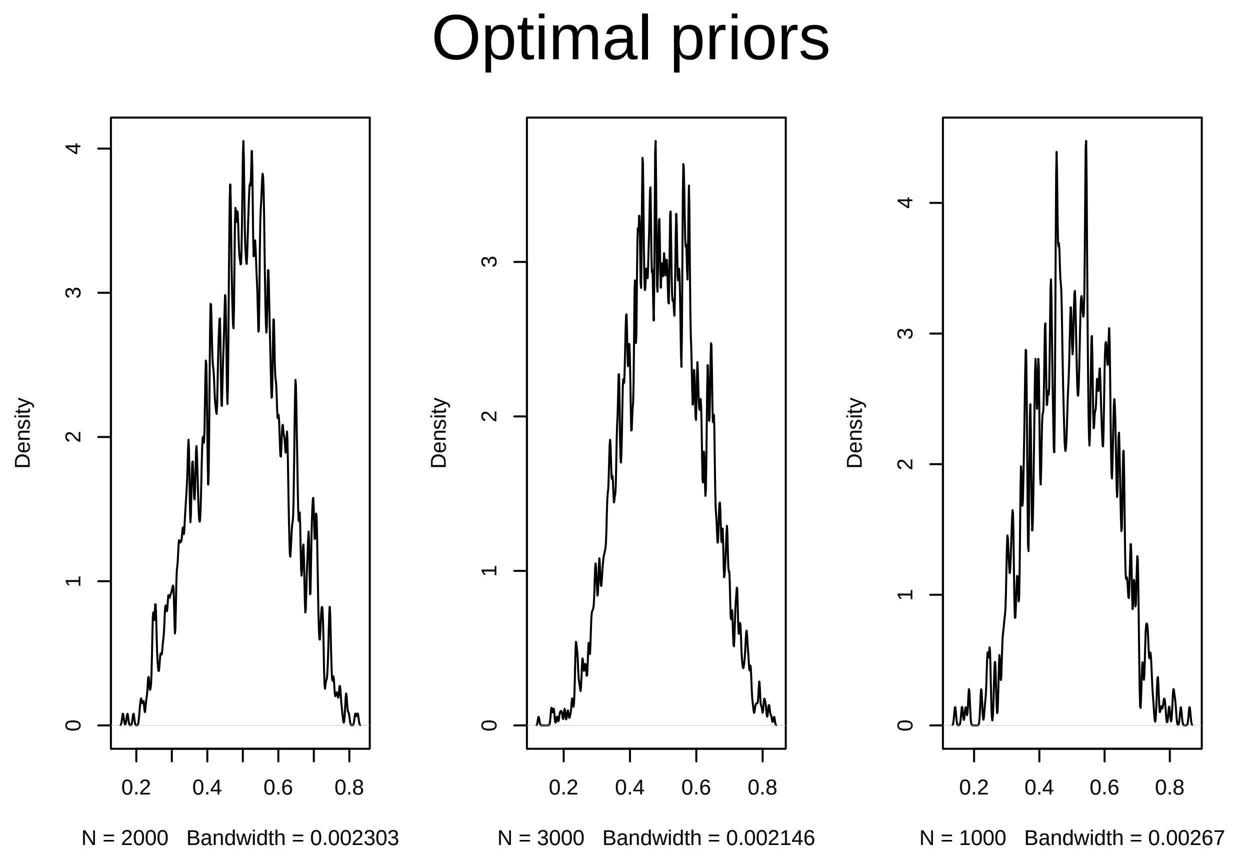
```
prior2 <- extract.prior(m1_2)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 3.8 seconds.
```

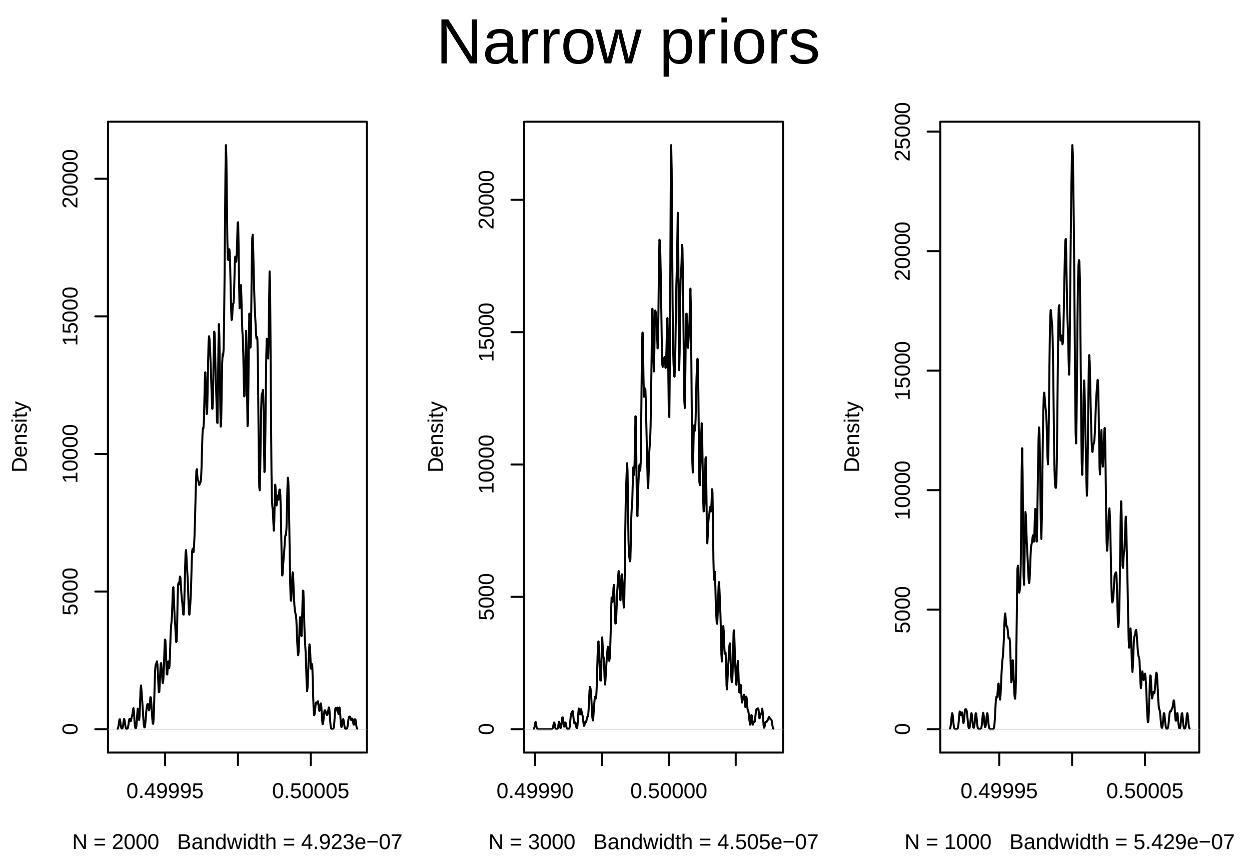
```
prior3 <- extract.prior(m1_3)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 2.8 seconds.
```

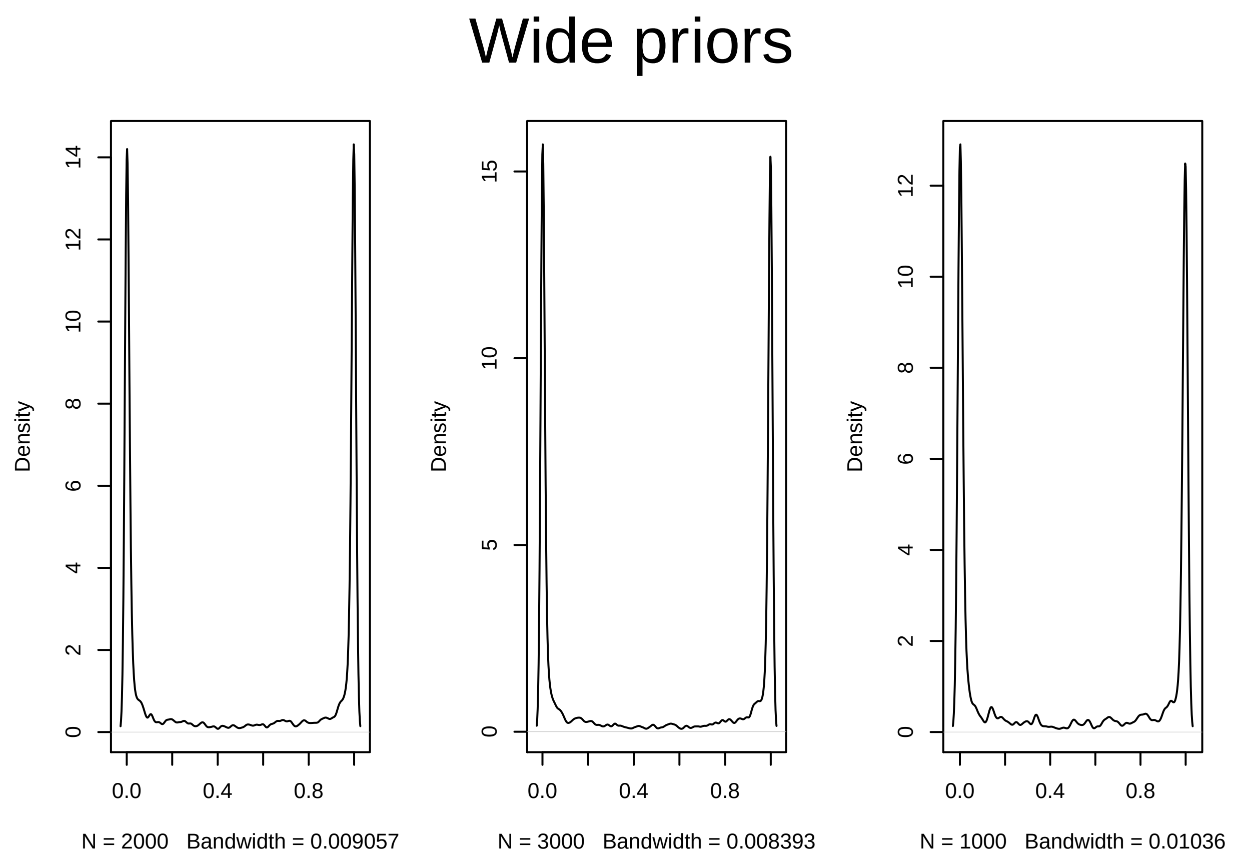
```
par(mfrow=c(1,3))
dens(inv_logit(prior1$aX), adj=0.1)
dens(inv_logit(prior1$aC), adj=0.1)
dens(inv_logit(prior1$bA), adj=0.1)
mtext("Optimal priors", side=3, line=-2, cex=2, outer = TRUE)
```



```
par(mfrow=c(1,3))
dens(inv_logit(prior2$aX), adj=0.1)
dens(inv_logit(prior2$aC), adj=0.1)
dens(inv_logit(prior2$bA), adj=0.1)
mtext("Narrow priors", side=3, line=-2, cex=2, outer = TRUE)
```



```
par(mfrow=c(1,3))
dens(inv_logit(prior3$aX), adj=0.1)
dens(inv_logit(prior3$aC), adj=0.1)
dens(inv_logit(prior3$bA), adj=0.1)
mtext("Wide priors", side=3, line=-2, cex=2, outer = TRUE)
```



These density plots confirm that my choice of priors (optimal) are at least reasonable. When using priors with a std of 10, the model is very sure of either surival or not, which i know isn't the case in the data. When using very narrow priors, it is highly unlikely that the real values are in this distribution space.

Testing conditional independencies

```r
impliedConditionalIndependencies(dag)
```

```
## A _||_ C
## A _||_ X
## C _||_ X
```

```r
# A _||_ C
m3 <- ulam(
  alist(

    ## A ~ C

    A ~ dnorm(mu, sigma),
    mu <- aC[C],
    aC[C] ~ dnorm(0, 0.5),
    sigma ~ dexp(1)
  ), data = dat_sim, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 2.2 seconds.
```

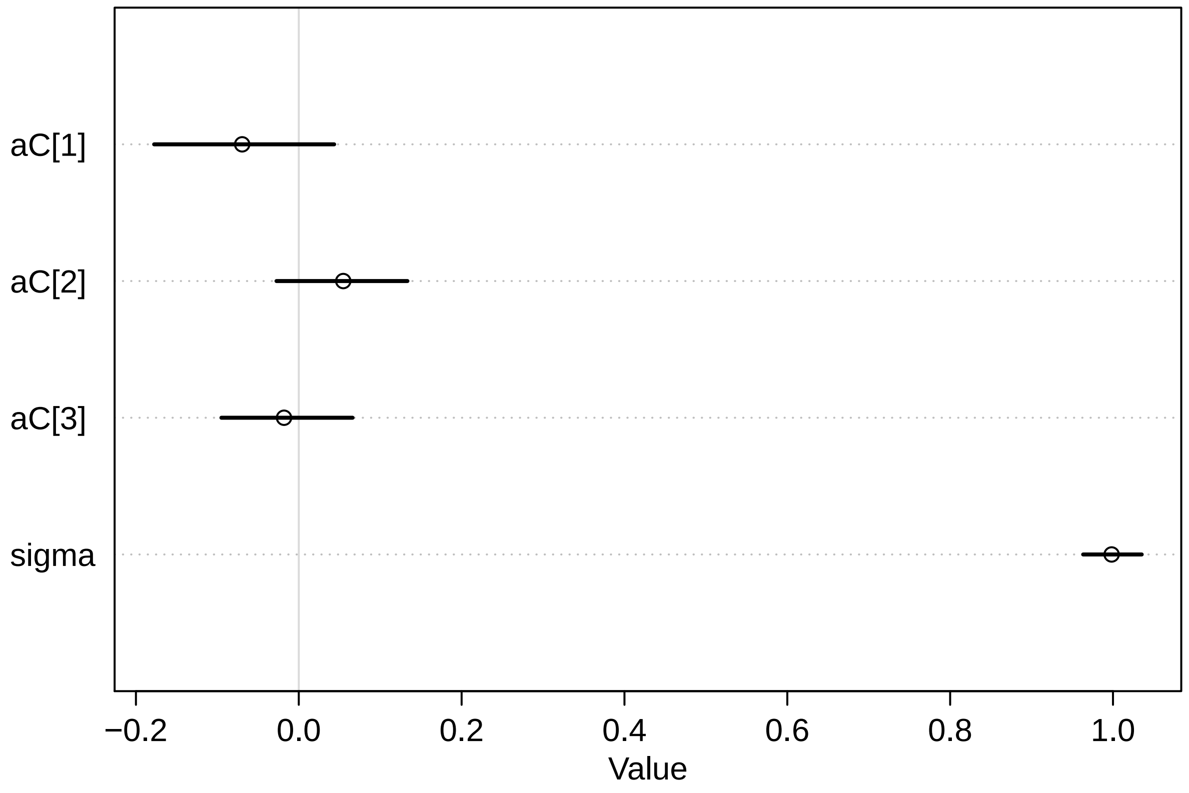
```r
# A _||_ X
# I need X as integer for it to be the outcome, otherwise i get an error
dat_sim_X <- dat_sim
dat_sim_X$X <- as.integer(dat_sim_X$X)-1

m4 <- ulam(
  alist(

    ## X ~ A

    X ~ dbinom(1, p),
    logit(p) <- a + bA*A,
    a ~ dnorm(0, 0.5),
    bA ~ dnorm(0, 0.5)

  ), data = dat_sim_X, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 3.5 seconds.
```

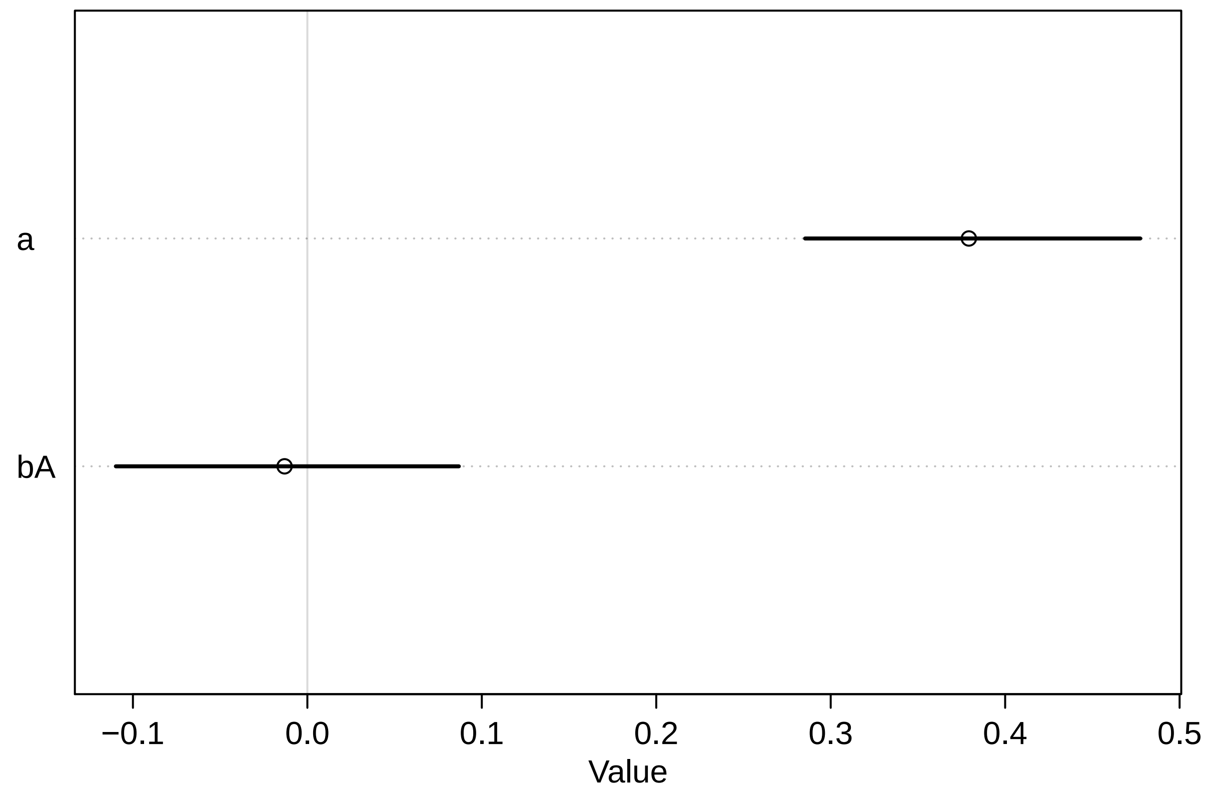
```r
# C _||_ X
m5 <- ulam(
  alist(

    ## X ~ C

    X ~ dbinom(1, p),
    logit(p) <- aC[C],
    aC[C] ~ dnorm(0, 0.5)

  ), data = dat_sim_X, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 3.6 seconds.
```

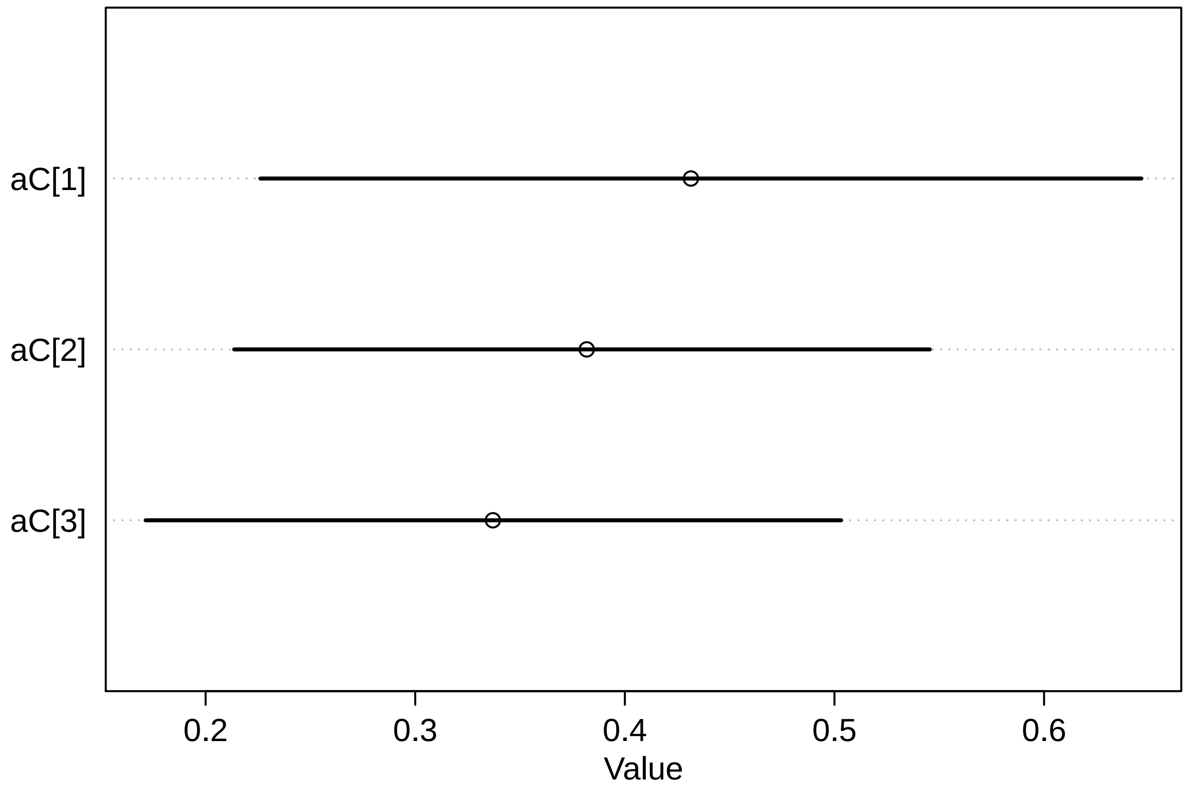
```r
precis_plot(precis(m3, depth = 2))
```



```r
precis_plot(precis(m4, depth = 2))
```



```r
precis_plot(precis(m5, depth = 2))
```



Since all my predictors are completely independent of each other (i.e. there is no other variables which must be conditioned upon when modeling), I've made three models, to assess whether these three are truly independent. If so, there should be little to no predictive power of modeling one variable by another. This seems to be true for sex on class and sex on age, but there seems to be some effect of age on class. Since i have simulated the data, I know that these to variables are independent of each other, so I must conclude that the correlation is merely spurious.

7. Assess whether the DAG is compatible with the data

```
adjustmentSets(dag, exposure = "X", outcome = "S")
```

```
##  {}
```

I do not need to stratify on any variables to get the individual effects of my variables. Below is the total effects model using the real data.

```
# Preprocessing data
df_train <- df_train %>%
  select(Survived, Pclass, Age, Sex)

# Omitting NAs
df_train <- na.omit(df_train)
```

```
# Standardizing
dat = list(

  A = scale(df_train$Age),
  C = as.factor(df_train$Pclass),
  X = as.factor(df_train$Sex),
  S = as.integer(df_train$Survived)
)

# Total effects
m1_real <- ulam(
  alist(

    ## S ~ A + X + C

    S ~ dbinom(1, p),
    logit(p) <- aX[X] + aC[C] + bA*A,
    aX[X] ~ dnorm(0, 0.5),
    aC[C] ~ dnorm(0, 0.5),
    bA ~ dnorm(0, 0.5)

  ), data = dat, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 4.0 seconds.
```

Running the model with the real data yields quite similar parameter values as when using my simulated data. So far so good.

```
impliedConditionalIndependencies(dag)
```

```
## A _||_ C
## A _||_ X
## C _||_ X
```

```
# Testing conditional independencies

# A _||_ C
m6 <- ulam(
  alist(

    ## A ~ C

    A ~ dnorm(mu, sigma),
    mu <- aC[C],
    aC[C] ~ dnorm(0, 0.5),
    sigma ~ dexp(1)

  ), data = dat, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 1.4 seconds.
```

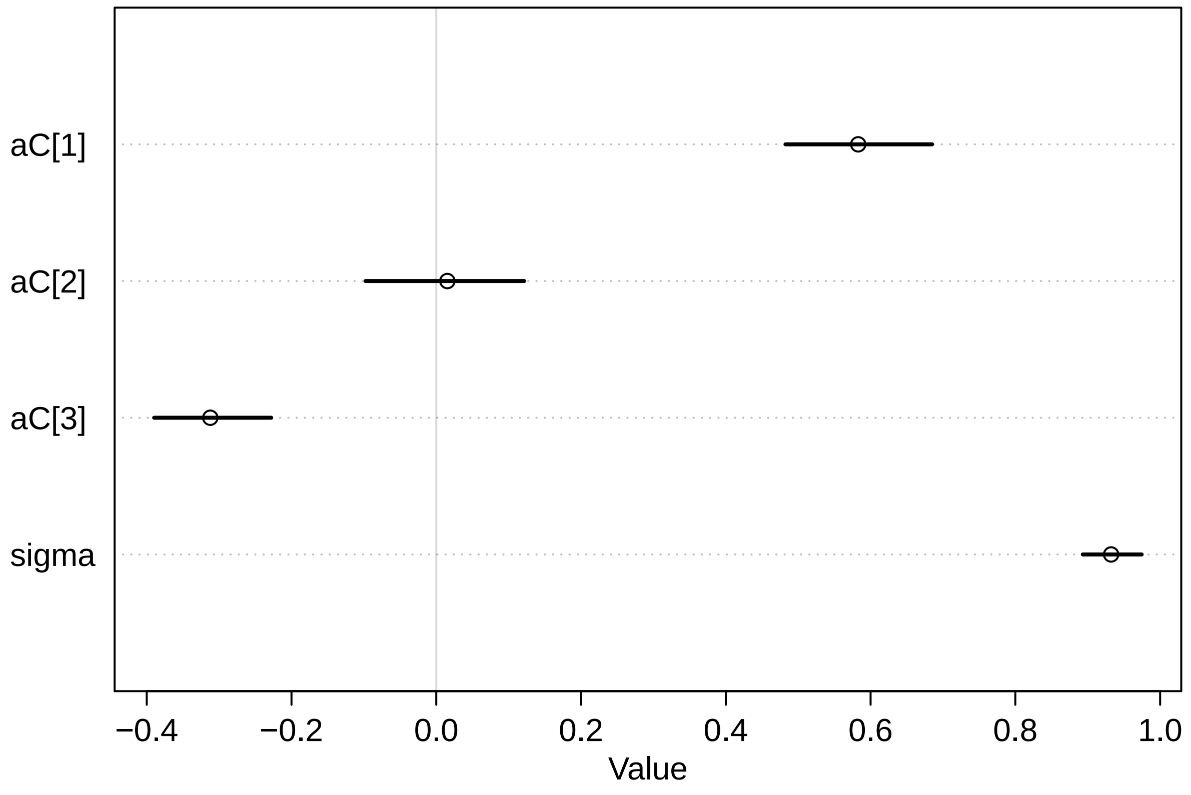
```
# A _||_ X
# I need X as integer for it to be the outcome, otherwise i get an error
dat_X <- dat
dat_X$X <- as.integer(dat_X$X)-1

m7 <- ulam(
  alist(

    ## X ~ A

    X ~ dbinom(1, p),
    logit(p) <- a + bA*A,
    a ~ dnorm(0, 0.2),
    bA ~ dnorm(0, 0.5)

  ), data = dat_X, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 2.1 seconds.
```

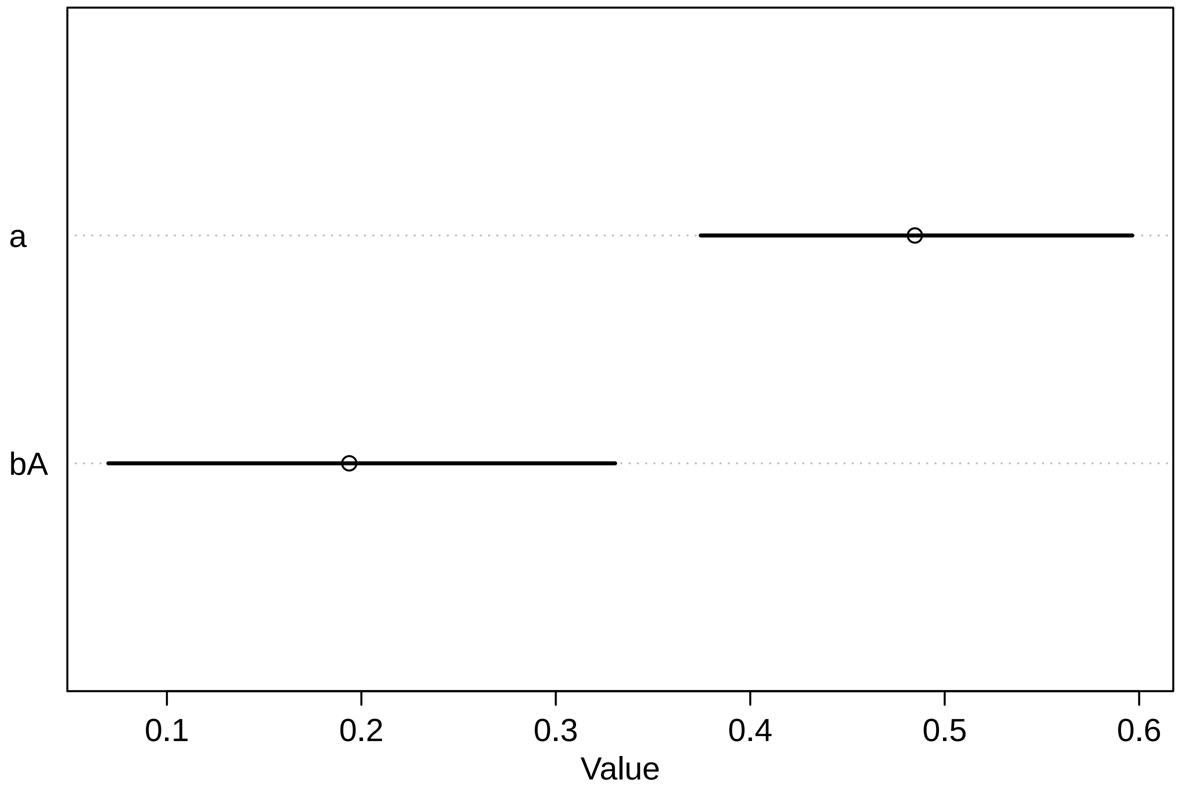
```
# C _||_ X
m8 <- ulam(
  alist(

    ## X ~ C

    X ~ dbinom(1, p),
    logit(p) <- aC[C],
    aC[C] ~ dnorm(0, 0.5)

  ), data = dat_X, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 2.0 seconds.
```

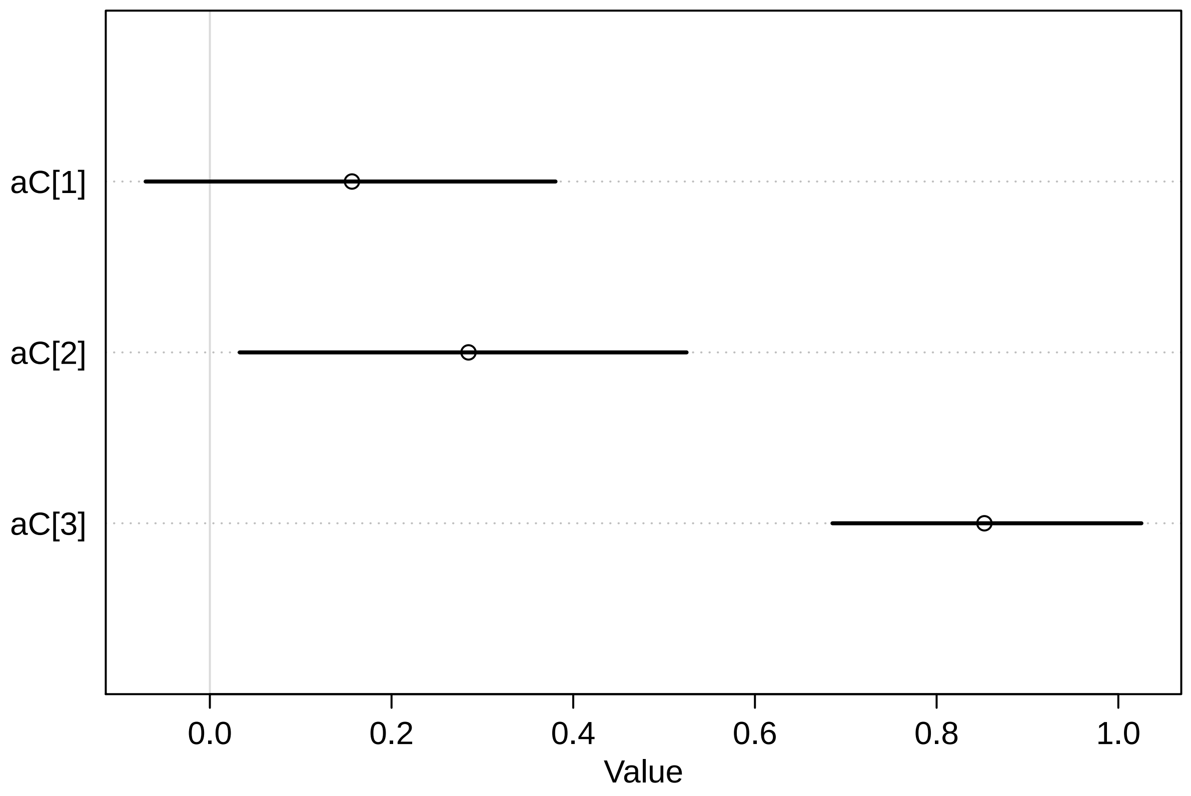
```
precis_plot(precis(m6, depth = 2))
```



```r
precis_plot(precis(m7, depth = 2))
```

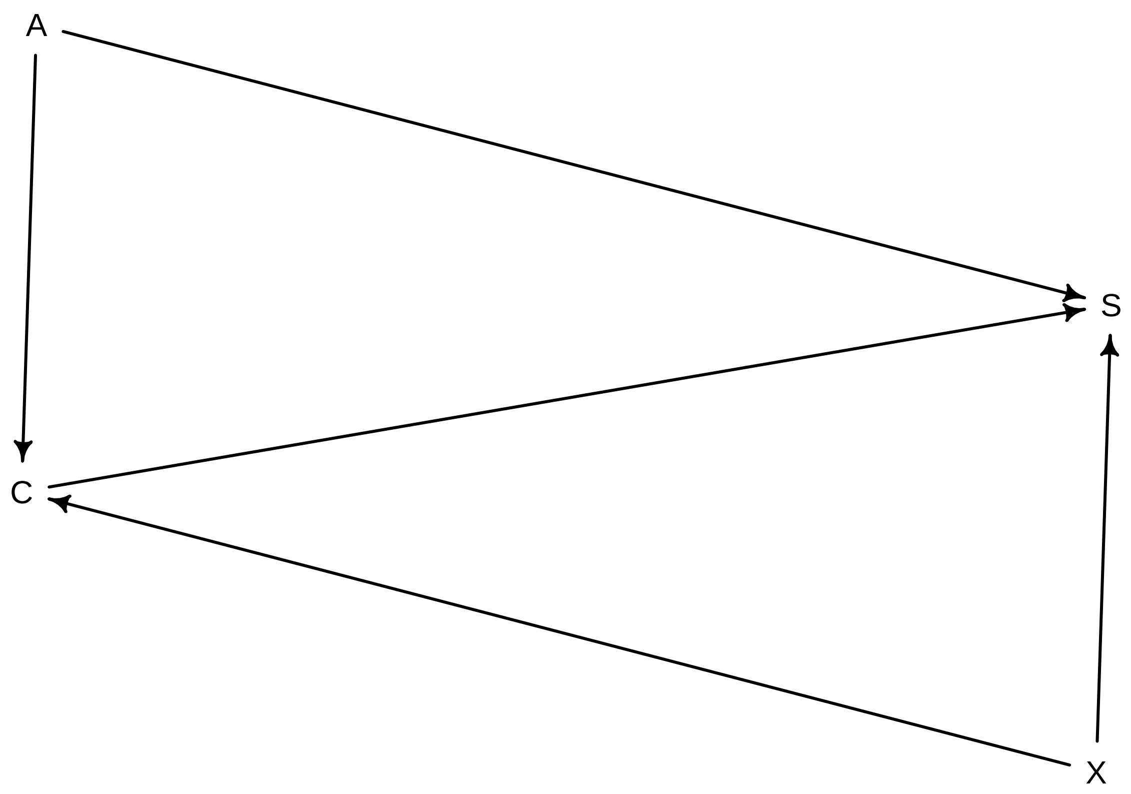


```
precis_plot(precis(m8, depth = 2))
```



It appears that i was too naive. In the real data, both age and sex seems to predict class in some way. I'll incorporate this in a new DAG.

```
dag2 <- dagitty( "dag {
    A -> S
    A -> C
    C -> S
    X -> S
    X -> C
}")

drawdag(dag2)
```


$$\begin{tikzcd}
A \arrow[rrr] \arrow[d] & & & S \\
C \arrow[rrru] & & & \\
& & & X \arrow[uu] \arrow[lllu]
\end{tikzcd}$$

This also makes for a much more interesting DAG. I'll check the conditional independencies

```
impliedConditionalIndependencies(dag2)
```

```
## A _||_ X
```

```
adjustmentSets(dag2, exposure = "A", outcome = "S")
```

```
##  {}
```

```
adjustmentSets(dag2, exposure = "C", outcome = "S")
```

```
## { A, X }
```

```
adjustmentSets(dag2, exposure = "X", outcome = "S")
```

```
##  {}
```

This doesn't change much regarding the statistical model structure, as I now know that i must include all three variables to not have any backdoors open. Therefore i will continue to include all three predictors in my model.

8. Do model comparison

Comparing my models with different priors

```
# Modeling the real data with different priors

# Very narrow priors
m2_real <- ulam(
  alist(

    ## S ~ A + X + C

    S ~ dbinom(1, p),
    logit(p) <- aX[X] + aC[C] + bA*A,
    aX[X] ~ dnorm(0, 0.0001),
    aC[C] ~ dnorm(0, 0.0001),
    bA ~ dnorm(0, 0.0001)

  ), data = dat, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 2.8 seconds.
```

```
# Very broad priors
m3_real <- ulam(
  alist(

    ## S ~ A + X + C

    S ~ dbinom(1, p),
    logit(p) <- aX[X] + aC[C] + bA*A,
    aX[X] ~ dnorm(0, 10),
    aC[C] ~ dnorm(0, 10),
    bA ~ dnorm(0, 10)

  ), data = dat, log_lik = TRUE, refresh = 0
)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 14.0 seconds.
```

```
# Comparing model with different priors

compare(m1_real, m2_real, m3_real)
```

```
##             WAIC           SE       dWAIC       dSE        pWAIC       weight
## m3_real 657.7938 3.137852e+01   0.0000000        NA 5.164043e+00 5.278468e-01
## m1_real 658.0168 2.840432e+01   0.2230049  3.353378 3.916188e+00 4.721532e-01
## m2_real 989.8152 1.108017e-04 332.0214070 31.400565 5.183112e-06 4.216726e-73
```

```
compare(m1_real, m2_real, m3_real, func = PSIS)
```

```
##              PSIS           SE       dPSIS       dSE        pPSIS       weight
## m3_real 657.8901 3.140563e+01   0.0000000        NA 5.212205e+00 5.246856e-01
## m1_real 658.0877 2.842773e+01   0.1976451  3.354852 3.951670e+00 4.753144e-01
## m2_real 989.8152 1.108797e-04 331.9250840 31.405676 5.219084e-06 4.398280e-73
```

Even though the model with too broad priors has lower WAIC and PSIS score than my optimal-prior model, I know that this model doesn't make sense with my knowledge of the data.

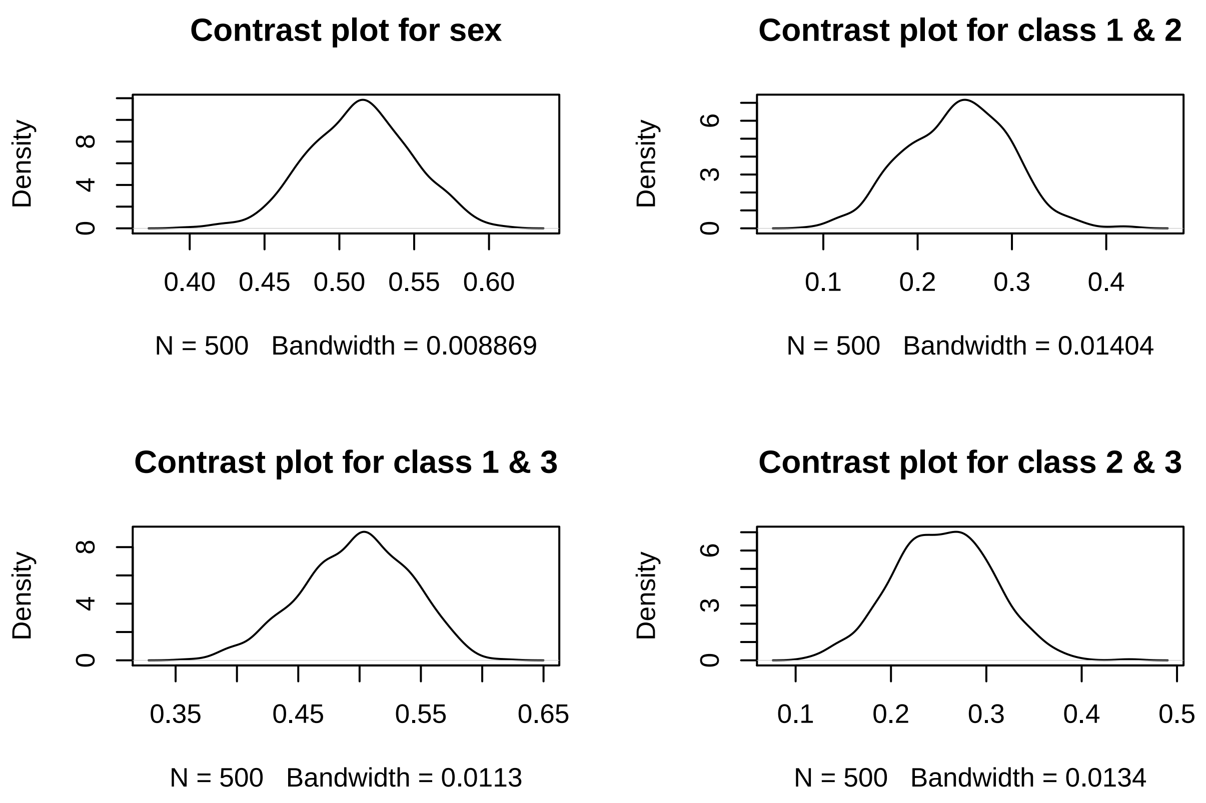
Making contrast plots for the model with optimal priors

```
# Contrast plots
posterior1 <- extract.samples(m1_real)
```

```
precis(m1_real, depth = 3)
```

```
##               mean         sd       5.5%      94.5%    n_eff     Rhat4
## aX[1]  1.19732218 0.23380651  0.8474721  1.5754515 192.6123 0.9997944
## aX[2] -1.10271850 0.22654923 -1.4429459 -0.7313372 152.7867 0.9987861
## aC[1]  1.12221616 0.24673582  0.7624285  1.5230088 162.0975 1.0045081
## aC[2]  0.03744118 0.26092560 -0.3609912  0.4723566 159.4691 0.9983063
## aC[3] -1.08966322 0.24374227 -1.4732511 -0.6996403 172.6997 0.9999650
## bA    -0.45877469 0.09627833 -0.6157826 -0.2965598 284.3390 1.0037926
```

```
par(mfrow=c(2,2))
plot(density(inv_logit(posterior1$aX[,1]) - inv_logit(posterior1$aX[,2])), main="Contrast plot for sex")
plot(density(inv_logit(posterior1$aC[,1]) - inv_logit(posterior1$aC[,2])), main="Contrast plot for clas
plot(density(inv_logit(posterior1$aC[,1]) - inv_logit(posterior1$aC[,3])), main="Contrast plot for clas
plot(density(inv_logit(posterior1$aC[,2]) - inv_logit(posterior1$aC[,3])), main="Contrast plot for clas
```



The contrast plot for sex shows that women have somewhere between 50-55% higher chance of surviving than men. Passengers traveling on 1st class are somewhere between 20-25% more likely to survive than passengers on 2nd class, and somewhere between 47-52% more likely to survive than passengers on 3rd class. Passengers on 2nd class are somewhere between 26-29% more likely to survive than passengers on 3rd class.

Trying out a model with an interaction between sex and class

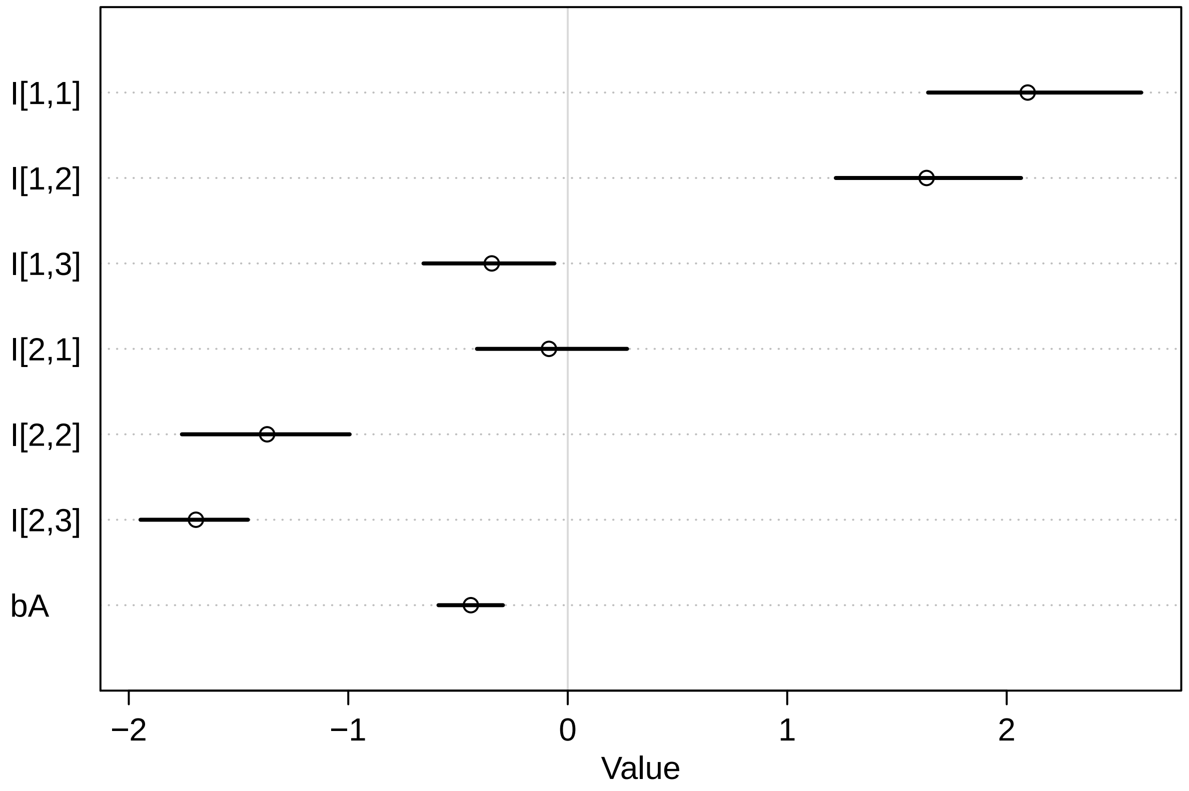
```
# Interaction between sex and class
m1_real_int <- ulam(
  alist(

    #

    S ~ dbinom(1, p),
    logit(p) <- I[X,C]+ bA*A,
    matrix[X,C]:I ~ normal(0, 0.5),
    bA ~ dnorm(0, 0.5)
    ), data=dat, cores = 4, log_lik = TRUE, refresh = 0)
```

```
## Running MCMC with 1 chain, with 1 thread(s) per chain...
##
## Chain 1 finished in 2.4 seconds.
```

```
# Checking the parameter estimates
precis(m1_real_int, depth=3)
```

```
##                mean         sd       5.5%       94.5%    n_eff     Rhat4
## I[1,1]  2.09573442 0.28804123  1.6418713  2.61295635 946.4364 1.0084985
## I[1,2]  1.63500270 0.26523794  1.2212565  2.06553130 620.1547 0.9988637
## I[1,3] -0.34608945 0.18947653 -0.6573058 -0.06086797 516.7923 0.9980583
## I[2,1] -0.08556839 0.20766112 -0.4136982  0.27041044 604.3273 0.9979988
## I[2,2] -1.36954964 0.22995236 -1.7581423 -0.99358299 629.3549 0.9984277
## I[2,3] -1.69413646 0.15272819 -1.9466989 -1.45697130 519.5613 0.9982090
## bA     -0.44141394 0.09813678 -0.5889760 -0.29520627 527.3032 0.9986198
```

```
precis_plot(precis(m1_real_int, 3))
```



The interaction model shows that for both sexes, 1st class is the safest. Though for females, 1st and 2nd class are close to each other, with 3rd class being less safe. For males, both 2nd and 3rd class are relatively unsafe compared to 1st class.

```
# Comparing interaction model to no interaction model
compare(m1_real, m1_real_int)
```

```
##                 WAIC       SE    dWAIC      dSE    pWAIC      weight
## m1_real_int 643.4044 24.72788  0.00000       NA 4.683829 0.999329062
## m1_real     658.0168 28.40432 14.61233 7.935132 3.916188 0.000670938
```

The interaction model also has a slightly lower WAIC score, and a smaller standard error as well.

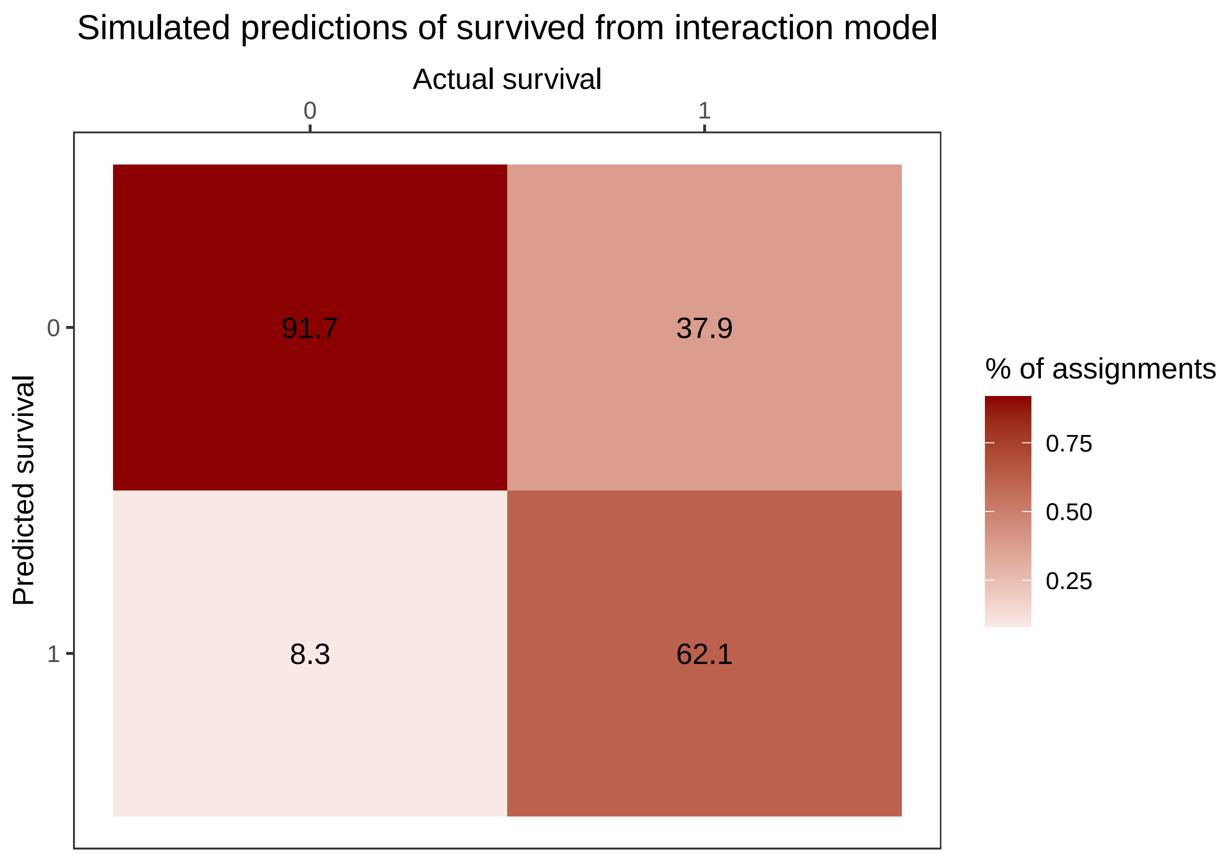
9. Use the statistical model to do inference

```
# Effect sizes and posterior predictions
post_int <- extract.samples(m1_real_int)
post_precis <- precis(post_int, depth=3)
survive_probability <- inv_logit(post_precis$mean)
Category <- c("Woman 1. Class", "Woman 2. Class", "Woman 3. Class", "Men 1. Class", "Men 2. Class", "Men

df <- tibble(Category, survive_probability)
df <- df %>%
  filter(Category != "Age")

df
```

```
## # A tibble: 6 x 2
##   Category       survive_probability
##   <chr>                        <dbl>
## 1 Woman 1. Class               0.890
## 2 Woman 2. Class               0.479
## 3 Woman 3. Class               0.837
## 4 Men 1. Class                 0.203
## 5 Men 2. Class                 0.414
## 6 Men 3. Class                 0.155
```

```
# Using model to predict survival on new data
survival_pred <- sim(m1_real_int, data=dat)
prediction <- round(colMeans(survival_pred), 0)
survived <- dat$S
survived_df <- data.frame(survived, prediction)
```

```
# Plotting confusion matrix to show predictions
pacman::p_load(caret, scales)
survived_df %>%
  count(survived, prediction) %>%
  mutate(across(c(survived, prediction), ~str_wrap(., 20))) %>%
  group_by(survived) %>%
  mutate(percent = n / sum(n)) %>%
  ggplot(aes(x = survived, y = reorder(prediction, desc(prediction)), fill = percent)) +
  geom_tile() +
  scale_fill_gradient2(high = "darkred") +
  geom_text(aes(label = round(percent*100, digits = 1))) +
  scale_x_discrete(position = "top") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 0, hjust = 0.5),
        panel.grid = element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  labs(x = "Actual survival",
       y = "Predicted survival",
       fill = "% of assignments",
       title = "Simulated predictions of survived from interaction model")
```



The data frame shows the average predictions for survival based on the interaction of class and sex with age as a fixed effect.

Aboard the Titanic, women travelling on 1st and 3rd class had highest probability of surviving. Both men and women on 2nd class were around chance-level of surviving, a bit lower for men though. Ultimately, according to the model, men on 1st and 3rd class had a very small chance of survival. The estimated parameter for age shows that as age increases, people are less likely to survive, although the effect isn't that large.

Unfortunately, my DAG does not fit the data. Through further exploration of the data, better inference could be made.

The table below shows the simulated predictions for survival based on the interaction model. It has high accuracy predicting deaths, but my model doesn't predict survival that well (many false alarms / type 2 errors). This is expected as my data doesn't fit the DAG.