Methods 4 – Portfolio Assignment 2

```
pacman::p_load(rethinking, dagitty, tidyverse, dplyr, ggplot2, robustHD, purrr,
               knitr, tinytex)
```

Task 1: The DAG (LR)

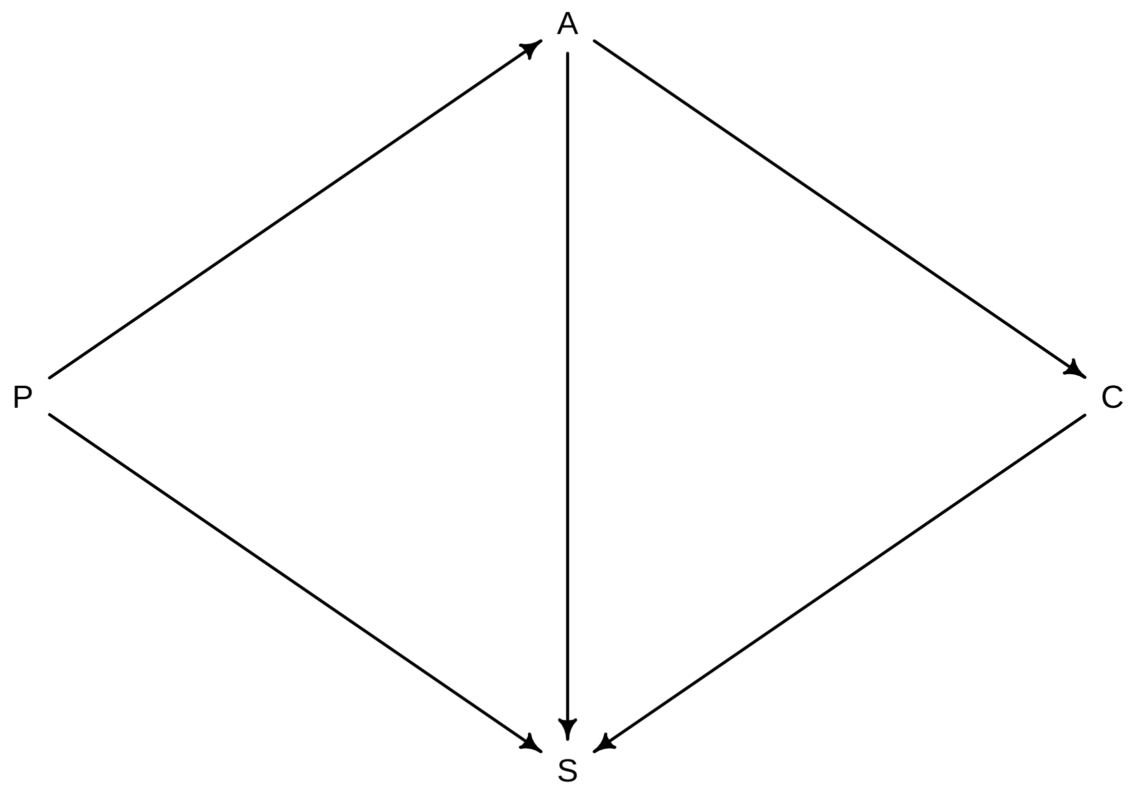
A) Come up with an incredibly interesting and scientifically important made-up *example* for a phenomenon to investigate. Decide on two variables (an outcome and a predictor) that you would like to investigate the relation between.

We will investigate the relation between sleep and alcohol consumption. We assume that the probability of having a healthy sleep pattern depends on: - How many alcoholic drink are consumed (if any) (predictor) - Furthermore, the consumption of alcohol is dependent on whether you attended a party or not. - Lastly, we assume that sleep is dependent on whether or not you have consumed a caffeinated drink within five hours before bed.

B) 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* using dagitty (this is a nice tool: http://dagitty.net/dags.html)

```
DAG <- dagitty("dag{S <- P -> A -> S; A -> C -> S}")
coordinates(DAG) <- list(x= c(P = -1, A = 0, S =0 , C = 1),
                         y = c(P = 1, A = 0, S = 2, C = 1))
drawdag(DAG)
```


$$\begin{tikzcd}
 & A \arrow[dd] \arrow[dr] & \\
P \arrow[ur] \arrow[dr] & & C \arrow[dl] \\
 & S &
\end{tikzcd}$$

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

The DAG consists of one pipe (A -> C -> S) and one fork (A <- P -> S)

D) 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()`.

To avoid backdoor paths we will condition on P (party) in the model, because P affects both the outcome S (sleep) and the predictor A (alcohol).

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

```
## { P }
```

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

When inspecting the DAG we can see that the only two variables that are not directly connected are P and C. P affects C through A, which means that when we know A, P gives no additional knowledge about C. We test thus using the function `impliedConditionalIndependencies()`.

```
impliedConditionalIndependencies(DAG)
```

```
## C _||_ P | A
```

Caffeine consumption is independent of P, when we know A: whether or not a given person has consumed caffeine within five hours of going to bed, is independent of whether or not they attended a party, when we know the amount of alcohol they consumed.

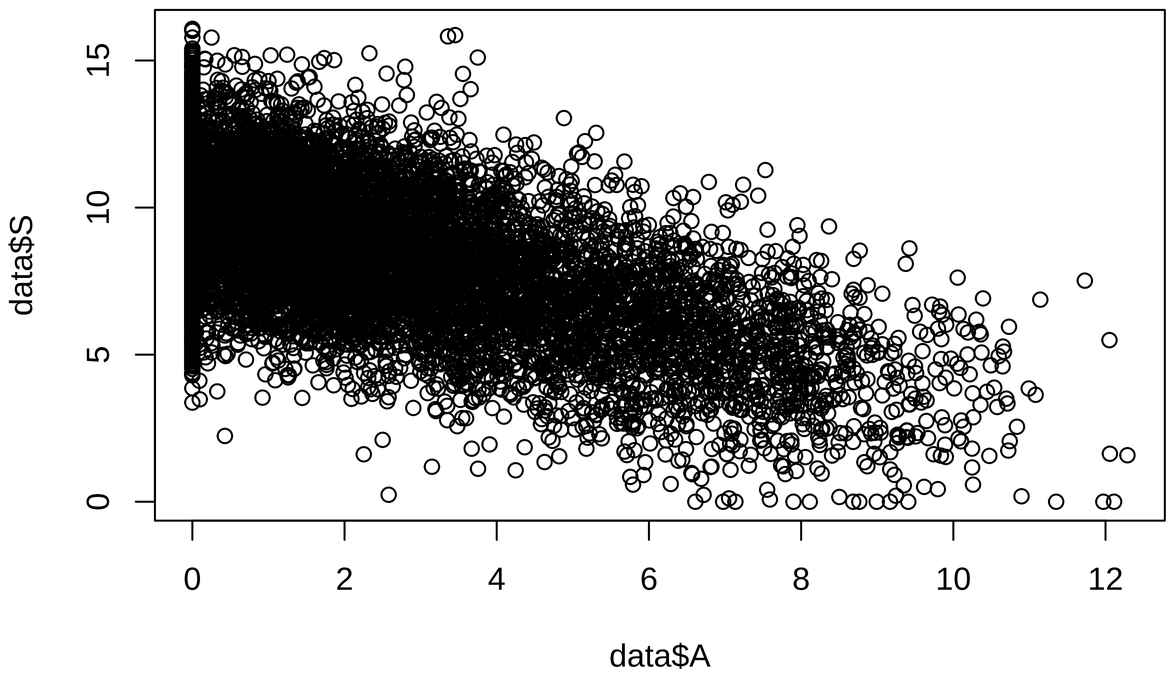
Task 2: The data (SM)

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.

```r
n <- 10000 # Number pf observations
set.seed(4)

b_PA <- 5 # The effect of party on alcohol
b_AC <- 0.05 # The effect of alcohol on caffeine
b_AS <- -0.5 # The effect of alcohol on sleep quality
b_PS <- -1 # The effect of party on sleep quality
b_CS <- -0.5 # The effect of caffeine on sleep quality

P <- rbinom(n, 1, 0.2) # Probability of going to a party any given day is 20 %
A <- rnorm(n, mean = 1+b_PA*P,sd = 2) # Simulating alcohol consumption dependent
                                      # on going to a party
A <- pmax(A, 0) # Applying lower bound of 0.
C <- rbinom(n, size = 1, prob = 0.1+b_AC*A) # Simulating caffeine dependent on
                                            # alcohol
S <- rnorm(n, 10+b_AS*A+b_PS*P+b_CS*C, 2) # Simulating sleep based on all of the
# direct paths pointing to sleep in the DAG + setting intercept to 10 +
# standard deviation of 1
S <- pmax(S, 0) # Applying lower bound of 0 since your quality of sleep cannot
# be negative (at least not on this scale)

data <- data.frame(P = P, A = A, C = C, S = S)

precis(data)
```

```
##       mean        sd      5.5%     94.5%     histogram
## P 0.197800 0.3983604 0.000000  1.000000
## A 2.304233 2.4284741 0.000000  7.156296
## C 0.212800 0.4093078 0.000000  1.000000
## S 8.564488 2.5519032 4.275193 12.397002
```

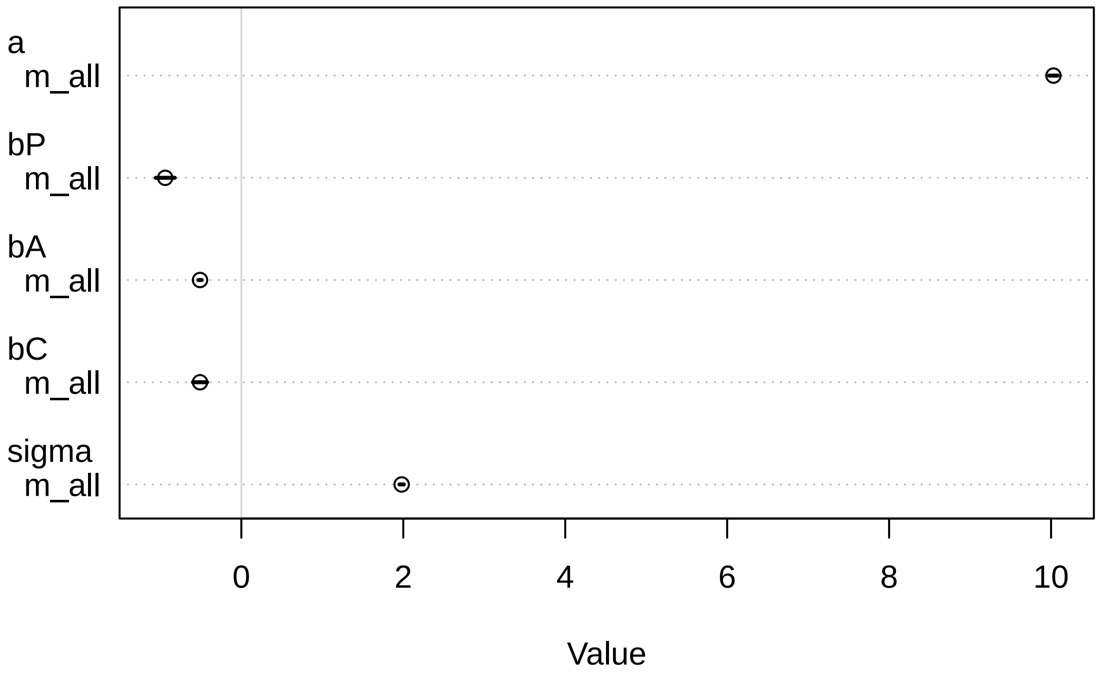
```r
plot(data$A, data$S)
```



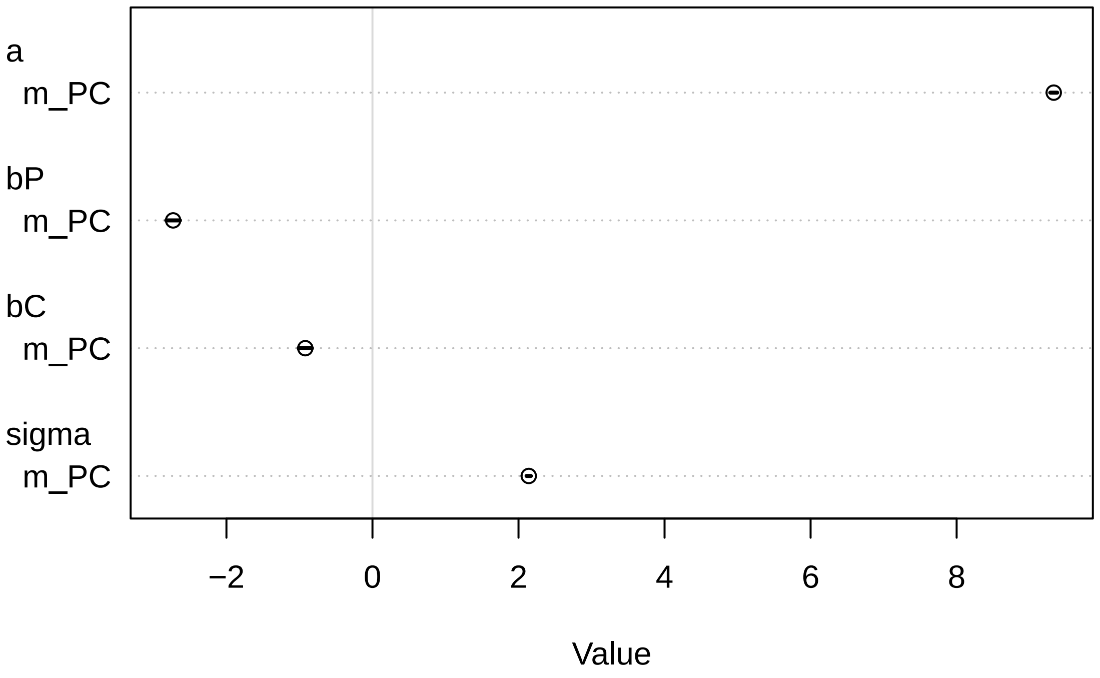
Task 3: Statistics (MST)

Run multiple linear regressions using the simulated data 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.

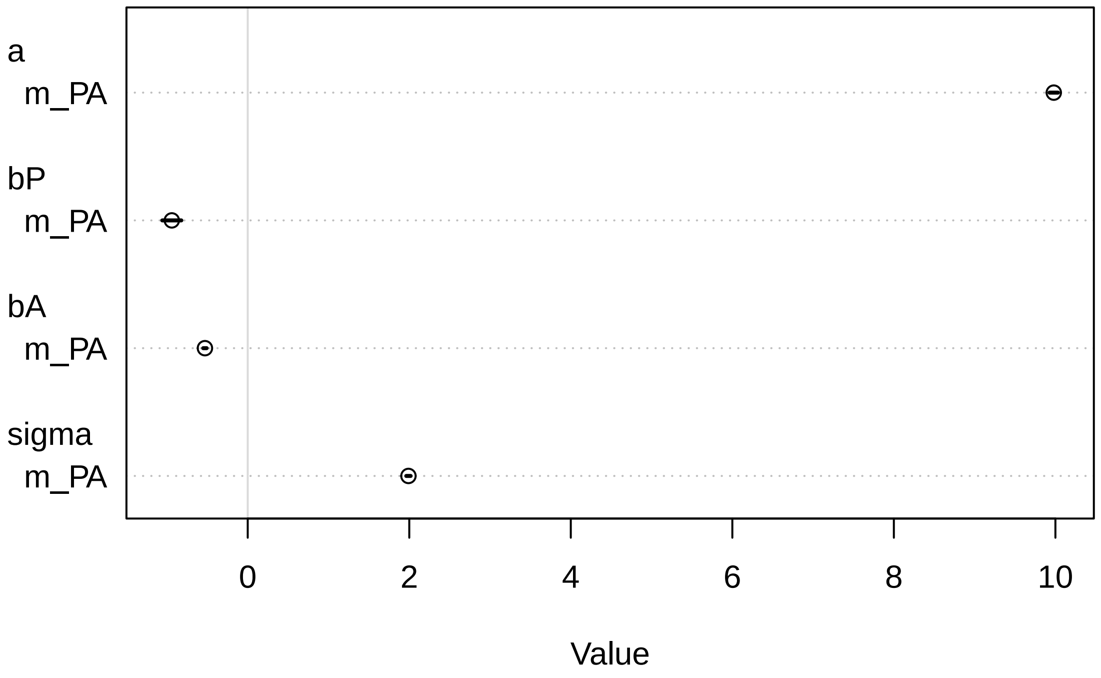
```
m_all <- quap(
    alist(
        S ~ dnorm(mu, sigma),
        mu <- a + bP * P + bA * A + bC*C,
        a ~ dnorm(10, 0.1),
        bP ~ dnorm(-1, 0.1), #believed effect of P on S
        bA ~ dnorm(-0.5, 0.1), #believed effect of A on S
        bC ~ dnorm(-0.5, 0.1), #believed effect of C on S
        sigma ~ dexp(1)),
    data = data)


m_PC <- quap(
    alist(
        S ~ dnorm(mu, sigma),
        mu <- a + bP * P + bC * C,
        a ~ dnorm(10, 0.1),
        bP ~ dnorm(-1, 0.1), #believed effect of P on S
        bC ~ dnorm(-0.5, 0.1), #believed effect of c on S
        sigma ~ dexp(1)),
    data = data)

m_PA <- quap(
    alist(
        S ~ dnorm(mu, sigma),
        mu <- a + bP * P + bA * A,
        a ~ dnorm(10, 0.1),
        bP ~ dnorm(-1, 0.1), #believed effect of P on S
        bA ~ dnorm(-0.5, 0.1), #believed effect of A on S
        sigma ~ dexp(1)),
    data = data)


m_CA <- quap(
    alist(
        S ~ dnorm(mu, sigma),
        mu <- a + bC * C + bA * A,
        a ~ dnorm(10, 0.1),
        bC ~ dnorm(-0.5, 0.1), #believed effect of P on S
        bA ~ dnorm(-0.5, 0.1), #believed effect of A on S
        sigma ~ dexp(1)),
    data = data)


plot(coeftab(m_all))
```



```
plot(coeftab(m_PC))
```



```
plot(coeftab(m_PA))
```



```
precis(m_all)
```

```
##             mean          sd       5.5%      94.5%
## a     10.0323213 0.02706930  9.9890593 10.0755833
## bP    -0.9422784 0.06012647 -1.0383721 -0.8461847
## bA    -0.5098325 0.01106201 -0.5275118 -0.4921533
## bC    -0.5072554 0.04492240 -0.5790501 -0.4354607
## sigma  1.9792032 0.01399320  1.9568394  2.0015670
```

```
precis(m_PC)
```

```
##             mean          sd       5.5%      94.5%
## a      9.3308176 0.02416652  9.2921948  9.3694404
## bP    -2.7257524 0.04814199 -2.8026926 -2.6488122
## bC    -0.9229648 0.04685222 -0.9978437 -0.8480859
## sigma  2.1445167 0.01525182  2.1201413  2.1688920
```

```
precis(m_PA)
```

```
##             mean          sd      5.5%      94.5%
## a      9.9830984 0.02684054  9.940202 10.0259948
## bP    -0.9372168 0.06031789 -1.033616 -0.8408172
## bA    -0.5349113 0.01088024 -0.552300 -0.5175226
## sigma  1.9893092 0.01406465  1.966831  2.0117872
```

```
precis(m_CA)
```

```
##              mean          sd       5.5%      94.5%
## a      10.1066763 0.026824125 10.0638062 10.1495464
## bC     -0.5017368 0.045175719 -0.5739363 -0.4295373
## bA     -0.6247882 0.008334767 -0.6381088 -0.6114677
## sigma   1.9933350 0.014093062  1.9708116  2.0158584
```

We run the different models to check if our simulated data matches our DAG. We observe that when we include both A (our predictor) and P (closing the back door) our data is compatible with our DAG.

Task 4: Messing it up (TI)

A) Change your model from *Task 3* to have an open back door path and see if you would make a false inference.

```
m_A <- quap(
    alist(
        S ~ dnorm(mu, sigma),
        mu <- a + bA * A,
        a ~ dnorm(10, 0.1),
        bA ~ dnorm(-0.05, 0.1), #believed effect of A on S
        sigma ~ dexp(1)),
    data = data)

precis(m_A)
```

```
##              mean          sd       5.5%      94.5%
## a      10.0510532 0.026579062 10.0085747 10.0935317
## bA     -0.6460344 0.008083117 -0.6589528 -0.6331161
## sigma   2.0031539 0.014162772  1.9805191  2.0257888
```

When we do not include P, we have an open back door, and the model observes a greater effect of A on S than the true effect which is -0.5. This results in a false inference.

B) Simulate some data that doesn't fit the DAG and fit the model from *Task 3* above to the new simulated data. You should now get results that doesn't fit the assumptions of your DAG.

We add an extra variable darkness (D) that influences S and P. This does not align with the DAG.

```
#Simulating data that doesn't fit the DAG
b_DS <- 1 #adding a new edge from D to S
D <- rbinom(n, size = 1, prob = 0.5) #adding new variable D (darkness)

b_DP <- 0.5 #The effect of darkness on party

P_2 <- rbinom(n, 1, 0.2 + b_DP*D) # Probability of going to a party any given
                                  # day is 20 %
A_2 <- rnorm(n, mean = 1+b_PA*P,sd = 2) # Simulating alcohol dependent on going
                                        # to a party
A_2 <- pmax(A, 0) # Applying lower bound of 0. You cannot drink minus amount of
                  # drinks
```

```
C_2 <- rbinom(n, size = 1, prob = 0.1+b_AC*A) # Simulating caffeine dependent on
                                              # alcohol

S_2 <- rnorm(n, 10+b_AS*A_2+b_PS*P_2+b_CS*C+b_DS*D, 2)
S_2 <- pmax(S, 0)

data_2 <- data.frame(P = P_2, A = A_2, C = C_2, D = D, S = S_2)
precis(data_2)
```

```
##       mean        sd     5.5%     94.5%     histogram
## P 0.445700 0.4970676 0.000000  1.000000
## A 2.304233 2.4284741 0.000000  7.156296
## C 0.217300 0.4124290 0.000000  1.000000
## D 0.499700 0.5000249 0.000000  1.000000
## S 8.564488 2.5519032 4.275193 12.397002
```

```
m_PA_2 <- quap(
    alist(
        S ~ dnorm(mu, sigma),
        mu <- a + bP * P + bA * A,
        a ~ dnorm(10, 1),
        bP ~ dnorm(-1, 0.1), #believed effect of P on S
        bA ~ dnorm(-0.5, 0.1), #believed effect of A on S
        sigma ~ dexp(1)
    ),
    data = data_2
)

precis(m_PA_2)
```

```
##             mean          sd       5.5%       94.5%
## a     10.1196886 0.032013863 10.0685243 10.17085292
## bP    -0.1331555 0.037441550 -0.1929943 -0.07331669
## bA    -0.6494822 0.008226293 -0.6626294 -0.63633495
## sigma  2.0042983 0.014184864  1.9816281  2.02696845
```

We observe that the effect of P on S and the effect of A on S does not align with the simulated effects, since we now have an extra variable D, that we do not account for when using the model from task 3.