Causal Inference and SEM: Lab 1 DAGs

Edited by Noémi Schuurman, original materials of Oisín Ryan

In these exercises you will get some hands-on experience with Directed Acyclic Graphs (DAGs) and Structural Causal Models (SCMs).

1. Data and Packages

Throughout this practical you will make use of various R-packages. If you haven't already, install the following packages.

```
# packages for working with DAGs
install.packages("qgraph")
install.packages("dagitty")
install.packages("ggdag")

# tidyverse - i assume you already have this
library(tidyverse)

# packages for testing independence of variables
install.packages("CondIndTests")
install.packages("dHSIC")
install.packages("ppcor")

# the installation of the packages required for the PC algorithm is a little more involved. You need to download some dependencies using `Bioconductor`, an alternative to CRAN used for bioinformatics packages. The code below should get you all set up and ready to go. If it doesn't work, check that you have R version 4.0.0 or higher, and that you have installed and configured [Rtools](https://cran.r-project.org/bin/windows/Rtools/).

# The graph, RBGL & Rgraphviz packages are needed for pcalg
# but are only available on Bioconductor but not CRAN
if (!requireNamespace("BiocManager", quietly = TRUE)){
    install.packages("BiocManager")
}

BiocManager::install("graph")
BiocManager::install("RBGL")
BiocManager::install("Rgraphviz")

install.packages("pcalg")
# Note - you likely need R 4.0.0 or higher for this
```

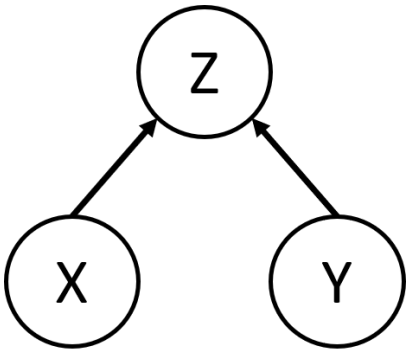
You will also make use of a simulated dataset `ex5_data.RDS` in exercise 2.5 below.

2. Graphs and Adjacency Matrices

There are many different ways to draw DAGs in R. Here we introduce you to two.

For the first approach, we need to introduce some general terminology that is used for graphical models. In the graphical modeling literature, the edges in a graph are represented by an *adjacency matrix*, typically denoted A. This is simply a $p \times p$ matrix of 0's and 1's indicating which of the p variables share an edge. $A_{ij} = 1$ indicates that there is an arrow from variable i to variable j, that is $X_i \rightarrow X_j$

For example, the three-variable “collider” DAG

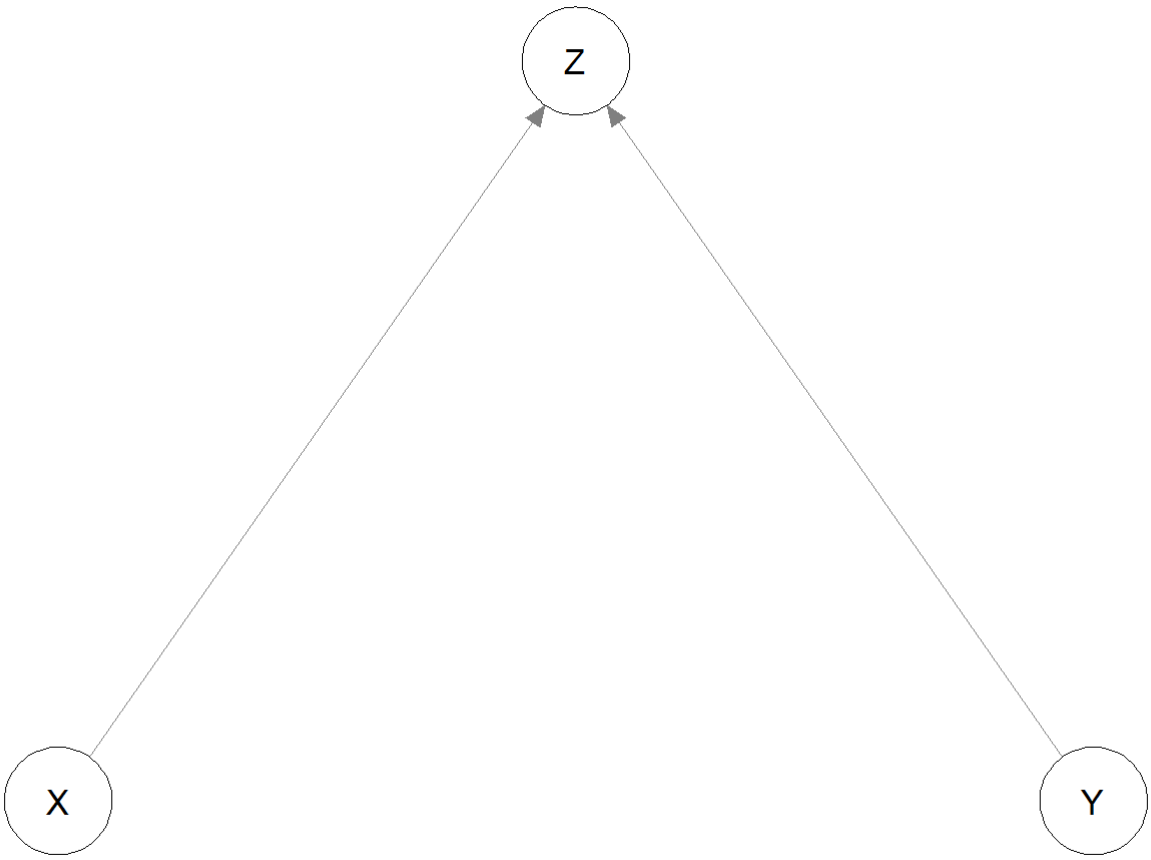


has adjacency matrix (with rows and columns in the order X, Y, Z)

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

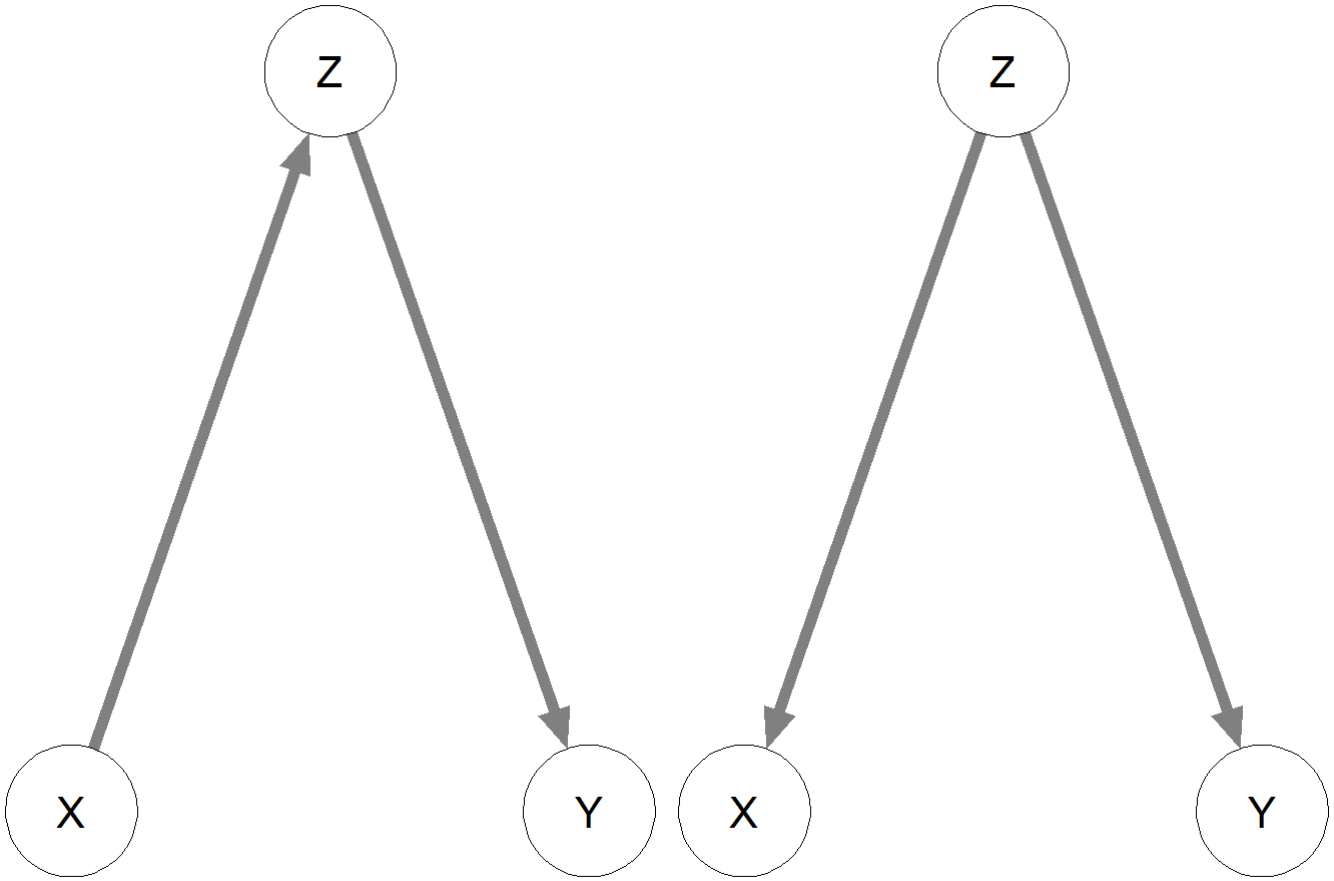
We can plot this graph in R using the package `qgraph`. We can understand `qgraph` as a general purpose package for plotting graphical models: It is not specially designed for use with DAGs.

```
library(qgraph)

varnames <- c("X","Y","Z")
Adj <- matrix(c(0,0,1,
                0,0,1,
                0,0,0), 3,3, byrow = TRUE,
              dimnames = list(varnames,varnames))

qgraph(Adj,
       labels = c("X","Y","Z"), # not necessary if Adj has dimnames
       #you can provide a custom layout by giving the x-y co-ordinates of each node
       layout = rbind(c(-1,-1),
                      c(1,-1),
                      c(0,1)))
```



▶ To be sure you've got to grips with adjacency matrices, also re-create a three-variable chain DAG, and a fork DAG.

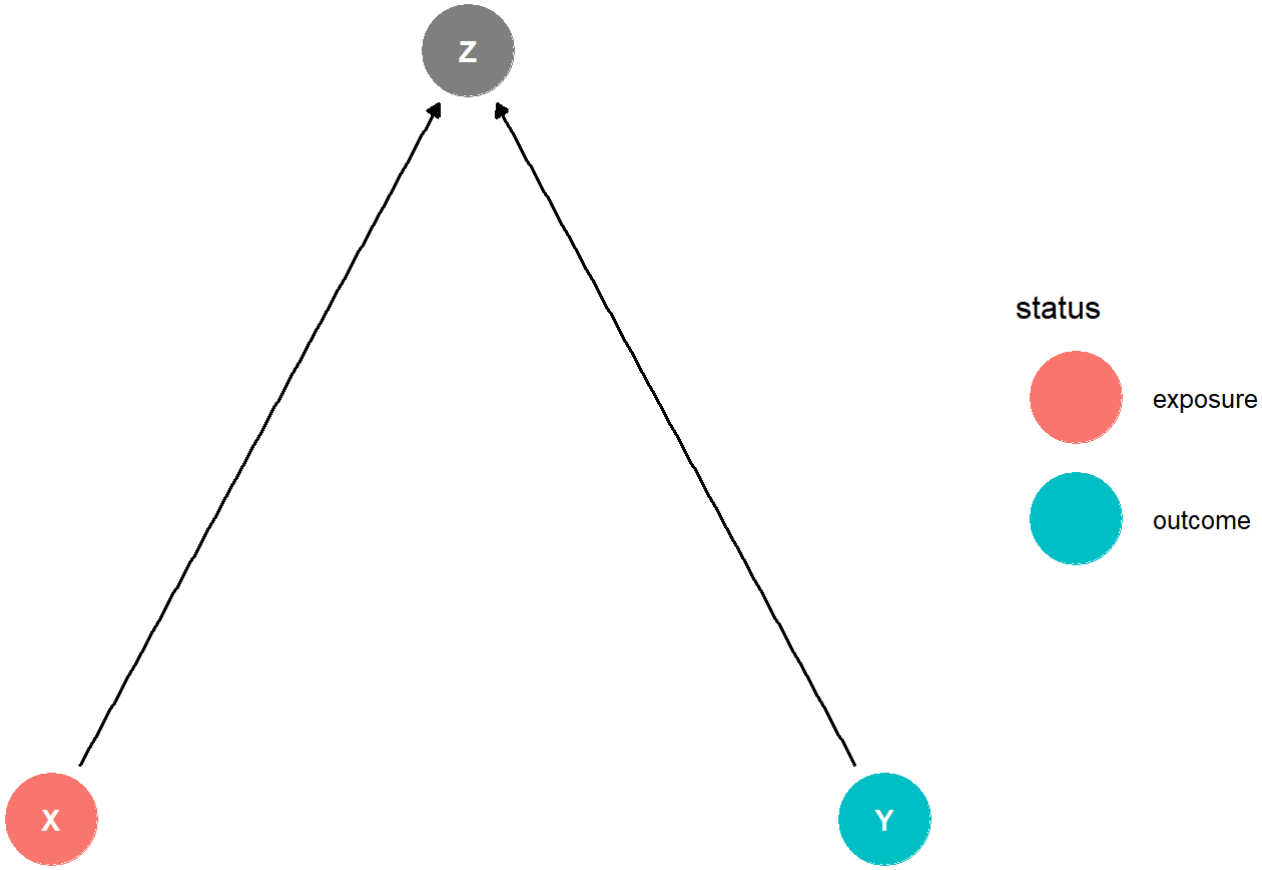
```
varnames <- c("X","Y","Z")
Adj_chain <- matrix(c(0,0,1,
                      0,0,0,
                      0,1,0), 3,3, byrow = TRUE,
              dimnames = list(varnames,varnames))

Adj_fork <- matrix(c(0,0,0,
                     0,0,0,
                     1,1,0), 3,3, byrow = TRUE,
              dimnames = list(varnames,varnames))

# Plotting optional parameters
laymat <- rbind(c(-1,-1),
                c(1,-1),
                c(0,1))
# change qgraph node, edge and arrow-head size
vsize =15; esize = 10; asize = 10

par(mfrow = c(1,2))
qgraph(Adj_chain,
       layout = laymat,
       vsize =vsize, esize = esize, asize = asize)
qgraph(Adj_fork,
       layout = laymat,
       vsize =vsize, esize = esize, asize = asize)
```



A second helpful way of plotting DAGs is using the `ggdag` package. `ggdag` provides an interface between `ggplot` and the `dagitty` package, a package designed to make DAG-based analysis easier. This means you can do things like derive adjustment sets and list paths in a DAG once you've defined it as an object. If you don't like to use `ggplot`, you can also access dagitty in a GUI form on the dagitty website (http://www.dagitty.net/).

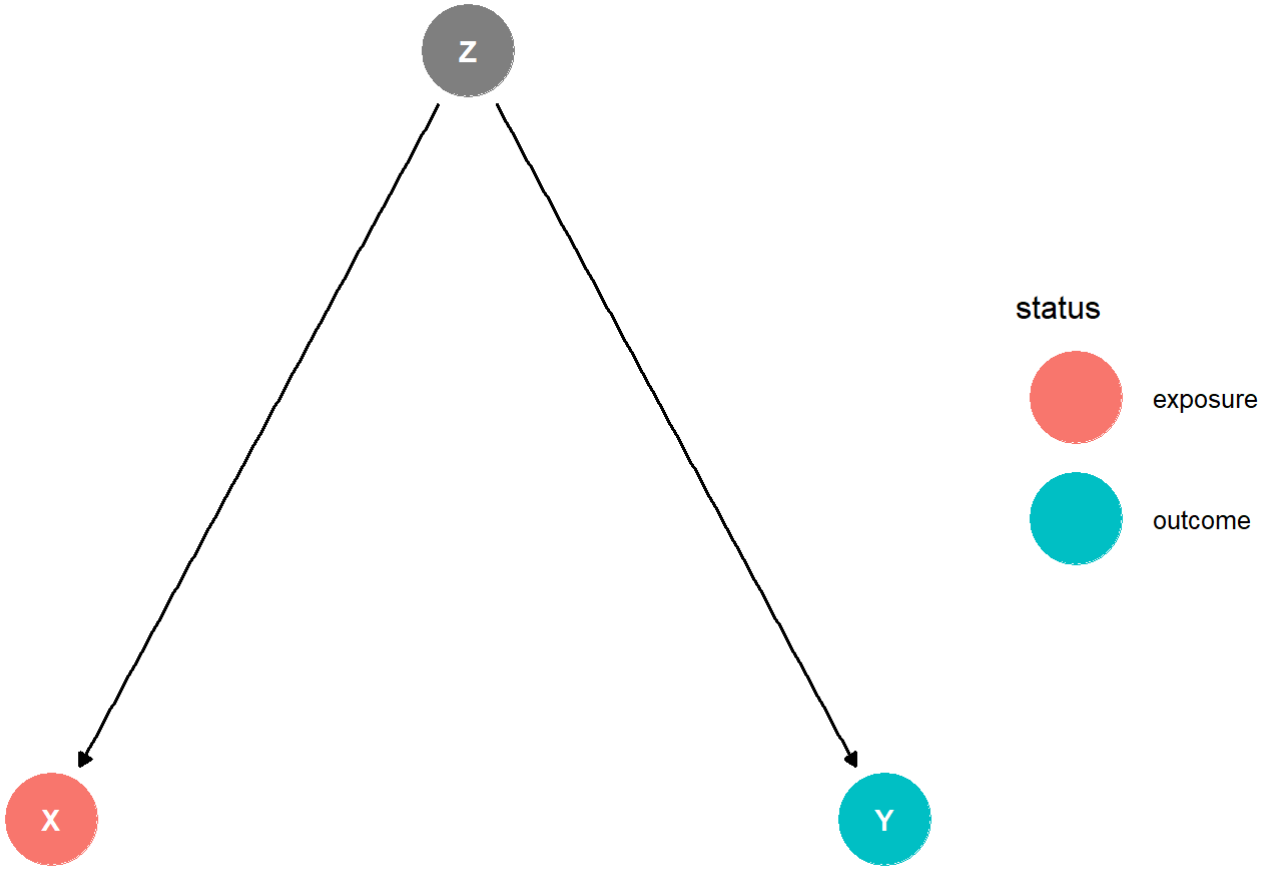
We can create the collider DAG using the syntax below

```
coldag <- dagify(
  Z ~ X + Y,
   exposure = "X", # the "cause" variable you are interested in
   outcome = "Y", # the "effect" variable you are interested in
  # optional: give co-ordinates of the variables in the plot
  coords = list(x = c(X = -1, Y = 1, Z = 0),
                y = c(X = 0, Y = 0, Z = 1))
)
ggdag_status(coldag) + theme_dag()
```

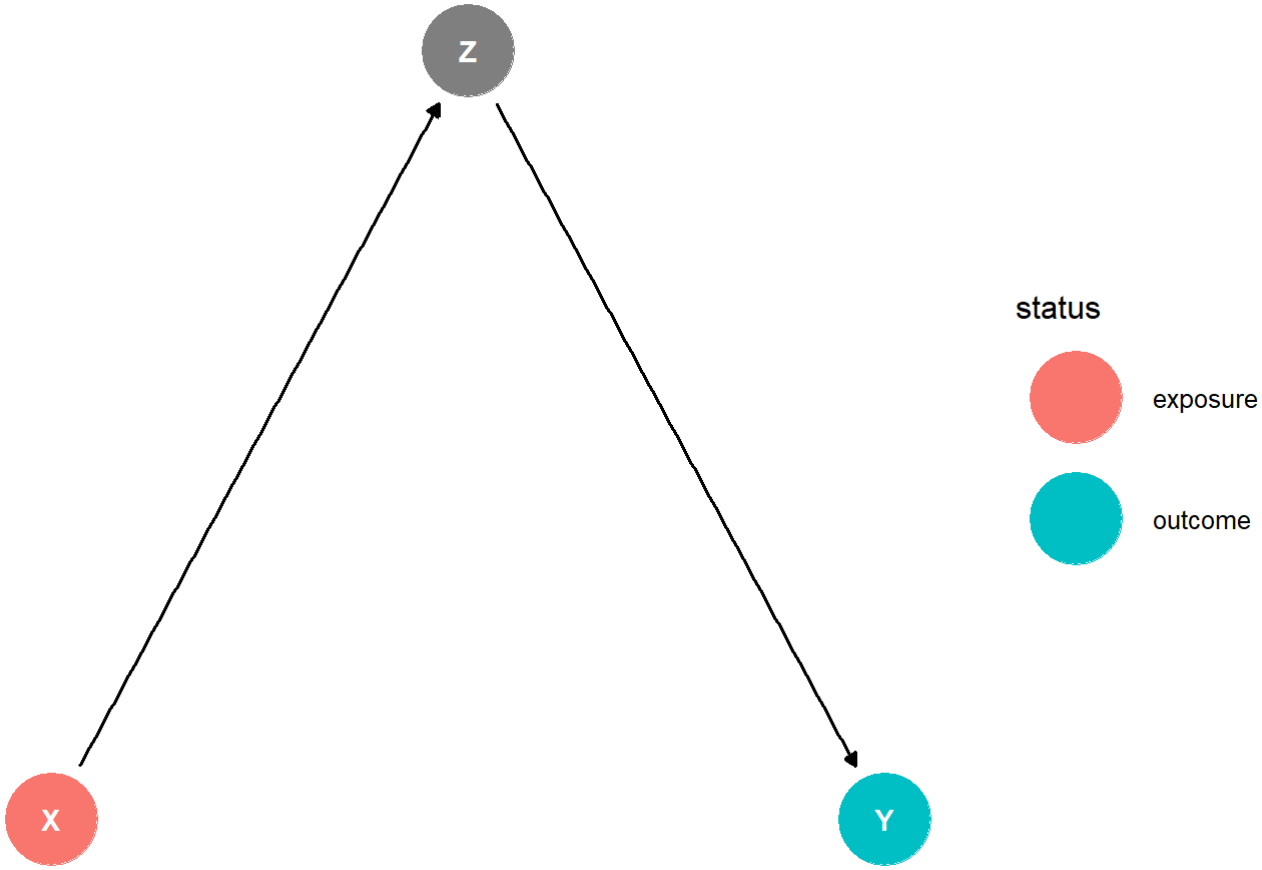


▶ Re-create the chain and fork DAGs in the lecture using the `ggdag` package.

```
ccdag <- dagify(
  Y ~ Z,
  X ~ Z,
   exposure = "X", # the "cause" variable you are interested in
   outcome = "Y", # the "effect" variable you are interested in
  # optional: give co-ordinates of the variables in the plot
  coords = list(x = c(X = -1, Y = 1, Z = 0),
                y = c(X = 0, Y = 0, Z = 1))
)
ggdag_status(ccdag) + theme_dag()
```




```
meddag <- dagify(
  Z ~ X,
  Y ~ Z,
   exposure = "X", # the "cause" variable you are interested in
   outcome = "Y", # the "effect" variable you are interested in
  # optional: give co-ordinates of the variables in the plot
  coords = list(x = c(X = -1, Y = 1, Z = 0),
                y = c(X = 0, Y = 0, Z = 1))
)
ggdag_status(meddag) + theme_dag()
```

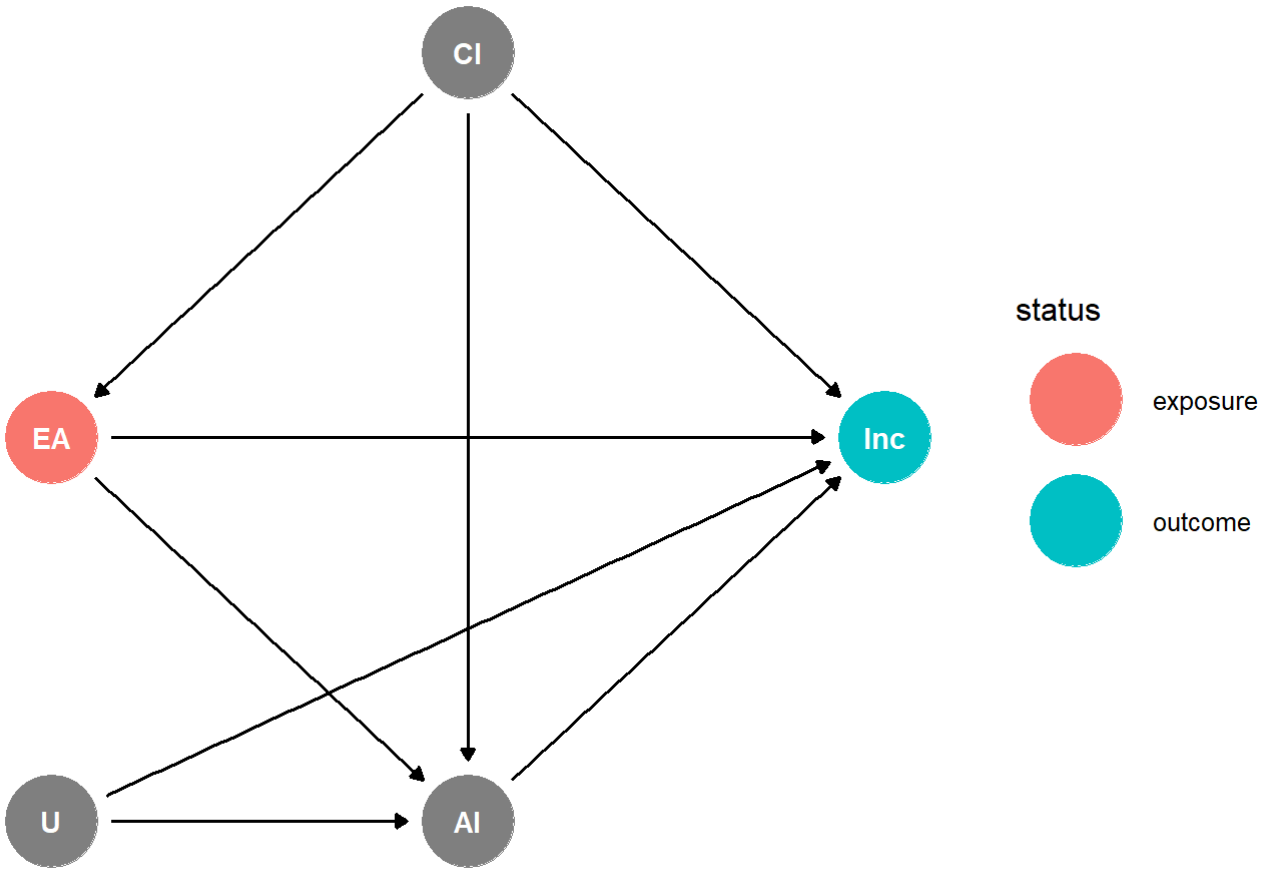


3. d-seperation and DAGs

Imagine we are a team of researchers who are interested in estimating the causal effect of *Educational Attainment* on adult *Income* levels. We draw the following DAG, representing our beliefs that

```
- Educational Attainment (EA) is caused by Childhood Intelligence (CI)
- Adult Intelligence (AI) is caused by Childhood Intelligence, Educational Attainment, a
nd a variety of other factors we call U
- Income (Inc) is caused by Childhood and Adult Intelligence, Educational Attainment, an
d the unnamed factor(s) U
- The unnamed U factor(s) do not cause Educational Attainment or Childhood Intelligence
```

```
eddag <- dagify(
  EA ~ CI,
  AI ~ CI + EA + U,
  Inc ~ EA + CI + AI + U ,
  exposure = "EA", # the "cause" variable you are interested in
  outcome = "Inc", # the "effect" variable you are interested in
  # optional: give co-ordinates of the variables in the plot
  coords = list(x = c(EA = -1,CI = 0,AI =0 ,Inc =1 , U = -1),
                y = c(EA = 0,CI =1 ,AI = -1,Inc = 0, U = -1))
)
ggdag_status(eddag) + theme_dag()
```



▶ List all of the paths between *EA* and *Inc* in the DAG. Tip: There are **seven** in total.

EA → Inc
EA → AI → Inc
EA ← CI → Inc
EA ← CI → AI → Inc
EA → AI ← U → Inc
EA ← CI → AI ← U → Inc
EA → AI ← CI → Inc

```
##
##     EA -> Inc
##     EA <- CI -> Inc
##
##     EA <- CI -> AI -> Inc
##     EA <- CI -> AI <- U -> Inc
##
##     EA -> AI -> Inc
##     EA -> AI <- U -> Inc
##     EA -> AI <- CI -> Inc
```

▶ Based on what you know about the true DAG, try to classify these paths into categories. If we want to estimate the causal effect of *EA* on *Inc*, what paths result in non-causal statistical associations and should be blocked? What paths transmit causal associations, and should be left open? What paths are already blocked?

```
## Path that transmits direct causal effects
##     EA -> Inc
##
##     Path that transmits indirect causal effects and should not be blocked
##     EA -> AI -> Inc
##
##     Paths that transmit non-causal associations and need to be blocked
##     EA <- CI -> Inc
##     EA <- CI -> AI -> Inc
##
##     Paths that are blocked but would transmit non-causal associations
##     if the collider AI was conditioned on
##     EA <- CI -> AI <- U -> Inc
##     EA -> AI <- U -> Inc
##     EA -> AI <- CI -> Inc
```

▶ If we want to estimate the causal effect of *EA* on *Inc*, what is the valid adjustment set? That is, what variable(s) should the researchers condition on? Try to derive this using your d-seperation rules. Check your answer by using `adjustmentSets(eddag)`.

```
## Childhood Intelligence (CI) is the valid adjustment set.
##     It is necessary and sufficient to condition on Childhood Intelligence when estima
ting the effect of EA on Inc, because there are two open backdoor paths through CI. Cond
itioning on CI blocks those paths.
##     We should avoid conditioning on AI because it is a collider, which would open new
non-causal paths.
##     We do not need to condition on U
```

```
adjustmentSets(eddag)
```

```
## { CI }
```

▶ Imagine our team of researchers is designing their data collection for this study, and they're trying to decide what variables they need to measure. What is your advice?

cause, DV, confounder

```
## Based on our analysis of the DAG above, we would only need to collect the variables E
A, Inc and Childhood Intelligence. We do not need to measure U or Adult Intelligence, th
ough of course it's generally good practice to collect other relevant information when p
ossible.
```

4. Predictive vs Causal Analysis

Load the simulated dataset `mdata.RDS`

```
mdata <- readRDS("mdata.RDS")
```

The data is simulated based on a hypothetical epidemiological study. Researchers are interested in whether using mosquito nets decreases an individual's risk of contracting malaria. They have collected data from 1,500 households in an unnamed country. In this country there is a voluntary program where high-income individuals can obtain free mosquito nets. The researchers surveyed households and recorded whether or not they used mosquito nets `nets`, the average household income `Income`, the overall `health` status of the household, and assessed the risk of someone in the household contracting malaria `mal`. `nets` is a binary variable, (1 = uses nets, 0 = doesn't use nets) but all other variables are continuous and centered approximately around zero: zero indicates a normal income/health/risk status, with negative values lower than normal and higher values higher than normal.

(average)

4.1 Predictive Analysis

Let's try to analyze this dataset without using any DAG or potential-outcome ideas. Let's imagine for a moment that we are only interested in **predicting** the malaria variable, and we are open to using all variable in the dataset.

▶ Build the best prediction model for `mal` that you can, using every variable in the dataset. The only restriction is that `nets` should be included as predictor, but otherwise you're free to approach this however you like. Use your favourite prediction tool.

▶ Once you've done this, try to interpret the predictive relationship in the ensuing model between `nets` and `mal` as an estimate of the causal effect. How strong is the causal effect, according to this procedure?

```
## I'll just use linear regression for this, with no fancy extras
```

```
mod1 <- lm(mal ~ net + Income + health, data = mdata)
summary(mod1)
```

```
##
## Call:
## lm(formula = mal ~ net + Income + health, data = mdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.29029 -0.24001 -0.00301  0.24026  1.28304
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -0.004812   0.004905   -0.981    0.327
## net         -0.504389   0.008946  -56.379   <2e-16 ***
## Income      -0.748618   0.004485 -166.920   <2e-16 ***
## health       0.248199   0.002029  122.298   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3527 on 14996 degrees of freedom
## Multiple R-squared:  0.8859, Adjusted R-squared:  0.8859
## F-statistic: 3.881e+04 on 3 and 14996 DF,  p-value: < 2.2e-16
```

```
## If we were to interpret the regression coefficients in a vague *causal* way we might infer that using nets decreases malaria risk by about .5 points.
##  If we choose to extend this interpretation to the other variables in the model, we would infer that Income has a stronger negative effect on malaria risk, and that health has a weakly positive causal effect on malaria risk.
```

4.2 Causal Analysis

Of course, these conclusions shouldn't be taken at face value. The researchers are interested in the **causal effect** of net usage on malaria risk. But here, we have only analyzed the **predictive relationship** between nets and malaria risk, conditional on income and health: that is, what information does net usage add to our prediction of malaria risk above and beyond the predictive information supplied by income and health.

Now let's approach this analysis using a causal perspective. First, let's begin by representing our beliefs about the causal system in a DAG. We believe that:

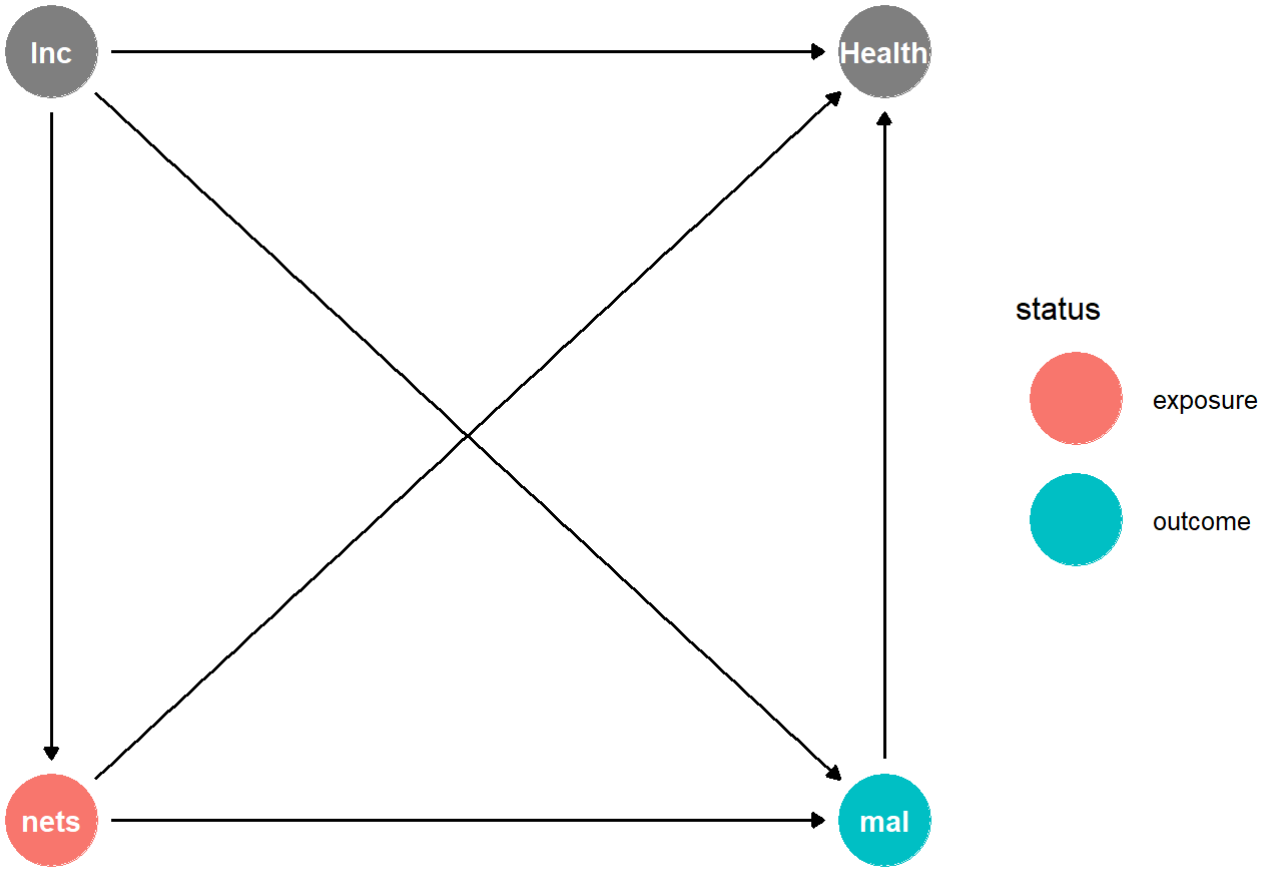
```
- net usage causes malaria risk (in that it decreases malaria risk)
- Income causes both net usage (higher incomes have more access to nets), and malaria ri
sk
- Overall health status is caused by both Income and malaria risk
```

net → mal
Income → net + mal + health
mal → health

▶ Use `dagify()` to draw the DAG which corresponds with these beliefs about the causal system.

mal = net + Income
net = Income
health = Inc + mal

```
maldag <- dagify(
  mal ~ nets + Inc,
  nets ~ Inc,
  Health ~ nets + Inc + mal,
  exposure = "nets", # the "cause" variable you are interested in
  outcome = "mal", # the "effect" variable you are interested in
  # optional: give co-ordinates of the variables in the plot
  coords = list(x = c(nets = -1, mal = 1, Inc = -1, Health= 1),
                y = c(nets = -1, mal = -1, Inc = 1, Health= 1))
)
ggdag_status(maldag) + theme_dag()
```



▶ Using d-seperation rules, can you read off what variable(s) you should control for in order to estimate the causal effect of nets on mal? What is the valid adjustment set?

```
## There is a backdoor path through Income, so we should block that by conditioning on I
ncome
##      Health is a collider variable, so we should not condition on it
##      The valid adjustment set is Income
```

▶ Check your answer to the previous question by using the function `adjustmentSets()`

```
adjustmentSets(maldag)
```

```
## { Inc }
```

▶ Use what you learned in the previous questions to estimate the causal relationship of `nets` on `mal`. For now, let's make the simplifying assumptions that all variables share linear relationships. This means that to get insights into the causal relationship, you should fit a linear regression model where `nets` as well as the variables in the valid adjustment set are predictors, and `mal` is the outcome.

```
cemod <- lm(mal ~ net + Income, data = mdata)
summary(cemod)
```

```
cemod <- lm(mal ~ net + Income, data = mdata)
summary(cemod)
```

```
##
## Call:
## lm(formula = mal ~ net + Income, data = mdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.06251 -0.33863 -0.00107  0.34271  2.16639
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0001781  0.0069316  -0.026    0.979
## net         -1.0046934  0.0112443 -89.352   <2e-16 ***
## Income      -0.4959763  0.0056258 -88.160   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4985 on 14997 degrees of freedom
## Multiple R-squared:  0.7721, Adjusted R-squared:  0.7721
## F-statistic: 2.54e+04 on 2 and 14997 DF,  p-value: < 2.2e-16
```

▶ Compare your estimate of the causal parameter with your "causal interpretation" of the predictive relationship. What do you notice?

```
## The regression coefficient of net on mal in this model (conditional on Income) is -1
##  This effect is twice as large as the predictive relationship I estimated, that is, when I conditioned on Income and Health.
##  Conditioning on Income only allows me to estimate the causal relationship between nets and malaria risk. Conditioning on Income AND health, however, yields a biased estimate. The bias comes from conditioning on the collider Health.
```

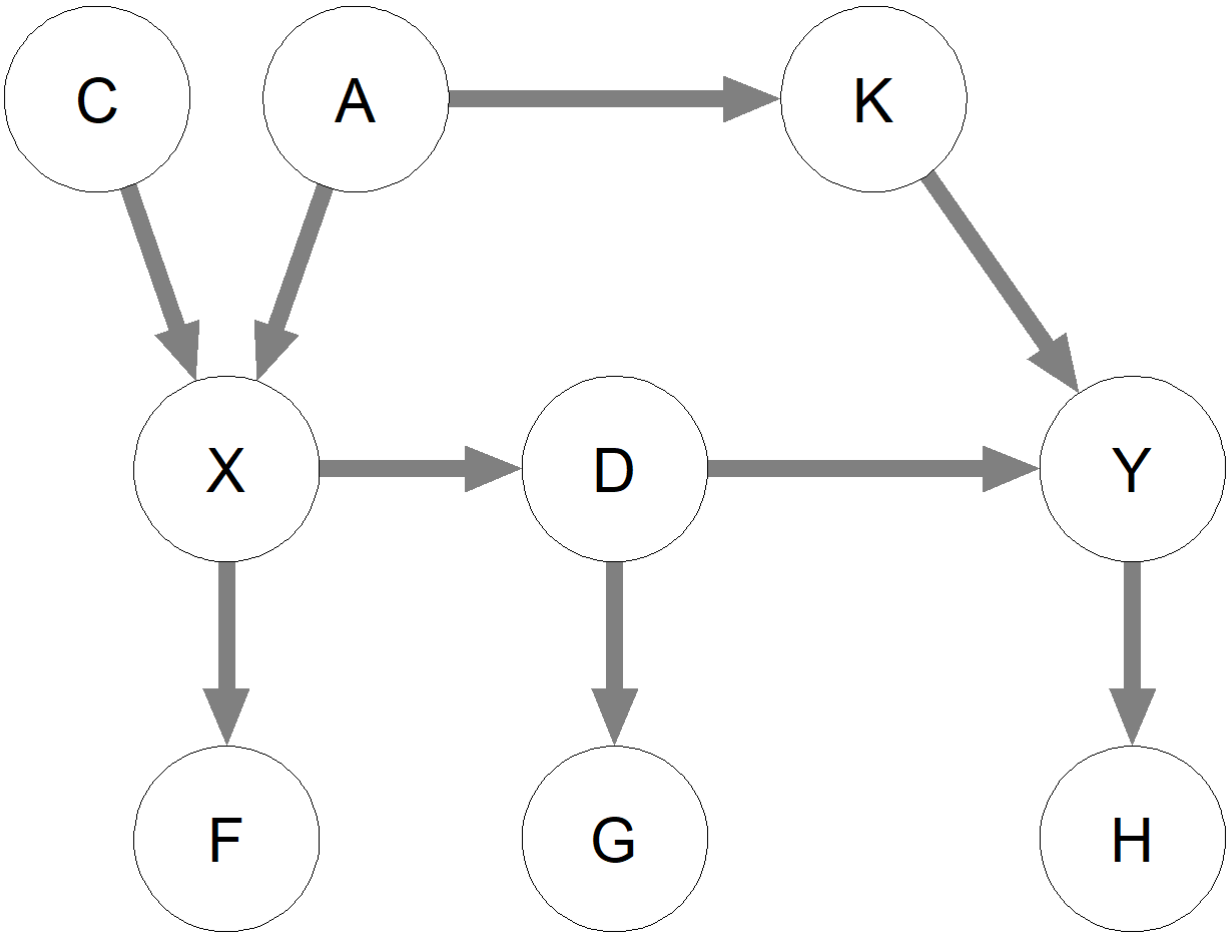
5. Valid Adjustment Sets

In this exercise we will provide you with some simulated data. First, load this data

```
data <- readRDS(file = "ex5_data.RDS")
```

These data have been generated from an SCM with the following DAG structure:

```
Adj <- rbind(c(0,0,0,1,0,0,0,0,0),
             c(0,0,1,1,0,0,0,0,0),
             c(0,0,0,0,0,0,0,1,0),
             c(0,0,0,0,1,1,0,0,0),
             c(0,0,0,0,0,0,0,0,0),
             c(0,0,0,0,0,0,1,1,0),
             c(0,0,0,0,0,0,0,0,0),
             c(0,0,0,0,0,0,0,0,1),
             c(0,0,0,0,0,0,0,0,0))

names <- c("C", "A", "K", "X", "F", "D", "G", "Y", "H")
dimnames(Adj) = list(names,names)



# Make a nice layout (x-y coords)
laymat <- matrix(
    c(-1,   1,
     -.5,   1,
      .5,   1,
    -.75,   0,
    -.75,  -1,
       0,   0,
       0,  -1,
       1,   0,
       1,  -1),9,2,byrow = T)

vsize =15; esize = 10; asize = 10

qgraph(Adj,
        layout = laymat,
       vsize =vsize, esize = esize, asize = asize)
```



First, let's practice using d-seperation rules to determine which variables are conditionally dependent and independent of one another. For instance, we can read off the DAG the following two independence relations:

- C and A are marginally independent
- X and G are independent conditional on D

We now want to *test* whether that conditional independence holds using our data. If you're willing to assume Gaussian variables and linear relationships, we can do this using the t-tests of slopes provided by normal linear regression. However, a more robust way to do this is by using independence tests that don't require linearity assumptions. These are implemented in special R-packages already

- `dhsic.test` from the package `dHSIC` tests marginal (unconditional) independence
- `CondIndTest` from the package `CondIndTests` tests conditional dependence

Ho = they are independent!

The null-hypothesis in each case is that the two variables are (conditionally) *independent*.

```
library(CondIndTests)
library(dHSIC)

# The null hypothesis is that C and A are marginally independent. p = .83. We fail to re
ject the null hypothesis. So, they are statistically independent
dhsic.test(data[,"C"], data[,"A"])
```

```
## $statistic
## [1] 0.1981519
##
## $crit.value
## [1] 0.5751409
##
## $p.value
## [1] 0.8331668
##
## $time
## GramMat   dHSIC CritVal
##    0.01    0.00    1.30
##
## $bandwidth
## [1] 0.6238528 0.6542447
```

not sig ⇒ not reject H0 ⇒ C & A are independent!

```
# From the DAG we know that X and G are independent given D.
# We test the null hypothesis that they are independent with alpha of .05.
# Resulting p-value = .3. We fail to reject the null hypothesis that they are independen
t.
CondIndTest(data[,"X"], data[,"G"], data[,"D"])
```

```
## $Y1
## $Y1$testStatistic
## [1] 0.07320998
##
## $Y1$criticalValue
## [1] 0.1453813
##
## $Y1$pvalue_individual
## [1] 0.3008275
##
##
## $pvalue
## [1] 0.3008275
```

▶ Write down two **true** and two **false** conditional independence statements about the above DAG, based on d-seperation rules. Test these statements as we did above.

```
# You already have an example of two true statements above
# One incorrect statement would be C and A are independent given X. We know that
# they are in fact d-connected (conditioning on a collider)

CondIndTest(data[,"C"], data[,"A"], data[,"X"])
```

```
## $Y1
## $Y1$testStatistic
## [1] 218.3039
##
## $Y1$criticalValue
## [1] 30.87318
##
## $Y1$pvalue_individual
## [1] 0
##
##
## $pvalue
## [1] 0
```

```
# p < .001, we reject the null hypothesis
```
→ they're dependent conditioning on X.

Take it we want to estimate the causal effect of X on Y. Assume linear relationships and Gaussian noise (Note: if non-linear, the size of the effect may differ for different levels of X.)

▶ Use d-seperation to find 2 different valid adjustment sets.

▶ Assume linear relationships and Gaussian noise. Estimate the causal effect twice, once using each adjustment set. Note: The true causal effect of this simulated dataset is equal to two. Did you (approximately) obtain the correct result?

```
model1 <- lm(Y~X+A, data=as.data.frame(data))
```

```
# There are a few different valid adjustment sets
# The easiest is: A (parent of X)
model1 <- lm(Y~X+A, data=as.data.frame(data))
print(model1)
```

```
##
## Call:
## lm(formula = Y ~ X + A, data = as.data.frame(data))
##
## Coefficients:
## (Intercept)            X            A
##     -0.0774       1.9136       1.9046
```

```
# In general we tend to prefer the smallest possible adjustment set.
# This is called a ``minimally sufficient adjustment set''

# Another valid possibility is: A, K
model2 <- lm(Y~X+A +K, data=as.data.frame(data))
print(model2)
```

```
##
## Call:
## lm(formula = Y ~ X + A + K, data = as.data.frame(data))
##
## Coefficients:
## (Intercept)            X            A            K
##    -0.05662      1.92373     -0.13055      2.00816
```

★ because X is a collider & F is a child of X!!

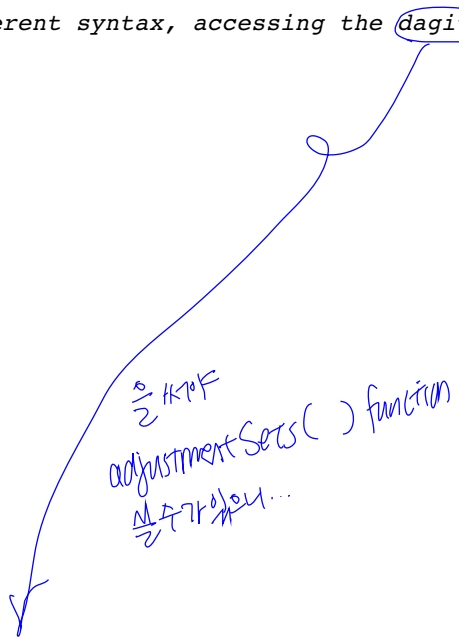
```
# Another is: F, K (conditioning on F opens up a new path, but condition on K blocks it)
model3 <- lm(Y~X+F +K, data=as.data.frame(data))
print(model3)
```

```
##
## Call:
## lm(formula = Y ~ X + F + K, data = as.data.frame(data))
##
## Coefficients:
## (Intercept)            X            F            K
##    -0.05888      2.01163     -0.02374      1.92245
```

```
###All of these models give approximately the same results, which is very close to the t
rue causal effect, which is equal to 2.
```

▶ Input the DAG in this exercise into dagitty. Get all valid adjustment sets and check your answers to the earlier exercise.

```
# Here i use a slightly different syntax, accessing the dagitty function directly
graph <- dagitty("dag{
        X[exposure]
        Y[outcome]
        C -> X
        A -> X
        X -> F
        A -> K
        K -> Y
        X -> D
        D -> G
        D -> Y
        Y -> H
}")

adjustmentSets(graph, type = "all")
```




```
## { A }
## { A, C }
## { A, F }
## { A, C, F }
## { K }
## { A, K }
## { C, K }
## { A, C, K }
## { F, K }
## { A, F, K }
## { C, F, K }
## { A, C, F, K }
```

```
adjustmentSets(graph, type = "minimal")
```

```
## { K }
## { A }
```

Notice that `adjustmentSets` has three different options. `type = "all"` will give all possible adjustment sets, while `type = "minimal"` will give only the *minimal sufficient adjustment sets*. The latter is basically the smallest possible adjustment sets (not containing any extra variables that are not necessary to block a particular path). You can read more about dagitty in this manual (http://www.dagitty.net/history/v1.0/manual.pdf) and of course in the package documentation (https://cran.r-project.org/web/packages/dagitty/dagitty.pdf).

6. Generating Data from an SCM.

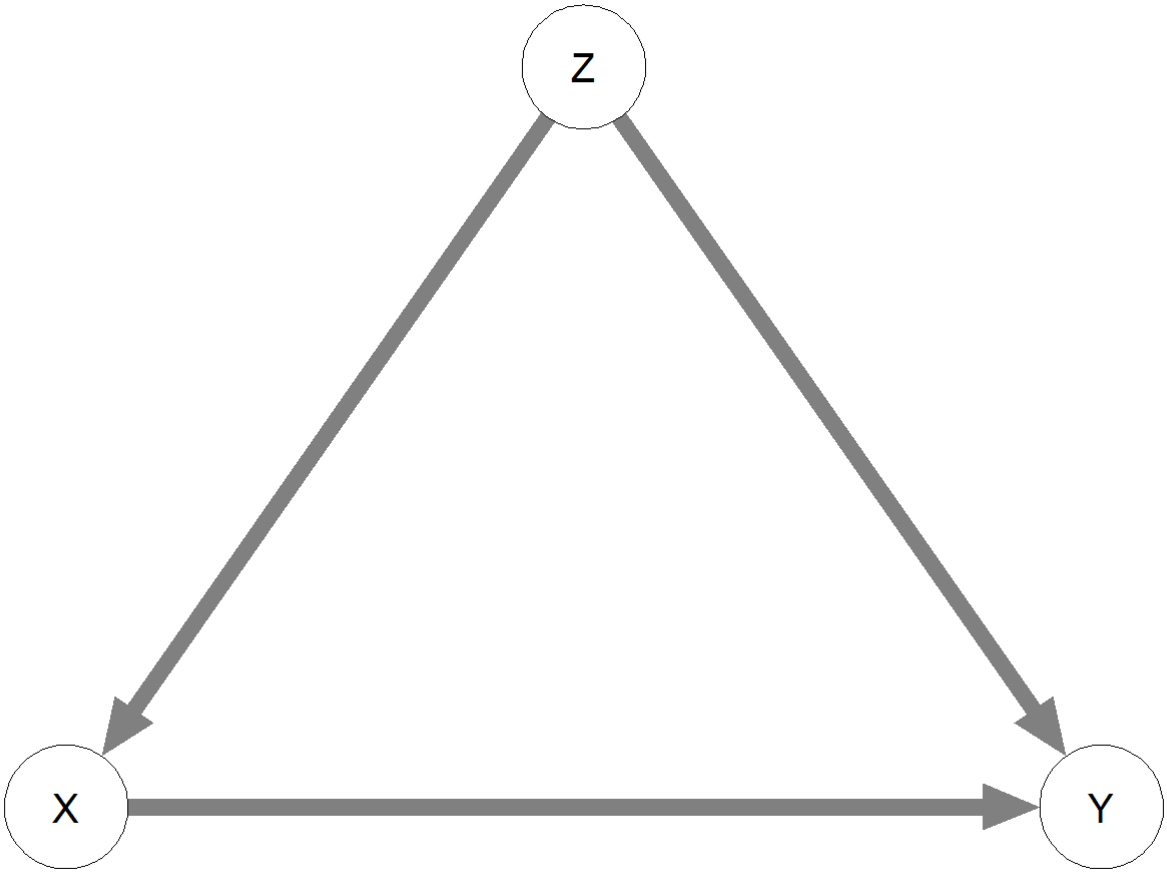
In the lecture we looked at an example SCM with linear relations and normally distributed residuals:

$$\begin{cases} Z := \epsilon_Z \\ X := 2Z + \epsilon_X \\ Y := 1X + 2Z + \epsilon_Y \end{cases} \quad \text{where } \epsilon_X, \epsilon_Z, \epsilon_Y \text{ are iid, } \sim \mathcal{N}(0, 1)$$

▶ Plot the DAG associated with this SCM.

```
# ------- Q1: Plot the DAG -----------
varnames <- c("X","Y","Z")
Adj <- matrix(c(0,1,0,
                0,0,0,
                1,1,0), 3,3, byrow = TRUE,
              dimnames = list(varnames,varnames))
```

```
laymat <- rbind(c(-1,-1),
                c(1,-1),
                c(0,1))
qgraph(Adj,
       layout = laymat,
       vsize = 10, esize = 10, asize = 10)
```



▶ Generate data based on this SCM. To do this:

1) Specify a sample size for the data you wish to generate (call it n, for example.)
2) Figure out which variable you need to generate first (Hint: start with exogenous variables, variables that are solely predictors in the model and do not serve as a dependent variable.)
3) You can use function `rnorm()` to sample random values from a normal distribution, for you specified sample size. Pick the appropriate mean and standard deviation based on the SCM.
4) Calculate dependent variables using your generated predictor variables, the regression coefficient values specified in the SCM, and generate residuals with rnorm().

```
#  -------  Q2: Generate data -----------
set.seed(3456)
n <- 500
Z <- rnorm(n,0,1)
X <- 2*Z + rnorm(n,0,1)
Y <- 1*X + 2*Z + rnorm(n,0,1)
```

▶ Using the simulated data, estimate the causal effect of X on Y. Here we are pretending to be researchers who know something about the SCM (the DAG and that the relationships are linear), and want to estimate a causal effect from observational data.

```
# I need to adjust for Z because of the open backdoor path
# That means I can run the following regression model:
coef1 <- lm(Y ~ X + Z)$coefficients
```

You have now specified an SCM, simulated observational data from it, and estimated the causal effect from that simulated data. Of course, we knew the causal effect of X on Y already, since it was in the equations we used to generate the observational data.

7. Causal Discovery using Conditional Independence Methods

In these exercises we will get to understand how causal discovery methods based on testing for conditional independence work. In part 7.1 you'll be asked to perform your own version of the PC algorithm from the ground up. This involves quite a few steps which will help you understand what's going on, but if you get stuck, we recommend skipping ahead to 7.2 and coming back to it later.

7.1: Do-it-yourself CI-based discovery

In this exercise you will get some practice with causal discovery through conditional independence testing. We have provided you with a dataset with four random variables:

```
data <- readRDS("data_cd_ex1.RDS")
```

In the lecture we discussed that the most basic approach to doing this is to test all possible (marginal and) conditional independence relations present in the data, and then try to draw all corresponding DAGs. In this first exercise we will guide you through how to do this step by step. In the next exercise, you'll try out software for causal discovery that uses slightly more sophisticated strategies.

▶ Write down all possible conditional and marginal independence relations it is possible to test in this dataset. We have four variables in total, so that means we need to test: a) all marginal dependencies, b) all conditional dependencies where we condition on one other variable, and c) all conditional dependencies where we condition on two other variables. There are *24* in total!

```
print(marginal_string)
```

```
##      DV1  DV2
## [1,] "X1" "X2"
## [2,] "X1" "X3"
## [3,] "X1" "X4"
## [4,] "X2" "X3"
## [5,] "X2" "X4"
## [6,] "X3" "X4"
```

$_4C_2 = \frac{4 \cdot 3}{2} = 6$

```
print(cond1_string)
```

```
##       DV1  DV2  Conditional On
##  [1,] "X1" "X2" "X3"
##  [2,] "X1" "X2" "X4"
##  [3,] "X1" "X3" "X2"
##  [4,] "X1" "X3" "X4"
##  [5,] "X1" "X4" "X2"
##  [6,] "X1" "X4" "X3"
##  [7,] "X2" "X3" "X1"
##  [8,] "X2" "X3" "X4"
##  [9,] "X2" "X4" "X1"
## [10,] "X2" "X4" "X3"
## [11,] "X3" "X4" "X1"
## [12,] "X3" "X4" "X2"
```

$_4C_2 \times {}_2C_1 = 6 \times 2 = 12$

```
print(cond2_string)
```

```
##      DV1  DV2  Conditional on And
## [1,] "X1" "X2" "X3"           "X4"
## [2,] "X1" "X3" "X2"           "X4"
## [3,] "X1" "X4" "X2"           "X3"
## [4,] "X2" "X3" "X1"           "X4"
## [5,] "X2" "X4" "X1"           "X3"
## [6,] "X3" "X4" "X1"           "X2"
```

$_4C_2 \times {}_2C_2 = 6$

▶ Assume normal errors and linear relationships. This means that we can use *correlations* to test for marginal independence and *partial correlations* to test for conditional independence (use an alpha level of .05). You can use the `ppcor` package for this, for example. The `cor.test` function can be used for calculating marginal correlations and accompanying p-values; the `pcor.test` function when you want to condition on one variable; the `pcor` function tests independence between pairs of variables, given all other variables in the dataset.

```
# Remember that the null hypothesis for each test is that the two variables are independ
ent.
# Use an alpha of .05 for each test.

# Test Marginal Independence using
martest1 <- cor.test(data[,"X1"], data[,"X2"])
martest1$p.value

# Test Conditional Independence using
#library(ppcor)
ctest1 <- pcor.test(data[,"X1"], data[,"X2"], data[,"X3"])
ctest1$p.value
```

```
alpha <- .05

# we can automate the correlation tests a bit like so:

# First let's test those marginal correlations
marg_p <- apply(marginal_string,1,function(r){
  cor.test(data[,r[1]], data[,r[2]])$p.value
})

# Now test the first set of conditional dependencies
c1_p <- apply(cond1_string,1,function(r){
  pcor.test(data[,r[1]], data[,r[2]], data[,r[3]])$p.value
})

# and the second set of conditional dependencies
c2_pmat <- pcor(data)$p.value # matrix of p-values for each pair given all other variabl
es
c2_p <- c2_pmat[lower.tri(c2_pmat)]

# Remember that the null hypothesis for each test is that the two variables are independ
ent

#Here are the results formatted in a table :
marg <- cbind(marginal_string,ifelse(marg_p < alpha, "Dependent", "Independent"))
c1 <- cbind(cond1_string,ifelse(c1_p < alpha, "Dependent", "Independent"))
c2 <- cbind(cond2_string,ifelse(c2_p < alpha, "Dependent", "Independent"))
marg; c1 ; c2
```

```
##      DV1  DV2
## [1,] "X1" "X2" "Dependent"
## [2,] "X1" "X3" "Dependent"
## [3,] "X1" "X4" "Dependent"
## [4,] "X2" "X3" "Dependent"
## [5,] "X2" "X4" "Dependent"
## [6,] "X3" "X4" "Dependent"
```

```
##       DV1  DV2  Conditional On
##  [1,] "X1" "X2" "X3"           "Dependent"
##  [2,] "X1" "X2" "X4"           "Dependent"
##  [3,] "X1" "X3" "X2"           "Dependent"
##  [4,] "X1" "X3" "X4"           "Dependent"
##  [5,] "X1" "X4" "X2"           "Dependent"
##  [6,] "X1" "X4" "X3"           "Dependent"
##  [7,] "X2" "X3" "X1"           "Independent"
##  [8,] "X2" "X3" "X4"           "Dependent"
##  [9,] "X2" "X4" "X1"           "Dependent"
## [10,] "X2" "X4" "X3"           "Dependent"
## [11,] "X3" "X4" "X1"           "Dependent"
## [12,] "X3" "X4" "X2"           "Dependent"
```

```
##      DV1  DV2  Conditional on And
## [1,] "X1" "X2" "X3"           "X4" "Dependent"
## [2,] "X1" "X3" "X2"           "X4" "Dependent"
## [3,] "X1" "X4" "X2"           "X3" "Independent"
## [4,] "X2" "X3" "X1"           "X4" "Dependent"
## [5,] "X2" "X4" "X1"           "X3" "Dependent"
## [6,] "X3" "X4" "X1"           "X2" "Dependent"
```

▶ List all of the *independencies* that you find. That is, what variables are marginally or conditionally independent of one another and under what conditions?

```
## [1] "X2 is Independent of X3 given X1"
```

```
## [1] "X1 is Independent of X4 given {X2 , X3 }"
```

stat faithfulness DAG

Now we want to use the independence relations we found above to infer the underlying DAG structure, assuming sufficiency and faithfulness. You could of course take a ``brute force" approach to this by drawing all of the four-variable DAGs that are possible, and ruling out one-by-one those that don't imply those independence relations. But it turns out there is an easier and more efficient way to do this. This method uses two principles. Here's the first:

Principle 1: Two variables A and B are directly connected in the DAG (either $A \rightarrow B$ OR $B \rightarrow A$) if and *only if* they are dependent conditional on **every possible subset** of the other variables.

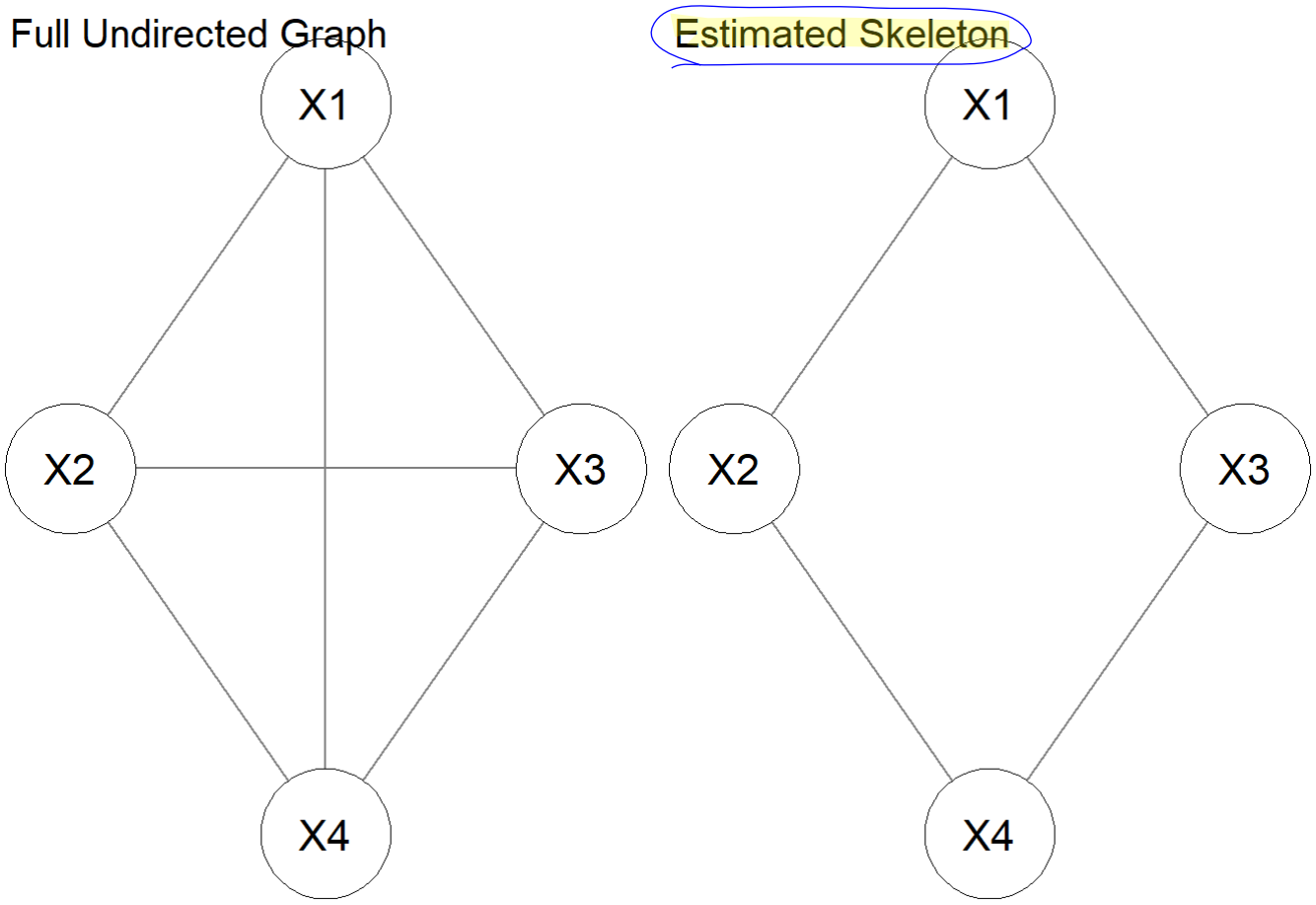
▶ Use this first principle to draw the **skeleton** of the DAG. Start by drawing an **undirected** graph where every variable is connected to every other variable. Then, remove edges between variables if they are either marginally or conditionally independent in **any** of the tests in the previous exercise. Tip: Use `qgraph` to make your undirected graph. Undirected graphs have a symmetric adjacency matrix, but you can also use the `directed = FALSE` option.

① Create a full undirected graph

```
#library(qgraph)

# Adj matrix for a ``full'' undirected graph
adj_full <- matrix(1,4,4)
diag(adj_full) <- 0

# make the layout custom (optional)
layout = matrix(c(0,1,-1,0,1,0,0,-1),4,2,byrow = T)

# Make the ``full'' graph
qgraph(adj_full, labels = names, layout = layout, directed = FALSE, title = "Full Undire
cted Graph", title.cex = 1.25, vsize = 15)
```

②

```
# Remove the edges between X2 - X3 and X1- x4
adj_full <- matrix(1,4,4)
diag(adj_full) <- 0
adj <- adj_full
adj[2,3] <- adj[3,2] <- 0
adj[1,4] <- adj[4,1] <- 0

# make the layout custom (optional)
layout = matrix(c(0,1,-1,0,1,0,0,-1),4,2,byrow = T)

par(mfrow=c(1,2))
qgraph(adj_full, labels = names, layout = layout, directed = FALSE, title = "Full Undire
cted Graph", title.cex = 1.25, vsize = 15)
qgraph(adj, labels = names, layout = layout, directed = FALSE, title = "Estimated Skelet
on", title.cex = 1.25, vsize = 15)
```



Having obtained the DAG skeleton, we can now try to give a *direction* to as many edges as possible. Recall from the lecture that our ability to orient the arrows in a DAG using only conditional independence information is reliant on the presence of **collider** structures. This leads us to our second principle for inferring DAG structures:

Principle 2: If our skeleton contains a triplet $A - B - C$, we can orientate the arrows as $A \rightarrow B \leftarrow C$ if and only if A and C are **dependent** conditional on *every set of variables containing* B.

This is a slightly trickier principle to wrap your head around, but it essentially is just a re-statement of the d-seperation rules for colliders.

▶ Use this second principle to give a direction to as many arrows as possible in the skeleton. What is the resulting CPDAG? Tip: With qgraph, use `bidirectional = TRUE` or see the help for the `directed` argument.

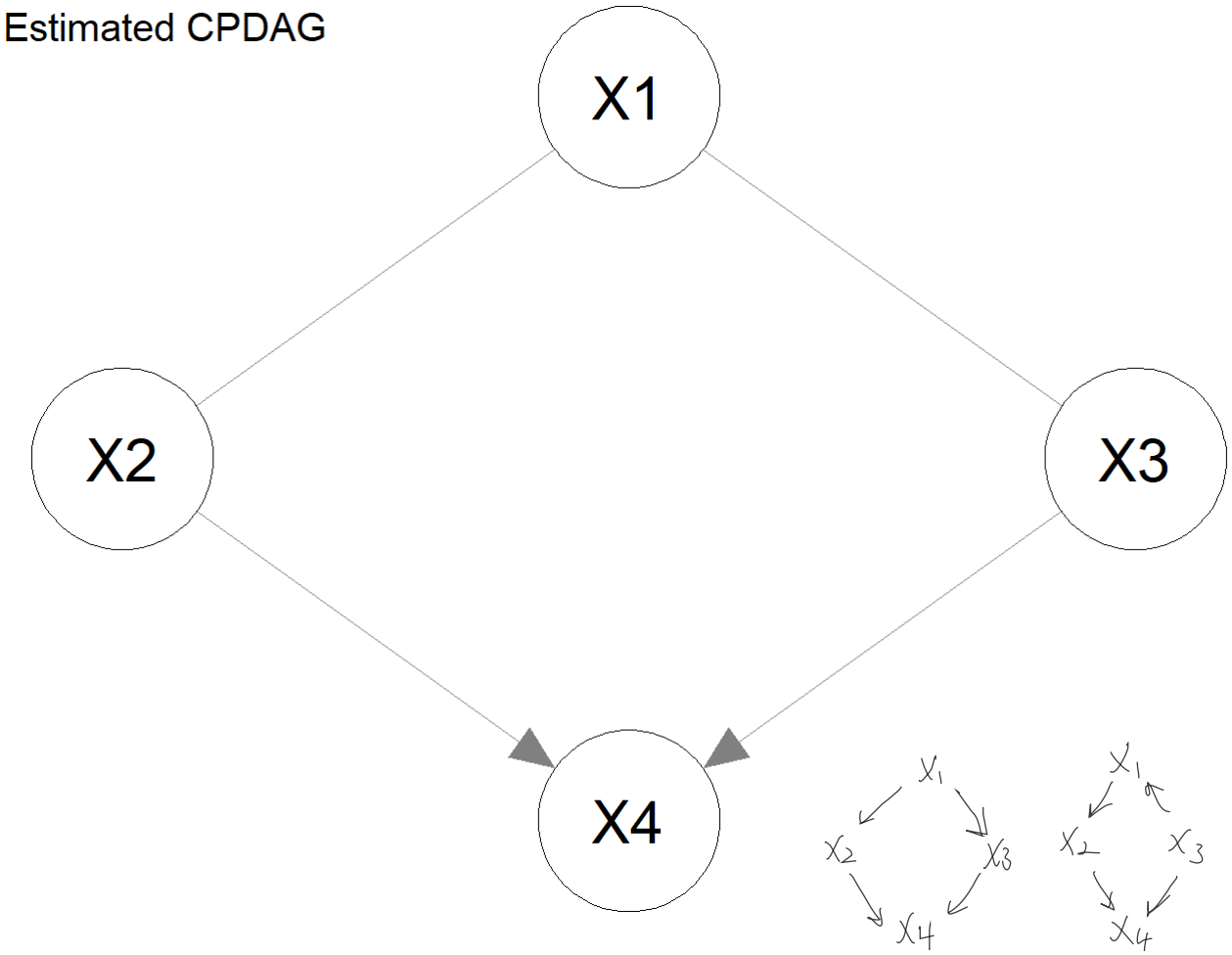
```
## There are four triplets you must consider:
##  A) X2 - X1 - X3
##  B) X1 - X3 - X4
##  C) X1 - X2 - X4
##  D) X2 - X4 - X3
```

```
## We rule out A) because we found above that
##  X2 and X3 are independent given X1
```

```
## We rule out C) and B) because we found that
## X1 is independent of X4 given {X2 , X3}
```

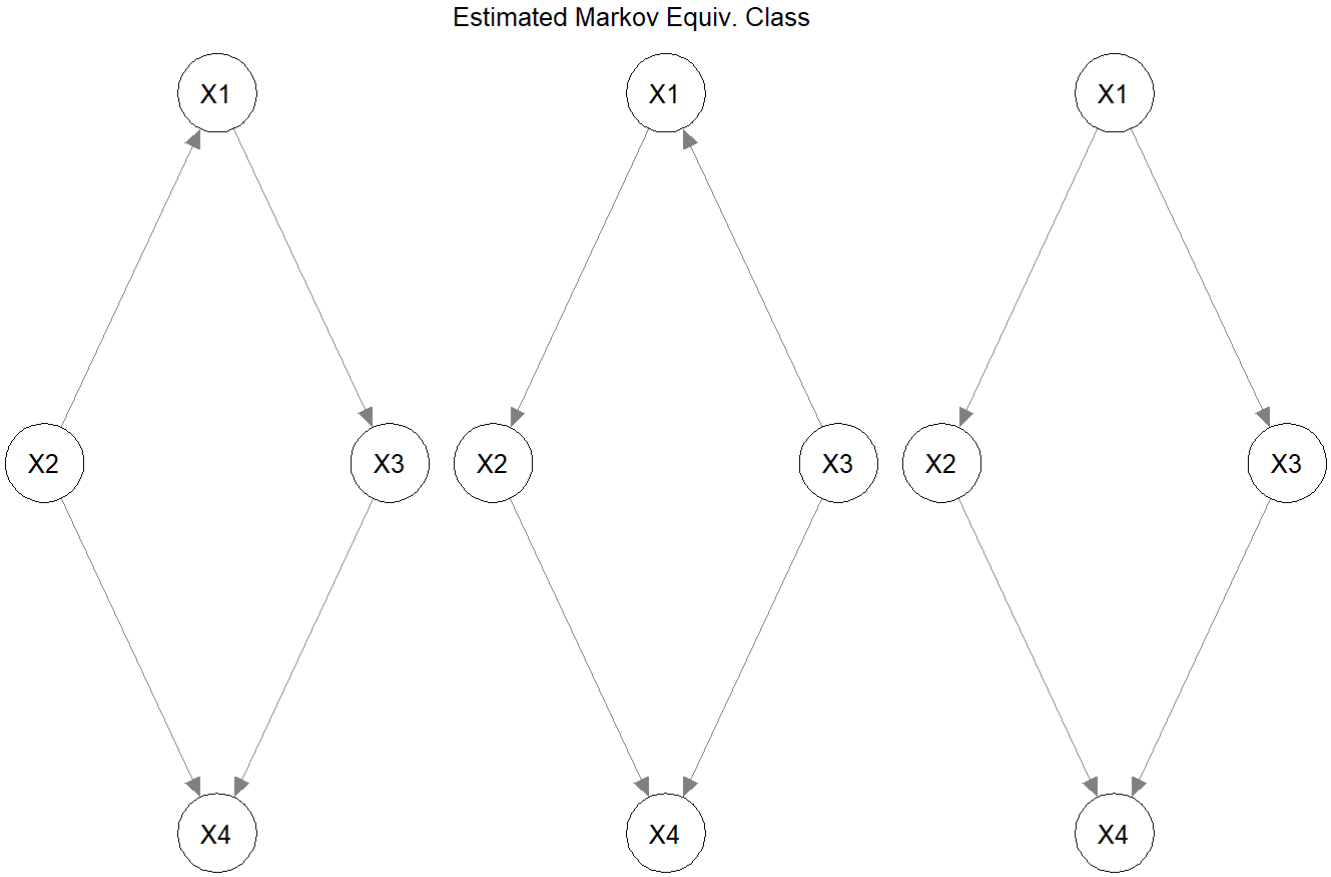
```
## But X2 and X3 are always dependent (given either X4 or {X1, X4})
##  That means X2 -> X4 <- X3
```

```
cpdag <- adj
cpdag[4,2] <- 0 # we know that this arrow goes X2 -> X4
cpdag[4,3] <- 0 # we know the direction is X3 -> X4

# extra touch - making a matrix that indicates we have a mix of directed (true) and undi
rected (false) edges
cptf <- matrix(FALSE, 4,4)
cptf[2,4] <- cptf[3,4] <- TRUE

par(mfrow = c(1,1))
qgraph(cpdag, labels = names, layout = layout, directed = cptf, title = "Estimated CPDA
G", title.cex = 1.25, asize = 8, vsize = 15)
```



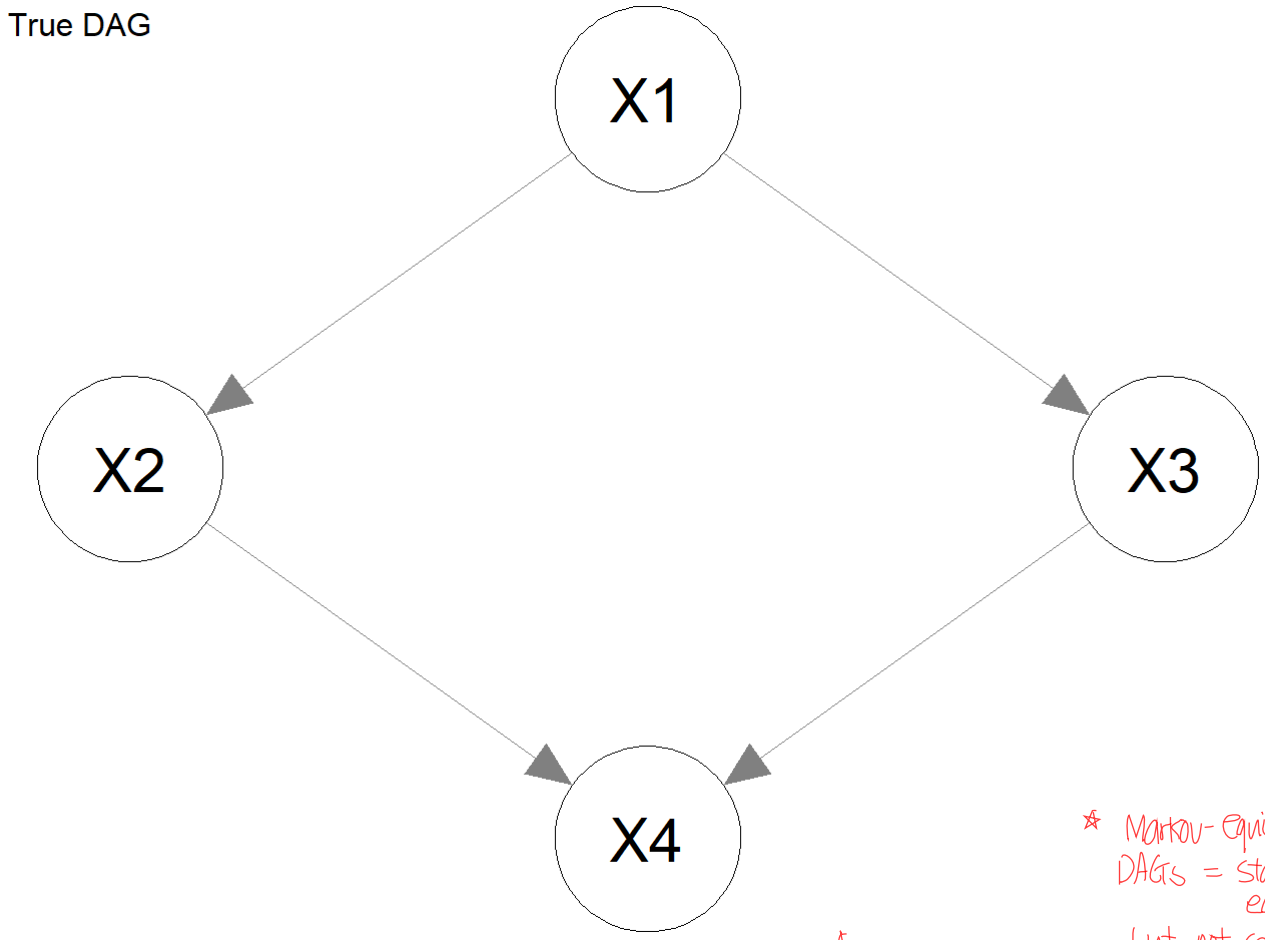
▶ Draw all of the DAGs that make up the estimated Markov Equivalence Class. You might find it helpful to do this with pen and paper first before transferring to qgraph.

```
dag1 <- matrix(c(
   0  ,  0  ,  1  ,  0,
   1  ,  0  ,  0  ,  1,
   0  ,  0  ,  0  ,  1,
   0  ,  0  ,  0  ,  0
), 4, 4, byrow = T)

dag2 <- matrix(c(
   0  ,  1  ,  0  ,  0,
   0  ,  0  ,  0  ,  1,
   1  ,  0  ,  0  ,  1,
   0  ,  0  ,  0  ,  0
), 4, 4, byrow = T)

dag3 <- matrix(c(
   0  ,  1  ,  1  ,  0,
   0  ,  0  ,  0  ,  1,
   0  ,  0  ,  0  ,  1,
   0  ,  0  ,  0  ,  0
), 4, 4, byrow = T)

par(mfrow = c(1,3))
qgraph(dag1, labels = names, layout = layout, directed = TRUE, asize = 8, vsize = 15)
qgraph(dag2, labels = names, layout = layout, directed = TRUE, title = "Estimated Markov
Equiv. Class", title.cex = 1.25, asize = 8, vsize = 15)
qgraph(dag3, labels = names, layout = layout, directed = TRUE, asize = 8, vsize = 15)
```




```
## Notice that not all arrow orientations are allowed
##
##     You cannot create any new collider structures that aren't already in the CPDAG
##
##     We already ruled those out in the last step!
```

The true DAG structure used for data generation is given below. Check that this is included in your estimated Markov Equivalence Class!



* Markov-equivalence class DAGs = statistically equivalent but not causally equivalent!

Notice that each of the graphs in the Markov Equivalence class imply the same set of statistically dependencies we would expect to see in observational data. This is one way in which we might say that different causal models are *statistically equivalent*, or to be more precise, **markov equivalent**. However, these DAGs are not *causally equivalent*.

▶ Imagine that we are interested in the causal effect of X_1 on X_4. Use linear regression to estimate the effect from the observational data, using each of the DAGs in the Markov Equivalence class in turn to derive how this should be done (Assume the effects are linear). What do the different DAGs and your accompanying estimates of the causal effects imply?

```
## According to the first DAG, we should adjust for X2
```

```
m1 <- lm(X4 ~ X1 + X2, data = as.data.frame(data))$coefficients
m1
```

```
## (Intercept)          X1          X2
##   0.1304026   2.5873460  -1.7917264
```

```
#the estimated causal effect of X1 on X4 is equal to ~2.6
```

```
## According to the second DAG, we should adjust for X3
```

```
m2 <- lm(X4 ~ X1 + X3, data = as.data.frame(data))$coefficients
m2
```

```
## (Intercept)          X1          X3
## -0.07670387 -0.99882035  2.55057292
```

```
#the estimated causal effect of X1 on X4 is equal to ~-1
```

```
## According to the third (true) DAG, we should not adjust for anything
```

```
m3 <- lm(X4 ~ X1, data = as.data.frame(data))$coefficients
m3
```

```
## (Intercept)          X1
##  0.07310298  1.59165977
```

```
#the estimated causal effect of X1 on X4 is equal to ~1.6
```

```
## Even though each DAG is compatible with the same conditional (in)dependence relations
\
##       They each imply quite a different causal effect of the same variable.
##       We could find out which DAG is the right one by performing an experiment wih an
intervention and comparing our estimates....\
##       But, we won't do that in this lab!
```

Because we simulated this data, we know the true causal effect of $X1$: The effect of $X1$ on $X4$ is 1.65.

7.2: Intro PC algorithm

The approach we took in the previous exercise worked, but was actually quite inefficient. Consider again the two principles we used to create first a skeleton and second a CPDAG from conditional independence tests in the previous exercise.

- In order to omit the edge $X2 - X3$ from the skeleton, all we needed to know was that they were independent given $X1$. Remember: If two variables are directly causally dependent $X2 \rightarrow X3$ or $X2 \leftarrow X3$, then they will *never* be statistically independent, no matter what we condition on! So, once we knew that $X2$

and $X3$ were independent given $X1$, there was actually no need to also test whether they were independent given $X4$ or given $X1, X4$. Since independence only tells us that we should remove an edge from the skeleton, and we already removed the edge $X2 - X3$, the information given by those last two tests wasn't used to make any decisions about the skeleton, so we never needed to do those tests in the first place.

- Once we have the skeleton, we need to look for potential collider structures by looking at *triplets* $A - C - B$ where there is no direct edge between A and B. If we have such a structure, we then need to test whether A and B are dependent given C.

② Find colliders
Check triplets

Rather than test all possible conditional independence relations, we could design an algorithm which uses these two insights in order to more efficiently estimate a CPDAG. Luckily, Spirtes, Glymour & Scheines (2000) already had this insight: This is the exact logic of their **PC algorithm**.

A full description of how the `pcalg` package works can be found here (https://cran.r-project.org/web/packages/pcalg/vignettes/pcalgDoc.pdf).

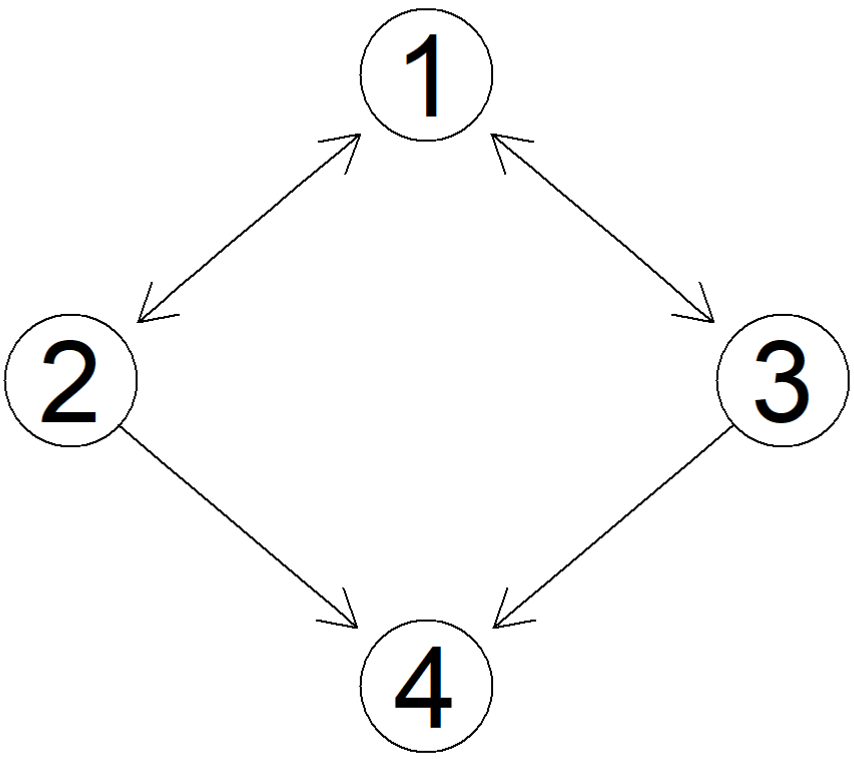
The function `pcalg::pc()` estimates the Markov Equivalence Class (CPDAG) using conditional independence tests as described above (assuming sufficiency and faithfulness). In order to do this, we need to tell the function what conditional independence test should be used thought the `indepTest` argument. `pcalg` comes with three pre-defined independence tests: `gaussCItest` for Gaussian variables, based on partial correlations, as well as `discCItest` and `binCItest` for discrete and binary variables, respectively. We also need to define an appropriate alpha level to be used for these independence tests. Here, let's use the `alpha = .05`.

Finally, for reasons of computational efficiency, `pc()` doesn't work with *raw data* but instead with a list containing *sufficient statistics* (`suffStat`): A summary of the relevant information from which the conditional independence tests can be calculated. Although this may seem strange in the current context, this helps speed things up when we have very large datasets. For Gaussian variables, the sufficient statistics are a) the correlation matrix and b) the sample size. See the examples under the `pc()` help file (`?pc`) for examples with binary and discrete data. Unfortunately, in the basic pc-algorithm and the current package, discrete and continuous variables can not be combined in the same graph.

```
suffStat <- list(C = cor(data), n = nrow(data))
```

Let's put all of this information together and use the PC algorithm with the data we gave you in the last exercise. Use the `pc()` function to estimate a Markov Equivalence Class and plot it.

```
pc_fit1 <- pc(suffStat = suffStat, indepTest = gaussCItest,
p = ncol(data), alpha = 0.01)
# This is the default plotting method for pcalg - uses Rgraphviz
plot(pc_fit1, main = "Inferred CPDAG using pcalg")
```

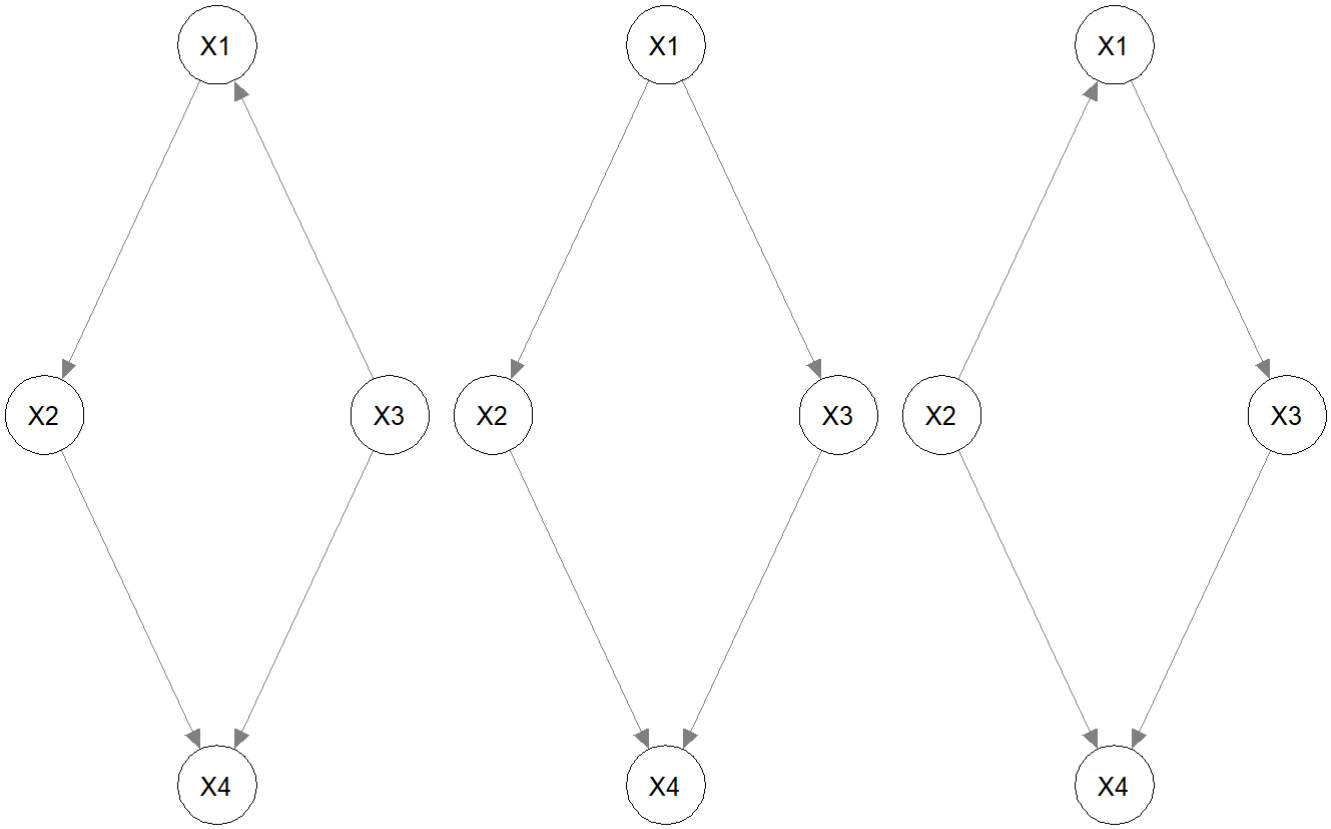



```
# You can also extract the adjacency matrix and plot using ggraph
# Note that you have to transpose the matrix; pcalg writes matrices from column to row
 # cpdag_mat <- as(pc_fit1,"matrix")
 # ggraph(t(cpdag_mat), labels = names, layout = layout, directed = cptf, title = "Estim
ated CPDAG", title.cex = 1.25, asize = 8)
```

The `pc` function provides you with the CPDAG directly (though notice that, in the background, first the skeleton is estimated using `skeleton` and then the collider structures are found as described earlier). This should be the same graph you estimated in the previous exercise.

We can get all of the separate DAGs in the equivalence class using the following code

```
# Extract the adjacency matrix of the cpdag from pc_fit1
cpdag_mat <- as(pc_fit1,"matrix")

# Each row is a DAG adjacency matrix in vector form (by rows)
res1 <- pdag2allDags(cpdag_mat)

# We can get the adjacency matrix of an individual DAG using
res1_dags <- list()
for(i in 1:nrow(res1$dags)){
  res1_dags[[i]] <- t(matrix(res1$dags[i,],4,4,byrow = TRUE))
}
# Notice we have to transpose the adjacency matrix here for qgraph!

# We can plot each of these just as we did above
par(mfrow = c(1,3))
for(i in 1:3){
  qgraph(res1_dags[[i]], labels = names, layout = layout, directed = TRUE, asize = 8, vs
ize = 15)
}
```



We can use the `ida()` function to estimate the effect of an intervention according to each of the DAGs in the Markov Equivalence set. To do this we need to provide the output of the `pc` function and the covariance matrix of the data. So, to find the causal effect of X_1 on X_4 we would use the code

```
ida(1,4,cov(data), pc_fit1@graph, verbose = TRUE)
```

```
##
##
## x= 1 y= 4
## pa1=
## pa2= 2 3
## Fit - y: 4 x: 1 |b.hat= 1.59166
## Fit - y: 4 x: 1 2 |b.hat= 2.587346
## Fit - y: 4 x: 1 3 |b.hat= -0.9988204
```

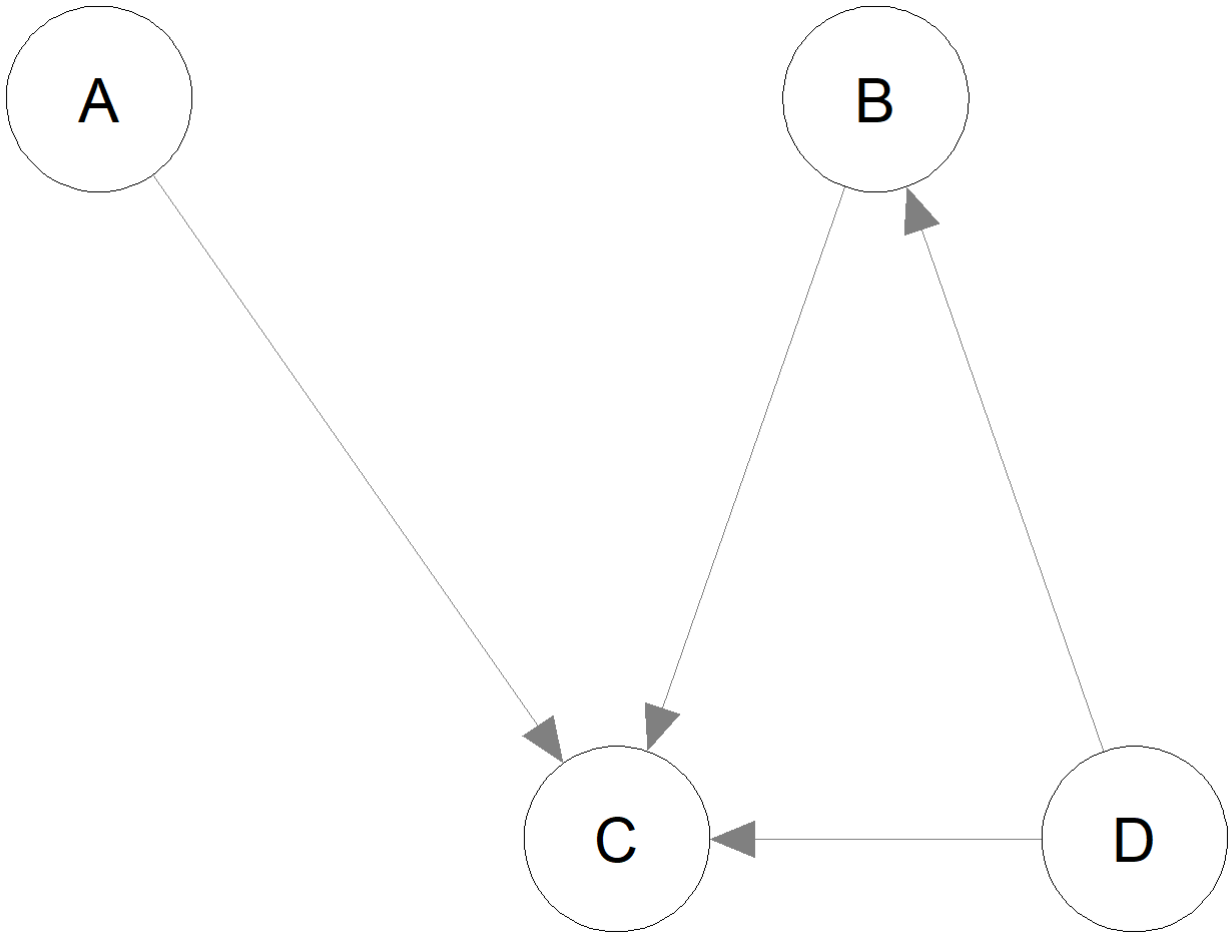
```
## [1]  1.5916598  2.5873460 -0.9988204
```

This output gives us three numbers, each corresponding to an estimate of the effect of this intervention in one of the three DAGs in the Markov Equivalence set. By specifying `verbose = TRUE` we can see what regressions each number comes from. Compare this to the results you obtained from manually calculating this in the previous exercise - you should see you get *approximately* the same results. Differences come from the fact that we use raw data above, and only summary statistics here. (Note that the order of the effects does not necessarily correspond to the order in which the DAGs were plotted earlier!)

7.3: PC algorithm in action

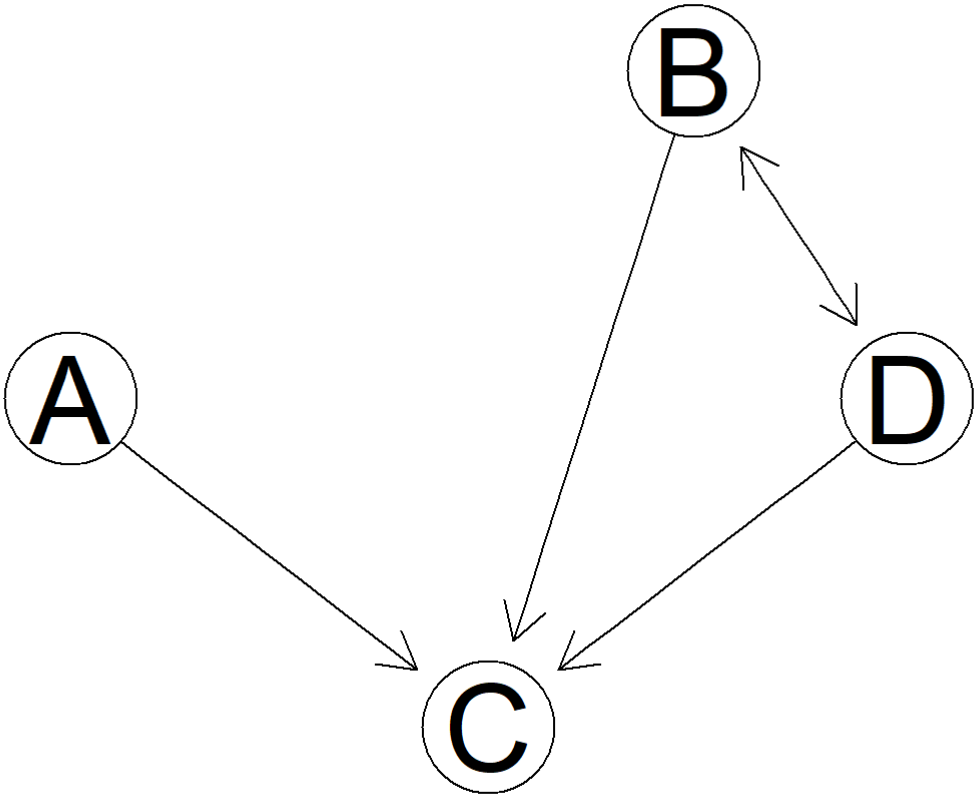
Now that you've seen how the `pcalg` package works, let's try it out on a new example. Suppose that you know the true DAG is given by the following graph

```
names <- LETTERS[1:4]
adjmat2 <- matrix(c(0,0,1,0,
                    0,0,1,0,
                    0,0,0,0,
                    0,1,1,0),4,4,byrow = TRUE)
lay2 <- matrix(c(-1,1,
                 .5,1,
                  0,0,
                  1,0),4,2,byrow = TRUE)
qgraph(adjmat2, labels = names, layout = lay2, directed = TRUE, asize = 8, vsize = 15)
```



▶ What do you expect the PC algorithm to be able to recover? What is the CPDAG for this DAG?

```
# You can work this out by hand, or ``cheat'' by using the pcalg package
adjmat2 <- matrix(c(0,0,1,0,
                    0,0,1,0,
                    0,0,0,0,
                    0,1,1,0),4,4,byrow = TRUE)
colnames(adjmat2) <- rownames(adjmat2) <- names
g2 <- as(adjmat2,"graphNEL") # convert to graphNEL object for pcalg
# This function converts a DAG to a CPDAG
cpdag2 <- dag2cpdag(g2)
plot(cpdag2)
```



Suppose that we have the following data, generated according to the above DAG

```
set.seed(1234)
n <- 3000
```

① SCM

```
A <- rnorm(n)
D <- rnorm(n)
B <- 0.50 * D + rnorm(n)
C <- -0.75 * A + B + D +  rnorm(n)

data2 <- cbind(A, B, C, D)
```

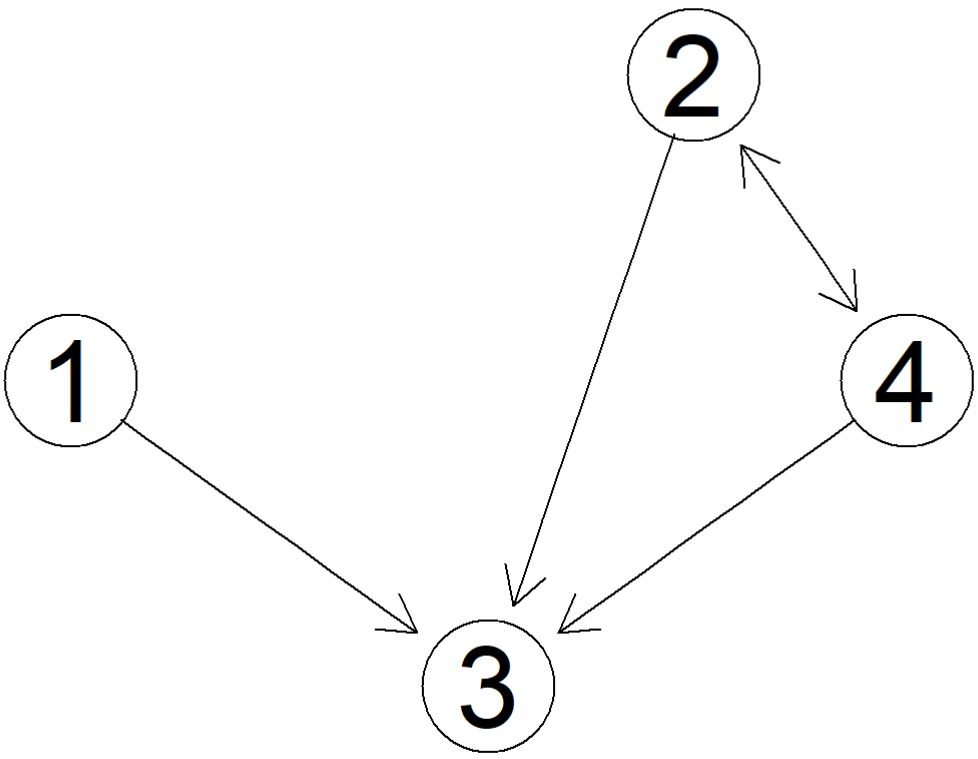
▶ Use the `pcalg` package to estimate the CPDAG. Do you obtain the correct CPDAG?

② Sufficient statistic = correlation matrix & sample size (n)

```
suffStat2 <- list(C = cor(data2), n = nrow(data2))
```

③ specify pc algorithm

```
pc_fit2 <- pc(suffStat = suffStat2, indepTest = gaussCItest,
p = ncol(data), alpha = 0.01)
# This is the default plotting method for pcalg - uses Rgraphviz
plot(pc_fit2, main = "Inferred CPDAG using pcalg")
```



▶ Estimate the causal effect of A on C

▶ Estimate the effect on D on C

```
# Part 1: causal effect of A on C
ida(1,3,cov(data2), pc_fit2@graph, verbose = TRUE)
```

A C

```
##
##
## x= 1 y= 3
## pa1=
## pa2=
## Fit - y: 3 x: 1 |b.hat= -0.7318286
```

```
## [1] -0.7318286
```

```
# Part 2: causal effect of D on C
ida(4,3,cov(data2), pc_fit2@graph, verbose = TRUE)
```

D C

```
##
##
## x= 4 y= 3
## pa1=
## pa2= 2
## Fit - y: 3 x: 4 |b.hat= 1.480331
## Fit - y: 3 x: 4 2 |b.hat= 1.011716
```

we get 2 becuz there're 2 possible paths { direct 4→3, indirect 4←2→3

```
## [1] 1.480331 1.011716
```

```
## Just as we saw in the previous answer, we are typically uncertain about causal effect
s \
##     because we are uncertain about the causal structure! \
##     Here however, we see the power of identifying collider structures. In this case w
e have identified \
##     that A -> C, so we only get a single estimate for the effect of A on C. \
##     However, we are not certain whether D only has a direct effect, or also an indire
ct effect on C\
##     Because we don't know whether B -> D or D -> B, we get two different estimates of
the intervention effect!
```

4→3, 4←2→3

Bonus. Generating/Simulating Data from an SCM II.

In this exercise you can get some more practice with simulating data from a structural causal model. We'll then use this simulated data to explore different statistical dependencies that are present in our causal system.

In the lecture we described how a collider structure $X \rightarrow Z \leftarrow Y$ implies the following independence and dependence relationships:

- X and Y are marginally independent
- X and Y are conditionally dependent given Z

I want you to simulate data from an SCM with three variables in a collider structure. So, you need to create your own SCM for a collider DAG and generate data from it. I want you to do this so that both X and Y are independent of each other, but they have a positive effect on Z. For the rest, choose any type of distribution you like for the variables (incl residuals), and any form of relationship (linear, non-linear) for the effects of the predictors on Z.

<Generate Data>

(1) ▶ Start by specifying the model equations for your SCM. Pick distributions for X and Y (1-1) to generate the type of variable of your choice (e.g., you could use a bernoulli distribution for a dichotomous variables, normal distribution for a continuous variable, ….etc). Specify some form of regression model for Z, such that Z depends on X and Y. (1-2)
For example, if Z is to be binary, consider specifying a logistic regression model for Z. If Z is to be continuous, you

could consider a normal linear regression model to generate Z, for example. Note that if you use non-linear relationships, the size of the causal effect on your outcome variable will differ depending on the size of the score of your predictor variable (e.g., changing from X=0 to X=1 will not necessarily have the same effect on Y as changing from X=1 to X=2)!

② ▶ Next, simulate data from your SCM. Generate 2000 observations of X and Y, such that they are independent from one another. To do this, you need a function to sample values for X and Y from the probability distribution you specified in your SCM equations. You can use function `rbinom` to sample values for bernoulli (dichotomous) variables and `rnorm` to sample values for normally distributed variables. See the information on other distributions in R here (https://stat.ethz.ch/R-manual/R-devel/library/stats/html/Distributions.html)

① ② pick distribution for X & Y

```
# First, set a random seed so we can reproduce our results
set.seed(45)
# Pick a large sample size
n <- 2000

# For example, you could use two normal distributions to generate X and Y


X <- rnorm(n, mean = 0, sd = 1)
Y <- rnorm(n, mean =0, sd = 1)

#In the above we have created data for 2000 people. Both the X and Y variables have a me
an of 0 and standard deviation of 1, and the two variables are unrelated.
```

② specify Z using a regression model

▶ Now, generate Z based on your specified regression model and the predictors your generated.

normal linear model ~ Z = continuous normal

```
#Example for three normal variables X Y Z,
# where Z is a linear function of X and Y with normally distributed residuals.
# I chose regression coefficients equal to 1 for X and 2 for Y
# I chose a residual standard deviation of 0.2.
# Z <- 1*X + 2*Y + rnorm(n, mean = 0, sd = .2)
# I could also have specified this like so
# Z <- rnorm(n, mean = 1*X + 2*Y, sd = .2 )
```

logistic reg ~ Z = dichotomous

```
# Example where X and Y are normal, but Z is a dichotomous variable.
# First, I specify the linear function for the log-odds of Z as a function of X and Y:
l <- 3*X + 4*Y #I chose logistic regression coefficients equal to 3 and 4 respectively.
# Now we transform the generated log-odds to probabilities using a standard logistic reg
ression equation
p_z <- 1/(1+ exp(-1*l)) # If this confuses you - wikipedia logistic regression!
# Now we want to sample binary values for Z based on these probabilities so we obtain ou
r dichotomous variable. We do this by drawing 0/1 values from from a binomial/bernoulli
 distribution based on the probabilities:
Z_binom <- sapply(p_z,function(vec) rbinom(1,1,prob = vec))
```

▶ Now, estimate and/or visualize the dependency between X and Y while ignoring Z (i.e. compute their marginal dependency).

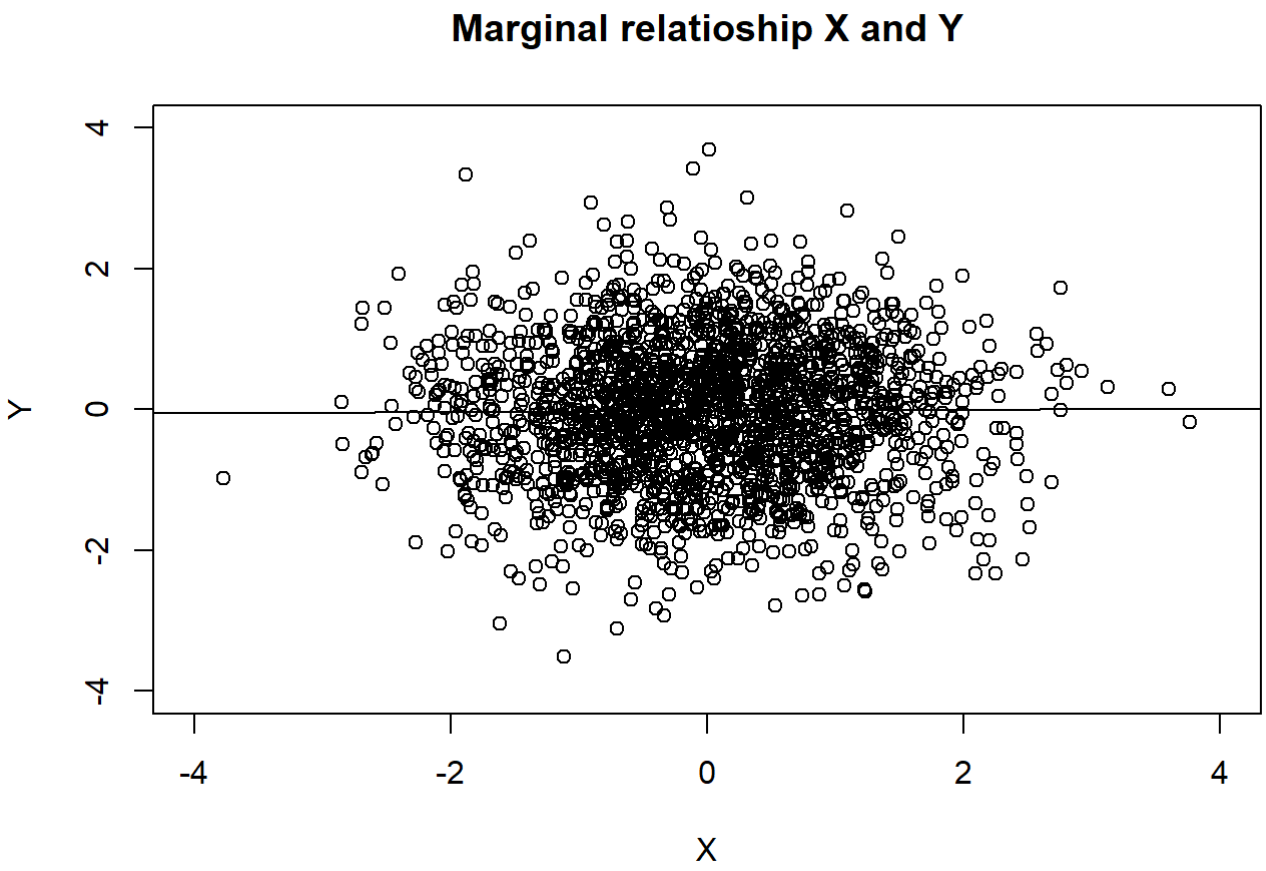
```
# A simple descriptive such as a correlation could do
cor(X,Y)
```

```
## [1] 0.008543307
```

```
# Or we can use linear regression
summary(lm(Y~X))$coefficients
```

```
##                Estimate Std. Error    t value  Pr(>|t|)
## (Intercept) -0.016817755 0.02185768 -0.7694207 0.4417345
## X            0.008275609 0.02167007  0.3818912 0.7025827
```

```
coef1 <- lm(Y~X)$coefficients

# We can visualize this like this, for example:
plot(X, Y, main = "Marginal relatioship X and Y", xlim = c(-4,4), ylim = c(-4,4))
abline(a = coef1[1], b = coef1[2], col = "black")
```



▶ Finally, estimate and/or visualize the conditional dependency between X and Y given Z. Think about the different ways in which you can condition on a variable! For instance, you can try to control for Z statistically (as in a regression model), or select data points based on their Z value. What do you see in comparison to the previous step? Can you explain the results?

```
# There are many ways to condition on Z.

### For the example with a continuous Z with normal residuals:###
# Conditioning with regression:
# summary(lm(Y ~ X + Z))$coefficients

# We can also, for instance, condition on Z by selecting only the people with specific values for Z
#  # We can also, for instance, select only high values of Z.
# sel <- Z > 0
 # If we "unknowingly" select only high Z individuals, we get a negative dependency between X and Y
# summary(lm(Y[sel] ~ X[sel]))$coefficients
# coef2 <- lm(Y[sel] ~ X[sel])$coefficients
 # The same applies here if we select only low Z individuals, we get a negative dependency between X and Y
# summary(lm(Y[!sel] ~ X[!sel]))$coefficients
# coef3 <- lm(Y[!sel] ~ X[!sel])$coefficients

 #lets visualize this, to clarify what is happening.
# plot(X[!sel], Y[!sel],
#     main = "Relationship X and Y conditional on Z", xlim = c(-4,4), ylim = c(-4,4),
#     col = "blue", xlab = "X", ylab = "Y")
# points(X[sel], Y[sel], col = "red")

# abline(a = coef2[1], b = coef2[2], col = "red") #regression line only high Z
# abline(a = coef3[1], b = coef3[2], col = "blue") #regression line only low Z
# abline(a = coef1[1], b = coef1[2], col = "black") #regression line any Z

##For the example with dichotomous Z##
# Regression coefficient here is positive
summary(lm(Y ~ X + Z_binom))$coefficients
```

```
##               Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) -0.7661356 0.02418147 -31.68275 7.866886e-179
## X           -0.3364738 0.01794835 -18.74678  2.326920e-72
## Z_binom      1.4990965 0.03620703  41.40346 4.798553e-271
```

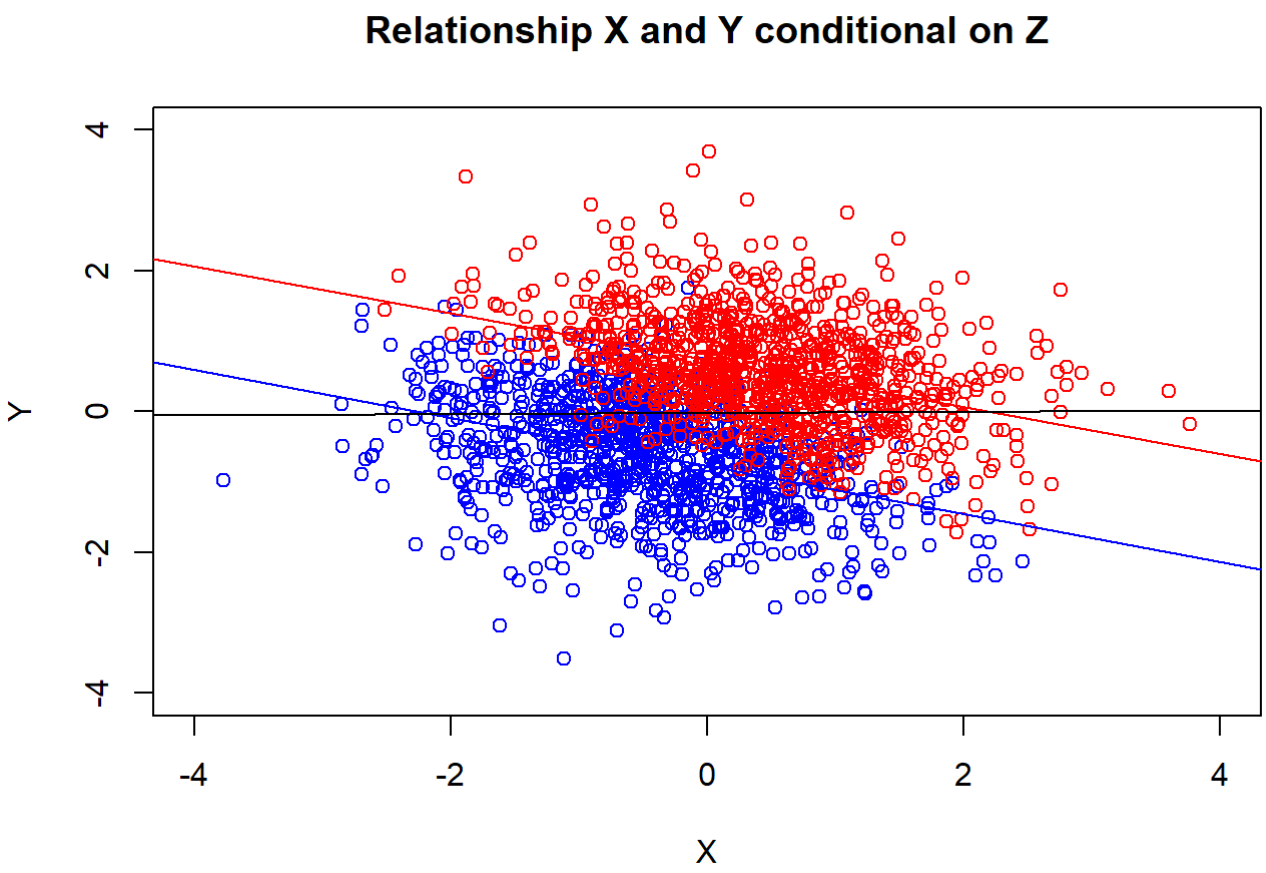
```
# Another way to condition on Z is to select only high Z individuals
sel <- (Z_binom == 1)
# Now let's run a regresion with Y predicted by X, but in the "subpopulation" of high Z
 people
summary(lm(Y[sel] ~ X[sel]))$coefficients
```

```
##               Estimate Std. Error   t value      Pr(>|t|)
## (Intercept)  0.7309869 0.02514922  29.06599 5.771813e-135
## X[sel]      -0.3322148 0.02512709 -13.22138  6.979360e-37
```

```
coef4 <- lm(Y[sel] ~ X[sel])$coefficients

summary(lm(Y[!sel] ~ X[!sel]))$coefficients
```

```
##              Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) -0.768079 0.02605493 -29.47922 5.311695e-138
## X[!sel]     -0.340588 0.02563632 -13.28537  3.253957e-37
```

```
coef5 <- lm(Y[!sel] ~ X[!sel])$coefficients

plot(X[!sel], Y[!sel],
    main = "Relationship X and Y conditional on Z", xlim = c(-4,4), ylim = c(-4,4),
    col = "blue", xlab = "X", ylab = "Y")
points(X[sel], Y[sel], col = "red")

abline(a = coef4[1], b = coef4[2], col = "red") #regression line for Z=1 group
abline(a = coef5[1], b = coef5[2], col = "blue") #regression line for Z=0 group
abline(a = coef1[1], b = coef1[2], col = "black") #regression line for all Z
```


Relationship X and Y conditional on Z
Y
X
4
2
0
-2
-4
-4
-2
0
2
4