The four horsemen of causal doom

Rafał Urbaniak, Nikodem Lewandowski
(LoPSE research group, University of Gdansk)

Multicolinearity

```
data <- as.data.frame(read_xpt("crimeLife.xpt"))
small <- data[,c(38,54, 69,70,75)]
names(small) <- c("convictions10to14","conductDisorder",
                  "family", "brothers", "sisters")
head(small, n = 3)
```

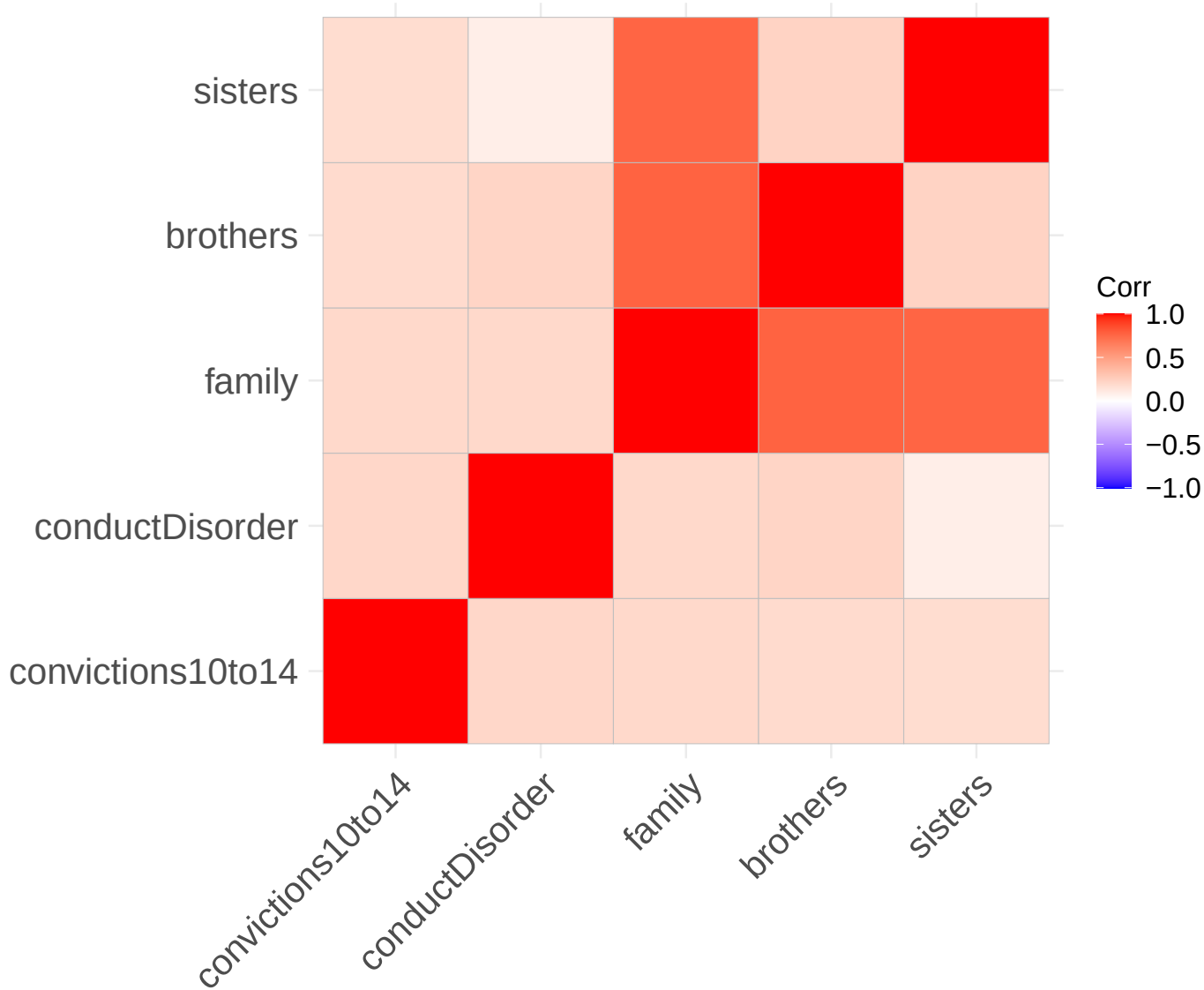
```
##   convictions10to14 conductDisorder family brothers sisters
## 1                 1                1      2        1       2
## 2                 1                2      1        1       1
## 3                 2                4      5        4       2
```

Watch out:

- 1 means zero, 15 means 14
- disorder are actually categories based on teacher and interviewer ratings
- family size: no of siblings 1 means zero.
- brothers and sisters are also categories

Multicolinearity

```
cors <- cor(small, method = "spearman")
ggcorrplot(cors) + corSize
```



Multicolinearity

```
set.seed(123)
convictionsDisorder <- ulam(
  alist(
    convictions10to14 ~ dnorm( mu , sigma ) ,
    mu <- a + d * conductDisorder,
    a ~ dunif(0,20) ,
    d ~ dnorm( 0 , .5 ) ,
    sigma ~ dunif( 0 , 10 )
  ) , data=small, log_lik = TRUE )
```

```
precis(convictionsDisorder)
```

```
##             mean         sd      5.5%     94.5%    n_eff     Rhat4
## a      0.5726957 0.27862271 0.1248490 1.0114766 168.1379 1.033089
## d      0.5785618 0.10263369 0.4303709 0.7429845 164.1248 1.027074
## sigma 2.3675437 0.08292879 2.2538133 2.5114716 234.0522 1.006964
```

Multicolinearity

```
set.seed(123)
convictionsAllPredictors <- ulam(
  alist(
    convictions10to14 ~ dnorm( mu , sigma ) ,
    mu <- a + d * conductDisorder +
      f * family + b * brothers + s * sisters,
    a ~ dunif(0,20) ,
    d ~ dnorm( 0 , .5 ) ,
    f ~ dnorm( 0 , .5 ) ,
    b ~ dnorm( 0 , .5 ) ,
    s ~ dnorm( 0 , .5 ) ,
    sigma ~ dunif( 0 , 10 )
  ) , data=small, log_lik = TRUE )
```

```
precis(convictionsAllPredictors)
```

```
##              mean         sd         5.5%      94.5%    n_eff      Rhat4
## a       0.1081210 0.09914214  0.006793236  0.2927177 241.9774 0.9991276
## d       0.3243791 0.08132320  0.188445045  0.4404111 334.7369 0.9981002
## f      -0.3953825 0.24008387 -0.795158510 -0.0138054 161.1953 0.9984832
## b       0.5472605 0.22215415  0.180163030  0.8835733 202.1591 0.9980338
## s       0.6068266 0.21711732  0.268673820  0.9645994 165.3614 0.9988570
## sigma   2.2780210 0.07768604  2.154735000  2.4063900 308.7138 1.0002456
```

Multicolinearity

```
set.seed(666)
convictionsFamily <- ulam(
  alist(
    convictions10to14 ~ dnorm( mu , sigma ) ,
    mu <- a + d * conductDisorder + f * family,
    a ~ dunif(0,20) ,
    d ~ dnorm( 0 , .5 ) ,
    f ~ dnorm( 0 , .5 ) ,
    sigma ~ dunif( 0 , 10 )
  ) , data=small, log_lik = TRUE )
```

```
precis(convictionsFamily)
```

```
##             mean         sd       5.5%     94.5%    n_eff      Rhat4
## a      0.2076457 0.16337786 0.01830126 0.5230144 296.6785 1.0006334
## d      0.4187173 0.08803188 0.28455213 0.5645891 179.8419 0.9985192
## f      0.2467413 0.06986919 0.13589510 0.3606546 161.0151 1.0024382
## sigma 2.3098005 0.08347942 2.18393060 2.4508232 246.8665 0.9983015
```

Multicolinearity

```
compare(convictionsAllPredictors, convictionsFamily, convictionsDisorder)
```

```
##                              WAIC       SE    dWAIC       dSE    pWAIC
## convictionsAllPredictors 1851.666 68.32488  0.00000        NA 9.256251
## convictionsFamily        1863.250 68.26536 11.58385  4.583978 9.000348
## convictionsDisorder      1877.225 68.65968 25.55924 10.712589 8.051456
##                                weight
## convictionsAllPredictors 9.969544e-01
## convictionsFamily        3.042806e-03
## convictionsDisorder      2.809037e-06
```

Multicolinearity and milk

```
data(milk)
d <- milk
d$K <- scale( d$kcal.per.g )
d$F <- scale( d$perc.fat )
d$L <- scale( d$perc.lactose )
```

Multicolinearity and milk

```
milkFat <- quap(
  alist(
    K ~ dnorm( mu , sigma ) ,
    mu <- a + bF*F ,
    a ~ dnorm( 0 , 0.2 ) ,
    bF ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
  ) , data=d )

precis(milkFat)
```

```
##                   mean         sd       5.5%     94.5%
## a     -1.122893e-05 0.07725495 -0.1234796 0.1234571
## bF     8.619137e-01 0.08426431  0.7272431 0.9965844
## sigma  4.510385e-01 0.05871419  0.3572019 0.5448752
```

Multicolinearity and milk

```
milkLactose <- quap(
  alist(
    K ~ dnorm( mu , sigma ) ,
    mu <- a + bL*L ,
    a ~ dnorm( 0 , 0.2 ) ,
    bL ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
  ) , data=d )


precis(milkLactose)
```

```
##                  mean         sd       5.5%       94.5%
## a       7.923156e-06 0.06661798 -0.1064605  0.1064763
## bL     -9.024329e-01 0.07133080 -1.0164333 -0.7884325
## sigma   3.804758e-01 0.04958609  0.3012277  0.4597240
```

Multicolinearity and milk

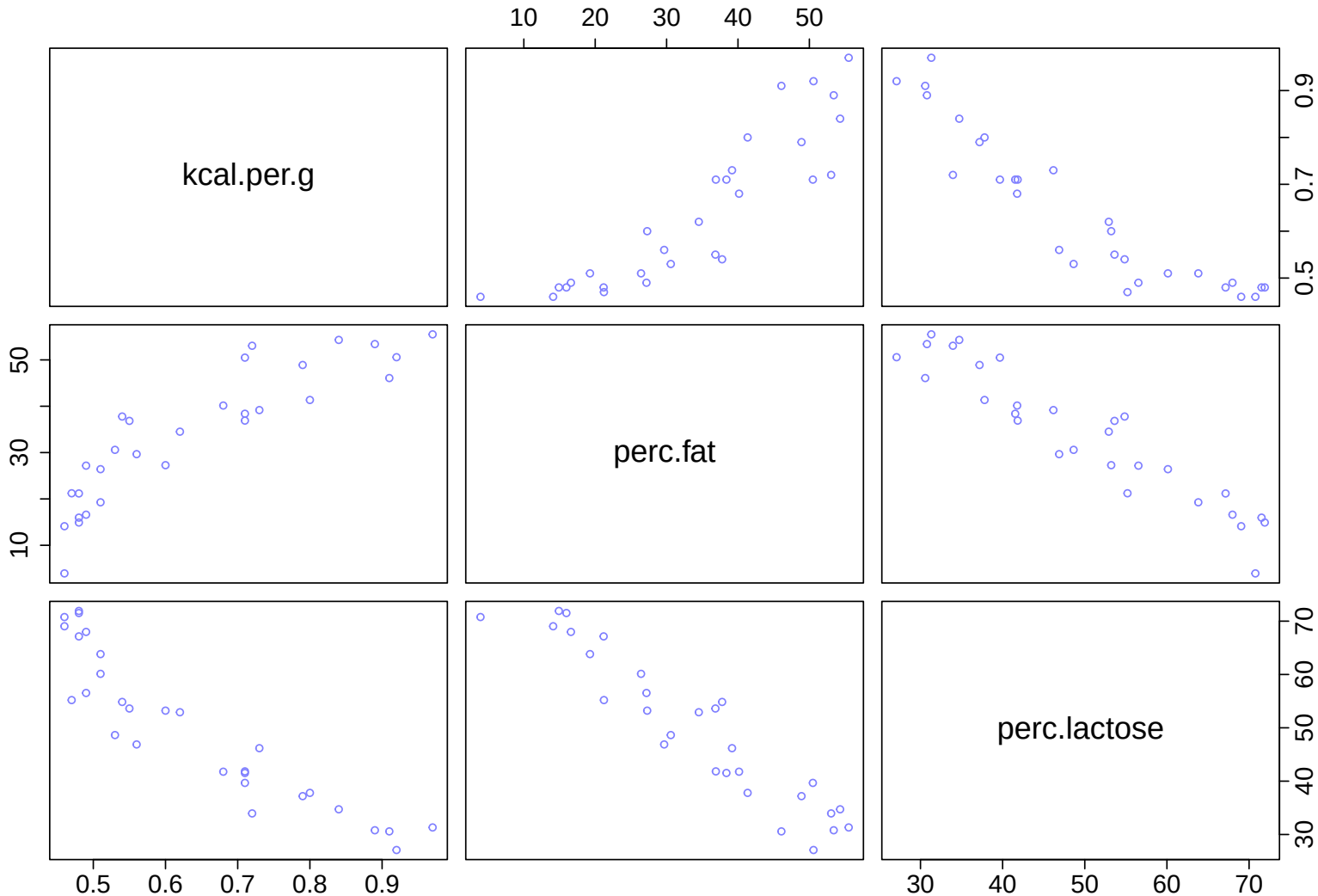
```
milkFatLactose <- quap(
  alist(
    K ~ dnorm( mu , sigma ) ,
    mu <- a + bF*F + bL*L ,
    a ~ dnorm( 0 , 0.2 ) ,
    bF ~ dnorm( 0 , 0.5 ) ,
    bL ~ dnorm( 0 , 0.5 ) ,
    sigma ~ dexp( 1 )
  ) ,
  data=d )


precis(milkFatLactose)
```

```
##                   mean         sd        5.5%       94.5%
## a     -2.864712e-08 0.06603585 -0.10553807  0.1055380
## bF     2.435012e-01 0.18357887 -0.04989332  0.5368957
## bL    -6.780795e-01 0.18377694 -0.97179050 -0.3843684
## sigma  3.767423e-01 0.04918410  0.29813661  0.4553480
```

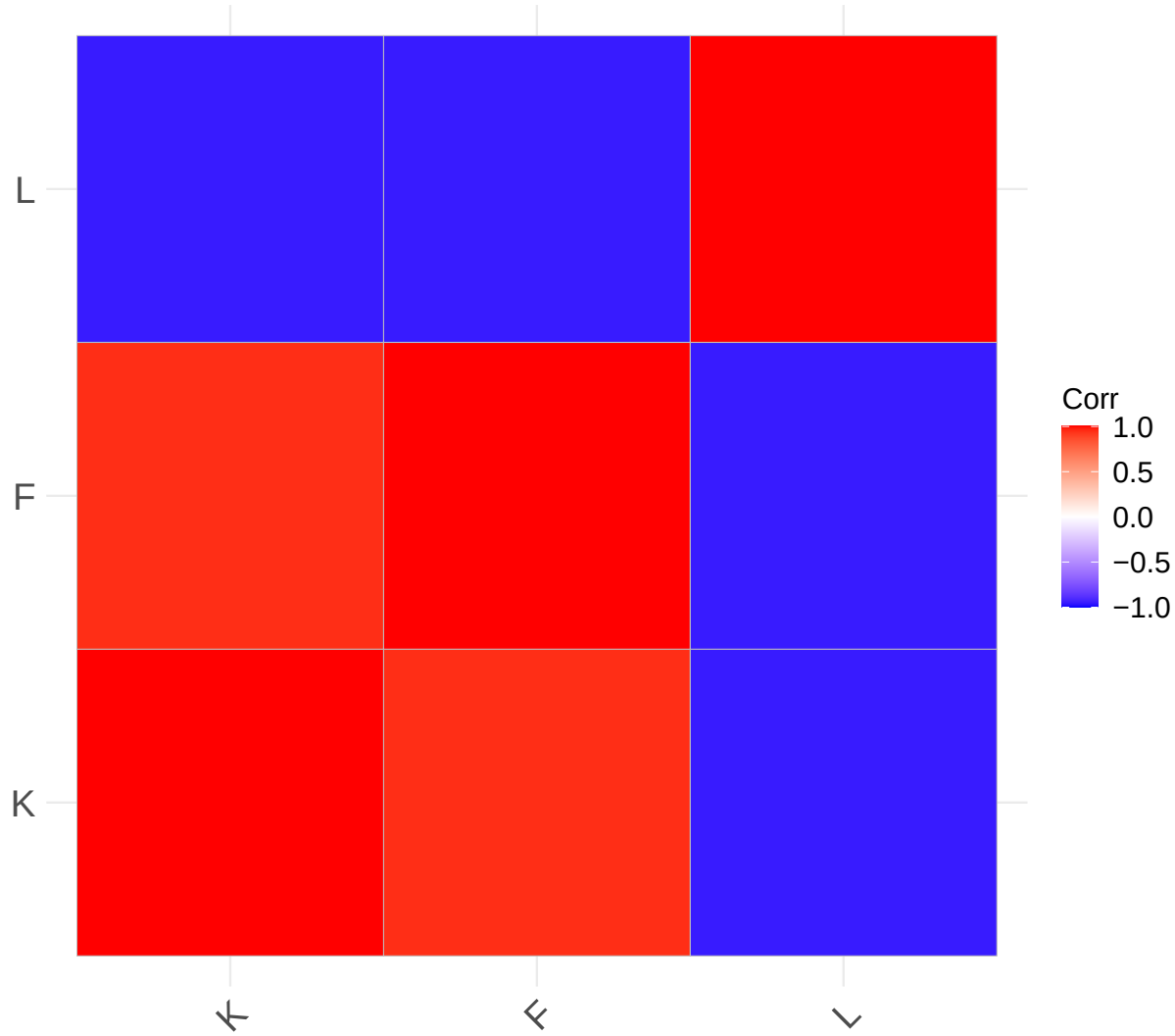
Multicolinearity and milk

```
pairs( ~ kcal.per.g + perc.fat + perc.lactose,
       data=d , col=rangi2, cex.axis = 1.6)
```



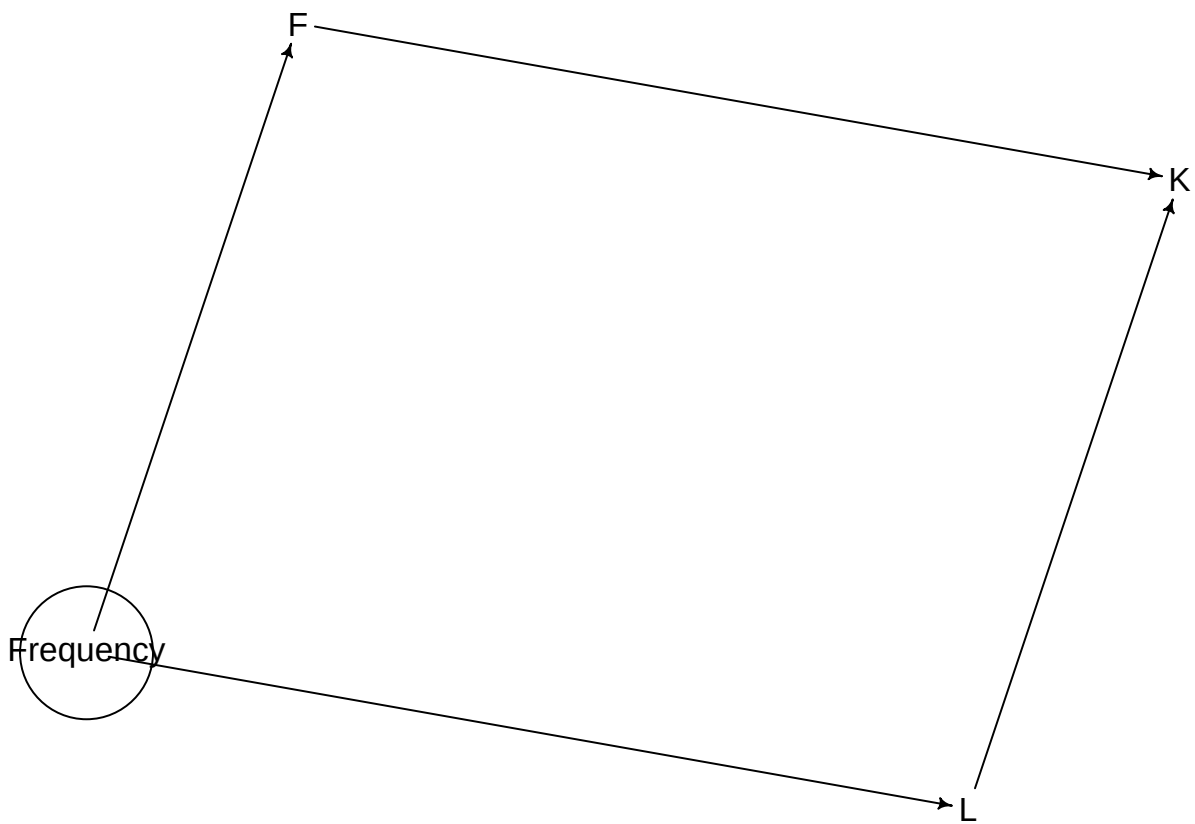
Multicolinearity and milk

```
dcor <- d[,9:11]
milkCor <- cor(dcor, method = "spearman")
ggcorrplot(milkCor) + corSize
```



Multicolinearity and milk

```
milkDAG <- dagitty( "dag {
L <- Frequency ->  F
L -> K <- F
Frequency [unobserved]
                    }")
drawdag(milkDAG, cex= 1.6, radius = 14, goodarrow = TRUE)
```



Confounding

The notion

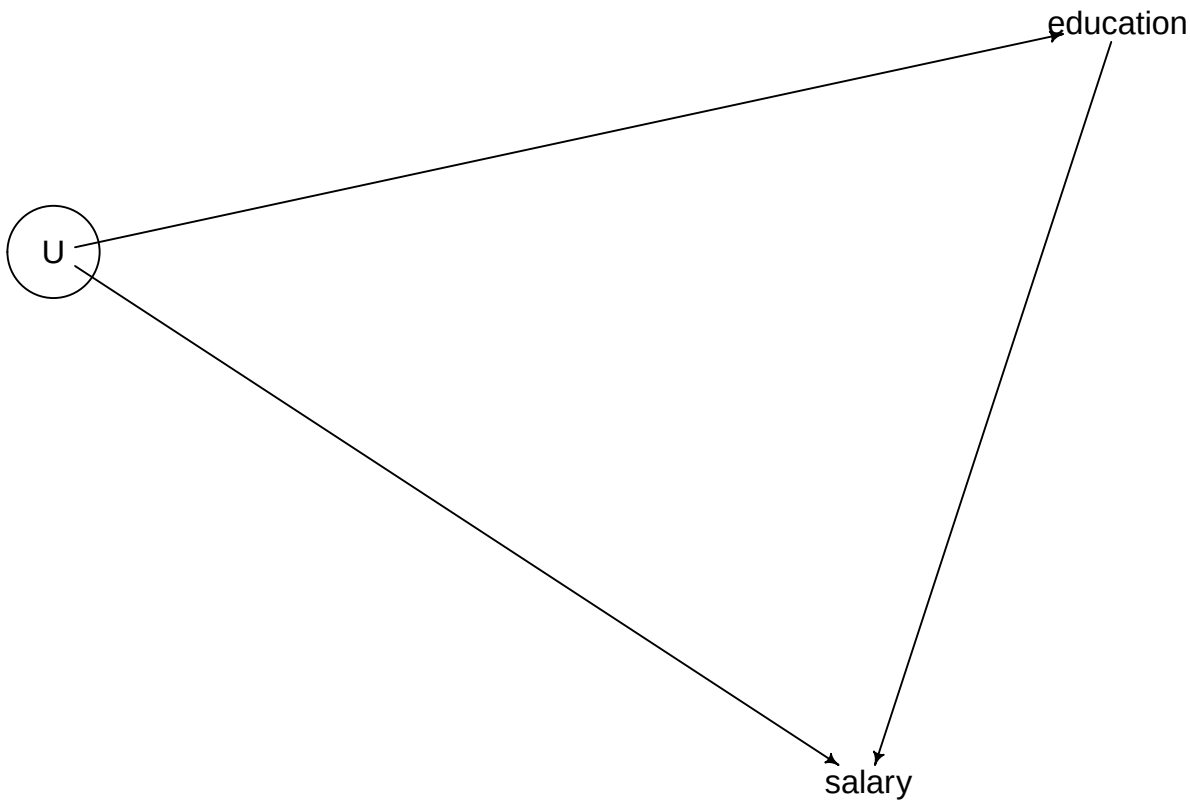
Context in which the association between an outcome and a predictor is not the same as it would be had we experimentally intervened on the predictor.

But when?

- Sometimes, because we didn't condition on a variable.
- Sometimes, because we **did** condition on a variable, too!

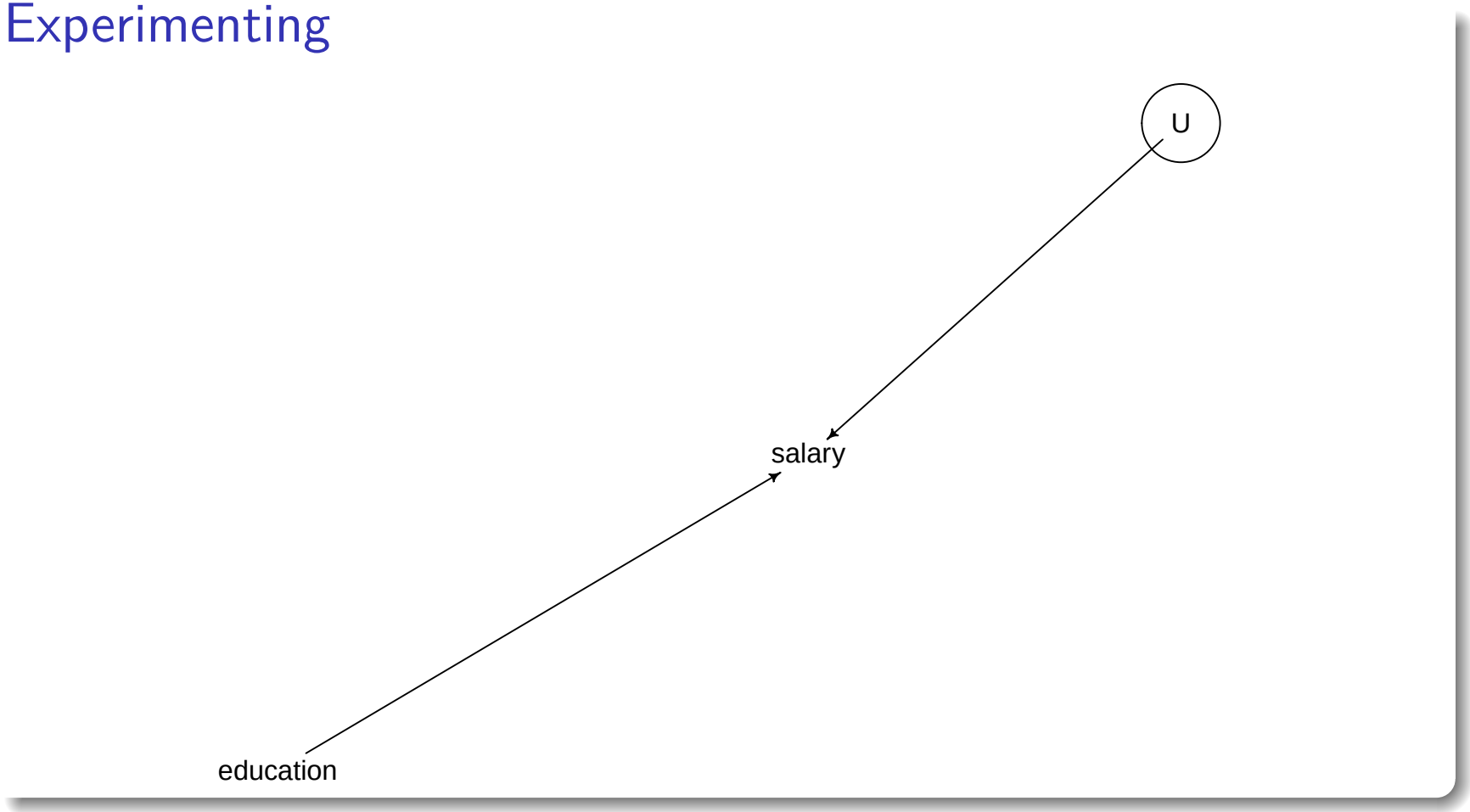
Confounding

Non-causal paths



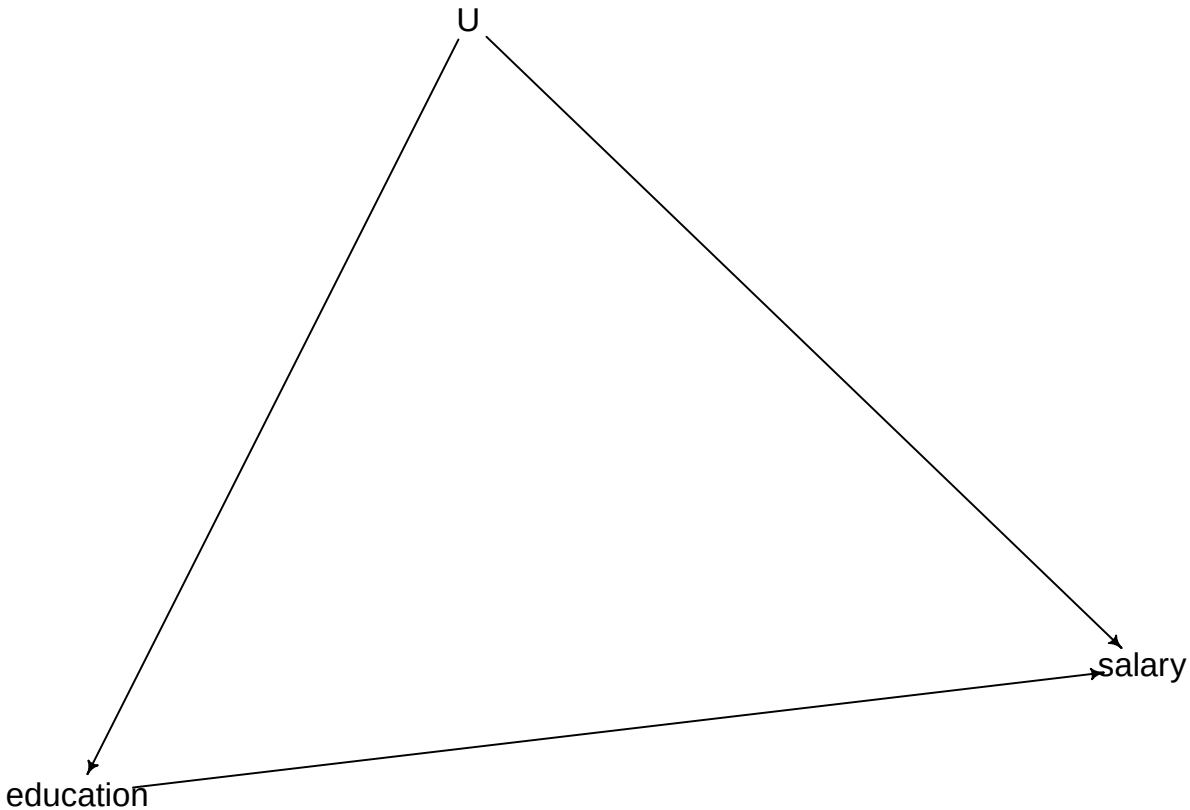
Confounding

Experimenting



Confounding

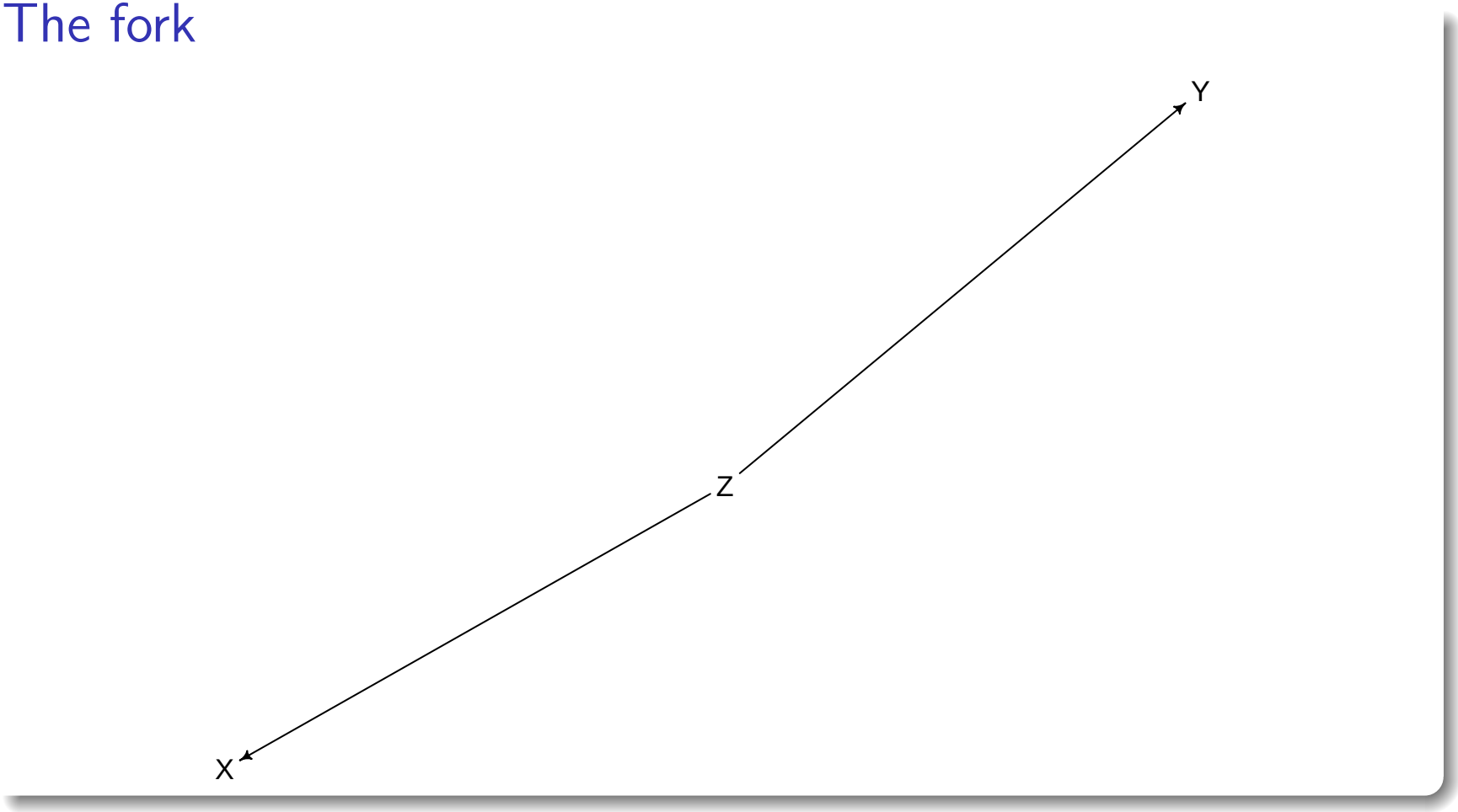
Conditioning



In this case conditioning on U blocks the non-causal path.

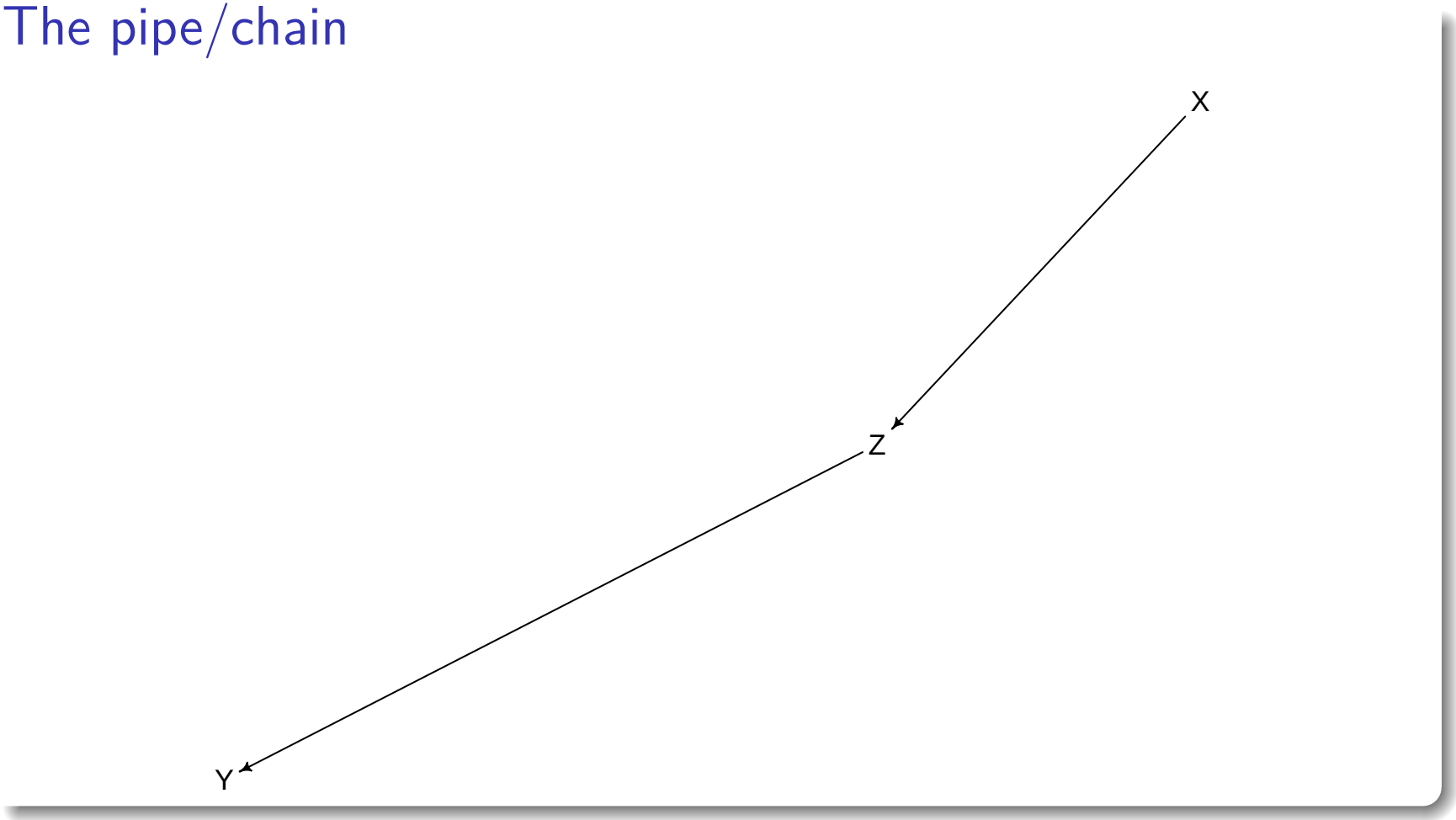
The four horsemen of causal doom

The fork



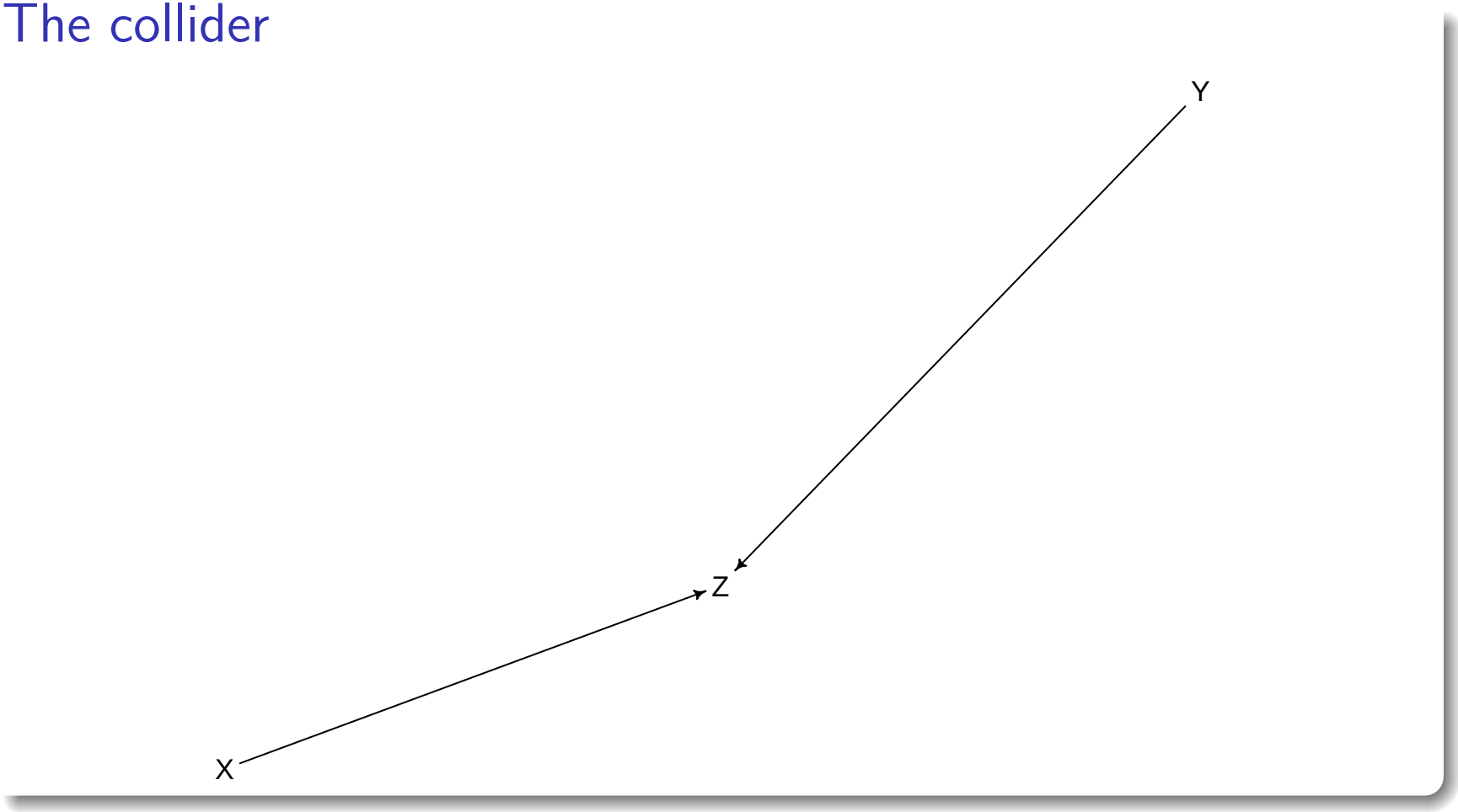
The four horsemen of causal doom

The pipe/chain



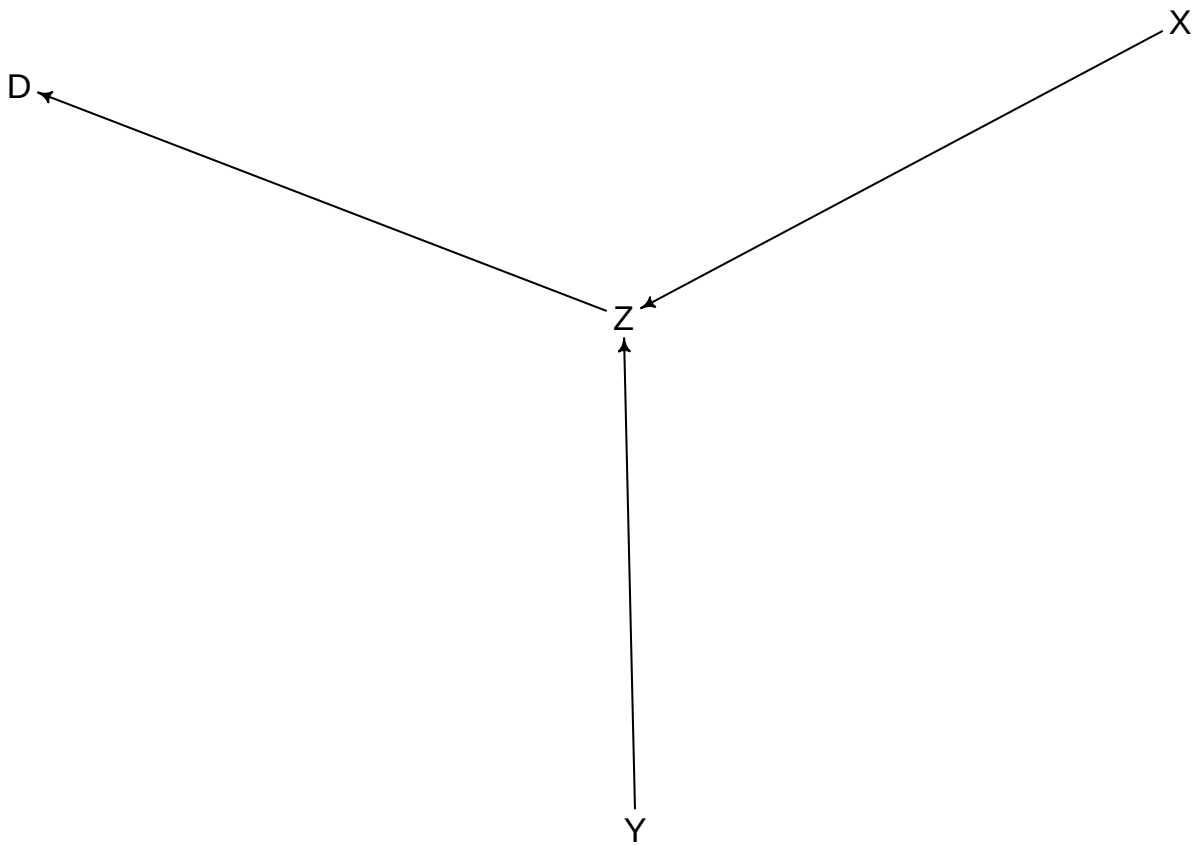
The four horsemen of causal doom

The collider

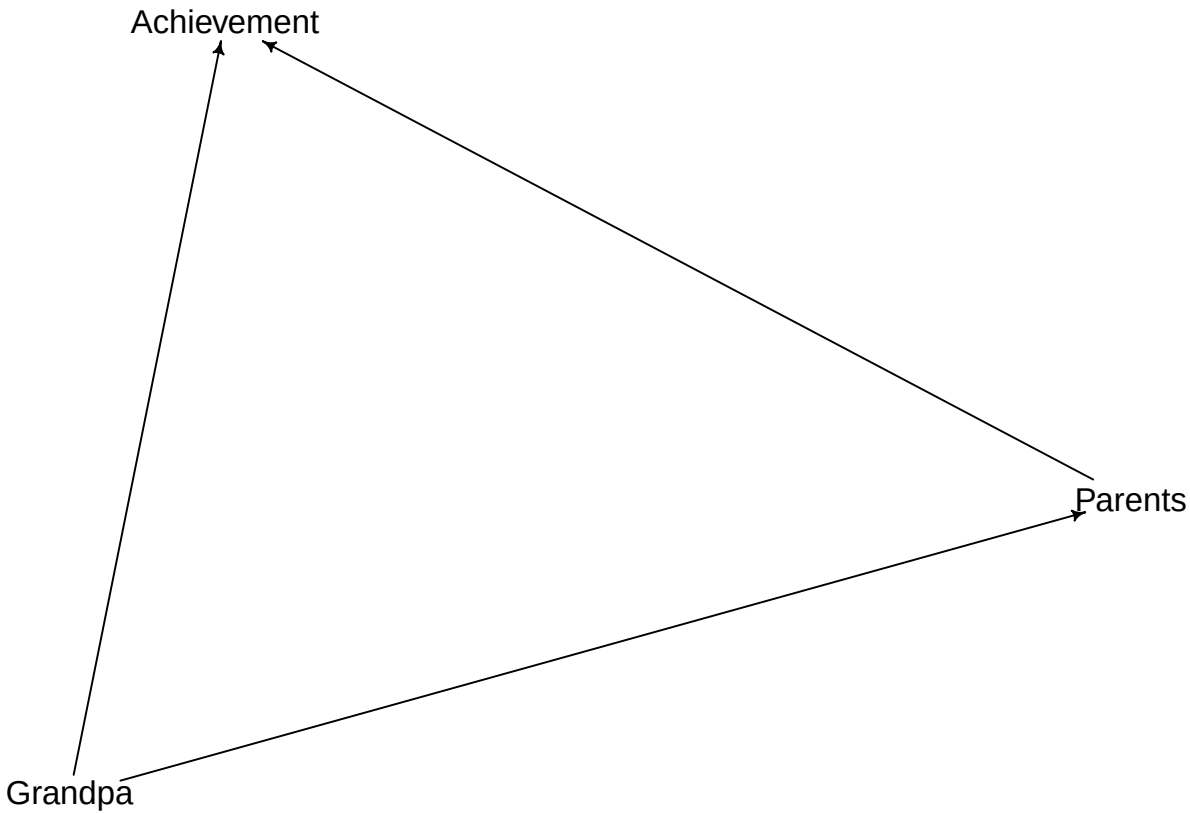


The four horsemen of causal doom

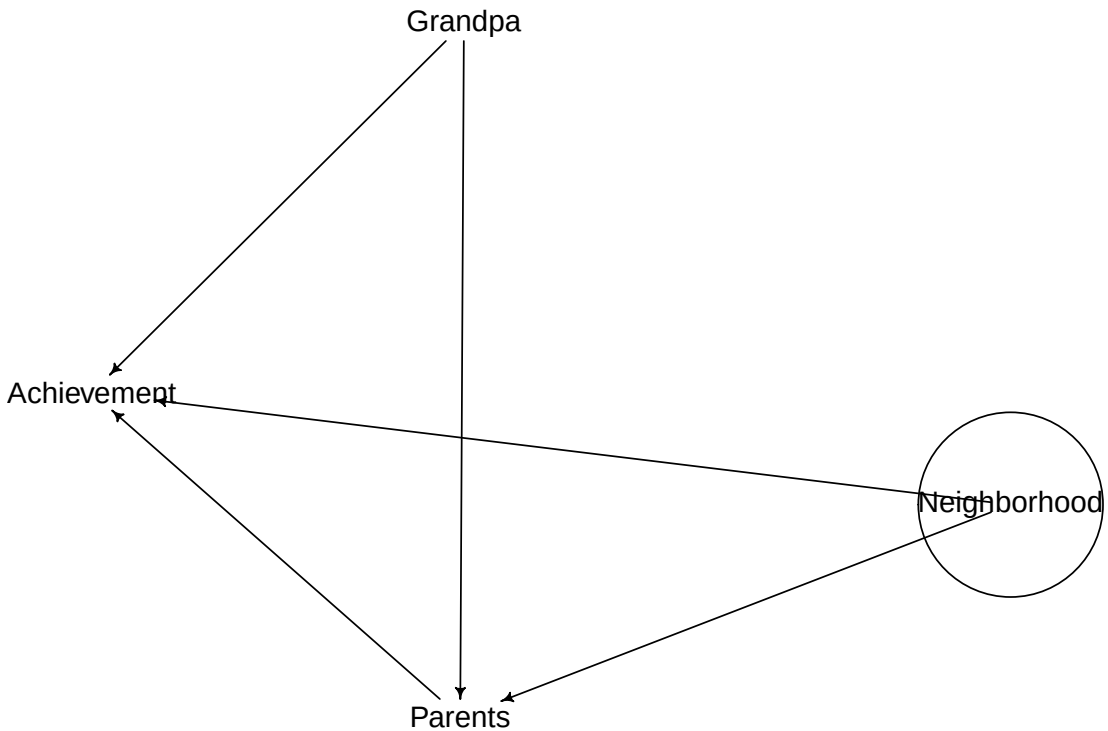
The descendant



DAG haunting



DAG haunting



DAG haunting

```
N <- 200 # number of grandparent-parent-child triads
b_GP <- 1 # direct effect of G on P
b_GC <- 0 # direct effect of G on C
b_PC <- 1 # direct effect of P on C
b_U <- 2 # direct effect of U on P and C

set.seed(666)
U <- 2*rbern( N , 0.5 ) - 1
G <- rnorm( N )
P <- rnorm( N , b_GP*G + b_U*U )
C <- rnorm( N , b_PC*P + b_GC*G + b_U*U )
d <- data.frame( C=C , P=P , G=G , U=U )

head(d)
```

```
##           C         P          G  U
## 1  4.470212  1.484056 -0.9474073  1
## 2 -5.622656 -3.797801 -1.8167107 -1
## 3  6.219381  3.632877  1.9855817  1
## 4 -5.482757 -1.710978  0.8167635 -1
## 5 -6.788545 -4.842863 -0.9996414 -1
## 6  6.473529  2.479559  0.5611222  1
```

DAG haunting

```
hauntingPG <- quap(
  alist(
    C ~ dnorm( mu , sigma ),
    mu <- a + b_PC*P + b_GC*G,
    a ~ dnorm( 0 , 1 ),
    c(b_PC,b_GC) ~ dnorm( 0 , 1 ),
    sigma ~ dexp( 1 )
  ), data=d )

precis(hauntingPG )
```

```
##                mean         sd       5.5%      94.5%
## a      -0.02654038 0.09093145 -0.1718664  0.1187856
## b_PC    1.85667685 0.04085572  1.7913815  1.9219722
## b_GC   -0.78084187 0.09626299 -0.9346887 -0.6269950
## sigma   1.28608697 0.06400163  1.1838000  1.3883739
```

DAG haunting

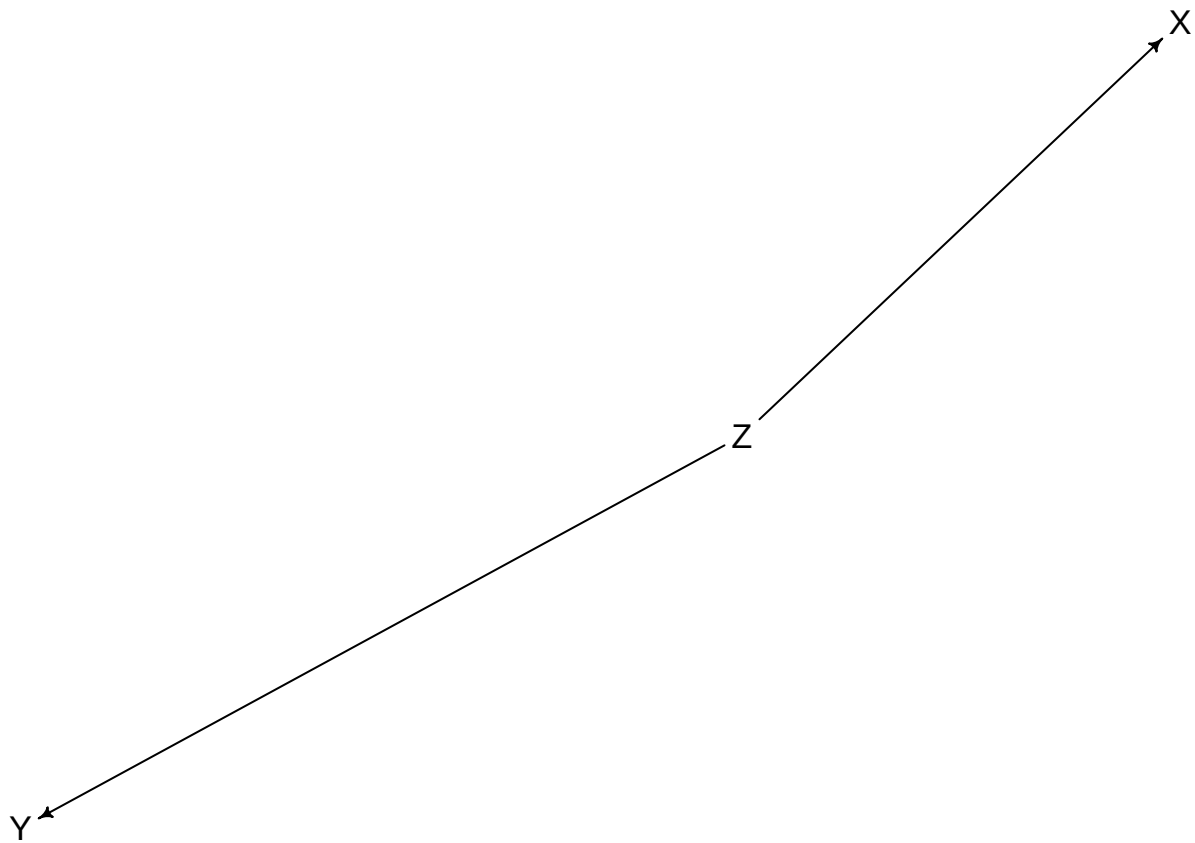
```
hauntingPGU <- quap(
  alist(
    C ~ dnorm( mu , sigma ),
    mu <- a + b_PC*P + b_GC*G + b_U*U,
    a ~ dnorm( 0 , 1 ),
    c(b_PC,b_GC,b_U) ~ dnorm( 0 , 1 ),
    sigma ~ dexp( 1 )
  ), data=d )


precis(hauntingPGU)
```

```
##                mean         sd        5.5%     94.5%
## a      -0.004607422 0.06668077 -0.11117618 0.1019613
## b_PC    1.023325948 0.07010848  0.91127905 1.1353728
## b_GC    0.215707902 0.10373426  0.04992052 0.3814953
## b_U     2.052574709 0.15577245  1.80362025 2.3015292
## sigma   0.940962614 0.04690247  0.86600340 1.0159218
```

Shut the backdoor

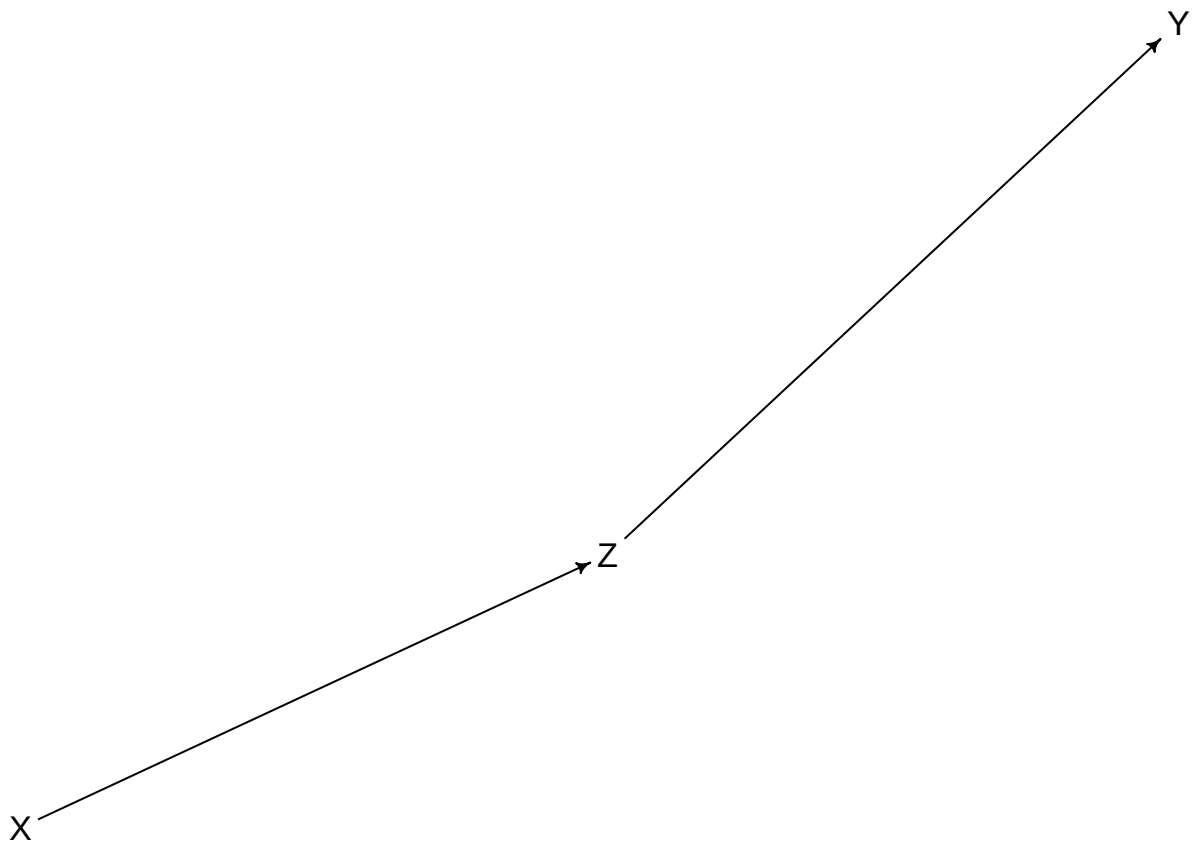
The fork



$I(X, Y|Z)$ (think ice-cream)

Shut the backdoor

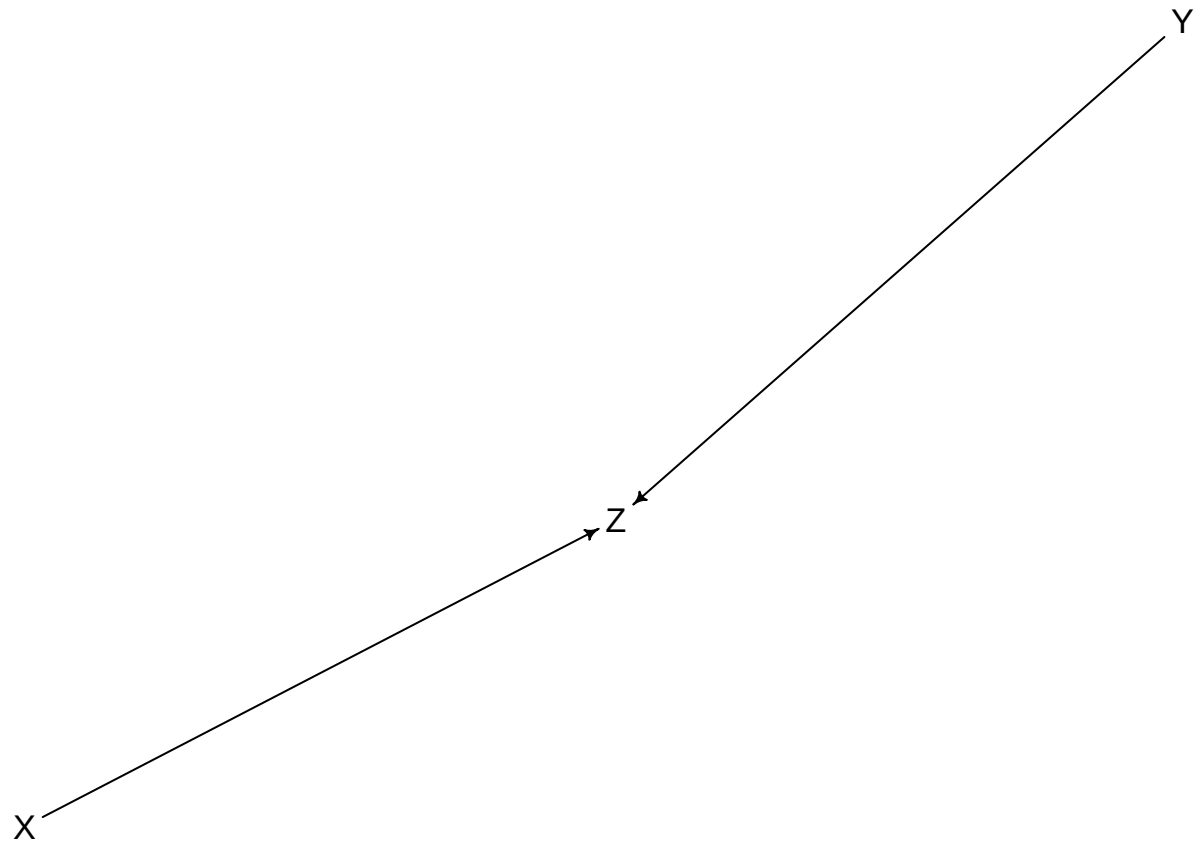
The pipe/chain



$I(X, Y|Z)$ (think fungi)

Shut the backdoor

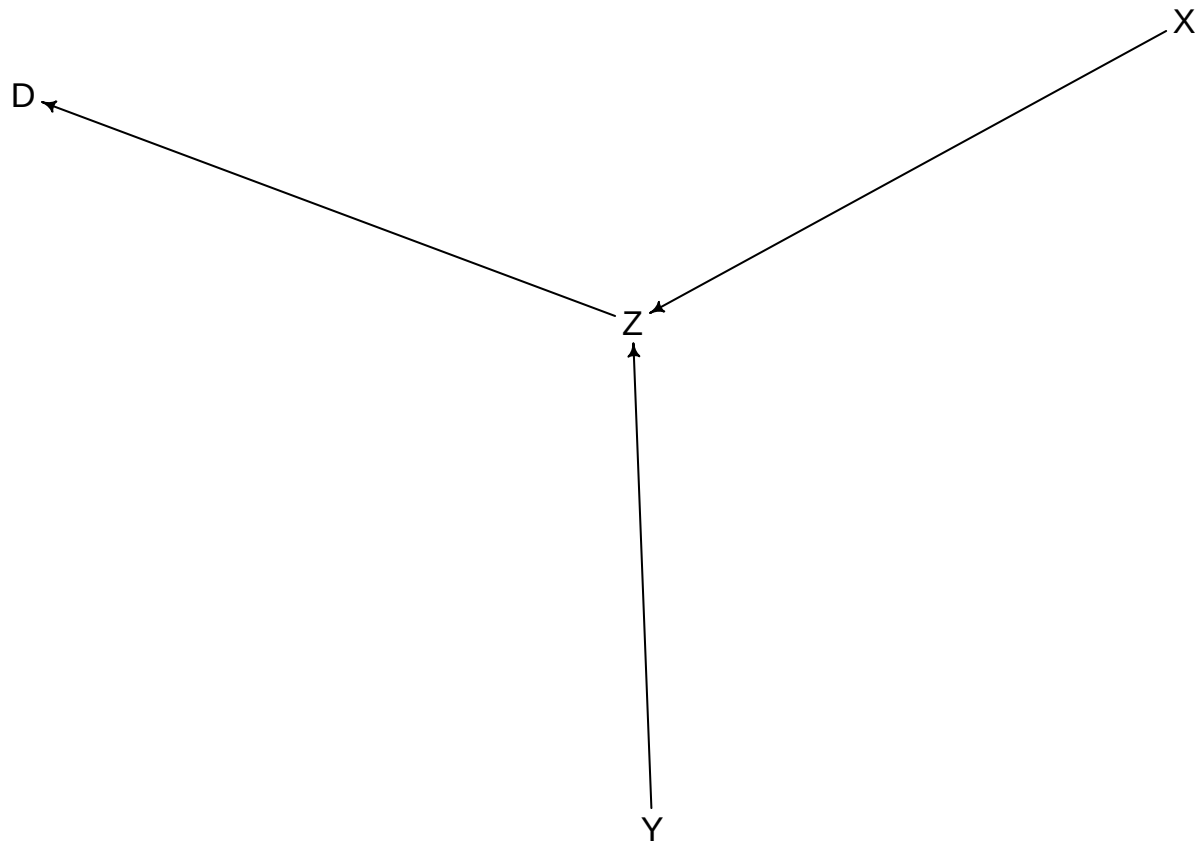
The collider



$\neg I(X, Y|Z)$ (think parents and neighborhood)

Shut the backdoor

The descendant



$\neg I(X, Y|D)$ (you learn something about Z as well)

Shut the backdoor

Steps

1. List all of the paths.
2. Classify each path as open or closed. A path is open unless it contains a collider.
3. Classify each path as a backdoor path or not. A backdoor path has an arrow entering X.
4. If any backdoor is open, close it by conditioning.

Dagittize

Further examples: http://dagitty.net/primer/

```
paths(grandDAG, from = "Grandpa", to = "Achievement")
```

```
## $paths
## [1] "Grandpa -> Achievement"            "Grandpa -> Parents -> Achievement"
##
## $open
## [1] TRUE TRUE
```

```
paths(grandDAG2, from = "Grandpa", to = "Achievement")
```

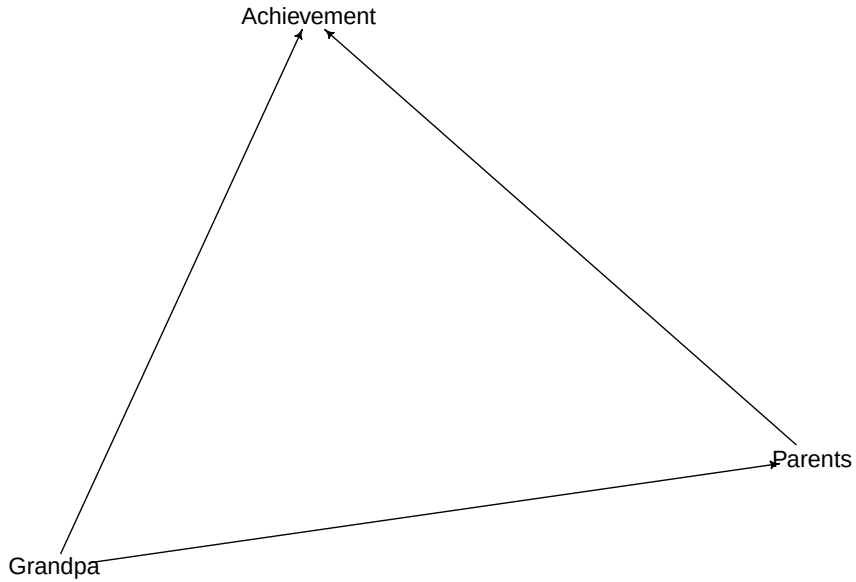
```
## $paths
## [1] "Grandpa -> Achievement"
## [2] "Grandpa -> Parents -> Achievement"
## [3] "Grandpa -> Parents <- Neighborhood -> Achievement"
##
## $open
## [1]  TRUE  TRUE FALSE
```

```
paths(grandDAG2, from = "Grandpa", to = "Achievement", Z = "Parents")
```

```
## $paths
## [1] "Grandpa -> Achievement"
## [2] "Grandpa -> Parents -> Achievement"
## [3] "Grandpa -> Parents <- Neighborhood -> Achievement"
##
## $open
## [1]  TRUE FALSE  TRUE
```

Dagittize

```
drawdag(grandDAG, cex= 1.6, goodarrow = TRUE, radius = 19)
```




```
adjustmentSets(grandDAG, exposure = "Grandpa", outcome = "Achievement", effect = "total")
```

```
##  {}
```

```
adjustmentSets(grandDAG, exposure = "Grandpa", outcome = "Achievement", effect = "direct")
```

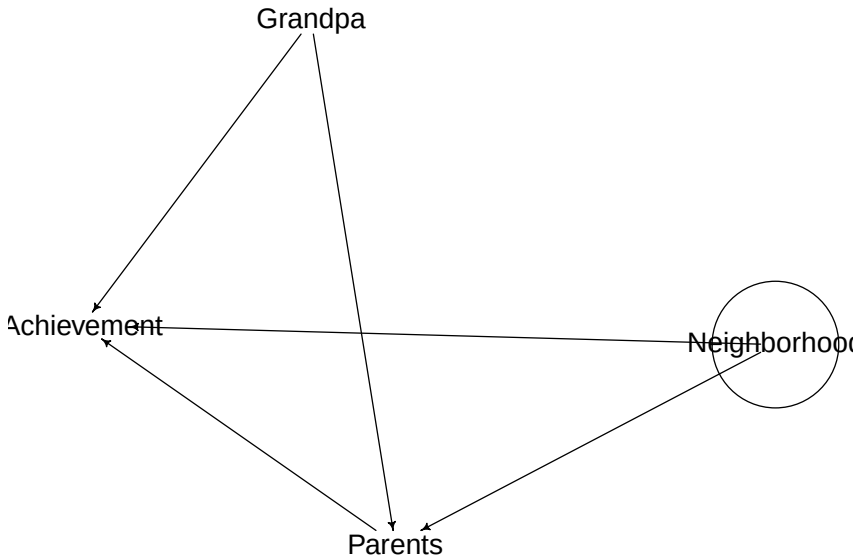
```
## { Parents }
```

```
adjustmentSets(grandDAG, exposure = "Parents", outcome = "Achievement", effect = "direct")
```

```
## { Grandpa }
```

Dagittize

```
drawdag(grandDAG2, cex= 2, goodarrow = TRUE, radius = 20)
```




```
adjustmentSets(grandDAG2, exposure = "Grandpa",
               outcome = "Achievement",  effect = "total")
```

```
##  {}
```

```
adjustmentSets(grandDAG2, exposure = "Grandpa",
               outcome = "Achievement",  effect = "direct")
adjustmentSets(grandDAG2, exposure = "Parents",
               outcome = "Achievement", effect = "total")
adjustmentSets(grandDAG2, exposure = "Parents",
               outcome = "Achievement", effect = "direct")
```

d-separation

Test implications

1. $dsep(X, Y|Z) \Rightarrow I(X, Y|Z)$
2. $I(X, Y|Z) \not\Rightarrow dsep(X, Y|Z)$
3. Z can be empty.

```
impliedConditionalIndependencies(
  forkDAG )
```

```
## X _||_ Y | Z
```

```
impliedConditionalIndependencies(pipeDAG)
```

```
## X _||_ Y | Z
```

```
impliedConditionalIndependencies(colliderDAG, type = "all.pairs")
```

```
## X _||_ Y
```

d-separation

Definition

If G is a directed graph in which X, Y and Z are disjoint sets of vertices, then X and Y are d-connected by Z in G if and only if there exists an undirected path p between some vertex in X and some vertex in Y such that for every collider C on p, either $C4$ or a descendent of C is in Z, and no non-collider on p is in Z. X and Y are d-separated by Z in G if and only if they are not d-connected by Z in G.

Properties

1. $dsep(X, Y|Z) \Rightarrow I(X, Y|Z)$
2. $I(X, Y|Z) \not\Rightarrow dsep(X, Y|Z)$
3. Z can be empty.