

4. 回帰分析

honocat

2026-01-05

ロジットの逆関数(ロジスティック関数)を用意する。

```
inv_logit <- function(x) {  
  return(1 / (1 + exp(-x)))  
}
```

回帰分析のシミュレーション

単回帰

■(1) 処置変数が二値の場合

処置変数 $D_i \in \{0, 1\}$ と結果変数 $Y_i \in \mathbb{R}$ を考える。 Y_i が D_i の関数で、

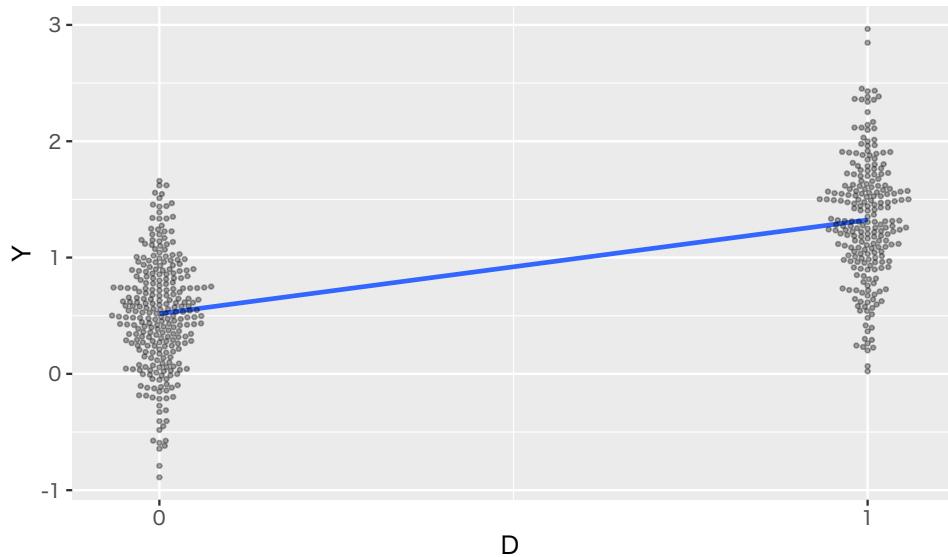
$$Y_i = 0.5 + 0.8D_i + e_i$$

と表されるとする。この状況をシミュレートする。

```
set.seed(12345)  
N <- 500  
D <- rbinom(N, size = 1, prob = 0.4)  
Y <- 0.5 + 0.8 * D + rnorm(N, mean = 0, sd = 0.5)  
df_simple <- tibble(Y, D)
```

D と Y の関係を可視化する。

```
scat_simple <- ggplot(df_simple, aes(x = D, y = Y)) +  
  geom_smooth(method = 'lm', se = FALSE) +  
  geom_beeswarm(shape = 20, alpha = 1 / 3, size = 1) +  
  scale_x_continuous(breaks = 0 : 1)  
plot(scat_simple)
```



回帰直線の傾きは、

```
fit_simple <- lm(Y ~ D, data = df_simple)
coef(fit_simple)[2]
```

```
D
0.8061476
```

である。これは、処置群($D = 1$ の群)の Y の平均値と、統制群($D = 0$)の群の Y の平均値の差である。

```
with(df_simple,
      mean(Y[D == 1]) - mean(Y[D == 0]))
```

```
[1] 0.8061476
```

このように、回帰直線の傾きは、説明変数の値で条件付けた結果変数の期待値の差である。また、回帰直線の切片は、

```
coef(fit_simple)[1]
```

```
(Intercept)
0.5173247
```

であり、これは統制群($D = 0$ の群)の Y の平均値、

```
with(df_simple, mean(Y[D == 0]))
```

```
[1] 0.5173247
```

に等しい。

■(2) 処置変数が連続の場合

処置変数 $D_i \sim \text{Uniform}(-2, 2)$ と結果変数 $Y_i \in \mathbb{R}$ を考える。 Y_i が D_i の関数で、

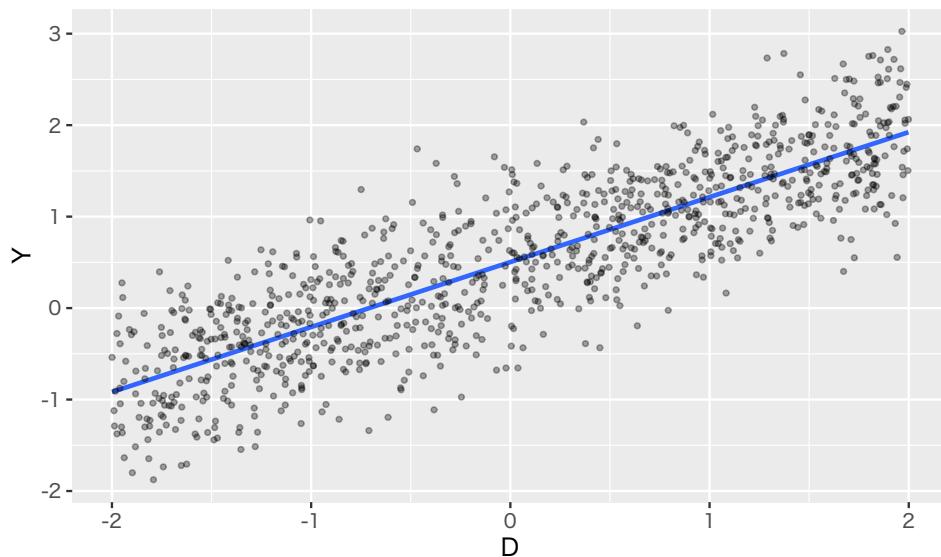
$$Y_i = 0.5 + 0.7D_i + e_i$$

と表されるとする。この状況をシミュレートする。

```
set.seed(321)
N <- 1000
D <- runif(N, min = -2, max = 2)
Y <- 0.5 + 0.7 * D + rnorm(N, mean = 0, sd = 0.5)
df_cont <- tibble(Y, D)
```

D と Y の関係を可視化する。

```
scat_cont <- ggplot(df_cont, aes(x = D, y = Y)) +
  geom_smooth(method = 'lm', se = FALSE) +
  geom_point(shape = 20, alpha = 1 / 3)
plot(scat_cont)
```



回帰直線の傾きは、

```
fit_cont <- lm(Y ~ D, data = df_cont)
coef(fit_cont)[2]
```

```
D  
0.710264
```

である。これは、処置の値を 1 単位増やしたとき、平均すると Y の平均値がどれだけ増加するかを示している。

観測された個体を D の値によって $[-2, 1), [-1, 0), [0, 1), [1, 2]$ の 4 つのグループに分け、それぞれのグループで Y の平均値を計算し、さらに隣のグループとの差を取る。

```
df_cont <- df_cont |>
  mutate(group = case_when(
    D < -1 ~ 1,
    D < 0 ~ 2,
    D < 1 ~ 3,
    TRUE ~ 4
  ))
(b1 <- with(df_cont, mean(Y[group == 2]) - mean(Y[group == 1])))
```

```
[1] 0.6512046
```

```
(b2 <- with(df_cont, mean(Y[group == 3]) - mean(Y[group == 2])))
```

```
[1] 0.7395212
```

```
(b3 <- with(df_cont, mean(Y[group == 4]) - mean(Y[group == 3])))
```

```
[1] 0.7123079
```

説明変数である D の値が 1 増えるごとに、回帰係数に概ね近い値の数だけ、 Y の平均値が増加していることがわかる。それぞれのグループに属する観測値の数を重みにして加重平均をとると、

```
n_groups <- table(df_cont$group)[c(1, 2, 2, 3, 3, 4)]
weight <- c(sum(n_groups[1 : 2]),
           sum(n_groups[3 : 4]),
           sum(n_groups[5 : 6]))
weighted.mean(c(b1, b2, b3), w = weight)
```

```
[1] 0.7016254
```

となり、傾きの推定値に近い値が得られる。

このように、回帰直線の傾きは、説明変数の値で条件付けた結果変数の期待値の差である。また、回帰直線の切片は、

```
coef(fit_cont)[1]
```

```
(Intercept)
```

```
0.5032906
```

であり、これは $D = 0$ のときの Y の期待値の推定値である。シミュレーションデータで、 D が 0 に近い値の Y の平均値を計算すると、

```
epsilon <- 0.2
with(df_cont, mean(Y[D > - epsilon & D < epsilon]))
```

```
[1] 0.4909429
```

となり、推定値に概ね一致する。

重回帰

処置変数 $D_i \in \{0, 1\}$ と結果変数 $Y_i \in \mathbb{R}$ に加え、共変量 $X \in \{-4, -3, \dots, 4\}$ を考える。 Y_i が D_i と X_i の関係で、

$$Y_i = 0.5 + 0.7D_i + 0.3X_i + e_i$$

と表されるとする。また、 D_i の値は X_i に依存して、次のように決まることにする。

$$\begin{aligned} D_i &\sim \text{Bernoulli}(\theta_i) \\ \theta_i &= \text{logit}^{-1}(0.5X_i) \end{aligned}$$

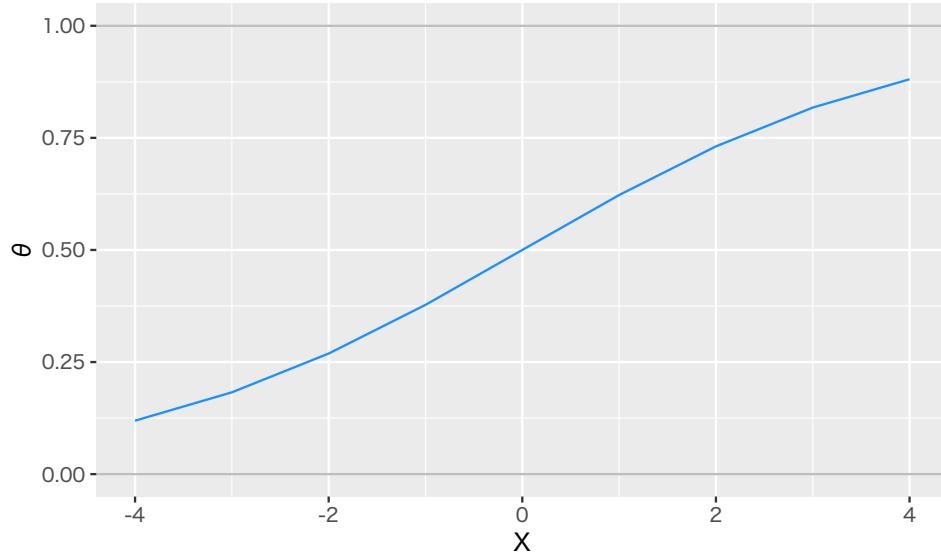
つまり、 X が大きくなるほど、 D_i が 1 になる確率が大きくなると仮定する。 θ と X の関係を可視化すると、以下のようになる。

```
df <- tibble(X = -4 : 4) |>
  mutate(theta = inv_logit(0.5 * X))
p_logistic <- ggplot(df, aes(x = X, y = theta)) +
  geom_hline(yintercept = c(0, 1), color = 'gray') +
  geom_line(color = 'dodgerblue') +
```

```

  labs(y = expression(theta))
plot(p_logistic)

```



この状況をシミュレートする。

```

set.seed(111)
N <- 1e3
X <- sample(-4 : 4, size = N, replace = TRUE)
theta <- inv_logit(0.5 * X)
D <- rbinom(N, size = 1, prob = theta)
Y <- 0.5 + 0.7 * D + 0.3 * X + rnorm(N, mean = 0, sd = 0.5)
df_multiple <- tibble(Y, D, X)

```

Y を D と X に回帰する(long regression)と、 D の係数の推定値は、

```

fit_long <- lm(Y ~ D + X, data = df_multiple)
coef(fit_long)[2]

```

```

D
0.6905941

```

となる。 Y を D のみに回帰する(short regression)と、 D の推定値は、

```

fit_short <- lm(Y ~ D, data = df_multiple)
coef(fit_short)[2]

```

```
D  
1.516601
```

となり、推定値が過大推定されている。この値は、観測された平均値の 2 群間の差である。

```
with(df_multiple, mean(Y[D == 1]) - mean(Y[D == 0]))
```

```
[1] 1.516601
```

重回帰によるブロッキング

重回帰を行う代わりに、`df_multiple` を共変量 X の値ごとにブロックに分け、単回帰によってブロックごとに D の係数を推定してみる。

$X = x$ のブロックで傾きを推定する関数を用意する。

```
delta_X <- function(x, data) {  
  fit <- data |>  
  filter(X == x) |>  
  lm(Y ~ D, data = _)  
  return(coef(fit)[2])  
}
```

X の値ごとに、傾きを推定する。

```
delta_vec <- sapply(-4 : 4, delta_X, data = df_multiple)  
names(delta_vec) <- str_c('X = ', -4 : 4)  
delta_vec
```

```
X = -4      X = -3      X = -2      X = -1      X = 0      X = 1      X = 2      X = 3  
0.8446962  0.6572588  0.7475040  0.6917050  0.6252307  0.7681240  0.5260543  0.7046942  
X = 4  
0.6230728
```

ブロックごとに処置効果を求めることができた。このように、ブロックに分けて単回帰を行えば、必ずしも過大推定にはならないことがわかる。これらの値を元に、全体の処置効果を加重平均で求めてみる。

ATE を求めるために $X = x$ となるブロックの重みを $\Pr(X_i = x)$ とすると、

```
w_ate <- table(df_multiple$X)  
weighted.mean(delta_vec, w = w_ate)
```

```
[1] 0.687105
```

となる。重回帰によって求めた推定値との差は、

```
coef(fit_long)[2] - weighted.mean(delta_vec, w = w_ate)
```

```
D  
0.003489067
```

である。2つの推定方法にはあまり違いが無いようである。

同様に、ATT（処置群における平均処置効果）を求めるために、 $X = x$ となるブロックの重みを $\Pr(X_i = x | D_i = 1)$ とすると、

```
df_treated <- df_multiple |>  
  filter(D == 1)  
w_att <- table(df_treated$X)  
weighted.mean(delta_vec, w = w_att)
```

```
[1] 0.6648775
```

となる。重回帰によって、求めた推定値との差は、

```
coef(fit_long)[2] - weighted.mean(delta_vec, w = w_att)
```

```
D  
0.02571656
```

となる。

重回帰の推定値とブロックごとの単回帰から求めた ATE の推定値の差を取るシミュレーションを繰り返してみる。

まずは、2つの推定値の差を1回計算するための関数を作る。

```
sim_reg_block <- function(N = 1e3,  
                           delta = 0.7, beta = 0.3, lambda = 0.5) {  
  ## delta : D の因果効果  
  ## beta   : X と Y の関係の強さ  
  ## lambda: X と D の関係の強さ  
  
  X <- sample(-4 : 4, size = N, replace = TRUE)}
```

```

theta <- inv_logit(lambda * X)
D <- rbinom(N, size = 1, prob = theta)
Y <- 0.5 + delta * D + beta * X + rnorm(N, mean = 0, sd = 0.5)
df <- tibble(Y, D, X)
fit <- lm(Y ~ D + X)
delta_vec <- sapply(-4 : 4, delta_X, data = df)
weight <- table(X)
dif <- coef(fit)[2] - weighted.mean(delta_vec, w = weight)
names(dif) <- 'difference'
return(dif)
}

```

この関数を一度使ってみる。

```
sim_reg_block()
```

```

difference
-0.02281703

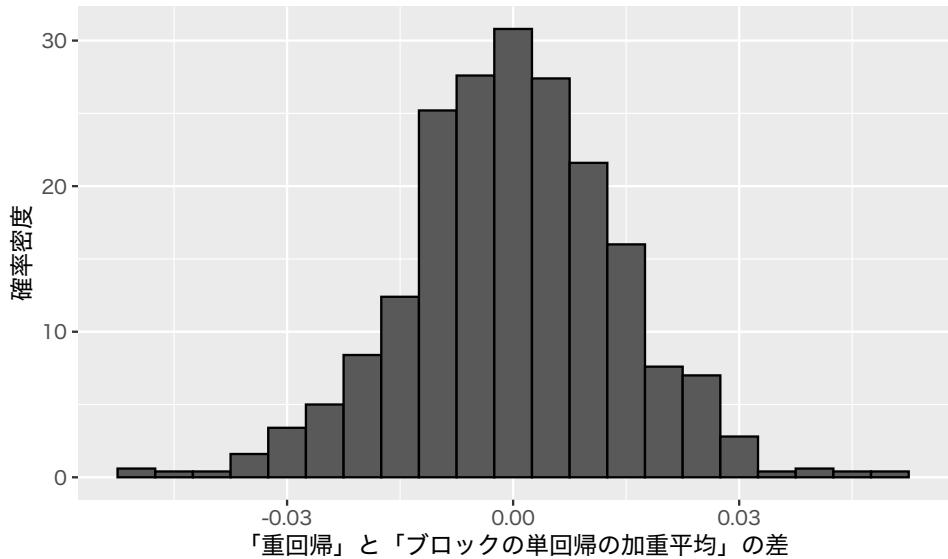
```

1,000 回繰り返して結果を可視化する。

```

sim_rb <- replicate(1e3, sim_reg_block())
hist_sim_rb <- tibble(difference = sim_rb) |>
  ggplot(aes(x = difference, y = after_stat(density))) +
  geom_histogram(binwidth = 0.005, color = 'black') +
  labs(x = '「重回帰」と「ブロックの単回帰の加重平均」の差',
       y = '確率密度')
plot(hist_sim_rb)

```



5% タイルと 95% タイルを取ると、

```
quantile(sim_rb, p = c(0.05, 0.95))
```

5%	95%
-0.02336985	0.02359641

となる。つまり、シミュレーションの値のうち 90% が、[-0.023, 0.024] の間にある。推定しようとしている値は 0.70 なので、2 つの推定方法の間には平均的には差がないと考えて良さそうだ。

このように重回帰の結果は、ブロックごとに単回帰してその加重平均を求めた場合とほぼ同じ結果になることがわかる。

重回帰分析によるセレクションバイアスの除去

トピック 2 で考えた、病院と健康状態の例をもう一度考える。

トピック 2 では、以下の関数でセレクションバイアスのシミュレーションをした(一部書き換え)。

```
sim_hospital <- function(delta = 0.6,
                           N      = 1e4,
                           mu     = c(0.75, 0.6, 0.4, 0.3, 0.2),
                           sigma = rep(0.1, 5)) {
  # sigma は健康状態ごとに変えてもいいことにする
  if (length(sigma) == 1) sigma <- rep(sigma, 5)
  h_hidden <- sample(1 : 5, size = N, replace = TRUE,
                     prob = c(1, 2, 3, 2, 1))
```

```

prob <- rnorm(N, mean = mu[h_hidden], sd = sigma[h_hidden])
prob <- case_when(
  prob > 1 ~ 1,
  prob < 0 ~ 0,
  TRUE ~ prob
)
D <- rbinom(N, size = 1, prob = prob)
Y <- h_hidden + delta * D + rnorm(N)
fit <- lm(Y ~ D)
return(coef(fit)[2])
}

```

このシミュレーションを 1,000 回実行する。

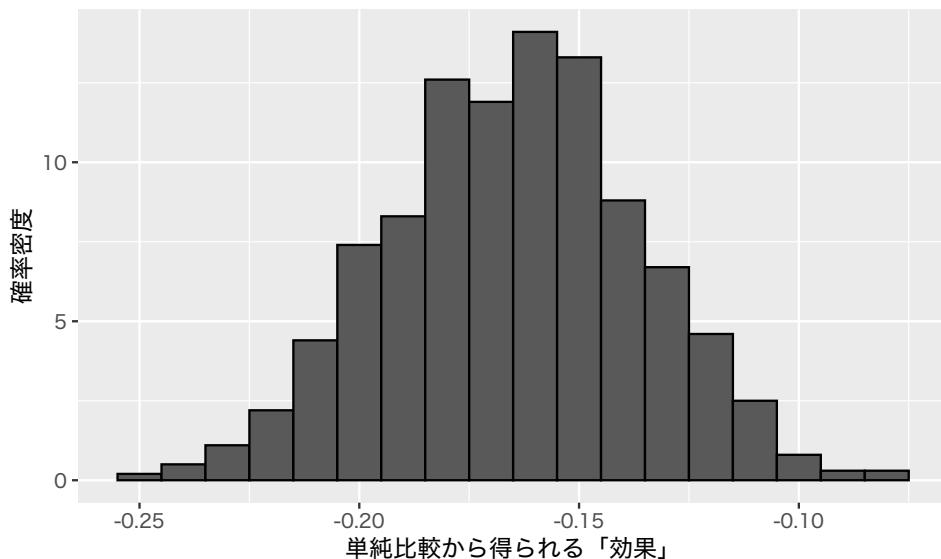
```
sb1 <- replicate(1e3, sim_hospital())
```

この結果を図示する。

```

hist_sb1 <- tibble(delta = sb1) |>
  ggplot(aes(x = delta, y = after_stat(density))) +
  geom_histogram(color = 'black', binwidth = 0.01) +
  labs(x = '単純比較から得られる「効果」', y = '確率密度')
plot(hist_sb1)

```



処置効果を 0.6 に設定しているのに、セルフセレクションのせいで効果が過小推定されている。問題は、セルフセレクションに使われるもとの健康状態 `h_hidden` が未観測であることだった。

`h_hidden` が観測されていれば、重回帰によってセレクションバイアスを取り除くことができる。関数を書き換えてみる。

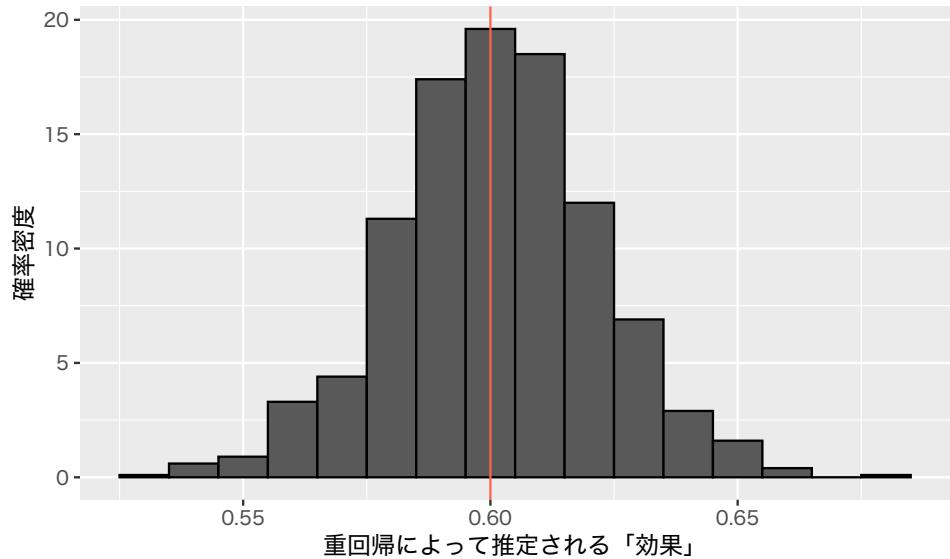
```
sim_hospital2 <- function(delta = 0.6,
                           N      = 1e4,
                           mu    = c(0.75, 0.6, 0.4, 0.3, 0.2),
                           sigma = rep(0.1, 5)) {
  if (length(sigma) == 1) sigma <- rep(sigma, 5)
  h_hidden <- sample(1 : 5, size = N, replace = TRUE,
                     prob = c(1, 2, 3, 2, 1))
  prob <- rnorm(N, mean = mu[h_hidden], sd = sigma[h_hidden])
  prob <- case_when(
    prob > 1 ~ 1,
    prob < 0 ~ 0,
    TRUE ~ prob
  )
  D <- rbinom(N, size = 1, prob = prob)
  Y <- h_hidden + delta * D + rnorm(N)
  fit <- lm(Y ~ D + h_hidden) # h_hidden を含む重回帰
  return(coef(fit)[2])
}
```

この関数で、シミュレーションを 1,000 回実行する。

```
sb2 <- replicate(1e3, sim_hospital2())
```

この結果を可視化する。

```
hist_sb2 <- tibble(delta = sb2) |>
  ggplot(aes(x = delta, y = after_stat(density))) +
  geom_histogram(color = 'black', binwidth = 0.01) +
  geom_vline(xintercept = 0.6, color = 'tomato') +
  labs(x = '重回帰によって推定される「効果」', y = '確率密度')
plot(hist_sb2)
```



このように、セレクションの原因となっている変数を重回帰モデルに加えることができれば、セレクションバイアスを除去することができる。

これは、セレクションの原因が観測されているときだけ。

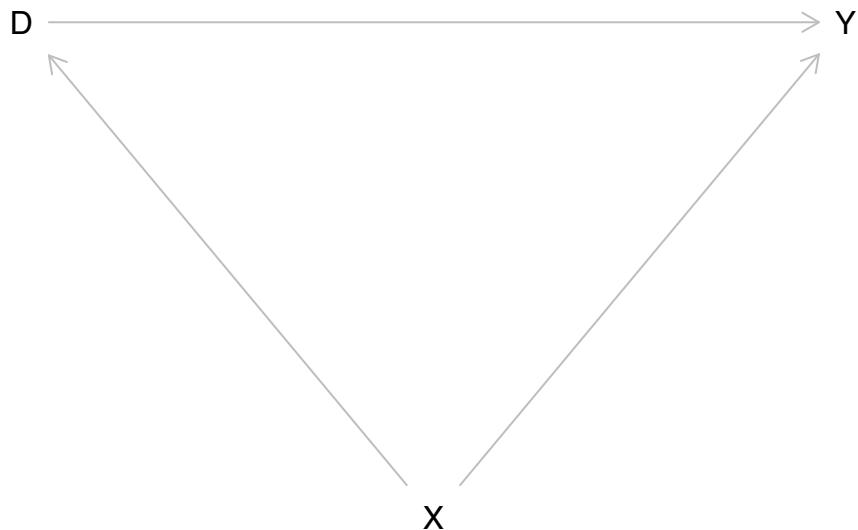
DAG (有向非巡回グラフ)

DAG (directed acyclic graph, 有向非巡回グラフ)を R で描く方法を簡単に説明する。

R で DAG を描くために、dagitty と ggdag の 2 つのパッケージを使う。両方使えると便利。

まず、dagitty だけで描いてみる。dagitty::dagitty() で DAG を定義し、それぞれの変数の位置を dagitty::coordinates() で指定する。

```
dag1 <- dagitty('dag{Y <- D; D <- X; Y <- X}')
coordinates(dag1) <- list(x = c(D = 0, X = 1, Y = 2),
                           y = c(D = 0, X = 1, Y = 0))
plot(dag1)
```

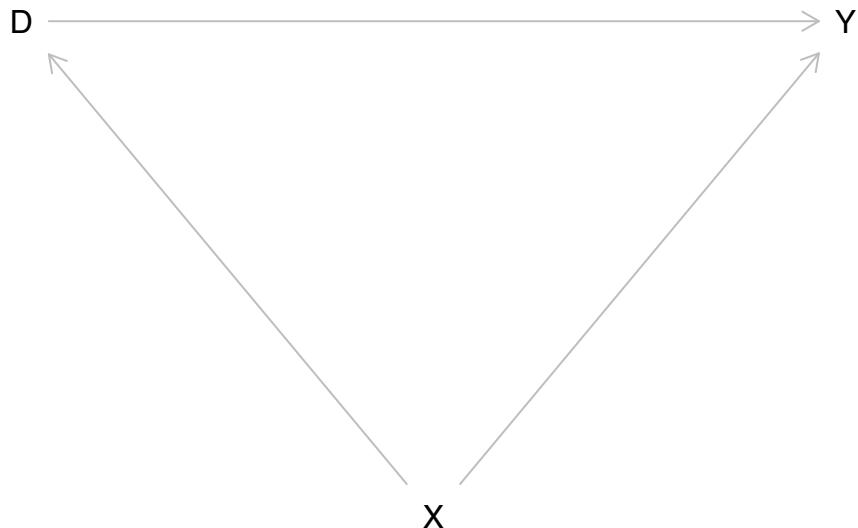


同じ DAG を描くための指定方法は色々ある。

```

dag2 <- dagitty('dag{D <- X -> Y; Y <- D}')
coordinates(dag2) <- coordinates(dag1)
plot(dag2)

```

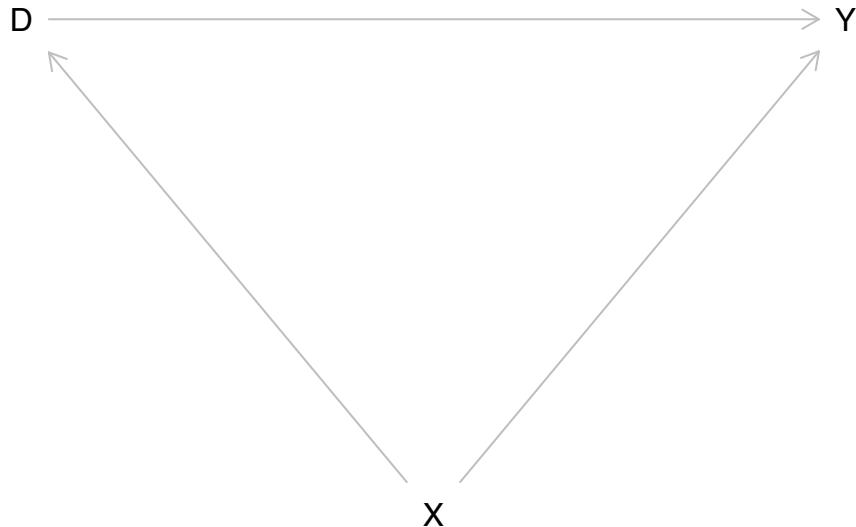


あるいは、次のようにしてもよい。

```

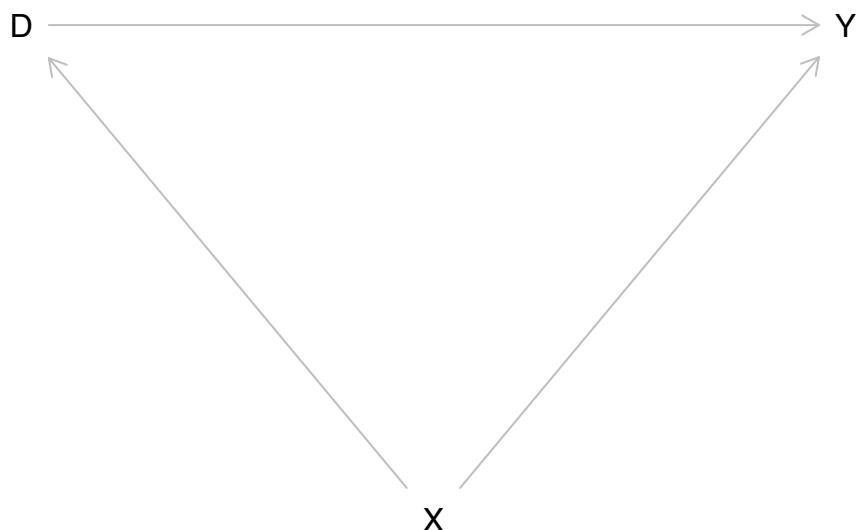
dag3 <- dagitty('dag{{D Y} <- X; Y <- D}')
coordinates(dag3) <- coordinates(dag1)
plot(dag3)

```



変数の役割を指定することもできる。その際、処置は `exposure` (曝露)、結果は `outcome`、未観測の変数は `unobserved` とする。

```
dag4 <- dagitty('dag{
  {D Y} <- X; Y <- D
  D [exposure]
  Y [outcome]
  X [unobserved]}')
coordinates(dag4) <- coordinates(dag1)
plot(dag4)
```

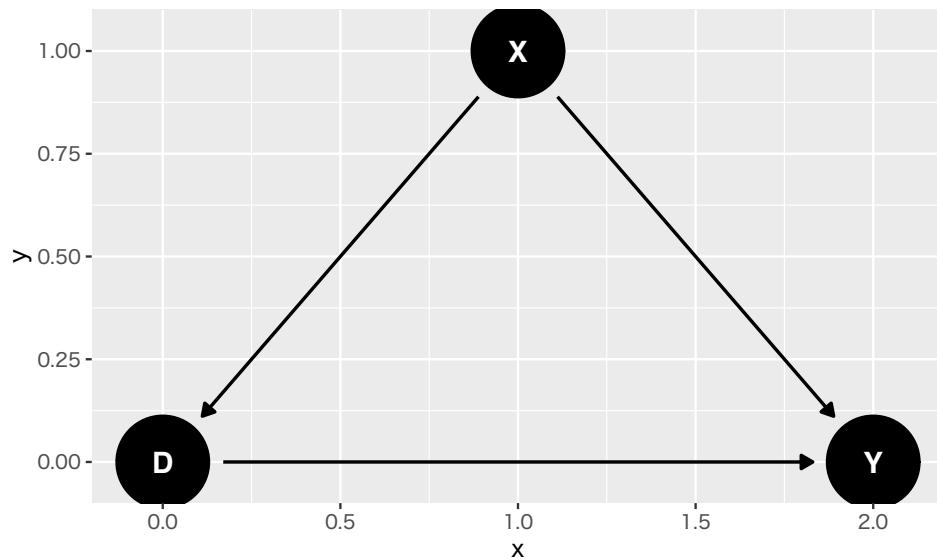


これを、`ggdag::tidy_dagitty()` で tidy なデータに変換する。

```
tidy_dag1 <- tidy_dagitty(dag4)
```

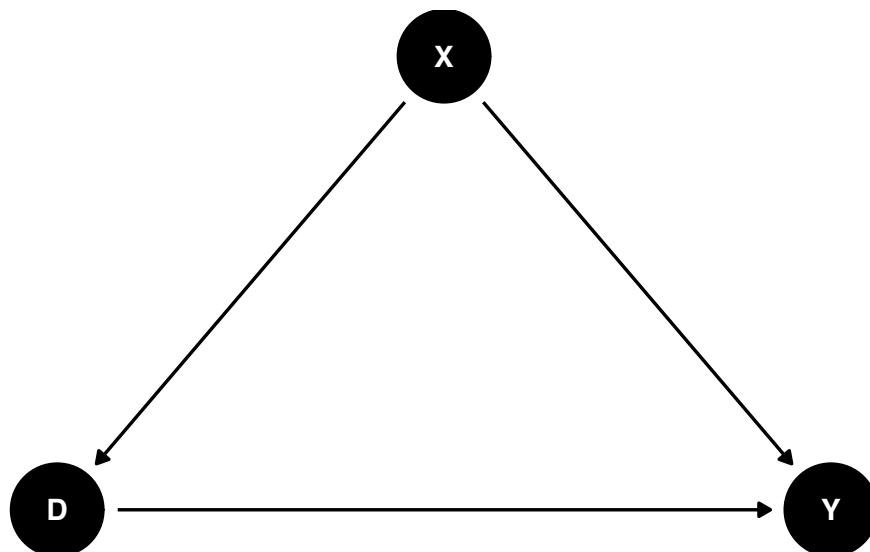
ggdag::ggdag() で ggplot2 風の DAG を描く。

```
ggdag(tidy_dag1)
```



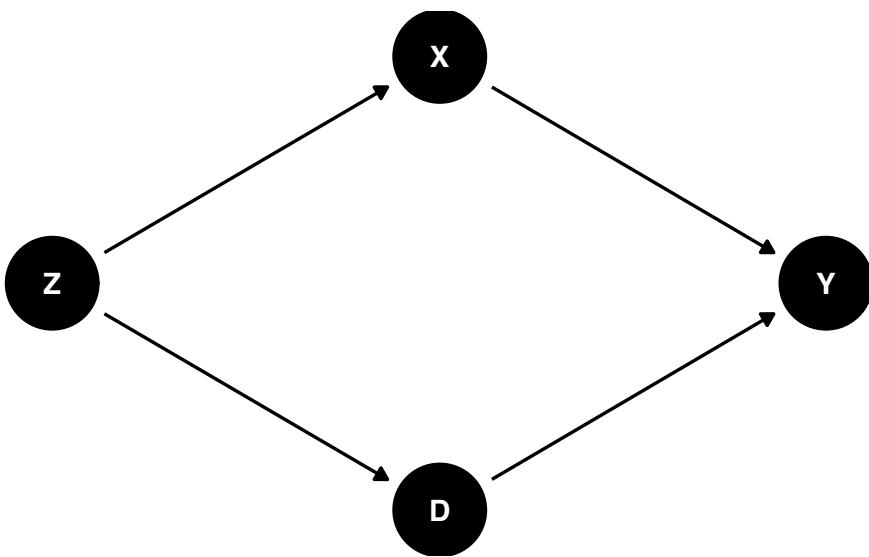
テーマを theme_dag() に変える。

```
ggdag(tidy_dag1) + theme_dag()
```



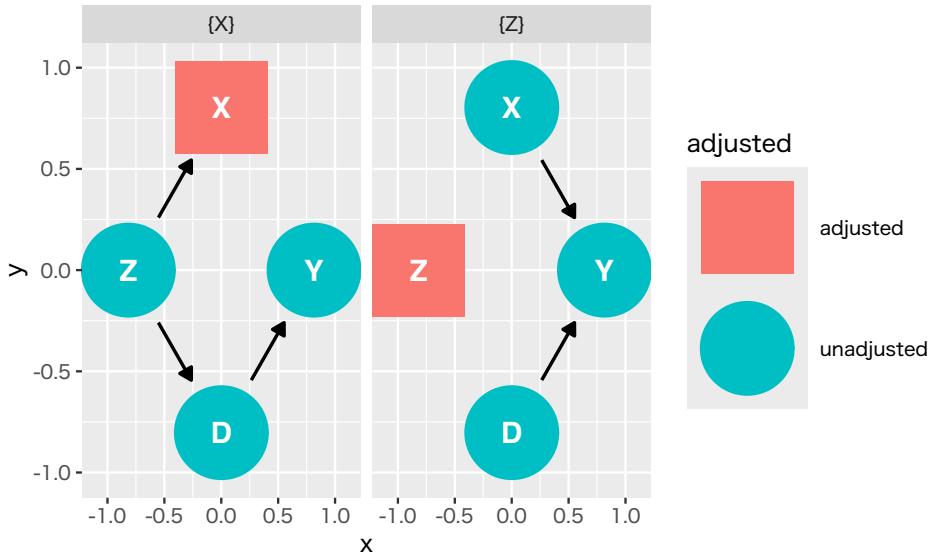
dagitty を使わずに、ggdag::dagify() で DAG を描くこともできる。

```
tidy_dag2 <- dagify(
  Y ~ D + X,
  D ~ Z,
  X ~ Z,
  exposure = 'D',
  outcome  = 'Y'
) |>
  tidy_dagitty()
ggdag(tidy_dag2) + theme_dag()
```



この DAG から、処置 D と結果 Y の間には交絡因子があることがわかる。では、どの変数を重回帰に含めればよいか。`ggdag::ggdag_adjustment_set()` で明らかにすることができる。

```
ggdag_adjustment_set(tidy_dag2)
```



`adjusted` とされている変数を重回帰に含めると、そこで経路が塞がれ、バックドア経路が断ち切られることがわかる。ここで、X または Z をコントロールすれば良いことがわかる。

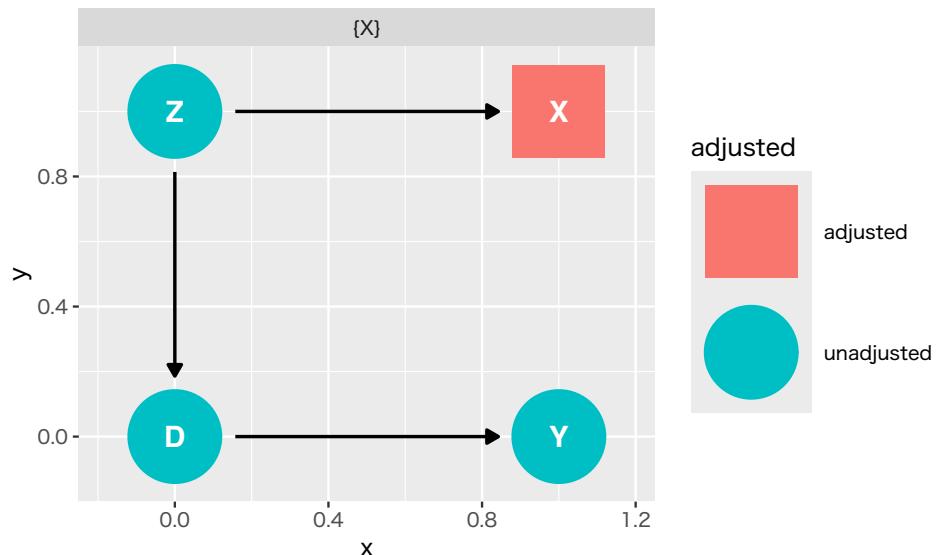
Z が未観測だったらどうなるだろうか。

```
tidy_dag3 <- dagify(
  Y ~ D + X,
  D ~ Z,
  X ~ Z,
  exposure = 'D',
  outcome  = 'Y',
  latent    = 'Z',
  coords    = list(x = c(D = 0, Y = 1, Z = 0, X = 1),
                  y = c(D = 0, Y = 0, Z = 1, X = 1))
) |>
  tidy_dagitty()
ggdag(tidy_dag3) + theme_dag()
```



`coord` で各変数(node)の座標を指定した以外、DAG 自体に変化はない。コントロールすべき変数はどれか。

```
ggdag_adjustment_set(tidy_dag3)
```



先程とは異なり、Z は未観測でコントロールできないので、X だけが候補として挙げられる。

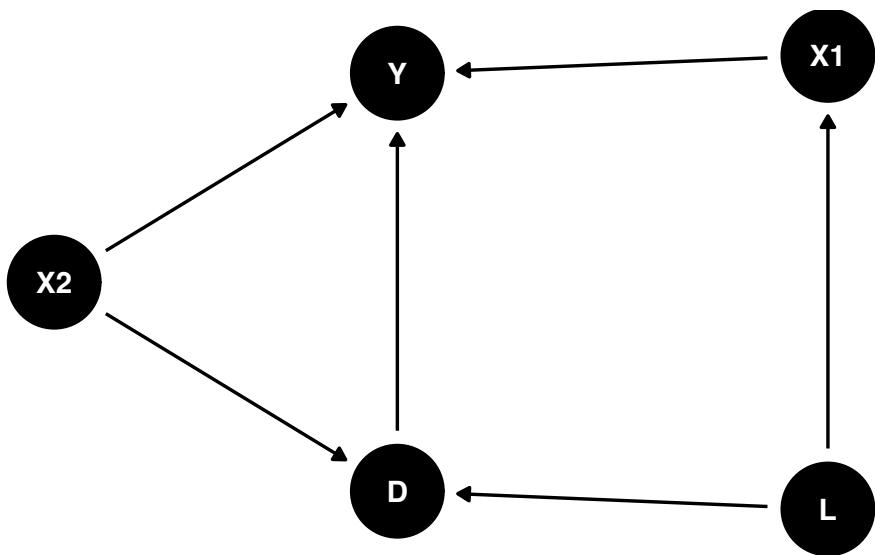
複数のコントロールが必要な場合は次のようにする。

```
tidy_dag4 <- dagify(
  Y ~ D + X1 + X2,
  D ~ L + X2,
  X1 ~ L,
```

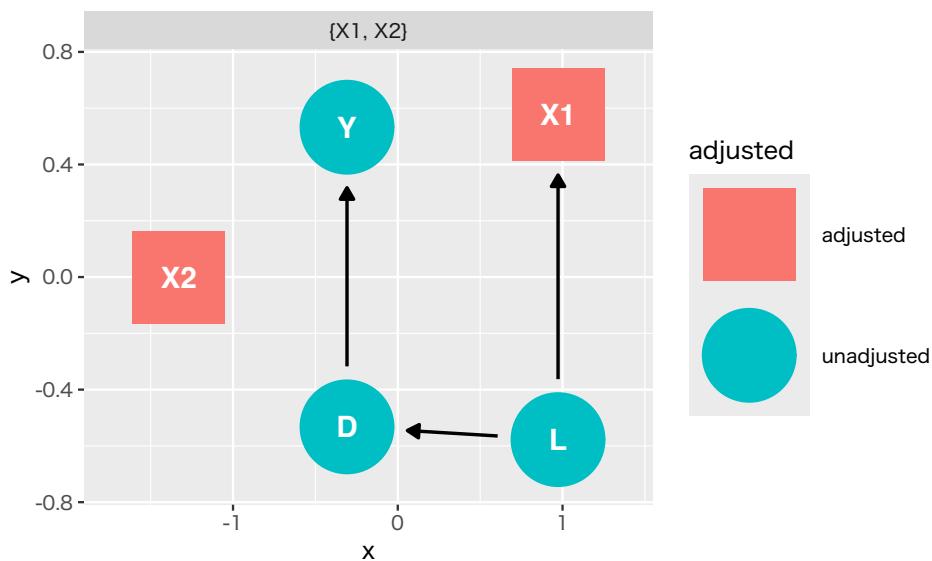
```

outcome  = 'Y',
exposure = 'D',
latent   = 'L'
) |> tidy_dagitty()
ggdag(tidy_dag4) + theme_dag()

```



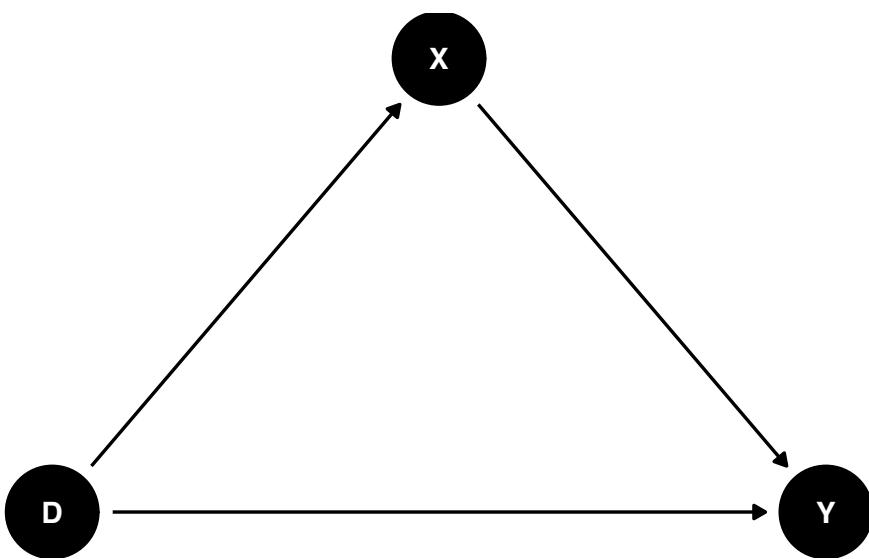
```
ggdag_adjustment_set(tidy_dag4)
```



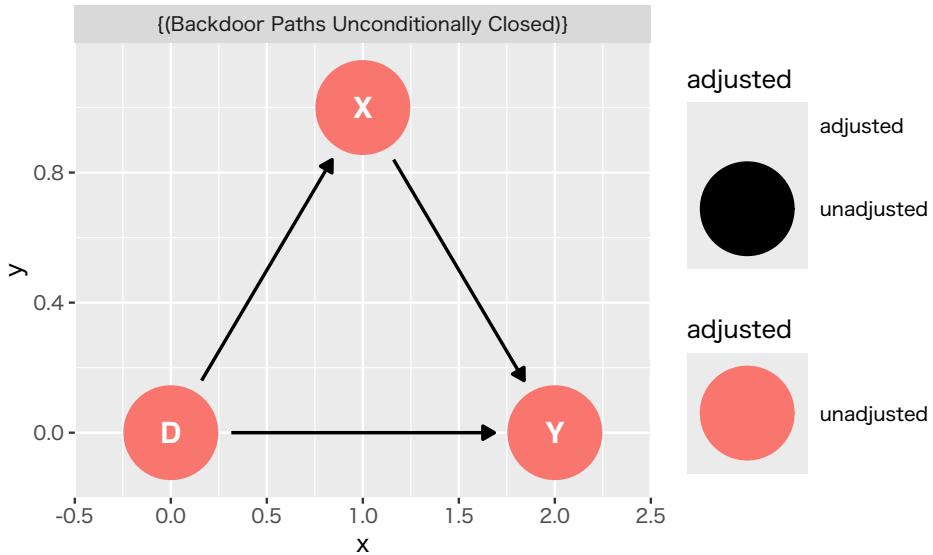
X_1 と X_2 の両者をコントロールする必要があることがわかる。

次に、処置後変数がある DAG を作り、コントロールすべきか調べてみる。

```
dag_pt1 <- dagitty('dag{
    Y <- X <- D; Y <- D
    D [exposure]
    Y [outcome]
  }')
coordinates(dag_pt1) <- list(x = c(D = 0, Y = 2, X = 1),
                             y = c(D = 0, Y = 0, X = 1))
tidy_dag_pt1 <- tidy_dagitty(dag_pt1)
ggdag(tidy_dag_pt1) + theme_dag()
```



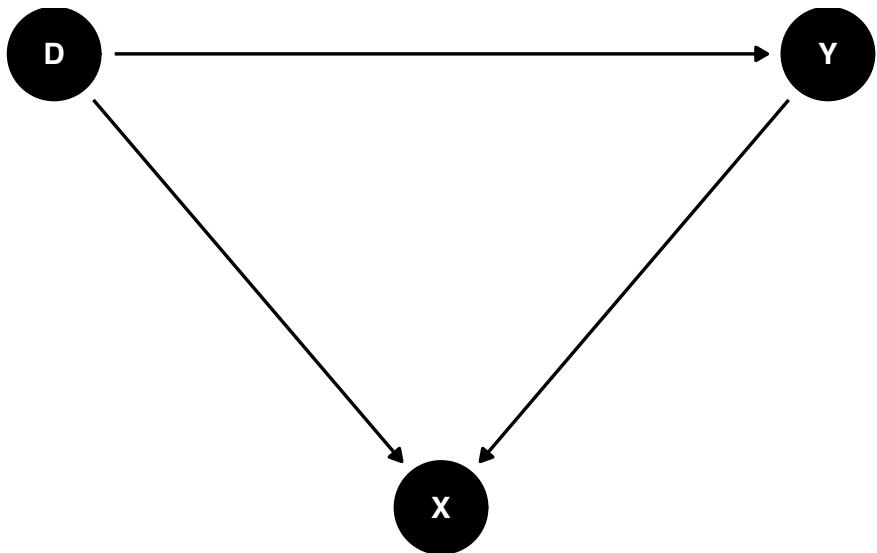
```
ggdag_adjustment_set(tidy_dag_pt1)
```



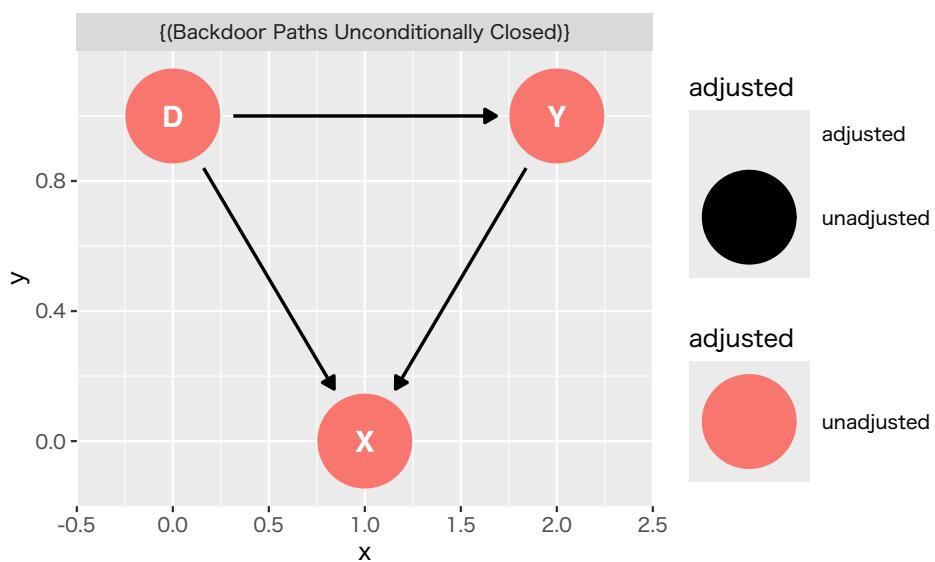
‘Backdoor Paths Unconditionally Closed’（コントロールなしでバックドアは閉じています）と表示され、Xをコントロールする必要がないことがわかる。

次のような場合はどうか。

```
tidy_dag_pt2 <- dagify(
  Y ~ D,
  X ~ D + Y,
  exposure = 'D',
  outcome  = 'Y',
  coords    = list(x = c(D = 0, Y = 2, X = 1),
                   y = c(D = 1, Y = 1, X = 0))
) |> tidy_dagitty()
ggdag(tidy_dag_pt2) + theme_dag()
```



```
ggdag_adjustment_set(tidy_dag_pt2)
```



先程と同様に、‘Backdoor Paths Unconditionally Closed’（コントロールなしでバックドアは閉じています）を表示され、X（共通効果、合流点、collider）をコントロールする必要はないことがわかる。