Drawing DAGs in R

Justin Smith

2023-10-31

What is a DAG?

- DAG stands for Directed Acyclic Graph
- It is a method to understand relationships between variables
- We use it to evaluate whether we can estimate a causal relationship

Why Draw DAGs?

- There might come a time when you want to draw a DAG in a paper or assignment
- It might look ugly if you draw by hand and include a photo
- Fortunately there are tools to draw it in R

Required Packages

- There are multiple packages to draw DAGs in R
- The key packages are:
  - `ggdag`
  - `dagitty`
- We will use `ggdag`
  - This is an extension of `dagitty` designed to work in the tidyverse

Creating DAG Data

- You can create a basic dag object with the `dagify` function
- Below we create a dag with variables w, x, and y

```
dag <- dagify(y~w + x, w ~x)
dag
```

```
## dag {
## w
## x
## y
## w -> y
## x -> w
## x -> y
## }
```

- This will save information on the variables and the direction of the relationships
  - Here we have said that w and x cause y
  - x causes w

- The way this information is stored and displayed is not very intuitive
- You can make it cleaner by using the `tidy_dagitty()` function

```
dag <- dagify(y~w + x, w ~x) %>% tidy_dagitty()
dag
```

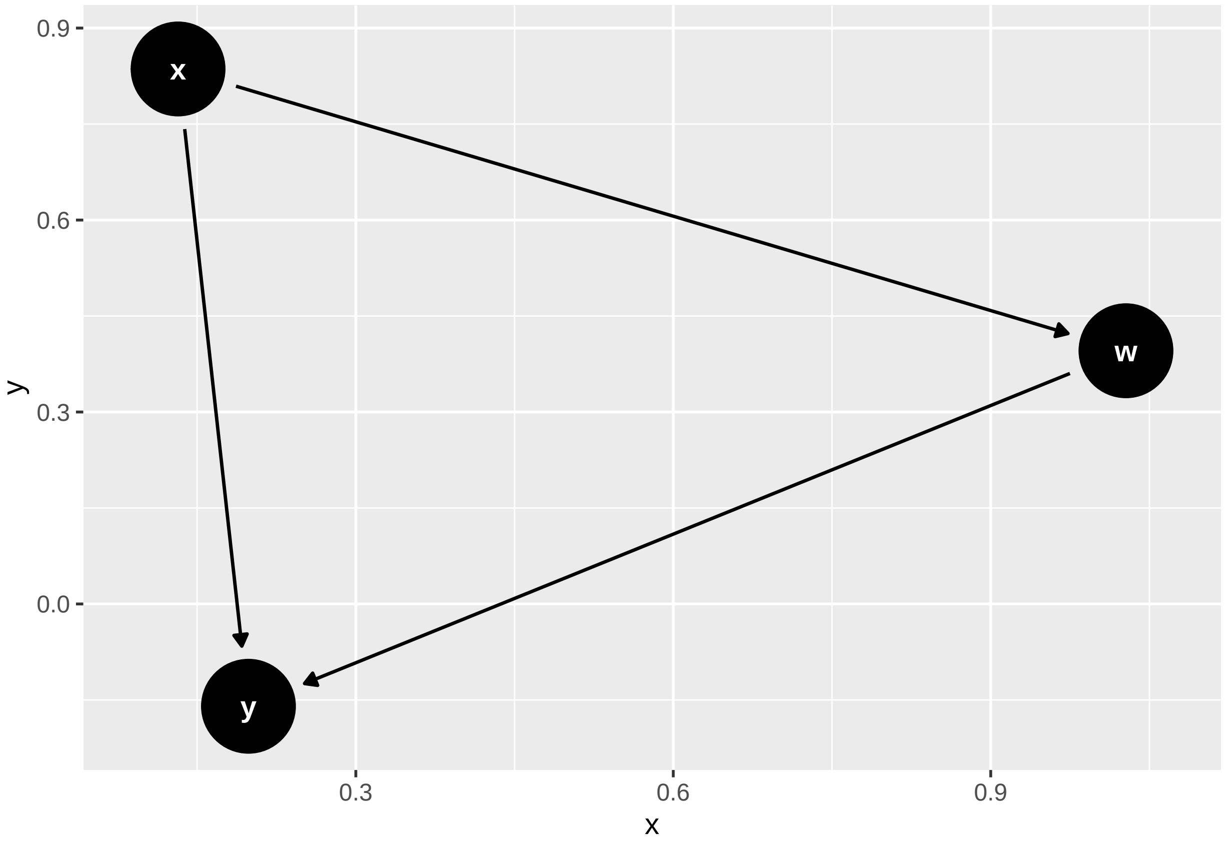
```
## # A DAG with 3 nodes and 3 edges
## #
## # A tibble: 4 x 8
##   name       x       y direction to       xend   yend circular
##   <chr>  <dbl>   <dbl> <fct>     <chr>   <dbl>  <dbl> <lgl>
## 1 w     1.02    0.682  ->        y      0.0369  0.863 FALSE
## 2 x     0.372  -0.0802 ->        w      1.02    0.682 FALSE
## 3 x     0.372  -0.0802 ->        y      0.0369  0.863 FALSE
## 4 y     0.0369  0.863  <NA>      <NA>  NA      NA     FALSE
```

- This stores all the same information, but in a tibble (data frame)

Plotting the DAG with `ggdag()`

- You can plot DAGs in a few ways
- We start with the `ggdag()` function
- To plot the DAG from above we can write

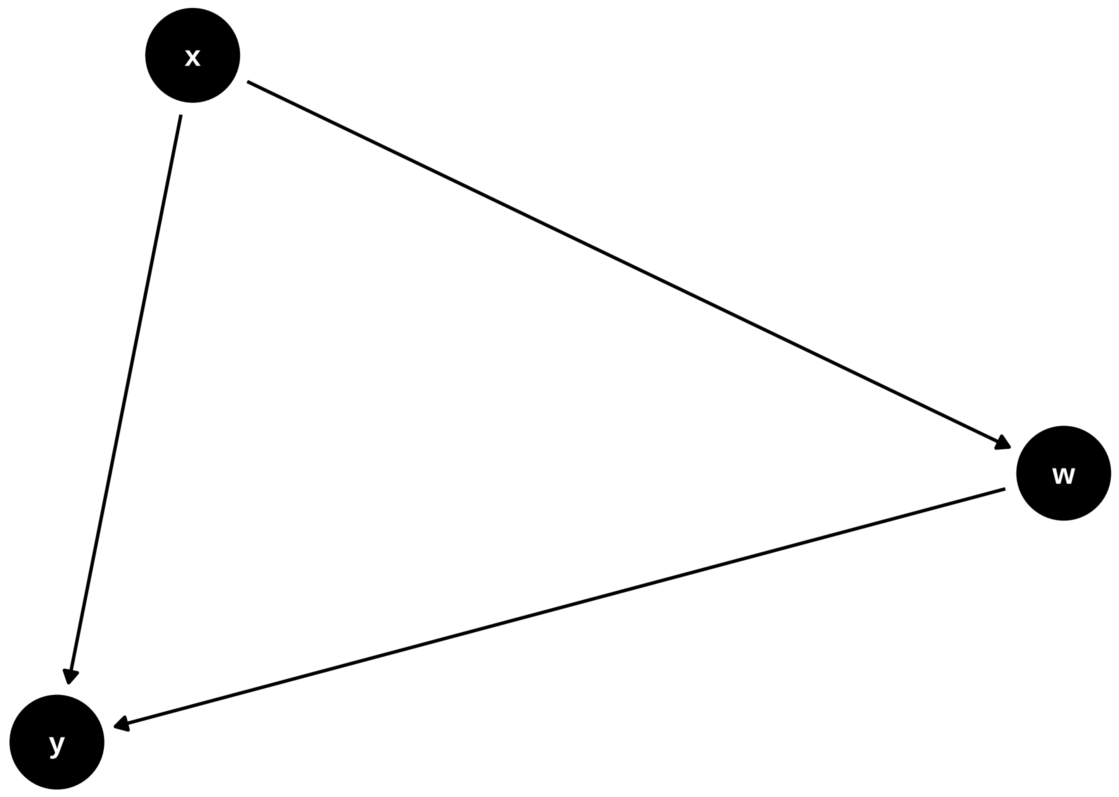
```
dag <- dagify(y~w + x, w ~x) %>% tidy_dagitty()
ggdag(dag)
```



- It takes all the information from the dag object

- Notice how it treats the plot area just like a scatterplot
    - You can move the dots around to suit the way you want it to look
- When you are drafting a DAG it is useful to have the x and y axis scales
- But in the final version you probably do not want them there
- You can remove with the `theme_dag()` layer
    - This will leave you with just the nodes, arrows, text

```
dag <- dagify(y~w + x, w ~x) %>% tidy_dagitty()
ggdag(dag) + theme_dag()
```



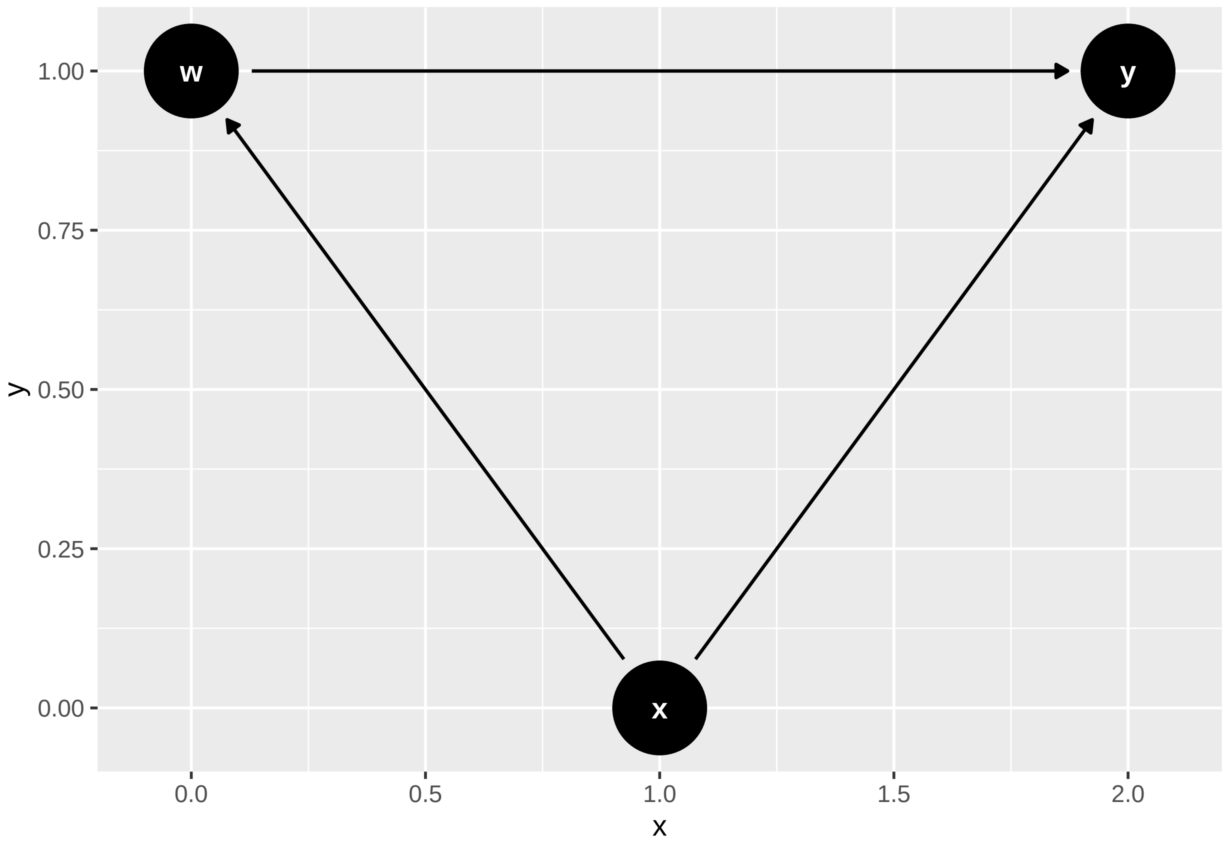
Moving the Node Positions

- With a 3-variable DAG, the default node positions usually work okay
- But sometimes you want to move the nodes around
- You can specify the coordinates of each node
    - Then feed them into the DAG data

```
coord_dag<-list(x = c(x = 1, w = 0, y = 2), y = c(x = 0, w = 1, y = 1))
dag <- dagify(y~w + x, w ~x, coords = coord_dag) %>% tidy_dagitty()
dag
```

```
## # A DAG with 3 nodes and 3 edges
## #
## # A tibble: 4 x 8
##   name      x     y direction to     xend  yend circular
##   <chr> <int> <int> <fct>     <chr> <int> <int> <lgl>
## 1 w          0      1 ->        y          2      1 FALSE
## 2 x          1      0 ->        w          0      1 FALSE
## 3 x          1      0 ->        y          2      1 FALSE
## 4 y          2      1 <NA>      <NA>      NA     NA FALSE
```

```
ggdag(dag)
```



- In the DAG above we set the following coordinates for nodes
  - w goes in the (0,1) position
  - x goes in the (1,0) position
  - y goes in the (2,1) position
  - The xend and yend are set automatically
  - We removed the theme to see the coordinates

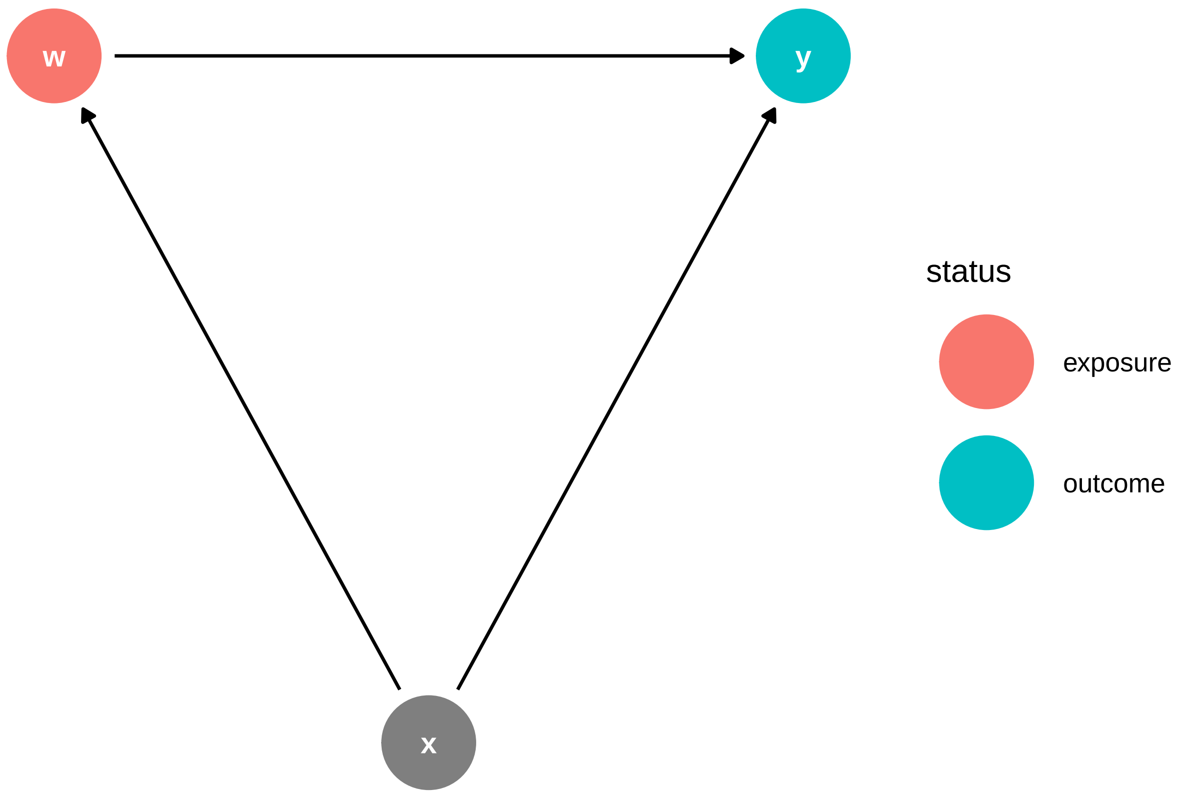
Identifying Treatment and Control

- You might want to identify the treatment and outcome variable
- That can be done in the `dagify()` function
- You can then plot with these variables identified using `ggdag_status()`

```
coord_dag<-list(x = c(x = 1, w = 0, y = 2), y = c(x = 0, w = 1, y = 1))
dag <- dagify(y~w + x, w ~x, coords = coord_dag, exposure = "w", outcome = "y") %>% tidy_dagitty()
dag
```

```
## # A DAG with 3 nodes and 3 edges
## #
## # Exposure: w
## # Outcome: y
## #
## # A tibble: 4 x 8
##   name      x     y direction to     xend  yend circular
##   <chr> <int> <int> <fct>     <chr> <int> <int> <lgl>
## 1 w         0     1 ->        y         2     1 FALSE
## 2 x         1     0 ->        w         0     1 FALSE
## 3 x         1     0 ->        y         2     1 FALSE
## 4 y         2     1 <NA>      <NA>     NA    NA FALSE
```

```
ggdag_status(dag) + theme_dag()
```



Plotting with ggplot2

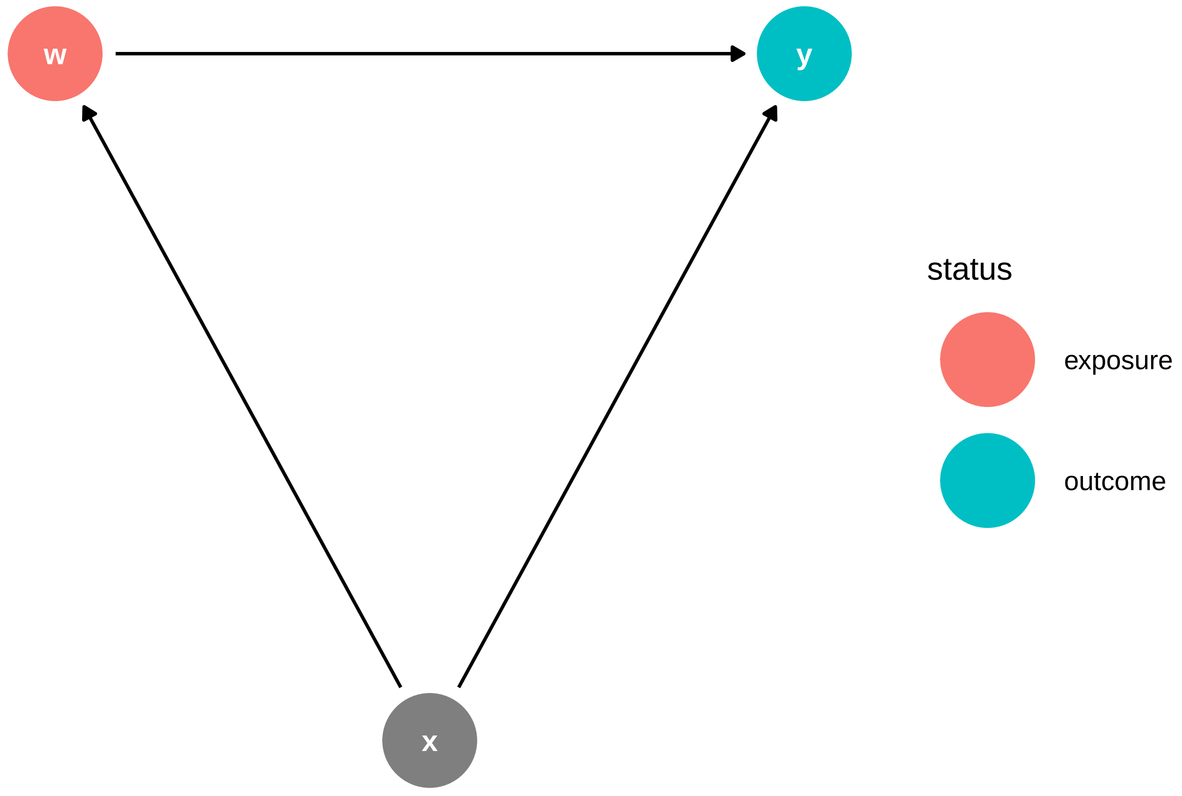
- ‘Many DAGs involve more than three variables
- You may wantggdag() uses ggplot2 in the background
- It is possible to plot directly with ggplot
- The same DAG from above is coded as

```
coord_dag<-list(x = c(x = 1, w = 0, y = 2), y = c(x = 0, w = 1, y = 1))
dag <- dagify(y~w + x, w ~x, coords = coord_dag, exposure = "w", outcome = "y") %>%
  tidy_dagitty() %>%
  node_status()

dag
```

```
## # A DAG with 3 nodes and 3 edges
## #
## # Exposure: w
## # Outcome: y
## #
## # A tibble: 4 x 9
##   name      x     y direction to     xend  yend circular status
##   <chr> <int> <int> <fct>     <chr> <int> <int> <lgl>    <fct>
## 1 w         0     1 ->        y         2     1 FALSE    exposure
## 2 x         1     0 ->        w         0     1 FALSE    <NA>
## 3 x         1     0 ->        y         2     1 FALSE    <NA>
## 4 y         2     1 <NA>      <NA>     NA    NA FALSE    outcome
```

```
ggplot(dag,aes(x = x, y = y, xend = xend, yend = yend, color = status)) +
  geom_dag_point() +
  geom_dag_edges() +
  geom_dag_text(col="white") +
  theme_dag() +
  scale_color_hue(breaks = c("exposure", "outcome"))
```



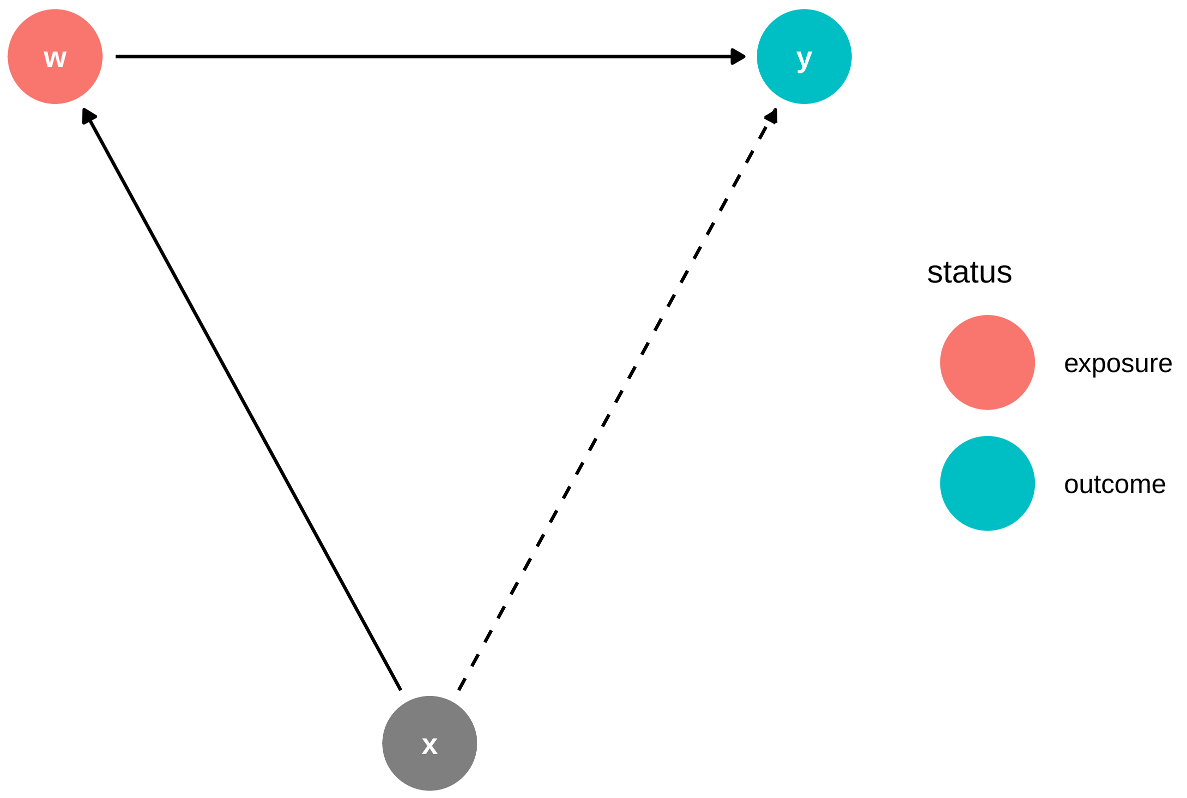
- You can also adjust other parts of the graph like line type
- Suppose you want to make the $x \rightarrow y$ line dashed

```
coord_dag<-list(x = c(x = 1, w = 0, y = 2), y = c(x = 0, w = 1, y = 1))
dag <- dagify(y~w + x, w ~x, coords = coord_dag, exposure = "w", outcome = "y") %>%
  tidy_dagitty() %>%
  node_status() %>%
  mutate(linetype = ifelse(name == "x" & to == "y", "dashed", "solid"))

dag
```

```
## # A DAG with 3 nodes and 3 edges
## #
## # Exposure: w
## # Outcome: y
## #
## # A tibble: 4 x 10
##   name      x     y direction to     xend  yend circular status   linetype
##   <chr> <int> <int> <fct>     <chr> <int> <int> <lgl>    <fct>    <chr>
## 1 w         0     1 ->        y         2     1 FALSE    exposure solid
## 2 x         1     0 ->        w         0     1 FALSE    <NA>     solid
## 3 x         1     0 ->        y         2     1 FALSE    <NA>     dashed
## 4 y         2     1 <NA>      <NA>     NA    NA FALSE    outcome  solid
```

```
ggplot(dag,aes(x = x, y = y, xend = xend, yend = yend, color = status)) +
  geom_dag_point() +
  geom_dag_edges(aes(edge_linetype = linetype)) +
  geom_dag_text(col="white") +
  theme_dag() +
  scale_color_hue(breaks = c("exposure", "outcome"))
```


w
y
x
status
exposure
outcome