Package ‘ggdag’

January 12, 2021

Title  Analyze and Create Elegant Directed Acyclic Graphs

Version  0.2.3

Description  Tidy, analyze, and plot directed acyclic graphs (DAGs). ‘ggdag’ is built on top of ‘dagitty’, an R package that uses the ‘DAGitty’ web tool (<http://dagitty.net>) for creating and analyzing DAGs. ‘ggdag’ makes it easy to tidy and plot ‘dagitty’ objects using ‘ggplot2’ and ‘ggraph’, as well as common analytic and graphical functions, such as determining adjustment sets and node relationships.

License  MIT + file LICENSE

URL  https://github.com/malcolmbarrett/ggdag

BugReports  https://github.com/malcolmbarrett/ggdag/issues

Depends  ggplot2 (>= 3.0.0), R (>= 3.2.0)

Imports  dagitty, dplyr, forcats, ggraph (>= 2.0.0), ggrepel, igraph, magrittr, pillar, purrr, stringr, tibble, tidygraph

Suggests  knitr, rmarkdown, spelling, testthat (>= 2.1.0), covr, vdiffr (>= 0.3.0)

VignetteBuilder  knitr

Encoding  UTF-8

Language  en-US

LazyData  true

RoxygenNote  7.1.1

NeedsCompilation  no

Author  Malcolm Barrett [aut, cre] (<https://orcid.org/0000-0003-0299-5825>)

Maintainer  Malcolm Barrett <malcolmbarrett@gmail.com>

Repository  CRAN

Date/Publication  2021-01-12 06:40:02 UTC
\textbf{R topics documented:}

activate_collider_paths ............................................. 3
Adjust for variables .................................................. 3
as.data.frame.tidy_dagitty ........................................... 5
as.tbl.tidy_dagitty .................................................... 5
Assess d-separation between variables ................................ 6
Assess familial relationships between variables ..................... 9
as_tbl_graph ......................................................... 13
Canonicalize DAGs .................................................... 14
Colliders ............................................................... 15
coordinates ............................................................. 17
Covariate Adjustment Sets ............................................ 18
dag ................................................................. 19
DAG Edges ............................................................. 20
DAG Labels ............................................................. 24
dagify ................................................................. 25
dplyr ................................................................. 26
Equivalent DAGs and Classes ......................................... 28
Exogenous Variables ................................................... 31
expand_plot ........................................................... 32
fortify ................................................................. 32
geom_dag_collider_edges .............................................. 33
geom_dag_edges ........................................................ 35
geom_dag_text ........................................................ 37
ggdag ................................................................. 39
ggdag_classic .......................................................... 40
ggplot.tidy_dagitty ...................................................... 42
ggrepel functions ....................................................... 42
Instrumental Variables ................................................. 45
is.tidy_dagitty ........................................................ 46
is_confounder .......................................................... 47
Nodes ................................................................. 47
Pathways ............................................................... 50
print.tidy_dagitty ...................................................... 52
Quick Plots for Common DAGs ......................................... 52
remove_axes ............................................................ 55
scale_adjusted .......................................................... 56
simulate_data ........................................................... 57
tbl_df.tidy_dagitty ...................................................... 58
Test if Variable Is Collider ............................................ 58
theme_dag_blank ........................................................ 59
theme_dag_grey ........................................................ 60
tidy_dagitty ............................................................. 60
Variable Status ......................................................... 61

\textbf{Index} 63
activate_collider_paths

Activate paths opened by stratifying on a collider

Description

Stratifying on colliders can open biasing pathways between variables. activate_collider_paths activates any such pathways given a variable or set of variables to adjust for and adds them to the tidy_dagitty.

Usage

activate_collider_paths(.tdy_dag, adjust_for, ...)

Arguments

- **.tdy_dag**: input graph, an object of class tidy_dagitty or dagitty
- **adjust_for**: a character vector, the variable(s) to adjust for.
- **...**: additional arguments passed to tidy_dagitty()

Value

A tidy_dagitty with additional rows for collider-activated pathways

See Also

control_for(), ggdag_adjust(), geom_dag Collider_edges()

Examples

dag <- dagify(m ~ x + y, x ~ y)

collided_dag <- activate_collider_paths(dag, adjust_for = "m")
collided_dag

Adjust for variables Adjust for variables and activate any biasing paths that result

Description

Adjust for variables and activate any biasing paths that result
Adjust for variables

Usage

control_for(.tdy_dag, var, as_factor = TRUE, ...)

adjust_for(.tdy_dag, var, as_factor = TRUE, ...)

ggdag_adjust(
  .tdy_dag,
  var = NULL,
  ..., 
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL,
  collider_lines = TRUE
)

Arguments

.tdy_dag  input graph, an object of class tidy_dagitty or dagitty
var       a character vector, the variable(s) to adjust for.
as_factor logical. Should the adjusted column be a factor?
...       additional arguments passed to tidy_dagitty()
node_size size of DAG node
text_size size of DAG text
label_size size of label text
text_col  color of DAG text
label_col color of label text
node      logical. Should nodes be included in the DAG?
stylized  logical. Should DAG nodes be stylized? If so, use geom_dag_nodes and if not use geom_dag_point
text      logical. Should text be included in the DAG?
use_labels a string. Variable to use for geom_dag_repel_label(). Default is NULL.
collider_lines logical. Should the plot show paths activated by adjusting for a collider?

Value

a tidy_dagitty with a adjusted column for adjusted variables, as well as any biasing paths that arise, or a ggplot
as.data.frame.tidy_dagitty

Examples

dag <- dagify(m ~ a + b, x ~ a, y ~ b)
control_for(dag, var = "m")
ggdag_adjust(dag, var = "m")

as.data.frame.tidy_dagitty

Convert a tidy_dagitty object to data.frame

Description

Convert a tidy_dagitty object to data.frame

Usage

## S3 method for class 'tidy_dagitty'
as.data.frame(x, row.names, optional, ...)

Arguments

x an object of class tidy_dagitty
row.names NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
optional logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional. Note that all of R’s base package as.data.frame() methods use optional only for column names treatment, basically with the meaning of data.frame(*, check.names = !optional)
...
optional arguments passed to as.data.frame()

as.tbl.tidy_dagitty

Convert a tidy_dagitty object to tbl

Description

Convert a tidy_dagitty object to tbl

Usage

## S3 method for class 'tidy_dagitty'
as.tbl(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'tidy_dagitty'
as_tibble(x, row.names = NULL, optional = FALSE, ...)
Assess d-separation between variables

D-relationship between variables

Arguments

- `x` an object of class `tidy_dagitty`
- `row.names` NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
- `optional` logical. If TRUE, setting row names and converting column names (to syntactic names: see `make.names`) is optional. Note that all of R's base package `as.data.frame()` methods use `optional` only for column names treatment, basically with the meaning of `data.frame(*, check.names = !optional)`
- `...` optional arguments passed to `dplyr::as_tibble()`

Description

D-separation is a key concept in causal structural models. Variables are d-separated if there are no open paths between them. The `node_d*()` functions label variables as d-connected or d-separated. The `ggdag_d*()` functions plot the results. The `*_dconnected()`, `*_dseparated()`, and `*_drelationship()` functions essentially produce the same output and are just different ways of thinking about the relationship. See `dagitty::dseparated()` for details.

Usage

```r
node_dconnected(
  .tdy_dag,
  from = NULL,
  to = NULL,
  controlling_for = NULL,
  as_factor = TRUE,
  ...
)

node_dseparated(
  .tdy_dag,
  from = NULL,
  to = NULL,
  controlling_for = NULL,
  as_factor = TRUE
)

node_drelationship(
  .tdy_dag,
  from = NULL,
  to = NULL,
)```
Assess d-separation between variables

controlling_for = NULL,
as_factor = TRUE

)  
ggdag_drelationship(
  .tdy_dag,
  from = NULL,
  to = NULL,
  controlling_for = NULL,
  ...,
  edge_type = "link_arc",
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL,
  collider_lines = TRUE

)  
ggdag_dseparated(
  .tdy_dag,
  from = NULL,
  to = NULL,
  controlling_for = NULL,
  ...,
  edge_type = "link_arc",
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL,
  collider_lines = TRUE

)  
ggdag_dconnected(
  .tdy_dag,
  from = NULL,
  to = NULL,
  controlling_for = NULL,
  ...,
Assess d-separation between variables

```
edge_type = "link_arc",
node_size = 16,
text_size = 3.88,
label_size = text_size,
text_col = "white",
label_col = text_col,
node = TRUE,
stylized = FALSE,
text = TRUE,
use_labels = NULL,
collider_lines = TRUE
```

Arguments

- `.tdy_dag` input graph, an object of class tidy_dagitty or dagitty
- `from` a character vector, the starting variable (must by in DAG). If NULL, checks DAG for exposure variable.
- `to` a character vector, the ending variable (must by in DAG). If NULL, checks DAG for outcome variable.
- `controlling_for` a character vector, variables in the DAG to control for.
- `as_factor` logical. Should the d_relationship variable be a factor?
- `...` additional arguments passed to tidy_dagitty()
- `edge_type` a character vector, the edge geom to use. One of: "link_arc", which accounts for directed and bidirected edges, "link", "arc", or "diagonal"
- `node_size` size of DAG node
- `text_size` size of DAG text
- `label_size` size of label text
- `text_col` color of DAG text
- `label_col` color of label text
- `node` logical. Should nodes be included in the DAG?
- `stylized` logical. Should DAG nodes be stylized? If so, use geom_dag_nodes and if not use geom_dag_point
- `text` logical. Should text be included in the DAG?
- `use_labels` a string. Variable to use for geom_dag_repel_label(). Default is NULL.
- `collider_lines` logical. Should the plot show paths activated by adjusting for a collider?

Value

a tidy_dagitty with a d_relationship column for variable D relationship or a ggplot
Assess familial relationships between variables

Examples

```r
dag <- dagify(m ~ x + y)
dag %>% ggdag_drelationship("x", "y")
dag %>% ggdag_drelationship("x", "y", controlling_for = "m")

dag %>%
  node_dseparated("x", "y") %>%
  ggplot(aes(x = x, y = y, xend = xend, yend = yend, shape = adjusted, col = d_relationship)) +
  geom_dag_edges() +
  geom_dag_collider_edges() +
  geom_dag_node() +
  geom_dag_text(col = "white") +
  theme_dag() + scale_adjusted()

dag %>%
  node_dconnected("x", "y", controlling_for = "m") %>%
  ggplot(aes(x = x, y = y, xend = xend, yend = yend, shape = adjusted, col = d_relationship)) +
  geom_dag_edges() +
  geom_dag_collider_edges() +
  geom_dag_node() +
  geom_dag_text(col = "white") +
  theme_dag() +
  scale_adjusted()

dagify(m ~ x + y, m_jr ~ m) %>%
  tidy_dagitty(layout = "nicely") %>%
  node_dconnected("x", "y", controlling_for = "m_jr") %>%
  ggplot(aes(x = x, y = y, xend = xend, yend = yend, shape = adjusted, col = d_relationship)) +
  geom_dag_edges() +
  geom_dag_collider_edges() +
  geom_dag_node() +
  geom_dag_text(col = "white") +
  theme_dag() +
  scale_adjusted()
```

Description

Parents and children are those nodes that either directly cause or are caused by the variable, respectively. Ancestors and descendants are those nodes that are on the path to or descend from the variable. The node_*() functions label variables depending on their relationship. The ggdag_*() functions plot the results. See dagitty::children for details.

Usage

```r
node_children(.tdy_dag, .var, as_factor = TRUE)
```
Assess familial relationships between variables

```r
node_parents(.tdy_dag, .var, as_factor = TRUE)
node_ancestors(.tdy_dag, .var, as_factor = TRUE)
node_descendants(.tdy_dag, .var, as_factor = TRUE)
node_markov_blanket(.tdy_dag, .var, as_factor = TRUE)
node_adjacent(.tdy_dag, .var, as_factor = TRUE)

# ggdag algorithms

ggdag_children(
  .tdy_dag,
  .var,
  ...,
  edge_type = "link_arc",
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL
)

ggdag_parents(
  .tdy_dag,
  .var,
  ...,
  edge_type = "link_arc",
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL
)

ggdag_ancestors(
  .tdy_dag,
  .var,
  ...,
  edge_type = "link_arc",
```
Assess familial relationships between variables

```r
node_size = 16,
text_size = 3.88,
label_size = text_size,
text_col = "white",
label_col = text_col,
node = TRUE,
stylized = FALSE,
text = TRUE,
use_labels = NULL
)

ggdag_descendants(
  .tdy_dag,
  .var,
  ..., 
  edge_type = "link_arc",
  node_size = 16,
text_size = 3.88,
label_size = text_size,
text_col = "white",
label_col = text_col,
node = TRUE,
stylized = FALSE,
text = TRUE,
use_labels = NULL
)

ggdag_markov_blanket(
  .tdy_dag,
  .var,
  ..., 
  edge_type = "link_arc",
  node_size = 16,
text_size = 3.88,
label_size = text_size,
text_col = "white",
label_col = text_col,
node = TRUE,
stylized = FALSE,
text = TRUE,
use_labels = NULL
)

ggdag_adjacent(
  .tdy_dag,
  .var,
  ..., 
  edge_type = "link_arc",
```
Assess familial relationships between variables

```r
node_size = 16,
text_size = 3.88,
label_size = text_size,
text_col = "white",
label_col = text_col,
node = TRUE,
stylized = FALSE,
text = TRUE,
use_labels = NULL
```

**Arguments**

- `.tdy_dag` input graph, an object of class `tidy_dagitty` or `dagitty`
- `.var` a character vector, the variable to be assessed (must by in DAG)
- `as_factor` logical. Should the relationship variable be a factor?
- `...` additional arguments passed to `tidy_dagitty()`
- `edge_type` a character vector, the edge geom to use. One of: "link_arc", which accounts for directed and bidirected edges, "link", "arc", or "diagonal"
- `node_size` size of DAG node
- `text_size` size of DAG text
- `label_size` size of label text
- `text_col` color of DAG text
- `label_col` color of label text
- `node` logical. Should nodes be included in the DAG?
- `stylized` logical. Should DAG nodes be stylized? If so, use `geom_dag_nodes` and if not use `geom_dag_point`
- `text` logical. Should text be included in the DAG?
- `use_labels` a string. Variable to use for `geom_dag_repel_label()`. Default is `NULL`.

**Value**

a `tidy_dagitty` with an column related to the given relationship for variable D relationship or a `ggplot`

**Examples**

```r
dag <- dagify(y ~ x + z2 + w2 + w1,
x ~ z1 + w1,
  z1 ~ w1 + v,
  z2 ~ w2 + v,
  w1 ~ w2)

ggdag_children(dag, "w1")
dag %>%
```
as_tbl_graph

node_children("w1") %>%
ggplot(aes(x = x, y = y, xend = xend, yend = yend, color = children)) +
geom_dag_edges() +
geom_dag_node() +
geom_dag_text(col = "white") +
geom_dag_label_repel(aes(label = children, fill = children), col = "white", show.legend = FALSE) +
theme_dag() +
scale_adjusted() +
scale_color_hue(breaks = c("parent", "child"))
ggdag_parents(dag, "y")
ggdag_ancestors(dag, "x")
ggdag_descendants(dag, "w1")
dag %>%
node_parents("y") %>%
ggplot(aes(x = x, y = y, xend = xend, yend = yend, color = parent)) +
geom_dag_edges() +
geom_dag_point() +
geom_dag_text(col = "white") +
geom_dag_label_repel(aes(label = parent, fill = parent), col = "white", show.legend = FALSE) +
theme_dag() +
scale_adjusted() +
scale_color_hue(breaks = c("parent", "child"))

as_tbl_graph

Convert DAGS to tidygraph

Description

A thin wrapper to convert tidy_dagitty and dagitty objects to tbl_graph, which can then be used to work in tidygraph and ggraph directly. See tidygraph::as_tbl_graph().

Usage

## S3 method for class 'tidy_dagitty'
as_tbl_graph(x, directed = TRUE, ...)

## S3 method for class 'dagitty'
as_tbl_graph(x, directed = TRUE, ...)

Arguments

x an object of class tidy_dagitty or dagitty

directed logical. Should the constructed graph be directed? Default is TRUE

... other arguments passed to as_tbl_graph
**Value**

a tbl_graph

**Examples**

```r
library(ggraph)
library(tidygraph)
butterfly_bias() %>%
as_tbl_graph() %>%
ggraph() +
geom_edge_diagonal() +
geom_node_point()
```

---

**Canonicalize DAGs**

**Canonicalize a DAG**

**Description**

Takes an input graph with bidirected edges and replaces every bidirected edge `x <-> y` with a substructure `x <- L -> y`, where `L` is a latent variable. See `dagitty::canonicalize()` for details. Undirected edges are not currently supported in ggdag.

**Usage**

```r
node_canonical(.dag, ...)

ggdag_canonical(
  .tdy_dag, 
  ..., 
  edge_type = "link_arc", 
  node_size = 16, 
  text_size = 3.88, 
  label_size = text_size, 
  text_col = "white", 
  label_col = text_col, 
  node = TRUE, 
  stylized = FALSE, 
  text = TRUE, 
  use_labels = NULL 
)
```

**Arguments**

- `.dag, .tdy_dag` input graph, an object of class tidy_dagitty or dagitty
- `...` additional arguments passed to tidy_dagitty()
Colliders

edge_type  
a character vector, the edge geom to use. One of: "link_arc", which accounts for 
directed and bidirected edges, "link", "arc", or "diagonal"

node_size  
size of DAG node

text_size  
size of DAG text

label_size  
size of label text

text_col  
color of DAG text

label_col  
color of label text

node  
logical. Should nodes be included in the DAG?

stylized  
logical. Should DAG nodes be stylized? If so, use geom_dag_nodes and if not 
use geom_dag_point

text  
logical. Should text be included in the DAG?

use_labels  
a string. Variable to use for geom_dag_repel_label(). Default is NULL.

Value

taxidy_dagitty that includes L or a ggplot

Examples

dag <- dagify(y ~ x + z, x ~~ z)

ggdag(dag)

node Canonical(dag)

nodeCanonical(dag)

node_collider(.dag, as_factor = TRUE, ...)

ggdag_collider(
  .tdy_dag,
  ..., 
  edge_type = "link_arc", 
  node_size = 16, 
  text_size = 3.88, 
)

Description

Detects any colliders given a DAG. node_collider tags colliders and ggdag_collider plots all 
exogenous variables.

Usage

node_collider(.dag, as_factor = TRUE, ...)

ggdag_collider(
  .tdy_dag,
  ..., 
  edge_type = "link_arc", 
  node_size = 16, 
  text_size = 3.88, 
)
label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL
)

Arguments

.dag, .tdy_dag  input graph, an object of class tidy_dagitty or dagitty
as_factor   treat collider variable as factor
...   additional arguments passed to tidy_dagitty()
edge_type   a character vector, the edge geom to use. One of: "link_arc", which accounts for
directed and bidirected edges, "link", "arc", or "diagonal"
node_size   size of DAG node
text_size   size of DAG text
label_size   size of label text
text_col   color of DAG text
label_col   color of label text
node   logical. Should nodes be included in the DAG?
stylized   logical. Should DAG nodes be stylized? If so, use geom_dag_nodes and if not
  use geom_dag_point
text   logical. Should text be included in the DAG?
use_labels   a string. Variable to use for geom_dag_repel_label(). Default is NULL.

Value

da tidy_dagitty with a collider column for colliders or a ggplot

Examples

dag <- dagify(m ~ x + y, y ~ x)
node_collder(dag)
ggdag_collider(dag)
coordinates

Manipulate DAG coordinates

Description

Manipulate DAG coordinates

Usage

coords2df(coord_list)
coords2list(coord_df)

Arguments

coord_list a named list of coordinates
coord_df a data.frame with columns x, y, and name

Value

either a list or a data.frame with DAG node coordinates

Examples

library(dagitty)
coords <- list(
x = c(A = 1, B = 2, D = 3, C = 3, F = 3, E = 4, G = 5, H = 5, I = 5),
y = c(A = 0, B = 0, D = 1, C = 0, F = -1, E = 0, G = 1, H = 0, I = -1)
)
coord_df <- coords2df(coords)
coords2list(coord_df)

x <- dagitty('dag(
  G <-> H <-> I <-> G
  D <-> B -> C -> I <- F <- B <-> A
  H <- E <-> C -> G <-> D
)
')
coordinates(x) <- coords2list(coord_df)
Covariate Adjustment Sets

Description

See `dagitty::adjustmentSets()` for details.

Usage

```r
dag_adjustment_sets(.tdy_dag, exposure = NULL, outcome = NULL, ...)

ggdag_adjustment_set(.tdy_dag,
  exposure = NULL,
  outcome = NULL,
  ...,
  shadow = FALSE,
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL,
  expand_x = expansion(c(0.25, 0.25)),
  expand_y = expansion(c(0.2, 0.2)))
```

Arguments

- `.tdy_dag` input graph, an object of class `tidy_dagitty` or `dagitty`
- `exposure` a character vector, the exposure variable. Default is `NULL`, in which case it will be determined from the DAG.
- `outcome` a character vector, the outcome variable. Default is `NULL`, in which case it will be determined from the DAG.
- `...` additional arguments to `adjustmentSets`
- `shadow` logical. Show paths blocked by adjustment?
- `node_size` size of DAG node
- `text_size` size of DAG text
- `label_size` size of label text
- `text_col` color of DAG text
**Description**

A convenience wrapper for dagitty::dagitty("dag...")
Usage

dag(...)  

Arguments

...  

a character vector in the style of dagitty. See dagitty::dagitty for details.

Value

a dagitty

Examples

dag("{x m} -> y")

---

DAG Edges   Directed DAG edges

Description

Directed DAG edges

Usage

geom_dag_edges_link(
  mapping = NULL,
  data = NULL,
  arrow = grid::arrow(length = grid::unit(5, "pt"), type = "closed"),
  position = "identity",
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

geom_dag_edges_arc(
  mapping = NULL,
  data = NULL,
  curvature = 0.5,
  arrow = grid::arrow(length = grid::unit(5, "pt"), type = "closed"),
  position = "identity",
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE,
  fold = FALSE,
\begin{verbatim}
geom_dag_edges_diagonal(
  mapping = NULL,
  data = NULL,
  position = "identity",
  arrow = grid::arrow(length = grid::unit(5, "pt"), type = "closed"),
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE,
  curvature = 1,
  n = 100,
  lineend = "butt",
  linejoin = "round",
  linemitre = 1,
  label_colour = "black",
  label_alpha = 1,
  label_parse = FALSE,
  check_overlap = FALSE,
  angle_calc = "rot",
  force_flip = TRUE,
  label_dodge = NULL,
  label_push = NULL,
  ...
)
\end{verbatim}

\begin{verbatim}
geom_dag_edges_fan(
  mapping = NULL,
  data = NULL,
  position = "identity",
  arrow = grid::arrow(length = grid::unit(5, "pt"), type = "closed"),
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE,
  spread = 0.7,
  ...
)
\end{verbatim}
n = 100,
lineend = "butt",
linejoin = "round",
linemitre = 1,
label_colour = "black",
label_alpha = 1,
label_parse = FALSE,
check_overlap = FALSE,
angle_calc = "rot",
force_flip = TRUE,
label_dodge = NULL,
label_push = NULL,

Arguments

mapping  Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data  The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame., and will be used as the layer data.

arrow  specification for arrow heads, as created by arrow()

position  Position adjustment, either as a string, or the result of a call to a position adjustment function.

na.rm  If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values

show.legend  logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes  If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders().

...  Other arguments passed to ggraph::geom_edge_*(

curvature  The bend of the curve. 1 approximates a halfcircle while 0 will give a straight line. Negative number will change the direction of the curve. Only used if layout circular = FALSE.

fold  Logical. Should arcs appear on the same side of the nodes despite different directions. Default to FALSE.

n  The number of points to create along the path.

lineend  Line end style (round, butt, square).
DAG Edges

linejoin  Line join style (round, mitre, bevel).
linemitre  Line mitre limit (number greater than 1).
label_colour  The colour of the edge label. If NA it will use the colour of the edge.
label_alpha  The opacity of the edge label. If NA it will use the opacity of the edge.
label_parse  If TRUE, the labels will be parsed into expressions and displayed as described in `grDevices::plotmath()`.
check_overlap  If TRUE, text that overlaps previous text in the same layer will not be plotted. check_overlap happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling `geom_label()` or `geom_text()`.
angle_calc  Either 'none', 'along', or 'across'. If 'none' the label will use the angle aesthetic of the geom. If 'along' The label will be written along the edge direction. If 'across' the label will be written across the edge direction.
force_flip  Logical. If angle_calc is either 'along' or 'across' should the label be flipped if it is on it's head. Default to TRUE.
label_dodge  A `grid::unit()` giving a fixed vertical shift to add to the label in case of angle_calc is either 'along' or 'across'
label_push  A `grid::unit()` giving a fixed horizontal shift to add to the label in case of angle_calc is either 'along' or 'across'

Aesthetics

`geom_dag_edges_link`, `geom_dag_edges_arc`, `geom_dag_edges_diagonal`, and `geom_dag_edges_fan` understand the following aesthetics. Bold aesthetics are required.

• x
• y
• xend
• yend
• edge_colour
• edge_width
• edge_linetype
• edge_alpha
• start_cap
• end_cap
• label
• label_pos
• label_size
• angle
• hjust
• vjust
• family
• fontface
• lineheight

geom_dag_edges_arc and geom_dag_edges_diagonal also require circular, but this is automatically set.

geom_dag_edges_fan requires to and from, but these are also automatically set.

Examples

```r
p <- dagify(y ~ x + z2 + w2 + w1,
            x ~ z1 + w1,
            z1 ~ w1 + v,
            z2 ~ w2 + v,
            L ~ w1 + w2) %>%
ggplot(aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_dag_point() +
  geom_dag_text() +
  theme_dag()

p + geom_dag_edges_link()
p + geom_dag_edges_arc()
p + geom_dag_edges_diagonal()
p + geom_dag_edges_fan()
```

--

DAG Labels

<table>
<thead>
<tr>
<th>DAG Labels</th>
<th>DAG labels</th>
</tr>
</thead>
</table>

Description

Label or otherwise retrieve labels from objects of either class tidy_dagitty or dagitty

Usage

```
label(x) <- value

## S3 replacement method for class 'dagitty'
label(x) <- value

## S3 replacement method for class 'tidy_dagitty'
label(x) <- value

dag_label(.tdy_dag, labels = NULL)

label(.tdy_dag)

has_labels(.tdy_dag)
```
dagify

Arguments

- `x` an object of either class tidy_dagitty or dagitty
- `value` a character vector
- `.tdy_dag` an object of class tidy_dagitty
- `labels` a character vector

Value

`label` returns the label attribute of `x`

Examples

```r
labelled_dag <- dagify(y ~ z, x ~ z) %>%
  tidy_dagitty() %>%
  dag_label(labels = c("x" = "exposure", "y" = "outcome", "z" = "confounder"))
has_labels(labelled_dag)
```

---

**dagify**

Create a dagitty DAG using R-like syntax

Description

dagify() creates dagitty DAGs using a more R-like syntax. It currently accepts formulas in the usual R style, e.g. `y ~ x + z`, which gets translated to `y <- \{x z\}`, as well as using a double tilde (`~~`) to graph bidirected variables, e.g. `x1 ~~ x2` is translated to `x1 <-> x2`.

Usage

```r
dagify(
  ..., 
  exposure = NULL, 
  outcome = NULL, 
  latent = NULL, 
  labels = NULL, 
  coords = NULL 
)
```

Arguments

- `...` formulas, which are converted to dagitty syntax
- `exposure` a character vector for the exposure (must be a variable name in the DAG)
- `outcome` a character vector for the outcome (must be a variable name in the DAG)
- `latent` a character vector for any latent variables (must be a variable name in the DAG)
- `labels` a named character vector, labels for variables in the DAG
- `coords` coordinates for the DAG nodes. Can be a named list or a data.frame with columns `x`, `y`, and `name`
Value

a dagitty DAG

See Also

dag(), coords2df(), coords2list()

Examples

dagify(y ~ x + z, x~ z)

coords <- list(
  x = c(A = 1, B = 2, D = 3, C = 3, F = 3, E = 4, G = 5, H = 5, I = 5),
  y = c(A = 0, B = 0, D = 1, C = 0, F = -1, E = 0, G = 1, H = 0, I = -1)
)

dag <- dagify(G ~~ H,
  G ~~ I,
  I ~~ G,
  H ~~ I,
  D ~ B,
  C ~ B,
  I ~ C + F,
  F ~ B,
  B ~ A,
  H ~ E,
  C ~ E + G,
  G ~ D, coords = coords)

dagitty::is.dagitty(dag)

ggdag(dag)

dag2 <- dagify(y ~ x + z2 + w2 + w1,
  x ~ z1 + w1,
  z1 ~ w1 + v,
  z2 ~ w2 + v,
  w1 ~~ w2,
  exposure = "x",
  outcome = "y")

ggdag(dag2)
**dpolyr**

## Description

Dplyr verb methods for tidy_dagitty objects.

## Usage

```r
## S3 method for class 'tidy_dagitty'
select(.data, ...)

## S3 method for class 'tidy_dagitty'
filter(.data, ...)

## S3 method for class 'tidy_dagitty'
mutate(.data, ...)

## S3 method for class 'tidy_dagitty'
summarise(.data, ...)

## S3 method for class 'tidy_dagitty'
distinct(.data, ..., .keep_all = FALSE)

## S3 method for class 'tidy_dagitty'
arrange(.data, ...)

## S3 method for class 'tidy_dagitty'
group_by(.data, ...)

## S3 method for class 'tidy_dagitty'
ungroup(.data, ...)

## S3 method for class 'tidy_dagitty'
transmute(.data, ...)

## S3 method for class 'tidy_dagitty'
distinct(.data, ..., .keep_all = FALSE)

## S3 method for class 'tidy_dagitty'
full_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)

## S3 method for class 'tidy_dagitty'
inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)

## S3 method for class 'tidy_dagitty'
left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)

## S3 method for class 'tidy_dagitty'
right_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```
anti_join(x, y, by = NULL, copy = FALSE, ...)
## S3 method for class 'tidy_dagitty'
semi_join(x, y, by = NULL, copy = FALSE, ...)
## S3 method for class 'tidy_dagitty'
slice(.data, ..., .dots = list())
## S3 method for class 'tidy_dagitty'
select_(.data, ..., .dots = list())
## S3 method for class 'tidy_dagitty'
filter_(.data, ..., .dots = list())
## S3 method for class 'tidy_dagitty'
mutate_(.data, ..., .dots = list())
## S3 method for class 'tidy_dagitty'
summarise_(.data, ..., .dots = list())
## S3 method for class 'tidy_dagitty'
arrange_(.data, ..., .dots = list())
## S3 method for class 'tidy_dagitty'
slice_(.data, ..., .dots = list())

Arguments

.data data object of class tidy_dagitty
... other arguments passed to the dplyr function
.dots, x, y, by, copy, suffix, .keep_all
  see corresponding function in package dplyr

Examples

library(dplyr)
tidy_dagitty(m_bias()) %>%
  group_by(name) %>%
  summarize(n = n())
**Equivalent DAGs and Classes**

**Description**

Returns a set of complete partially directed acyclic graphs (CPDAGs) given an input DAG. CPDAGs are Markov equivalent to the input graph. See `dagitty::equivalentDAGs()` for details. `node_equivalent_dags()` returns a set of DAGs, while `node_equivalent_class()` tags reversible edges. `ggdag_equivalent_dags()` plots all equivalent DAGs, while `ggdag_equivalent_class()` plots all reversible edges as undirected.

**Usage**

```r
node_equivalent_dags(.dag, n = 100, layout = "auto", ...)

ggdag_equivalent_dags(
  .tdy_dag,
  ..., 
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = "black",
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL
)

node_equivalent_class(.dag, layout = "auto")

ggdag_equivalent_class(
  .tdy_dag,
  expand_x = expansion(c(0.1, 0.1)),
  expand_y = expansion(c(0.1, 0.1)),
  breaks = ggplot2::waiver(),
  ..., 
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL
)
```

**Arguments**

- `.dag` input graph, an object of class `tidy_dagitty` or `dagitty`
- `n` maximal number of returned graphs.
layout  
a layout available in ggraph. See \texttt{ggraph::create_layout()} for details.

...  
optional arguments passed to \texttt{ggraph::create_layout()}

.tdy_dag  
an object of class tidy_dagitty or dagitty

node.size  
size of DAG node

text.size  
size of DAG text

label.size  
size of label text

text.col  
color of DAG text

label.col  
color of label text

node  
logical. Should nodes be included in the DAG?

stylized  
logical. Should DAG nodes be stylized? If so, use \texttt{geom_dag_nodes} and if not use \texttt{geom_dag_point}

text  
logical. Should text be included in the DAG?

use.labels  
a string. Variable to use for \texttt{geom_dag_repel_label()}. Default is \texttt{NULL}.

expand_x  
Vector of range expansion constants used to add some padding around the data, to ensure that they are placed some distance away from the axes. Use the convenience function \texttt{ggplot2::expansion()} to generate the values for the expand argument.

expand_y  
Vector of range expansion constants used to add some padding around the data, to ensure that they are placed some distance away from the axes. Use the convenience function \texttt{ggplot2::expansion()} to generate the values for the expand argument.

breaks  
One of:

\begin{itemize}
  \item \texttt{NULL} for no breaks
  \item \texttt{waiver()} for the default breaks computed by the transformation object
  \item A numeric vector of positions
  \item A function that takes the limits as input and returns breaks as output
\end{itemize}

Value

a tidy_dagitty with at least one DAG, including a dag column to identify graph set for equivalent DAGs or a reversible column for equivalent classes, or a ggplot

Examples

g_ex <- dagify(y ~ x + z, x ~ z)
g_ex %>% node_equivalent_class()
g_ex %>% ggdag_equivalent_dags()
Exogenous Variables

Description

`node_exogenous` tags exogenous variables given an exposure and outcome. `ggdag_exogenous` plots all exogenous variables. See `dagitty::exogenousVariables()` for details.

Usage

```r
node_exogenous(.dag, ...)
```

```r
ggdag_exogenous(
  .tdy_dag,
  ..., 
  node_size = 16,
  text_size = 3.88,
  edge_type = "link_arc",
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL
)
```

Arguments

- `.dag, .tdy_dag`: input graph, an object of class `tidy_dagitty` or `dagitty`
- `...`: additional arguments passed to `tidy_dagitty()`
- `node_size`: size of DAG node
- `text_size`: size of DAG text
- `edge_type`: a character vector, the edge geom to use. One of: "link_arc", which accounts for directed and bidirected edges, "link", "arc", or "diagonal"
- `label_size`: size of label text
- `text_col`: color of DAG text
- `label_col`: color of label text
- `node`: logical. Should nodes be included in the DAG?
- `stylized`: logical. Should DAG nodes be stylized? If so, use `geom_dag_nodes` and if not use `geom_dag_point`
- `text`: logical. Should text be included in the DAG?
- `use_labels`: a string. Variable to use for `geom_dag_repel_label()`. Default is `NULL`. 
Value

a tidy_dagitty with an exogenous column for exogenous variables or a ggplot

Examples

dag <- dagify(y ~ x1 + x2 + x3, b ~ x1 + x2)
ggdag_exogenous(dag)
node_exogenous(dag)

Description

expand_plot() is a convenience function that expands the scales of a ggplot, as the large node sizes in a DAG will often get clipped in themes that don’t have DAGs in mind.

Usage

expand_plot(
  expand_x = expansion(c(0.1, 0.1)),
  expand_y = expansion(c(0.1, 0.1))
)

Arguments

expand_x, expand_y

Fortify a tidy_dagitty object for ggplot2

Description

Fortify a tidy_dagitty object for ggplot2

Usage

## S3 method for class 'tidy_dagitty'
fortify(model, data = NULL, ...)

## S3 method for class 'dagitty'
fortify(model, data = NULL, ...)
geom_dag Collider Edges

Arguments

- **model**: an object of class tidy_dagitty or dagitty
- **data**: (not used)
- **...**: (not used)

Description

Adjusting for a collider activates pathways between the parent of the collider. This geom adds a curved edge between any such parent nodes.

Usage

```
geom_dag_collider_edges(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ..., size = 0.6,
  curvature = 0.5,
  angle = 90,
  ncp = 5,
  arrow = NULL,
  lineend = "butt",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

- **mapping**: Set of aesthetic mappings created by `aes()` or `aes()`. If specified and `inherit.aes` = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
- **data**: The data to be displayed in this layer. There are three options:
  - If NULL, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).
stat
The statistical transformation to use on the data for this layer, as a string.

position
Position adjustment, either as a string, or the result of a call to a position adjustment function.

...
Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

size
a numeric vector of length 1. Edge width

curvature
A numeric value giving the amount of curvature. Negative values produce left-hand curves, positive values produce right-hand curves, and zero produces a straight line.

angle
A numeric value between 0 and 180, giving an amount to skew the control points of the curve. Values less than 90 skew the curve towards the start point and values greater than 90 skew the curve towards the end point.

ncp
The number of control points used to draw the curve. More control points creates a smoother curve.

arrow
specification for arrow heads, as created by arrow().

lineend
Line end style (round, butt, square).

na.rm
If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders().

Examples

library(dagitty)
dagify(m ~ a + b, x ~ a, y ~ b) %>%
tidy_dagitty() %>%
control_for("m") %>%
ggplot(aes(x = x, y = y, xend = xend, yend = yend, shape = adjusted)) +
geom_dag_edges() +
geom_dag Collider_edges() +
geom_dag_point() +
geom_dag_text() +
theme_dag() +
scale_adjusted()
**Description**

Directed and bidirected DAG edges

**Usage**

```r
gem_dag_edges(
  mapping = NULL,
  data_directed = filter_direction("->"),
  data_bidirected = filter_direction("<->"),
  curvature = 0.3,
  arrow_directed = grid::arrow(length = grid::unit(5, "pt"), type = "closed"),
  arrow_bidirected = grid::arrow(length = grid::unit(5, "pt"), ends = "both", type = "closed"),
  position = "identity",
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE,
  fold = FALSE,
  ...
)
```

**Arguments**

- **mapping**
  Set of aesthetic mappings created by `aes()` or `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

- **data_directed, data_bidirected**
  The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame., and will be used as the layer data.

- **curvature**
  The bend of the curve. 1 approximates a halfcircle while 0 will give a straight line. Negative number will change the direction of the curve. Only used if `layout.circular = FALSE`.

- **arrow_directed, arrow_bidirected**
  Specification for arrow heads, as created by `arrow()`

- **position**
  Position adjustment, either as a string, or the result of a call to a position adjustment function.

- **na.rm**
  If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values
show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders().

fold Logical. Should arcs appear on the same side of the nodes despite different directions. Default to FALSE.

... Other arguments passed to ggraph::geom_edge_*()

Aesthetics

geom_dag_edges understand the following aesthetics. Bold aesthetics are required.

• x
• y
• xend
• yend
• edge_colour
• edge_width
• edge_linetype
• edge_alpha
• start_cap
• end_cap
• label
• label_pos
• label_size
• angle
• hjust
• vjust
• family
• fontface
• lineheight

geom_dag_edges also uses geom_dag_edges_arc, which requires the circular aesthetic, but this is automatically set.
Examples

dagify(y ~ x + z2 + w2 + w1,
    x ~ z1 + w1,
    z1 ~ w1 + v,
    z2 ~ w2 + v,
    w1 ~~ w2) %>%
  ggplot(aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_dag_edges() +
  geom_dag_point() +
  geom_dag_text() +
  theme_dag()

Description

Node text

Usage

geom_dag_text(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  parse = FALSE,
  nudge_x = 0,
  nudge_y = 0,
  check_overlap = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = NA
)

Arguments

mapping Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes()}. If specified and \texttt{inherit.aes} = \texttt{TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply \texttt{mapping} if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}. A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{fortify()} for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).

**stat**
The statistical transformation to use on the data for this layer, as a string.

**position**
Position adjustment, either as a string, or the result of a call to a position adjustment function. Cannot be jointly specified with `nudge_x` or `nudge_y`.

**...**
Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

**parse**
If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath.

**nudge_x**
Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with `position`.

**nudge_y**
Horizontal and vertical adjustment to nudge labels by. Useful for offsetting text from points, particularly on discrete scales. Cannot be jointly specified with `position`.

**check_overlap**
If TRUE, text that overlaps previous text in the same layer will not be plotted. `check_overlap` happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling `geom_label()` or `geom_text()`.

**na.rm**
If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

**show.legend**
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

**inherit.aes**
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

### Aesthetics

`geom_dag_text` understand the following aesthetics (required aesthetics are in bold):

- x
- y
- label
- alpha
- angle
- colour
- family
- fontface
- group
Examples

g <- dagify(m ~ x + y, y ~ x)
g %>%
tidy_dagitty() %>%
ggplot(aes(x = x, y = y, xend = xend, yend = yend)) +
geom_dag_point() +
geom_dag_edges() +
geom_dag_text() +
theme_dag()

Description

ggdag() is a wrapper to quickly plot DAGs.

Usage

ggdag(
  .tdy_dag, 
  ..., 
  edge_type = "link_arc", 
  node_size = 16, 
  text_size = 3.88, 
  label_size = text_size, 
  text_col = "white", 
  label_col = "black", 
  node = TRUE, 
  stylized = FALSE, 
  text = TRUE, 
  use_labels = NULL 
)

Arguments

  .tdy_dag       input graph, an object of class tidy_dagitty or dagitty
  ...           additional arguments passed to tidy_dagitty()
  edge_type      a character vector, the edge geom to use. One of: "link_arc", which accounts for
directed and bidirected edges, "link", "arc", or "diagonal"
node_size  size of DAG node

text_size  size of DAG text

label_size  size of label text

text_col  color of DAG text

label_col  color of label text

node  logical. Should nodes be included in the DAG?

stylized  logical. Should DAG nodes be stylized? If so, use geom_dag_nodes and if not use geom_dag_point

text  logical. Should text be included in the DAG?

use_labels  a string. Variable to use for geom_dag_repel_label(). Default is NULL.

Value

a ggplot

See Also

ggdag_classic()

Examples

dag <- dagify(y ~ x + z2 + w2 + w1,
    x ~ z1 + w1,
    z1 ~ w1 + v,
    z2 ~ w2 + v,
    w1 ~ w2)

ggdag(dag)
ggdag(dag) + theme_dag_blank()

ggdag(dagitty::randomDAG(5, .5))

---

**ggdag_classic**

Quickly plot a DAG in ggplot2

Description

ggdag_classic() is a wrapper to quickly plot DAGs in a more traditional style.
ggdag_classic

Usage

```r
ggdag_classic(  
  .tdy_dag,  
  ...,  
  size = 8,  
  label_rect_size = NULL,  
  text_label = "name",  
  text_col = "black"  
)
```

Arguments

- `.tdy_dag` input graph, an object of class `tidy_dagitty` or `dagitty`
- `...` additional arguments passed to `tidy_dagitty()`
- `size` text size, with a default of 8.
- `label_rect_size` specify the fontsize argument in `ggraph::label_rect`; default is `NULL`, in which case it is scaled relative to `size`
- `text_label` text variable, with a default of "name"
- `text_col` text color, with a default of "black"

Value

a `ggplot`

See Also

`ggdag()`

Examples

```r
dag <- dagify(y ~ x + z2 + w2 + w1,  
  x ~ z1 + w1,  
  z1 ~ w1 + v,  
  z2 ~ w2 + v,  
  w1 ~~ w2)

ggdag_classic(dag)  
ggdag_classic(dag) + theme_dag_blank()  

ggdag_classic(dagitty::randomDAG(5, .5))
```
### ggplot.tidy_dagitty

Create a new ggplot

**Description**

Create a new ggplot

**Usage**

```r
## S3 method for class 'tidy_dagitty'
ggplot(data = NULL, mapping = aes(), ...)

## S3 method for class 'dagitty'
ggplot(data = NULL, mapping = aes(), ...)
```

**Arguments**

- `data` : Default dataset to use for plot. If not already a data.frame, will be converted to one by `fortify()`. If not specified, must be supplied in each layer added to the plot.
- `mapping` : Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
- `...` : Other arguments passed on to methods. Not currently used.

### ggrepel functions

**Repulsive textual annotations**

**Description**

These functions are minor modifications of those in the ggrepel package. `geom_dag_text_repel` adds text directly to the plot. `geom_dag_label_repel` draws a rectangle underneath the text, making it easier to read. The text labels repel away from each other and away from the data points.

**Usage**

```r
dag_text_repel(
  mapping = NULL,
  data = NULL,
  parse = FALSE,
  ...,
  box.padding = 0.35,
  point.padding = 1.5,
  segment.color = "#666666",
  fontface = "bold",
  segment.size = 0.5,
)```
ggrepel functions

arrow = NULL,
force = 1,
max.iter = 2000,
nudge_x = 0,
nudge_y = 0,
na.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE
}

geom_dag_label_repel(
  mapping = NULL,
  data = NULL,
  parse = FALSE,
  ...,
  box.padding = grid::unit(0.35, "lines"),
  label.padding = grid::unit(0.25, "lines"),
  point.padding = grid::unit(1.5, "lines"),
  label.r = grid::unit(0.15, "lines"),
  label.size = 0.25,
  segment.color = "grey50",
  segment.size = 0.5,
  arrow = NULL,
  force = 1,
  max.iter = 2000,
  nudge_x = 0,
  nudge_y = 0,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

Arguments

mapping Set of aesthetic mappings created by \texttt{aes} or \texttt{aes_}. If specified and \texttt{inherit.aes} = TRUE (the default), is combined with the default mapping at the top level of the plot. You only need to supply mapping if there isn’t a mapping defined for the plot.

data A data frame. If specified, overrides the default data frame defined at the top level of the plot.

parse If TRUE, the labels will be parsed into expressions and displayed as described in \texttt{plotmath}

... other arguments passed on to \texttt{layer}. There are three types of arguments you can use here:

- Aesthetics: to set an aesthetic to a fixed value, like \texttt{colour = "red"} or \texttt{size = 3}.
- Other arguments to the layer, for example you override the default \texttt{stat} associated with the layer.
- Other arguments passed on to the stat.

**box.padding** Amount of padding around bounding box, as unit or number. Defaults to 0.25. (Default unit is lines, but other units can be specified by passing `unit(x,"units")`).

**point.padding** Amount of padding around labeled point, as unit or number. Defaults to 0. (Default unit is lines, but other units can be specified by passing `unit(x,"units")`).

**segment.color, segment.size** See `ggrepel::geom_text_repel()`

**fontface** A character vector. Default is "bold"

**arrow** specification for arrow heads, as created by `arrow`

**force** Force of repulsion between overlapping text labels. Defaults to 1.

**max.iter** Maximum number of iterations to try to resolve overlaps. Defaults to 10000.

**nudge_x** Horizontal and vertical adjustments to nudge the starting position of each text label.

**nudge_y** Horizontal and vertical adjustments to nudge the starting position of each text label.

**na.rm** If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.

**show.legend** logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

**inherit.aes** If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders`.

**label.padding** Amount of padding around label, as unit or number. Defaults to 0.25. (Default unit is lines, but other units can be specified by passing `unit(x,"units")`).

**label.r** Radius of rounded corners, as unit or number. Defaults to 0.15. (Default unit is lines, but other units can be specified by passing `unit(x,"units")`).

**label.size** Size of label border, in mm.

### Examples

```r
library(ggraph)
library(ggrepel)
library(dagitty)

# Create a DAG

g <- dagify(m ~ x + y,
            y ~ x,
            exposure = "x",
            outcome = "y",
            latent = "m",
            labels = c("x" = "Exposure", "y" = "Outcome", "m" = "Collider"))

g %>% tidy_dagitty() %>%
  ggplot(aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_dag_edges() +
  geom_dag_point() +
  geom_dag_text_repel(aes(label = name), show.legend = FALSE) +
  theme_dag()

g %>% tidy_dagitty() %>%
```
Instrumental Variables

```
dag_label(labels = c("x" = "This is the exposure",
                 "y" = "Here’s the outcome",
                 "m" = "Here is where they collide")) %>%
ggplot(aes(x = x, y = y, xend = xend, yend = yend)) +
geom_dag_edges() +
geom_dag_point() +
geom_dag_text() +
geom_dag_label_repel(aes(label = label, fill = label),
                     col = "white", show.legend = FALSE) +
theme_dag()
```

---

**Instrumental Variables**

*Find Instrumental Variables*

### Description

`node_instrumental` tags instrumental variables given an exposure and outcome. `ggdag_instrumental` plots all instrumental variables. See `dagitty::instrumentalVariables()` for details.

### Usage

```
node_instrumental(.dag, exposure = NULL, outcome = NULL, ...)
```

```
ggdag_instrumental(
  .tdy_dag,
  exposure = NULL,
  outcome = NULL,
  ..., 
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL
)
```

### Arguments

- `.dag, .tdy_dag` input graph, an object of class `tidy_dagitty` or `dagitty`
- `exposure` character vector of length 1, name of exposure variable. Default is `NULL`, in which case it will check the input DAG for exposure.
- `outcome` character vector of length 1, name of exposure variable. Default is `NULL`, in which case it will check the input DAG for exposure.
is.tidy_dagitty

Additional arguments passed to `tidy_dagitty()`

- `node_size` (size of DAG node)
- `text_size` (size of DAG text)
- `label_size` (size of label text)
- `text_col` (color of DAG text)
- `label_col` (color of label text)
- `node` (logical. Should nodes be included in the DAG?)
- `stylized` (logical. Should DAG nodes be stylized? If so, use `geom_dag_nodes` and if not use `geom_dag_point`)
- `text` (logical. Should text be included in the DAG?)
- `use_labels` (a string. Variable to use for `geom_dag_repel_label()`. Default is NULL.)

Value

A `tidy_dagitty` with an instrumental column for instrumental variables or a ggplot

Examples

```r
library(dagitty)

node_instrumental(dagitty("dag{ i->x->y; x<->y }", "x", "y")
ggdag_instrumental(dagitty("dag{ i->x->y; i2->x->y; x<->y }", "x", "y")
```

Description

Test for object class for `tidy_dagitty`

Usage

`is.tidy_dagitty(x)`

Arguments

- `x` (object to be tested)
is_confounder

Assess if a variable confounds a relationship

Description

Assess if a variable confounds a relationship

Usage

is_confounder(.tdy_dag, z, x, y, direct = FALSE)

Arguments

.args     input graph, an object of class tidy_dagitty or dagitty
z          a character vector, the potential confounder
x, y      a character vector, the variables z may confound.
direct    logical. Only consider direct confounding? Default is FALSE

Value

Logical. Is the variable a confounder?

Examples

dag <- dagify(y ~ z, x ~ z)

is_confounder(dag, "z", "x", "y")
is_confounder(dag, "x", "z", "y")

Nodes

DAG Nodes

descr

Description

gem_dag_node and gem_dag_point are very similar to ggplot2::geom_point but with a few defaults changed. geom_dag_node is slightly stylized and includes an internal white circle, while geom_dag_point plots a single point.
Usage

dag_node(
    mapping = NULL,
    data = NULL,
    position = "identity",
    ..., 
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE
)

dag_point(
    mapping = NULL,
    data = NULL,
    position = "identity",
    ..., 
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE
)

Arguments

mapping  Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data      The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
position  Position adjustment, either as a string, or the result of a call to a position adjustment function.
...       Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
na.rm     If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. `borders()`.

### Aesthetics

`geom_dag_node` and `geom_dag_point` understand the following aesthetics (required aesthetics are in bold):

- x
- y
- alpha
- colour
- fill
- shape
- size
- stroke
- filter

`geom_dag_node` also accepts:

- internal_colour

### Examples

```r
g <- dagify(m ~ x + y, y ~ x)
p <- g %>%
tidy_dagitty() %>%
ggplot(aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_dag_edges() +
  theme_dag()

p +
  geom_dag_node() +
  geom_dag_text()

p +
  geom_dag_point() +
  geom_dag_text()
```
Find Pathways Between Variables

Description

node_paths finds the pathways between a given exposure and outcome. ggdag_paths plots all pathways. See dagitty::paths() for details.

Usage

dag_paths(
  .dag,
  from = NULL,
  to = NULL,
  adjust_for = NULL,
  directed = FALSE,
  paths_only = FALSE,
  ...
)

ggdag_paths(
  .tdy_dag,
  from = NULL,
  to = NULL,
  adjust_for = NULL,
  directed = FALSE,
  shadow = FALSE,
  ...
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  stylized = FALSE,
  text = TRUE,
  use_labels = NULL
)

ggdag_paths_fan(
  .tdy_dag,
  from = NULL,
  to = NULL,
  adjust_for = NULL,
  directed = FALSE,
  ...
  shadow = FALSE,
spread = 0.7,
node_size = 16,
text_size = 3.88,
label_size = text_size,
text_col = "white",
label_col = text_col,
node = TRUE,
stylized = FALSE,
text = TRUE,
use_labels = NULL
)

Arguments

.dag, .tdy_dag  input graph, an object of class tidy_dagitty or dagitty
from       character vector of length 1, name of exposure variable. Default is NULL, in which case it will check the input DAG for exposure.
to         character vector of length 1, name of exposure variable. Default is NULL, in which case it will check the input DAG for exposure.
adjust_for character vector, a set of variables to control for. Default is NULL.
directed   logical. Should only directed paths be shown?
paths_only logical. Should only open paths be returned? Default is FALSE, which includes every variable and edge in the DAG regardless if they are part of the path.
...           additional arguments passed to tidy_dagitty()
shadow      logical. Show edges not in path? Ignored if paths_only is TRUE
node_size   size of DAG node
text_size   size of DAG text
label_size  size of label text
text_col    color of DAG text
label_col   label color
node        logical. Should nodes be included in the DAG?
stylized    logical. Should DAG nodes be stylized? If so, use geom_dag_nodes and if not use geom_dag_point
text        logical. Should text be included in the DAG?
use_labels  a string. Variable to use for geom_dag_repel_label(). Default is NULL.
spread      the width of the fan spread

Value

a tidy_dagitty with a path column for path variables and a set grouping column or a ggplot
Quick Plots for Common DAGs

Examples

```r
confounder_triangle(x_y_associated = TRUE) %>%
  ggdag_paths(from = "x", to = "y")

butterfly_bias(x_y_associated = TRUE) %>%
  ggdag_paths_fan(shadow = TRUE)
```

```r
print.tidy_dagitty
Print a tidy_dagitty
```

Description

Print a tidy_dagitty

Usage

```r
## S3 method for class 'tidy_dagitty'
print(x, ...)
```

Arguments

- `x`: an object of class tidy_dagitty
- `...`: optional arguments passed to `print()`

Quick Plots for Common DAGs

Quickly create a DAGs with common structures of bias

Description

base functions create an object of class dagitty; ggdag_* functions are wrappers that also call ggdag() on the dagitty object.

Usage

```r
m_bias(
  x = NULL,
  y = NULL,
  a = NULL,
  b = NULL,
  m = NULL,
  x_y_associated = FALSE
)
```
butterfly_bias(
  x = NULL,
  y = NULL,
  a = NULL,
  b = NULL,
  m = NULL,
  x_y_associated = FALSE
)

confounder_triangle(x = NULL, y = NULL, z = NULL, x_y_associated = FALSE)

collider_triangle(x = NULL, y = NULL, m = NULL, x_y_associated = FALSE)

mediation_triangle(x = NULL, y = NULL, m = NULL, x_y_associated = FALSE)

ggdag_m_bias(
  x = NULL,
  y = NULL,
  a = NULL,
  b = NULL,
  m = NULL,
  x_y_associated = FALSE,
  edge_type = "link_arc",
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
  styled = FALSE,
  text = TRUE,
  use_labels = NULL
)

ggdag_butterfly_bias(
  x = NULL,
  y = NULL,
  a = NULL,
  b = NULL,
  m = NULL,
  x_y_associated = FALSE,
  edge_type = "link_arc",
  node_size = 16,
  text_size = 3.88,
  label_size = text_size,
  text_col = "white",
  label_col = text_col,
  node = TRUE,
Quick Plots for Common DAGs

```r
stylized = FALSE,
   text = TRUE,
   use_labels = NULL
)

ggdag_confounder_triangle(
   x = NULL,
   y = NULL,
   z = NULL,
   x_y_associated = FALSE,
   edge_type = "link_arc",
   node_size = 16,
   text_size = 3.88,
   label_size = text_size,
   text_col = "white",
   label_col = text_col,
   node = TRUE,
   styled = FALSE,
   text = TRUE,
   use_labels = NULL
)

ggdag Collider_triangle(
   x = NULL,
   y = NULL,
   m = NULL,
   x_y_associated = FALSE,
   edge_type = "link_arc",
   node_size = 16,
   text_size = 3.88,
   label_size = text_size,
   text_col = "white",
   label_col = text_col,
   node = TRUE,
   styled = FALSE,
   text = TRUE,
   use_labels = NULL
)

ggdag_mediation_triangle(
   x = NULL,
   y = NULL,
   m = NULL,
   x_y_associated = FALSE,
   edge_type = "link_arc",
   node_size = 16,
   text_size = 3.88,
   label_size = text_size,
   text_col = "white",
   label_col = text_col,
   node = TRUE,
   styled = FALSE,
   text = TRUE,
   use_labels = NULL
)
```
text_col = "white",
label_col = text_col,
node = TRUE,
stylized = FALSE,
text = TRUE,
use_labels = NULL
)

Arguments

x, y, a, b, m, z  Character vector. Optional label. Default is NULL
x_y_associated Logical. Are x and y associated? Default is FALSE.
edge_type       a character vector, the edge geom to use. One of: "link_arc", which accounts for
directed and bidirected edges, "link", "arc", or "diagonal"
node_size       size of DAG node
text_size       size of DAG text
label_size      size of label text
text_col        color of DAG text
label_col       color of label text
node            logical. Should nodes be included in the DAG?
stylized        logical. Should DAG nodes be stylized? If so, use geom_dag_nodes and if not
                use geom_dag_point
text            logical. Should text be included in the DAG?
use_labels      a string. Variable to use for geom_dag_repel_label(). Default is NULL.

Value

a DAG of class dagitty or a ggplot

Examples

m_bias() %>% ggdag_adjust("m")
ggdag_confounder_triangle()

Description

remove_axes() and remove_grid() are convenience functions that removes the axes and grids
from a ggplot, respectively. This is useful when you want to use an existing theme, e.g. those
included in ggplot2, for a DAG.
Usage

```r
remove_axes()
remove_grid()
```

Examples

```r
ggdag(confounder_triangle()) +
theme_bw() +
remove_axes()
```

---

**scale_adjusted**

Common scale adjustments for DAGs

Description

*scale_adjusted()* is a convenience function that implements ways of visualizing adjustment for
a variable. By convention, a square shape is used to indicate adjustment and a circle when not ad-
justed. Arrows out of adjusted variables are often eliminated or de-emphasized, and *scale_adjusted()*
uses a lower alpha for these arrows. When adjusting a collider, a dashed line is sometimes used to
demarcate opened pathways, and *scale_adjusted()* does this whenever *geom_dag_collider_edges()*
is used. *scale_dag()* is deprecated in favor of *scale_adjusted()*.

Usage

```r
scale_adjusted()
```

```r
scale_dag(breaks = ggplot2::waiver())
```

Arguments

<table>
<thead>
<tr>
<th>breaks</th>
<th>One of:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• NULL for no breaks</td>
</tr>
<tr>
<td></td>
<td>• waiver() for the default breaks computed by the transformation object</td>
</tr>
<tr>
<td></td>
<td>• A numeric vector of positions</td>
</tr>
<tr>
<td></td>
<td>• A function that takes the limits as input and returns breaks as output</td>
</tr>
</tbody>
</table>
**simulate_data**

*Simulate Data from Structural Equation Model*

**Description**

This is a thin wrapper for the `simulateSEM()` function in `dagitty` that works with tidied `dagitty` objects. It treats the input DAG as a structural equation model, generating random path coefficients and simulating corresponding data. See `dagitty::simulateSEM()` for details.

**Usage**

```r
simulate_data(
  .tdy_dag,
  b.default = NULL,
  b.lower = -0.6,
  b.upper = 0.6,
  eps = 1,
  N = 500,
  standardized = TRUE
)
```

**Arguments**

- `.tdy_dag` the input DAG, which can be a `tidy_dagitty` or `dagitty` object.
- `b.default` default path coefficient applied to arrows for which no coefficient is defined in the model syntax.
- `b.lower` lower bound for random path coefficients, applied if `b.default = NULL`.
- `b.upper` upper bound for path coefficients.
- `eps` residual variance (only meaningful if `standardized=FALSE`).
- `N` number of samples to generate.
- `standardized` whether a standardized output is desired (all variables have variance 1).

**Value**

a `tbl` with N values for each variable in `.tdy_dag`

**Examples**

```r
dagify(y ~ z, x ~ z) %>%
tidy_dagitty() %>%
simulate_data()
```
### tbl_df.tidy_daggity

*Convert a tidy_dagitty object to tbl_df*

#### Description

Convert a tidy_dagitty object to tbl_df

#### Usage

```r
tbl_df.tidy_daggity(.tdy_dag)
```

#### Arguments

- `.tdy_dag` an object of class tidy_dagitty

### Test if Variable Is Collider

*Detecting colliders in DAGs*

#### Description

Detecting colliders in DAGs

#### Usage

- `is_collider(.dag, .var, downstream = TRUE)`
- `is_downstreamCollider(.dag, .var)`

#### Arguments

- `.dag` an input graph, an object of class tidy_dagitty or dagitty
- `.var` a character vector of length 1, the potential collider to check
- `downstream` Logical. Check for downstream colliders? Default is TRUE.

#### Value

Logical. Is the variable a collider or downstream collider?
Examples

```r
dag <- dagify(m ~ x + y, m_jr ~ m)
is_collider(dag, "m")
is_downstream_collider(dag, "m_jr")

# a downstream collider is also treated as a collider
is_collider(dag, "m_jr")

# but a direct collider is not treated as a downstream collider
is_downstream_collider(dag, "m")
```

Description

Minimalist DAG themes

Usage

```r
theme_dag_blank(base_size = 12, base_family = "", ...)  
theme_dag(base_size = 12, base_family = "", ...)  
theme_dag_grid(base_size = 12, base_family = "", ...)  
```

Arguments

- `base_size` base font size, given in pts.
- `base_family` base font family
- `...` additional arguments passed to `theme()`

Examples

```r
ggdag(m_bias()) + theme_dag_blank() # the default  
```
theme_dag_grey  

**Simple grey themes for DAGs**

**Description**
Simple grey themes for DAGs

**Usage**
```
theme_dag_grey(base_size = 12, base_family = "", ...)  
theme_dag_gray(base_size = 12, base_family = "", ...)  
theme_dag_grey_grid(base_size = 12, base_family = "", ...)  
theme_dag_gray_grid(base_size = 12, base_family = "", ...)  
```

**Arguments**
- `base_size` base font size, given in pts.  
- `base_family` base font family  
- `...` additional arguments passed to `theme()`

**Examples**
```
ggdag(m_bias()) + theme_dag_grey()  
```

tidy_dagitty  

**Tidy a dagitty object**

**Description**
Tidy a dagitty object

**Usage**
```
tidy_dagitty(.dagitty, seed = NULL, layout = "nicely", ...)  
```

**Arguments**
- `.dagitty` a dagitty  
- `seed` a numeric seed for reproducible layout generation  
- `layout` a layout available in ggraph. See `ggraph::create_layout()` for details.  
- `...` optional arguments passed to `ggraph::create_layout()`
**Variable Status**

Find variable status

**Description**

Detects variable status given a DAG (exposure, outcome, latent). See `dagitty::VariableStatus()` for details.

**Usage**

```r
node_status(.dag, as_factor = TRUE, ...)
```

```r
ggdag_status(
  .tdy_dag,
  ...
)
```

```r
dag <- dagitty("dag {
  Y <- X <- Z1 <- V -> Z2 -> Y
  Z1 <- W1 <-> W2 -> Z2
  X <- W1 -> Y
  X <- W2 -> Y
  X [exposure]
  Y [outcome]
}")
```

```r
tidy_dagitty(dag)
```

```r
tidy_dagitty(dag, layout = "fr") %>%
  ggplot(aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_dag_node() +
  geom_dag_text() +
  geom_dag_edges() +
  theme_dag()
```
Variables

```r
stylized = FALSE,
text = TRUE,
use_labels = NULL
```

**Arguments**

- `.dag`, `.tdy_dag` input graph, an object of class `tidy_dagitty` or `dagitty`
- `as_factor` treat status variable as factor
- `...` additional arguments passed to `tidy_dagitty()`
- `edge_type` a character vector, the edge geom to use. One of: "link_arc", which accounts for directed and bidirected edges, "link", "arc", or "diagonal"
- `node_size` size of DAG node
- `text_size` size of DAG text
- `label_size` size of label text
- `text_col` color of DAG text
- `label_col` color of label text
- `node` logical. Should nodes be included in the DAG?
- `stylized` logical. Should DAG nodes be stylized? If so, use `geom_dag_nodes` and if not use `geom_dag_point`
- `text` logical. Should text be included in the DAG?
- `use_labels` a string. Variable to use for `geom_dag_repel_label()`. Default is `NULL`.

**Details**

`node_collider` tags variable status and `ggdag_collider` plots all variable statuses.

**Value**

A `tidy_dagitty` with a status column for variable status or a `ggplot`

**Examples**

```r
dag <- dagify(l ~ x + y,
             y ~ x,
             exposure = "x",
             outcome = "y",
             latent = "l")
node_status(dag)
ggdag_status(dag)
```
Index

activate Collider Paths, 3
Adjust for variables, 3
adjust_for (Adjust for variables), 3
aes, 43
aes(), 33, 37, 48
aes_, 43
anti_join (tidy_dagitty (dplyr)), 26
arrange (tidy_dagitty (dplyr)), 26
arrange_.tidy_dagitty (dplyr), 26
arrow, 44
as.data.frame (tidy_dagitty), 5
as.tbl.tidy_dagitty, 13
as_tibble (tidy_dagitty)
(as.tbl.tidy_dagitty), 5
Assess d-separation between variables, 6
Assess familial relationships between variables, 9
borders, 44
borders(), 34, 38, 49
butterfly_bias (Quick Plots for Common DAGs), 52
Canonicalize DAGs, 14
collider_triangle (Quick Plots for Common DAGs), 52
Colliders, 15
cofounder_triangle (Quick Plots for Common DAGs), 52
control_for (Adjust for variables), 3
control_for(), 3
coordinates, 17
cords2df (coordinates), 17
cords2df(), 26
cords2list (coordinates), 17
cords2list(), 26
Covariate Adjustment Sets, 18
dag, 19
DAG Edges, 20
DAG Labels, 24
dag(), 26
dag_adjustment_sets (Covariate Adjustment Sets), 18
dag_label (DAG Labels), 24
dag_paths (Pathways), 50
dagify, 25
dagitty, 20
dagitty::adjustmentSets(), 18
dagitty::canonicalize(), 14
dagitty::children, 9
dagitty::dseparated(), 6
dagitty::equivalentDAGs(), 29
dagitty::exogenousVariables(), 31
dagitty::instrumentalVariables(), 45
dagitty::paths(), 50
dagitty::simulateSEM(), 57
dagitty::VariableStatus(), 61
distinct.tidy_dagitty (dplyr), 26
dplyr, 26
dplyr::as_tibble(), 6
Equivalent DAGs and Classes, 28
Exogenous Variables, 31
expand_plot, 32
filter.tidy_dagitty (dplyr), 26
filter_.tidy_dagitty (dplyr), 26
fortify, 32
fortify(), 33, 37, 42, 48
full_join.tidy_dagitty (dplyr), 26
geom_dag_collider_edges, 33
geom_dag_collider_edges(), 3, 56
geom_dag_edges, 35
geom_dag_edges_arc (DAG Edges), 20
geom_dag_edges_diagonal (DAG Edges), 20
geom_dag_edges_fan (DAG Edges), 20

geom_dag_edges_link (DAG Edges), 20
geom_dag_label_repel (ggrepel functions), 42
geom_dag_node (Nodes), 47
geom_dag_point (Nodes), 47
geom_dag_text, 37
geom_dag_text_repel (ggrepel functions), 42
ggdag, 39
ggdag(), 41
ggdag_adjacent (Assess familial relationships between variables), 9
ggdag_adjust (Adjust for variables), 3
ggdag_adjust(), 3
ggdag_adjustment_set (Covariate Adjustment Sets), 18
ggdag_ancestors (Assess familial relationships between variables), 9
ggdag_butterfly_bias (Quick Plots for Common DAGs), 52
ggdag_canonical (Canonicalize DAGs), 14
ggdag_children (Assess familial relationships between variables), 9
ggdag_classic, 40
ggdag_classic(), 40
ggdag.collider (Colliders), 15
ggdag.collider_triangle (Quick Plots for Common DAGs), 52
ggdag_confounder_triangle (Quick Plots for Common DAGs), 52
ggdag_dconnected (Assess d-separation between variables), 6
ggdag_descendants (Assess familial relationships between variables), 9
ggdag_drelationship (Assess d-separation between variables), 6
ggdag_dseparated (Assess d-separation between variables), 6
ggdag_equivalent_class (Equivalent DAGs and Classes), 28
ggdag_equivalent_dags (Equivalent DAGs and Classes), 28
ggdag_exogenous (Exogenous Variables), 31
ggdag_instrumental (Instrumental Variables), 45
ggdag_m_bias (Quick Plots for Common DAGs), 52
ggdag_markov_blanket (Assess familial relationships between variables), 9
ggdag_mediation_triangle (Quick Plots for Common DAGs), 52
ggdag_parents (Assess familial relationships between variables), 9
ggdag_paths (Pathways), 50
ggdag_paths_fan (Pathways), 50
ggdag_status (Variable Status), 61
ggplot(), 33, 37, 48
ggplot.dagitty (ggplot.tidy_dagitty), 42
ggplot.tidy_dagitty, 42
ggplot2::geom_point, 47
ggrepel::geom_text_repel(), 44
gremlins::plotmath(), 23
grid::unit(), 23
group_by.tidy_dagitty (dplyr), 26
has_labels (DAG Labels), 24
inner_join.tidy_dagitty (dplyr), 26
Instrumental Variables, 45
is.tidy_dagitty, 45
is_confounder (Test if Variable Is Collider), 58
is_confounder, 47
is_downstream_confounder (Test if Variable Is Collider), 58
label (DAG Labels), 24
label< (DAG Labels), 24
layer, 43
layer(), 34, 38, 48
left_join.tidy_dagitty (dplyr), 26
m_bias (Quick Plots for Common DAGs), 52
mediation_triangle (Quick Plots for Common DAGs), 52
mutate.tidy_dagitty (dplyr), 26
mutate_.tidy_dagitty (dplyr), 26
INDEX

node_adjacent (Assess familial relationships between variables), 9
node_ancestors (Assess familial relationships between variables), 9
node_canonical (Canonicalize DAGs), 14
node_children (Assess familial relationships between variables), 9
nodeCollider (Colliders), 15
node_dconnected (Assess d-separation between variables), 6
node_descendants (Assess familial relationships between variables), 9
node_drelationship (Assess d-separation between variables), 6
node_dseparated (Assess d-separation between variables), 6
node_equivalent_class (Equivalent DAGs and Classes), 28
node_equivalent_dags (Equivalent DAGs and Classes), 28
node_exogenous (Exogenous Variables), 31
node_instrumental (Instrumental Variables), 45
node_markov_blanket (Assess familial relationships between variables), 9
node_parents (Assess familial relationships between variables), 9
node_status (Variable Status), 61
Nodes, 47

Pathways, 50
print.tidy_dagitty, 52

Quick Plots for Common DAGs, 52

remove_axes, 55
remove_grid (remove_axes), 55
right_join.tidy_dagitty (dplyr), 26

scale_adjusted, 56
scale_dag (scale_adjusted), 56
select.tidy_dagitty (dplyr), 26

select_.tidy_dagitty (dplyr), 26
semi_join.tidy_dagitty (dplyr), 26
simulate_data, 57
slice.tidy_dagitty (dplyr), 26
slice_.tidy_dagitty (dplyr), 26
summarise.tidy_dagitty (dplyr), 26
summarise_.tidy_dagitty (dplyr), 26
tbl_df.tidy_dagitty, 58
Test if Variable Is Collider, 58
theme_dag (theme_dag_blank), 59
theme_dag_blank, 59
theme_dag_gray (theme_dag_grey), 60
theme_dag_grey (theme_dag_grey), 60
theme_dag_grey_grid (theme_dag_grey), 60
theme_dag_grid (theme_dag_blank), 59
tidy_dagitty, 60
tidygraph::as_tbl_graph(), 13
transmute.tidy_dagitty (dplyr), 26
ungroup.tidy_dagitty (dplyr), 26

Variable Status, 61