# Package 'causact'

July 25, 2020

**Type** Package

**Title** Accelerated Bayesian Analytics with DAGs

**Version** 0.3.3

**Description** Accelerate Bayesian analytics workflows in 'R' through interactive modelling, visualization, and inference. Define probabilistic graphical models using directed acyclic graphs (DAGs) as a unifying language for business stakeholders, statisticians, and programmers. This package relies on the sleek and elegant 'greta' package for Bayesian inference. 'greta', in turn, is an interface into 'TensorFlow' from 'R'. Install 'greta' using instructions available here: <http://www.causact.com/install-tensorflow-greta-and-causact.html>. See <http://github.com/flyaflya/causact> or <http://causact.com> for more documentation.

**License** MIT + file LICENSE

**URL** https://github.com/flyaflya/causact, https://causact.com

**BugReports** https://github.com/flyaflya/causact/issues

**SystemRequirements** Python and TensorFlow are needed for Bayesian inference computations; Python (>= 2.7.0) with header files and shared library; TensorFlow (= v1.14; https://www.tensorflow.org/); TensorFlow Probability (= v0.7.0; https://www.tensorflow.org/probability/)

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 3.2.0)

**Imports** DiagrammeR (>= 1.0.6), dplyr (>= 0.8.5), magrittr (>= 1.5), ggplot2 (>= 3.3.0), rlang (>= 0.4.6), greta (>= 0.3.1), purrr (>= 0.3.4), tidyr (>= 1.0.3), igraph (>= 1.2.5), stringr (>= 1.4.0), cowplot (>= 1.0.0), coda (>= 0.19.3), forcats (>= 0.5.0), htmlwidgets (>= 1.5.1), rstudioapi (>= 0.11)

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Author** Adam Fleischhacker [aut, cre, cph],
Daniela Dapena [ctb],
Rose Nguyen [ctb],
Jared Sharpe [ctb]

**Maintainer** Adam Fleischhacker <ajf@udel.edu>

**Repository** CRAN

**Date/Publication** 2020-07-25 04:40:06 UTC

# R topics documented:

addPriorGroups *Add a column to tidy dataframe of draws that groups parameters by their prior distribution. All parameters with the same prior distribution recieve the same index.*

## Description

Add a column to tidy dataframe of draws that groups parameters by their prior distribution. All parameters with the same prior distribution recieve the same index.

## Usage

```
addPriorGroups(drawsDF)
```

## Arguments

| | |
|---|---|
| drawsDF | the dataframe created by `dag_greta()` where each row represents one draw of MCMC output. Two columns are expected, `param` - the parameter name, `value` - the realized value, and a third column, `priorGroup`, is appended as an integer grouping parameters by their prior distributions. The data for this third column is stored in an environment called `cacheEnv` when the `dag_greta()` function is called. Any parameters with the same prior end up in the same prior group. Used by `dagp_plot()` to group parameters when plotted. |

## Value

a tidy dataframe of posterior draws. Useful for passing to `dagp_plot()` or for creating plots using `ggplot()`.

baseballData *Dataframe of 12,145 observations of baseball games in 2010*

## Description

Dataframe of 12,145 observations of baseball games in 2010

## Usage

```
baseballData
```

## Format

A data frame with 12145 rows and 5 variables:

**Date** date game was played

**Home** abbreviation for home team (i.e. stadium where game played)

**Visitor** abbreviation for visiting team

**HomeScore** Runs scored by the home team

**VisitorScore** Runs scored by the visiting team

---

`beachLocDF` *Dataframe where each row represents data about one of the 26 mile markers (fake) from mile 0 to mile 2.5 along the Ocean City, MD beach/boardwalk.*

---

## Description

Dataframe where each row represents data about one of the 26 mile markers (fake) from mile 0 to mile 2.5 along the Ocean City, MD beach/boardwalk.

## Usage

```
beachLocDF
```

## Format

A data frame with 26 rows and 3 variables:

**mileMarker** a number representing a location on the Ocean City beach/boardwalk.

**beachgoerProb** The probability of any Ocean City, MD beachgoer (during the hot swimming days) exiting the beach at that mile marker.

**expenseEst** The estimated annual expenses of running a business at that location on the beach. It is assumed a large portion of the expense is based on commercial rental rates at that location. More populated locations tend to have higher expenses.

`carModelDF` *Dataframe of 1000 (fake) observations of whether certain car buyers were willing to get information on a credit card speciailizing in rewards for adventure travellers.*

## Description

Dataframe of 1000 (fake) observations of whether certain car buyers were willing to get information on a credit card speciailizing in rewards for adventure travellers.

## Usage

```
carModelDF
```

## Format

A data frame with 1000 rows and 3 variables:

**customerID** a unique id of a potential credit card customer. They just bought a car and are asked if they want information on the credit card.

**carModel** The model of car purchased.

**getCard** Whether the customer expressed interest in hearing more about the card.

`chimpanzeesDF` *Data from behavior trials in a captive group of chimpanzees, housed in Lousiana. From Silk et al. 2005. Nature 437:1357-1359 and further popularized in McElreath, Richard. Statistical rethinking: A Bayesian course with examples in R and Stan. CRC press, 2020. Experiment*

## Description

Data from behavior trials in a captive group of chimpanzees, housed in Lousiana. From Silk et al. 2005. Nature 437:1357-1359 and further popularized in McElreath, Richard. Statistical rethinking: A Bayesian course with examples in R and Stan. CRC press, 2020. Experiment

## Usage

```
chimpanzeesDF
```

## Format

A data frame with 504 rows and 9 variables:

**actor** name of actor

**recipient** name of recipient (NA for partner absent condition)

**condition** partner absent (0), partner present (1)

**block** block of trials (each actor x each recipient 1 time)

**trial** trial number (by chimp = ordinal sequence of trials for each chimp, ranges from 1-72; partner present trials were interspersed with partner absent trials)

**prosoc_left** prosocial_left : 1 if prosocial (1/1) option was on left

**chose_prosoc** choice chimp made (0 = 1/0 option, 1 = 1/1 option)

**pulled_left** which side did chimp pull (1 = left, 0 = right)

**treatment** narrative description combining condition and prosoc_left that describes the side the prosical food option was on and whether a partner was present

## Source

Silk et al. 2005. Nature 437:1357-1359..

---

`corruptDF` *Dataframe of 174 observations where information on the human developmet index (HDI) and the corruption perceptions index (CPI) both exist. Each observation is a country.*

---

## Description

Dataframe of 174 observations where information on the human developmet index (HDI) and the corruption perceptions index (CPI) both exist. Each observation is a country.

## Usage

```
corruptDF
```

## Format

A data frame with 174 rows and 7 variables:

**country** country name

**region** region name as given with CPI rating

**countryCode** three letter abbreviation for country

**regionCode** four letter or less abbreviation for country

**population** 2017 country population

**CPI2017** The Corruption Perceptions Index score for 2017: A country/territory's score indicates the perceived level of public sector corruption on a scale of 0-100, where 0 means that a country is perceived as highly corrupt and a 100 means that a country is perceived as very clean.

**HDI2017** The human development index score for 2017: the Human Development Index (HDI) is a measure of achievement in the basic dimensions of human development across countries. It is an index made from a simple unweighted average of a nation's longevity, education and income and is widely accepted in development discourse.

## Source

http://www.transparency.org/cpi CPI data available from www.transparency.org/cpi. Accessed Oct 1, 2018. Consumer Perception Index 2017 by Transparency International is licensed under CC-BY- ND 4.0.

http://hdr.undp.org/en/content/human-development-index-hdi HDA data accessed on Oct 1, 2018.

https://data.worldbank.org/ Population data accessed on Oct 1, 2018.

---

`dagp_plot` *Plot posterior distribution of latent parameters in* `causact_graph` *model.*

---

## Description

The graph object should be of class `causact_graph` and created using `dag_create()`.

## Usage

```
dagp_plot(drawsDF, densityPlot = FALSE)
```

## Arguments

| | |
|---|---|
| `drawsDF` | the dataframe output of `dag_greta(mcmc=TRUE)` where each column is a parameter and each row a single draw from a representative sample. |
| `densityPlot` | If `TRUE`, each parameter gets its own density plot. If `FALSE` (recommended usage), parameters are grouped into facets based on whether they share the same prior or not. 10 and 90 percent credible intervals are displayed for the posterior distributions. |

## Value

a credible interval plot of all latent posterior distribution parameters.

## Examples

```
# A simple example
posteriorDF = data.frame(x = rnorm(100),
y = rexp(100),
z = runif(100))
posteriorDF %>%
dagp_plot(densityPlot = TRUE)

# More complicated example requiring 'greta'
## Not run:
library(greta)
# Create a 2 node graph
graph = dag_create() %>%
  dag_node("Get Card","y",
         rhs = bernoulli(theta),
         data = carModelDF$getCard) %>%
  dag_node(descr = "Card Probability by Car",label = "theta",
           rhs = beta(2,2),
           child = "y")
graph %>% dag_render()

# below requires Tensorflow installation
drawsDF = graph %>% dag_greta(mcmc=TRUE)
drawsDF %>% dagp_plot()

## End(Not run)

# A multiple plate example
library(dplyr)
poolTimeGymDF = gymDF %>%
mutate(stretchType = ifelse(yogaStretch == 1,
                            "Yoga Stretch",
                            "Traditional")) %>%
group_by(gymID,stretchType,yogaStretch) %>%
  summarize(nTrialCustomers = sum(nTrialCustomers),
            nSigned = sum(nSigned))
graph = dag_create() %>%
  dag_node("Cust Signed","k",
           rhs = binomial(n,p),
           data = poolTimeGymDF$nSigned) %>%
  dag_node("Probability of Signing","p",
           rhs = beta(2,2),
           child = "k") %>%
  dag_node("Trial Size","n",
           data = poolTimeGymDF$nTrialCustomers,
           child = "k") %>%
  dag_plate("Yoga Stretch","x",
            nodeLabels = c("p"),
            data = poolTimeGymDF$stretchType,
            addDataNode = TRUE) %>%
  dag_plate("Observation","i",
            nodeLabels = c("x","k","n")) %>%
```

```
  dag_plate("Gym","j",
            nodeLabels = "p",
            data = poolTimeGymDF$gymID,
            addDataNode = TRUE)
graph %>% dag_render()
## Not run:
# below requires Tensorflow installation
drawsDF = graph %>% dag_greta(mcmc=TRUE)
drawsDF %>% dagp_plot()

## End(Not run)
```

---

`dag_create` *Create a graph object focused on drawing a DAG.*

---

## Description

Generates a `causact_graph` graph object that is set-up for drawing DAG graphs.

## Usage

```
dag_create()
```

## Value

a list object of class 'causact_graph' consisting of 6 dataframes. Each data frame is responsible for storing information about nodes, edges, plates, and the relationships among them.

## Examples

```
# With `dag_create()` we can create an empty graph and
# add in nodes (`dag_node()`), add edges (`dag_edge`), and
# view the graph with `dag_render()`.
dag_create()
```

---

`dag_diagrammer` *The graph object should be of class* `causact_graph` *and created using* `dag_create()`.

---

## Description

The graph object should be of class `causact_graph` and created using `dag_create()`.

## Usage

```
dag_diagrammer(graph, wrapWidth = 24, shortLabel = FALSE)
```

### Arguments

| | |
|---|---|
| graph | a graph object of class causact_graph created using dag_create(). |
| wrapWidth | a required character label that describes the node. |
| shortLabel | a longer more descriptive character label for the node. |

### Value

a graph object of class dgr_graph. Useful for further customizing graph displays using the DiagrammeR package.

### Examples

```
library("DiagrammeR")
dag_create() %>%
dag_node("Get Card","y",
         rhs = bernoulli(theta),
         data = carModelDF$getCard) %>%
  dag_diagrammer() %>%
  render_graph(title = "DiagrammeR Version of causact_graph")
```

---

## dag_dim *Add dimension information to* causact_graph

---

### Description

Internal function that is used as part of rendering graph or running greta.

### Usage

```
dag_dim(graph)
```

### Arguments

| | |
|---|---|
| graph | a graph object of class causact_graph created using dag_create(). |

### Value

a graph object of class causact_graph with populated dimension information.

## dag_edge *Add an edge (or edges) between nodes in a graph object.*

### Description

With a graph object of class causact_graph created from dag_create, add an edge between nodes in the graph. Vector recycling is used for all arguments.

### Usage

```
dag_edge(graph, from, to, type = as.character(NA))
```

### Arguments

| | |
|---|---|
| graph | a graph object of class causact_graph. |
| from | a character vector representing the parent nodes label or description from which the edge is connected. |
| to | the child node label or description from which the edge is connected. |
| type | character string used to represent the DiagrammeR line type (e.g. "solid"). Use type = "extract" to encourage causact to only pass indexed elements of the parent node to each instance of the child node. Specify type = "solid" to override any automated extract behavior. |

### Value

a graph object of class dgr_graph with additional edges created by this function.

### Examples

```
# Create a graph with 2 connected nodes
dag_create() %>%
  dag_node("X") %>%
  dag_node("Y") %>%
  dag_edge(from = "X", to = "Y") %>%
  dag_render(shortLabel = TRUE)
```

## dag_greta *Generate a representive sample of the posterior distribution*

### Description

The input graph object should be of class causact_graph and created using dag_create(). The specification of a completely consistent joint distribution is left to the user. Helpful error messages are scheduled for future versions of the causact package.

## Usage

```
dag_greta(graph, mcmc = TRUE, meaningfulLabels = TRUE, ...)
```

## Arguments

| | |
|---|---|
| graph | a graph object of class causact_graph representing a complete and conistent specification of a joint distribution. |
| mcmc | a logical value indicating whether to sample from the posterior distribution. When mcmc=FALSE, the greta code is printed to the console, but not executed. The user can cut and paste the code to another script for running line-by-line. This option is most useful for debugging purposes. When mcmc=TRUE, the code is executed and outputs a dataframe of posterior draws. |
| meaningfulLabels | a logical value indicating whether to replace the indexed variable names in draws with abbreviated names representing the factor value corresponding to the index. This argument is treated as TRUE regardless of user input. The ability to retain numerical indexing will be in a subsequent release. |
| ... | additional arguments to be passed onto greta::mcmc(). |

## Value

If mcmc=TRUE, returns a dataframe of posterior distribution samples corresponding to the input 'causact_graph'. Each column is a parameter and each row a draw from the posterior sample output. If mcmc=FALSE, running dag_greta returns a character string of code that would help the user create three objects representing the posterior distribution:

1. draws: An mcmc.list object containing raw output from the HMCMC sampler used by greta.
2. drawsDF: A wide data frame with all latent variables as columns and all draws as rows. This data frame is useful for calculations based on the posterior
3. tidyDrawsDF: A long data frame with each draw represented on one line. This data frame is useful for plotting posterior distributions.

## Examples

```
library(greta)
graph = dag_create() %>%
  dag_node("Get Card","y",
           rhs = bernoulli(theta),
           data = carModelDF$getCard) %>%
  dag_node(descr = "Card Probability by Car",label = "theta",
           rhs = beta(2,2),
           child = "y") %>%
  dag_node("Car Model","x",
           data = carModelDF$carModel,
           child = "y") %>%
  dag_plate("Car Model","x",
            data = carModelDF$carModel,
            nodeLabels = "theta")
```

```
graph %>% dag_render()
gretaCode = graph %>% dag_greta(mcmc=FALSE)
## Not run:
## default functionality returns a data frame
# below requires Tensorflow installation
drawsDF = graph %>% dag_greta()
drawsDF %>% dagp_plot()

## End(Not run)
```

---

dag_merge *Merge two non-intersect* causact_graph *objects*

---

## Description

Generates a single causact_graph graph object that combines the multiple provided graphs.

## Usage

```
dag_merge(graph1, ...)
```

## Arguments

| | |
|---|---|
| graph1 | A causact_graph objects to be merged with |
| ... | As many causact_graph's as wish to be merged |

## Value

a merged graph object of class causact_graph. Useful for creating simple graphs and then merging them into a more complex structure.

## Examples

```
# With `dag_merge()` we
# reset the node ID's and all other item ID's,
# bind together the rows of all given graphs, and
# add in nodes and edges later
# with other functions
# to connect the graph.
#
# THE GRAPHS TO BE MERGED MUST BE DISJOINT
# THERE CAN BE NO IDENTICAL NODES OR PLATES
# IN EACH GRAPH TO BE MERGED, AT THIS TIME


g1 = dag_create() %>%
 dag_node("Demand for A","dA",
           rhs = normal(15,4)) %>%
  dag_node("Supply for A","sA",
```

```
          rhs = uniform(0,100)) %>%
  dag_node("Profit for A","pA",
          rhs = min(sA,dA)) %>%
  dag_edge(from = c("dA","sA"),to = c("pA"))


g2 <- dag_create() %>%
  dag_node("Demand for B","dB",
          rhs = normal(20,8)) %>%
  dag_node("Supply for B","sB",
          rhs = uniform(0,100)) %>%
  dag_node("Profit for B","pB",
          rhs = min(sB,dB)) %>%
  dag_edge(from = c("dB","sB"),to = c("pB"))

g1 %>% dag_merge(g2) %>%
  dag_node("Total Profit", "TP",
          rhs = sum(pA,pB)) %>%
  dag_edge(from=c("pA","pB"), to=c("TP")) %>%
  dag_render()
```

---

dag_node *Add a node to an existing* causact_graph *object*

---

## Description

The graph object should be of class causact_graph and created using dag_create().

## Usage

```
dag_node(
  graph,
  descr = as.character(NA),
  label = as.character(NA),
  rhs = NA,
  child = as.character(NA),
  data = NULL,
  obs = FALSE,
  keepAsDF = FALSE,
  extract = as.logical(NA),
  dec = FALSE,
  det = FALSE
)
```

## Arguments

| | |
|---|---|
| graph | a graph object of class causact_graph. An initial object gets created using dag_create(). |
| descr | a longer more descriptive character label for the node. |

| | |
|---|---|
| label | a shorter character label for referencing the node (e.g. "X","beta"). |
| rhs | either a greta distribution such as uniform,normal,lognormal,bernoulli, etc. or an R expression. Greta distribution arguments are optional. Valid values include normal(mu,sigma),greta::normal, normal, and normal(6,2). R computation/expression examples include alpha+beta*x or ilogit(alpha + gamma + beta). If a distribution is given, this is a random/stochastic node, if a formula is given it is a deterministic node once given the values of its parents. Quotes should not be used as all function/computations should consist of R objects, functions, and constants. |
| child | an optional character vector of existing node labels. Directed edges from the newly created node to the supplied nodes will be created. |
| data | a vector or data frame (with observations in rows and variables in columns). |
| obs | a logical value indicating whether the node is observed. Assumed to be TRUE when data argument is given. |
| keepAsDF | a logical value indicating whether the data argument should be split into one random varaible node per column or kept together as a random matrix for matrix computation. Defaults to creating one node per column of the data frame. |
| extract | a logical value. When TRUE, child nodes will try to extract an indexed value from this node. When FALSE, the entire random object (e.g. scalar, vector, matrix) is passed to children nodes. Only use this argument when overriding default behavior seen using dag_render(). |
| dec | a logical value indicating whether the node is a decision node. Used to show nodes as rectangles instead of ovals when using dag_render(). |
| det | a logical value indicating whether the node is a deterministic function of its parents Used to draw a double-line (i.e. peripheries = 2) around a shape when using dag_render(). Assumed to be TRUE when rhs is a formula. |

## Value

a graph object of class causact_graph with an additional node(s).

## Examples

```
library(greta)
# Create an empty graph and add 2 nodes by using
# the `dag_node()` function twice
graph2 = dag_create() %>%
  dag_node("Get Card","y",
         rhs = bernoulli(theta),
         data = carModelDF$getCard) %>%
  dag_node(descr = "Card Probability by Car",label = "theta",
           rhs = beta(2,2),
           child = "y")
graph2 %>% dag_render()


# The Eight Schools Example from Gelman et al.:
```

```
schools_dat <- data.frame(y = c(28,  8, -3,  7, -1,  1, 18, 12),
sigma = c(15, 10, 16, 11,  9, 11, 10, 18), schoolName = paste0("School",1:8))

graph = dag_create() %>%
  dag_node("Treatment Effect","y",
           rhs = normal(theta, sigma),
           data = schools_dat$y) %>%
  dag_node("Std Error of Effect Estimates","sigma",
           data = schools_dat$sigma,
           child = "y") %>%
  dag_node("Exp. Treatment Effect","theta",
           child = "y",
           rhs = avgEffect + schoolEffect) %>%
  dag_node("Pop Treatment Effect","avgEffect",
           child = "theta",
           rhs = normal(0,30)) %>%
  dag_node("School Level Effects","schoolEffect",
           rhs = normal(0,30),
           child = "theta") %>%
  dag_plate("Observation","i",nodeLabels = c("sigma","y","theta")) %>%
  dag_plate("School Name","school",
            nodeLabels = "schoolEffect",
            data = schools_dat$schoolName,
            addDataNode = TRUE)

graph %>% dag_render()
## Not run:
# below requires Tensorflow installation
graph %>% dag_greta(mcmc=TRUE)
tidyDrawsDF %>% dagp_plot()

## End(Not run)
```

---

`dag_plate` *Create a plate representation for repeated nodes.*

---

## Description

Given a graph object of class `causact_graph`, create collections of nodes that should be repeated i.e. represent multiple instances of a random variable, random vector, or random matrix. When nodes are on more than one plate, graph rendering will treat each unique combination of plates as separate plates.

## Usage

```
dag_plate(
  graph,
  descr,
  label,
  nodeLabels,
```

```
  data = as.character(NA),
  addDataNode = FALSE,
  rhs = NA
)
```

## Arguments

| | |
|---|---|
| graph | a graph object of class dgr_graph created using dag_create(). |
| descr | a longer more descriptive label for the cluster/plate. |
| label | a short character string to use as an index. |
| nodeLabels | a character vector of node labels or descriptions to include in the list of nodes. |
| data | a vector representing the categorical data whose unique values become the plate index. To use with addDataNode = TRUE, this vector should represent observations of a variable that can be coerced to a factor. |
| addDataNode | a logical value. When addDataNode = TRUE, the code attempts to add a node of observed data that is used as an index for extracting the correct parameter from parent nodes that are on the newly created plate. Verify the graphical model using dag_render() to ensure correct behavior. |
| rhs | Optional rhs expression for when addDataNode = TRUE. This can be either a greta distribution such as uniform,normal,lognormal,bernoulli, etc. or an R expression. Greta distribution arguments are optional. Valid values include normal(mu,sigma),greta::normal, normal, and normal(6,2). R computation/expression examples include alpha+beta*x or ilogit(alpha + gamma + beta). If a distribution is given, this is a random/stochastic node, if a formula is given it is a deterministic node once given the values of its parents. Quotes should not be used as all function/computations should consist of R objects, functions, and constants. |

## Value

an expansion of the input causact_graph object with an added plate representing the repetition of nodeLabels for each unique value of data.

## Examples

```
# single plate example
graph = dag_create() %>%
dag_node("Get Card","y",
         rhs = bernoulli(theta),
         data = carModelDF$getCard) %>%
  dag_node(descr = "Card Probability by Car",label = "theta",
           rhs = beta(2,2),
           child = "y") %>%
  dag_node("Car Model","x",
           data = carModelDF$carModel,
           child = "y") %>%
  dag_plate("Car Model","x",
            data = carModelDF$carModel,
```

```
            nodeLabels = "theta")
graph %>% dag_render()

# multiple plate example
library(dplyr)
poolTimeGymDF = gymDF %>%
mutate(stretchType = ifelse(yogaStretch == 1,
                            "Yoga Stretch",
                            "Traditional")) %>%
group_by(gymID,stretchType,yogaStretch) %>%
  summarize(nTrialCustomers = sum(nTrialCustomers),
            nSigned = sum(nSigned))
graph = dag_create() %>%
  dag_node("Cust Signed","k",
           rhs = binomial(n,p),
           data = poolTimeGymDF$nSigned) %>%
  dag_node("Probability of Signing","p",
           rhs = beta(2,2),
           child = "k") %>%
  dag_node("Trial Size","n",
           data = poolTimeGymDF$nTrialCustomers,
           child = "k") %>%
  dag_plate("Yoga Stretch","x",
            nodeLabels = c("p"),
            data = poolTimeGymDF$stretchType,
            addDataNode = TRUE) %>%
  dag_plate("Observation","i",
            nodeLabels = c("x","k","n")) %>%
  dag_plate("Gym","j",
            nodeLabels = "p",
            data = poolTimeGymDF$gymID,
            addDataNode = TRUE)
graph %>% dag_render()
```

---

dag_render *Render the graph as an htmlwidget*

---

## Description

Using a causact_graph object, render the graph in the RStudio Viewer.

## Usage

```
dag_render(
  graph,
  shortLabel = FALSE,
  wrapWidth = 24,
  width = NULL,
  height = NULL
)
```

### Arguments

| | |
|---|---|
| graph | a graph object of class dgr_graph. |
| shortLabel | a logical value. If set to TRUE, distribution and formula information is suppressed. Meant for communication with non-statistical stakeholders. |
| wrapWidth | a numeric value. Used to restrict width of nodes. Default is wrap text after 24 characters. |
| width | a numeric value. an optional parameter for specifying the width of the resulting graphic in pixels. |
| height | a numeric value. an optional parameter for specifying the height of the resulting graphic in pixels. |

### Value

Returns an object of class grViz and htmlwidget that is also rendered in the RStudio viewer for interactive buidling of graphical models.

### Examples

```
# Render a simple graph
dag_create() %>%
  dag_node("Demand","X") %>%
  dag_node("Price","Y", child = "X") %>%
  dag_render()

# Hide the mathematical details of a graph
dag_create() %>%
  dag_node("Demand","X") %>%
  dag_node("Price","Y", child = "X") %>%
  dag_render(shortLabel = TRUE)
```

---

## delivDF *117,790 line items associated with 23,339 shipments.*

---

### Description

A dataset containing the line items, mostly parts, asssociated with 23,339 shipments from a US-based warehouse.

### Usage

```
delivDF
```

## Format

A data frame (tibble) with 117,790 rows and 5 variables:

**shipID** unique ID for each shipment

**plannedShipDate** shipment date promised to customer

**actualShipDate** date the shipment was actually shipped

**partID** unique part identifier

**quantity** quantity of partID in shipment

## Source

Adam Fleischhacker

---

`generate_dot2` *Generate DOT code using a graph object*

---

## Description

Generates Graphviz DOT code as an R character object using DiagrammeR graph object.

## Usage

```
generate_dot2(graph)
```

## Arguments

| | |
|---|---|
| `graph` | A graph object of class `dgr_graph`. |

## Value

a character vector of length 1 containing Graphviz DOT code.

---

`grViz2` *R + viz.js*

---

## Description

Make diagrams in R using viz.js with infrastructure provided by htmlwidgets.

## Usage

```
grViz2(
  diagram = "",
  engine = "dot",
  allow_subst = TRUE,
  options = NULL,
  width = NULL,
  height = NULL
)
```

## Arguments

| | |
|---|---|
| diagram | spec for a diagram as either text, filename string, or file connection. |
| engine | string for the Graphviz layout engine; can be dot (default), neato, circo, or twopi. For more information see viz.js usage. |
| allow_subst | a boolean that enables/disables substitution functionality. |
| options | parameters supplied to the htmlwidgets framework. |
| width | an optional parameter for specifying the width of the resulting graphic in pixels. |
| height | an optional parameter for specifying the height of the resulting graphic in pixels. |

## Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

---

# gymDF *Dataframe of 44 observations of free crossfit classes data Each observation indicates how many students that participated in the free month of crossfit signed up for the monthly membership afterwards*

---

## Description

Dataframe of 44 observations of free crossfit classes data Each observation indicates how many students that participated in the free month of crossfit signed up for the monthly membership afterwards

## Usage

```
gymDF
```

## Format

A data frame with 44 rows and 5 variables:

**gymID** unique gym identifier

**nTrialCustomers** number of unique customers taking free trial classes

**nSigned** number of customers from trial that sign up for membership

**yogaStretch** whether trial classes included a yoga type stretch

**timePeriod** month number, since inception of company, for which trial period was offered

---

`houseDF` *Dataframe of 1,460 observations of home sales in Ames, Iowa. Known as The Ames Housing dataset, it was compiled by Dean De Cock for use in data science education. Each observation is a home sale. See* `houseDFDescr` *for more info.*

---

## Description

Dataframe of 1,460 observations of home sales in Ames, Iowa. Known as The Ames Housing dataset, it was compiled by Dean De Cock for use in data science education. Each observation is a home sale. See `houseDFDescr` for more info.

## Usage

```
houseDF
```

## Format

A data frame with 1,460 rows and 37 variables:

**SalePrice** the property's sale price in dollars. This is the target variable

**MSSubClass** The building class

**MSZoning** The general zoning classification

**LotFrontage** Linear feet of street connected to property

**LotArea** Lot size in square feet

**Street** Type of road access

**LotShape** General shape of property

**Utilities** Type of utilities available

**LotConfig** Lot configuration

**Neighborhood** Physical locations within Ames city limits

**BldgType** Type of dwelling

**HouseStyle** Style of dwelling

**OverallQual** Overall material and finish quality

**OverallCond** Overall condition rating

**YearBuilt** Original construction date

**YearRemodAdd** Remodel date

**ExterQual** Exterior material quality

**ExterCond** Present condition of the material on the exterior

**BsmtQual** Height of the basement

**BsmtCond** General condition of the basement

**BsmtExposure** Walkout or garden level basement walls

**BsmtUnfSF** Unfinished square feet of basement area

**TotalBsmtSF** Total square feet of basement area

**1stFlrSF** First Floor square feet

**2ndFlrSF** Second floor square feet

**LowQualFinSF** Low quality finished square feet (all floors)

**GrLivArea** Above grade (ground) living area square feet

**FullBath** Full bathrooms above grade

**HalfBath** Half baths above grade

**BedroomAbvGr** Number of bedrooms above basement level

**TotRmsAbvGrd** Total rooms above grade (does not include bathrooms)

**Functional** Home functionality rating

**GarageCars** Size of garage in car capacity

**MoSold** Month Sold

**YrSold** Year Sold

**SaleType** Type of sale

**SaleCondition** Condition of sale

## Source

https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data Accessed Jan 22, 2019. Kaggle dataset on "House Prices: Advanced Regression Techniques".

---

`houseDFDescr` *Dataframe of 523 descriptions of data values from "The Ames Housing dataset", compiled by Dean De Cock for use in data science education. Each observation is a possible value from a varaible in the* `houseDF` *dataset.*

---

## Description

Dataframe of 523 descriptions of data values from "The Ames Housing dataset", compiled by Dean De Cock for use in data science education. Each observation is a possible value from a varaible in the `houseDF` dataset.

### Usage

```
houseDFDescr
```

### Format

A data frame with 260 rows and 2 variables:

**varName** the name and description of a variable stored in the houseDF dataset

**varValueDescr** The value and accompanying interpretation for values in the houseDF dataset

### Source

https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data Accessed Jan 22, 2019. Kaggle dataset on "House Prices: Advanced Regression Techniques".

---

`meaningfulLabels` *Store meaningful parameter labels prior to running* `dag_greta()` *of* `greta::mcmc()`. *When* `greta` *creates posterior distributions for multi-dimensional parameters, it creates an often meaningless number system for the parameter (e.g. beta[1,1], beta[2,1], etc.). Since parameter dimensionality is often determined by a* `factor`*, this function creates labels from the factors unqiue values.* `replaceLabels()` *applies the text labels stored using this function to the* `greta` *output. The meaningful parameter names are stored in an environment,* `cacheEnv`.

---

### Description

Store meaningful parameter labels prior to running `dag_greta()` of `greta::mcmc()`. When `greta` creates posterior distributions for multi-dimensional parameters, it creates an often meaningless number system for the parameter (e.g. beta[1,1], beta[2,1], etc.). Since parameter dimensionality is often determined by a `factor`, this function creates labels from the factors unqiue values. `replaceLabels()` applies the text labels stored using this function to the `greta` output. The meaningful parameter names are stored in an environment, `cacheEnv`.

### Usage

```
meaningfulLabels(graph)
```

### Arguments

| | |
|---|---|
| `graph` | a `causact_graph` object. |

### Value

a data frame `meaningfulLabels` stored in an environment named `cacheEnv` that contains a lookup table between greta labels and meaningful labels.

`prodLineDF` *Product line and product category assignments for 12,026 partID's.*

## Description

A dataset containing partID attributes.

## Usage

```
prodLineDF
```

## Format

A data frame (tibble) with 117,790 rows and 5 variables:

**partID** unique part identifier

**productLine** a product line associated with the partID

**prodCategory** a product category associated with the partID

## Source

Adam Fleischhacker

`rbern` *Make dbern,pbern,qbern,rbern available for implementation of the Bernoulli distribution functions.*

## Description

Density, distribution function, quantile function and random generation for the benoulli distribution with parameter `prob`.

## Usage

```
rbern(n, prob)
```

## Arguments

| | |
|---|---|
| `n` | number of observations. If `length(n) > 1`, the length is taken to be the number required. |
| `prob` | probability of success of each trial |

## Value

A vector of 0's and 1's representing failure and success.

## Examples

```
#Return a random result of a Bernoulli trial given \code{prob}.
rbern(n =1, prob = 0.5)
```

---

`replaceLabels` *Replace parameter labels in a* `mcmc.list` *with more meaningful labels after they are created by running* `dag_greta()`. *When* `greta` *creates posterior distributions for multi-dimensional parameters, it creates an often meaningless number system for the parameter (e.g. beta[1,1], beta[2,1], etc.). Since parameter dimensionality is often determined by a* `factor`*, this functionality restores the text labels associated with the underlying factor whose coefficients are being estimated (e.g. beta_varValue1, beta_varValue2). The meaningful parameter names are stored in an environment,* `cacheEnv`*, created by a call to* `dag_greta()`.

---

## Description

Replace parameter labels in a `mcmc.list` with more meaningful labels after they are created by running `dag_greta()`. When `greta` creates posterior distributions for multi-dimensional parameters, it creates an often meaningless number system for the parameter (e.g. beta[1,1], beta[2,1], etc.). Since parameter dimensionality is often determined by a `factor`, this functionality restores the text labels associated with the underlying factor whose coefficients are being estimated (e.g. beta_varValue1, beta_varValue2). The meaningful parameter names are stored in an environment, `cacheEnv`, created by a call to `dag_greta()`.

## Usage

```
replaceLabels(draws)
```

## Arguments

| | |
|---|---|
| `draws` | an `mcmc.list` object created by `dag_greta()`. |

## Value

an `mcmc.list` with more meanignful names that get created during a `dag_greta` function call.

## replace_in_spec2 *Razor-like template for diagram specification*

### Description

Use Razor-like syntax to define a template for use in a `grViz` diagram.

### Usage

```
replace_in_spec2(spec)
```

### Arguments

| | |
|---|---|
| `spec` | string spec to be parsed and evaluated. |

## schoolsDF *This example, often referred to as 8-schools, was popularized by its inclusion in Bayesian Data Analysis (Gelman, Carlin, & Rubin 1997).*

### Description

This example, often referred to as 8-schools, was popularized by its inclusion in Bayesian Data Analysis (Gelman, Carlin, & Rubin 1997).

### Usage

```
schoolsDF
```

### Format

A data frame with 8 rows and 3 variables:

**y** estimated treatment effect at a particular school

**sigma** standard error of the treamtment effect estimate

**schoolName** an identifier for the school represented by this row

setDirectedGraphTheme *Set DiagrammeR defaults for graphical models*

## Description

setDirectedGraph returns a graph with good defaults.

## Usage

```
setDirectedGraphTheme(dgrGraph)
```

## Arguments

| | |
|---|---|
| dgrGraph | A DiagrammeR graph |

## Value

An updated version of dgrGraph with good defaults for graphical models.

return a dgrGraph object with the color and shape defaults used by the causact package.

## Examples

```
library(DiagrammeR)
create_graph() %>% add_node() %>% render_graph()  # default DiagrammeR aesthetics
create_graph() %>% add_node() %>% setDirectedGraphTheme() %>% render_graph() ## causact aesthetics
```

ticketsDF *Dataframe of 55,167 observations of the number of tickets written by NYC precincts each day Data modified from https://github.com/stan-dev/stancon_talks/tree/master/2018/Contributed-Talks/01_auerbach which originally sourced data from https://opendata.cityofnewyork.us/*

## Description

Dataframe of 55,167 observations of the number of tickets written by NYC precincts each day Data modified from https://github.com/stan-dev/stancon_talks/tree/master/2018/Contributed-Talks/01_auerbach which originally sourced data from https://opendata.cityofnewyork.us/

## Usage

```
ticketsDF
```

### Format

A data frame with 55167 rows and 4 variables:

**precinct** unique precinct identifier representing precinct of issuing officer

**date** the date on which ticket violations occurred

**month_year** the month_year extracted from date column

**daily_tickets** Number of tickets issued out of precinct on this day

---

`totalBeachgoersRepSample` *A representative sample from a random variable that represents the annual number of beach goers to Ocean City, MD beaches on hot days. Think of this representative sample as coming from either a prior or posterior distribution. An example using this sample is can be found in The Business Analyst's Guide To Business Analytics at http://causact.com.*

---

### Description

A representative sample from a random variable that represents the annual number of beach goers to Ocean City, MD beaches on hot days. Think of this representative sample as coming from either a prior or posterior distribution. An example using this sample is can be found in The Business Analyst's Guide To Business Analytics at http://causact.com.

### Usage

```
totalBeachgoersRepSample
```

### Format

A 4,000 element vector.

**totalBeachgoersRepSample** a draw from a representative sample of total beachgoers to Ocean City, MD.

---

`%>%` *The magrittr pipe*

---

### Description

causact uses the pipe function, `%>%` to turn function composition into a series of imperative statements.

### Value

Pipe a value forward into a function- or call expression and return the function on the ‘rhs‘ with the ‘lhs‘ used as the first argument.

# Index