Package 'theorytools'

June 23, 2025

Type Package

Title FAIR Theory Construction

Version 0.1.1

Description An integrated suite of tools for creating, maintaining, and reusing FAIR (Findable, Accessible, Interoperable, Reusable) theories. Designed to support transparent and collaborative theory development, the package enables users to formalize theories, track changes with version control, assess pre-empirical coherence, and derive testable hypotheses. Aligning with open science principles and workflows, 'theorytools' facilitates the systematic improvement of theoretical frameworks and enhances their discoverability and usability.

License GPL (>= 3)

Encoding UTF-8

URL https://cjvanlissa.github.io/theorytools/

RoxygenNote 7.3.2

Imports worcs (>= 0.1.16), gert, gh, cli, jsonlite, curl, knitr, dagitty, yaml, tidySEM (>= 0.2.8), methods, Deriv

Suggests rmarkdown, testthat (>= 3.0.0), pkgdown, webexercises, withr, usethis, fs, ggplot2, igraph, bookdown

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 3.5)

LazyData true

Config/Needs/website rmarkdown

NeedsCompilation no

Author Caspar J. Van Lissa [aut, cre] (ORCID: <https://orcid.org/0000-0002-0808-5024>)

Maintainer Caspar J. Van Lissa <c.j.vanlissa@tilburguniversity.edu>

Repository CRAN

Date/Publication 2025-06-23 12:40:02 UTC

Contents

add_readme_fair_theory . . . 2
add_significance . . . 3
add_theory_file . . . 3
add_to_pkgdown . . . 4
add_zenodo_json . . . 4
create_fair_theory . . . 5
derive_formula . . . 7
download_theory . . . 8
filter_conditional_independencies . . . 9
lsac . . . 9
mottistefanidi2012 . . . 10
quizz . . . 11
select_controls . . . 13
simulate_data . . . 14
use_webex_vignette . . . 15
webex_vignette . . . 16

Index **17**

`add_readme_fair_theory` *Add Readme File*

Description

Writes a README file to a specific path.

Usage

```
add_readme_fair_theory(path, title, ...)
```

Arguments

`path`	Character, indicating the directory in which to create the FAIR theory.
`title`	Character, indicating the theory title. Default: `NULL`
`...`	Additional arguments passed to other functions.

Value

Invisibly returns a logical value, indicating whether the function was successful or not.

Examples

```
add_readme_fair_theory(path = tempdir(), title = "My Theory")
```

add_significance *Add Significance Asterisks*

Description

Given a `data.frame` with a column containing p-values or two columns containing the lower- and upper bounds of a confidence interval, adds a column of significance asterisks.

Usage

```
add_significance(x, p_column = NULL, ci_lb = NULL, ci_up = NULL, alpha = 0.05)
```

Arguments

`x`	A `data.frame`
`p_column`	Atomic character, referring to the name of the column of p-values. If this is provided, the confidence interval is ignored. Default: `NULL`
`ci_lb`	Atomic character, referring to the name of the column of the lower bound of a confidence interval. Default: `NULL`
`ci_up`	Atomic character, referring to the name of the column of the upper bound of a confidence interval. Default: `NULL`
`alpha`	Significance level, default: `.05`

Value

A `data.frame`

Examples

```
tmp <- add_significance(head(iris))
```

add_theory_file *Add Theory File*

Description

Writes a theory file to a specific path.

Usage

```
add_theory_file(path, theory_file = "theory.txt")
```

Arguments

path	Character, indicating the directory in which to create the FAIR theory.
theory_file	Character, referring to existing theory file(s) to be copied, or a new theory file to be created. Default NULL does nothing.

Value

Invisibly returns a logical value, indicating whether the function was successful or not.

Examples

```
add_theory_file(path = tempdir(), theory_file = "theory.txt")
```

---

add_to_pkgdown *Add webexercises helper files to pkgdown*

Description

Adds the necessary helper files to an existing pkgdown project.

Usage

```
add_to_pkgdown(pkgdown_dir = ".")
```

Arguments

pkgdown_dir	The base directory for your pkgdown project

Value

No return value, called for side effects.

---

add_zenodo_json *Add 'Zenodo' JSON File*

Description

Writes a '.zenodo.json' file to the specified path.

Writes a README file to a specific path.

Usage

```
add_zenodo_json(path, title, upload_type, keywords)

add_zenodo_json_theory(path, title, keywords)
```

Arguments

path	Character, indicating the directory in which to create the FAIR theory.
title	Character, indicating the theory title. Default: NULL
upload_type	Character, indicating the upload type.
keywords	Character vector of keywords.

Value

Invisibly returns a logical value, indicating whether the function was successful or not.

See Also

read_json

Examples

```
add_zenodo_json(path = tempdir(), title = "Theory Title",
                upload_type = "software", keywords = "R")
add_zenodo_json_theory(path = tempdir(), title = "My Theory",
                       keywords = "secondkeyword")
add_zenodo_json_theory(path = tempdir(), title = "My Theory",
                       keywords = c("secondkeyword", "thirdkeyword"))
```

---

create_fair_theory *Create FAIR Theory Repository*

Description

Partly automates the process of creating a FAIR theory repository, see Details.

Usage

```
create_fair_theory(
  path,
  title = NULL,
  theory_file = NULL,
  remote_repo = NULL,
  add_license = "cc0",
  ...
)
```

Arguments

path	Character, indicating the directory in which to create the FAIR theory.
title	Character, indicating the theory title. Default: NULL
theory_file	Character, referring to existing theory file(s) to be copied, or a new theory file to be created. Default NULL does nothing.
remote_repo	Name of a 'GitHub' repository that exists or should be created on the current authenticated user's account, see gh_whoami, Default: NULL
add_license	PARAM_DESCRIPTION, Default: 'cc0'
...	Additional arguments passed to other functions.

Details

The following steps are executed sequentially:

1. Create a project folder at path
2. Initialize a local 'Git' repository at path
3. If remote_repo refers to a user's existing 'GitHub' repository, add it as remote to the local 'Git' repository. Otherwise, create a new 'GitHub' repository by that name and add it as remote.
4. Add theory file. If theory_file refers to an existing file, copy it to path. If theory_file refers to a new file, create it in path.
5. Add the license named by add_license
6. Add a README.md file
7. Add 'Zenodo' metadata so that it recognizes the repository as a FAIR theory
8. If it is possible to push to the remote repository, use git_update to push the repository to 'GitHub'

Value

Invisibly returns a logical value, indicating whether the function was successful or not.

See Also

git_repo add_license_file, git_update

Examples

```
# Create a theory with no remote repository (for safe testing)
theory_dir <- file.path(tempdir(), "theory")
create_fair_theory(path = theory_dir,
                   title = "This is My Theory",
                   theory_file = "theory.txt",
                   remote_repo = NULL,
                   add_license = "cc0")

# Create a theory with a remote repository
```

```
## Not run:
theory_dir <- file.path(tempdir(), "theory_github")
out <- create_fair_theory(path = theory_dir,
                          title = "This is My GitHub Theory",
                          theory_file = "theory.txt",
                          remote_repo = "delete_test",
                          add_license = "ccby")

## End(Not run)
```

`derive_formula` *Derive Formulae From Augmented DAG*

Description

Uses the `form` attribute of edges in an augmented DAG to construct formulae for the relationship between `exposure` and `outcome`.

Usage

```
derive_formula(x, exposure, outcome, data = NULL, ...)
```

Arguments

`x`	An object of class `dagitty`.
`exposure`	Atomic character, indicating the exposure node in `x`.
`outcome`	Atomic character, indicating the outcome node in `x`.
`data`	Optional, a `data.frame` to be used as environment for the formulae. Default: `NULL`
`...`	Additional arguments, passed to `adjustmentSets`

Details

The `form` attribute of an augmented DAG of class `dagitty` should contain information about the functional form for the relationships specified in the DAG. For this function, the `form` attribute must be an additive function as also accepted by `formula`. The `form` attribute may contain a leading intercept and constant slopes, which will be parsed out. If the `form` attribute does not meet these requirements, the resulting `formula` may be invalid. For example:

- `form=".5+x1"` would return `~x1`.
- `form="2*x1*x2"` would return `~x1+x2+x1:x2`.
- `form="-.2-.2*I(x3^2)"` would return `~I(x3^2)`.

Value

A `list` of objects with class `formula`.

See Also

hasArg get_edges adjustmentSets

Examples

```
x <- dagitty::dagitty('dag {
C
O
X
Y
O <- X [form="I(X^2)"]
C -> X
Y -> O [form="Y*X"]
}')
f1 <- derive_formula(x, outcome = "O", exposure = "X")
f2 <- derive_formula(x, outcome = "O", exposure = "Y")
```

---

download_theory *Download FAIR Theory*

Description

Downloads a FAIR theory archive from a 'Git' remote repository or 'Zenodo'.

Usage

```
download_theory(id, path = ".")
```

Arguments

id	URL of the 'Git' repository or DOI of the 'Zenodo' archive.
path	Character, indicating the directory in which to create the FAIR theory.

Examples

```
download_theory(id = "https://github.com/cjvanlissa/tripartite_model.git",
path = file.path(tempdir(), "tripartite_git"))
download_theory(id = "10.5281/zenodo.14921521",
path = file.path(tempdir(), "tripartite_zenodo"))
```

filter_conditional_independencies *Filter Conditional Independencies*

Description

Removes all conditional independencies, obtained using impliedConditionalIndependencies, based on the variables available in data.

Usage

```
filter_conditional_independencies(x, data)
```

Arguments

x	An object of class dagitty.cis.
data	A data.frame.

Value

An object of class dagitty.cis, or NULL if no conditional independencies remain.

See Also

impliedConditionalIndependencies

Examples

```
dag <- dagitty::dagitty('dag {
x1 -> y
x2 -> y}')
df <- data.frame(x1 = rnorm(10), y = rnorm(10))
cis <- dagitty::impliedConditionalIndependencies(dag)
cis <- filter_conditional_independencies(cis, df)
is.null(cis)
```

lsac *Synthetic Data: Longitudinal Study of Australian Children*

Description

This synthetic dataset, based on "Growing Up in Australia - the Longitudinal Study of Australian Children" (LSAC). This longitudinal study covers a representative sample of about 10.000 children and their families, and aims to examine children's development from early childhood through to adolescence and adulthood. All variables pertain to the mother; note that fathers also play an important and sometimes unique role in children's emotional development.

Usage

```
data(lsac)
```

Format

A data frame with 8214 rows and 6 variables.

Details

Except for "coping", all variables were created by taking the row means of several items, omitting missing values. The corresponding item names from the LSAC codebook are given below.

warmth	numeric	fpa03m1-fpa03m6	Parental warmth scale
relationship_quality	numeric	fre04m1-fre04m7	Hendrick relationship quality scale
temperament_negreact	numeric	fse13a1-fse13a4	Temperament scale for reactivity
emotion_regulation	numeric	fse03c3a-fse03c3e	SDQ Emotional problems scale
social_functioning	numeric	fgd04b2a-fgd04b2e	Peds QL social functioning
coping	numeric	fhs26m2	Level of coping

References

Australian Institute of Family Studies. (2020, March 23). Growing Up in Australia.

---

mottistefanidi2012 *Adaptation of Adolescent Immigrants in Greece*

Description

This simulated dataset, based on work by Motti-Stefanidi and colleagues (2012), assesses the adaptation and well-being of adolescent immigrants in Greek schools.

Usage

```
data(mottistefanidi2012)
```

Format

A data frame with 1057 rows and 17 variables.

Details

GRK1	ordered	Greek language ability at age 13
IMM1	ordered	Immigrant status at age 13 (0 = no, 1 = yes)
ADV1	ordered	Adversity at age 13
INV1	ordered	Parental School Involvement at age 13
SEF1	ordered	Self-efficacy at age 13
GPA1	ordered	Academic achievement at age 13

GPA2	`ordered`	Academic achievement at age 14
GPA3	`ordered`	Academic achievement at age 15
CON1	`ordered`	Conduct at age 13
CON2	`ordered`	Conduct at age 14
CON3	`ordered`	Conduct at age 15
EMO1	`ordered`	Emotional symptoms at age 13
EMO2	`ordered`	Emotional symptoms at age 14
EMO3	`ordered`	Emotional symptoms at age 15
POP1	`numeric`	Peer popularity at age 13
POP2	`numeric`	Peer popularity at age 14
POP3	`numeric`	Peer popularity at age 15

References

Motti-Stefanidi, F., Asendorpf, J. B., & Masten, A. S. (2012). The adaptation and well-being of adolescent immigrants in Greek schools: A multilevel, longitudinal study of risks and resources. Development and Psychopathology, 24(2), 451–473. doi:10.1017/S0954579412000090

`quizz` *Create a Quiz*

Description

Convenience function for creating a basic quiz in HTML format from the arguments captured by `...`, where the type of each question is determined automatically from the class of the arguments.

Usage

```
quizz(
  ...,
  render_if = knitr::is_html_output(),
  title = "Quiz",
  show_box = TRUE,
  show_check = TRUE
)
```

Arguments

`...`	Each argument should be a named vector, see Details.
`render_if`	Logical, whether or not to render the output. By default, this argument uses `knitr::is_html_output()`.
`title`	Atomic character, default: 'Quiz'
`show_box`	Logical, whether or not to draw a box around the quiz. Default: `TRUE`
`show_check`	Logical, whether or not to show a button to check answers. Default: `TRUE`

Details

The function renders questions captured by the arguments in `...`. The name of each argument is the text of the question. The value of each argument determined the question type and its correct answer. The following types of questions are supported:

`"torf()"` The argument should be a single value of type `logical`, e.g.: `"The answer to this question is true." = TRUE`

`"mcq()"` The argument should be a vector of type `character`. The first element is taken as the correct answer; the order of answers is randomized. E.g.: `"This multiple choice question has three answers." = c("Correct", "Incorrect", "Not sure")`

`"fitb()"` The argument should be of type `numeric`. If the vector is atomic, the first element is taken as the correct answer, e.g.: `"Provide an exact floating point answer of 0.81" = 0.81`. If the vector has two elements, the second element is taken as the tolerance `tol`, e.g.: `"Here, 0.8 will be correct." = c(0.81, 0.01)`. If the vector is of type `integer`, the tolerance is set to zero, e.g.: `"The answer is 4." = 4L`

Alternatively, `...` may contain a single atomic character referring to a text file that contains the questions, see examples.

Value

`NULL`, this function is called for its side effect of printing HTML code using `cat()`.

See Also

`is_latex_output` `mcq`, `torf`, `fitb`

Examples

```
# Quiz from arguments:
invisible(capture.output(theorytools:::quizz(
"The answer to this question is true." = TRUE,
"This multiple choice question has three answers." =
 c(answer = "Correct", "Incorrect", "Not sure"),
"Provide an exact floating point answer of 0.81" = 0.81,
render_if = TRUE
)))
# From a file:
quizz_file <- tempfile()
writeLines(
c("The answer is true. = TRUE",
"The answer is correct = c(answer = \"Correct\", \"Incorrect\", \"Not sure\")",
"The answer is exactly .81 = 0.81",
"But here, .8 is also fine = c(0.81, .01)",
"Write the word 'true' = c('true', 'TRUE')",
"Here, answer exactly 4. = 4L")
, quizz_file)
invisible(capture.output(theorytools:::quizz(quizz_file, render_if = TRUE)))
```

`select_controls` *Select Covariate Adjustment Sets from Data*

Description

Wraps adjustmentSets to construct a dataset with covariates that (asymptotically) allow unbiased estimation of causal effects from observational data.

Usage

```
select_controls(
  x,
  data,
  exposure = NULL,
  outcome = NULL,
  which_set = c("first", "sample", "all"),
  ...
)
```

Arguments

`x`	An input graph of class `dagitty`.
`data`	A `data.frame` or object coercible by `as.data.frame()`.
`exposure`	Atomic character, name of the exposure variable.
`outcome`	Atomic character, name of the outcome variable.
`which_set`	Atomic character, indicating which set of covariates to select in case there are multiple. Valid choices are in `c("first", "sample", "all")`, see Value.
`...`	Other arguments passed to adjustmentSets

Value

If `which_set = "all"`, returns a list of `data.frames` to allow for sensitivity analyses. Otherwise, returns a `data.frame`.

See Also

`adjustmentSets`

Examples

```
dag <- dagitty::dagitty('dag {x -> y}')
df <- data.frame(x = rnorm(10), y = rnorm(10))
df1 <- select_controls(dag, df, exposure = "x", outcome = "y")
class(df1) == "data.frame"
df2 <- select_controls(dag, df, exposure = "x", outcome = "y", which_set = "sample")
class(df2) == "data.frame"
lst1 <- select_controls(dag, df, exposure = "x", outcome = "y", which_set = "all")
class(lst1) == "list"
```

`simulate_data` *Simulate Data from DAG*

Description

Simulates data from an (augmented) DAG, respecting the optional metadata fields `form`, for functional form of relationships, and `distribution`, for the distributions of exogenous nodes and residuals.

Usage

```
simulate_data(
  x,
  beta_default = round(runif(1, min = -0.6, max = 0.6), 2),
  n = 500,
  run = TRUE,
  duplicated = "unique"
)
```

Arguments

`x`	An object of class `dagitty`.
`beta_default`	Function used to specify missing edge coefficients. Default: `runif(1, min = -0.6, max = 0.6)`
`n`	Atomic integer defining the sample size, default: `500`
`run`	Logical, indicating whether or not to run the simulation. Default: `TRUE`.
`duplicated`	Atomic character, indicating how to resolve duplicate terms from multiple edges pointing to the same node. Default: `"unique"`. See Details.

Details

Data is simulated sequentially, first from exogenous nodes and then from their descendants. If `x` is an augmented DAG with metadata indicating the functional `form` of relationships and `distribution` of exogenous nodes and residuals, this information is used. If this information is absent, nodes and residuals are assumed to be normally distributed, and edges are assumed to be linear, with coefficients samples based on `beta_default`.

The argument `duplicated` controls how multiplicative terms are merged across edges pointing to the same outcome node. The default `duplicated = "unique"` removes terms that are duplicated across edges (i.e., if two edges point to node `"O"`, and both edges specify `.5*E`, the resulting function will say `.5*E`. However, if one edge specifies `.2*E` and the other specifies `.3*E`, they are not duplicated and will be added. Alternatively, `duplicated = "add"` just sums terms across all edges pointing into the same outcome node.

Value

If run is TRUE, this function returns a data.frame with an additional attribute called attr(, which = "script") that contains the script for simulating data. If run is FALSE, this function returns the script as character vector.

See Also

get_nodes, get_edges exogenousVariables

Examples

```
x <- dagitty::dagitty('
dag {
  X [distribution="rbinom(size = 2, prob = .5)"]
  Z [distribution="rexp()"]
  Y [distribution="rnorm()"]

  X -> Y [form="0.5+X"]
  Z -> Y [form="2*Z"]
  A -> X
}
')
txt <- simulate_data(x, n = 5, run = FALSE)
df <- simulate_data(x, n = 5, run = TRUE)
df_from_txt <- eval(parse(text = txt))
```

---

use_webex_vignette *Create a webexercises vignette*

Description

Wraps use_vignette to add a vignette or article to vignettes/ with support for webexercises.

Usage

```
use_webex_vignette(name, title = NULL, type = c("vignette", "article"))
```

Arguments

name	Atomic character, vignette name. See use_vignette.
title	Atomic character, vignette title. See use_vignette.
type	Atomic character, one of c("vignette", "article"), defaults to "vignette".

Value

Returns NULL invisibly, called for its side effects.

Examples

```
## Not run:
use_webex_vignette("vignette_with_quiz.Rmd", "Quiz people with webexercises")

## End(Not run)
```

`webex_vignette` *Create Vignette with webexercises Support*

Description

This function wraps `rmarkdown::html_document` to configure compilation to embed the default webexercises CSS and JavaScript files in the resulting HTML.

Usage

```
webex_vignette(...)
```

Arguments

`...`	Additional function arguments to pass to `html_document`.

Details

Call this function as the `output_format` argument for the `render` function when compiling HTML documents from RMarkdown source.

Value

R Markdown output format to pass to 'render'.

See Also

`render`, `html_document`

Index

∗ **datasets**
    lsac, 9
    mottistefanidi2012, 10

add_license_file, *6*
add_readme_fair_theory, 2
add_significance, 3
add_theory_file, 3
add_to_pkgdown, 4
add_zenodo_json, 4
add_zenodo_json_theory
        *(*add_zenodo_json*)*, 4
adjustmentSets, *7, 8*, *13*

create_fair_theory, 5

derive_formula, 7
download_theory, 8

exogenousVariables, *15*

filter_conditional_independencies, 9
fitb, *12*
formula, *7*

get_edges, *8*, *15*
get_nodes, *15*
gh_whoami, *6*
git_repo, *6*
git_update, *6*

hasArg, *8*
html_document, *16*

impliedConditionalIndependencies, *9*
is_latex_output, *12*

lsac, 9

mcq, *12*
mottistefanidi2012, 10

quizz, 11

read_json, *5*
render, *16*

select_controls, 13
simulate_data, 14

torf, *12*

use_vignette, *15*
use_webex_vignette, 15

webex_vignette, 16