Package 'causaloptim'

March 25, 2022

Encoding UTF-8

Type Package

Title An Interface to Specify Causal Graphs and Compute Bounds on Causal Effects

Version 0.9.2

Date 2022-03-25

Maintainer Michael C Sachs <sachsmc@gmail.com>

Description When causal quantities are not identifiable from the observed data, it still may be possible to bound these quantities using the observed data. We outline a class of problems for which the derivation of tight bounds is always a linear programming problem and can therefore, at least theoretically, be solved using a symbolic linear optimizer. We extend and generalize the approach of Balke and Pearl (1994) <doi:10.1016/B978-1-55860-332-5.50011-0> and we provide a user friendly graphical interface for setting up such problems via directed acyclic graphs (DAG), which only allow for problems within this class to be depicted. The user can then define linear constraints to further refine their assumptions to meet their specific problem, and then specify a causal query using a text interface. The program converts this user defined DAG, query, and constraints, and returns tight bounds. The bounds can be converted to R functions to evaluate them for specific datasets, and to latex code for publication. The methods and proofs of tightness and validity of the bounds are described in a preprint by Sachs, Gabriel, and Sjölander (2021) <https://sachsmc.github.io/causaloptim/articles/CausalBoundsMethods.pdf>.

License MIT + file LICENSE

Imports methods, Rcpp (>= 1.0.1), shiny, rcdd

Depends R (>= 3.5.0), igraph

LinkingTo Rcpp

RoxygenNote 7.1.1

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/sachsmc/causaloptim

BugReports https://github.com/sachsmc/causaloptim/issues

Config/testthat/edition 3

NeedsCompilation yes

Author Michael C Sachs [aut, cre],
Erin E Gabriel [aut],
Arvid Sjölander [aut],
Gustav Jonzon [ctb] ((improved vertex enumeration)),
Alexander A Balke [ctb] ((C++ code)),
Colorado Reed [ctb] ((graph-creator.js))

Repository CRAN

Date/Publication 2022-03-25 14:00:07 UTC

R topics documented:

`causaloptim-package` *An Interface to Specify Causal Graphs and Compute Bounds on Causal Effects*

Description

Specify causal graphs using a visual interactive interface and then analyze them and compute symbolic bounds for the causal effects in terms of the observable parameters.

Details

Run the shiny app by results <- specify_graph(). See detailed instructions in the vignette browseVignettes("causaloptim").

Author(s)

Michael C Sachs, Arvid Sjölander, Gustav Jonzon, Alexander Balke, Colorado Reed, and Erin Gabriel Maintainer: Michael C Sachs <sachsmc at gmail.com>

References

A. Balke and J. Pearl, "Counterfactual Probabilities: Computational Methods,Bounds, and Applications" UCLA Cognitive Systems Laboratory, Technical Report (R-213-B). In R. Lopez de Mantaras and D. Poole (Eds.), Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI-94), Morgan Kaufmann, San Mateo, CA, 46-54, July 29-31, 1994. https://ftp.cs.ucla.edu/pub/stat_ser/R213-B.pdf .

See Also

`browseVignettes('causaloptim')`

`analyze_graph` *Analyze the causal graph to determine constraints and objective*

Description

The graph must contain certain edge and vertex attributes which are documented in the Details below. The shiny app run by specify_graph will return a graph in this format.

Usage

```
analyze_graph(graph, constraints, effectt)
```

Arguments

`graph`	An aaa-igraph-package object that represents a directed acyclic graph with certain attributes. See Details.
`constraints`	A vector of character strings that represent the constraints on counterfactual quantities
`effectt`	A character string that represents the causal effect of interest

Details

The graph object must contain the following named vertex attributes:

name The name of each vertex must be a valid R object name starting with a letter and no special characters. Good candidate names are for example, Z1, Z2, W2, X3, etc.

leftside An indicator of whether the vertex is on the left side of the graph, 1 if yes, 0 if no.

latent An indicator of whether the variable is latent (unobserved). There should always be a variable Ul on the left side that is latent and a parent of all variables on the left side, and another latent variable Ur on the right side that is a parent of all variables on the right side.

nvals The number of possible values that the variable can take on, the default and minimum is 2 for 2 categories (0,1).

In addition, there must be the following edge attributes:

rlconnect An indicator of whether the edge goes from the right side to the left side. Should be 0 for all edges.

edge.monotone An indicator of whether the effect of the edge is monotone, meaning that if V1 -> V2 and the edge is monotone, then a > b implies V2(V1 = a) >= V2(V1 = b). Only available for binary variables (nvals = 2).

The effectt parameter describes your causal effect of interest. The effectt parameter must be of the form

`p{V11(X=a)=a; V12(X=a)=b;...}` op1 `p{V21(X=b)=a; V22(X=c)=b;...}` op2 ...

where Vij are names of variables in the graph, a, b are numeric values from 0:(nvals - 1), and op are either - or +. You can specify a single probability statement (i.e., no operator). Note that the probability statements begin with little p, and use curly braces, and items inside the probability statements are separated by ;. The variables may be potential outcomes which are denoted by parentheses. Variables may also be nested inside potential outcomes. Pure observations such as `p{Y = 1}` are not allowed if the left side contains any variables. If the left side contains any variables, then they mush be ancestors of the intervention set variables (or the intervention variables themselves). All of the following are valid effect statements:

`p{Y(X = 1) = 1}` -`p{Y(X = 0) = 1}`

`p{X(Z = 1) = 1; X(Z = 0) = 0}`

`p{Y(M(X = 0),X = 1) = 1}` -`p{Y(M(X = 0),X = 0) = 1}`

The constraints are specified in terms of potential outcomes to constrain by writing the potential outcomes, values of their parents, and operators that determine the constraint (equalities or inequalities). For example, `X(Z = 1) >= X(Z = 0)`

Value

A an object of class "linearcausalproblem", which is a list with the following components. This list can be passed to optimize_effect which interfaces with Balke's code. Print and plot methods are also available.

variables Character vector of variable names of potential outcomes, these start with 'q' to match Balke's notation

parameters Character vector of parameter names of observed probabilities, these start with 'p' to match Balke's notation

constraints Character vector of parsed constraints

objective Character string defining the objective to be optimized in terms of the variables

p.vals Matrix of all possible values of the observed data vector, corresponding to the list of parameters.

q.vals Matrix of all possible values of the response function form of the potential outcomes, corresponding to the list of variables.

parsed.query A nested list containing information on the parsed causal query.

objective.nonreduced The objective in terms of the original variables, before algebraic variable reduction. The nonreduced variables can be obtained by concatenating the columns of q.vals.

response.functions List of response functions.

graph The graph as passed to the function.

R A matrix with coefficients relating the p.vals to the q.vals p = R * q

c0 A vector of coefficients relating the q.vals to the objective function theta = c0 * q

iqR A matrix with coefficients to represent the inequality constraints

Examples

```
### confounded exposure and outcome
b <- igraph::graph_from_literal(X -+ Y, Ur -+ X, Ur -+ Y)
V(b)$leftside <- c(0,0,0)
V(b)$latent <- c(0,0,1)
V(b)$nvals <- c(2,2,2)
E(b)$rlconnect <- E(b)$edge.monotone <- c(0, 0, 0)
analyze_graph(b, constraints = NULL, effectt = "p{Y(X = 1) = 1} - p{Y(X = 0) = 1}")
```

---

btm_var *Recursive function to get the last name in a list*

Description

Recursive function to get the last name in a list

Usage

```
btm_var(x, name = NULL)
```

Arguments

x	a list
name	name of the top element of the list

Value

The name of the deepest nested list element

---

const.to.sets *Translate lists of constraints to lists of vectors*

Description

Translate lists of constraints to lists of vectors

Usage

```
const.to.sets(constr, objterms)
```

Arguments

constr	List of constraint terms as character strings
objterms	Vector of terms in the objective function

---

constant_term *Compute the scalar product of two numeric vectors of the same length*

Description

A helper function for evaluate_objective.

Usage

```
constant_term(numbers1, numbers2)
```

Arguments

numbers1, numbers2	Two numeric vectors of the same length.

Value

A string consisting of the value of the scalar product of numbers1 and numbers2.

create_effect_vector *Translate target effect to vector of response variables*

Description

Translate target effect to vector of response variables

Usage

```
create_effect_vector(effect, graph, obsvars, respvars, q.list, variables)
```

Arguments

effect	Effect list, as returned by parse_effect
graph	The graph
obsvars	Vector of observed variable vertices from the graph
respvars	Response function, as returned by create_response_function
q.list	List with q matrices, as returned by create_q_matrix
variables	Vector of qs names

Value

A list with the target effect in terms of qs

create_q_matrix *Translate response functions into matrix of counterfactuals*

Description

Translate response functions into matrix of counterfactuals

Usage

```
create_q_matrix(respvars, right.vars, cond.vars, constraints)
```

Arguments

respvars	A list of functions as returned by create_response_function
right.vars	Vertices of graph on the right side
cond.vars	Vertices of graph on the left side
constraints	A vector of character strings that represent the constraints

Value

A list of 3 data frames of counterfactuals and their associated labels

create_response_function *Translate regular DAG to response functions*

Description

Translate regular DAG to response functions

Usage

```
create_response_function(graph, right.vars, cond.vars)
```

Arguments

graph	An aaa-igraph-package object that represents a directed acyclic graph must contain edge attributes named "leftside" and "lrconnect" that takes values 0 and 1. Only one edge may have a value 1 for lrconnect. The shiny app returns a graph in this format.
right.vars	Vertices of graph on the right side
cond.vars	Vertices of graph on the left side

Value

A list of functions representing the response functions

create_R_matrix *Create constraint matrix*

Description

Matrix and text representation of constraints on observed probabilities

Usage

```
create_R_matrix(
  graph,
  obsvars,
  respvars,
  p.vals,
  parameters,
  q.list,
  variables
)
```

Arguments

graph	The graph
obsvars	Vector of observed variable vertices from the graph
respvars	Response function, as returned by create_response_function
p.vals	Observed probability matrix
parameters	Vector of ps names
q.list	List with q matrices, as returned by create_q_matrix
variables	Vector of qs names

Value

A list with the R matrix and the string representation

---

evaluate_objective *Compute the scalar product of a vector of numbers and a vector of both numbers and strings*

Description

A helper function for opt_effect.

Usage

```
evaluate_objective(c1_num, p, y)
```

Arguments

c1_num	A numeric column matrix.
p	A character vector.
y	A numeric vector whose length is the sum of the lengths of c1_num and p.

Value

A string consisting of an affine expression in p corresponding to the scalar product of c(c1_num,p) with y.

expand_cond *Expand potential outcome conditions*

Description

Expand potential outcome conditions

Usage

```
expand_cond(cond, obsnames)
```

Arguments

cond	Text string of the condition
obsnames	Vector of names of observed variables

find_cycles *Find cycles in a graph*

Description

Find cycles in a graph

Usage

```
find_cycles(g)
```

Arguments

g	an igraph object

Value

A list of vectors of integers, indicating the vertex sequences for the cycles found in the graph

interpret_bounds *Convert bounds string to a function*

Description

Convert bounds string to a function

Usage

```
interpret_bounds(bounds, parameters)
```

Arguments

bounds	The bounds element as returned by optimize_effect
parameters	Character vector defining parameters, as returned by analyze_graph

Value

A function that takes arguments for the parameters, i.e., the observed probabilities and returns a vector of length 2: the lower bound and the upper bound.

Examples

```
b <- graph_from_literal(X -+ Y, Ur -+ X, Ur -+ Y)
V(b)$leftside <- c(0,0,0)
V(b)$latent <- c(0,0,1)
V(b)$nvals <- c(2,2,2)
E(b)$rlconnect <- E(b)$edge.monotone <- c(0, 0, 0)
obj <- analyze_graph(b, constraints = NULL, effectt = "p{Y(X = 1) = 1} - p{Y(X = 0) = 1}")
bounds <- optimize_effect(obj)
bounds_func <- interpret_bounds(bounds$bounds, obj$parameters)
bounds_func(.1, .1, .4, .3)
# vectorized
do.call(bounds_func, lapply(1:4, function(i) runif(5)))
```

latex_bounds *Latex bounds equations*

Description

Latex bounds equations

Usage

```
latex_bounds(bounds, parameters, prob.sym = "P", brackets = c("(", ")"))
```

Arguments

bounds	Vector of bounds as returned by optimize_effect
parameters	The parameters object as returned by analyze_graph
prob.sym	Symbol to use for probability statements in latex, usually "P" or "pr"
brackets	Length 2 vector with opening and closing bracket, usually c("(", ")"), or c("\", "\")

Value

A character string with latex code for the bounds

Examples

```
b <- graph_from_literal(X -+ Y, Ur -+ X, Ur -+ Y)
V(b)$leftside <- c(0,0,0)
V(b)$latent <- c(0,0,1)
V(b)$nvals <- c(2,2,2)
E(b)$rlconnect <- E(b)$edge.monotone <- c(0, 0, 0)
obj <- analyze_graph(b, constraints = NULL, effectt = "p{Y(X = 1) = 1} - p{Y(X = 0) = 1}")
bounds <- optimize_effect(obj)
latex_bounds(bounds$bounds, obj$parameters)
latex_bounds(bounds$bounds, obj$parameters, "Pr")
```

---

linear_expression *Compute the scalar product of a vector of numbers and a vector of strings*

Description

A helper function for evaluate_objective.

Usage

```
linear_expression(numbers, strings)
```

Arguments

numbers	A numeric vector.
strings	A character vector of the same length as numbers.

Value

A string consisting of the corresponding linear combination, including the sign of its first term.

linear_term *Compute the product of a single numeric scalar and a single string*

Description

A helper function for `linear_expression`.

Usage

```
linear_term(number, string)
```

Arguments

number	A numeric vector of length 1.
string	A character vector of length 1.

Value

A string consisting of the concatenation of `number` and `string`, including its sign.

list_to_path *Recursive function to translate an effect list to a path sequence*

Description

Recursive function to translate an effect list to a path sequence

Usage

```
list_to_path(x, name = NULL)
```

Arguments

x	A list of vars as returned by parse_effect
name	The name of the outcome variable

Value

a list of characters describing the path sequence

numberOfValues *Get the number of values of a given variable in the graph*

Description

Get the number of values of a given variable in the graph

Usage

```
numberOfValues(graph, varname)
```

Arguments

graph	An igraph object.
varname	A string. The name of a vertex in 'graph'.

Value

An integer greater than 1. The number of values of 'varname'.

optimize_effect *Run the Balke optimizer*

Description

Given a object with the linear programming problem set up, compute the bounds using the c++ code developed by Alex Balke. Bounds are returned as text but can be converted to R functions using interpret_bounds, or latex code using latex_bounds.

Usage

```
optimize_effect(obj)
```

Arguments

obj	Object as returned by analyze_graph

Value

An object of class "balkebound" that contains the bounds and logs as character strings

Examples

```
b <- graph_from_literal(X -+ Y, Ur -+ X, Ur -+ Y)
V(b)$leftside <- c(0,0,0)
V(b)$latent <- c(0,0,1)
V(b)$nvals <- c(2,2,2)
E(b)$rlconnect <- E(b)$edge.monotone <- c(0, 0, 0)
obj <- analyze_graph(b, constraints = NULL, effectt = "p{Y(X = 1) = 1} - p{Y(X = 0) = 1}")
optimize_effect(obj)
```

`optimize_effect_2` *Run the optimizer*

Description

Given an object with the linear programming problem set up, compute the bounds using rcdd. Bounds are returned as text but can be converted to R functions using interpret_bounds, or latex code using latex_bounds.

Usage

```
optimize_effect_2(obj)
```

Arguments

`obj`	Object as returned by analyze_graph

Value

An object of class "balkebound" that contains the bounds and logs as character strings

Examples

```
b <- graph_from_literal(X -+ Y, Ur -+ X, Ur -+ Y)
V(b)$leftside <- c(0,0,0)
V(b)$latent <- c(0,0,1)
V(b)$nvals <- c(2,2,2)
E(b)$rlconnect <- E(b)$edge.monotone <- c(0, 0, 0)
obj <- analyze_graph(b, constraints = NULL, effectt = "p{Y(X = 1) = 1} - p{Y(X = 0) = 1}")
optimize_effect_2(obj)
```

opt_effect *Compute a bound on the average causal effect*

Description

This helper function does the heavy lifting for optimize_effect_2. For a given casual query, it computes either a lower or an upper bound on the corresponding causal effect.

Usage

```
opt_effect(opt, obj)
```

Arguments

opt	A string. Either "min" or "max" for a lower or an upper bound, respectively.
obj	An object as returned by the function analyze_graph. Contains the casual query to be estimated.

Value

An object of class optbound; a list with the following named components:

- expr is the *main* output; an expression of the bound as a print-friendly string,
- type is either "lower" or "upper" according to the type of the bound,
- dual_vertices is a numeric matrix whose rows are the vertices of the convex polytope of the dual LP,
- dual_vrep is a V-representation of the dual convex polytope, including some extra data.

parse_constraints *Parse text that defines a the constraints*

Description

Parse text that defines a the constraints

Usage

```
parse_constraints(constraints, obsnames)
```

Arguments

constraints	A list of character strings
obsnames	Vector of names of the observed variables in the graph

Value

A data frame with columns indicating the variables being constrained, what the values of their parents are for the constraints, and the operator defining the constraint (equality or inequalities).

---

parse_effect *Parse text that defines a causal effect*

Description

Parse text that defines a causal effect

Usage

```
parse_effect(text)
```

Arguments

text	Character string

Value

A nested list that contains the following components:

vars For each element of the causal query, this indicates potential outcomes as names of the list elements, the variables that they depend on, and the values that any variables are being fixed to.

oper The vector of operators (addition or subtraction) that combine the terms of the causal query.

values The values that the potential outcomes are set to in the query.

pcheck List of logicals for each element of the query that are TRUE if the element is a potential outcome and FALSE if it is an observational quantity.

---

pastestar *Paste with asterisk sep*

Description

Paste with asterisk sep

Usage

```
pastestar(...)
```

Arguments

...	Things to paste together

plot.linearcausalproblem *Plot the graph from the causal problem*

Description

Plot the graph from the causal problem

Usage

```
## S3 method for class 'linearcausalproblem'
plot(x, ...)
```

Arguments

x	object of class "linearcausaloptim"
...	Not used

Value

Nothing

plot_graphres *Plot the analyzed graph object*

Description

Special plotting method for igraphs of this type

Usage

```
plot_graphres(graphres)
```

Arguments

graphres	an igraph object

Value

None

print.linearcausalproblem *Print the causal problem*

Description

Print the causal problem

Usage

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

Arguments

x	object of class "linearcausaloptim"
...	Not used

Value

x, invisibly

print_nvals *Print the number of values of each variable/vertex of the analyzed graph object*

Description

Print the number of values of each variable/vertex of the analyzed graph object

Usage

```
print_nvals(graphres)
```

Arguments

graphres	an igraph object

Value

None

reduce.sets *Algebraically reduce sets*

Description

Identifies and reduces redundant variables

Usage

```
reduce.sets(sets)
```

Arguments

sets	List of constraints as sets of variables

shortentxt *Shorten strings to 80 characters wide*

Description

Shorten strings to 80 characters wide

Usage

```
shortentxt(x)
```

Arguments

x	String

Value

A string with line breaks to keep the width less than 80 characters

simulate_bounds *Simulate bounds*

Description

Run a simple simulation based on the bounds. For each simulation, sample the set of counterfactual probabilities from a uniform distribution, translate into a multinomial distribution, and then compute the objective and the bounds in terms of the observable variables.

Usage

```
simulate_bounds(obj, bounds, nsim = 1000)
```

Arguments

obj	Object as returned by analyze_graph
bounds	Object as returned by optimize_effect
nsim	Number of simulation replicates

Value

A data frame with columns: objective, bound.lower, bound.upper

Examples

```
b <- graph_from_literal(X -+ Y, Ur -+ X, Ur -+ Y)
V(b)$leftside <- c(0,0,0)
V(b)$latent <- c(0,0,1)
V(b)$nvals <- c(2,2,2)
E(b)$rlconnect <- E(b)$edge.monotone <- c(0, 0, 0)
obj <- analyze_graph(b, constraints = NULL, effectt = "p{Y(X = 1) = 1} - p{Y(X = 0) = 1}")
bounds <- optimize_effect(obj)
simulate_bounds(obj, bounds, nsim = 5)
```

specify_graph *Shiny interface to specify network structure and compute bounds*

Description

This launches the Shiny interface in the system's default web browser. The results of the computation will be displayed in the browser, but they can also be returned to the R session by assigning the result of the function call to an object. See below for information on what is returned.

Usage

```
specify_graph()
```

Value

If the button "Exit and return graph object" is clicked, then only the graph is returned as an aaa-igraph-package object.

If the bounds are computed and the button "Exit and return objects to R" is clicked, then a list is returned with the following elements:

graphres The graph as drawn and interpreted, an aaa-igraph-package object.

obj The objective and all necessary supporting information. This object is documented in analyze_graph. This can be passed directly to optimize_effect_2.

bounds.obs Object of class 'balkebound' as returned by optimize_effect_2.

constraints Character vector of the specified constraints. NULL if no constraints.

effect Text describing the causal effect of interest.

boundsFunction Function that takes parameters (observed probabilities) as arguments, and returns a vector of length 2 for the lower and upper bounds.

---

symb.subtract *Symbolic subtraction*

Description

Like setdiff but doesn't remove duplicates x1 - x2

Usage

```
symb.subtract(x1, x2)
```

Arguments

x1	First term (subtract from)
x2	Second term (subtract)

---

update_effect *Update the effect in a linearcausalproblem object*

Description

If you want to use the same graph and response function, but change the effect of interest, this can save some computation time.

Usage

```
update_effect(obj, effectt)
```

Arguments

`obj`	An object as returned by analyze_graph
`effectt`	A character string that represents the causal effect of interest

Value

A object of class linearcausalproblem, see analyze_graph for details

Index