Package ‘causaloptim’

May 7, 2020

Encoding UTF-8
Type Package
Title An Interface to Specify Causal Graphs and Compute Bounds on Causal Effects
Version 0.7.1
Date 2020-05-07
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 (2020) <https://sachsmc.github.io/causaloptim/articles/CausalBoundsMethods.pdf>.

License MIT + file LICENSE
Imports methods, Rcpp (>= 1.0.1), shiny
Depends igraph
LinkingTo Rcpp
RoxygenNote 7.1.0
Suggests knitr, rmarkdown
VignetteBuilder knitr
URL https://github.com/sachsmc/causaloptim

BugReports https://github.com/sachsmc/causaloptim/issues
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, Alexander Balke, Colorado Reed, and Erin Gabriel
Maintainer: Michael C Sachs <sachsmc at gmail.com>

References


See Also

browseVignettes('causaloptim')

analyze_graph

Analyze the causal graph to determine constraints and objective

Description

The 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.

Usage

analyze_graph(graph, constraints, effectt)

Arguments

graph An aaaigraph-package object that represents a directed acyclic graph
constraints A vector of character strings that represent the constraints
effectt A character string that represents the causal effect of interest

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.

t## Examples

### confounded exposure and outcome

```r
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)
E(b)$rlconnect <- E(b)$edge.monotone <- c(0, 0, 0)
analyze_graph(b, constraints = NULL, effectt = "p(Y(1) = 1) - p(Y(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

```r
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

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)
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")

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"

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)
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")
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

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

```r
b <- graph_from_literal(X -+ Y, U_r -+ X, U_r -+ Y)
V(b)$leftside <- c(0,0,0)
V(b)$latent <- c(0,0,1)
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)
```

### parse_constraints

Parse text that defines the constraints

**Description**

Parse text that defines the constraints

**Usage**

```r
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**

```r
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 (0 or 1).

**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**

```r
## 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**

```r
plot_graphres(graphres)
```

**Arguments**

- `graphres` an igraph object

**Value**

None

---

**print.linearcausalproblem**

*Print the causal problem*

**Description**

Print the causal problem

**Usage**

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

**Arguments**

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

**Value**

`x`, invisibly
reduce.sets  \hspace{1cm} Algebraically reduce sets

Description
Identifies and reduces redundant variables

Usage
reduce.sets(sets)

Arguments
sets  List of constraints as sets of variables

shortentxt  \hspace{1cm} Shorten strings to 80 characters wide

Description
Shorten strings to 80 characters wide

Usage
shortentxt(x)

Arguments
\hspace{1cm} \hspace{1cm} x  String

simulate_bounds  \hspace{1cm} 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)
### 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**

```r
specify_graph()
```

**Value**

If the button “Exit and return graph object” is clicked, then only the graph is returned as an `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 `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`.
- **bounds.obs** Object of class 'balkebound' as returned by `optimize_effect`.
- **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.

**Arguments**

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

**Examples**

```r
b <- graph_from_literal(X — Y, Ur — X, Ur — Y)
V(b)$leftside <- c(0,0,0)
V(b)$latent <- c(0,0,1)
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)
```
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)     |
Index

aaa-igraph-package, 3, 13
analyze_graph, 3, 6–8, 13

btm_var, 4

causaloptim (causaloptim-package), 2
causaloptim-package, 2
const.to.sets, 5

expand_cond, 5

find_cycles, 6

interpret_bounds, 6, 8

latex_bounds, 7, 8
list_to_path, 8

optimize_effect, 3, 6, 7, 8, 13

parse_constraints, 9
parse_effect, 9
pastestar, 10
plot.linearcausalproblem, 10
plot.graphres, 11
print.linearcausalproblem, 11

reduce.sets, 12

shortentxt, 12
simulate_bounds, 12
specify_graph, 13
symb.subtract, 14