Package ‘R6causal’

August 6, 2021

Type Package
Title R6 Class for Structural Causal Models
Version 0.6.0
Maintainer Juha Karvanen <juha.karvanen@iki.fi>
Description The implemented R6 class ‘SCM’ aims to simplify working with structural causal models. The missing data mechanism can be defined as a part of the structural model. The class contains methods for 1) defining a structural causal model via functions, text or conditional probability tables, 2) printing basic information on the model, 3) plotting the graph for the model using packages ‘igraph’ or ‘qgraph’, 4) simulating data from the model, 5) applying an intervention, 6) checking the identifiability of a query using the R packages ‘causaleffect’ and ‘dosearch’, 7) defining the missing data mechanism, 8) simulating incomplete data from the model according to the specified missing data mechanism and 9) checking the identifiability in a missing data problem using the R package ‘dosearch’. In addition, there are functions for running experiments and doing counterfactual inference using simulation.
License AGPL-3
Encoding UTF-8
RoxygenNote 7.1.1
Imports causaleffect, data.table, dosearch, igraph, R6, stats
Suggests rmarkdown, knitr, qgraph, sqldf
VignetteBuilder knitr
NeedsCompilation no
Author Juha Karvanen [aut, cre] (<https://orcid.org/0000-0001-5530-769X>)
Repository CRAN
Date/Publication 2021-08-06 08:20:13 UTC

R topics documented:

backdoor .................................................. 2
backdoor_md .............................................. 2
counterfactual ............................................. 3
backdoor

Variable \( z \) fulfills the back-door criterion for \( P(y|do(x)) \).

**Usage**

\[
\text{backdoor}
\]

**Format**

An object of class SCM (inherits from R6) of length 21.

**Examples**

\[
\text{backdoor}
\]
\[
\text{backdoor}\$plot()
\]

---

**backdoor_md**

Variable \( z \) fulfills the back-door criterion for \( P(y|do(x)) \). Variable \( z \) is missing completely at random. The missingness of variables \( x \) and \( y \) depend on \( z \).

**Usage**

\[
\text{backdoor_md}
\]

**Format**

An object of class SCM (inherits from R6) of length 21.

**Examples**

\[
\text{backdoor_md}
\]
\[
\text{backdoor_md}\$plot()
\]
counterfactual  

Counterfactual inference via simulation

Description

Counterfactual inference via simulation

Usage

counterfactual(
  scm,  
situation, 
target, 
ifunction, 
n,  
slotsize = 10000, 
maxslots = 100, 
situationSQL = FALSE
)

Arguments

scm  
An SCM object

situation  
A list or a character string. The list has the following elements:
  • do : NULL or a list containing named elements 'target' and 'ifunction' that specify the intervention carried out in the situation
  • condition : either a string that gives an SQL query ("select x,y,z from DATA where") or a data.table consisting of the valid rows
The character string specifies an SQL query ("select x,y,z from DATA where")

target  
A vector of variable names that specify the target variable(s) of the counterfactual intervention.

ifunction  
A list of functions for the counterfactual intervention.

n  
Size of the data to be simulated

slotsize  
A scalar, the number of rows to be simulated for a slot

maxslots  
A scalar, the maximum number of slots

situationSQL  
Logical, is the situation defined as an SQL query, defaults FALSE

Value

A data table representing the situation after the counterfactual intervention
Examples

cfdata <- counterfactual(backdoor,
situation = list(do = list(target = "x", ifunction = 0),
condition = data.table::data.table( x = 0, y = 0)),
target = "x",
ifunction = 1,
n = 100000)
mean(cfdata$y)

frontdoor

SCM "frontdoor" used in the examples.

Description

Variable z fulfills the front-door criterion for \(P(y|do(x))\)

Usage

frontdoor

Format

An object of class SCM (inherits from R6) of length 21.

Examples

frontdoor
frontdoor$plot()

generate_condprob

Define structural function by a conditional probability table

Description

Define structural function by a conditional probability table

Usage

generate_condprob(ycondx, x, Umerge_expr = NULL)
Arguments

- **ycondx**: A data table or a data frame with the following structure
  - 1st column: variable to be generated, "Y"
  - middle columns: the parents of the 1st column variable, "X"
  - last column: the probability the case specified be the other columns, "P(Y|do(X))"
- **x**: A data table or a data frame that contains data on the variables in the middle columns of **ycondx**, "X" and one or more columns giving data on U-variables.
- **Umerge_expr**: A character string specifying how the U-variables will be combined when the value "Y" is generated, e.g. "u" or "(u1+u2)/2". The result of the expression should be a random number in the interval [0,1].

Value

A data table containing the generated variable, "Y"

Examples

```r
ycondx <- data.table::data.table(y =rep(c(0,1), each = 3), x=rep(1:3, 2),
prob = c(0.2,0.6,0.1,0.8,0.4,0.9))
x <- data.table::data.table(x = sample(1:3, 20, replace = TRUE),
uy = stats::runif(20), uy2 = stats::runif(20))
generate_condprob(ycondx, x, Umerge_expr = "(uy+uy2)/2")
```

Description

Package R6causal implements an R6 class for structural causal models (SCM) with latent variables and missing data mechanism. The class contains methods for 1) defining a structural causal model via functions, text or conditional probability tables, 2) printing basic information on the model, 3) plotting the graph for the model using packages `igraph` or `qgraph`, 4) simulating data from the model, 5) applying an intervention, 6) checking the identifiability of a query using the R packages `causaleffect` and `dosearch`, 7) defining the missing data mechanism, 8) simulating incomplete data from the model according to the specified missing data mechanism and 9) checking the identifiability in a missing data problem using the R package `dosearch`. In addition, there are functions for running experiments and doing counterfactual inference using simulation.

References

run_experiment  
Conduct a sequence of interventions and collect the simulated data.

Description
Conduct a sequence of interventions and collect the simulated data.

Usage
run_experiment(SCM, intervene, response, n)

Arguments
- **SCM**: An SCM object
- **intervene**: A list where the names of the elements are the variables to be intervened and the values of the elements are vectors specifying the values set in the intervention
- **response**: A vector of the names of the response variables
- **n**: Size of the data to be simulated for each intervention

Value
A list containing the values of the response variables for all intervention combinations

Examples
backdoor_experiment <- run_experiment(backdoor,  
  intervene = list(x = c(0,1)),  
  response = "y",  
  n = 10000)

colMeans(backdoor_experiment$response_list$y)

SCM  
R6 Class for structural causal models

Description
R6 Class for structural causal models
R6 Class for structural causal models

Details
An R6 class for structural causal models (SCM) with latent variables and missing data mechanism. There are methods for defining, printing, plotting, intervening and simulating SCMs.
Active bindings

vflist List of the structural functions of observed variables.
vfperm List of the names of observed variables.
simdata Data table containing data simulated from the SCM.
simdata_md Data table containing data simulated from the SCM where missing values are indicated by NA.
igraph The graph of the SCM in the igraph form (without the missing data mechanism).
igraph_bidirected The graph of the SCM in the igraph form where latent variables are presented by bidirected arcs.
igraph_md The graph of the SCM in the igraph form including the missing data mechanism.
toporder A vector giving the topological order of variables.
toporderv A vector giving the topological order of observed variables.
graphtext A character string that gives the edges of the graph of the SCM (without the missing data mechanism).
graphtext_md A character string that gives the edges of the graph of the SCM including the missing data mechanism.
name The name of the SCM.

Methods

Public methods:

- SCM$new()
- SCM$print()
- SCM$plot()
- SCM$intervene()
- SCMsimulate()
- SCM$causal.effect()
- SCM$dosearch()
- SCM$clone()

Method new(): Create a new SCM object.

Usage:
SCM$new(
  name,
uflist,
vflist,
rflist = NULL,
rprefix = "R_",
starsuffix = "_md"
)

Arguments:

name Name.
uflist A named list containing the functions for latent variables.
vflist A named list containing the functions for observed variables.
rflist A named list containing the functions for missingness indicators.
rprefix The prefix of the missingness indicators.
starsuffix The suffix for variables with missing data.

Returns: A new ‘SCM’ object.

Examples:
backdoor <- SCM$new("backdoor",
  uflist = list(
    uz = function(n) {return(stats::runif(n))},
    ux = function(n) {return(stats::runif(n))},
    uy = function(n) {return(stats::runif(n))}
  ),
  vflist = list(
    z = function(uz) {
      return(as.numeric(uz < 0.4))
    },
    x = function(ux, z) {
      return(as.numeric(ux < 0.2 + 0.5*z))
    },
    y = function(uy, z, x) {
      return(as.numeric(uy < 0.1 + 0.4*z + 0.4*x))
    }
  )
)

Method print(): Print a summary of the SCM object.
  Usage:
  SCM$print()
  Examples:
  backdoor

Method plot(): Plot the DAG of the SCM object.
  Usage:
  SCM$plot(subset = "uvr", method = "igraph", ...)
  Arguments:
  subset Variable groups to be plotted: "uvr","vr","uv", or "v".
  method Plotting method: "qgraph" or "igraph".
  ... other parameters passed to the plotting method
  Examples:
  backdoor$plot()
  backdoor$plot("v")

Method intervene(): Apply an intervention to the SCM object.
  Usage:
  SCM$intervene(target, ifunction)
  Arguments:
target Name(s) of the variables in vflist to be intervened.
ifunction Either numeric value(s) or new structural function(s) for the target variables.

Examples:
# A simple intervention
target <- backdoor$clone()  # making a copy
target$intervene("x",1)  # applying the intervention
target$plot()  # to see that arrows incoming to x are cut

# An intervention that redefines a structural equation
target_yz <- backdoor$clone()  # making a copy
target_yz$intervene("y",
    function(uy, z) {return(as.numeric(uy < 0.1 + 0.8*z ))})  # making y a function of z only
target_yz$plot()  # to see that arrow x -> y is cut

Method simulate(): Simulate data from the SCM object. Creates or updates simdata. If
no_missing_data = FALSE, creates or updates also simdata_md

Usage:
SCM$simulate(n = 1, no_missing_data = FALSE, fixedvars = NULL)

Arguments:

n Number of observations to be generated.
no_missing_data Logical, should the generation of missing data skipped? (defaults FALSE).
fixedvars List of variables that remain unchanged.

Examples:
target$simulate(10)
target$simdata

Method causal.effect(): Is a causal effect identifiable from observational data? Calls the
implementation of ID algorithm from package causaleffect. See the documentation of causal.effect
for the details.

Usage:
SCM$causal.effect(y, x, ...)

Arguments:
y A vector of character strings specifying target variable(s).
x A vector of character strings specifying intervention variable(s).
... Other parameters passed to causal.effect.

Returns: An expression for the joint distribution of the set of variables (y) given the inter-
vention on the set of variables (x) conditional on (z) if the effect is identifiable. Otherwise an
error is thrown describing the graphical structure that witnesses non-identifiability. @examples
target$causal.effect(y = "y", x = "x")

Method dosearch(): Is a causal effect or other query identifiable from given data sources?
Calls dosearch from the package dosearch. See the documentation of dosearch for the details.

Usage:
SCM$dosearch(
    data,
    query,
    transportability = NULL,
    selection_bias = NULL,
    missing_data = NULL,
    control = NULL
)

Arguments:
- **data**: Character string specifying the data sources.
- **query**: Character string specifying the query of interest.
- **transportability**: Other parameters passed to dosearch().
- **selection_bias**: Other parameters passed to dosearch().
- **missing_data**: Other parameters passed to dosearch().
- **control**: List of control parameters passed to dosearch().

Returns: An object of class dosearch.

Examples:
backdoor$dosearch(data = "p(x,y,z)", query = "p(y|do(x))")

Method **clone()**: The objects of this class are cloneable with this method.

Usage:
SCM$clone(deep = FALSE)

Arguments:
- **deep**: Whether to make a deep clone.

Examples

```r
## Method `SCM$new`

backdoor <- SCM$new("backdoor",
    uflist = list(
        uz = function(n) {return(stats::runif(n))},
        ux = function(n) {return(stats::runif(n))},
        uy = function(n) {return(stats::runif(n))}
    ),
    vflist = list(
        z = function(uz) {
            return(as.numeric(uz < 0.4)),
        x = function(ux, z) {
            return(as.numeric(ux < 0.2 + 0.5*z)),
        y = function(uy, z, x) {
            return(as.numeric(uy < 0.1 + 0.4*z + 0.4*x))
        }
    )
)
```

```
## Method `SCM$print`

```

```
## Method `SCM$plot`

```

```
## Method `SCM$intervene`

```

# A simple intervention
backdoor_x1 <- backdoor$clone() # making a copy
backdoor_x1$intervene("x", 1) # applying the intervention
backdoor_x1$plot() # to see that arrows incoming to x are cut

# An intervention that redefines a structural equation
backdoor_yz <- backdoor$clone() # making a copy
backdoor_yz$intervene("y",
    function(uy, z) {return(as.numeric(uy < 0.1 + 0.8*z ))}) # making y a function of z only
backdoor_yz$plot() # to see that arrow x -> y is cut

```

```
## Method `SCMsimulate`

```

```
## Method `SCM$dosearch`

```

```
backdoor$dosearch(data = "p(x,y,z)", query = "p(y|do(x))")
```

---

**trapdoor**

*SCM "trapdoor" used in the examples.*

---

**Description**

Variable z is a trapdoor variable for P(y|do(x))
Usage

trapdoor

Format

An object of class SCM (inherits from R6) of length 21.

References


Examples

trapdoor
trapdoor$plot()
Index

* datasets
  backdoor, 2
  backdoor_md, 2
  frontdoor, 4
  trapdoor, 11

backdoor, 2
backdoor_md, 2

causal.effect, 9
counterfactual, 3
dosearch, 9
frontdoor, 4
generate_condprob, 4
R6causal, 5
run_experiment, 6

SCM, 6
trapdoor, 11