

Package ‘noncomplyR’

July 23, 2025

Type Package

Title Bayesian Analysis of Randomized Experiments with Non-Compliance

Version 1.0

Date 2017-07-25

Author Scott Coggeshall [aut, cre]

Maintainer Scott Coggeshall <sscogges@uw.edu>

Description Functions for Bayesian analysis of data from randomized experiments with non-compliance. The functions are based on the models described in Imbens and Rubin (1997) <doi:10.1214/aos/1034276631>. Currently only two types of outcome models are supported: binary outcomes and normally distributed outcomes. Models can be fit with and without the exclusion restriction and/or the strong access monotonicity assumption. Models are fit using the data augmentation algorithm as described in Tanner and Wong (1987) <doi:10.2307/2289457>.

License GPL-2

LazyData TRUE

RoxygenNote 5.0.1

Imports MCMCpack (>= 1.4.0), stats

Suggests knitr

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2017-08-24 08:30:38 UTC

Contents

cace	2
compliance_chain	3
noncomplyR	4
summarize_chain	4
vitaminA	5

Index	6
--------------	----------

cace	<i>Compute the Posterior Distribution of the CACE</i>
------	---

Description

cace takes a sample from the posterior distribution of the model parameters and computes the corresponding posterior distribution of the Complier Average Causal Effect.

Usage

```
cace(chain, outcome_model, strong_access)
```

Arguments

chain	a matrix containing the draws from the posterior distribution of the model parameters. The matrix should either be the result of a call to <code>compliance_chain</code> or have the same structure as one.
outcome_model	a character string indicating which outcome model was used in fitting the model, either "binary" for a dichotomous outcome or "normal" for the Normal model.
strong_access	a logical indicating whether the strong access monotonicity assumption was made when fitting the model

Value

a vector containing the draws from the posterior distribution of the CACE

Examples

```
# CACE based on a subset of the vitaminA dataset
set.seed(4923)
chain <- compliance_chain(vitaminA[sample(1:nrow(vitaminA), 1000),], outcome_model = "binary",
  exclusion_restriction = TRUE, strong_access = TRUE, n_iter = 10, n_burn = 1)

cace(chain, outcome_model = "binary", strong_access = TRUE)

# matrix representing the samples from the posterior distribution of the model parameters
posterior_mat <- matrix(rnorm(10*8, mean = 10), nrow = 10, ncol = 8)
cace(posterior_mat, "normal", strong_access = TRUE)
```

compliance_chain	<i>Data Augmentation for Non-compliance analysis</i>
------------------	--

Description

compliance_chain fits a Bayesian non-compliance model by running a single chain of the data augmentation algorithm

Usage

```
compliance_chain(dat, outcome_model = NULL, exclusion_restriction = T,
  strong_access = T, starting_values = NULL, hyper_parameters = NULL,
  n_iter = 10000, n_burn = 1000)
```

Arguments

dat	a data frame. The first column of the data frame should be the outcome variable, the second column should be the treatment assignment variable, and the third column should be the indicator for the treatment actually received.
outcome_model	a character string indicating how the outcome should be modeled. Either "normal" for the normal model or "binary" for the binary model.
exclusion_restriction	a logical value indicating whether the exclusion restriction assumption should be made.
strong_access	a logical value indicating whether the strong access monotonicity assumption should be made.
starting_values	the initial parameter values. If NULL, then the initial parameter values are based on either a random draw from the prior distribution (for the binary model) or sample statistics (for the normal model).
hyper_parameters	a numerical vector containing the values that determine the prior distribution for the model parameters. If NULL, then the hyper parameters are chosen to give non-informative or reference priors.
n_iter	number of iterations of the data augmentation algorithm to perform.
n_burn	number of initial iterations to discard.

Value

a matrix containing the draws from the posterior distribution.

Examples

```
# runs 10 iterations of the data augmentation algorithm on a subset of the vitaminA data
set.seed(4923)
compliance_chain(vitaminA[sample(1:nrow(vitaminA), 1000),], outcome_model = "binary",
exclusion_restriction = TRUE, strong_access = TRUE, n_iter = 10, n_burn = 1)
```

noncomplyR	<i>noncomplyR</i>
------------	-------------------

Description

noncomplyR

Examples

```
## See function help files for examples
```

summarize_chain	<i>Posterior Inference based on a Sample from the Posterior</i>
-----------------	---

Description

summarize_chain provides posterior summaries based off a sample from the posterior distribution.

Usage

```
summarize_chain(chain, digits = 3)
```

Arguments

- chain a numeric vector containing the samples from the posterior distribution.
- digits the number of decimal places

Value

a list containing the posterior mean, median, and quantile-based credible intervals calculated from the values in the chain.

Examples

```
# Suppose the posterior distribution was Normal(15, 5)
posterior_chain <- rnorm(100, 15, 5); summarize_chain(posterior_chain)
```

vitaminA*Vitamin A Randomized Trial Data Set*

Description

A dataset containing the results of an RCT investigating the effect of Vitamin A supplementation on mortality. The variables are as follows:

Usage

```
data(vitaminA)
```

Format

A data frame with 23682 rows and 3 variables

Details

- `survived`. indicator of survival (`survived = 1`) or death (`survived = 0`)
- `vitaminA_assigned`. indicator of assignment to Vitamin A (`vitaminA_assigned = 1`) or placebo (`vitaminA_assigned = 0`)
- `vitaminA_received`. indicator of whether Vitamin A supplements were received (`vitaminA_received = 1`) or not received (`vitaminA_received = 0`)

Examples

```
## Not run: vitaminA
```

Index

* **datasets**

vitaminA, [5](#)

cace, [2](#)

compliance_chain, [3](#)

noncomplyR, [4](#)

noncomplyR-package (noncomplyR), [4](#)

summarize_chain, [4](#)

vitaminA, [5](#)