Package: PSor (via r-universe)

October 8, 2026

Type Package

Title Semiparametric Principal Stratification Analysis Beyond Monotonicity

Version 0.1.0

Description Estimates principal causal effects under principal stratification using a margin-free, conditional odds ratio sensitivity parameter. This framework unifies the monotonicity assumption and the counterfactual intermediate independence assumption, allowing for robust analysis when monotonicity may not hold. Computes point estimates, standard errors, and confidence intervals for conditionally doubly robust and debiased machine learning estimators. The methodological details are described in Tong, Kahan, Harhay, and Li (2025) <doi:10.48550/arXiv.2501.17514>.

License MIT + file LICENSE

URL https://github.com/deckardt98/PSor

BugReports https://github.com/deckardt98/PSor/issues

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Imports stats, SuperLearner, caret, dplyr, geex, magrittr, numDeriv

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Config/pak/sysreqs cmake make libicu-dev

Repository https://deckardt98.r-universe.dev

Date/Publication 2026-09-30 03:24:05 UTC

RemoteUrl https://github.com/deckardt98/psor

RemoteRef HEAD

RemoteSha 6d960d61e02c363f796efc8615ca45888bfdef24

Contents

PSor.fit 2

Index **5**

`PSor.fit` *Compute the principal causal effects with an assumed odds ratio sensitivity parameter*

Description

'PSor.fit' is the main function of the package. It estimates the principal causal effects under principal stratification, assuming an odds ratio sensitivity parameter. The function returns point estimates, standard error estimates, and the confidence intervals for both parametric Conditionally Doubly Robust (CDR) estimators and Debiased Machine Learning (DML) estimators.

Usage

```
PSor.fit(
  out.formula,
  ps.formula,
  pro.formula,
  df,
  out.name,
  int.name,
  trt.name,
  cov.names,
  or,
  SLmethods = c("SL.glm", "SL.rpart", "SL.nnet"),
  n.fold,
  scale = "RD",
  alpha = 0.05
)
```

Arguments

`out.formula`	A formula for the outcome model, e.g., 'Y ~ X1 + X2'.
`ps.formula`	A formula for the principal score model (for the intermediate outcome D), e.g., 'D ~ X1 + X2'.
`pro.formula`	A formula for the propensity score model (for the treatment Z), e.g., 'Z ~ X1 + X2'.
`df`	The data frame containing the variables.
`out.name`	A character string specifying the name of the outcome variable (Y).
`int.name`	A character string specifying the name of the intermediate variable (D).
`trt.name`	A character string specifying the name of the treatment variable (Z).

cov.names	A character vector of covariate names.
or	The odds ratio sensitivity parameter. Can be a single numeric value, a vector of values corresponding to each individual, or 'Inf' for the monotonicity assumption.
SLmethods	A character vector of learning algorithms to be used by 'SuperLearner' for the DML estimators Defaults to 'c("SL.glm", "SL.rpart", "SL.nnet")'.
n.fold	The number of folds for cross-fitting in the DML procedure.
scale	The scale for the causal estimand. Can be "RD" (Risk Difference), "RR" (Risk Ratio), or "OR" (Odds Ratio). Defaults to "RD".
alpha	The significance level for constructing confidence intervals. Defaults to 0.05.

Value

A data frame containing the point estimates, standard errors, and confidence intervals for the causal effects within each principal stratum ("Always-Takers", "Compliers", "Never-Takers", and "Defiers" if applicable). Both CDR and DML results are provided.

Examples

```
# Generate a perfectly balanced, minimal toy dataset for extremely fast CRAN checks
set.seed(2026)
n <- 50

# Generate 1 covariate
X1 <- rnorm(n)

# Force perfectly balanced treatment (Z) and intermediate outcome (D)
Z <- sample(rep(c(0, 1), n / 2))
D <- sample(rep(c(0, 1), n / 2))

# Generate final outcome
Y <- rnorm(n, mean = Z + D + X1, sd = 1)

my_data <- data.frame(Y, D, Z, X1)

# Run estimation using a single covariate, SL.mean, and 2 folds for < 1s execution
results <- PSor.fit(
  out.formula = Y ~ X1,
  ps.formula = D ~ X1,
  pro.formula = Z ~ X1,
  df = my_data,
  out.name = "Y",
  int.name = "D",
  trt.name = "Z",
  cov.names = "X1",
  or = Inf,
  SLmethods = "SL.mean",
  n.fold = 2,
  scale = "RD"
)
```

```
print(results)
```

Index

PSor.fit, 2