Package 'remoteoutcome'

October 24, 2025

Type Package

Title Program Evaluation with Remotely Sensed Variables

Version 0.1.0

Description Implements nonparametric methods for estimating treatment effects when outcomes are measured using remotely sensed variables (RSVs) such as satellite images or mobile phone data. Based on Rambachan, Singh, and Viviano (2025) ``Program Evaluation with Remotely Sensed Outcomes'' <arXiv:2411.10959>. The package provides estimators that combine experimental and observational samples under a stability assumption, allowing for valid inference without rate conditions on machine learning predictions.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports dplyr (>= 1.0.0),
randomForest (>= 4.6-14),
boot (>= 1.3-28),
stats

Suggests modelsummary (>= 0.7.0),
ggplot2 (>= 3.3.0),
knitr,
rmarkdown,
testthat (>= 3.0.0),
fixest (>= 0.10.0)

VignetteBuilder knitr

RoxygenNote 7.3.2

URL https://github.com/asheshrambachan/remoteoutcome

BugReports https://github.com/asheshrambachan/remoteoutcome/issues

Contents

`coef.rsv` *Extract coefficients from rsv objects*

Description

Extract coefficients from rsv objects

Usage

```
## S3 method for class 'rsv'
coef(object, ...)
```

Arguments

`object`	An object of class "rsv".
`...`	Additional arguments (unused).

Value

The treatment effect coefficient.

`confint.rsv` *Confidence intervals for rsv objects*

Description

Confidence intervals for rsv objects

Usage

```
## S3 method for class 'rsv'
confint(object, parm = "D", level = 0.95, ...)
```

Arguments

`object`	An object of class "rsv".
`parm`	Parameter name. Must be "D" (default).
`level`	Confidence level (default 0.95). Note: this returns the CI computed during estimation based on the alpha parameter. To change the confidence level, re-run `rsv_estimate()` with a different alpha.
`...`	Additional arguments (unused).

Value

A matrix with lower and upper confidence bounds.

print.rsv *Print method for rsv objects*

Description

Print method for rsv objects

Usage

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

Arguments

x	An object of class "rsv".
...	Additional arguments (unused).

rsv_estimate *RSV Treatment Effect Estimator*

Description

Estimates treatment effects using remotely sensed variables (RSVs) following Rambachan, Singh, and Viviano (2025). Implements Algorithm 1 from the main text for binary outcomes without pretreatment covariates.

Usage

```
rsv_estimate(
  Y = NULL,
  D = NULL,
  S_e = NULL,
  S_o = NULL,
  R = NULL,
  pred_Y = NULL,
  pred_D = NULL,
  pred_S_e = NULL,
  pred_S_o = NULL,
  theta_init = NULL,
  eps = 0.01,
  method = c("split", "crossfit", "none"),
  ml_params = list(),
  se = TRUE,
  se_params = list()
)
```

Arguments

`Y`	Outcome variable (binary, `NA` where not observed).
`D`	Treatment indicator (binary, `NA` where not observed).
`S_e`	Experimental sample indicator (0 or 1).
`S_o`	Observational sample indicator (0 or 1).
`R`	Remotely sensed variable. Required if predictions are not provided.
`pred_Y`	(Optional) Predicted $P[Y \mid R, S_o = 1]$, $\text{PRED}_Y(R)$. If provided, other predictions must also be provided.
`pred_D`	(Optional) Predicted $P[D \mid R, S_e = 1]$, $\text{PRED}_D(R)$.
`pred_S_e`	(Optional) Predicted $P(S_e = 1 \mid R)$, $\text{PRED}_{S_e}(R)$.
`pred_S_o`	(Optional) Predicted $P(S_o = 1 \mid R)$, $\text{PRED}_{S_o}(R)$.
`theta_init`	Initial estimate of the treatment effect on the train data.
`eps`	Small constant for numerical stability of `sigma2` estimate (default `1e-2`).
`method`	Prediction fitting method; one of `"split"` (default), `"crossfit"`, or `"none"`. `"split"` = simple sample split; `"crossfit"` = K-fold cross-fitting; `"none"` = use all data for training/testing.
`ml_params`	List of parameters for random forest: **ntree** Number of trees **classwt_Y** Class weights for `pred_Y` model; default `c(10, 1)`) **seed** User specified seed passed to each `ranger` function for reproducibility; default `NULL` **cores** Number of cores used by 'ranger' for parallel training; default `1` **nfolds** Number of folds for cross-fitting (default `5`). **train_ratio** Proportion for training in sample split (default `0.5`).
`se`	Logical; compute standard errors via bootstrap? (default `TRUE`).
`se_params`	List of bootstrap parameters: **B** Number of bootstrap replications (default `1000`) **fix_seed** If `TRUE`, deterministic seeding is used with 'set.seed(b)' for the *b*-th replication (default `FALSE`) **cores** Number of cores for bootstrap replications (default `1`). **clusters** Clusters for the bootstrap. If `NULL`, uses individual-level bootstrap

Details

The function supports two interfaces:

1. Provide fitted predictions `pred_Y`, `pred_D`, `pred_S_e`, `pred_S_o` directly.
2. Provide raw data `(Y, D, S, R)` and the function fits predictions using random forests.

The function also supports sample splitting and K-fold cross-fitting for prediction fitting.

Value

A list of class "rsv" with components:

coef Treatment effect estimate.

se Standard error (if se = TRUE).

denominator_se Standard error of the denominator of the treatment effect (if se = TRUE)

n_obs Sample size in observational sample.

n_exp Sample size in experimental sample.

n_both Sample size in both samples.

numerator Numerator of the treatment effect estimate

denominator Denominator of the treatment effect estimate

method Prediction fitting method used.

call The matched call.

Examples

```
## Not run:
library(dplyr)
library(remoteoutcome)

# Example data: data_real (included in package)
Y <- data_real$Ycons # binary outcome
D <- data_real$D # binary treatment
R <- data_real %>% select(starts_with("luminosity"), starts_with("satellite")) # remotely sensed variable
S_e <- !is.na(D) & (rowSums(is.na(R)) == 0) # experimental sample indicator (Observe D, R)
S_o <- !is.na(Y) & (rowSums(is.na(R)) == 0) #  observational sample indicator (Observe Y, R)
clusters <- data_real$clusters # Subdistrict-level cluster identifiers

# Example 1: No sample splitting
result <- rsv_estimate(
Y = Y, D = D, S_e = S_e, S_o = S_o, R = R,
method = "none",
ml_params = list(seed = 42, cores = 7),
se_params = list(fix_seed = TRUE, clusters = clusters, cores = 7)
)
print(result)

# Example 2: With sample splitting
result <- rsv_estimate(
Y = Y, D = D, S_e = S_e, S_o = S_o, R = R,
method = "split",
ml_params = list(train_ratio = 0.5, seed = 42, cores = 7),
se_params = list(fix_seed = TRUE, clusters = clusters)
)
print(result)

# Example 3: With cross fitting
result <- rsv_estimate(
Y = Y, D = D, S_e = S_e, S_o = S_o, R = R,
method = "crossfit",
ml_params = list(nfold = 5, seed = 42, cores = 7),
se_params = list(fix_seed = TRUE, clusters = clusters)
)
```

```
print(result)

# Example 4: Custom ML parameters
result <- rsv_estimate(
  Y = Y,
  D = D,
  S_e = S_e,
  S_o = S_o,
  R = R,
  eps = 1e-2,
  method = "none",
  ml_params = list(       # Customize random forest parameters:
    ntree = 100,          #   Number of trees
    classwt_Y = c(10, 1), #   Class weights for PRED_Y model
    seed = 42,            #   A random seed for each RF for reproducibility
    cores = 7             #   Number of cores used for training
  ),
  se = TRUE,
  se_params = list(       # Customize cluster-bootstrap standard errors:
    B = 1000,             #   Number of bootstrap replications
   clusters = clusters,  #  Cluster identifiers for clustered sampling, if not provided, use individual-level bo
    fix_seed = TRUE,      #   Enables deterministic seeding for reproducibility
    cores = 7             #   Number of cores for bootstrap replications
  )
)
print(result)

# Example 5: User provides fitted predictions
# Example data: pred_real_Ycons (included in package)
result <- rsv_estimate(
  Y = pred_real_Ycons$Y,
  D = pred_real_Ycons$D,
  S_e = pred_real_Ycons$S_e,
  S_o = pred_real_Ycons$S_o,
  pred_Y = pred_real_Ycons$pred_Y,
  pred_D = pred_real_Ycons$pred_D,
  pred_S_e = pred_real_Ycons$pred_S_e,
  pred_S_o = pred_real_Ycons$pred_S_o,
  se = TRUE,
 se_params = list(B = 1000, fix_seed = TRUE, cores = 7, clusters = pred_real_Ycons$clusters),
)
print(result)

## End(Not run)
```

`summary.rsv` *Summary method for rsv objects*

Description

Summary method for rsv objects

Usage

```
## S3 method for class 'rsv'
summary(object, ...)
```

Arguments

object	An object of class "rsv".
...	Additional arguments (unused).

---

vcov.rsv *Extract variance-covariance matrix from rsv objects*

Description

Extract variance-covariance matrix from rsv objects

Usage

```
## S3 method for class 'rsv'
vcov(object, ...)
```

Arguments

object	An object of class "rsv".
...	Additional arguments (unused).

Value

A 1x1 matrix containing the variance of the treatment effect.

Index