

# Package ‘remoteoutcome’

November 17, 2025

**Type** Package

**Title** Program Evaluation with Remotely Sensed Variables

**Version** 1.0.0

**Description** Provides tools for estimating treatment effects using remotely sensed variables (RSVs) such as satellite images or mobile phone data. Implements the nonparametric methods developed in Rambachan, A., Singh, R., and Viviano, D. (2025) ``Program Evaluation with Remotely Sensed Outcomes" <[arXiv:2411.10959](https://arxiv.org/abs/2411.10959)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**LazyDataCompression** xz

**Depends** R (>= 4.0.0)

**Imports** ranger (>= 0.17.0),  
dplyr,  
parallel,  
stats,  
utils

**Suggests** ggplot2,  
fixest,  
knitr,  
rmarkdown,  
kableExtra,  
latex2exp,  
stringr,  
tibble,  
tidyr

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

**URL** <https://github.com/asheshrambachan/remoteoutcome>

**BugReports** <https://github.com/asheshrambachan/remoteoutcome/issues>

Contents

coef.rsv . . . . .	2
confint.rsv . . . . .	2
create_data_real . . . . .	3
create_data_synth . . . . .	4
pred_real_Ycons . . . . .	6
print.rsv . . . . .	8
rsv_estimate . . . . .	9
summary.rsv . . . . .	12
vcov.rsv . . . . .	13
<b>Index</b>	<b>14</b>

---

coef.rsv	<i>Extract coefficients from rsv objects</i>
----------	--

---

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	<i>Confidence intervals for rsv objects</i>
-------------	---

---

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 <code>rsv_estimate()</code> with a different alpha.
...	Additional arguments (unused).

**Value**

A matrix with lower and upper confidence bounds.

---

create_data_real	<i>Create Real Experimental/Observational Split Dataset</i>
------------------	---

---

**Description**

Transforms smartcard\_data into the real experimental/observational split where treatment is observed in "Experimental: Treated (2010)", "Experimental: Untreated (2011)", and "Experimental: Untreated (2012)" waves and outcomes are observed in the "Experimental: Untreated (2011)" and "Observational (N/A)" waves.

**Usage**

```
create_data_real(data)
```

**Arguments**

data	smartcard_data data frame included in the package.
------	--

**Details**

The function implements the real experimental design through the following steps:

1. Villages from "Experimental: Treated (2010)", "Experimental: Untreated (2011)", "Experimental: Untreated (2012)" waves are marked as `S_e = TRUE`
2. Villages from "Experimental: Untreated (2011)" and "Observational (N/A)" waves are marked as `S_o = TRUE`
3. Final sample indicator is created:
  - `S = "both"` if `S_e = TRUE` and `S_o = TRUE` (experimental and observational villages)
  - `S = "e"` if `S_e = TRUE` and `S_o = FALSE` (experimental only)
  - `S = "o"` if `S_e = FALSE` and `S_o = TRUE` (observational only)
4. Treatment visible only when `S = "e"` or `"both"`; outcomes visible only when `S = "o"` or `"both"`

**Value**

A data frame similar to `smartcard_data`, containing:

**shrid2** SHRUG village identifier

**spillover\_20km** Spillover indicator

**S** Sample indicator: "e" = experimental only, "o" = observational only, "both" = experimental and observational samples

**D** Treatment indicator, NA when not observed (only observed when S = "e" or "both")

**Ycons, Ylowinc, Ymidinc** Outcome variables, NA when not observed (only observed when S = "o" or "both")

**clusters** Cluster identifier (Subdistrict (mandal) name + District name in Andhra Pradesh)

**luminosity\_\*** VIIRS nighttime lights features

**satellite\_\*** MOSAICS satellite feature

**See Also**

[smartcard\\_data](#) for the input dataset structure, [create\\_data\\_synth](#) for creating synthetic sample splits

**Examples**

```
## Not run:
# Load complete dataset
data(smartcard_data, package = "remoteoutcome")

# Create real experimental/observational split
data_real <- create_data_real(smartcard_data)

# Sample membership distribution
table(data_real$S)

# Treatment and outcome overlap
with(data_real, table(observe_D = !is.na(D), observe_Y = !is.na(Ycons)))

## End(Not run)
```

---

create\_data\_synth

*Create Synthetic Experimental/Observational Split Dataset*

---

**Description**

Transforms the complete dataset (`smartcard_data`) into a synthetic experimental/observational split where top 50 observational sample. This creates an artificial separation for testing RSV methods.

**Usage**

```
create_data_synth(data)
```

## Arguments

**data** smartcard\_data data frame included in the package.

## Details

The function creates a synthetic experimental/observational split through the following steps:

1. Original experimental villages are marked as `S_e = TRUE`
2. Top 50
3. Final sample indicator is created:
  - `S = "both"` if `S_e = TRUE` and `S_o = TRUE` (experimental and observational villages)
  - `S = "e"` if `S_e = TRUE` and `S_o = FALSE` (experimental only)
  - `S = "o"` if `S_e = FALSE` and `S_o = TRUE` (observational only)
4. Treatment visible only when `S = "e"` or `"both"`; outcomes visible only when `S = "o"` or `"both"`

## Value

A data frame similar to `smartcard_data`, containing:

**shrid2** SHRUG village identifier

**spillover\_20km** Spillover indicator

**S** Sample indicator: "e" = experimental only, "o" = observational only, "both" = experimental and observational samples

**D** Treatment indicator, NA when not observed (only observed when `S = "e"` or `"both"`)

**Ycons, Ylowinc, Ymidinc** Outcome variables, NA when not observed (only observed when `S = "o"` or `"both"`)

**clusters** Cluster identifier (Subdistrict (mandal) name + District name in Andhra Pradesh)

**luminosity\_\*** VIIRS nighttime lights features

**satellite\_\*** MOSAICS satellite features

## See Also

[smartcard\\_data](#) for the input dataset structure, [create\\_data\\_real](#) for creating real experimental splits

## Examples

```
## Not run:
# Load complete dataset
data(smartcard_data, package = "remoteoutcome")

data_synth <- create_data_synth(smartcard_data)

# Sample distribution
table(data_synth$S)

# Treatment and outcome overlap
with(data_synth, table(observe_D = !is.na(D), observe_Y = !is.na(Ycons)))

## End(Not run)
```

---

pred_real_Ycons	<i>Predicted Values and Observations for Consumption Outcome (Real Sample Split)</i>
-----------------	--

---

## Description

Pre-computed predictions and observations from the RSV estimation procedure for the consumption outcome (Ycons) using the real experimental/observational sample split. This dataset allows users to run `rsv_estimate` with pre-fitted predictions, avoiding the computational cost of re-fitting random forest models.

## Usage

pred\_real\_Ycons

## Format

A data frame with 8,312 rows and 9 columns:

**Y** Binary outcome: consumption indicator (1 = bottom quartile, 0 = otherwise). NA for experimental-only observations where outcomes are not observed.

**D** Binary treatment indicator (1 = treated, 0 = control). NA for observational-only observations where treatment is not assigned.

**S\_e** Experimental sample indicator: 1 if unit is in experimental sample (treatment observed), 0 otherwise.

**S\_o** Observational sample indicator: 1 if unit is in observational sample (outcome observed), 0 otherwise.

**pred\_Y** Predicted probability  $P(Y = 1 \mid R, S_o = 1)$ , where  $R$  includes VIIRS nighttime lights and MOSAICS satellite features. Fitted using random forest on observational sample.

**pred\_D** Predicted probability  $P(D = 1 \mid R, S_e = 1)$ . Fitted using random forest on experimental sample.

**pred\_S\_e** Predicted probability  $P(S_e = 1 \mid R)$ . Fitted using random forest on full sample.

**pred\_S\_o** Predicted probability  $P(S_o = 1 \mid R)$ . Fitted using random forest on full sample.

**clusters** Cluster identifier (subdistrict name + district name) for computing cluster-robust standard errors.

## Details

### Generation Process:

This dataset was generated using the following procedure:

#### 1. Data preparation:

- Started with data\_real (real experimental/observational split)
- Merged with satellite features (MOSAICS 4,000-dimensional features)
- Constructed remotely sensed variable  $R$  from VIIRS nighttime lights (2012-2021) and MOSAICS features

#### 2. Sample indicators:

- $S_e = 1$  for units where treatment  $D$  and covariates  $R$  are both observed (experimental sample)
- $S_o = 1$  for units where outcome  $Y$  and covariates  $R$  are both observed (observational sample)
- Some units have both  $S_e = 1$  and  $S_o = 1$  (overlap sample)

### 3. Prediction fitting:

- Used method = "none" (no sample splitting - all data used for both training and prediction)
- Fitted four random forest models using ranger:
  - pred\_Y:  $E[Y \mid R, S_o = 1]$  fitted on observational sample
  - pred\_D:  $E[D \mid R, S_e = 1]$  fitted on experimental sample
  - pred\_S\_e:  $P(S_e = 1 \mid R)$  fitted on full sample
  - pred\_S\_o:  $P(S_o = 1 \mid R)$  fitted on full sample
- Random forest parameters: 100 trees, class weights  $c(10, 1)$  for pred\_Y model (up-weighting rare outcome), seed = 42

### 4. Initial estimate:

- Computed theta\_init (initial treatment effect estimate) on the training data using the predictions
- Stored as an attribute: attr(pred\_real\_Ycons, "theta\_init")

## Attribute

The dataset has one attribute:

**theta\_init** Initial estimate of the treatment effect computed on the training data. This is used as a starting value in the RSV estimation procedure. Access with attr(pred\_real\_Ycons, "theta\_init").

## See Also

- [rsv\\_estimate](#) for using this dataset to estimate treatment effects
- [smartcard\\_data](#) for the underlying data without predictions
- [vignette\(package = "remoteoutcome"\)](#) for examples

## Examples

```
# Load the dataset
data(pred_real_Ycons, package = "remoteoutcome")

# Examine structure
str(pred_real_Ycons)

# Check sample sizes
table(S_e = pred_real_Ycons$S_e, S_o = pred_real_Ycons$S_o)

# View prediction distributions
summary(pred_real_Ycons[, c("pred_Y", "pred_D", "pred_S_e", "pred_S_o")])

# Access the initial treatment effect estimate
attr(pred_real_Ycons, "theta_init")

## Not run:
```

```

# Estimate treatment effect using pre-computed predictions
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,
  method = "predictions",
  theta_init = attr(pred_real_Ycons, "theta_init"),
  se = TRUE,
  se_params = list(
    B = 1000,
    fix_seed = TRUE,
    clusters = pred_real_Ycons$clusters
  ),
  cores = 7
)

# View results
print(result)
confint(result)

## End(Not run)

```

---

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("crossfit", "split", "none", "predictions"),
  ml_params = list(),
  se = TRUE,
  se_params = list(),
  cores = 1
)
```

**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: <b>ntree</b> Number of trees <b>classwt_Y</b> Class weights for pred_Y model; default c(10, 1)) <b>seed</b> User specified seed passed to each ranger function for reproducibility; default NULL <b>nfolds</b> Number of folds for cross-fitting (default 5). <b>train_ratio</b> 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>B</b> Number of bootstrap replications (default 1000) <b>fix_seed</b> If TRUE, deterministic seeding is used with 'set.seed(b)' for the *b*-th replication (default FALSE) <b>clusters</b> Clusters for the bootstrap. If NULL, uses individual-level bootstrap
cores	Number of cores used by either 'ranger' or bootstrap replications; default 1

## 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)
path_to_satellite_features <- "path/to/satellite_features.rds"
satellite_features <- readRDS(path_to_satellite_features) %>%
  select(-contains("lat"), -contains("lon"))
```

```

data("data_real", package = "remoteoutcome")
data_real <- inner_join(data_real, satellite_features, by = "shrid2")

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),
  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),
  se_params = list(fix_seed = TRUE, clusters = clusters),
  cores = 7
)
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),
  se_params = list(fix_seed = TRUE, clusters = clusters),
  cores = 7
)
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
  ),

```

```

    se = TRUE,
    se_params = list(      # Customize cluster-bootstrap standard errors:
      B = 1000,            #   Number of bootstrap replications
      clusters = clusters, #   Cluster identifiers for clustered sampling
      fix_seed = TRUE,     #   Enables deterministic seeding for reproducibility
    ),
    cores = 7
  )
  print(result)

# Example 5: User provides fitted predictions
# Example data: pred_real_Ycons (included in package)
data("pred_real_Ycons", package = "remoteoutcome")

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,
  method = "predictions",
  theta_init = attr(pred_real_Ycons, "theta_init"),
  # ml_params = list(
  #   train_ratio = 0.2, #   If theta_init is not provided, 20% of the data
  #                     #   will be used to estimate theta_init.
  #   seed = 42         #   A random seed for each RF for reproducibility
  # ),
  se = TRUE,
  se_params = list(B = 1000, fix_seed = TRUE, clusters = pred_real_Ycons$clusters),
  cores = 7
)
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

## \* datasets

pred\_real\_Ycons, [6](#)

coef.rsv, [2](#)

confint.rsv, [2](#)

create\_data\_real, [3](#), [5](#)

create\_data\_synth, [4](#), [4](#)

pred\_real\_Ycons, [6](#)

print.rsv, [8](#)

rsv\_estimate, [6](#), [7](#), [9](#)

smartcard\_data, [4](#), [5](#), [7](#)

summary.rsv, [12](#)

vcov.rsv, [13](#)