Package: OPSR (via r-universe)

October 13, 2026

Title Ordered Probit Switching Regression

Version 1.0.1

Description Estimates ordered probit switching regression models - a Heckman type selection model with an ordinal selection and continuous outcomes. Different model specifications are allowed for each treatment/regime. For more details on the method, see Wang & Mokhtarian (2024) <doi:10.1016/j.tra.2024.104072> or Chiburis & Lokshin (2007) <doi:10.1177/1536867X0700700202>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports car, Formula, graphics, MASS, maxLik, methods, mvtnorm, Rcpp, Rdpack (>= 0.7), sandwich, stats, texreg, utils

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 3.5.0)

LazyData true

RdMacros Rdpack

Suggests gridExtra, gridGraphics, sampleSelection, scales, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/dheimgartner/OPSR

BugReports https://github.com/dheimgartner/OPSR/issues

NeedsCompilation yes

Author Daniel Heimgartner [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-0643-8690>), Xinyi Wang [aut] (ORCID: <https://orcid.org/0000-0002-3564-9147>)

Maintainer Daniel Heimgartner <d.heimgartners@gmail.com>

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

Date/Publication 2025-07-28 15:50:18 UTC

RemoteUrl https://github.com/cran/OPSR

RemoteRef HEAD

RemoteSha 8e2d95a2965884bb1e8f8ef6acaf33821058b886

Contents

OPSR-package . . . 3
anova.opsr . . . 3
extract,opsr-method . . . 4
loglik_cpp . . . 5
model.frame.opsr . . . 6
model.matrix.opsr . . . 7
opsr . . . 7
opsr.fit . . . 10
opsr_2step . . . 11
opsr_check_omp . . . 12
opsr_check_start . . . 12
opsr_max_threads . . . 13
opsr_null_model . . . 13
opsr_prepare_coefs . . . 14
opsr_simulate . . . 15
opsr_step . . . 15
opsr_te . . . 16
pairs.opsr.te . . . 17
plot.opsr . . . 19
predict.opsr . . . 20
print.anova.opsr . . . 21
print.ate . . . 22
print.opsr.te . . . 23
print.summary.opsr . . . 23
print.summary.opsr.te . . . 24
print.te . . . 25
summary.opsr . . . 25
summary.opsr.te . . . 26
telework_data . . . 27
timeuse_data . . . 29

Index **31**

OPSR-package *OPSR: Ordered Probit Switching Regression*

Description

Estimates ordered probit switching regression models - a Heckman type selection model with an ordinal selection and continuous outcomes. Different model specifications are allowed for each treatment/regime. For more details on the method, see Wang & Mokhtarian (2024) doi:10.1016/j.tra.2024.104072 or Chiburis & Lokshin (2007) doi:10.1177/1536867X0700700202.

Author(s)

Maintainer: Daniel Heimgartner <d.heimgartners@gmail.com> (ORCID) [copyright holder]

Authors:

- Xinyi Wang <xinyi174@mit.edu> (ORCID)

See Also

Useful links:

- https://github.com/dheimgartner/OPSR
- Report bugs at https://github.com/dheimgartner/OPSR/issues

anova.opsr *ANOVA for OPSR Model Fits*

Description

Conducts likelihood ratio tests for one or more OPSR model fits.

Usage

```
## S3 method for class 'opsr'
anova(object, ...)
```

Arguments

object	an object of class "opsr".
...	additional objects of class "opsr". See also the 'Details' section.

Details

If only a single object is passed then the model is compared to the null model (opsr_null_model). If more than one object is specified, a likelihood ratio test is conducted for each pair of neighboring models. It is conventional to list the models from smallest to largest, but this is up to the user.

Value

An object of class "anova.opsr".

See Also

stats::anova, print.anova.opsr

Examples

```
sim_dat <- opsr_simulate()
dat <- sim_dat$data
model <- ys | yo ~ xs1 + xs2 | xo1 + xo2
fit <- opsr(model, dat)
fit_null <- opsr_null_model(fit)
fit_intercept <- update(fit, ~ . | 1)

anova(fit)
anova(fit_null, fit_intercept, fit)
```

---

extract,opsr-method *Extract Method for OPSR Model Fits*

Description

This is the main method called when using functions from the texreg-package.

Usage

```
## S4 method for signature 'opsr'
extract(
  model,
  beside = FALSE,
  include.structural = TRUE,
  include.selection = TRUE,
  include.outcome = TRUE,
  include.pseudoR2 = FALSE,
  include.R2 = FALSE,
  repeat.gofs = FALSE,
  ...
)
```

Arguments

model	an object of class "opsr".
beside	if TRUE, prints structural, selection and outcome coefficients side-by-side.
include.structural	whether or not structural coefficients should be printed.

include.selection
: whether or not selection coefficients should be printed.

include.outcome
: whether or not outcome coefficients should be printed.

include.pseudoR2
: whether or not the pseudo R2 statistic for the selection component should be printed. See also the 'Details' section.

include.R2
: whether or not the R2 statistic for the outcome components should be printed.

repeat.gofs
: if beside = TRUE whether or not to repeat the gofs.

...
: additional arguments passed to summary.opsr.

Details

The extract method is called internally. Higher-level functions from the texreg-package pass arguments via ... to extract.

include.pseudoR2 reports both the "equally likely" (EL) and "market share" (MS) pseudo R2.

Value

A texreg-class object representing the statistical model.

See Also

texreg-package, texreg::texreg, texreg::screenreg and related functions.

Examples

```
sim_dat <- opsr_simulate()
dat <- sim_dat$data
model <- ys | yo ~ xs1 + xs2 | xo1 + xo2
fit <- opsr(model, dat)
fit_null <- opsr_null_model(fit)
fit_intercept <- update(fit, ~ . | 1)

texreg::screenreg(fit)
texreg::screenreg(fit, beside = TRUE)
texreg::screenreg(fit, beside = TRUE, include.pseudoR2 = TRUE, include.R2 = TRUE)
texreg::screenreg(list(fit_null, fit_intercept, fit))
```

---

loglik_cpp *Interface to C++ Log-Likelihood Implementation*

Description

This is the main computation engine wrapped by opsr.fit.

Usage

```
loglik_cpp(theta, W, X, Y, weights, nReg, nThreads)
```

Arguments

theta	named coefficient vector as parsed from formula interface opsr.
W	list of matrices with explanatory variables for selection process for each regime.
X	list of matrices with expalanatory varialbes for outcome process for each regime.
Y	list of vectors with continuous outcomes for each regime.
weights	vector of weights. See also opsr.
nReg	integer number of regimes.
nThreads	number of threads to be used by OpenMP (should be max. nReg).

Value

Numeric vector of (weighted) log-likelihood contributions.

See Also

opsr.fit, loglik_R

---

model.frame.opsr *Extracting the Model Frame from OPSR Model Fits*

Description

Extracting the Model Frame from OPSR Model Fits

Usage

```
## S3 method for class 'opsr'
model.frame(formula, ...)
```

Arguments

formula	an object of class "opsr".
...	a mix of further arguments such as data, na.action or subset, passed to the default method.

Value

A data.frame containing the variables used in formula$formula.

See Also

stats::model.frame

model.matrix.opsr *Construct Design Matrices for OPSR Model Fits*

Description

Construct Design Matrices for OPSR Model Fits

Usage

```
## S3 method for class 'opsr'
model.matrix(object, data, .filter = NULL, ...)
```

Arguments

object	an object of class "opsr".
data	a data frame containing the terms from object$formula. Passed to model.frame.opsr. Can be omitted.
.filter	used internally in predict.opsr for counterfactual predictions.
...	further arguments passed to or from other methods.

Value

A list of lists with the design matrices W (selection process) and X (outcome process). Both of these lists have object$nReg elements (a separate design matrix for each regime).

See Also

model.frame.opsr, stats::model.matrix

opsr *Fitting Ordered Probit Switching Regression Models*

Description

High-level formula interface to the workhorse opsr.fit.

Usage

```
opsr(
  formula,
  data,
  subset,
  weights,
  na.action,
  start = NULL,
```

```
  fixed = NULL,
  method = "BFGS",
  iterlim = 1000,
  printLevel = 2,
  nThreads = 1,
  .get2step = FALSE,
  .useR = FALSE,
  .censorRho = TRUE,
  .loglik = FALSE,
  ...
)
```

Arguments

formula	an object of class "Formula" "formula": A symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which opsr is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process. (See additional details in the 'Details' section of the model.frame documentation.).
weights	an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If non-NULL, then observation-specific log-likelihood contributions are multiplied by their corresponding weight before summing.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
start	a numeric vector with the starting values (passed to maxLik::maxLik). If no starting values are provided, reasonable values are auto-generated via the Heckman 2-step procedure opsr_2step. The structure of start has to conform with opsr's expectations. See opsr_check_start for further details.
fixed	parameters to be treated as constants at their start values. If present, it is treated as an index vector of start parameters (passed to maxLik::maxLik).
method	maximzation method (passed to maxLik::maxLik).
iterlim	maximum number of iterations (passed to maxLik::maxLik).
printLevel	larger number prints more working information (passed to maxLik::maxLik).
nThreads	number of threads to be used. Do not pass higher number than number of ordinal outcomes. See also opsr_check_omp and opsr_max_threads.
.get2step	if TRUE, returns starting values as generated by opsr_2step. Will not proceed with the maximum likelihood estimation.
.useR	if TRUE usese loglik_R. Go grab a coffe.
.censorRho	if TRUE, rho starting values are censored to lie in the interval [-0.85, 0.85].

.loglik	if TRUE, returns the vector of log-likelihood values given the parameters passed via start.
...	further arguments passed to maxLik::maxLik.

Details

Models for opsr are specified symbolically. A typical model has the form ys | yo ~ terms_s | terms_o1 | terms_o2 | .... ys is the ordered (numeric) response vector (starting from 1, in integer-increasing fashion). For the terms specification the rules of the regular formula interface apply (see also stats::lm). The intercept in the terms_s (selection process) is excluded automatically (no need to specify -1). If the user wants to specify the same process for all continuous outcomes, two processes are enough (ys | yo ~ terms_s | terms_o). Note that the model is poorly identifiable if terms_s == terms_o (same regressors are used in selection and outcome processes).

Value

An object of class "opsr" "maxLik" "maxim".

Examples

```
## simulated data
sim_dat <- opsr_simulate()
dat <- sim_dat$data  # 1000 observations
sim_dat$sigma  # cov matrix of errors
sim_dat$params  # ground truth

## specify a model
model <- ys | yo ~ xs1 + xs2 | xo1 + xo2 | xo1 + xo2 | xo1 + xo2
model <- ys | yo ~ xs1 + xs2 | xo1 + xo2  # since we use the same specification...

## estimate
fit <- opsr(model, dat)

## inference
summary(fit)

## using update and model comparison
fit_updated <- update(fit, ~ . | 1)  # only intercepts for the continuous outcomes
## null model
fit_null <- opsr_null_model(fit)

## likelihood ratio test
anova(fit_null, fit_updated, fit)

## predict
p1 <- predict(fit, group = 1, type = "response")
p2 <- predict(fit, group = 1, counterfact = 2, type = "response")
plot(p1, p2)
abline(a = 0, b = 1, col = "red")

## produce formatted tables
```

```
texreg::screenreg(fit, beside = TRUE, include.pseudoR2 = TRUE, include.R2 = TRUE)
```

---

opsr.fit *Fitter Function for Ordered Probit Switching Regression Models*

Description

This is the basic computing engine called by opsr used to fit ordinal probit switching regression models. Should usually *not* be used directly. The log-likelihood function is implemented in C++ which yields a considerable speed-up. Parallel computation is implemented using OpenMP.

Usage

```
opsr.fit(
  Ws,
  Xs,
  Ys,
  start,
  fixed,
  weights,
  method,
  iterlim,
  printLevel,
  nThreads,
  .useR = FALSE,
  .loglik = FALSE,
  ...
)
```

Arguments

Ws	list of matrices with explanatory variables for selection process for each regime.
Xs	list of matrices with expalanatory varialbes for outcome process for each regime.
Ys	list of vectors with continuous outcomes for each regime.
start	a numeric vector with the starting values (passed to maxLik::maxLik).
fixed	parameters to be treated as constants at their start values. If present, it is treated as an index vector of start parameters (passed to maxLik::maxLik).
weights	a vector of weights to be used in the fitting process. Has to conform with order (w <- weights[order(Z)], where Z is the ordinal outcome).
method	maximzation method (passed to maxLik::maxLik).
iterlim	maximum number of iterations (passed to maxLik::maxLik).
printLevel	larger number prints more working information (passed to maxLik::maxLik).
nThreads	number of threads to be used. Do not pass higher number than number of ordinal outcomes. See also opsr_check_omp and opsr_max_threads.

.useR	if TRUE, usese loglik_R. Go grab a coffe.
.loglik	if TRUE, returns the vector of log-likelihood values given the parameters passed via start.
...	further arguments passed to maxLik::maxLik.

Value

object of class "maxLik" "maxim".

See Also

maxLik::maxLik, loglik_cpp, opsr

---

opsr_2step *Heckman Two-Step Estimation*

Description

This is a utility function, used in opsr and should not be used directly. Tow-step estimation procedure to generate reasonable starting values.

Usage

```
opsr_2step(W, Xs, Z, Ys)
```

Arguments

W	matrix with explanatory variables for selection process.
Xs	list of matrices with expalanatory varialbes for outcome process for each regime.
Z	vector with ordinal outcomes (in integer increasing fashion).
Ys	list of vectors with continuous outcomes for each regime.

Details

These estimates can be retrieved by specifying .get2step = TRUE in opsr.

Value

Named vector with starting values passed to opsr.fit.

Remark

Since the Heckman two-step estimator includes an estimate in the second step regression, the resulting OLS standard errors and heteroskedasticity-robust standard errors are incorrect (Greene 2002).

References

Greene WH (2002). *LIMDEP Version 8.0 Econometric Modeling Guide, vol. 2.*. Econometric Software, Plainview, New York.

See Also

opsr.fit, opsr_prepare_coefs

---

opsr_check_omp *Check Whether* OpenMP *is Available*

Description

Check Whether OpenMP is Available

Usage

```
opsr_check_omp()
```

Value

boolean

---

opsr_check_start *Check the User-Specified Starting Values*

Description

This is a utility function, used in opsr and should not be used directly. It is included here to document the expected structure of opsr's start argument. Makes sure, the start vector conforms to the expected structure. Adds the expected parameter names to the numeric vector. Therefore the user has to conform to the expected order. See 'Details' for further explanation.

Usage

```
opsr_check_start(start, W, Xs)
```

Arguments

start	vector of starting values.
W	matrix with explanatory variables for selection process.
Xs	list of matrices with expalanatory varialbes for outcome process for each regime.

Details

Expected order: 1. kappa threshold parameters (for ordered probit model), 2. parameters of the selection process (names starting with `s_`), 3. parameters of the outcome processes (names starting with `o[0-9]_`), 4. sigma, 5. rho. If the same outcome process specification is used in the `formula`, the starting values have to be repeated (i.e., the length of the `start` vector has to correspond to the total number of estimated parameters in the model).

Value

Named numeric vector conforming to the expected structure.

See Also

`opsr_2step`

`opsr_max_threads` *Check Maximum Number of Threads Available*

Description

Check Maximum Number of Threads Available

Usage

```
opsr_max_threads()
```

Value

integer

See Also

`opsr_check_omp`

`opsr_null_model` *Null Model for OPSR Model fits*

Description

Intercept-only model with no error correlation.

Usage

```
opsr_null_model(object, ...)
```

Arguments

object	an object of class "opsr".
...	further arguments passed to opsr.

Value

An object of class "opsr.null" "opsr".

Examples

```
sim_dat <- opsr_simulate()
dat <- sim_dat$data
model <- ys | yo ~ xs1 + xs2 | xo1 + xo2
fit <- opsr(model, dat)
fit_null <- opsr_null_model(fit)
summary(fit_null)
```

---

opsr_prepare_coefs *Prepares Coefficients for Likelihood Function*

Description

Extracts the coefficients for each regime

Usage

```
opsr_prepare_coefs(theta, nReg)
```

Arguments

theta	named coefficient vector as parsed from formula interface opsr.
nReg	integer number of regimes.

Value

Named list of length nReg

Examples

```
sim_dat <- opsr_simulate()
dat <- sim_dat$data
model <- ys | yo ~ xs1 + xs2 | xo1 + xo2
start <- opsr(model, dat, .get2step = TRUE)
opsr_prepare_coefs(start, 3)
```

opsr_simulate *Simulate Data from an OPSR Process*

Description

Simulates data from an ordered probit process and separate (for each regime) OLS process where the errors follow a multivariate normal distribution.

Usage

```
opsr_simulate(nobs = 1000, sigma = NULL, seed = NULL)
```

Arguments

nobs	number of observations to simulate.
sigma	the covariance matrix of the multivariate normal.
seed	a single value, interpreted as an integer, or NULL passed to set.seed.

Details

Three ordinal outcomes are simulated and the distinct design matrices (W and X) are used (if W == X the model is poorely identified). Variables ys and xs in data correspond to the selection process and yo, xo to the outcome process.

Value

Named list:

params	ground truth parameters.
data	simulated data (as observed by the researcher). See also 'Details' section.
errors	error draws from the multivariate normal (as used in the latent process).
sigma	assumed covariance matrix (to generate errors).

opsr_step *Step Function for OPSR Model Fits*

Description

Excludes all coefficients with p-values below pval and fits again.

Usage

```
opsr_step(object, pval, log = new.env(), .step = 1, ...)
```

Arguments

object	an object of class "opsr".
pval	coefficients with p-values < pval are dropped.
log	environment to keep track of changes to object (in particular variables being eliminated). This could be useful in a function that repeatedly calls opsr_step internally in order to keep track of the full elimination history.
.step	used to generate identifier in log environment.
...	additional arguments passed to update (and hence opsr).

Value

An object of class "opsr".

Examples

```
sim_dat <- OPSR::opsr_simulate()
dat <- sim_dat$data
dat$xo3 <- runif(n = nrow(dat))
dat$xo4 <- factor(sample(c("level1", "leve2", "level3"), nrow(dat), replace = TRUE))
f <- ys | yo ~ xs1 + xs2 + log(xo3) | xo1 + xo2 + xo3 + xo4 | xo1 + xo2 + xo3 | xo1 + xo2
fit <- OPSR::opsr(f, dat, printLevel = 0)
fit_step <- opsr_step(fit, pval = 0.1)
texreg::screenreg(list(fit, fit_step))
```

---

opsr_te *Treatment Effect Computations for OPSR Model Fits*

Description

Treatment Effect Computations for OPSR Model Fits

Usage

```
opsr_te(
  object,
  type = c("response", "unlog-response", "prob", "mills", "correction", "Xb"),
  weights = NULL,
  ...
)
```

Arguments

object	object an object of class "opsr".
type	see predict.opsr for details.
weights	a vector of weights. If NULL then weights from object will be used.
...	additional arguments passed to predict.opsr.

Details

This function only prepares the input to a further call to summary.opsr.te.

Value

An object of class "opsr.te".

See Also

summary.opsr.te

Examples

```
sim_dat <- opsr_simulate()
dat <- sim_dat$data
weights <- runif(nrow(dat))
fit <- opsr(ys | yo ~ xs1 + xs2 | xo1 + xo2, dat = dat, weights = weights,
            printLevel = 0)
te <- opsr_te(fit, type = "response")
print(te)
summary(te)

te_w <- opsr_te(fit, type = "response", weights = rep(1, nrow(dat)))
summary(te_w)

pairs(te)
```

---

pairs.opsr.te *Pairs Plot for OPSR TE Objects*

Description

Pairs Plot for OPSR TE Objects

Usage

```
## S3 method for class 'opsr.te'
pairs(
  x,
  pch = 21,
  labels.diag = paste0("T", 1:x$nReg),
  labels.reg = paste0("G", 1:x$nReg),
  col = 1:x$nReg,
  add.rug = TRUE,
  lower.digits = 0,
  diag.digits = 0,
  lwd.dens = 1.5,
  diag.cex.text = 1,
```

```
  upper.digits = 2,
  upper.cex.text = 2,
  prefix = "",
  postfix = "",
  lty.diag = 1,
  ...
)
```

Arguments

x	an object of class "opsr.te".
pch	plotting 'character', i.e., symbol to use. See also pch.
labels.diag	labels used in the diagonal panels.
labels.reg	labels for the treatment regimes.
col	colour vector.
add.rug	if TRUE, adds rugs to the lower panels.
lower.digits	rounding of the digits in the lower panel.
diag.digits	rounding of the digits in the diagonal panel.
lwd.dens	linewidth of the densities in the diagonal panel.
diag.cex.text	cex for the text in the diagonal panel.
upper.digits	rounding of the digits in the upper panel.
upper.cex.text	cex for the text in the upper panel.
prefix	for the number plotted in the upper panel.
postfix	for the number plotted in the upper panel.
lty.diag	linetype for the diagonal panel.
...	further arguments passed to or from other methods.

Details

Presents all potential counterfactual outcomes. The diagonal depicts distributions in any given treatment regime and separate by the current (factual) treatment group. The weighted mean values are shown as red numbers. The lower triangular panels compare the model-implied (predicted) outcomes of two treatment regimes again separate by current treatment group. The red line indicates the 45-degree line of equal outcomes while the red squares depict again the weighted mean values. The upper triangular panels show (weighted) average treatment effects.

Value

Returns x invisibly.

See Also

pairs

Examples

```
sim_dat <- opsr_simulate()
dat <- sim_dat$data
weights <- runif(nrow(dat))
fit <- opsr(ys | yo ~ xs1 + xs2 | xo1 + xo2, dat = dat, weights = weights,
            printLevel = 0)
te <- opsr_te(fit, type = "response")
print(te)
summary(te)

te_w <- opsr_te(fit, type = "response", weights = rep(1, nrow(dat)))
summary(te_w)

pairs(te)
```

---

plot.opsr *Plot Method for OPSR Model Fits*

Description

Wrapper around pairs.opsr.te.

Usage

```
## S3 method for class 'opsr'
plot(
  x,
  type = c("response", "unlog-response", "prob", "mills", "correction", "Xb"),
  weights = NULL,
  ...
)
```

Arguments

x	an object of class "opsr".
type	see predict.opsr for details.
weights	a vector of weights. If NULL then weights from object will be used.
...	further arguments passed to pairs.opsr.te.

Value

Returns x invisibly.

See Also

opsr_te, pairs.opsr.te

predict.opsr *Predict Method for OPSR Model Fits*

Description

Obtains predictions for the selection process (probabilities), the outcome process, or returns the inverse mills ratio. Handles also log-transformed outcomes.

Usage

```
## S3 method for class 'opsr'
predict(
  object,
  newdata,
  group,
  counterfact = NULL,
  type = c("response", "unlog-response", "prob", "mills", "correction", "Xb"),
  delta = 1,
  ...
)
```

Arguments

object	an object of class "opsr".
newdata	an optional data frame in which to look for variables used in object$formula. See also model.matrix.opsr. Ensure that at least one selection outcome matches the 'group' argument (see 'Examples').
group	predict outcome of this group (regime).
counterfact	counterfactual group.
type	type of prediction. Can be abbreviated. See 'Details' section for more information.
delta	constant that was added during the continuity correction ($log(Y_j + \delta)$). Only applies for type = "unlog-response".
...	further arguments passed to or from other methods.

Details

Elements are NA_real_ if the group does not correspond to the observed regime (selection outcome). This ensures consistent output length.

If the type argument is "response" then the continuous outcome is predicted. Use "unlog-response" if the outcome response was log-transformed (i.e., either in the formula specification or during data pre-processing). "prob" returns the probability vector of belonging to group, "mills" returns the inverse mills ratio, "correction" the heckman correction (i.e., $\rho_j * \sigma_j *$ mills) and "Xb" returns $X\beta$.

Value

a vector of length `nrow(newdata)` (or data used during estimation).

See Also

`stats::predict`

Examples

```
set.seed(123)
sim_dat <- opsr_simulate()
dat <- sim_dat$data
model <- ys | yo ~ xs1 + xs2 | xo1 + xo2
fit <- opsr(model, dat)
p <- predict(fit, group = 1, type = "response")

fit_log <- update(fit, . | log(yo) ~ .)
p_unlog <- predict(fit, group = 1, type = "unlog-response")

## newdata
newdat <- dat[1:4, ]
unique(newdat$ys)  # available selection outcomes
predict(fit, newdata = newdat, group = 2)
```

---

print.anova.opsr *Print Method for ANOVA OPSR Objects*

Description

Print Method for ANOVA OPSR Objects

Usage

```
## S3 method for class 'anova.opsr'
print(
  x,
  digits = max(getOption("digits") - 2L, 3L),
  signif.stars = getOption("show.signif.stars"),
  print.formula = TRUE,
  ...
)
```

Arguments

`x`	an object of class `"anova.opsr"`.
`digits`	minimal number of *significant* digits, see `print.default`.

signif.stars	if TRUE, P-values are additionally encoded visually as 'significance stars' in order to help scanning of long coefficient tables. It defaults to the show.signif.stars slot of options.
print.formula	if TRUE, the formulas of the models are printed.
...	further arguments passed to stats::printCoefmat.

Value

Prints tables in a 'pretty' form and returns x invisibly.

See Also

stats::printCoefmat, anova.opsr

---

print.ate *Print Method for ATE Objects*

Description

Print Method for ATE Objects

Usage

```
## S3 method for class 'ate'
print(x, digits = max(3L, getOption("digits") - 3L), signif.legend = TRUE, ...)
```

Arguments

x	an object of class "ate".
digits	minimum number of significant digits to be used for most numbers (passed to stats::printCoefmat).
signif.legend	if TRUE, a legend for the 'significance stars' is printed.
...	further arguments passed to or from other methods.

Value

Prints x in 'pretty' form and returns reformatted treatment effects invisibly.

print.opsr.te *Print Method for OPSR ATE Objects*

Description

Print Method for OPSR ATE Objects

Usage

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

Arguments

x	an object of class "opsr.te".
...	further arguments passed to summary.opsr.te.

Details

This is just a wrapper around summary.opsr.te and a subsequent call to print.summary.opsr.te.

Value

Returns x invisibly.

See Also

print.summary.opsr.te

print.summary.opsr *Print Method for Summary OPSR Objects*

Description

Print Method for Summary OPSR Objects

Usage

```
## S3 method for class 'summary.opsr'
print(x, digits = max(3L, getOption("digits") - 3L), print.call = TRUE, ...)
```

Arguments

x	and object of class "summary.opsr"
digits	minimum number of significant digits to be used for most numbers (passed to stats::printCoefmat).
print.call	if TRUE, prints the underlying opsr call.
...	further arguments passed to or from other methods.

Value

Prints summary in 'pretty' form and returns x invisibly.

See Also

stats::printCoefmat, summary.opsr

---

print.summary.opsr.te *Print Method for Summary OPSR TE Objects*

Description

Print Method for Summary OPSR TE Objects

Usage

```
## S3 method for class 'summary.opsr.te'
print(x, digits = max(3L, getOption("digits") - 3L), print.call = FALSE, ...)
```

Arguments

x	an object of class "summary.opsr.te".
digits	minimum number of significant digits to be used for most numbers (passed to stats::printCoefmat).
print.call	if TRUE, prints the underlying call.
...	further arguments passed to or from other methods.

Value

Prints x in 'pretty' form and returns it invisibly.

print.te *Print Method for TE Objects*

Description

Print Method for TE Objects

Usage

```
## S3 method for class 'te'
print(x, digits = max(3L, getOption("digits") - 3L), signif.legend = TRUE, ...)
```

Arguments

x	an object of class "te".
digits	minimum number of significant digits to be used for most numbers (passed to stats::printCoefmat).
signif.legend	if TRUE, a legend for the 'significance stars' is printed.
...	further arguments passed to or from other methods.

Value

Prints x in 'pretty' form and returns reformatted treatment effects invisibly.

summary.opsr *Summarizing OPSR Model Fits*

Description

Follows the convention that opsr does the bare minimum model fitting and inference is performed in summary.

Usage

```
## S3 method for class 'opsr'
summary(object, rob = TRUE, ...)
```

Arguments

object	an object of class "opsr".
rob	if TRUE, the sandwich::sandwich covariance matrix extimator is used.
...	further arguments passed to or from other methods.

Value

An object of class `"summary.opsr"`. In particular the elements `GOF`, `GOFcomponents` and `wald` require further explanation:

`GOF`	Contains the conventional *goodness of fit* indicators for the full model. `LL2step` is the log-likelihood of the Heckman two-step solution (if the default starting values were used). `LLfinal` is the log-likelihood at final convergence and `AIC`, `BIC` the corresponding information critereon.
`GOFcomponents`	Contains the *goodness of fit* for the model components. `LLprobit` is the log-likelihood (LL) contribution of the ordered probit model. `LLprobitEl` the LL of the "equally likely" and `LLprobitMs` the LL of the "market share" model. With these three metrics the pseudo R2 is computed and returned as `pseudoR2el` and `pseudoR2ms`. `R2` reports the usual coefficient of determination (for the continuous outcomes jointly and for each regime separately).
`wald`	Contains the results of two *Wald-tests* as conducted with help of `car::linearHypothesis`. The two H0 hypothesis are 1. All coefficients of the explanatory variables are 0 and 2. The rho parameters (capturing error correlation) are zero.

---

summary.opsr.te *Summarizing OPSR TE Objects*

Description

This function computes weighted treatment effects and corresponding weighted paired t-tests.

Usage

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

Arguments

`object`	an object of class `"opsr.te"`.
`...`	further arguments passed to or from other methods.

Value

An object of class `"summary.opsr.te"` containing among others:

- `ate`: An object of class `"ate"`.
- `te`: An object of class `"te"`.

The p-values of the weighted paired t-test are attached as attributes.

`telework_data` *Telework Data*

Description

Telework data as used in Wang and Mokhtarian (2024).

Usage

```
telework_data
```

Format

Data frame with numeric columns

id Respondent ID

weight Sample weight

vmd Weekly vehicle-miles traveled

vmd_ln Log-transformed VMD, the dependent variable of the outcome model

twing_status Teleworking status: 1=Non-TWer, 2=Non-usual TWer, 3=Usual TWer

female Sex: female

age_mean Mean-centered age

age_mean_sq Sqaure of mean-centered age

race_white Race: white only

race_black Race: black only

race_other Race: other

edu_1 Education: high school or lower

edu_2 Education: some college

edu_3 Education: BA or higher

hhincome_1 Household income: less than $50,000

hhincome_2 Household income: $50,000 to $99,999

hhincome_3 Household income: $100,000 or more

flex_work Flexible work schedule

work_fulltime Full-time worker

twing_feasibility Teleworking feasibility (days/month)

vehicle Number of household vehicles

child Number of children

urban Residential location: urban

suburban Residential location: suburban

smalltown Residential location: small town

rural Residential location: rural

att_prolargehouse Attitude: pro-large-house

att_proactivemode Attitude: pro-active-mode

att_procarowning Attitude: pro-car-owning

att_wif Attitude: work-interferes-with-family

att_proteamwork Attitude: pro-teamwork

att_tw_effective_teamwork Attitude: TW effective teamwork

att_tw_enthusiasm Attitude: TW enthusiasm

att_tw_location_flex Attitude: TW location flexibility

region_waa Region indicator: respondents from WAA MSA

References

Wang X, Mokhtarian PL (2024). "Examining the Treatment Effect of Teleworking on Vehicle-Miles Driven: Applying an Ordered Probit Selection Model and Incorporating the Role of Travel Stress." *Transportation Research Part A*, **186**, 104072. doi:10.1016/j.tra.2024.104072.

Examples

```
## model as in Xinyi & Mokhtarian (2024)
f <-
  ## ordinal and continuous outcome
  twing_status | vmd_ln ~
  ## selection model
  edu_2 + edu_3 + hhincome_2 + hhincome_3 +
  flex_work + work_fulltime + twing_feasibility +
  att_proactivemode + att_procarowning +
  att_wif + att_proteamwork +
  att_tw_effective_teamwork + att_tw_enthusiasm + att_tw_location_flex |
  ## outcome model NTW
  female + age_mean + age_mean_sq +
  race_black + race_other +
  vehicle + suburban + smalltown + rural +
  work_fulltime +
  att_prolargehouse + att_procarowning +
  region_waa |
  ## outcome model NUTW
  edu_2 + edu_3 + suburban + smalltown + rural +
  work_fulltime +
  att_prolargehouse + att_proactivemode + att_procarowning |
  ## outcome model UTW
  female + hhincome_2 + hhincome_3 +
  child + suburban + smalltown + rural +
  att_procarowning +
  region_waa

fit <- opsr(f, telework_data)
texreg::screenreg(fit, beside = TRUE, include.pseudoR2 = TRUE, include.R2 = TRUE)
```

`timeuse_data` *TimeUse+ Data*

Description

TimeUse+ (Winkler et al. 2024) was a tracking study conducted at the Institute for Transport Planning and Systems (IVT) at ETH Zurich. The data allows researchers to investigate time-use and mobility patterns. Full data access can be requested via doi:10.3929/ethz-b-000634868.

Usage

```
timeuse_data
```

Format

Data frame with numeric and factor columns

id Respondent ID

start_tracking Indicator for the months when the participant started tracking

weekly_km Weekly distance covered (in km) across all modes

log_weekly_km Log of `weekly_km`

wfh_days Based on tracked work episodes. Average full working days spent in home office during a typical week.

wfh Derived from `wfh_days`: NTW=Non-TWer, NUTW=Non-usual TWer (< 3 days/week), UTW=Usual TWer(3+ days/week)

commute_km Map matched commute distance in km

log_commute_km Log of `commute_km`

age Age

car_access Has access to a household car

dogs Household owns dogs

driverlicense Owns driver's license for cars

educ_higher Higher education (e.g., university)

fixed_workplace Main work location does not change regularly

grocery_shopper Does usually do the grocery shopping

hh_income Gross household income per month (CHF)

hh_size Total household size

isco_clerical International standard classification of Occupations (ISCO-08): Clerical support workers

isco_craft International standard classification of Occupations (ISCO-08): Craft related trade workers

isco_elementary International standard classification of Occupations (ISCO-08): Elementary occupations

isco_managers International standard classification of Occupations (ISCO-08): Managers

isco_plant International standard classification of Occupations (ISCO-08): Plant and machine operators, and assemblers

isco_professionals International standard classification of Occupations (ISCO-08): Professionals

isco_service International standard classification of Occupations (ISCO-08): Service and sales workers

isco_agri International standard classification of Occupations (ISCO-08): Skilled agricultural, forestry and fishery workers

isco_tech International standard classification of Occupations (ISCO-08): Technicians and associate professionals

married Married

n_children Number of household members below the age of 18

freq_onl_order Orders products online more than once a month

parking_home Has at least one reserved parking space at home

parking_work Has at least one reserved parking space at work location

permanent_employment Employment contract type: permenent (unlimited employed)

rents_home Residential situation: Rents home

res_loc Residential location

sex_male Gender

shift_work Whether participant works in shifts

swiss Swiss citizen

vacation Participant took time off of work during study

workload Workload (% of full-time employment which is 41.7 h/week)

young_kids Whether children aged 12 or younger live in the household

Details

The data comprises employed individuals only and is based on valid days only. A valid day has at least 20h of information where 70% of the events were validated by the user. The telework variables are based on tracked work activities.

References

Winkler C, Meister A, Axhausen KW (2024). "The TimeUse+ Data Set: 4 Weeks of Time Use and Expenditure Data Based on GPS Tracks." *Transportation*, 1–27. doi:10.1007/s11116024105171.

Index

∗ **datasets**
    telework_data, 27
    timeuse_data, 29

anova.opsr, 3, *22*
as.data.frame, *8*

car::linearHypothesis, *26*

data.frame, *6*

extract,opsr-method, 4
extract.opsr *(*extract,opsr-method*)*, 4

loglik_cpp, 5, *11*
loglik_R, *6*, *8*, *11*

maxLik::maxLik, *8–11*
model.frame, *8*
model.frame.opsr, 6, *7*
model.matrix.opsr, 7, *20*

na.exclude, *8*
na.fail, *8*
na.omit, *8*

OPSR *(*OPSR-package*)*, 3
opsr, *6*, 7, *10–12*, *14*, *16*, *24, 25*
OPSR-package, 3
opsr.fit, *5–7*, 10, *11, 12*
opsr_2step, *8*, 11, *13*
opsr_check_omp, *8*, *10*, 12, *13*
opsr_check_start, *8*, 12
opsr_max_threads, *8*, *10*, 13
opsr_null_model, *3*, 13
opsr_prepare_coefs, *12*, 14
opsr_simulate, 15
opsr_step, 15
opsr_te, 16, *19*
options, *8*, *22*

pairs, *18*
pairs.opsr.te, 17, *19*
pch, *18*
plot.opsr, 19
predict.opsr, *7*, *16*, *19*, 20
print.anova.opsr, *4*, 21
print.ate, 22
print.default, *21*
print.opsr.te, 23
print.summary.opsr, 23
print.summary.opsr.te, *23*, 24
print.te, 25

sandwich::sandwich, *25*
set.seed, *15*
stats::anova, *4*
stats::lm, *9*
stats::model.frame, *6*
stats::model.matrix, *7*
stats::predict, *21*
stats::printCoefmat, *22*, *24, 25*
summary.opsr, *5*, *24*, 25
summary.opsr.te, *17*, *23*, 26

telework_data, 27
texreg::screenreg, *5*
texreg::texreg, *5*
timeuse_data, 29

update, *16*