

Package: OPSR (via r-universe)

November 22, 2024

Title Ordinal Probit Switching Regression

Version 0.1.2.9001

Description Estimates ordinal 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](https://doi.org/10.1016/j.tra.2024.104072)> or Chiburis & Lokshin (2007) <[doi:10.1177/1536867X0700700202](https://doi.org/10.1177/1536867X0700700202)>.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 2.10)

LazyData true

RdMacros Rdpack

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

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

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

Config/pak/sysreqs cmake make texlive libicu-dev libssl-dev

Repository <https://dheimgartner.r-universe.dev>

RemoteUrl <https://github.com/dheimgartner/opsr>

RemoteRef HEAD

RemoteSha 53f787540e329eab22a79ad22683d8eccc27f751

Contents

OPSR-package	2
anova.opsr	3
extract,opsr-method	4
loglik_cpp	5
model.frame.opsr	6
model.matrix.opsr	6
opsr	7
opsr.fit	9
opsr_2step	10
opsr_check_omp	11
opsr_check_start	12
opsr_max_threads	12
opsr_null_model	13
opsr_prepare_coefs	14
opsr_simulate	14
predict.opsr	15
print.anova.opsr	16
print.summary.opsr	17
summary.opsr	18
telework_data	19

Index	22
--------------	-----------

OPSR-package

OPSR: Ordinal Probit Switching Regression

Description

Estimates ordinal 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](https://doi.org/10.1016/j.tra.2024.104072) or Chiburis & Lokshin (2007) [doi:10.1177/1536867X0700700202](https://doi.org/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,
  ...
)
```

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.
...	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 explanatory variables 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	<i>Extracting the Model Frame from OPSR Model Fits</i>
------------------	--

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	<i>Construct Design Matrices for OPSR Model Fits</i>
-------------------	--

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\$ n Reg elements (a separate design matrix for each regime).

See Also

[model.frame.opsr](#), [stats::model.matrix](#)

opsr

Fitting Ordinal 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,
  ...
)
```

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.).

<code>weights</code>	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.
<code>na.action</code>	a function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of <code>options</code> , and is <code>na.fail</code> if that is unset. The 'factory-fresh' default is <code>na.omit</code> . Another possible value is NULL, no action. Value <code>na.exclude</code> can be useful.
<code>start</code>	a numeric vector with the starting values (passed to <code>maxLik::maxLik</code>). If no starting values are provided, reasonable values are auto-generated via the Heckman 2-step procedure <code>opsr_2step</code> . The structure of <code>start</code> has to conform with <code>opsr</code> 's expectations. See <code>opsr_check_start</code> for further details.
<code>fixed</code>	parameters to be treated as constants at their <code>start</code> values. If present, it is treated as an index vector of <code>start</code> parameters (passed to <code>maxLik::maxLik</code>).
<code>method</code>	maximization method (passed to <code>maxLik::maxLik</code>).
<code>iterlim</code>	maximum number of iterations (passed to <code>maxLik::maxLik</code>).
<code>printLevel</code>	larger number prints more working information (passed to <code>maxLik::maxLik</code>).
<code>nThreads</code>	number of threads to be used. Do not pass higher number than number of ordinal outcomes. See also <code>opsr_check_omp</code> and <code>opsr_max_threads</code> .
<code>.get2step</code>	if TRUE, returns starting values as generated by <code>opsr_2step</code> . Will not proceed with the maximum likelihood estimation.
<code>.useR</code>	if TRUE use <code>loglik_R</code> . Go grab a coffe.
<code>.censorRho</code>	if TRUE, rho starting values are censored to lie in the interval [-0.85, 0.85].
<code>...</code>	further arguments passed to <code>maxLik::maxLik</code> .

Details

Models for `opsr` are specified symbolically. A typical model has the form $ys \mid yo \sim \text{terms}_s \mid \text{terms}_{o1} \mid \text{terms}_{o2} \mid \dots$. 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 ($ys \mid yo \sim \text{terms}_s \mid \text{terms}_o$). 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 Ordinal 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,
    ...
)

```

Arguments

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

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. Two-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 explanatory variables 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 explanatory variables for outcome process for each regime.

Details

Expected order: 1. kappa threshold parameters (for ordinal 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	<i>Null Model for OPSR Model fits</i>
-----------------	---------------------------------------

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 ordinal probit process and separate (for each regime) OLS process where the errors follow a multivariate normal distribution.

Usage

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

Arguments

nobs number of observations to simulate.
sigma the covariance matrix of the multivariate normal.

Details

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

Value

Named list:

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

`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"),
  ...
)
```

Arguments

<code>object</code>	an object of class "opsr".
<code>newdata</code>	an optional data frame in which to look for variables used in <code>object\$formula</code> . See also model.matrix.opsr .
<code>group</code>	predict outcome of this group (regime).
<code>counterfact</code>	counterfactual group.
<code>type</code>	type of prediction. Can be abbreviated. See 'Details' section for more information.
<code>...</code>	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 during estimation. "prob" returns the probability vector of belonging to group and "mills" returns the inverse mills ratio.

Value

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

See Also

[stats::predict](#)

Examples

```
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")
```

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 <i>significant</i> 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.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](#)

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

<code>object</code>	an object of class "opsr".
<code>rob</code>	if TRUE, the <code>sandwich::sandwich</code> covariance matrix estimator is used.
<code>...</code>	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:

<code>GOF</code>	Contains the conventional <i>goodness of fit</i> indicators for the full model. <code>LL2step</code> is the log-likelihood of the Heckman two-step solution (if the default starting values were used). <code>LLfinal</code> is the log-likelihood at final convergence and <code>AIC</code> , <code>BIC</code> the corresponding information critereon.
<code>GOFcomponents</code>	Contains the <i>goodness of fit</i> for the model components. <code>LLprobit</code> is the log-likelihood (LL) contribution of the ordinal probit model. <code>LLprobitE1</code> the LL of the "equally likely" and <code>LLprobitMs</code> the LL of the "market share" model. With these three metrics the pseudo R2 is computed and returned as <code>pseudoR2e1</code> and <code>pseudoR2ms</code> . <code>R2</code> reports the usual coefficient of determination (for the continuous outcomes jointly and for each regime separately).
<code>wald</code>	Contains the results of two <i>Wald-tests</i> as conducted with help of <code>car::linearHypothesis</code> . The two H0 hypothesis are 1. All coefficients of the explanatory variables are 0 and 2. The rho parameters (capturing error correlation) are zero.

telework_data	<i>Telework data</i>
---------------	----------------------

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 Square 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)
```


Index

* datasets

- telework_data, 19
- anova.oprsr, 3, 17
- as.data.frame, 7
- car::linearHypothesis, 18
- data.frame, 6
- extract,oprsr-method, 4
- extract.oprsr (extract,oprsr-method), 4
- loglik_cpp, 5, 10
- loglik_R, 5, 8, 10
- maxLik::maxLik, 8, 10
- model.frame, 7
- model.frame.oprsr, 6, 6, 7
- model.matrix.oprsr, 6, 15
- na.exclude, 8
- na.fail, 8
- na.omit, 8
- OPSR (OPSR-package), 2
- oprsr, 5, 7, 9–14, 17, 18
- OPSR-package, 2
- oprsr.fit, 5, 7, 9, 11
- oprsr_2step, 8, 10, 12
- oprsr_check_omp, 8, 10, 11, 13
- oprsr_check_start, 8, 12
- oprsr_max_threads, 8, 10, 12
- oprsr_null_model, 3, 13
- oprsr_prepare_coefs, 11, 14
- oprsr_simulate, 14
- options, 8, 17
- predict.oprsr, 6, 15
- print.anova.oprsr, 3, 16
- print.default, 17
- print.summary.oprsr, 17
- sandwich::sandwich, 18
- stats::anova, 3
- stats::lm, 8
- stats::model.frame, 6
- stats::model.matrix, 7
- stats::predict, 16
- stats::printCoefmat, 17
- summary.oprsr, 4, 17, 18
- telework_data, 19
- texreg::screenreg, 5
- texreg::texreg, 5