Package 'semiIVreg'

December 28, 2024

Title Semi-Instrumental Variable (semi-IV) Regression

Version 1.0.0

Description semi-Instrumental Variable (semi-IV) estimation for general models, from Bruneel-Zupanc (2024). The main semiivreg() function is designed to work as easily as lm() or ivreg(). It automatically provides estimation of the marginal treatment effects (MTE) as well as eventual ATE (if the treatment effects were, in fact, homogenous).

Encoding UTF-8

Suggests knitr,
rmarkdown,
roxygen2,
ivreg

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports data.table, ggplot2, gridExtra, MASS, stats, KernSmooth, sandwich, lmtest, nprobust

License MIT + file LICENSE

Depends R (>= 2.10)

LazyData true

URL https://www.cbruneel.com/, https://cbruneelzupanc.github.io/semiIVreg/

VignetteBuilder knitr

Contents

construct_data *Data construction functions*

Description

These functions are used to construct the data from a given formula. Handles change from factor into several dummies for example. They also create reference individuals at which to evaluate the MTE and MTR (if no default is provided). For the numerical variables, take the average of the variable; for the factors, take the first level.

Usage

```
construct_data(formula, data)

transform_factor(formula, data)
```

Arguments

formula	The formula of the model
data	The original dataset

Kappa_fun *Control functions transformations for selection probabilities*

Description

These functions provides pre-specified transformations to control flexibly for the selection probabilities in the regression.
These correspond to $\kappa_d(p)$ and corresponding $k_d(u)$ in Bruneel-Zupanc (2024).
Special functions are used for homogenous treatment effect specifications because the code is different. July 2024: for now, only polynomial transformations are encoded.

Usage

```
Kappa_fun(p, pol_degree = 5)

kdu_transform_fun(u, d, pol_degree = 5)

Kappa_homogenous_fun(p, pol_degree = 5)

ku_transform_homogenous_fun(u, pol_degree = 5)
```

Arguments

p	Vector of propensity scores to transform into a flexible function.
pol_degree	Degree of the polynomial transformation.
d	Which potential outcome to consider (only needed for $k_d(u)$ with heterogenous treatment effects).

Details

See Andresen (2018) or Bruneel-Zupanc (2024) for computation details linking $\kappa_d(p)$ and the corresponding corresponding $k_d(u)$.
$\kappa_1(p) = E(U_1|U_D \leq p)$ and $\kappa_0(p) = E(U_0|U_D > p)$ while $k_d(u) = E(U_d|U_D = u)$.

In the case of homogenous treatment effects: $k_0(u) = k_1(u)$. This provides some restriction on κ, hence the special functions.

Examples

```
v = seq(0.1, 0.9, by=0.1)
# Transformations for general Heterogenous TE functions:
Kappa_fun(p=v, pol_degree=6)
k1u = kdu_transform_fun(v, d=1, pol_degree=6)

# Transformations for Homogenous TE functions:
Kappa_homogenous_fun(p=v, pol_degree=6)
ku = ku_transform_homogenous_fun(v, pol_degree=6) # no d anymore, same for both d here;
```

`mtr_est_poly` *MTE and MTR sub-estimation functions*

Description

These functions allow to estimate the mte and mtr, and their confidence intervals, based on coefficients estimated from the main model in the main function. More details can be found in Bruneel-Zupanc (2024).
Different formulas must be applied depending on whether the treatment is homogenous or heterogenous.

Bandwidth selection for local polynomial regression. Different methods are available: "mse-dpi", "mse-rot" (from nprobust) or "arbitrary" (fixed bandwidth to a fraction of the support). Can provide bandwidth for the main function or for any of its derivative order. The bandwidth can be computed on a subsample of the data of size `bw_subsamp_size` to speed up the computation.

The `lpoly` function estimates a (weighted) local polynomial regression of a specified degree at given evaluation points. It supports derivative estimation and allows for heteroscedasticity-consistent standard errors. External weights (weights) are allowed.

Usage

```
mtr_est_poly(data, seq_u,
             bw0 = NULL, bw1=NULL, bw_y0 = NULL, bw_y1=NULL, bw_method = 1/5, kernel,
                    bw_subsamp_size = NULL, fast_locpoly = FALSE,
                    fast_robinson1 = TRUE, fast_robinson2 = FALSE,
             pol_degree1, pol_degree2, var_outcome, var_treatment, var_w0, var_w1, var_covariates,

mtr_fun_poly(ref_indiv, eval_v, est0, est1, kv, se_type, conf_level)

lbw_select(x, y, kernel, degree, drv, bw_method = "arbitrary", bw_subsamp_size)

wlocpol(x, y, bandwidth, degree = 2, drv = 1, kernel = "gaussian", weights=NULL, x_eval=NULL, gridsiz
```

```
mtr_coeff(coeff, vcov, var_cov_2nd, est_method="sieve")

mtr_est(coeff, vcov, names_var, df)

mtr_predict_sieve(coeff, vcov, ref_indiv, var_treatment, var_cov_2nd, pol_degree, seq_u, t_value, e
```

Arguments

seq_u	Sequence of v at which to compute the prediction for kd(v).
bw_method	Method to compute the bandwidth (if bandwdith is NULL)
bw_subsamp_size	Size of the subsample to compute the bandwidth. Default is NULL (no subsample). If larger than sample size, it is ignored.
se_type	"HC1", "nonrobust" (for baseline homoscedastic), ...
ref_indiv	Newdata (reference individuals) at which to compute the predictions.
x	Vector of x values
y	Vector of y values
degree	Degree of the polynomial: recommended to set to drv + 1
drv	Derivative order of the function to be estimated.
supp	Support of X in the complete data (not necessarily of the realized X in the current subsample), in order to compute the bandwidth if the rule is to take a fraction of the support. Ensure that same bandwidth on both samples D=0 and D=1. By default supp=NULL, in which case recompute the support of x directly.
bandwidth	Pre-specified bandwidth
weights	Vector of external weights (in addition to the kernel weights)
x_eval	Vector of evaluation points. If not pre-specified, use a grid (of gridsize). Default = NULL.
gridsize	Size of the grid of evenly spaced points for x. Default is 201. Only relevant if x_eval = NULL.
fast_locpoly	Default is FALSE. If fast_locpoly is TRUE, will use the locpoly function from Kernsmooth library to speed up the computation. This is only possible if no external weights are used. If the kernel is not set to Gaussian, the locpoly function will change it to Gaussian if fast_poly is TRUE.
coeff	Vector of (stacked) coefficients for mtr0, mtr1.
vcov	Covariance matrix of these coefficients.
est_method	Either "sieve" or "homogenous".
names_var	Names of the variables corresponding to the coefficients.
df	Degrees of freedom for the p-values.
var_cov_2nd	Names of the covariates and semi-IVs
‘fast_robinson1‘	Default is TRUE to speed things up in a first stage (if many covariates in particular). If TRUE, will use the locpoly function from Kernsmooth library to speed up the computation of the Robinson double residual first stage. This is only possible if no external weights are used. Fast Locpoly will enforce a gaussian kernel.

`'fast_robinson2'`
: Default is FALSE. If TRUE, will use the locpoly function from Kernsmooth library to speed up the computation of the Robinson double residual second stage. This is only possible if no external weights are used. Fast Locpoly will enforce a gaussian kernel. Default is FALSE for the second stage because fast_locpoly returns no standard errors and the gain in time is not so important for the second stage.

`bw`
: Pre-specified bandwidth

Value

Returns the raw (stacked regression) coefficients and covariance matrix corresponding to mtr0, mtr1 and mte function.

Returns the mtr0, mtr1 and mte estimates tables with their standard errors and p-values. Also exports the corresponding estimates and vcov matrices.

Returns mtr0, mtr1, and mte estimates with confidence intervals for the specified ref_indiv.

`mtr_plot_fun` *Marginal Treatment Effect (MTE) and Responses (MTR) plots*

Description

Plot the MTR and MTE estimated curves with their confidence intervals.

Usage

```
mtr_plot_fun(dat_plot, common_supp)

mte_plot_fun(dat_plot, common_supp)
```

Arguments

`dat_plot`
: Data frame with the estimated MTE and MTR values and their confidence intervals. Must contain specific variables: `Phat`, `mtr0`, `mtr1`, `mtr0_lwr`, `mtr1_lwr`, `mtr0_upr`, `mtr1_upr`, `mte`, `mte_lwr`, `mte_upr`.

`common_supp`
: Vector of two values indicating the common support of the plot. Default is the full support 0,1.

`conf_band`
: Indicates whether to plot the confidence intervals. Default is "TRUE".

`colMTE`, `colD0`, `colD1`
: Color of the MTE, MTR0 and MTR1 curves.

Details

Attention: by default in `semiivreg` the confidence intervals are computed analytically, and include an error because the first stage propensity score. This is corrected in `semiivreg_boot` by bootstrapping the entire estimation to obtain the confidence intervals.

roydata *Generalized Roy Data with Heterogenous Treatment Effects*

Description

A data frame of 100,000 observations drawn from a simulated Roy model with heterogenous treatment effects using `simul_data()`.

Usage

```
data(roydata)
```

Format

The data contains the following information which would be observed in a standard dataset:

y The observed outcome.

d The treatment.

w0, w1 The semi-IVs entering only D=0 and D=1.

Xbinary, Xcontinuous Two covariates, one binary and one continuous.

It also reports the typically unobserved potential outcomes and shocks:

y0, y1 The unobserved potential outcomes.

P The unobserved true treatment probability.

latent, V, Ud, U0, U1 The unobserved shocks V. Ud is the normalized V ranks. U0 and U1 are the outcome shocks. `latent` gives the latent utility term in the selection equation.

The data was generated using the following R code:

```
N=100000; set.seed(1234)
model_type = "heterogenous"
param_error = c(1, 1, 0.6, 0.5)
param_Z = c(0, 0, 0, 0, 1.5, 1.5, 0.9)
param_p = c(0, -0.7, 0.7, 0, 0, 0)
param_y0 = c(3.2, 0.8, 0, 0)
param_y1 = c(3.2+0.4, 0.5, 0, 0)
param_genX = c(0.4, 0, 2)

roydata = simul_data(N, model_type, param_y0, param_y1,
                     param_p, param_Z, param_genX, param_error)
```

roydata2 *Generalized Roy Data with Homogenous Treatment Effects*

Description

A data frame of 100,000 observations drawn from a simulated Roy model with homogenous treatment effects using `simul_data()`.

Usage

```
data(roydata2)
```

Format

The data contains the following information which would be observed in a standard dataset:

y The observed outcome.

d The treatment.

w0, w1 The semi-IVs entering only D=0 and D=1.

Xbinary, Xcontinuous Two covariates, one binary and one continuous.

It also reports the typically unobserved potential outcomes and shocks:

y0, y1 The unobserved potential outcomes.

P The unobserved true treatment probability.

latent, V, Ud, U0, U1 The unobserved shocks V. Ud is the normalized V ranks. U0 and U1 are the outcome shocks. `latent` gives the latent utility term in the selection equation.

The data was generated using the following R code:

```
N = 100000; set.seed(1234)
model_type = "homogenous"
param_error = c(1, 1.5, -0.6)
param_Z = c(0, 0, 0, 0, 1.5, 1.5, 0.9)
param_p = c(0, -0.5, 0.5, 0, 0, 0)
param_y0 = c(3.2, 0.8, 0, 0)
param_y1 = c(3.2+0.4, 0.5, 0, 0)
param_genX = c(0.4, 0, 2)

roydata2 = simul_data(N, model_type, param_y0, param_y1,
                      param_p, param_Z, param_genX, param_error)
```

semiivreg *Semi-IV Regression Function*

Description

Semi-IV regression function from Bruneel-Zupanc (2024). Syntax inspired from ivreg. Returns MTE and MTR curves with confidence intervals. The estimation is almost *instantaneous* (a few seconds at most).

By default, return analytic standard errors not accounting for the fact that the propensity score is estimated in a first stage in `semiivreg`. Use `semiivreg_boot` to obtain 'correct' bootstrapped confidence intervals (takes a bit longer).

Usage

```
semiivreg(formula, data, propensity_formula=NULL, propensity_data = NULL,
                  ref_indiv =NULL, firststage_model = "probit",
                  est_method = "locpoly", # "locpoly", "sieve", or "homogenous".
                  se_type = "HC1",
               bw0 = NULL, bw1 = NULL, bw_y0 = NULL, bw_y1 = NULL, bw_method = 1/5,
                  kernel="gaussian",
                  pol_degree_locpoly1 = 1, pol_degree_locpoly2 = 2,
                  pol_degree_sieve = 5, conf_level = 0.05,
              common_supp_trim=c(0,1), trimming_value=NULL, automatic_trim=FALSE,
                  plotting=TRUE, print_progress=FALSE)

semiivreg_boot(formula, Nboot=500, data, propensity_formula=NULL, ref_indiv =NULL,
                firststage_model="probit", est_method = "locpoly", se_type="HC1",
          bw0 = NULL, bw1 = NULL, bw_y0 = NULL, bw_y1 = NULL, bw_method = "rule-of-thumb",
               pol_degree_locpoly1 = 1, pol_degree_locpoly2 = 2,
               common_supp_trim=c(0,1), trimming_value = NULL,
          automatic_trim = FALSE, plotting=TRUE, conf_level = 0.05, CI_method = "curve", weight_var

semiiv_predict(semiiv, newdata, seq_v=NULL)
```

Arguments

`formula`	Formula of the regression, of the form outcome ~ treatment | semi-iv0 | semi-iv1 | commoncovariates. The treatment variable should be binary (0, 1). covariates with an effect that differs on D=1 and D=0 should be included in each semi-iv0 and semi-iv1. with est_method = "locpoly": cannot restrict covariates to have common effects (not implemented), so `commoncovariates` will just be estimated as having generally a different effect on Y0 and Y1.
`data`	Dataframe containing the data.
`propensity_formula`	Formula for the 1st stage. If nothing specified, just runs a probit of d ~ semi-iv0 + semi-iv1 + covariates (removing the redundant variables).

`propensity_data`
: Data used to compute the 1st stage; ignore by default set to NULL and = data. Mainly useful for internal bootstrap function is the first stage formula is different from the default one.

`ref_indiv`
: Specify the reference individual (in terms of covariates) at which we will evaluate the function.
By default takes the average value for all the covariates (on the trimmed dataset) to compute the average estimate. Remark: for factors, the average is computed on the dummy variables to get the proper average effect.

`firststage_model`
: By default, the first stage is a probit model. Can specify another model (e.g., "logit").

`est_method`
: Estimation method: default is "locpoly" for Robinson (1988) double residual regression for partially linear model. Other options include "sieve" to specify flexibly the control function as a polynomial with pol_degree_sieve, and "homogenous" which is a sieve where we also impose homogenous treatment effect.

`bw0`, `bw1`
: Bandwidth of the first residual regressions of (Y, Wd and X) on Phat.
Two possibilities: specify one value that is applied to all covariates (and Y), or specify a different bandwidth for the regression on each covariate. In the second case, need to be specified in the order of the covariates as specified in the model. Be very careful with factors.
Default NULL and computed using the specified bw_method. Ideally, if one factor covariate, apply the same bandwidth to all of the dummies created from the factor.

`bw_y0`, `bw_y1`
: Bandwidth of the second regression of Y (net of the effects of the covariates) on Phat. Default NULL and computed using the specified bw_method.

`bw_method`
: Method to compute the bandwidth of the local polynomial regressions (of the first-order derivative). Default option is 1/5, which arbitrarily sets bw0, bw1, bw_y0 and bw_y1 to 1/5th of the support (rounded to the 3th digit). Can place any fraction < 1.
Recommended alternatives include (global constant) bandwidth computations from nprobust package (Calonico, Cattaneo and Farrell, 2019) (i) "mse-dpi": direct plug-in MSE optimal bandwidth from Fan and Gijbels (1996). (ii) "mse-rot": rule-of-thumb implementation of the MSE-optimal bandwidth. These two methods take long with large sample: use bw_subsamp_size to speed up the computation.

`kernel`
: Kernel to use for the local polynomial regressions. Default is "gaussian" but can be "epanechnikov". Takes longer with Epanechnikov (cannot use fast locpoly implementation from KernSmooth).

`bw_subsamp_size`
: Size of the subsample to use for the bandwidth selection. Default is 10,000. Use `bw_subsamp_size = NULL` to use the full sample (may take time). Otherwise, recommend to set a number around 20,000 at most for reasonable computation time (exponentially increasing with sample size).
`bw_subsamp_size` introduces some randomness into the bandwidth selection procedure: recommended to set a seed before running semiivreg for reproducibility.

`pol_degree_locpoly1`
: Degree of the local polynomial regression of the covariates on Phat. Default is

1 as recommended by Fan and Gijbels (1996) because we want to estimate the regular function.

`pol_degree_locpoly2`
: Degree of the local polynomial regression of Y (net of the effects of the covariates) on Phat. Default is 2 as recommended by Fan and Gijbels (1996) because we want to estimate the derivative function.

`fast_robinson2`
: If TRUE, use locpoly from KernSmooth during the bootstrap. Default is TRUE to speed things up (because do not need to compute standard errors in bootstrap) Set to FALSE if want to use epanechnikov kernel, or if want to use weights.

`pol_degree_sieve`
: Degree of the polynomial transformation for the control function.

`se_type`
: Type of standard errors in main estimation and in each bootstrap replication. Can simplify by setting "nonrobust" which goes (slightly) faster.

`conf_level`
: Confidence level for the confidence intervals.

`common_supp_trim`
: Vector of two values indicating the set of propensity scores at which we will evaluate the function.
Default is the full support `[0,1]`. But can be trimmed manually.

`trimming_value`
: Can either be a vector c(0.05, 0.95) indicating the quantile of the propensity score above which and below which we keep the observations for both D=0 and D=1.
Can also be a single value, in which case symmetric trimming up and down.
Inserting a trimming_value generates automatic_trim = TRUE automatically.

`automatic_trim`
: If TRUE, the estimation of the second stage is done on the common_support only.

`weight_var`
: A variable of weights to be applied to the observations. Default is NULL, apply equal weights to all observations.
Implemented completely for `est_method = "sieve"` for now. For `locpoly`, the weights are not used when computing the "optimal bandwidth".

`plotting`
: TRUE if wants to plot at the end of the function, FALSE otherwise.

`print_progress`
: TRUE if wants to print the progress of the function, FALSE otherwise (default=FALSE).

`Nboot`
: Number of bootstrap samples.

`block_boot_var`
: Variable on which to base the block bootstrap. By default, = NULL for standard bootstrap.

`CI_method`
: "delta" for delta method, "curve" for bootstrap the MTE curves directly. With est_method = "locpoly", only "curve" method is possible.

`print_progress_main`
: Print progress of the main estimation or not.

`semiiv`
: Object returns from a semiivreg estimation.

`newdata`
: New data for which to predict the MTE and MTR.

`seq_v`
: Sequence of v at which to predict the MTE and MTR. By default: NULL fits the default interval of the original semiivreg (equally space grid of proba, with step size of 0.001 on the common support).

`‘fast_robinson1‘`
: Default is TRUE to speed things up in a first stage (if many covariates in particular). If TRUE, will use the locpoly function from Kernsmooth library to speed

up the computation of the Robinson double residual first stage. This is only possible if no external weights are used. Fast Locpoly will enforce a gaussian kernel.

`'fast_robinson2'`
: Default is FALSE. If TRUE, will use the locpoly function from Kernsmooth library to speed up the computation of the Robinson double residual second stage. This is only possible if no external weights are used. Fast Locpoly will enforce a gaussian kernel. Default is FALSE for the second stage because fast_locpoly returns no standard errors and the gain in time is not so important for the second stage.

Value

A list with the following elements:

`$data` Returns data of output estimation used to plot the MTE and MTR. In details:

- `$RES` Dataframe with the estimated MTE and MTR values (and their confidence intervals if `est_method="sieve"` or `"homogenous"`) for a sequence of unobservable resistance to treatment in the identifiable common support.
- `$data` Original data used for the estimation where we added the propensity score estimated, named `Phat`, and where we made the transformation of the eventual `factor` variables as dummies.
- `$ref_indiv` Reference individual(s) at which we evaluate the MTE and MTR.
- `$Xdat` Set of covariates (this output is used for the bootstrap).
- `$deltaX` Returns the estimated effects of the covariates and semi-IVs (without intercept) for the specified reference individuals.

`$estimate` Returns the estimation of:

- `$est, or $est0 and $est1` If `est_method = "locpoly"`, `est0` and `est1` returns the second stage estimates of the effect of the covariates and semi-IVs on their respective potential outcomes. Coming out of the double residual regression à la Robinson (1988), running a no-intercept OLS of the residuals Y-E(Yd|P) on the residuals of every semi-IVs, Wd-E(Wd|P), and covariates, X-E(X|P).

- `$mtr0, $mtr1 and $mte` If `est_method = "sieve"` or `"homogenous"`, returns the functional form estimated for both MTR and MTE.
- `$kv` Returns the estimated k_d(v) (=E(Ud|V=v)). Includes the constant. If sums with the effect of covariates and semi-IVs (deltadX), gives the mtr_d.
- `$propensity` First stage estimate of the propensity score.
- `$est_kappa` If `est_method = "sieve"` or `"homogenous"`, this returns the estimated model for E(Y|D=d, X, Wd, P). From this, we extract Kappad(P) = E(Ud | D=d, P=p) from which we compute the kd(v) and mtrd(v, x, wd) functions.
- `$avg_MTE` Average of the MTE over the identified common support. If full common support, it is an estimate of the ATE(x, w0, w1). If `est_method="homogenous"`, the MTE is constant so it also gives the ATE(x, w0, w1).

`$bw` Returns the bandwidth used (or estimated via `bw_method`) in the Robinson double residual regression.
`bw0` and `bw1` are the bandwidths of the first residual regressions of Yd, Wd and X on Phat.
`bw_y0` and `bw_y1` are the bandwidths of the second regression of Y (net of the effects of the covariates) on Phat. These are the one that matters for the smoothness of the MTE and MTR estimates.

`$plot` Returns separately the following plot objects: `supp` (support), `mtr`, `mte` and `mte2`. `mte` reports the estimation from "local IV" approach, with standard errors from the Robinson 2nd stage. `mte2` reports the MTE estimated as the difference between the MTR (without standard errors).

`$supp` Returns the common support of the propensity score `Phat` between the two treatment group.

`$call` Returns the call of the function and the covariates and semi-IVs used.

The estimated model

`semiivreg` estimates the marginal treatment effect (MTE) and marginal treatment response (MTR) of a binary treatment variable using semi-IVs, W0 and W1. As with standard IVs (see Andresen, 2018), we estimate a semi-parametric partially linear model, as described in Bruneel-Zupanc (2024). For more details on the model and estimation procedure, see the vignette `vignette("semiIVreg", package = "semiIVreg")`, also available online here. For more details on the use of the `semiivreg` function, see also the vignettes `vignette("semiIVreg_heterogenousTE", package = "semiIVreg")` and `vignette("semiIVreg_homogenousTE", package = "semiIVreg")`. For more details about causal inference with semi-IVs in general, see Bruneel-Zupanc (2024).

Caution about the Estimated Standard errors

By default, `est_method="locpoly"` returns no standard errors.
If `est_method="sieve"` or `est_method="homogenous"`, it returns **analytic standard errors**: but these are wrong because they do not account for the fact that the propensity score is estimated.
In any case, we recommend to use `semiivreg_boot` to obtain 'correct' bootstrapped confidence intervals. Implemented separately because the bootstrap takes more time, while the baseline `semiivreg` function is almost instantaneous.

Author(s)

Christophe Bruneel-Zupanc, `cbruneel.com`

References

Bruneel-Zupanc, C. (2024). Don't (fully) exclude me, it's not necessary! Identification with semi-IVs. arXiv preprint arXiv:2303.12667.

For empirical applications of the estimation of Marginal Treatment Effects with standard IVs, see for example:
Carneiro, P., Heckman, J. J., & Vytlacil, E. J. (2011). Estimating marginal returns to education. American Economic Review, 101(6), 2754-2781.

Brinch, C. N., Mogstad, M., & Wiswall, M. (2017). Beyond LATE with a discrete instrument. Journal of Political Economy, 125(4), 985-1039.

In particular, see Andresen, M. E. (2018). Exploring marginal treatment effects: Flexible estimation using Stata. The Stata Journal, 18(1), 118-158.

For double residual estimation of partially Linear models, see Robinson, P. M. (1988). Root-N-consistent semiparametric regression. Econometrica: Journal of the Econometric Society, 931-954.

For local polynomial regressions choice of degree & Bandwidth computation Fan, J., & Gijbels, I. (1996). Local polynomial modelling and its applications. Calonico, S., Cattaneo, M. D., & Farrell, M. H. (2019). nprobust: Nonparametric Kernel-Based Estimation and Robust Bias-Corrected Inference. Journal of Statistical Software, 91(8), 1–33. https://doi.org/10.18637/jss.v091.i08

Examples

```
# Load data:
data(roydata)

# Run the semi-IV regression
semiiv = semiivreg(y~d|w0|w1, data=roydata)
semiiv = semiivreg(y~d|w0|w1|Xbinary + Xcontinuous, data=roydata) # with covariates
semiiv = semiivreg(y~d|w0+Xbinary|w1+Xbinary|Xcontinuous, data=roydata)
# Xbinary has different effect on Y0 and Y1, Xcontinuous has the same.
semiiv = semiivreg(y~d|w0|w1, data=roydata, propensity_formula = d~w0+w1+w0:w1)
# if want to specify another first stage

semiiv$plot$mtr # if want to plot mtr_plot
```

---

simul_data *Simulate data from the Generalized Roy Model with semi-IVs*

Description

This function simulates data from the Generalized Roy Model with semi-IVs, following the simulation specified in Bruneel-Zupanc (2024).
For more details about the exact specification, see the vignettes here or by running `vignette("simul_data", package = "semiIVreg")`.

Usage

```
simul_data(N, model_type="heterogenous",
           param_y0, param_y1, param_p, param_Z, param_genX, param_error)
```

Arguments

N	Number of observations
model_type	Type of model: "heterogenous" or "homogenous"
param_y0	Parameters for Y0 = (delta0, beta0, beta0X1, beta0X2) i.e., intercept, effects on w0, X_1, X_2 on Y0.
param_y1	Parameters for Y1: (delta1, beta1, beta1X1, beta1X2). i.e., intercept, effects w1, X1, X2 on Y1.
param_p	Parameters for the selection: (alpha, alpha0, alpha1, alpha2, alphaX1, alphaX2) i.e., intercept and effects of w0, w1, w0w1, Xbinary, Xcontinuous on the latent utility.
param_Z	Parameters for the simulation of the semi-IVs: mean of W0 when X1=0, of W1 when X1=0, of W0 when X1=1, of W1 when X1=1; then variance of W0, W1, and covariance of W0 and W1.
param_genX	Parameters for the covariates: p_X1, mu_X2, sigma_X2.

`param_error` Parameters for the error terms: depends on model_type:
if heterogenous: variance of U0, U1, covariance of U0 and U1, variance of the cost (which has mean 0).
if homogenous: variance of U, variance of V, covariance of U and V.

Details

This function simulates data from the Generalized Roy Model with semi-IVs, following the simulation specified in Bruneel-Zupanc (2024).
For more details about the exact specification, see the vignette here or by running `vignette("simul_data", package = "semiIVreg")`. One can use it to simulate general model with heterogenous treatment effects, but also restricted ones with homogenous treatment effects.
`simul_data` was used to simulate the dataset available with this package, `data(roydata)` to obtain the simulated model with heterogenous treatment effect, and `data(roydata2)` to obtain the simulated model with homogenous treatment effect.

Value

A data frame with the following columns:

y The observed outcome.

d The treatment.

w0, w1 The semi-IVs entering only D=0 and D=1.

Xbinary, Xcontinuous Two covariates, one binary and one continuous.

y0, y1 The unobserved potential outcomes.

P The unobserved true treatment probability.

latent, V, Ud, U0, U1 The unobserved shocks V. Ud is the normalized V ranks. U0 and U1 are the outcome shocks. `latent` gives the latent utility term in the selection equation.

References

Bruneel-Zupanc, C. (2023). Don't (fully) exclude me, it's not necessary! Identification with semi-IVs. arXiv preprint arXiv:2303.12667.

Andresen, M. E. (2018). Exploring marginal treatment effects: Flexible estimation using Stata. The Stata Journal, 18(1), 118-158.

Heckman, J. J., Urzua, S., & Vytlacil, E. (2006). Understanding instrumental variables in models with essential heterogeneity. The Review of Economics and Statistics, 88(3), 389-432.

Heckman, J. J., & Vytlacil, E. J. (2007). Econometric evaluation of social programs, part II: Using the marginal treatment effect to organize alternative econometric estimators to evaluate social programs, and to forecast their effects in new environments. Handbook of econometrics, 6, 4875-5143.

Examples

```
N = 10000; set.seed(12345)

# Example 1: Heterogenous Treatment Effects.
model_type = "heterogenous"
param_error = c(1, 1, 0.6, 0.5) # var_u0, var_u1, cov_u0u1, var_cost
param_Z = c(0, 0, 0, 0, 1.5, 1.5, 0.9)
# meanW0 Xbinary0, meanW1 Xbinary0, meanW0 Xbinary1, meanW1 Xbinary1, varW0, varW1, covW0W1
param_p = c(0, -0.7, 0.7, 0, 0, 0) # constant, W0, W1, W0xW1, Xbinary, Xcontinuous
param_y0 = c(3.2, 0.8, 0, 0) # intercept, W0, Xbinary, Xcontinuous;
```

```
param_y1 = c(3.2+0.4, 0.5, 0, 0) # the +0.4 = ATE; W1, Xbinary, Xcontinuous;
param_genX = c(0.4, 0, 2)

data = simul_data(N, model_type, param_y0, param_y1, param_p, param_Z, param_genX, param_error)


# Example 2: Homogenous Treatment Effects (constant MTE)
model_type = "homogenous"
param_error = c(1, 1.5, -0.6) # var_u, var_v, cov_uv
param_Z = c(0, 0, 0, 0, 1.5, 1.5, 0.9)
param_p = c(0, -0.5, 0.5, 0, 0, 0) # the constant <=> mean_V
param_y0 = c(3.2, 0.8, 0, 0)
param_y1 = c(3.2+0.4, 0.5, 0, 0)
param_genX = c(0.4, 0, 2)

data1 = simul_data(N, model_type, param_y0, param_y1, param_p, param_Z, param_genX, param_error)


# Set the effects of w1 or w0 on its outcome to zero if want a valid IV, e.g.,
# param_y1 = c(3.2+0.4, 0, 0, 0) # w1 is a valid IV
# or: param_y0 = c(3.2, 0, 0, 0) # w0 is a valid IV
```

`supp_plot_fun` *Propensity score support plot*

Description

Returns the support plot by treatment status of the propensity score `Phat` included in a dataset.

Usage

```
supp_plot_fun(data, common_supp)
```

Arguments

`data`	Dataframe containing the treatment status, under a factor variable `Treatment` and the propensity score under the name `Phat`.
`common_supp`	Vector of two values indicating the common support of the plot. Default is the full support 0,1.
`colMTE`, `colD0`, `colD1`	Color of the MTE, MTR0 and MTR1 curves.

Examples

```
# Plot the true common support (with true - unobserved - propensity score)
# Using simulated data.
data(roydata); data=roydata;

# Syntax adjustment to use the function
data$Treatment = factor(data$d)
data$Phat = data$P # P is unobserved, we only know it because simulation here

#common_supp can be determined by looking at the plot - it's not necessary, just a graphical option
```

```
supp_P0 = c(min(data$Phat[which(data$d == 0)]), max(data$Phat[which(data$d== 0)]))
supp_P1 = c(min(data$Phat[which(data$d == 1)]), max(data$Phat[which(data$d == 1)]))
common_supp = c(max(supp_P0[1], supp_P1[1]), min(supp_P0[2], supp_P1[2]))

supp_plot = supp_plot_fun(data, common_supp); supp_plot
```

Index