Semiparametric Least Squares Inference for Causal Effects with R

Pierre Chausse*, Mihai Giurcanu†, Marinela Capanu‡, George Luta§

Abstract

This vignette explains how to use the causalSLSE package to estimate causal effects using the semiparametric least squares methods developed by Giurcanu et al. (2023). We describe the classes and methods implemented in the package as well as how they can be used to analyze synthetic and real data.

1 Introduction

This document presents the `causalSLSE` package describing the functions implemented in the package. It is intended for users interested in the details about the methods presented in Giurcanu et al. (2023) and how they are implemented. We first present the theory and then present the package in the following sections.

The general semiparametric additive regression model is

$$\begin{aligned} Y &= \beta_0(1-Z) + \beta_1 Z + \sum_{l=1}^{q} f_{l,0}(X_l)(1-Z) + \sum_{l=1}^{q} f_{l,1}(X_l)Z + \xi \\ &\equiv \beta_0(1-Z) + \beta_1 Z + f_0(X)(1-Z) + f_1(X)Z + \xi\,, \end{aligned} \tag{1}$$

where $Y \in \mathbb{R}$ is the response variable, Z is the treatment indicator defined as $Z = 1$ for the treated and $Z = 0$ for the nontreated, and $X \in \mathbb{R}^q$ is a q-vector of confounders. We approximate this model by the following regression model:

$$\begin{aligned} Y &= \beta_0(1-Z) + \beta_1 Z + \sum_{l=1}^{q} \psi_{l,0}^T U_{l,0}(1-Z) + \sum_{l=1}^{q} \psi_{l,1}^T U_{l,1} Z + \zeta \\ &\equiv \beta_0(1-Z) + \beta_1 Z + \psi_0^T U_0(1-Z) + \psi_1^T U_1 Z + \zeta\,, \end{aligned} \tag{2}$$

where $U_{l,k} = u_{l,k}(X_l) = (u_{j,l,k}(X_l) \ : \ 1 \le j \le p_{l,k}) \in \mathbb{R}^{p_{l,k}}$ is a vector of basis functions corresponding to the l^{th} nonparametric component of the k^{th} group $f_{l,k}(X_l)$, $\psi_{l,k} \in \mathbb{R}^{p_{l,k}}$ is an unknown vector of regression coefficients, $U_k = u_k(X) = (u_{l,k}(X_l) \ : \ 1 \le l \le q) \in \mathbb{R}^{p_k}$ and $\psi_k = (\psi_{l,k} \ : \ 1 \le l \le q) \in \mathbb{R}^{p_k}$, with $p_k = \sum_{l=1}^{q} p_{l,k}$. In this paper, we propose a data-driven method for selecting the vectors of basis functions $u_0(X)$ and $u_1(X)$. Note that we allow the number of basis functions $(p_{l,k})$ to differ across confounders and groups.

Let the following be the regression model estimated by least squares:

$$Y_i = \beta_0(1-Z_i) + \beta_1 Z_i + \psi_0^T U_{i,0}(1-Z_i) + \psi_1^T U_{i,1} Z_i + \zeta_i \text{ for } i = 1, ..., n\,, \tag{3}$$

*University of Waterloo, pchausse@uwaterloo.ca
†University of Chicago, giurcanu@uchicago.edu
‡Memorial Sloan Kettering Cancer Center, capanum@mskcc.org
§Georgetown University, George.luta@georgetown.edu

and $\hat{\beta}_0$, $\hat{\beta}_1$, $\hat{\psi}_0$ and $\hat{\psi}_1$ be the least squares estimators of the regression parameters. Then, the semiparametric least squares estimators (SLSE) of the average causal effect (ACE), causal effect on the treated (ACT) and causal effect on the nontreated (ACN) are defined respectively as follows:

$$\begin{aligned}
\text{ACE} &= \hat{\beta}_1 - \hat{\beta}_0 + \hat{\psi}_1^T \bar{U}_1 - \hat{\psi}_0^T \bar{U}_0 \\
\text{ACT} &= \hat{\beta}_1 - \hat{\beta}_0 + \hat{\psi}_1^T \bar{U}_{1,1} - \hat{\psi}_0^T \bar{U}_{0,1} \\
\text{ACN} &= \hat{\beta}_1 - \hat{\beta}_0 + \hat{\psi}_1^T \bar{U}_{1,0} - \hat{\psi}_0^T \bar{U}_{0,0}\,,
\end{aligned} \tag{4}$$

where $\bar{U}_k = \frac{1}{n}\sum_{i=1}^n U_{i,k}$, $\bar{U}_{k,1} = \frac{1}{n_1}\sum_{i=1}^n U_{i,k} Z_i$, $\bar{U}_{k,0} = \frac{1}{n_0}\sum_{i=1}^n U_{i,k}(1-Z_i)$, for $k = 0, 1$, and n_0 and n_1 are the sample sizes of the nontreated and treated groups respectively. As shown by Giurcanu et al. (2023), under some regularity conditions these estimators are consistent and asymptotically normal.

To derive the variance of these causal effect estimators, note that they can be expressed as a linear combination of the vector of least squares estimators. Let $\hat{\theta} = \{\hat{\beta}_0, \hat{\beta}_1, \hat{\psi}_0^T, \hat{\psi}_1^T\}^T$. Then, the causal effect estimators can be written as $\hat{D}_{\text{c}}^T \hat{\theta}$ for c=ACE, ACT or ACN, with $\hat{D}_{\text{ACE}} = \{-1, 1, -\bar{U}_0^T, \bar{U}_1^T\}^T$, $\hat{D}_{\text{ACT}} = \{-1, 1, -\bar{U}_{0,1}^T, \bar{U}_{1,1}^T\}^T$ and $\hat{D}_{\text{ACN}} = \{-1, 1, -\bar{U}_{0,0}^T, \bar{U}_{1,0}^T\}^T$. Since $\hat{D}_{\text{c}}$ is random, we need a first order Taylor expansion to derive the variance of the estimators. Assuming that the data set is iid and using the asymptotic properties of least squares estimators, we can show that the variance of ACE=$\hat{D}_{\text{ACE}}^T \hat{\theta}$ can be consistently estimated as follows (we can derive a similar expression for the ACT and ACN):

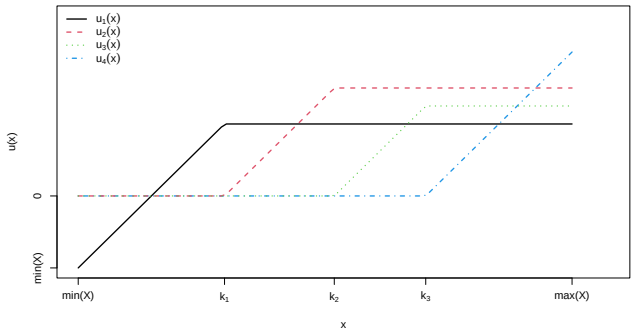
$$\hat{V}_{\text{ACE}} = \begin{pmatrix} -\hat{\beta}_0 & \hat{\beta}_1 & \hat{D}_{\text{ACE}}^T \end{pmatrix} \begin{pmatrix} \hat{\Sigma}_0 & \hat{\Sigma}_{0,1} & \hat{\Sigma}_{0,\hat{\theta}} \\ \hat{\Sigma}_{1,0} & \hat{\Sigma}_1 & \hat{\Sigma}_{1,\hat{\theta}} \\ \hat{\Sigma}_{\hat{\theta},0} & \hat{\Sigma}_{\hat{\theta},1} & \hat{\Sigma}_{\hat{\theta}} \end{pmatrix} \begin{pmatrix} -\hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{D}_{\text{ACE}} \end{pmatrix}, \tag{5}$$

where $\hat{\Sigma}_k = \widehat{\text{var}(\bar{U}_k)}$, $\hat{\Sigma}_{k,l} = \widehat{\text{cov}(\bar{U}_k, \bar{U}_l)}$, $\hat{\Sigma}_{k,\hat{\theta}} = \hat{\Sigma}_{\hat{\theta},k}^T = \widehat{\text{cov}(\bar{U}_k, \hat{\theta})}$, for $k, l = 0, 1$, and $\hat{\Sigma}_{\hat{\theta}}$ is a consistent estimator of the variance of $\hat{\theta}$. We will discuss the choice of the covariance matrix estimator $\hat{\Sigma}_{\hat{\theta}}$ in the next section.

To understand the package, it is important to know how the $u_{l,k}(X_l)$'s are defined. For clarity, let's write $U_{l,k} = u_{l,k}(X_l)$ as $U = u(X) = (u_j(X) \,:\, 1 \le j \le p) \in \mathbb{R}^p$. We just need to keep in mind that it is different for the treated and nontreated groups and also for different confounders. We describe here how to construct the local linear splines for a given confounder X in a given group. To this end, let $\{\kappa_1, \ldots, \kappa_{p-1}\}$ be a set of $p-1$ knots strictly inside the support of X satisfying $\kappa_1 < \kappa_2 < \ldots < \kappa_{p-1}$. In the case of local linear splines described in the paper, we have:

$$\begin{aligned}
u_1(x) &= xI(x \le \kappa_1) + \kappa_1 I(x > \kappa_1) \\
u_j(x) &= (x - \kappa_{j-1})I(\kappa_{j-1} \le x \le \kappa_j) + (\kappa_j - \kappa_{j-1})I(x > \kappa_j)\,, \quad 2 \le j \le p-1 \\
u_p(x) &= (x - \kappa_{p-1})I(x > \kappa_{p-1})
\end{aligned} \tag{6}$$

Therefore, if the number of knots is equal to 1, we only have two local linear splines. Since the knots must be strictly inside the support of X, for any categorical variable with two levels, the number of knots must be equal to zero. In this case, $u(x) = x$. For general ordinal variables, the number of knots cannot exceed the number of levels minus two. The following illustrates local linear spline functions when the number of knots is equal to 3:



Note that for the sample regression, the knots of X_l for group k, $l = 1, ..., q$, must be strictly inside the sample range of $(X_{i,l} \ : \ 1 \leq i \leq n, \ Z_i = k) \in \mathbb{R}^{n_k}$, where n_k is the sample size in group k, instead of inside the support of X_l.

The following section explains in details how to use the package to estimate the causal effects using this method, and the last section summarizes the package by providing a list of all objects and methods.

2 The `causalSLSE` package

2.1 The Semiparametric LSE model

Note that the regression model presented by Equation (3) can expressed as:

$$\begin{aligned} Y_i &= \beta_0 + \psi_0^T U_{i,0} + \zeta_{i,0} \text{ for } i \text{ s.t. } Z_i = 0 \\ Y_i &= \beta_1 + \psi_1^T U_{i,1} + \zeta_{i,1} \text{ for } i \text{ s.t. } Z_i = 1 \,. \end{aligned} \tag{7}$$

Estimating Equation (3) is identical to estimating the previous two models separately. The latter may even be numerically more accurate since it avoids many unnecessary operations. Also, as mentioned in the previous section, the knots and basis functions are obtained separately for the treated and nontreated. Therefore, we can see the model from Equation (3) as two semiparametric LSE (SLSE) models, one for the treated and one for the nontreated, and this is the approach that we take in the package. One benefit of this approach is to allow an extension to multiple treatment models. For example, a two treatment model, with two treated groups and one nontreated group, is like a one treatment model with one more SLSE model.

Since our causal SLSE model is a collection of SLSE models, we start by presenting how SLSE models are defined in the package. We ignore for now that our objective is to estimate causal effects and consider the following SLSE model:

$$\begin{aligned} Y &= \beta + \sum_{l=1}^{q} \psi_l^T U_l + \zeta \\ &\equiv \beta + \psi^T U + \zeta \,, \end{aligned} \tag{8}$$

where $U_l = u_l(X_l) = (u_{j,l}(X_l) \ : \ 1 \leq j \leq p_l) \in \mathbb{R}^{p_l}$, $\psi_l \in \mathbb{R}^{p_l}$ is an unknown vector of regression coefficients, $U = u(X) = (u_l(X_l) \ : \ 1 \leq l \leq q) \in \mathbb{R}^p$ and $\psi = (\psi_l \ : \ 1 \leq l \leq q) \in \mathbb{R}^p$, with $p = \sum_{l=1}^{q} p_l$. The next section explains how the knots are determined.

2.1.1 The starting knots

The starting knots are automatically generated by the function `slseKnots`. The following is the list of arguments of the function:

- **form**: A formula with the right-hand side being the list of covariates. If a left-hand side is provided, the `slseKnots` function will ignore it, because its purpose is only to generate the knots.
- **data**: A `data.frame` containing all variables included in the formula.
- **X**: Alternatively, we can input directly the matrix of covariates. If a matrix `X` is provided, the arguments `form` and `data` are ignored.
- **nbasis**: A function that determines the number of basis functions as explained in the procedure below. The default is `nbasis=function(n) n^0.3`.
- **knots**: This argument is used to set the knots manually. We will explain how to use this argument in the next section.

The following is the procedure implemented by the function `slseKnots`. It explains the procedure for any covariate X.

1. The starting number of knots, also equal to the number of basis functions minus 1, depends on the type of covariate. Unless it has a type that restricts the number of knots, which is explained below, it is determined by the argument `nbasis`. This is a function of one argument, the same size, and it returns the default number of basis functions. This number cannot be smaller than 2 (we will see other ways of forcing the number of basis functions to be equal to 1 below) and must be an integer. To be more specific, the number of basis functions is set to the maximum between 2 and the ceiling of what the `nbasis` function returns. For example, if the sample size is 500, the default starting number of basis functions is 7=`ceiling(500^0.3)`, which implies a starting number of knots of 6. It is possible to have a number of knots that does not depend on the sample size. All we need is to set the argument `nbasis` to a function that returns an integer, e.g., `nbasis=function(n) 4` for 4 basis functions or 3 knots.
2. Let $(p-1)$ be the number of knots determined in the previous step. The default knots are obtained by computing $p+1$ sample quantiles of X for equally spaced probabilities from 0 to 1, and by dropping the first and last quantiles. For example, if the number of knots is 3, then the initial knots are given by quantiles for the probabilities 0.25, 0.5 and 0.75.
3. We drop any duplicated knots and any knots equal to either the max or the min of X. If the resulting number of knots is equal to 0, the vector of knots is set to `NULL`. When the vector of knots is equal to `NULL` for a variable X, it means that $u(x) = x$.

The last step implies that the number of knots for all categorical variables with two levels is equal to 0. For nominal variables with a small number of levels, the number of knots, a subset of the levels, may be smaller than the ones defined by `nbasis`. For example, when the number of levels for a nominal variable is 3, the number of knots cannot exceed 1.

To illustrate how to use the package, we are using the dataset from Lalonde (1986). The purpose of this dataset is to estimate the causal effect of a training program on real income, but we ignore it for the moment. The dataset is included in the `causalSLSE` package and can be loaded as follows.

```
library(causalSLSE)
data(nsw)
```

The dependent variable is the real income in 1978 (`re78`) and the dataset contains the following covariates: the continuous variables age (`age`), education (`ed`) and the 1975 real income (`re75`), and the binary variables `black`, `hisp`, `married` and `nodeg`. We start by considering a model that includes the covariates `age`, `re75`, `ed`, and `married`. Since we do not need to specify the left-hand side, we can create the initial knots as follows

```
k <- slseKnots(form = ~ age + re75 + ed + married, data = nsw)
```

The function returns an object of class `slseKnots` and its `print` method produces a nice display separating confounders with and without knots. For example, the following are the starting knots:

```
print(k)
```

```
Covariates with no knots:
    married

Covariates with knots:
age :
      12.5% 25% 37.5% 50% 62.5% 75% 87.5%
Knots    18  19    21  23    25  27    31

re75 :
        50% 62.5%  75% 87.5%
Knots 936.2  2037 4023  8015

ed :
      12.5% 25% 37.5% 62.5% 87.5%
Knots     8   9    10    11    12
```

The sample size is equal to 722 and the default `nbasis` is $n^{0.3}$, which implies a default number of starting knots equal to 7 = `ceiling`($722^{0.3}$)-1. This is the number of knots we have for `age`. However, the number of knots for `ed` is 5 and it is 4 for `re75`. To understand why, the following shows the 7 default quantiles for `re75` and `ed` (the `type` argument of the `quantile` function is the same as it is implemented in the package):

```
p <- seq(0,1,len=9)[c(-1,-9)] # these are the probabilities with 7 knots
quantile(nsw[,'re75'], p, type=1)
```

```
    12.5%       25%     37.5%       50%     62.5%       75%     87.5%
   0.0000    0.0000    0.0000  936.1773 2036.7900 4023.2110 8015.4420
```

```
quantile(nsw[,'ed'], p, type=1)
```

```
12.5%   25% 37.5%   50% 62.5%   75% 87.5%
    8     9    10    10    11    11    12
```

We can see that the first three quantiles of `re75` are equal to its minimum, so they are removed. For the `ed` variable, 10 and 11 appear twice, so one 10 and one 11 must be removed.

Note that each object in the package is S3-class, so the elements can be accessed using the operator $. For example, we can extract the knots for `age` as follows:

```
k$age
```

```
12.5%   25% 37.5%   50% 62.5%   75% 87.5%
   18    19    21    23    25    27    31
```

Note that the covariates are listed in the "no knots" section when their values are set to `NULL`. In the above example, it is the case of `married` because it is a binary variable. As we can see its list of knots it set to `NULL`:

```
k$married
```

```
NULL
```

2.1.2 Creating a SLSE model

The SLSE model is created by the function `slseModel`. The arguments of the function are the same as for the `slseKnots` function except for the argument `X`, which is not needed. The difference is that `form` must include the left-hand side variable. For example, we can create a SLSE model using `re78` as dependent variable and the same covariates used in the previous section as follows:

```
mod1 <- slseModel(form = re78 ~ age + re75 + ed + married, data = nsw)
```

The function returns an object of class `slseModel` and its `print` method provides a summary of its specification:

```
print(mod1)
```

```
Semiparametric LSE Model
************************

Number of observations:  722
Selection method: Default

Covariates approximated by SLSE (num. of knots):
    age(7), re75(4), ed(5)
Covariates not approximated by SLSE:
    married
```

Note that the selection method is set to `Default` when the knots are selected using the procedure described in the previous section. Also, the `print` function shows the number of knots for each covariate inside the parentheses next to its name. In this example, we see that the initial number of knots for `age`, `re75` and `ed` are respectively 7, 4 and 5. The function selects the knots using the `slseKnots` and stores them in the object under `knots`. We can print them using the $ operator as follows and compare them with the ones obtained in the previous section:

```
mod1$knots
```

```
Covariates with no knots:
    married

Covariates with knots:
age :
      12.5% 25% 37.5% 50% 62.5% 75% 87.5%
Knots    18  19    21  23    25  27    31

re75 :
        50% 62.5%  75% 87.5%
Knots 936.2  2037 4023  8015

ed :
      12.5% 25% 37.5% 62.5% 87.5%
Knots     8   9    10    11    12
```

Note that we can also print the knots by running the command `print(mod1, which="selKnots")`.

In order to present another example with different types of covariates, the dataset `simDat4` is included in the package. This is a simulated dataset which contains special types of covariates. It helps to further illustrate how the knots are determined. The dataset contains a continuous variable `X1` with a large proportion of zeros, the categorical variable `X2` with 3 levels, an ordinal variable `X3` with 3 levels, and a binary variable `X4`. The levels for `X2` are {"first","second","third"} and for `X3` the levels are {1,2,3}.

```
data(simDat4)
mod2 <- slseModel(Y ~ X1 + X2 + X3 + X4, data = simDat4)
print(mod2, which="selKnots")
```

```
Semiparametric LSE Model: Selected knots
****************************************
Selection method: Default

Covariates with no knots:
    X2second, X2third, X4

Covariates with knots:
X1 :
      42.86% 57.14% 71.43% 85.71%
Knots 0.4258  2.304  7.911  17.58

X3 :
      42.86%
Knots      2
```

Character-type variables are automatically converted into factors. It is also possible to define a numerical variable like `X3` as a factor by using the function `as.factor` in the formula. We see that the 2 binary variables `X2second` and `X2third` are created and `X2first` is omitted to avoid multicollinearity. For the binary variable `X4`, the number of knots is set to 0, and for the ordinal variable `X3`, the number of knots is set to 1 because the min and max values 1 and 3 cannot be selected.

2.1.3 Selecting the knots manually

The user has control over the selection of knots through the argument `knots`. When the argument is missing (the default), all knots are set automatically as described above. One way to set the number of knots to 0 for all variables is to set the argument to `NULL`.

```
slseModel(re78 ~ age + re75 + ed + married, data = nsw, knots = NULL)
```

```
Semiparametric LSE Model
************************

Number of observations:  722
Selection method: User Based

Covariates approximated by SLSE (num. of knots):
    None
Covariates not approximated by SLSE:
    age, re75, ed, married
```

Notice that the selection method is defined as "User Based" whenever the knots are provided manually by the user. The other option is to provide a list of knots. For each variable, we have three options:

- `NA`: The knots are set automatically for this variable only.
- `NULL`: The number of knots is set to 0 for this variable only.
- A numeric vector: The vector cannot contain missing or duplicated values and must be strictly inside the sample range of the variable.

In the following, we describe all possible formats for the list of knots.

Case 1: An unnamed list of length equal to the number of covariates.

In that case, the knots must be defined in the same order as the order of the variables implied by the formula. For example, if we want to set an automatic selection for `age`, no knots for `ed` and the knots $\{1000, 5000, 10000\}$ for `re75`, we proceed as follows. Note that setting the value to `NA` or `NULL` has the same effect for the binary variable `married`.

```
selK <- list(NA, c(1000,5000,10000), NULL, NA)
mod <- slseModel(re78 ~ age + re75 + ed + married, data = nsw,
                 knots = selK)
print(mod, which = "selKnots")
```

```
Semiparametric LSE Model: Selected knots
****************************************
Selection method: User Based

Covariates with no knots:
    ed, married

Covariates with knots:
age :
      12.5% 25% 37.5% 50% 62.5% 75% 87.5%
Knots    18  19    21  23    25  27    31

re75 :
        k1   k2    k3
Knots 1000 5000 10000
```

Case 2: A named list of length equal to the number of covariates.

In that case, the order of the list of variables does not matter. The `slseModel` function will automatically reorder the variables to match the order implied by the formula. The names must match perfectly the variable names generated by R. In the following example, we want to add the interaction between `ed` and `age`. We want the same set of knots as in the previous example and no knots for the interaction term. The name of the interaction depends on how we enter it in the formula. For example, it is "age:ed" if we enter `age*ed` in the formula and "ed:age" if we enter `ed*age`. For factors, the names depend on which binary variable is omitted. Using the above example with the `simDat4` model, if we interact `X2` and `X4` by adding `X2*X4` to the formula, the names of the interaction terms are "X2second:X4" and "X2third:X4". When we are uncertain about the names, we can print the knots of a model with the default sets of knots. In the following, we change the order of variables to show that the order does not matter.

```
selK <- list(married = NA, ed = NULL, 'age:ed' = NULL, re75 = c(1000,5000,10000), age = NA)
model <- slseModel(re78 ~ age * ed + re75 + married, data = nsw, knots = selK)
print(model, which="selKnots")
```

```
Semiparametric LSE Model: Selected knots
****************************************
Selection method: User Based

Covariates with no knots:
    ed, married, age:ed

Covariates with knots:
age :
      12.5% 25% 37.5% 50% 62.5% 75% 87.5%
Knots    18  19    21  23    25  27    31

re75 :
        k1   k2    k3
Knots 1000 5000 10000
```

Case 3: A named list of length strictly less than the number of covariates.

The names of the selected variables must match perfectly the names generated by R and the order does not matter. This is particularly useful when the number of covariates is large. If we consider the previous example, the knots are set manually only for `age`. By default, all names not included in the list of knots are set to `NA`. Therefore, we can create the same model from the previous example as follows:

```
selK <- list(ed = NULL, 'age:ed' = NULL, re75 = c(1000,5000,10000))
model <- slseModel(re78 ~ age * ed + re75 + married, data = nsw, knots = selK)
print(model, which="selKnots")
```

```
Semiparametric LSE Model: Selected knots
****************************************
Selection method: User Based

Covariates with no knots:
    ed, married, age:ed

Covariates with knots:
age :
      12.5% 25% 37.5% 50% 62.5% 75% 87.5%
Knots    18  19    21  23    25  27    31

re75 :
        k1   k2    k3
Knots 1000 5000 10000
```

Note that the previous case offers an easy way of setting the number of knots to 0 for a subset of the covariates. For example, suppose we want to add more interaction terms and set the knots to 0 for all of them. We can proceed as follows.

```
selK <- list('age:ed' = NULL, 'ed:re75' = NULL, 'ed:married' = NULL)
model <- slseModel(re78 ~ age * ed + re75 * ed + married * ed,
                   data = nsw, knots = selK)
```

```
model
```

```
Semiparametric LSE Model
************************

Number of observations:  722
Selection method: User Based

Covariates approximated by SLSE (num. of knots):
    age(7), ed(5), re75(4)
Covariates not approximated by SLSE:
    married, age:ed, ed:re75, ed:married
```

Note also that `slseModel` deals with interaction terms as with any other variable. For example, `ed:black` is like a continuous variable with a large proportion of zeros. The following shows the default selected knots for `ed:black`.

```
model <- slseModel(re78 ~ age + ed * black, data = nsw)
model$knots[["ed:black"]]
```

```
  25% 37.5%   50% 62.5% 87.5%
    8     9    10    11    12
```

We can see that the number of knots is smaller than 7. This is because `ed:black` has many zeros and the quantiles equal to the minimum value are removed.

2.1.4 Methods for `slseModel` objects

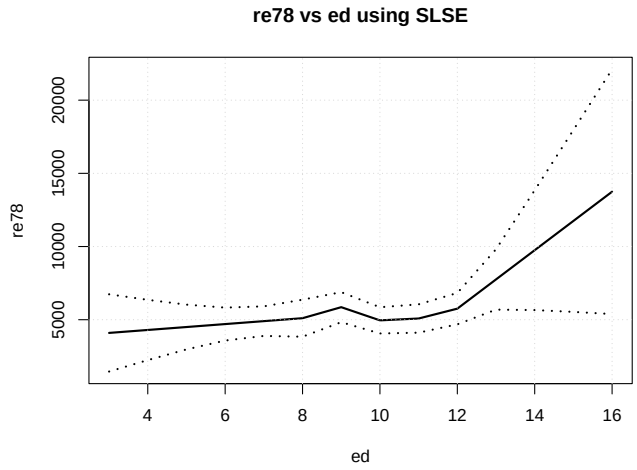
Other methods are registered for `slseModel` objects. For example, we can estimate `slseModel` objects using the `estSLSE` method and summarize the results using the `summary` method. The following is an example using a simpler model:

```
mod2 <- slseModel(form = re78 ~ ed + married, data = nsw)
fit2 <- estSLSE(mod2)
summary(fit2)
```

```
Semiparametric LSE
******************
Selection method: Default

Residuals:
   Min     1Q Median     3Q    Max
-11472  -4846  -1548   3195  55335

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3384.7     2304.0   1.469   0.1418
U.ed_1         201.2      328.0   0.613   0.5396
U.ed_2         752.2      824.6   0.912   0.3616
U.ed_3        -900.4      695.0  -1.296   0.1951
U.ed_4         126.5      672.5   0.188   0.8508
U.ed_5         672.8      735.9   0.914   0.3605
U.ed_6        1997.6     1105.1   1.808   0.0707 .
U.married      674.2      652.5   1.033   0.3015
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6212 on 714 degrees of freedom
Multiple R-squared:  0.02276,   Adjusted R-squared:  0.01317
```

We can also plot the predicted dependent variable as a function of education using the `plot` method and add a confidence region:

```
plot(fit2, "ed", interval="confidence", level=0.95)
```



In the next section, we present the `cslseModel` class which represents the causal-SLSE model of Equation (3). We will see that it is just a list of `slseModel` objects. Therefore, the methods registered for `cslseModel` objects are derived from `slseModel` methods. Since this vignette is about causal-SLSE, we choose to present these methods through the `cslseModel` object.

2.2 The causal-SLSE model (`cslseModel`)

The function `cslseModel` returns an object of class `cslseModel` (or causal-SLSE model). It is a list of `slseModel` objects, one for the treated and one for the nontreated. The function has the same arguments as `slseModel`, plus the argument `groupInd` that specifies which value of Z is associated with the treated and which one is associated with the nontreated. The default is `groupInd=c(treated = 1, nontreated = 0)`. It is possible to have other values or even characters as indicator, but the names must be treated and nontreated. We will allow more names in future version of the package once the multiple treatment method is implemented.

The argument `form` must include a formula linking the outcome and the treatment indicator and a formula listing the confounders, separated by the operator `|`. In the following example, we see the formula linking the outcome `re78` and the treatment indicator `treat`, and a list of confounders:

```
model1 <- cslseModel(re78 ~ treat | ~ age + re75 + ed + married, data = nsw)
```

Its `print` method summarizes the characteristics of the model. It is like the `slseModel` object, but the information is provided by treatment group

```
model1
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method: Default

Confounders approximated by SLSE (num. of knots):
    treated: age(5), re75(3), ed(4)
    nontreated: age(6), re75(4), ed(4)
Confounders not approximated by SLSE:
    treated: married
    nontreated: married
```

The object also contains additional information about the model stored as attributes:

```
attr(model1, "treatedVar")
```

```
[1] "treat"
```

```
attr(model1, "groupInd")
```

```
   treated nontreated
         1          0
```

This information is needed in order to compute the causal effects. The object `model1` is a list of 2 elements, `treated` and `nontreated`, which are `slseModel` objects. Following Section 2.1.2, we can therefore access the knots of a specific group as follows:

```
model1$treated$knots
```

```
Covariates with no knots:
    married

Covariates with knots:
age :
      16.67% 33.33% 50% 66.67% 83.33%
Knots     19     21  23     26     29

re75 :
       50% 66.67% 83.33%
Knots 1117   2657   6511

ed :
      16.67% 33.33% 50% 83.33%
Knots      9     10  11     12
```

Alternatively, we can use the `print` method for `slseModel` objects and print the knots of a specific group using the command `print(model1$treated, which="selKnots")`. To print the list of knots for both groups, we can print the `cslseModel` object as follows:

```
print(model1, which="selKnots")
```

```
treated
*******
Selection method: Default

Covariates with no knots:
    married

Covariates with knots:
age :
      16.67% 33.33% 50% 66.67% 83.33%
Knots     19     21  23     26     29

re75 :
       50% 66.67% 83.33%
Knots 1117   2657   6511

ed :
      16.67% 33.33% 50% 83.33%
Knots      9     10  11     12

nontreated
**********
Selection method: Default

Covariates with no knots:
    married

Covariates with knots:
age :
      14.29% 28.57% 42.86% 57.14% 71.43% 85.71%
Knots     18     20     22     25     27     31

re75 :
```

```
      42.86% 57.14% 71.43% 85.71%
Knots  240.1   1406   2856   7667

ed :
      14.29% 42.86% 57.14% 85.71%
Knots      9     10     11     12
```

To understand how to create a `cslseModel` when the treatment indicator is not binary, consider the dataset `simDat4` that we described in Section 2.1.2. The dataset also contains the variable `treat`, which is a character variable equal to "treat" when `Z=1` and "notreat" when `Z=0`. We can create a `cslseModel` object using `treat` instead of `Z` by specifying the value associated with each group in the argument `groupInd`:

```
model2 <- cslseModel(Y ~ treat | ~ X1 + X2 + X3 + X4, data = simDat4,
                     groupInd = c(treated = "treat", nontreated = "notreat"))
model2
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  246
Number of  nontreated :  254
Selection method: Default

Confounders approximated by SLSE (num. of knots):
    treated: X1(4), X3(1)
    nontreated: X1(4), X3(1)
Confounders not approximated by SLSE:
    treated: X2second, X2third, X4
    nontreated: X2second, X2third, X4
```

If some values of the treatment indicator variable differ from the values in `groupInd`, the function will return an error message.

2.2.1 Setting the knots manually

As for SLSE models, we can select the knots using the argument `knots`. The procedure is the same as in Section 2.1.3 (Cases 1 to 3), but we need to specify the name of the group associated with the knots. If `knots` is set to `NULL`, the number of knots is set to 0 for all confounders and all groups. If we only want the number of knots to be 0 for one group, we need to specify which group. For example, the number of knots is set to 0 for the treated only in the following:

```
selK <- list(treated=NULL)
cslseModel(re78 ~ treat | ~ age + re75 + ed + married, data = nsw,
           knots = selK)
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method for the treated: User Based
Selection method for the nontreated: Default

Confounders approximated by SLSE (num. of knots):
    treated: None
    nontreated: age(6), re75(4), ed(4)
Confounders not approximated by SLSE:
    treated: age, re75, ed, married
    nontreated: married
```

If a group is missing from the argument `knots`, the knots of the missing group are set automatically. For example, if we want to set the knots as in Case 1, but only for the nontreated and let `cslseModel` choose them for the treated, we would proceed as follows:

```
selK <- list(nontreated=list(NA, c(1000,5000,10000), NULL, NA))
model <- cslseModel(re78 ~ treat | ~ age + re75 + ed + married, data = nsw,
```

```
                    knots = selK)
print(model, which = "selKnots")
```

```
treated
*******
Selection method: Default

Covariates with no knots:
    married

Covariates with knots:
age :
      16.67% 33.33% 50% 66.67% 83.33%
Knots     19     21  23     26     29

re75 :
       50% 66.67% 83.33%
Knots 1117   2657   6511

ed :
      16.67% 33.33% 50% 83.33%
Knots      9     10  11     12

nontreated
**********
Selection method: User Based

Covariates with no knots:
    ed, married

Covariates with knots:
age :
      14.29% 28.57% 42.86% 57.14% 71.43% 85.71%
Knots     18     20     22     25     27     31

re75 :
        k1   k2    k3
Knots 1000 5000 10000
```

2.2.2 Estimating the model

Given the set of knots from the model object, the estimation is just a least squares method applied to the regression model given by:

$$Y = \beta_0(1-Z) + \beta_1 Z + \psi_0^T U_0(1-Z) + \psi_1^T U_1 Z + \zeta\,,$$

where $U_0 = u_0(X)$ and $U_1 = u_1(X)$ are defined above (which depend on the knots of the model). The method that estimates the model is `estSLSE` which has two arguments, but one of them is mainly used internally by other functions. We present them in case they are needed. The arguments are:

- `model`: A model created by the function `cslseModel`.
- `selKnots`: It is a list of one or two elements, one for each group. Each element is a list of integers to select knots for the associated group. For example, suppose we have 2 confounders with 5 knots each. If we want to estimate the model with only the first knot for the first confounder and knots 3 and 5 for the second confounder for the treated and all knots for the nontreated, we set `selKnots` to `list(treated=list(1L,c(3L, 5L)))`. By default it is missing and all the knots from the model are used.

We illustrate the use of `estSLSE` with a simple model containing 2 confounders and a maximum of one knot.

```
model <- cslseModel(re78 ~ treat | ~ age + married, data = nsw,
                    nbasis = function(n) 2)
```

Note that the `cslseModel` object is a list of `slseModel` objects. Also, we saw that the above model can be written as two regression models, one for each group. Therefore, the `estSLSE` method is simply estimating the `slseModel` objects separately. For example, we can obtain $\{\hat{\beta}_1, \hat{\psi}_1\}$ as follows:

```
estSLSE(model$treated)
```

```
Semiparametric LSE
******************
Selection method: Default


(Intercept)      U.age_1      U.age_2    U.married
    3754.98        89.25        22.22      1435.28
```

The output is an object of class `slseFit`. When we apply the method to `model`, the model is estimated for each group:

```
fit <- estSLSE(model)
fit
```

```
Causal Semiparametric LSE
*************************
Selection method: Default

treated
*******
(Intercept)      U.age_1      U.age_2    U.married
    3754.98        89.25        22.22      1435.28

nontreated
**********
(Intercept)      U.age_1      U.age_2    U.married
    4558.28        27.80       -12.51      -115.82
```

This is an object of class `cslseFit`, which is a list of `slseFit`. Like `cslseModel` objects, it also contains the information about the treatment indicator variable and the value associated with each group stored as attributes:

```
attr(fit, "treatedVar")
```

```
[1] "treat"
```

```
attr(fit, "groupInd")
```

```
   treated nontreated
         1          0
```

When we print, `fit`, the coefficients are separated by group. The coefficients for the treated correspond to $\{\hat{\beta}_1, \hat{\psi}_1\}$ and the coefficients for the nontreated correspond to $\{\hat{\beta}_0, \hat{\psi}_0\}$. We can access the estimated SLSE model for each group using the $ operator. For example, the following is the estimated model for the treated:

```
fit$treated
```

```
Semiparametric LSE
******************
Selection method: Default


(Intercept)      U.age_1      U.age_2    U.married
    3754.98        89.25        22.22      1435.28
```

A more detailed presentation of the results can be obtained using the `summary` method. The only arguments of `summary` are the `cslseFit` object and `vcov.`. The latter is a function that returns the estimated covariance matrix of the LSE. By default, it is equal to the `vcovHC` function of the `sandwich` package (Zeileis (2006)) with its default `type="HC3"`. The following is an example with the previous model using the HC0 type:

```
s <- summary(fit, type="HC0")
```

The object `s` is an object of class `summary.cslseFit`, which is a list of objects of class `summary.slseFit`,

one for each group. By default, if we print `s`, we will see the two LSE summary tables, one for each group. Alternatively, we can print the result for one group using the $ operator. For example, the following is the result for the nontreated:

```
s$nontreated
```

```
Semiparametric LSE
******************
Selection method: Default

Residuals:
   Min     1Q Median     3Q    Max
 -5198  -5031  -1364   3216  34341

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4558.28    2711.20   1.681   0.0927 .
U.age_1        27.80     135.20   0.206   0.8371
U.age_2       -12.51      54.80  -0.228   0.8194
U.married    -115.82     782.92  -0.148   0.8824
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5738 on 421 degrees of freedom
Multiple R-squared:  0.0001584, Adjusted R-squared:  -0.006966
```

We can see that the only knot for `age` in the nontreated group is 23:

```
model$nontreated$knots$age
```

```
50%
 23
```

Therefore, the coefficient of `U.age_1` is the effect of age for the nontreated on `re78` when `age`$\leq$ 23 and `U.age_2` is the effect when `age`$>$ 23.

The `extract` method

The package comes with an `extract` method for `slseFit` objects, which allows to present estimated SLSE models in a LaTeX table using the `texreg` package of Leifeld (2013). There is no `extract` method for `cslseFit` objects, but we can still use `texreg` by converting `cslseFit` objects into lists using the method `as.list`. Here is an example (the argument `table=FALSE` is used to avoid having a floating table):

```
library(texreg)
texreg(as.list(fit), table=FALSE)
```

	treated	nontreated
(Intercept)	3754.98	4558.28
	(4479.03)	(3101.89)
U.age_1	89.25	27.80
	(215.11)	(151.03)
U.age_2	22.22	−12.51
	(84.70)	(61.58)
U.married	1435.28	−115.82
	(1123.99)	(788.83)
R^2	0.01	0.00
Adj. R^2	−0.00	−0.01
Num. obs.	297	425

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

2.2.3 The `predict` method

The `predict` method is very similar to `predict.lm`. We use the same arguments: `object`, `interval`, `se.fit`, `newdata` and `level`. The difference is that it returns the predicted outcome for the treated and nontreated separately, and the argument `vcov.` provides a way of changing how the least squares covariance matrix is computed. By default, it is computed using `vcovHC` from the `sandwich` package. The function returns a list

of 2 elements, `treated` and `nontreated`. By default (`se.fit=FALSE` and `interval="none"`), each element contains a vector of predictions. Here is an example with the previously fitted model `fit`:

```
predict(fit, newdata = data.frame(treat = c(1,1,0,0),age = 20:23, married = 1))
```

```
$treated
[1] 6975.337 7064.591

$nontreated
[1] 5054.036 5081.834
```

If `interval` is set to "confidence", but `se.fit` remains equal to `FALSE`, each element contains a matrix containing the prediction, and the lower and upper confidence limits, with the confidence level determined by the argument `level` (set to 0.95 by default). Here is an example with the same fitted model:

```
predict(fit,  newdata = data.frame(treat = c(1,1,0,0),age = 20:23, married = 1),
        interval = "confidence")
```

```
$treated
       fit    lower    upper
1 6975.337 4646.673 9304.001
2 7064.591 4741.653 9387.528

$nontreated
       fit    lower    upper
3 5054.036 3574.096 6533.975
4 5081.834 3544.849 6618.820
```

If `se.fit` is set to `TRUE`, each element, treated or nontreated, is a list with the elements `pr`, containing the predictions, and `se.fit`, containing the standard errors. In the following, we show the result for the same fitted model:

```
predict(fit, newdata = data.frame(treat = c(1,1,0,0),age = 20:23, married = 1),
        se.fit = TRUE)
```

```
$treated
$treated$fit
[1] 6975.337 7064.591

$treated$se.fit
       1        2
1188.116 1185.194


$nontreated
$nontreated$fit
[1] 5054.036 5081.834

$nontreated$se.fit
       3        4
755.0851 784.1907
```

2.2.4 The `plot` method

The `predict` method is called by the `plot` method to visually represent the predicted outcome for the treated and nontreated with respect to a given confounder, controlling for the other variables in the model. Note that this method is very close to the `plot` method for `slseFit` objects. In fact, the arguments are the same with some exceptions that we briefly explain below. Since the predicted outcome is obtained separately for the treated and nontreated the method for `cslseFit` objects simply applies the method for `slseFit` objects to each group. The following is the list of arguments:

- **x**: An object of class `cslseFit` (or `slseFit`).
- **y**: An alias for `which` for compatibility with the generic `plot` function.
- **which**: confounder to plot against the outcome variable. It can be an integer (the position of the confounder) or a character (the name of the confounder)

- **interval**: The type of confidence interval to display. The default is "none". The alternative is "confidence".
- **level**: The confidence level when `interval="confidence"`. The default is 0.95.
- **fixedCov**: Optional named lists of fixed values for some or all other confounders in each group. The values of the confounders not specified are determined by the argument `FUN`. To fix some confounders for both groups, `fixedCov` is just a named list with the names being the variable names. To fix them to different values for the treated and nontreated, `fixedCov` is a named list of 1 or 2 elements (for the treated, nontreated or both), each element being a named list of values for the covariates. See the examples below. When applied to `slseFit` objects, it is just a named list with the variables names.
- **vcov.**: An optional function to compute the estimated matrix of covariance of the least squares estimators. This argument only affects the confidence intervals. The default is `vcovHC` from the `sandwich` package with `type="HC3"`.
- **add**: Should the curves be added to an existing plot? The default is `FALSE`.
- **addToLegend**: An optional character string to add to the legend next to "treated" and "nontreated". Note that a legend is not added when applied to `slseFit` objects, so this argument has no effect in that case.
- **addPoints**: Should we include the scatterplot of the outcome and confounder to the graph? The default is `FALSE`.
- **FUN**: A function to determine how the other confounders are fixed. The default is `mean`. Note that the function is applied to each group separately.
- **plot**: By default, the method produces a graph. Alternatively, we can set this argument to `FALSE` and it returns one `data.frame` per group with the variable selected by `which` and the prediction. This could be useful if one wants to design the graphs differently.
- **graphPar**: A list of graphical parameters if not satisfied with the default ones.
- `...`: Other arguments are passed to the `vcov.` function.

The default set of graphical parameters can be obtained by running the function `causalSLSE:::.initParCSLSE()` (or `causalSLSE:::.initParSLSE()` for `slseFit` objects). The function returns a list of four elements: `treated`, `nontreated`, `common`, `legend`. The first two are lists of two elements: `points` for the list of parameters of the scatterplot produced when `addPoints=TRUE` and `lines` for the line parameters. For example, we can see that the type of points for the treated is initially set to `pch=21` and their colour to 2:

```
causalSLSE:::.initParCSLSE()$treated$points
```

```
$pch
[1] 21

$col
[1] 2
```

The element `common` is for parameters not specific to a group like the main title or the axis labels, and `legend` are the parameters that control the legend (for `cslseFit` only). Note, however, that the colour and line shapes for the legend are automatically determined by the lines and points parameters of the `treated` and `nontreated` elements.

The default parameters can be modified by the argument `graphPar`. This argument must follow the structure of `causalSLSE:::.initParCSLSE()` (or `causalSLSE:::.initParSLSE()` for `slseFit` objects). For example, if we want a new title, new x-axis label, new type of lines for the treated, new type of points for the nontreated and a different position for the legend, we create the following `graphPar`:

```
graphPar <- list(treated = list(lines = list(lty=5, col=4)),
                 nontreated = list(points = list(pch=25, col=3)),
                 common = list(xlab = "MyNewLab", main="My New Title"),
                 legend = list(x = "top"))
```

In the following, we illustrate some examples.

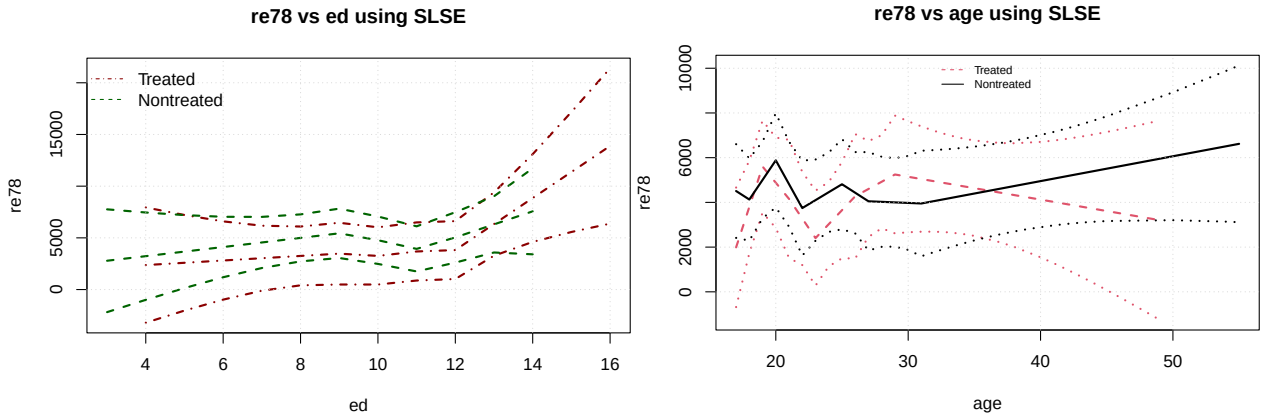
Example 1:

Consider the model:

```
model1 <- cslseModel(re78 ~ treat | ~ age + re75 + ed + married, data = nsw)
fit1 <- estSLSE(model1)
```

Suppose we want to compare the predicted income between the two treatment groups with respect to age or education, holding the other variables fixed to their group means (the default). The following are two examples with some of the default arguments modified. Note that `vcov.lm` is used in the first plot function and `vcovHC` (the default) of type HC1 in the second plot.

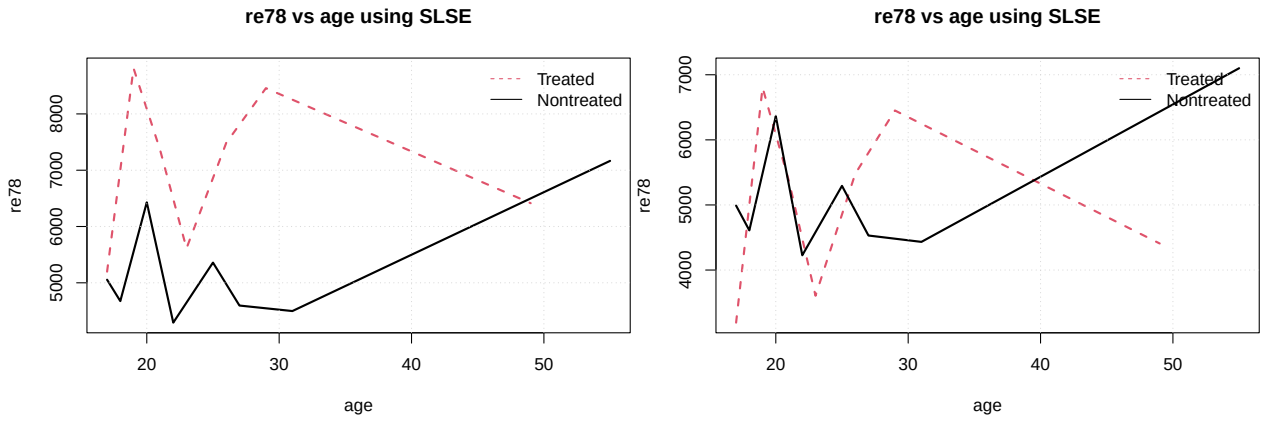
```
library(sandwich)
arg1 <- list(treated = list(lines = list(col = "darkred", lty = 4)),
             nontreated = list(lines = list(col = "darkgreen", lty = 2)),
             legend = list(x = "topleft"))
arg2 <- list(legend = list(x = "top", cex=0.7))
plot(fit1, "ed", vcov. = vcov, graphPar=arg1, interval = 'confidence')
plot(fit1, "age", interval = 'confidence', level = 0.9, type = "HC1", graphPar=arg2)
```



Example 2:

If we want to fix the other confounders using another function, we can change the argument `FUN`. The new function must be a function of one argument. For example, if we want to fix the other confounders to their group medians, we set `FUN` to `median` (no quotes). We proceed the same way for any function that requires only one argument. If the function requires more than one argument, we have to create a new function. For example, if we want to fix them to their group 20% empirical quantiles, we can set the argument to `function(x) quantile(x, .20)`. The following illustrates the two cases:

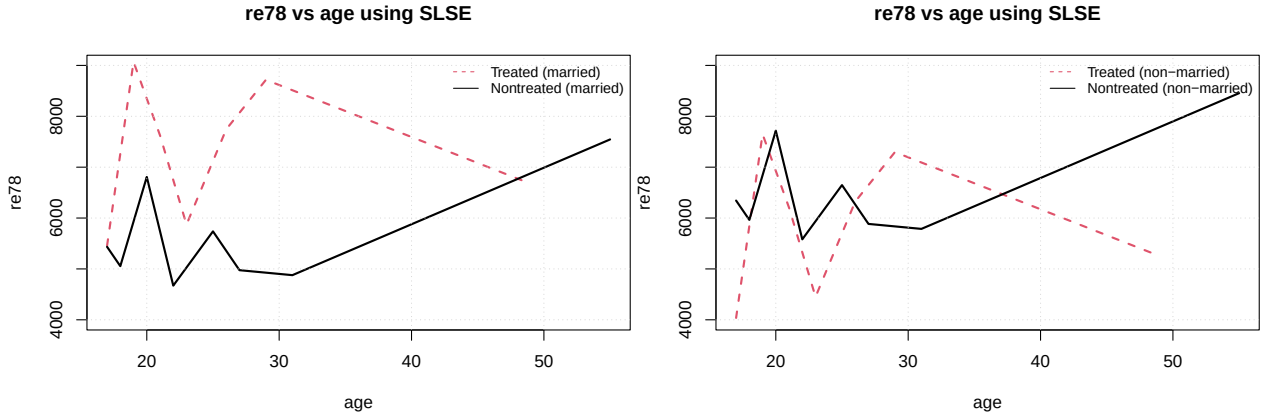
```
plot(fit1, "age", FUN = median)
plot(fit1, "age", FUN = function(x) quantile(x, 0.20))
```



Example 3:

It is also possible to set some of the other confounders to a specific value by changing the argument `fixedCov`. To fix some variables to the same values for both groups, `fixedCov` must be a named list with the names corresponding to the variables you want to fix. You can also add a description to the legend with the argument `addToLegend`. In the following `re75` is fixed at 10,000 and we compare the predicted outcome for the married individuals (the left graph) with the non-married ones (the right graph)

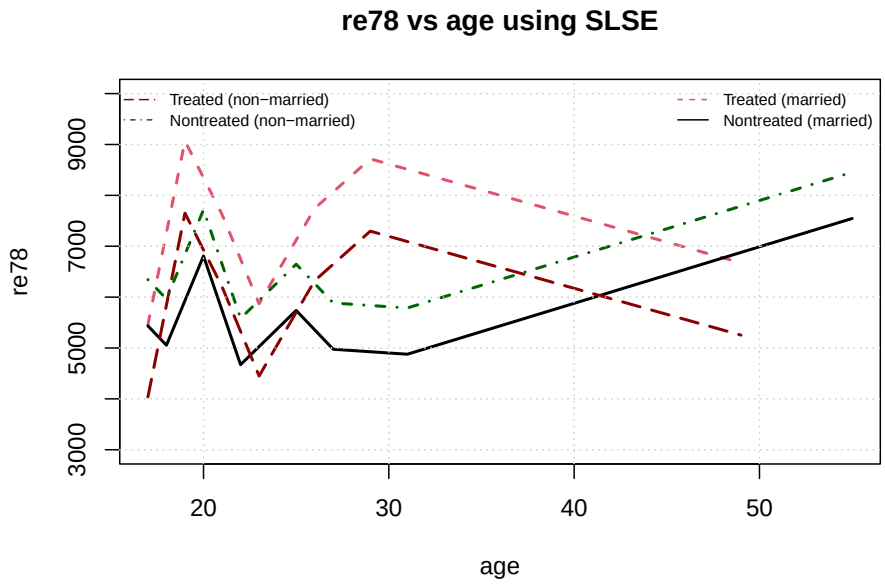
```
arg2 <- list(legend = list(cex = 0.8), common=list(ylim=c(4000,9000)))
plot(fit1, "age", fixedCov = list(married = 1, re75 = 10000),
     addToLegend = "married", graphPar = arg2)
plot(fit1, "age", fixedCov = list(married = 0, re75 = 10000),
     addToLegend = "non-married", graphPar = arg2)
```



Example 4:

To better compare the two groups, it is also possible to have them plotted on the same graph by setting the argument `add.` to `TRUE`. We just need to adjust some of the arguments to better distinguish the different curves. In the following example, we set the colors and line shapes to different values and change the position of the legend for the second set of lines.

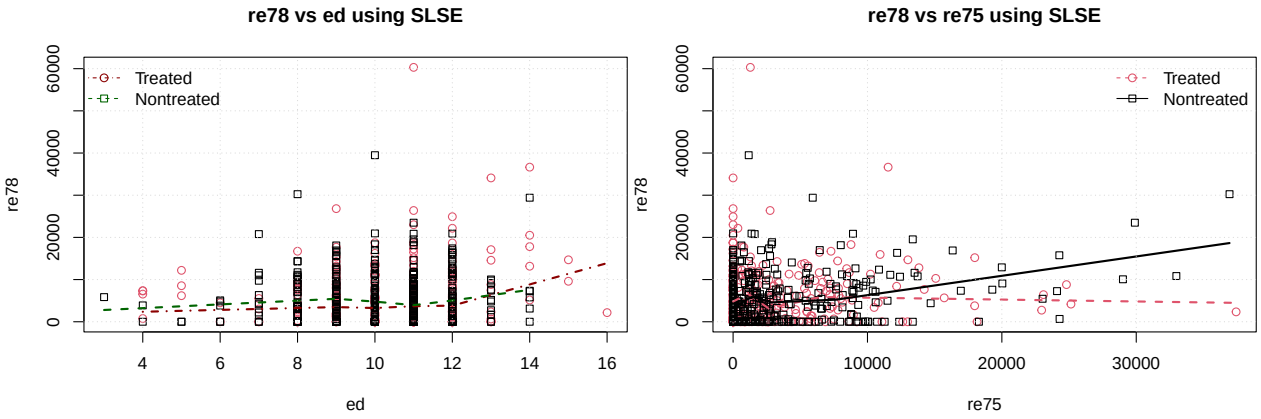
```
arg3 <- list(legend = list(cex = 0.7),
             common = list(ylim = c(3000, 10000)))
plot(fit1, "age", fixedCov = list(married = 1, re75 = 10000),
     addToLegend = "married", graphPar = arg3)
arg4 <- list(treated = list(lines = list(col = "darkred", lty = 5)),
             nontreated = list(lines = list(col = "darkgreen", lty = 4)),
             legend = list(x = "topleft", cex = 0.7))
plot(fit1, "age", fixedCov = list(married = 0, re75 = 10000),
     addToLegend = "non-married", add = TRUE, graphPar = arg4)
```



Example 5:

Finally, it is also possible to add the actual observations to the graph.

```
arg5 <- list(treated = list(lines = list(col = "darkred", lty = 4)),
             nontreated = list(lines = list(col = "darkgreen", lty = 2)),
             legend = list(x = "topleft"))
plot(fit1, "ed", addPoints = TRUE, graphPar = arg5)
plot(fit1, "re75", addPoints = TRUE)
```



2.2.5 Factors, interactions and functions of confounders

The package allows some of the confounders to be factors, functions of other confounders or interactions. For example, the dataset `simDat4` includes one factor, `X2`, with levels equal to "first", "second" and "third". We can include this confounder directly to the list of confounders. For example,

```
data(simDat4)
mod <- cslseModel(Y ~ Z | ~ X1 + X2 + X4, data = simDat4)
mod
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  246
Number of  nontreated :  254
Selection method: Default

Confounders approximated by SLSE (num. of knots):
```

```
    treated: X1(4)
    nontreated: X1(4)
Confounders not approximated by SLSE:
    treated: X2second, X2third, X4
    nontreated: X2second, X2third, X4
```

We see that R has created 2 binary variables, one for `X2="second"` and one for `X2="third"`. These two variables are automatically included in the group of confounders not approximated by SLSE because they are binary variables like `X4`. If we want to plot `Y` against `X1`, the binary variables `X2second`, `X2third` and `X4` are fixed to their group averages which, in case of binary variables, represent the proportions of ones in each group.

For interaction terms or functions of confounders, `FUN` is applied to the functions of confounders. This is how we have to proceed to obtain the average prediction in regression models. For example, if we interact `X2` and `X4`, we obtain:

```
data(simDat4)
mod <- cslseModel(Y ~ Z | ~ X1 + X2 * X4, data = simDat4)
mod
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  246
Number of  nontreated :  254
Selection method: Default

Confounders approximated by SLSE (num. of knots):
    treated: X1(4)
    nontreated: X1(4)
Confounders not approximated by SLSE:
    treated: X2second, X2third, X4, X2second:X4, X2third:X4
    nontreated: X2second, X2third, X4, X2second:X4, X2third:X4
```

In this case, when `FUN=mean`, `X2second:X4` is replaced by the proportion of ones in `X2second`×`X4` for each group. It is not replaced by the proportion of ones in `X2second` times the proportion of ones in `X4`. The same applies to functions of confounders. For functions of confounders, which can be defined in the formula using a built-in function like `log` or using the identity function `I()` (e.g. we can interact `X1` and `X4` by using `I(X1*X4)`), `FUN` is applied to the function (e.g. the average `log(X)` or the average `I(X1*X4)`).

To fix a factor to a specific level, we just set its value in the `fixedCov`. In the following example, we fix `X2` to "first", so `X2second` and `X2third` are set to 0.

```
fit <- estSLSE(mod)
plot(fit, "X1", fixedCov = list(X2 = "first"))
```

Note that if a function of confounders (or an interaction) involves the confounder we want to plot the outcome against, we factorize the confounder out, apply `FUN` to the remaining of the function and add the confounder back. For example, if we interact `X1` with `X4` and `FUN=mean`, `X1:X4` is replaced by `X1` times the proportion of ones in `X4` for each group.

2.3 Optimal selection of the knots

We have implemented two methods for selecting the knots: the backward semiparametric LSE (BLSE) and the forward semiparametric LSE (FLSE) methods. For each method, we have 3 criteria: the p-value threshold (PVT), the Akaike Information criterion (AIC), and the Bayesian Information criterion (BIC). Note that the consistency of the causal effect estimators has only been proved for the last two criteria in Giurcanu et al. (2023). The two selection methods can be summarized as follows:

We first compute one p-value per knot using either the BLSE or FLSE method:

BLSE:

1. We estimate the model with all knots included in the model.

2. For each knot, we test if the slopes of the basis functions adjacent to the knot are the same, and return the p-value.

FLSE:

1. We estimate the model by including a subset of the knots, one variable at the time. When we test a knot for one confounder, the number of knots is set to 0 for all other variables.
2. For each knot, we test if the adjacent slopes to the knot are the same, and return the p-value. The set of knots used for each test depends on the following:
   - Variables with 1 knot: we return the p-value of the test of equality of the slopes adjacent to the knot.
   - Variables with 2 knots: we include the two knots and return the p-values of the test of equality of the slopes adjacent to each knot.
   - Variables with p knots ($p > 2$): We test the equality of the slopes adjacent to knot i, for $i = 1, ..., p$, using the sets of knots $\{1, 2\}$, $\{1, 2, 3\}$, $\{2, 3, 4\}$, ..., $\{p-2, p-1, p\}$ and $\{p-1, p\}$ respectively.

Once we have the p-values, we proceed to step 3:

3. The knots are selected using one of the following criteria:
   - **PVT**: We remove all knots with a p-value greater than a specified threshold.
   - **AIC** or **BIC**: We order the p-values in ascending order. Then, starting with a model with no knots and going from the smallest to the highest p-value, we add the knot associated with the smallest remaining p-value one by one, estimate the model and return the information criterion. We select the model with the smallest value of the information citerion.

Note that the SLSE models for the treatment groups contained in `cslseModel` objects are estimated separately. However, the AIC and BIC are computed as if the they were estimated jointly as in Equation (2). As we will see below, a joint selection does not necessarily correspond to a selection done group by group.

The knot selection is done using the `selSLSE` method. The arguments are:

- **model**: An object of class `cslseModel`. The method also exists for `slseModel` objects. We will discuss it briefly at the end of this section.
- **selType**: This is the selection method. We have the choice between "FLSE" and "BLSE" (the default).
- **selCrit**: This is the criterion used by the selection method. We have the choice between "AIC" (the default), "BIC" or "PVT".
- **pvalT**: This is a function that returns the p-value threshold. It is a function of one argument, the average number of basis functions per confounder. The default is `function(p) 1/log(p)` and it is applied to each group separately. Therefore, the threshold may be different for the treated and non-treated. It is also possible to set it to a fixed threshold. For example, `function(p) 0.20` sets the threshold to 0.2. Note that when the function returns a value greater than 1, all knots are kept. This argument affects the result only when `selCrit` is set to "PVT".
- **vcovType**: The type of LSE covariance matrix used to compute the p-values. The options are "HC0" (the default), "HC1", "HC2", "HC3" and "Classical" (for the homoskedastic case). Using a heteroskedasticity robust covariance matrix is recommended, but there is not need to choose the usually recommended HC3 for the selection. The reason is that HC3 requires the hat values, which slows down the process, especially for FLSE, and simulations from Giurcanu et al. (2023) show that the choice of HC has very little effect on the selection. Note that the model is estimated separately for the treated

and nontreated. Therefore, we assume a different variance of the residuals even when `vcovType` is set to "Classical".

- **reSelect**: Should we recompute the optimal knots or use the ones already saved in the object. See below for more details.

The function returns a model of class `cslseModel` with the optimal selection of knots. For example, we can compare the starting knots of the following model, with the ones selected by the default arguments.

```
model1 <- cslseModel(re78 ~ treat | ~ age + re75 + ed + married, data = nsw)
model1
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method: Default

Confounders approximated by SLSE (num. of knots):
    treated: age(5), re75(3), ed(4)
    nontreated: age(6), re75(4), ed(4)
Confounders not approximated by SLSE:
    treated: married
    nontreated: married
```

```
model2 <- selSLSE(model1)
model2
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method: BLSE-AIC

Confounders approximated by SLSE (num. of knots):
    treated: age(2), re75(3), ed(1)
    nontreated: age(3), re75(2), ed(2)
Confounders not approximated by SLSE:
    treated: married
    nontreated: married
```

For example, the BLSE-AIC method has kept all knots from `re75` for the treated and kept two knots for the nontreated. The print method indicates which method was used to select the knots. It is possible to recover the p-values of all original knots by setting the argument `which` to `Pvalue`.

```
print(model2, which="Pvalues")
```

```
treated
*******
Covariates with no knots:
    married

Covariates with knots:
age :
          16.67%  33.33%     50%  66.67%  83.33%
Knots   19.00000 21.0000 23.0000 26.0000 29.0000
P-Value  0.03702  0.9019  0.1562  0.7867  0.5827

re75 :
              50%    66.67%    83.33%
Knots   1117.4390 2657.0570 6.511e+03
P-Value    0.2717    0.1169 8.143e-02

ed :
        16.67%  33.33%     50%  83.33%
Knots   9.0000 10.0000 11.0000 12.0000
P-Value 0.7064  0.7125  0.8924  0.2377
```

```
nontreated
**********
Covariates with no knots:
    married

Covariates with knots:
age :
         14.29%   28.57%   42.86%  57.14%  71.43%  85.71%
Knots   18.0000 20.00000 22.00000 25.0000 27.0000 31.0000
P-Value  0.3565  0.04433  0.08817  0.4204  0.6747  0.7247

re75 :
          42.86%    57.14%    71.43%    85.71%
Knots   240.1067 1405.5120 2856.2870 7666.8750
P-Value   0.6175    0.2553    0.4132    0.3843

ed :
        14.29%  42.86%  57.14%  85.71%
Knots   9.0000 10.0000 11.0000 12.0000
P-Value 0.2687  0.9006  0.1372  0.9393
```

In the following example, we use BLSE as selection method and BIC as criterion. Note that the BIC selects 0 knots for all confounders.

```
model3 <- selSLSE(model1, selType = "BLSE", selCrit = "BIC")
model3
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method: BLSE-BIC

Confounders approximated by SLSE (num. of knots):
    treated: None
    nontreated: None
Confounders not approximated by SLSE:
    treated: age, re75, ed, married
    nontreated: age, re75, ed, married
```

Since the `selSLSE` method returns a new model, we can apply the `estSLSE` to it:

```
estSLSE(selSLSE(model1, selType = "FLSE", selCrit = "BIC"))
```

```
Causal Semiparametric LSE
*************************
Selection method: FLSE-BIC

treated
*******
(Intercept)        U.age       U.re75         U.ed    U.married
 -388.96789     41.05403      0.02676    484.91610   1417.29125

nontreated
**********
(Intercept)        U.age       U.re75         U.ed    U.married
  4825.8776     -20.1057       0.2982       2.5002   -1094.0844
```

2.3.1 Selection for `slseModel` versus `cslseModel` objects

As mentioned in the previous section, the information criteria for `cslseModel` objects are computed as if the SLSE models of the treatment groups were estimated using Equation (2). This approach may lead to a selection different from what we would obtain by selecting the knots group by group. To see this, consider the following model and joint selection based on BLSE-AIC:

```
model1 <- cslseModel(re78 ~ treat | ~ age + re75 + ed + married, data = nsw)
model2 <- selSLSE(model1, selType="BLSE", selCrit="AIC")
model2
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method: BLSE-AIC

Confounders approximated by SLSE (num. of knots):
    treated: age(2), re75(3), ed(1)
    nontreated: age(3), re75(2), ed(2)
Confounders not approximated by SLSE:
    treated: married
    nontreated: married
```

We could also apply the same selection method, but group by group. In the following, we show how to select the knots group by group by using the `selSLSE` method for `slseModel` objects. Note that the Selection method is set to `BLSE-AIC (Sep.)` to indicate that it was done separately.

```
model3 <- model1
model3$treated <- selSLSE(model3$treated, selType="BLSE", selCrit="AIC")
model3$nontreated <- selSLSE(model3$nontreated, selType="BLSE", selCrit="AIC")
model3
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method: BLSE-AIC (Sep.)

Confounders approximated by SLSE (num. of knots):
    treated: age(3), re75(3), ed(1)
    nontreated: None
Confounders not approximated by SLSE:
    treated: married
    nontreated: age, re75, ed, married
```

We can see that the selected knots are quite different. For example, the number of knots of `age` for the treated is 2 when selected jointly and 3 when selected separately. Also, the number of knots of all confounders for the nontreated are set to 0 when the selection is done separately. This is very different from the joint selection. Since the joint selection is the one developed and studied by Giurcanu et al. (2023), it is the one we recommend.

Note that the PVT approach leads to identical selection whether it is done separately or jointly. The reason is that both approaches produce identical p-values, and the thresholds depend on the number of knots in each group.

2.3.2 Selections saved in `slseModel` objects.

Optimal selection of knots can be time consuming, especially for large sample sizes. To avoid having to recompute the selection each time we change the selection method and/or the criterion, every new selection is saved in the `slseModel` object. Let's consider the following model:

```
model <- cslseModel(re78 ~ treat | ~ age + re75 + ed + married, data = nsw)
model
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method: Default
```

```
Confounders approximated by SLSE (num. of knots):
    treated: age(5), re75(3), ed(4)
    nontreated: age(6), re75(4), ed(4)
Confounders not approximated by SLSE:
    treated: married
    nontreated: married
```

Suppose we select the knots using BLSE and AIC. We can replace the object by the new one.

```
model <- selSLSE(model, selType="BLSE", selCrit="AIC")
```

The main source of computing time for the selection comes from the number of regressions we need to estimate. Once estimated, all tests are simple operations, so the proportion of time allocated to testing the knots is negligible. Once we have the p-values, no additional regressions are needed for the `PVT` criterion, but we need one regression for each p-value for the AIC and BIC (plus 2 if we count the model with no knots). It is therefore worth saving the selection somewhere. This information is saved separately in each `slseModel` object under `selections`.

```
names(model$treated$selections)
```

```
[1] "originalKnots" "BLSE"
```

```
names(model$treated$selections)
```

```
[1] "originalKnots" "BLSE"
```

We see that the model keeps the original knots, so we loose no information by replacing the model object with the new one. The other element of the list is `BLSE` which contains information about the selection. If we want to compare `BLSE` with `FLSE`, we can call the `selSLSE` once more and replace `model` with the new one:

```
model <- selSLSE(model, selType="FLSE", selCrit="AIC")
names(model$treated$selections)
```

```
[1] "originalKnots" "BLSE"          "FLSE"
```

```
names(model$treated$selections)
```

```
[1] "originalKnots" "BLSE"          "FLSE"
```

The new selections are added to the model without deleting the previous ones. The following is what we can find in the element `BLSE` (or `FLSE`) of a given group:

```
names(model$treated$selections$BLSE)
```

```
[1] "pval"      "PVT"       "Threshold" "JAIC"      "JBIC"      "JIC"
```

The `pval` element is an object of class `pvalSLSE`. If we print it, we obtain the same output as when we print the model with the argument `which="Pvalues"`. The element `Threshold` is the p-value threshold used for the `PVT` criterion, `JIC` is a matrix of BIC and AIC values, ordered from the smallest to the highest p-value, and the selections are saved in the elements `PVT`, `JAIC` and `JBIC`. The `J` in front of IC, AIC and BIC means that the selection is based on the joint estimation of the SLSE models. There is no `J` in front of `PVT` because the selection using this criterion is identical if we proceed jointly or separately. Note that we do not see the `J` when we print the object, because it is assumed that AIC and BIC are obtained jointly for `cslseModel` objects. If we select the knots of one of the SLSE model separately, we will see the difference when we print the model. For example, the following replace the SLSE model of the treated with the SLSE model selected by AIC (separately):

```
model$treated <- selSLSE(model$treated, "BLSE", "AIC")
model
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method for the treated: BLSE-AIC (Sep.)
Selection method for the nontreated: FLSE-AIC
```

```
Confounders approximated by SLSE (num. of knots):
    treated: age(3), re75(3), ed(1)
    nontreated: age(2), re75(2), ed(2)
Confounders not approximated by SLSE:
    treated: married
    nontreated: married
```

The print output indicates that the selection was done jointly for the nontreated (just `AIC`) and separately for the treated. Now, the SLSE model for the treated has three more selection elements: `IC`, `AIC` and `BIC`:

```
names(model$treated$selections$BLSE)
```

```
[1] "pval"      "PVT"       "Threshold" "JAIC"      "JBIC"      "JIC"
[7] "AIC"       "BIC"       "IC"
```

With all selections saved in the model, how do we know which knots are used? We know it by printing the model or by printing the following knots attribute:

```
attr(model$nontreated$knots, "curSel")
```

```
$select
[1] "FLSE"

$crit
[1] "JAIC"
```

It tells us that the current set of knots for that model is BLSE with a joint AIC.

The elements `PVT`, `JAIC`, `JBIC`, `AIC` and `BIC` are lists of knots selection vectors in the same format as the argument `selKnots` of the `estSLSE` method. For example, the following is the list of selection vectors using BLSE with AIC for the treated:

```
model$treated$selections$BLSE$JAIC
```

```
$age
[1] 1 3

$re75
[1] 1 2 3

$ed
[1] 4

$married
NULL
```

For example, we see that the knots 1 and 3 of `age` are selected by AIC. We could use this selection list to estimate the model:

```
estSLSE(model$treated, model$treated$selections$BLSE$JAIC)
```

```
Semiparametric LSE
******************
Selection method: Manual selection


(Intercept)      U.age_1      U.age_2      U.age_3     U.re75_1     U.re75_2
 -2.691e+04    1.675e+03   -4.419e+02    6.016e+01    1.635e+00   -2.184e+00
   U.re75_3     U.re75_4       U.ed_1       U.ed_2    U.married
  6.541e-01   -4.615e-02    1.643e+02    2.516e+03    1.405e+03
```

Note that we never explicitly selected the BIC criterion above, but it was added to the model anyway. The reason is that the cost of computing the selection based on BIC is negligible once we do it for AIC. Since it is also negligible to add the selection by `PVT`, if the `selCrit` argument is set to `"AIC"` or `"BIC"`, all three selections are computed simultaneously and stored in the model. It is only when `selCrit` is set to `"PVT"` that the selection is done for this criterion only.

2.3.3 Select a saved selection with `update`

Now that we understand that all previous selections are saved in `slseModel` objects, how do we select them? This is done with the `update` method. The method is registered for `slseModel` and `cslseModel` objects and works very similarly. The arguments are `selType`, `selCrit` and `selKnots`. The latter is used to select the knots manually as explained in Section 2.2.2. The first two are as in the `selSLSE` method. The purpose of `update` is to replace the current knots by a selection saved in the model. If the selection we want is not in the model, `update` will return an error message. For example, the object `model` from the previous section has all selections saved, but the current is a mixture of FLSE-AIC and BLSE-AIC (Sep.).

```
model
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method for the treated: BLSE-AIC (Sep.)
Selection method for the nontreated: FLSE-AIC

Confounders approximated by SLSE (num. of knots):
    treated: age(3), re75(3), ed(1)
    nontreated: age(2), re75(2), ed(2)
Confounders not approximated by SLSE:
    treated: married
    nontreated: married
```

We can update the model to the FLSE with joint AIC selection as follows (AIC means joint AIC for `cslseModel`)

```
model <- update(model, selType="FLSE", selCrit="AIC")
model
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method: FLSE-AIC

Confounders approximated by SLSE (num. of knots):
    treated: age(3), re75(1), ed(1)
    nontreated: age(2), re75(2), ed(2)
Confounders not approximated by SLSE:
    treated: married
    nontreated: married
```

This is done without any computation. The `update` method simply replaces the knots using the appropriate list of selection vectors. If we want to recover the model with the initial set of knots, we simply set the argument `selType` to `"None"`. In the following, we see that the selection method is back to its initial set of knots:

```
update(model, selType="None")
```

```
Causal Semiparametric LSE Model
*******************************

Number of  treated :  297
Number of  nontreated :  425
Selection method: Default

Confounders approximated by SLSE (num. of knots):
    treated: age(5), re75(3), ed(4)
    nontreated: age(6), re75(4), ed(4)
Confounders not approximated by SLSE:
    treated: married
    nontreated: married
```

As a last application of the `update` method, the following shows that the joint AIC and the one applied to each group separately produces different results. Since we just set the `model` object to FLSE-AIC, the current treated model in `model` is based on the joint AIC criterion:

```
model$treated
```

```
Semiparametric LSE Model
************************

Number of observations:  297
Selection method: FLSE-JAIC

Covariates approximated by SLSE (num. of knots):
    age(3), re75(1), ed(1)
Covariates not approximated by SLSE:
    married
```

Since we have the AIC computed separately in the treated model, we can use `update` for `slseModel` to compare the two:

```
update(model$treated, "BLSE", "AIC")
```

```
Semiparametric LSE Model
************************

Number of observations:  297
Selection method: BLSE-AIC

Covariates approximated by SLSE (num. of knots):
    age(3), re75(3), ed(1)
Covariates not approximated by SLSE:
    married
```

The selection for `age` and `ed` are the same, but the AIC selects 3 knots for `re75` and the JAIC selects only 1 knot.

Note that the `selSLSE` method computes the selection only if the requested selection is not saved in the model, unless the argument `reSelect` is set to `TRUE`. Therefore, `selSLSE` is not different from `update` when the requested selection is saved in the model.

2.4 The `causalSLSE` method for `cslseFit` objects

The method `causalSLSE` estimates the causal effects from `cslseFit` objects using the knots included in the estimated model. The arguments of the method are:

- **object**: An object of class `cslseFit`.
- **causal**: What causal effect measure should the function compute? We have the choice between "ALL" (the default), "ACE", "ACT" or "ACN".
- **vcov.**: An alternative function used to compute the covariance matrix of the least squares estimates. This is the $\hat{\Sigma}_{\hat{\theta}}$ defined in the Introduction section. By default, `vcovHC` is used with `type="HC3"`. Simulations from Giurcanu et al. (2023) show that using `vcovHC` with `type="HC3"` produces the most accurate estimate of the variance of ACE, ACT and ACN in small and large samples.
- `...`: This is used to pass arguments to the `vcov.` function.

In the following example, we estimate the causal effect with the initial knots (without selection).

```
model1 <- cslseModel(re78 ~ treat | ~ age + re75 + ed + married, data=nsw)
fit1 <- estSLSE(model1)
ce <- causalSLSE(fit1)
ce
```

```
Causal Effect using Semiparametric LSE
**************************************
Selection method: Default
```

```
ACE = 825.4
ACT = 843.7
ACN = 812.6
```

The method returns an object of class `cslse` and its `print` method only prints the causal effect estimates. We can extract any causal estimate and its standard error by using the $ operator followed by the type of causal effect estimate. For example, the following is the ACE:

```
ce$ACE
```

```
     est       se
825.4222 505.7461
```

The object also contains the two `slseFit` objects and the model attributes that specify the name of the treatment indicator variable and the value of the indicator associated with each group.

For more details about the estimation, which includes standard errors and significance tests, we can use the `summary` method:

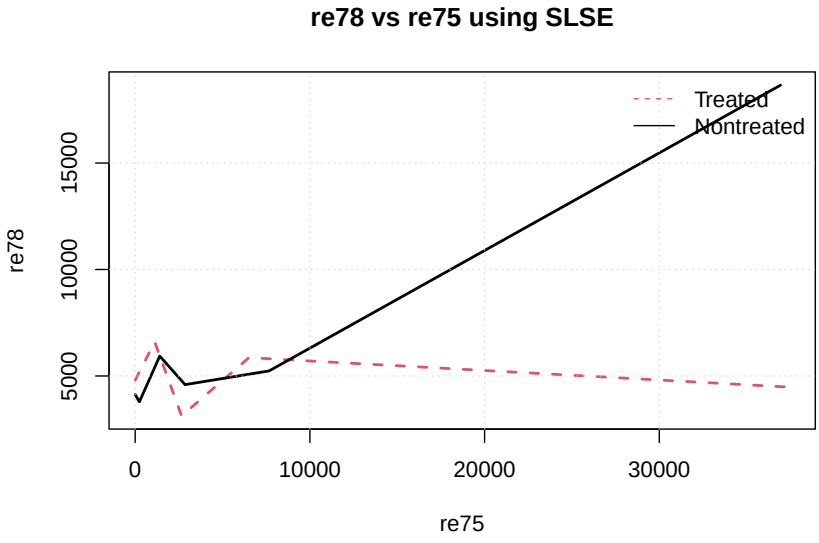
```
sce <- summary(ce)
sce
```

```
Causal Effect using Semiparametric LSE
**************************************
Selection method: Default

    Estimate Std. Error t value Pr(>|t|)
ACE    825.4      505.7   1.632    0.103
ACT    843.7      527.9   1.598    0.110
ACN    812.6      513.7   1.582    0.114
```

The `summary` method returns an object of class `summary.cslse` and the above output is produced by its `print` method. If needed, we can extract the above table using `$causal`. The summary tables for the treated and nontreated LSE can be extracted using `$lse`.

The `cslse` object inherits from the class `cslseFit`, so we can apply the `plot` (or the `predict`) method directly on this object as shown below:

```
plot(ce, "re75")
```



2.4.1 The `extract` method

The package also comes with an `extract` method for objects of class `cslse`. For example, we can compare different methods in a single table. In the following example, we compare the SLSE, BLSE-AIC and

FLSE-AIC:

```
library(texreg)
c1 <- causalSLSE(fit1)
fit2 <- estSLSE(selSLSE(model1, selType="BLSE"))
fit3 <- estSLSE(selSLSE(model1, selType="FLSE"))
c2 <- causalSLSE(fit2)
c3 <- causalSLSE(fit3)
texreg(list(SLSEC=c1, BLSE_AIC=c2, FLSE_AIC=c3), table=FALSE, digits=4)
```

	SLSE	BLSE	FLSE
ACE	825.4222	785.8421	845.4417
	(505.7461)	(483.5174)	(496.1856)
ACT	843.7084	843.3745	855.8981
	(527.8792)	(516.7354)	(513.6188)
ACN	812.6434	745.6371	838.1346
	(513.6616)	(478.0473)	(501.8498)
Num. knots (Nontreated)	14	7	6
Num. knots (Treated)	12	6	5
Num. confounders	4	4	4
Num. obs. (Nontreated)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.0925	0.0855	0.0839
Adj. R^2	0.0462	0.0567	0.0578

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

The arguments of the `extract` methods, which control what is printed and can be modified through the `texreg` function, are:

- **include.nobs**: Include the number of observations. The default is `TRUE`.
- **include.nknots**: Include the number of knots. The default is `TRUE`.
- **include.rsquared**: Include the R^2. The default is `TRUE`.
- **include.adjrs**: Include the adjusted R^2. The default is `TRUE`.
- **separated.rsquared** Should we return one R^2 per `slseModel`? By default it is set to `FALSE` and one R^2 is computed for the joint estimation of Equation (3). This argument applies also to the adjusted R^2.
- **which**: Which causal effects should be printed? The options are "ALL" (the default), "ACE", "ACT", "ACN", "ACE-ACT", "ACE-ACN" or "ACT-ACN".

Here is one example on how to change some arguments:

```
texreg(list(SLSE=c1, BLSE=c2, FLSE=c3), table=FALSE, digits=3,
       which="ACE-ACT", include.adjrs=FALSE, separated.rsquared=TRUE)
```

	SLSE	BLSE	FLSE
ACE	825.422	785.842	845.442
	(505.746)	(483.517)	(496.186)
ACT	843.708	843.374	855.898
	(527.879)	(516.735)	(513.619)
Num. knots (Nontreated)	14	7	6
Num. knots (Treated)	12	6	5
Num. confounders	4	4	4
Num. obs. (Nontreated)	425	425	425
Num. obs. (Treated)	297	297	297
R^2 (nontreated)	0.102	0.098	0.097
R^2 (treated)	0.074	0.064	0.063

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

2.5 The `causalSLSE` method for `cslseModel` objects

When applied directly to `cslseModel` objects, the `causalSLSE` method offers the possibility to select the knots and estimate the causal effects all at once. The method also returns an object of class `cslse`. The arguments are the same as for the method for `cslseFit` objects, plus the necessary arguments for the knots

selection. The following are the arguments not already defined for objects of class `cslseFit`. The details of these arguments are presented in Section 2.3.

- **object**: An object of class `cslseModel`.
- **selType**: This is the selection method. We have the choice between "SLSE" (the default), "FLSE" and "BLSE". The SLSE method performs no selection, so all knots from the model are kept.
- **selCrit**: This is the criterion used by the selection method when `selType` is set to "FLSE" or "BLSE". The default is "AIC".
- **pvalT**: This is a function that returns the p-value threshold. We explained this argument when we presented the `selSLSE` method.

For example, we can generate the previous table as follows:

```
c1 <- causalSLSE(model1, selType="SLSE")
c2 <- causalSLSE(model1, selType="BLSE")
c3 <- causalSLSE(model1, selType="FLSE")
texreg(list(SLSE=c1, BLSE=c2, FLSE=c3), table=FALSE, digits=4)
```

	SLSE	BLSE	FLSE
ACE	825.4222	785.8421	845.4417
	(505.7461)	(483.5174)	(496.1856)
ACT	843.7084	843.3745	855.8981
	(527.8792)	(516.7354)	(513.6188)
ACN	812.6434	745.6371	838.1346
	(513.6616)	(478.0473)	(501.8498)
Num. knots (Nontreated)	14	7	6
Num. knots (Treated)	12	6	5
Num. confounders	4	4	4
Num. obs. (Nontreated)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.0925	0.0855	0.0839
Adj. R^2	0.0462	0.0567	0.0578

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

Note that this `causalSLSE` method calls `selSLSE` for the knot selection. Therefore, the selection is done without any computation if the selection is already saved in the model object. It would therefore be inefficient to compare FLSE with AIC, BIC and PVT using the following, because it would involve recomputing all FLSE selections 3 times:

```
c1 <- causalSLSE(model1, selType="FLSE", selCrit="AIC")
c2 <- causalSLSE(model1, selType="FLSE", selCrit="BIC")
c3 <- causalSLSE(model1, selType="FLSE", selCrit="PVT")
```

It is better to add all `FLSE` selections in the model first and then call the `causalSLSE` method 3 times as follows:

```
model1 <- selSLSE(model1, selType="FLSE")
c1 <- causalSLSE(model1, selType="FLSE", selCrit="AIC")
c2 <- causalSLSE(model1, selType="FLSE", selCrit="BIC")
c3 <- causalSLSE(model1, selType="FLSE", selCrit="PVT")
```

2.6 The `causalSLSE` method for `formula` objects

This last method, offers an alternative way of estimating the causal effects. It allows the estimation in one step without having to first create a model. The arguments are the same as for the `cslseModel` function and the `causalSLSE` method for `cslseModel` objects. It creates the model, selects the knots and estimates the causal effects in one step. For example, we can create the previous table as follows:

```
c1 <- causalSLSE(re78 ~ treat | ~ age + re75 + ed + married, data=nsw,
                 selType="SLSE")
c2 <- causalSLSE(re78 ~ treat | ~ age + re75 + ed + married, data=nsw,
                 selType="BLSE")
c3 <- causalSLSE(re78 ~ treat | ~ age + re75 + ed + married, data=nsw,
```

```
                selType="FLSE")
texreg(list(SLSE=c1, BLSE=c2, FLSE=c3), table=FALSE, digits=4)
```

	SLSE	BLSE	FLSE
ACE	845.4417	785.8421	845.4417
	(496.1856)	(483.5174)	(496.1856)
ACT	855.8981	843.3745	855.8981
	(513.6188)	(516.7354)	(513.6188)
ACN	838.1346	745.6371	838.1346
	(501.8498)	(478.0473)	(501.8498)
Num. knots (Nontreated)	6	7	6
Num. knots (Treated)	5	6	5
Num. confounders	4	4	4
Num. obs. (Nontreated)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.0839	0.0855	0.0839
Adj. R^2	0.0578	0.0567	0.0578

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

Note that this method calls `cslseModel`, `selSLSE`, `estSLSE` and the method `causalSLSE` for `cslseFit` objects sequentially. It is easier to simply work with this method, but manually going through all steps may be beneficial to better understand the procedure. Also, it is more convenient to work with a model when we want to compare the different selection methods, or if we want to compare estimations with different types of standard errors. In particular, this approach does not offer the more efficient option of computing all selections once and save them in the model as explained at the end of the previous section.

3 Examples

3.1 A simulated data set from Model 1

In the package, the data set `datSim1` is generated using the following data generating process with a sample size of 300.

$$\begin{aligned} Y(0) &= 1 + X + X^2 + \epsilon(0) \\ Y(1) &= 1 - 2X + \epsilon(1) \\ Z &= \text{Bernoulli}[\Lambda(1 + X)] \\ Y &= Y(1)Z + Y(0)(1 - Z) \end{aligned}$$

where $Y(0)$ and $Y(1)$ are the potential outcomes, X, $\epsilon(0)$ and $\epsilon(1)$ are independent standard normal and $\Lambda(x)$ is the CDF of the standard logistic distribution. The causal effects ACE, ACT and ACN are approximately equal to -1, -1.6903 and 0.5867 (estimated using a sample size of 10^7). We can start by building the starting model:

```
data(simDat1)
mod <- cslseModel(Y ~ Z | ~ X, data = simDat1)
mod <- selSLSE(mod, "BLSE") ## Let's save them all first
mod <- selSLSE(mod, "FLSE")
```

Then we can compare three different methods:

```
c1 <- causalSLSE(mod, selType = "SLSE")
c2 <- causalSLSE(mod, selType = "BLSE", selCrit = "BIC")
c3 <- causalSLSE(mod, selType = "FLSE", selCrit = "BIC")
texreg(list(SLSE = c1, BLSE = c2, FLSE = c3), table = FALSE, digits = 4)
```

	SLSE	BLSE	FLSE
ACE	-1.5067^{***}	-1.4950^{***}	-1.4950^{***}
	(0.2706)	(0.2696)	(0.2696)
ACT	-1.9889^{***}	-1.9889^{***}	-1.9889^{***}
	(0.3138)	(0.3137)	(0.3137)
ACN	-0.1804	-0.1369	-0.1369
	(0.3174)	(0.3260)	(0.3260)
Num. knots (Nontreated)	2	2	2
Num. knots (Treated)	1	0	0
Num. confounders	1	1	1
Num. obs. (Nontreated)	80	80	80
Num. obs. (Treated)	220	220	220
R^2	0.7466	0.7431	0.7431
Adj. R^2	0.7414	0.7388	0.7388

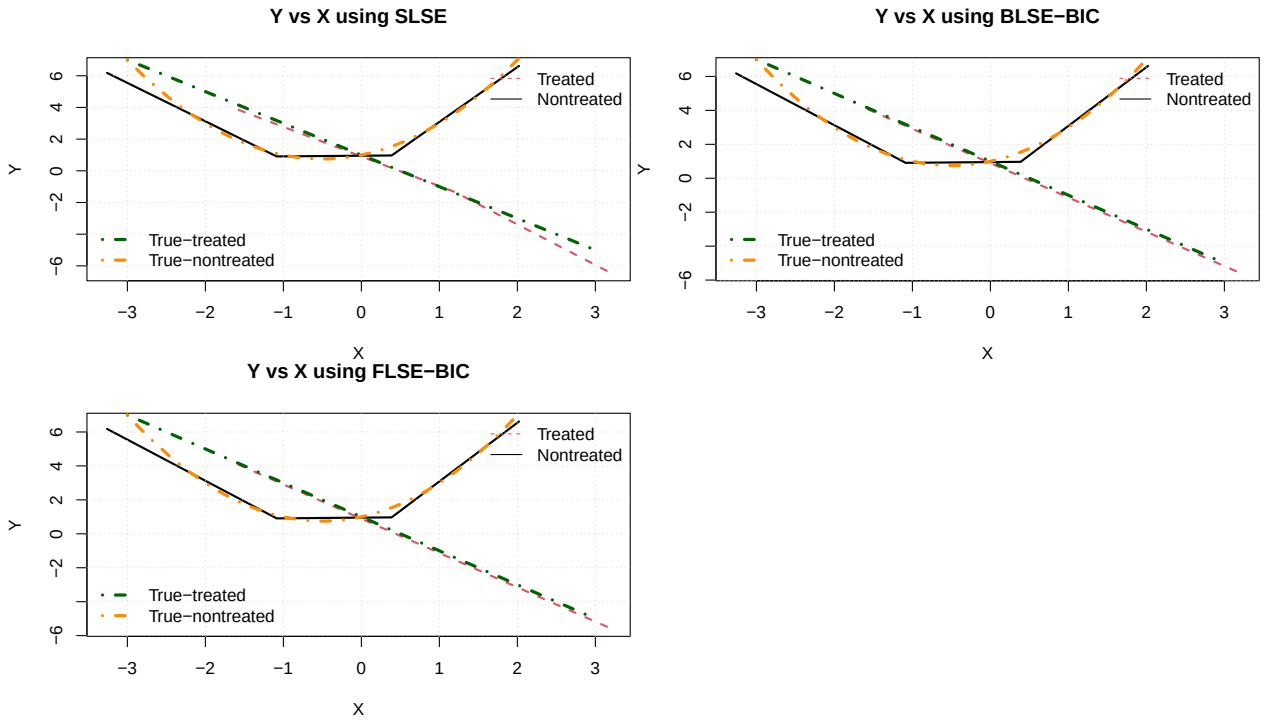
$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

We see that both selection methods choose to assign 0 knots for the treated group, which is not surprising since the true $f_1(x)$ is linear. We can compare the different fits.

```
list(common = list(main = "Y vs X using BLSE-BIC"))
```

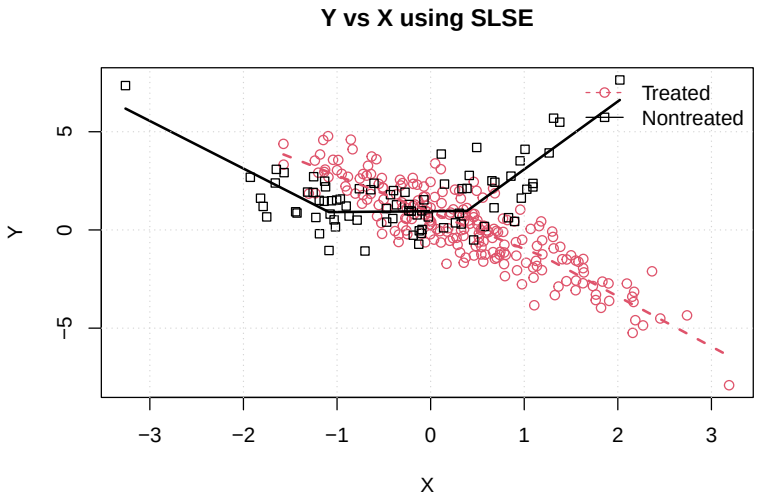
```
$common
$common$main
[1] "Y vs X using BLSE-BIC"
```

```
plot(c1, "X")
curve(1 - 2 * x, -3, 3, col = "darkgreen", lty = 4, lwd = 3, add = TRUE)
curve(1 + x + x^2, -3, 3, col = "darkorange", lty = 4, lwd = 3, add = TRUE)
legend("bottomleft", c("True-treated", "True-nontreated"),
       col=c("darkgreen", "darkorange"), lty = 4, lwd = 3, bty = 'n')
plot(c2, "X", graphPar = list(common = list(main = "Y vs X using BLSE-BIC")))
curve(1 - 2 * x, -3, 3, col="darkgreen", lty = 4, lwd = 3, add = TRUE)
curve(1 + x + x^2, -3, 3, col = "darkorange", lty = 4, lwd = 3, add = TRUE)
legend("bottomleft", c("True-treated", "True-nontreated"),
       col = c("darkgreen", "darkorange"), lty = 4, lwd = 3, bty = 'n')
plot(c3, "X", graphPar = list(common = list(main = "Y vs X using FLSE-BIC")))
curve(1 - 2 * x, -3, 3, col="darkgreen", lty = 4, lwd = 3, add = TRUE)
curve(1 + x + x^2, -3, 3, col = "darkorange", lty = 4, lwd = 3, add = TRUE)
legend("bottomleft", c("True-treated", "True-nontreated"),
       col = c("darkgreen", "darkorange"), lty = 4, lwd = 3, bty = 'n')
```



We see that the piecewise polynomials are very close to the true $f_0(x)$ and $f_1(x)$ for SLSE, BLSE and FLSE. We can see from the folllowing graph how the lines are fit through the observations by group.

```
plot(c1, "X", addPoints=TRUE)
```



3.2 A simulated data set from Model 2

The dataset `datSim2` is a change point regression model (with unknown location of change points) defined as follows:

$$\begin{aligned} Y(0) &= (1+X)I(X \leq -1) + (-1-X)I(X > -1) + \epsilon(0) \\ Y(1) &= (1-2X)I(X \leq 0) + (1+2X)I(X > 0) + \epsilon(1) \\ Z &= \text{Bernoulli}[\Lambda(1+X)] \\ Y &= Y(1)Z + Y(0)(1-Z) \end{aligned}$$

where $Y(0)$ and $Y(1)$ are the potential outcomes, $I(A)$ is the indicator function equal to 1 if A is true, and X, $\epsilon(0)$ and $\epsilon(1)$ are independent standard normal. The causal effects ACE, ACT and ACN are approximately equal to 3.763, 3.858 and 3.545 (estimated with a sample size of 10^7). We can compare the SLSE, BLSE-AIC and BLSE-BIC.

```
data(simDat2)
mod <- cslseModel(Y~Z | ~X, data=simDat2)
mod <- selSLSE(mod, "BLSE") ## We just add BLSE because we do not use FLSE
```

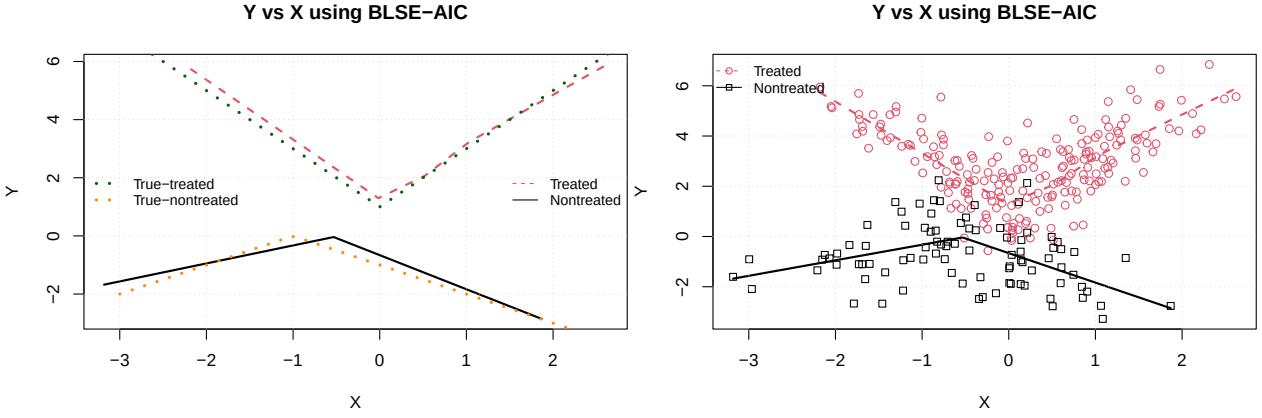
```
c1 <- causalSLSE(mod, selType = "SLSE")
c2 <- causalSLSE(mod, selType = "BLSE", selCrit = "BIC")
c3 <- causalSLSE(mod, selType = "BLSE", selCrit = "AIC")
texreg(list(SLSE = c1, BLSE.BIC = c2, BLSE.AIC = c3), table = FALSE, digits = 4)
```

	SLSE	BLSE.BIC	BLSE.AIC
ACE	3.9268***	3.9268***	3.9268***
	(0.1765)	(0.1765)	(0.1765)
ACT	3.9566***	3.9566***	3.9566***
	(0.1992)	(0.1992)	(0.1992)
ACN	3.8560***	3.8560***	3.8560***
	(0.2263)	(0.2263)	(0.2263)
Num. knots (Nontreated)	1	1	1
Num. knots (Treated)	3	3	3
Num. confounders	1	1	1
Num. obs. (Nontreated)	89	89	89
Num. obs. (Treated)	211	211	211
R^2	0.7827	0.7827	0.7827
Adj. R^2	0.7775	0.7775	0.7775

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

The following shows the fit of BLSE-AIC with the true $f_1(x)$ and $f_0(x)$, and the observations.

```
arg <-  list(common = list(main = "Y vs X using BLSE-AIC"),
             legend = list(x = "right", cex = 0.8))
plot(c2, "X", graphPar = arg)
curve((1 -2 * x) * (x <= 0) + (1 + 2 * x) * (x > 0), -3, 3,
      col = "darkgreen", lty = 3, lwd = 3, add = TRUE)
curve((1 + x) * (x <= -1) + (-1 - x) * (x > -1),
      -3, 3, col = "darkorange", lty = 3, lwd = 3, add = TRUE)
legend("left", c("True-treated", "True-nontreated"),
       col = c("darkgreen", "darkorange"), lty = 3, lwd = 3, bty = 'n', cex = .8)
arg$legend$x <- "topleft"
plot(c2, "X", addPoints = TRUE, graphPar = arg)
```



3.3 A simulated data set from Model 3

The data set `datSim3` is generated from model with multiple confounders defined as follows:

$$
\begin{aligned}
Y(0) &= [1 + X_1 + X_1^2] + [(1 + X_2)I(X_2 \le -1) + (-1 - X_2)I(X_2 > -1)] + \epsilon(0) \\
Y(1) &= [1 - 2X_1] + [(1 - 2X_2)I(X_2 \le 0) + (1 + 2X_2)I(X_2 > 0)] + \epsilon(1) \\
Z &= \text{Bernoulli}[\Lambda(1 + X_1 + X_2)] \\
Y &= Y(1)Z + Y(0)(1 - Z)\,,
\end{aligned}
$$

where $Y(0)$ and $Y(1)$ are the potential outcomes, X_1, X_2, $\epsilon(0)$ and $\epsilon(1)$ are independent standard normal. The causal effects ACE, ACT and ACN are approximately equal to 2.762, 2.204 and 3.922 (estimated with a sample size of 10^7). We can compare the SLSE, FLSE with AIC and FLSE with BIC.

```
data(simDat3)
mod <- cslseModel(Y ~ Z | ~ X1 + X2, data = simDat3)
mod <- selSLSE(mod, "FLSE") ## We just add FLSE because we do not use BLSE
```

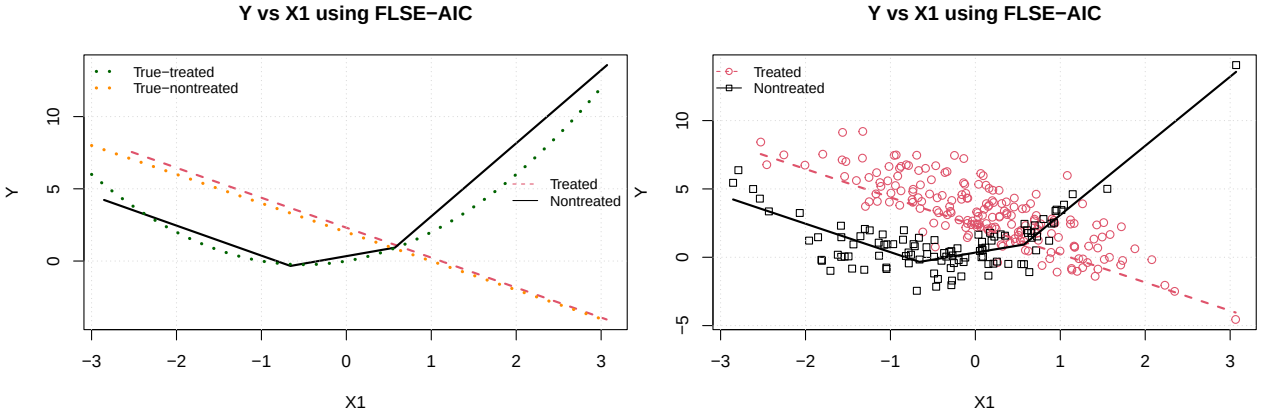
```
c1 <- causalSLSE(mod, selType = "SLSE")
c2 <- causalSLSE(mod, selType = "FLSE", selCrit = "BIC")
c3 <- causalSLSE(mod, selType = "FLSE", selCrit = "AIC")
texreg(list(SLSE = c1, FLSE.BIC = c2, FLSE.AIC = c3), table = FALSE, digits = 4)
```

	SLSE	FLSE.BIC	FLSE.AIC
ACE	2.4685***	2.4810***	2.4685***
	(0.2727)	(0.2671)	(0.2727)
ACT	2.0634***	2.0554***	2.0634***
	(0.3442)	(0.3354)	(0.3442)
ACN	3.2319***	3.2832***	3.2319***
	(0.3465)	(0.3454)	(0.3465)
Num. knots (Nontreated)	8	4	8
Num. knots (Treated)	6	2	6
Num. confounders	2	2	2
Num. obs. (Nontreated)	104	104	104
Num. obs. (Treated)	196	196	196
R^2	0.8743	0.8662	0.8743
Adj. R^2	0.8658	0.8611	0.8658

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

To illustrate the method, since we have two confounders, we need to plot the outcome against one confounder holding the other fixed. The default is to fix it to its sample mean. For the true curve, we fix it to its population mean, which is 0. We first look at the outcome against X_1. By fixing X_2 to 0, the true curve is $X_1 + X_1^2$ for the untreated and $2 - 2X_1$ for the treated. The following graphs show how the FLSE-BIC method fits the curves.

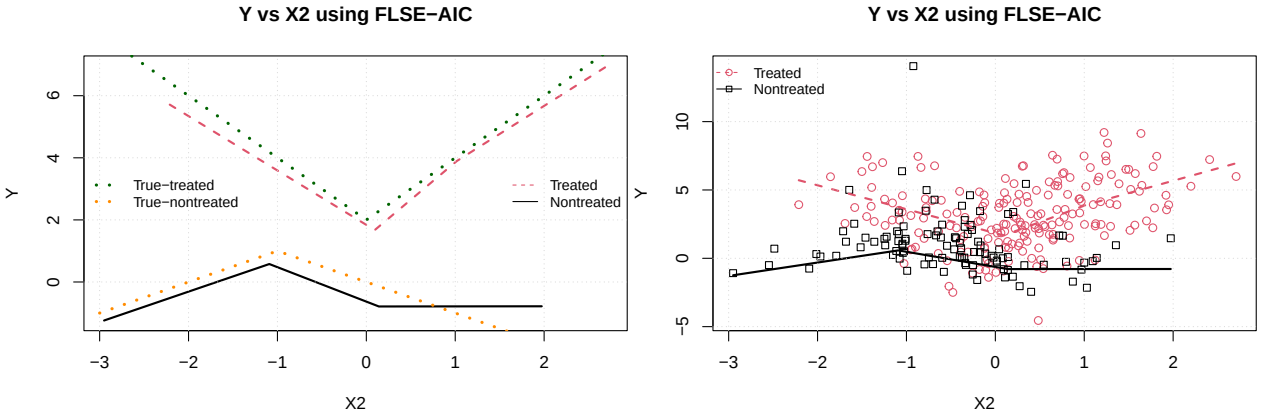
```
arg <-  list(common = list(main = "Y vs X1 using FLSE-AIC"),
             legend = list(x = "right", cex = 0.8))
plot(c2, "X1", graphPar = arg)
curve(x + x^2, -3, 3, col = "darkgreen", lty = 3, lwd = 3, add = TRUE)
curve(2 - 2 * x, -3, 3, col = "darkorange", lty = 3, lwd = 3, add = TRUE)
legend("topleft", c("True-treated", "True-nontreated"),
       col = c("darkgreen", "darkorange"), lty = 3, lwd = 3, bty = 'n', cex = .8)
arg$legend$x <- "topleft"
plot(c2, "X1", addPoints = TRUE, graphPar = arg)
```



If we fix X_1 to 0, the true curve is $1 + [(1 + X_2)I(X_2 \leq -1) + (-1 - X_2)I(X_2 > -1)]$ for the nontreated and $1 + [(1 - 2X_2)I(X_2 \leq 0) + (1 + 2X_2)I(X_2 > 0)]$ for the treated. The following graphs illustrates how these curves are approximated by FLSE-AIC.

```
arg <-  list(common = list(main = "Y vs X2 using FLSE-AIC"),
             legend = list(x = "right", cex = 0.8))
plot(c2, "X2", graphPar = arg)
curve(1 + (1 - 2 * x) * (x <= 0) + (1 + 2 * x) * (x > 0), -3, 3,
      col = "darkgreen", lty = 3, lwd = 3, add = TRUE)
curve(1 + (1 + x) * (x <= -1) + (-1 - x) * (x > -1),
      -3, 3, col = "darkorange", lty = 3, lwd = 3, add = TRUE)
legend("left", c("True-treated", "True-nontreated"),
       col = c("darkgreen", "darkorange"), lty = 3, lwd = 3, bty = 'n', cex = .8)
```

```
arg$legend$x <- "topleft"
plot(c2, "X2", addPoints = TRUE, graphPar = arg)
```



3.4 A simulated data set with product terms

The data set `datSim5` is generated using the following data generating process with a sample size of 300.

$$
\begin{aligned}
Y(0) &= [1 + X_1 + X_1^2] + [(1 + X_2)I(X_2 \leq -1) + (-1 - X_2)I(X_2 > -1)] \\
&\quad + [1 + X_1X_2 + (X_1X_2)^2] + \epsilon(0) \\
Y(1) &= [1 - 2X_1] + [(1 - 2X_2)I(X_2 \leq 0) + (1 + 2X_2)I(X_2 > 0)] \\
&\quad + [1 - 2X_1X_2] + \epsilon(1) \\
Z &= \text{Bernoulli}[\Lambda(1 + X_1 + X_2 + X_1X_2)] \\
Y &= Y(1)Z + Y(0)(1 - Z)\,,
\end{aligned}
$$

where $Y(0)$ and $Y(1)$ are the potential outcomes, and X_1, X_2, $\epsilon(0)$ and $\epsilon(1)$ are independent standard normal. The causal effects ACE, ACT and ACN are approximately equal to 1.763, 0.998 and 3.194 (estimated with a sample size of 10^7). We can compare the SLSE, FLSE-AIC and FLSE-BIC.

```
data(simDat5)
mod <- cslseModel(Y ~ Z | ~ X1 * X2, data = simDat5)
mod <- selSLSE(mod, "FLSE") ## We just add FLSE because we do not use BLSE
```

```
c1 <- causalSLSE(mod, selType = "SLSE")
c2 <- causalSLSE(mod, selType = "FLSE", selCrit = "BIC")
c3 <- causalSLSE(mod, selType = "FLSE", selCrit = "AIC")
texreg(list(SLSE = c1, FLSE.BIC = c2, FLSE.AIC = c3), table = FALSE, digits = 4)
```

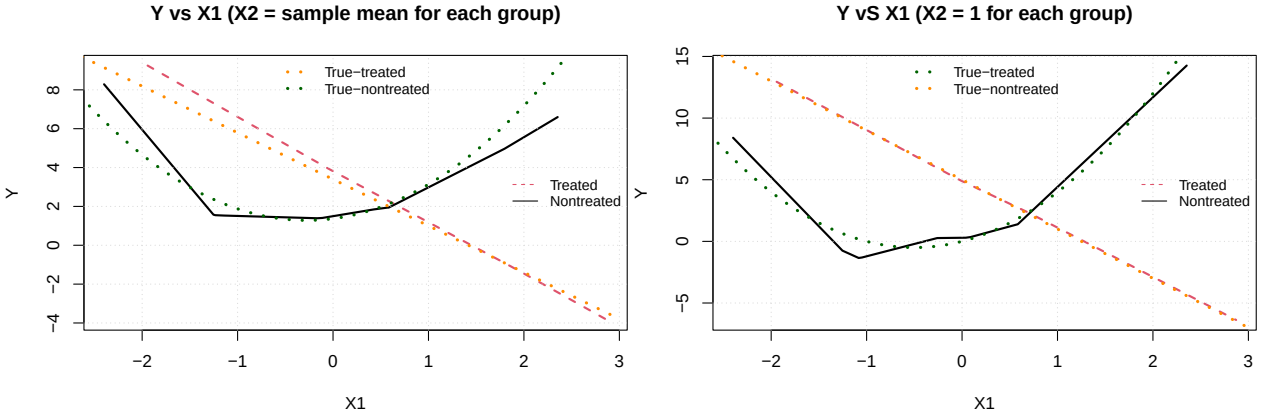
	SLSE	FLSE.BIC	FLSE.AIC
ACE	1.7951***	1.7142***	1.7951***
	(0.3812)	(0.3815)	(0.3812)
ACT	1.2321*	1.1083*	1.2321*
	(0.4950)	(0.4956)	(0.4950)
ACN	2.8560***	2.8560***	2.8560***
	(0.4479)	(0.4478)	(0.4479)
Num. knots (Nontreated)	9	7	9
Num. knots (Treated)	6	6	6
Num. confounders	3	3	3
Num. obs. (Nontreated)	104	104	104
Num. obs. (Treated)	196	196	196
R^2	0.8928	0.8898	0.8928
Adj. R^2	0.8843	0.8819	0.8843

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

In the case of multiple confounders with interactions, the shape of the fitted outcome with respect to one confounder depends on the value of the other confounders. Without interaction, changing the value of the

other confounders only shifts the fitted line without changing its shape. The following graphs compare the estimated relationship between Y and X_1 for X_2 equal to the group means (left graph) and 1 (right graph). Using a sample of 10^7, we obtain that $\mathrm{E}(X_2|Z=1)$ and $\mathrm{E}(X_2|Z=0)$ are approximately equal to 0.1982 and -0,3698, respectively. Therefore, the true curves are $(1.3698 + 0.6302x + 1.1368x^2)$ for the nontreated and $(3.3964 - 2.3964x)$ for the treated. If $X_2 = 1$, the true curves become $2x + 2x^2$ for the treated and $(5 - 4x)$ for the nontreated.

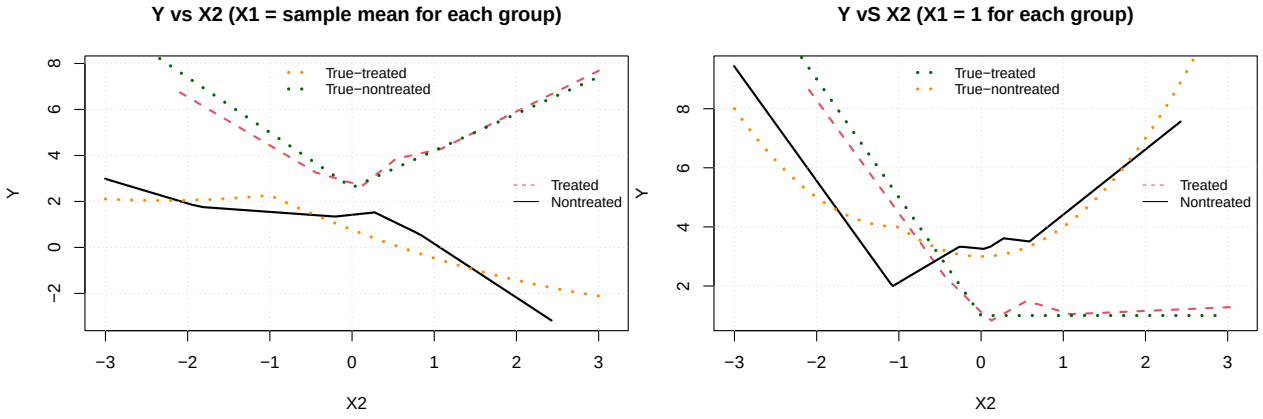
```
x20 <- mean(subset(simDat5, Z == 0)$X2)
x21 <- mean(subset(simDat5, Z == 1)$X2)
arg <-  list(common = list(main = "Y vs X1 (X2 = sample mean for each group)"),
             legend = list(x = "right", cex = 0.8))
plot(c2, "X1", fixedCov = list(nontreated = list(X2 = x20), treated = list(X2 = x21)),
     graphPar = arg)
curve(1.3698 + 0.6302 * x + 1.1368 * x^2, -3, 3,
      col = "darkgreen", lty = 3, lwd = 3, add = TRUE)
curve(3.3964 - 2.3964 * x, -3, 3, col = "darkorange", lty = 3, lwd = 3, add = TRUE)
legend("top", c("True-treated", "True-nontreated"),
       col=c("darkorange", "darkgreen"), lty = 3, lwd = 3, bty = 'n', cex = .8)
arg <-  list(common = list(main = "Y vS X1 (X2 = 1 for each group)"),
             legend = list(x = "right", cex = 0.8))
plot(c2, "X1", fixedCov = list(X2 = 1), graphPar = arg)
curve(2 * x + 2 * x^2, -3, 3, col = "darkgreen", lty = 3, lwd = 3, add = TRUE)
curve(5 - 4 * x, -3, 3, col = "darkorange", lty = 3, lwd = 3, add = TRUE)
legend("top", c("True-treated", "True-nontreated"),
       col = c("darkgreen", "darkorange"), lty = 3, lwd = 3, bty = 'n', cex = .8)
```



The following graphs illustrate the relationship between Y and X_2 for a given X_1. When X_1 is equal to its population group means (they are equal to the population means of X_2), the true curves are $[1.6036 - 0.3964x)(x \leq 0) + (1 + 2x)(x > 0)]$ for the treated and $[(1.767 - 0.3698x + 0.1368x^2) + (1 + x)(x \leq -1) + (-1 - x)(x > -1)]$ for the nontreated. If $X_1 = 1$, the true curves become $[-2x + (1 - 2x)(x \leq 0) + (1 + 2x)(x > 0)]$ for the treated and $[(4 + x + x^2) + (1 + x)(x \leq -1) + (-1 - x)(x > -1)]$ for the nontreated.

```
x10 <- mean(subset(simDat5, Z == 0)$X1)
x11 <- mean(subset(simDat5, Z == 1)$X1)
arg <-  list(common = list(main = "Y vs X2 (X1 = sample mean for each group)"),
             legend = list(x = "right", cex = 0.8))
plot(c2, "X2", fixedCov = list(nontreated = list(X1 = x10), treated = list(X1 = x11)),
     graphPar = arg)
curve(1.603900 - .3964 * x + (1 - 2 * x) * (x <= 0) + (1 + 2 * x) * (x > 0), -3, 3,
      col = "darkgreen", lty = 3, lwd = 3, add = TRUE)
curve(1.767 - 0.3698 * x + 0.1368 * x^2 + (1 + x) * (x <= -1) + (-1 - x) * (x > -1),
      -3, 3, col = "darkorange", lty = 3, lwd = 3, add = TRUE)
legend("top", c("True-treated", "True-nontreated"),
       col = c("darkorange", "darkgreen"), lty = 3, lwd = 3, bty = 'n', cex = .8)
arg$common$main <- "Y vS X2 (X1 = 1 for each group)"
plot(c2, "X2", fixedCov = list(X1 = 1), graphPar = arg)
curve(-2 * x + (1 - 2 * x) * (x <= 0) + (1 + 2 * x) * (x > 0), -3, 3,
      col = "darkgreen", lty = 3, lwd = 3, add = TRUE)
curve(4 + (1 + x) * (x <= -1) + (-1 - x) * (x > -1) + x + x^2,
      -3, 3, col = "darkorange", lty = 3, lwd = 3, add = TRUE)
```

```
legend("top", c("True-treated", "True-nontreated"),
       col = c("darkgreen", "darkorange"), lty = 3, lwd = 3, bty = 'n', cex = .8)
```



4 Summary of methods and objects

The following is a list of all objects from the package. For each object, we explain how it is constructed and give a list of the registered methods. For more details about the arguments of the different methods, see the help files. Note, however, that no help files exist for non-exported methods and the latter must be called using `causalSLSE:::` before the method names.

- **slseKnots**: The object is created by the function `slseKnots` and the only exported registered methods are `print` and `update`.
- **slseModel** and **cslseModel**: The objects are respectively created by the functions `slseModel` and `cslseModel`. The exported registered methods are `print`, `estSLSE` (estimate the regression model), `selSLSE` (optimal selection of knots), `update` (select knots from saved selections) and `causalSLSE` (to estimate the causal effects). There are two non-exported methods: `pvalSLSE` (used to compute the p-values) and `model.matrix` (to extract the matrix of confounders).
- **slseFit** and **cslseFit**: The objects are created by the method `estSLSE` (the former when applied to `slseModel` objects and the latter when applied to `cslseModel` objects) and the exported registered methods are `print`, `causalSLSE` (to compute the causal effects), `predict` (to predict the outcome), `plot` (to plot the outcome as a function of one confounder) and `summary` (to give more details about the least squares estimation). There is one non-exported method, `as.model`, which extracts the model object from the `slseFit` or `cslseFit` object.
- **slseFit**: One registered method that only applies to this class is `extract`. It is needed to generate LaTeX table with `texreg`.
- **cslseFit**: One registered method that only applies to this class is `as.list`. It converts the objects into a list of `slseFit` objects.
- **summary.slseFit** and **summary.cslseFit**: The objects are created by the `summary` method for `slseFit` and `cslseFit` objects. The only exported registered method is `print`.
- **cslse**: The object is created by any `causalSLSE` method. It inherits from `cslseFit` objects. The methods that are common through this inheritance are `plot` and `predict`. The exported registered methods specific to `cslse` objects are `print`, `summary` (to give more details about the causal effect estimation) and `extract` (a method needed for `texreg`). There is one non-exported method, `as.model`, which extracts the model object.

Note that the method `causalSLSE` is also registered for objects of class `formula`.

5 Experiments (to be removed before publishing)

```
data(simDat5)
mod <- cslseModel(Y ~ Z | ~ X1 * X2, data = simDat5)
getAlt <- function(which=c("ipw","matching"), ...)
{
    which <- match.arg(which)
    met <- c("ACE","ACT","ACN")
    l <- lapply(met, function(mi)
    {
        arg <- list(...)
        arg$type <- mi
        res <- do.call(which, arg)
    })
    if (length(l[[1]]$estim)==2)
    {
        l <- lapply(1:2, function(i) {
            obj <- lapply(l, function(li) {
                li$estim <- li$estim[i]
                li$se <- li$se[i]
                li})
```

```
            names(obj) <- met
            class(obj) <- "allAlt"
            obj})
        names(l) <- c("Matching", "BC_Matching")
    } else {
        names(l) <- met
        class(l) <- "allAlt"
    }
    l
}

setMethod("extract", signature = className("allAlt", package='causalSLSE'),
          definition = function (model, include.nobs = TRUE, ...)
          {
              se <- sapply(model, function(li) li$se)
              co <- sapply(model, function(li) li$estim)
              pval <- 2*pnorm(-abs(co/se))
              names(co) <- names(se) <- names(pval) <- names(model)
              gof <- numeric()
              gof.names <- character()
              gof.decimal <- logical()
              if (isTRUE(include.nobs)) {
                  n <- nrow(model[[1]]$data)
                  gof <- c(gof, n)
                  gof.names <- c(gof.names, "Num. obs.")
                  gof.decimal <- c(gof.decimal, FALSE)
              }
              tr <- createTexreg(coef.names = names(co), coef = co, se = se,
                                 pvalues = pval, gof.names = gof.names,
                                 gof = gof, gof.decimal = gof.decimal)
              return(tr)
          })

res <- getAlt("ipw", form=Y~Z, psForm=Z~X1*X2, data=simDat5, normalized=TRUE)
res2 <- getAlt("matching", form=Y~Z, psForm=Z~X1*X2, data=simDat5, bcForm=~X1*X2,
               balm=~X1*X2)
texreg(list(IPW=res, Matching=res2[[1]], BC.Matching=res2[[2]]))
```

```
\begin{table}
\begin{center}
\begin{tabular}{l c c c}
\hline
 & IPW & Matching & BC.Matching \\
\hline
ACE       & $2.23^{***}$ & $2.26^{***}$ & $2.16^{***}$ \\
          & $(0.34)$     & $(0.35)$     & $(0.35)$     \\
ACT       & $2.14^{***}$ & $2.07^{***}$ & $1.96^{***}$ \\
          & $(0.36)$     & $(0.44)$     & $(0.48)$     \\
ACN       & $2.51^{***}$ & $2.62^{***}$ & $2.54^{***}$ \\
          & $(0.55)$     & $(0.38)$     & $(0.44)$     \\
\hline
Num. obs. & $300$        & $300$        & $300$        \\
\hline
\multicolumn{4}{l}{\scriptsize{$^{***}p<0.001$; $^{**}p<0.01$; $^{*}p<0.05$}}
\end{tabular}
\caption{Statistical models}
\label{table:coefficients}
\end{center}
\end{table}
```

References

Giurcanu, M., M. Capanu, P. Chaussé, and G. Luta. 2023. "Efficient Semiparametric Inference for Causal Effects." *Working Paper*.

Lalonde, R. 1986. "Evaluating the Econometric Evaluations of Training Programs." *American Economic*

Review 76: 604–20.

Leifeld, Philip. 2013. "texreg: Conversion of Statistical Model Output in R to LaTeX and HTML Tables." *Journal of Statistical Software* 55 (8): 1–24. http://dx.doi.org/10.18637/jss.v055.i08.

Zeileis, Achim. 2006. "Object-Oriented Computation of Sandwich Estimators." *Journal of Statistical Software* 16 (9): 1–16. https://doi.org/10.18637/jss.v016.i09.