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 will present the package in the following sections.

The general semiparametric additive regression model is

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

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:

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

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:

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

*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 estimates 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 non-treated (ACN) are defined respectively as follows:

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

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 estimates. Let $\hat{\theta} = \{\hat{\beta}_0, \hat{\beta}_1, \hat{\psi}_0^T, \hat{\psi}_1^T\}^T$. Then, the causal effect estimates 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, 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):

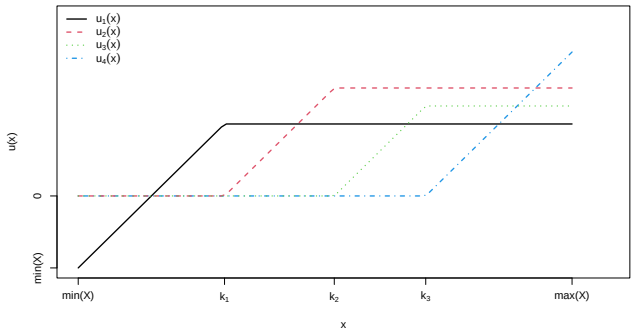
$$\hat{V}_{\text{ACE}} = \begin{pmatrix} -\hat{\beta}_0 & \hat{\beta}_1 & \hat{D}_{\text{ACE}}^T \end{pmatrix} \begin{pmatrix} \hat{\Sigma}_0 & \hat{\Sigma}_{0,1} & \hat{\Sigma}_{0,\hat{\theta}} \\ \hat{\Sigma}_{1,0} & \hat{\Sigma}_1 & \hat{\Sigma}_{1,\hat{\theta}} \\ \hat{\Sigma}_{\hat{\theta},0} &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

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 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:

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

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 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.

2 The `causalSLSE` package

2.1 The starting knots

By default, the knots for each variable are based on the following procedure applied separately for the treated and nontreated. The term `sample size` refers either to the number of observations in the treated or nontreated group.

1. The starting number of knots is a function of the sample size and is determined by the argument `nbasis`, a function of one argument, the sample size. The floor value of what the function returns is the number of basis functions. The starting number of knots is therefore equal to the `floor` of what the function returns minus 1 (or 0 if the function returns a value strictly less than 1). The default function is `function(n) n^0.3`. For example, if the total sample size is 500, with 200 treated and 300 nontreated, the starting number of knots in the treated and nontreated groups are respectively equal to 3=`floor(200^0.3)-1` and 4=`floor(300^0.3)-1`, respectively. 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 $p+1$ 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 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 dataset, called `nsw`, contains some continuous and categorical variables, so we can illustrate how knots are selected initially. The dataset is included in the `causalSLSE` package.

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

The outcome is the real income in 1978 (`re78`) and the purpose is to estimate the causal effect of a training program (`treat`) on `re78`. The dataset includes the continuous variables age (`age`), education (`ed`), the 1975 real income (`re75`), and binary variables (`black`, `hisp`, `married` and `nodeg`).

The process of selecting knots is the same for the treated and nontreated. In fact, the knots are created by calling the `slseKnots` function for each group. This function is not restricted to causal models. We could use it to generate knots for any SLSE regression model. But, estimating such models is not yet implemented in the package. The arguments of the function are:

- **form**: A formula with the right-hand side being the list of confounders. If a left-hand side is provided, the `slseKnots` function will ignore it.
- **data**: A `data.frame` containing all variables included in the formula.
- **X**: Instead of providing the function with a formula, we can input directly the matrix of confounders.
- **nbasis**: A function that determines the number of basis functions as explained above.
- **knots**: This argument is used to set the knots manually. We will explain how to use this argument in the next section.

We start by considering the confounders `age`, `re75`, `ed`, and `married`. To create a set of knots for a specific group using `slseKnots`, we can restrict the sample using the `subset` function. For example, we can create the initial knots for the treated and nontreated as follows:

```
k1 <- slseKnots(form = ~ age + re75 + ed + married,
                data = subset(nsw, treat==1))
k0 <- slseKnots(form = ~ age + re75 + ed + married,
                data = subset(nsw, treat==0))
```

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 for the treated:

```
k1
```

```
Covariates with no knots:
    married

Covariates with knots:
age :
      20% 40% 60% 80%
Knots  19  22  25  28

re75 :
        40%  60%  80%
Knots 357.9 1962 5589

ed :
      20% 40% 60% 80%
Knots   9  10  11  12
```

The sample size for the treated is 297. Given the default `nbasis`, it implies a number of starting knots equal to 4=`floor`$(297^{0.3})$-1. This is the number of knots we have for `ed` and `age`. However, the number of knots for `re75` is 3. The reason is that `re75` contains a large fraction of zeros. Since the 20^{th} percentile is equal to 0 and 0 is also the minimum value of `ed75`, it is dropped. This can be seen as follows (the `type` argument of the `quantile` function is the same as it is implemented in the package):

```
quantile(nsw[nsw$treat==1,'re75'], c(.2,.4,.6,.8), type=1)
```

```
      20%       40%       60%       80%
   0.0000  357.9499 1961.8640 5588.6640
```

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:

```
k1$age
```

```
20% 40% 60% 80%
 19  22  25  28
```

Note that the confounders 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`:

```
k1$married
```

```
NULL
```

Alternatively, we can create a list of `slseKnots` objects, one for the treated and one for the nontreated, using the `cslseKnots` function. This approach of generating the knots for each group will allow us to easily extend the package to models with multiple treatments. The function has the same arguments as `slseKnots`, but `form` must include a formula linking the outcome and the treatment indication 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 the list of confounders, which are the same as in the previous example::

```
k <- cslseKnots(form = re78 ~ treat | ~ age + re75 + ed + married,
                data = nsw)
```

The function returns an object of class `cslseKnots`. We can use its `print` method to print the knots, in which case the knots are printed for both groups:

```
k
```

```
treated
*******

Covariates with no knots:
    married

Covariates with knots:
age :
      20% 40% 60% 80%
Knots  19  22  25  28

re75 :
        40%  60%  80%
Knots 357.9 1962 5589

ed :
      20% 40% 60% 80%
Knots   9  10  11  12

nontreated
**********

Covariates with no knots:
    married

Covariates with knots:
age :
      16.67% 33.33% 50% 66.67% 83.33%
Knots     18     20  23     26     30

re75 :
        50% 66.67% 83.33%
Knots 823.3   2292   6567

ed :
      16.67% 33.33% 66.67% 83.33%
Knots      9     10     11     12
```

Note that `cslseKnots` objects are lists of `slseKnots` objects. Therefore, we can print the set of knots for a specific group using $ followed by its label (treated or nontreated). For example, we can print the set of knots for the treated by typing `k$treated`.

2.2 Creating a causal SLSE model

In general, users will not call `cslseKnots` explicitly. This function is called by the `cslseModel` function. The function returns an object of class `cslseModel` (or a causal SLSE model), which contains the information

about the outcome (Y), the treatment indicator (Z), the confounders (X) and their knots ($\kappa_{l,k}$). The arguments of the function are the same as for the `cslseKnots` function. Therefore, we can create a model with the same set of knots as in the previous section as follows:

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

Its print method summarizes the characteristics of the model. In particular, it indicates which confounders have a positive number of knots and which ones have no knots.

```
model1
```

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

Number of treated:  297
Number of nontreated:  425
Selection method: Default
Confounders approximated by SLSE:
    treated: age, re75, ed
    nontreated: age, re75, ed
Confounders not approximated by SLSE:
    treated: married
    nontreated: married
```

Note that the selection method is set to `Default`. We refer to this method when the knots are automatically selected by the method described in the previous section. Later in the document, we will present methods for optimally selecting a subset of `Default`.

The element `knots` of `model1` is the object of class `cslseKnots` created by the `cslseKnots` function, so we can print the knots as follows:

```
model1$knots
```

```
treated
*******

Covariates with no knots:
    married

Covariates with knots:
age :
      20% 40% 60% 80%
Knots  19  22  25  28

re75 :
        40%  60%  80%
Knots 357.9 1962 5589

ed :
      20% 40% 60% 80%
Knots   9  10  11  12

nontreated
**********

Covariates with no knots:
    married

Covariates with knots:
age :
      16.67% 33.33% 50% 66.67% 83.33%
Knots     18     20  23     26     30

re75 :
        50% 66.67% 83.33%
Knots 823.3   2292   6567

ed :
```

```
      16.67% 33.33% 66.67% 83.33%
Knots      9     10     11     12
```

Alternatively, we can print the knots by setting the `print` method argument `which` to "selKnots". Also, all `print` methods offers the possibility of changing the number of digits. For example, the following would print the knots using 4 digits:

```
print(model1, which="selKnots", digits=4)
```

We have also included, in the package, the simulated dataset `simDat4`, which contains special types of confounders. 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)
model2 <- cslseModel(Y ~ Z | ~ X1 + X2 + X3 + X4, data = simDat4)
model2$knots$nontreated
```

```
Covariates with no knots:
    X2second, X2third, X4

Covariates with knots:
X1 :
         40%   60%   80%
Knots 0.2531 2.912 12.11

X3 :
      40%
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.

2.2.1 Setting 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 for both groups are set automatically as described above. One way to set the number of knots to 0 for all variables in both groups is to set the argument to `NULL`.

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

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

Number of treated:  297
Number of nontreated:  425
Selection method: User Based
Confounders approximated by SLSE:
    treated: None
    nontreated: None
Confounders not approximated by SLSE:
    treated: age, re75, ed, married
    nontreated: age, re75, ed, married
```

If we want to set the number of knots to 0 for one group, we must specify the group using a named list. In the following, it is set to 0 for the treated only:

```
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: User Based
Confounders approximated by SLSE:
    treated: None
    nontreated: age, re75, ed
Confounders not approximated by SLSE:
    treated: age, re75, ed, married
    nontreated: 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 range of the variable for the group.

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

1. An unnamed list of length equal to the number of confounders. In that case, the knots must be defined in the same order of confounders implied by the formula.

   Suppose we want to set for the nontreated group an automatic selection for `age`, no knots for `ed`, the knots $\{1000, 5000, 10000\}$ for `re75`, and the knots to be automatically selected for the treated group. We proceed as follows. Note that setting the value to `NA` or `NULL` has the same effect for the binary variable `married`.

```
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")
```

```
Causal Semiparametric LSE Model: Selected knots
***********************************************

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

Covariates with no knots:
    married

Covariates with knots:
age :
      20% 40% 60% 80%
Knots  19  22  25  28

re75 :
        40%  60%  80%
Knots 357.9 1962 5589

ed :
      20% 40% 60% 80%
Knots   9  10  11  12

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

Covariates with no knots:
    ed, married

Covariates with knots:
```

```
age :
      16.67% 33.33% 50% 66.67% 83.33%
Knots     18     20  23     26     30

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

Note that the selection method is set to `Default` for the treated and `User Based` for the nontreated. This is because the knots were manually selected only for the nontreated. However, when we print the model, the selection method will be set to `User Based` whenever at least one of the sets of knots is `User Based` as in the following:

```
model

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

Number of treated:  297
Number of nontreated:  425
Selection method: User Based
Confounders approximated by SLSE:
    treated: age, re75, ed
    nontreated: age, re75
Confounders not approximated by SLSE:
    treated: married
    nontreated: ed, married
```

2. A named list of length equal to the number of confounders. In that case, the order of the list of variables does not matter. The `cslseModel` function will automatically reorder the variables to match the order implied by the formula. The names must match perfectly the confounder 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(nontreated=list(married = NA, ed = NULL, 'age:ed' = NULL,
                             re75 = c(1000,5000,10000), age = NA))
model <- cslseModel(re78 ~ treat | ~ age * ed + re75 + married, data = nsw,
                    knots = selK)
model$knots$nontreated

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

Covariates with knots:
age :
      16.67% 33.33% 50% 66.67% 83.33%
Knots     18     20  23     26     30

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

3. A named list of length strictly less than the number of confounders. The names of the selected confounders must match perfectly the names generated by R and the order does not matter. This is particularly useful when the number of confounders is large.

   If we consider the previous example, the knots are set manually only for `ed`, `ed:age` and `re75`. 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(nontreated=list(ed = NULL, 'age:ed' = NULL,
                             re75 = c(1000,5000,10000)))
model <- cslseModel(re78 ~ treat | ~ age * ed + re75 + married, data = nsw,
                    knots = selK)
model$knots$nontreated
```

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

Covariates with knots:
age :
      16.67% 33.33% 50% 66.67% 83.33%
Knots     18     20  23     26     30

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 confounders. 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()
selK$treated <- selK$nontreated <- list('age:ed' = NULL, 'ed:re75' = NULL,
                                        'ed:married' = NULL)
model <- cslseModel(re78 ~ treat | ~ age * ed + re75 * ed + married * ed,
                    data = nsw, knots = selK)
model
```

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

Number of treated:  297
Number of nontreated:  425
Selection method: User Based
Confounders approximated by SLSE:
    treated: age, ed, re75
    nontreated: age, ed, re75
Confounders not approximated by SLSE:
    treated: married, age:ed, ed:re75, ed:married
    nontreated: married, age:ed, ed:re75, ed:married
```

Note also that `cslseModel` deals with interaction terms as 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 <- cslseModel(re78 ~ treat | ~ age + ed * black, data = nsw)
model$knots$nontreated[["ed:black"]]
```

```
33.33%    50% 66.67%
     9     10     11
```

2.3 Estimating the model

Given the set of knots from the model object, the estimation is just a least squares method applied to the extended set of confounders defined as the local linear splines corresponding to the set of knots. The regression model is 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 three arguments, but two of them are 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`: As for the `knots` argument of `cslseModel`, 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 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 one knot per variable.

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

```
treated
*******

Covariates with no knots:
    married

Covariates with knots:
age :
      50%
Knots  23

nontreated
**********

Covariates with no knots:
    married

Covariates with knots:
age :
      50%
Knots  23
```

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

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


factor(treat)0  factor(treat)1        Xf0age_1        Xf0age_2      Xf0married
       4558.28         3754.98           27.80          -12.51         -115.82
      Xf1age_1        Xf1age_2      Xf1married
         89.25           22.22         1435.28
```

The object of class `slseFit` returned by `estSLSE` has its own print method that returns the coefficient estimates. A more detailed presentation of the results can be obtained using the `summary` method. The following is an example with the previous model.

```
summary(fit)
```

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

               Estimate Std. Error t value Pr(>|t|)
factor(treat)0  4558.28    2739.40   1.664   0.0961 .
factor(treat)1  3754.98    3704.37   1.014   0.3107
Xf0age_1          27.80     136.61   0.203   0.8387
Xf0age_2         -12.51      56.06  -0.223   0.8234
Xf0married      -115.82     795.35  -0.146   0.8842
Xf1age_1          89.25     185.53   0.481   0.6305
Xf1age_2          22.22      82.52   0.269   0.7877
Xf1married      1435.28    1226.68   1.170   0.2420
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Multiple R-squared:  0.009618,  Adjusted R-squared:  -9.119e-05
```

For example, the coefficient of `Xf0age_1` is the effect of age for the control on `re78` when `age`$\leq$ 23 and `Xf0age_2` is the effect when `age`$>$ 23. Note that the R^2 and adjusted R^2 are different from what we obtain using the summary of the `lm` object:

```
summary(fit$lm.out)[c("r.squared","adj.r.squared")]
```

```
$r.squared
[1] 0.4379272

$adj.r.squared
[1] 0.4316295
```

This is because our model does not explicitly contain an intercept and the R^2 is computed differently for models without an intercept. But, assuming that our model does not have an intercept is wrong since it includes both Z and $(1 - Z)$.

2.4 The `predict` and `plot` methods

The `predict` method is very similar to the `predict.lm` method. 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 (Zeileis (2006)). 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 only 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
```

The `predict` method is called by the `plot` method to visually assess the predicted outcome for the treated and nontreated with respect to a given confounder, controlling for the other variables in the model. The arguments of the `plot` method are:

- **x**: An object of class `slseFit`.
- **y**: An alias for `which` for compatibility with the generic `plot` function.
- **which**: confounder to plot against the outcome variable. It could 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 covariates 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.
- **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` 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".
- **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.
- **graphPar**: A list of graphical parameters if not satisfied with the default ones.
- `...`: Other arguments are passed to the `vcov.` function. For example, it is possible to change the type of `vcovHC` from the default HC3 to any available methods included in the `sandwich` package (Zeileis (2006)).

The default set of graphical parameters can be obtained by running the function `causalSLSE:::.initPar()`. 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:::.initPar()$treated$points
```

```
$pch
[1] 21

$col
[1] 2
```

The element `common` are for parameters not specific to a group like the main title or the axis labels and `legend` are the parameters that control the legend. 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:::.initPar()`. 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.

2.4.1 Plot examples

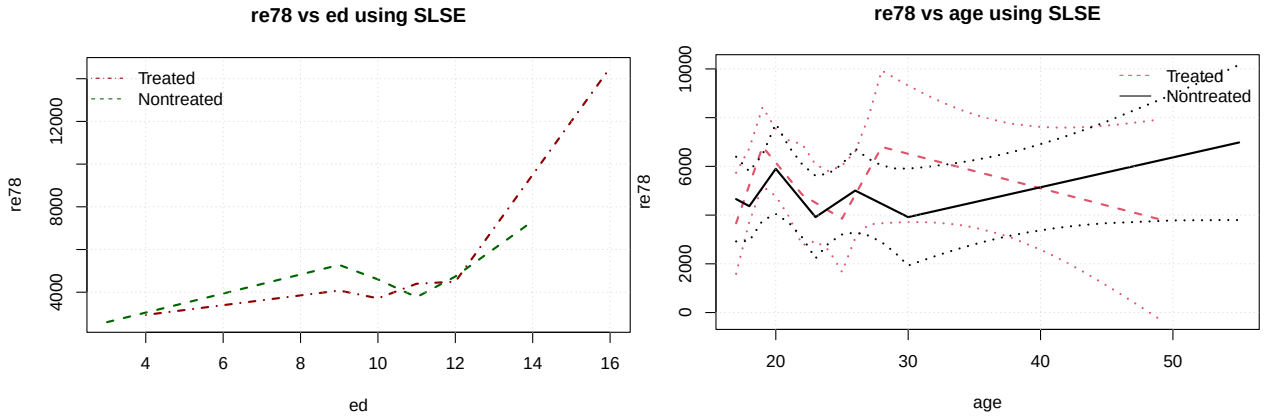
Example 1:

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.

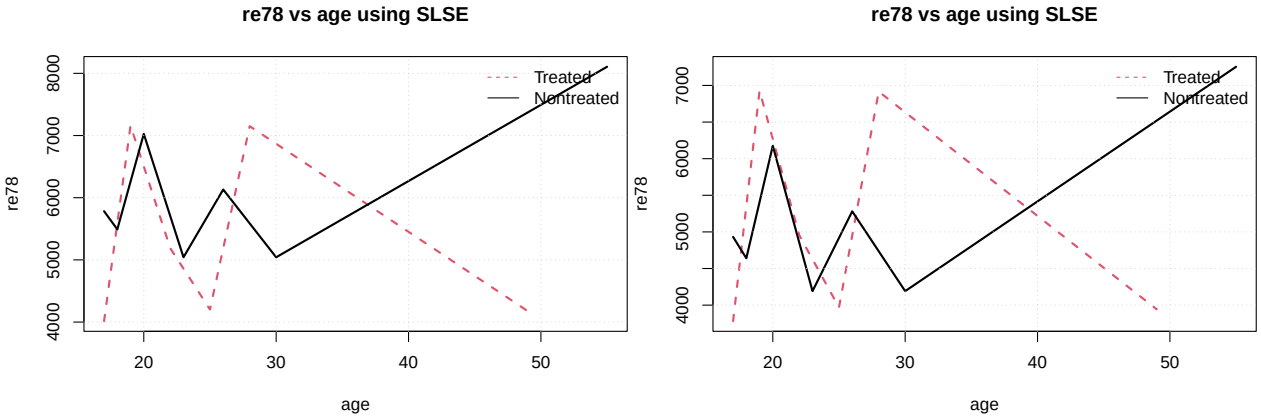
```
library(sandwich)
arg1 <- list(treated = list(lines = list(col = "darkred", lty = 4)),
             nontreated = list(lines = list(col = "darkgreen", lty = 2)),
             legend = list(x = "topleft"))
plot(fit1, "ed", vcov. = vcov, graphPar=arg1)
plot(fit1, "age", interval = 'confidence', level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



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 20% group empirical quantiles, we can set the argument to `function(x) quantile(x, .20)`. The following illustrates the two cases:

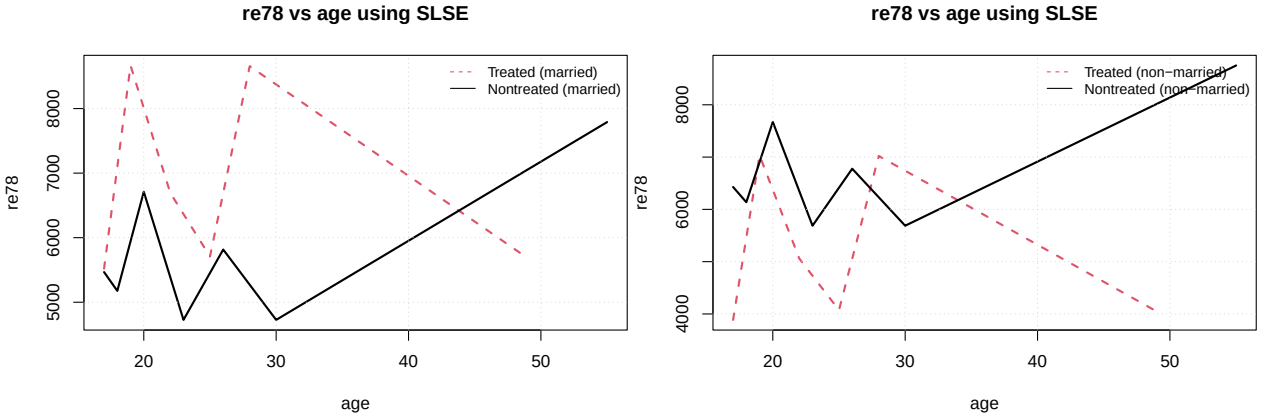
```
plot(fit1, "age", FUN = median)
plot(fit1, "age", FUN = function(x) quantile(x, 0.20))
```



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`.

```
arg2 <- list(legend = list(cex = 0.8))
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)
```

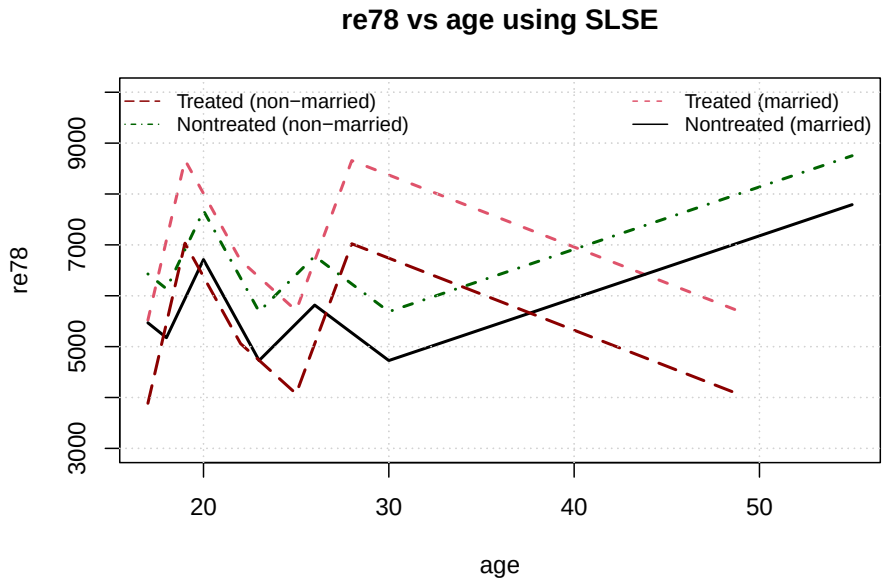


Example 4:

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 = 0.8),
             common = list(ylim = c(3000, 10000)))
plot(fit1, "age", fixedCov = list(married = 1, re75 = 10000),
```

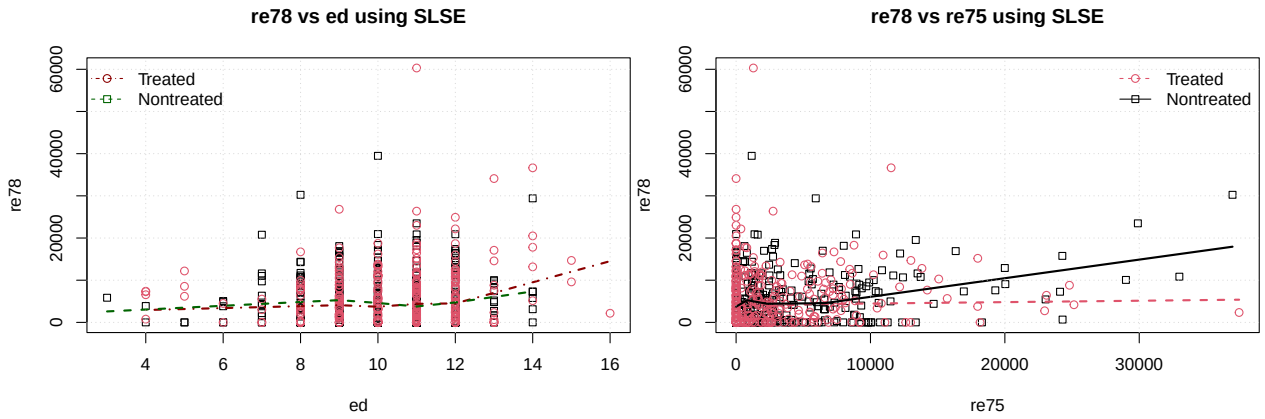
```
     addToLegend = "married", graphPar = arg3)
arg4 <- list(treated = list(lines = list(col = "darkred", lty = 5)),
             nontreated = list(lines = list(col = "darkgreen", lty = 4)),
             legend = list(x = "topleft", cex = 0.8))
plot(fit1, "age", fixedCov = list(married = 0, re75 = 10000),
     addToLegend = "non-married", add = TRUE, graphPar = arg4)
```



Example 5:

Finally, it is also possible to add the observed points 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)
```



2.4.2 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:
    treated: X1
    nontreated: X1
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:
    treated: X1
    nontreated: X1
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.

2.5 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). The two selection methods can be summarized as follows:

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.
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 descending order. Then, going from the largest to the smallest, we remove the knot associated with the p-value one by one, estimate the model and return the information criterion. We keep the model with the smallest information citerion.

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 is 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 $\{1, 2\}$, $\{1, 2, 3\}$, $\{2, 3, 4\}$, $\ldots$, $\{p-2, p-1, p\}$ and $\{p-1, p\}$ respectively.
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 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 keep the model with the smallest information citerion.

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

- **model**: An object of class `cslseModel`.
- **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 fix threshold. For example, `function(p) 0.20` sets the threshold to 0.2. This argument affects the result only when `selCrit` is set to "PVT".
- **vcov.**: An optional function to compute the least squares standard errors. By default, the p-values are computed using the `vcovHC` method from the `sandwich` package with `type="HC3"` (Zeileis (2006)).
- `...`: This is used to pass arguments to the `vcov.` function.

The function returns a model of class `cslseModel` with the optimal selection of knots. For example, we can compare the starting knots of `model1`, with the model selected by the default arguments.

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

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

treated
*******

Covariates with no knots:
    married

Covariates with knots:
age :
      20% 40% 60% 80%
Knots  19  22  25  28

re75 :
        40%  60%  80%
Knots 357.9 1962 5589

ed :
      20% 40% 60% 80%
Knots   9  10  11  12

nontreated
**********

Covariates with no knots:
    married

Covariates with knots:
age :
      16.67% 33.33% 50% 66.67% 83.33%
Knots     18     20  23     26     30

re75 :
        50% 66.67% 83.33%
Knots 823.3   2292   6567

ed :
      16.67% 33.33% 66.67% 83.33%
Knots      9     10     11     12
```

```
model2 <- selSLSE(model1)
print(model2,  which = "selKnots")
```

```
Causal Semiparametric LSE Model: Selected knots
***********************************************
Selection method: BLSE
Criterion: AIC

treated
*******

Covariates with no knots:
    re75, married

Covariates with knots:
age :
      20% 60% 80%
Knots  19  25  28

ed :
      80%
Knots  12
```

```
nontreated
**********

Covariates with no knots:
    married

Covariates with knots:
age :
      33.33% 50% 66.67% 83.33%
Knots     20  23     26     30

re75 :
        50% 83.33%
Knots 823.3   6567

ed :
      16.67% 66.67%
Knots      9     11
```

For example, the BLSE-AIC method has removed 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 :
             20%     40%     60%     80%
Knots   19.00000 22.0000 25.0000 28.0000
P-Value  0.03472  0.7597  0.3167  0.2257

re75 :
             40%       60%       80%
Knots   357.9499 1961.8640 5588.6640
P-Value   0.8648    0.7035    0.8177

ed :
           20%     40%     60%     80%
Knots   9.0000 10.0000 11.0000 12.0000
P-Value 0.6366  0.5837  0.8017  0.3286

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

Covariates with knots:
age :
         16.67%  33.33%     50%  66.67%  83.33%
Knots   18.0000 20.0000 23.0000 26.0000 30.0000
P-Value  0.4466  0.1067  0.1567  0.3234  0.2884

re75 :
             50%    66.67%    83.33%
Knots   823.2544 2292.1710 6567.3290
P-Value   0.3261    0.6295    0.3341

ed :
        16.67%  33.33%  66.67%  83.33%
Knots   9.0000 10.0000 11.0000 12.0000
P-Value 0.2813  0.9158  0.1879  0.8806
```

In the following example, we see 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
Criterion: BIC

Confounders approximated by SLSE:
    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"))
```

```
Semiparametric LSE
******************
Selection method: FLSE
Criterion: BIC


factor(treat)0  factor(treat)1          Xf0age         Xf0re75          Xf0ed
     4.826e+03      -3.890e+02      -2.011e+01       2.982e-01       2.500e+00
    Xf0married          Xf1age         Xf1re75          Xf1ed      Xf1married
    -1.094e+03       4.105e+01       2.676e-02       4.849e+02       1.417e+03
```

2.6 The `causalSLSE` method for `slseFit` objects

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

- **object**: An object of class `slseFit`.
- **causal**: What causality 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 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)
causalSLSE(fit1)
```

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


ACE = 814.3
ACT = 831.9
ACN = 802
```

We see that the selection method used to select the knots are set to `Default` because the knots were first selected by the default method and no additional selection method was used. The method returns an object of class `cslse` and its `print` method only prints the causal effect estimates. For more details about the estimation, which includes standard errors and significance tests, we can use the `summary` method:

```
ce <- causalSLSE(fit1)
sce <- summary(ce)
sce
```

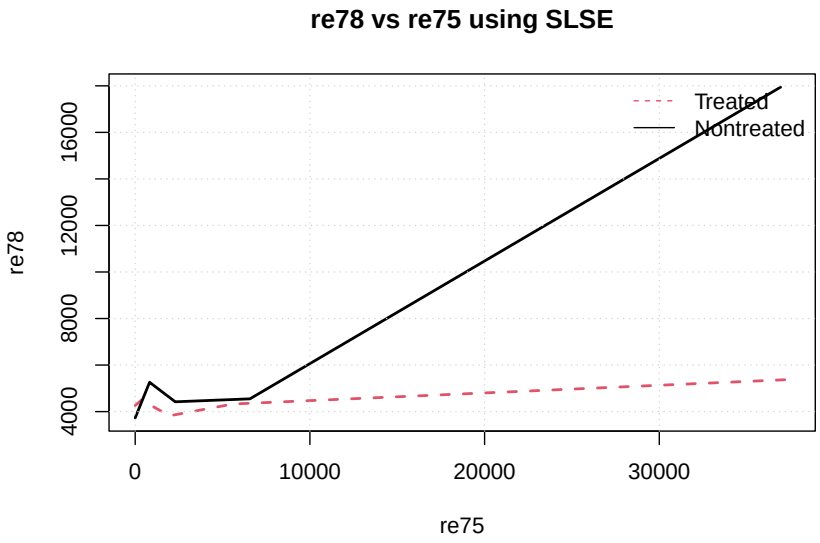
```
Causal Effect using Semiparametric LSE
**************************************
Selection method: Default

    Estimate Std. Error t value Pr(>|t|)
ACE    814.3      506.4   1.608    0.108
ACT    831.9      526.8   1.579    0.114
ACN    802.0      514.8   1.558    0.119
```

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 `$causal`, the least squares coefficients table using `$beta` or the list of knots using `$knots`.

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

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



2.6.1 The `extract` method

The package comes with an `extract` method for objects of class `cslse`, which is a required method for creating Latex tables using the `texreg` package (Leifeld (2013)). 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(SLSE=c1, BLSE=c2, FLSE=c3), table=FALSE, digits=4)
```

	SLSE	BLSE	FLSE
ACE	814.3083	818.1598	824.4901
	(506.3839)	(497.7885)	(496.4032)
ACT	831.8856	837.0768	852.4659
	(526.8465)	(519.2572)	(513.7079)
ACN	802.0249	804.9401	804.9401
	(514.7871)	(503.3558)	(502.7779)
Num. knots (Nontreated)	12	8	6
Num. knots (Treated)	11	4	4
Num. confounders	4	4	4
Num. obs. (Nontreated)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.0869	0.0852	0.0840
Adj. R^2	0.0445	0.0577	0.0592

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

Note that we refer to SLSE when no optimal selection method is used, which includes the `Default` and `User Based` methods. The option `table=FALSE`, from the `texreg` package, is used to remove the Latex floating table environment. With this option, the table appears right after the code instead of being placed somewhere else by Latex. 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`.
- **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,
       which="ACE-ACT", include.adjrs=FALSE)
```

	SLSE	BLSE	FLSE
ACE	814.31	818.16	824.49
	(506.38)	(497.79)	(496.40)
ACT	831.89	837.08	852.47
	(526.85)	(519.26)	(513.71)
Num. knots (Nontreated)	12	8	6
Num. knots (Treated)	11	4	4
Num. confounders	4	4	4
Num. obs. (Nontreated)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.09	0.09	0.08

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

2.7 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 the method for `slseFit` objects, plus the necessary arguments for the knots selection. The following are the arguments not already defined for objects of class `slseFit`. The details of these arguments are presented in Section 2.5.

- **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	814.3083	818.1598	824.4901
	(506.3839)	(497.7885)	(496.4032)
ACT	831.8856	837.0768	852.4659
	(526.8465)	(519.2572)	(513.7079)
ACN	802.0249	804.9401	804.9401
	(514.7871)	(503.3558)	(502.7779)
Num. knots (Nontreated)	12	8	6
Num. knots (Treated)	11	4	4
Num. confounders	4	4	4
Num. obs. (Nontreated)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.0869	0.0852	0.0840
Adj. R^2	0.0445	0.0577	0.0592

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

2.8 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	814.3083	818.1598	824.4901
	(506.3839)	(497.7885)	(496.4032)
ACT	831.8856	837.0768	852.4659
	(526.8465)	(519.2572)	(513.7079)
ACN	802.0249	804.9401	804.9401
	(514.7871)	(503.3558)	(502.7779)
Num. knots (Nontreated)	12	8	6
Num. knots (Treated)	11	4	4
Num. confounders	4	4	4
Num. obs. (Nontreated)	425	425	425
Num. obs. (Treated)	297	297	297
R^2	0.0869	0.0852	0.0840
Adj. R^2	0.0445	0.0577	0.0592

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

Note that this method calls `cslseModel`, `selSLSE`, `estSLSE` and the method `causalSLSE` for `slseFit` 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 standard errors.

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 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)
```

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	-1.4396^{***}	-1.4530^{***}	-1.4533^{***}
	(0.2770)	(0.2742)	(0.2700)
ACT	-1.9316^{***}	-1.9316^{***}	-1.9320^{***}
	(0.3180)	(0.3176)	(0.3131)
ACN	-0.0865	-0.1369	-0.1369
	(0.3338)	(0.3271)	(0.3254)
Num. knots (Nontreated)	2	2	1
Num. knots (Treated)	4	0	0
Num. confounders	1	1	1
Num. obs. (Nontreated)	80	80	80
Num. obs. (Treated)	220	220	220
R^2	0.7434	0.7386	0.7303
Adj. R^2	0.7354	0.7342	0.7266

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

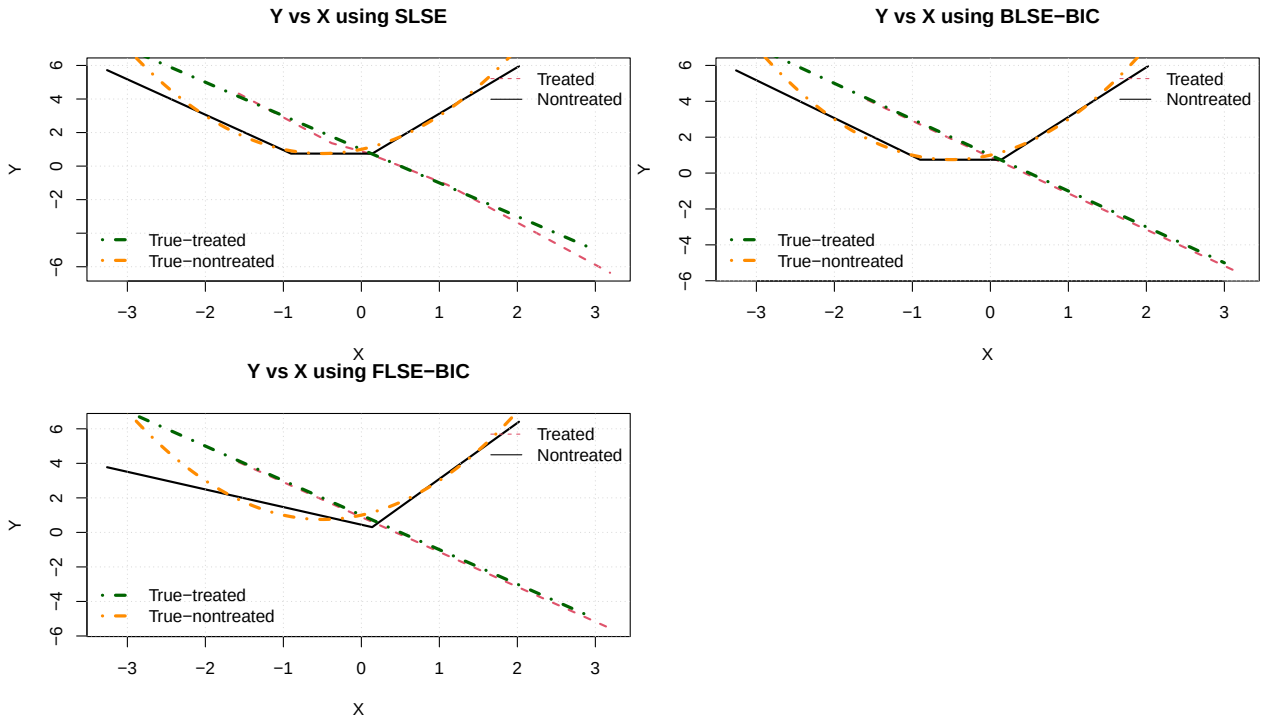
We see that both selection methods choose to assign 0 knots for the treated group, which is not surprising since the true $f_1(x)$ is linear. We can compare the different fits.

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

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

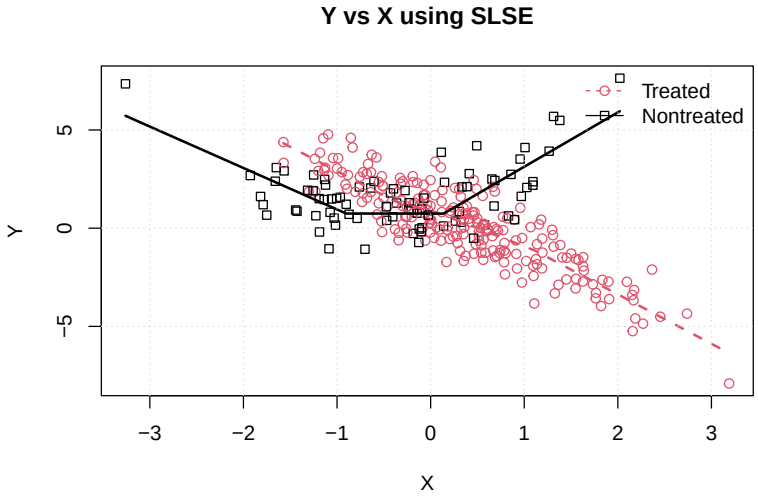
```
plot(c1, "X")
curve(1 - 2 * x, -3, 3, col = "darkgreen", lty = 4, lwd = 3, add = TRUE)
curve(1 + x + x^2, -3, 3, col = "darkorange", lty = 4, lwd = 3, add = TRUE)
legend("bottomleft", c("True-treated", "True-nontreated"),
       col=c("darkgreen", "darkorange"), lty = 4, lwd = 3, bty = 'n')
plot(c2, "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 $f_1(x)$ and $f_2(x)$ for SLSE and BLSE. However, the FLSE based on BIC does not do a good job. 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 $I(A)$ 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)
```

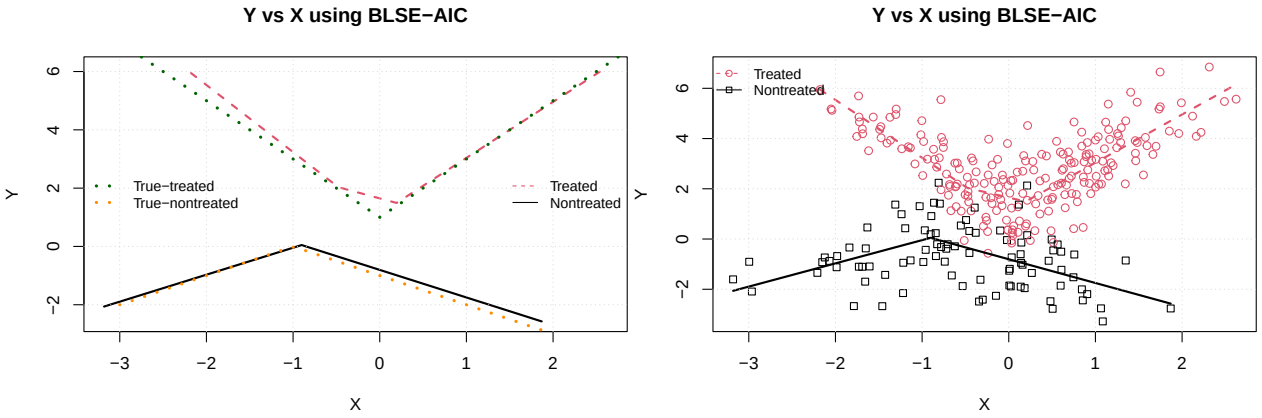
```
c1 <- causalSLSE(mod, selType = "SLSE")
c2 <- causalSLSE(mod, selType = "BLSE", selCrit = "BIC")
c3 <- causalSLSE(mod, selType = "BLSE", selCrit = "AIC")
texreg(list(SLSE = c1, BLSE.BIC = c2, BLSE.AIC = c3), table = FALSE, digits = 4)
```

	SLSE	BLSE.BIC	BLSE.AIC
ACE	3.9290***	3.9201***	3.9201***
	(0.1772)	(0.1756)	(0.1756)
ACT	3.9552***	3.9404***	3.9404***
	(0.1966)	(0.1938)	(0.1938)
ACN	3.8670***	3.8721***	3.8721***
	(0.2352)	(0.2337)	(0.2337)
Num. knots (Nontreated)	2	1	1
Num. knots (Treated)	3	2	2
Num. confounders	1	1	1
Num. obs. (Nontreated)	89	89	89
Num. obs. (Treated)	211	211	211
R^2	0.7833	0.7829	0.7829
Adj. R^2	0.7774	0.7784	0.7784

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

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:

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

where 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)
```

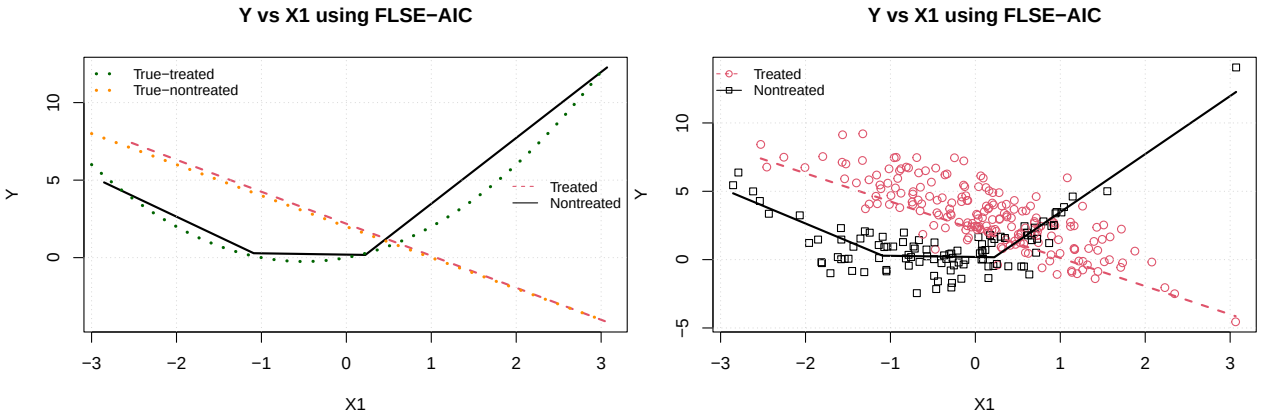
```
c1 <- causalSLSE(mod, selType = "SLSE")
c2 <- causalSLSE(mod, selType = "FLSE", selCrit = "BIC")
c3 <- causalSLSE(mod, selType = "FLSE", selCrit = "AIC")
texreg(list(SLSE = c1, FLSE.BIC = c2, FLSE.AIC = c3), table = FALSE, digits = 4)
```

	SLSE	FLSE.BIC	FLSE.AIC
ACE	2.4699***	2.4866***	2.4725***
	(0.3021)	(0.2997)	(0.2998)
ACT	2.0653***	2.0688***	2.0688***
	(0.3772)	(0.3731)	(0.3731)
ACN	3.2323***	3.2739***	3.2334***
	(0.3540)	(0.3523)	(0.3529)
Num. knots (Nontreated)	6	5	5
Num. knots (Treated)	6	3	4
Num. confounders	2	2	2
Num. obs. (Nontreated)	104	104	104
Num. obs. (Treated)	196	196	196
R^2	0.8630	0.8614	0.8625
Adj. R^2	0.8547	0.8551	0.8558

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

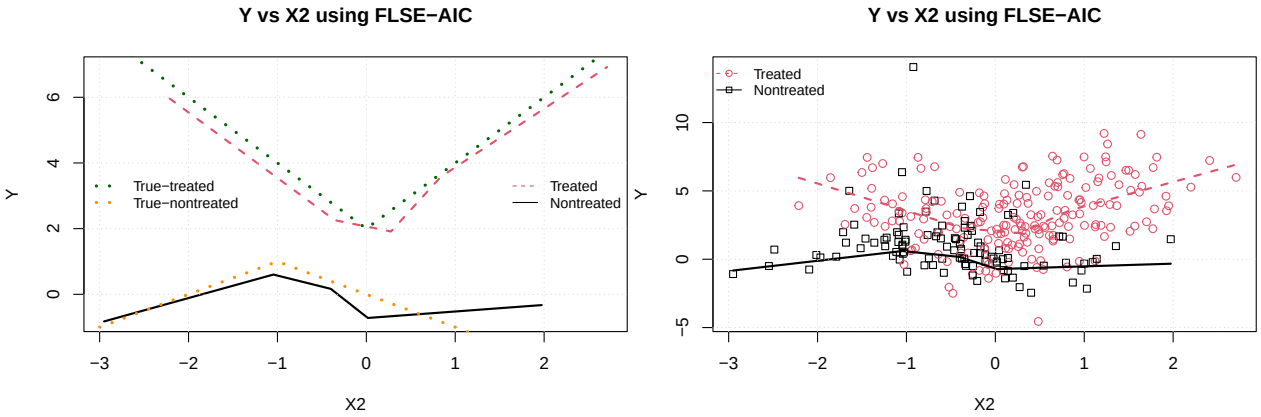
To illustrate the method, since we have two confounders, we need to plot the outcome against one confounder holding the other fixed. The default is to fix it to its sample mean. For the true curve, we fix it to its population mean, which is 0. We first look at the outcome against X_1. By fixing X_2 to 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 interactions

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

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

whereX_1, X_2, e and u 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)
```

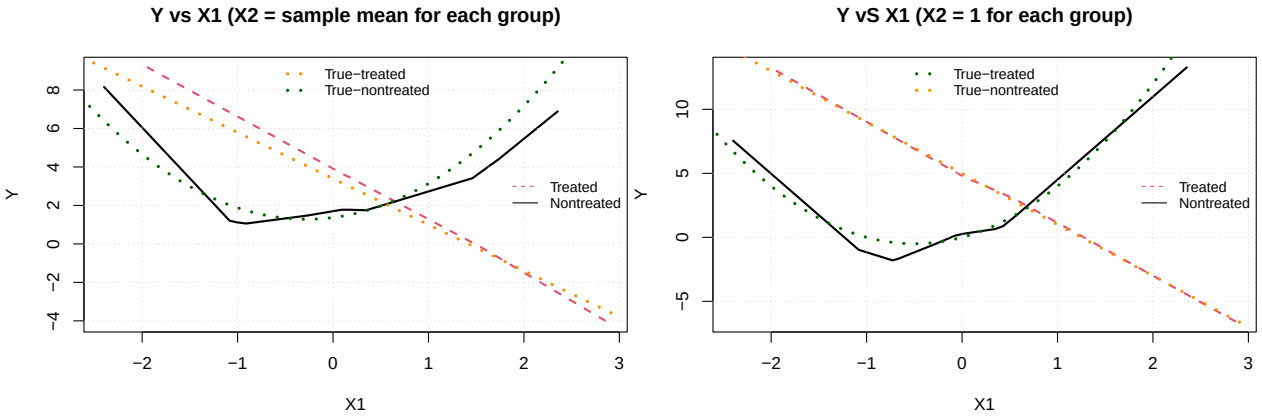
```
c1 <- causalSLSE(mod, selType = "SLSE")
c2 <- causalSLSE(mod, selType = "FLSE", selCrit = "BIC")
c3 <- causalSLSE(mod, selType = "FLSE", selCrit = "AIC")
texreg(list(SLSE = c1, FLSE.BIC = c2, FLSE.AIC = c3), table = FALSE, digits = 4)
```

	SLSE	FLSE.BIC	FLSE.AIC
ACE	1.7990***	1.7797***	1.7744***
	(0.3889)	(0.3962)	(0.3940)
ACT	1.2582*	1.2091*	1.2091*
	(0.5029)	(0.5143)	(0.5143)
ACN	2.8183***	2.8550***	2.8399***
	(0.4524)	(0.4535)	(0.4475)
Num. knots (Nontreated)	9	8	8
Num. knots (Treated)	9	5	6
Num. confounders	3	3	3
Num. obs. (Nontreated)	104	104	104
Num. obs. (Treated)	196	196	196
R^2	0.8909	0.8879	0.8894
Adj. R^2	0.8809	0.8799	0.8811

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

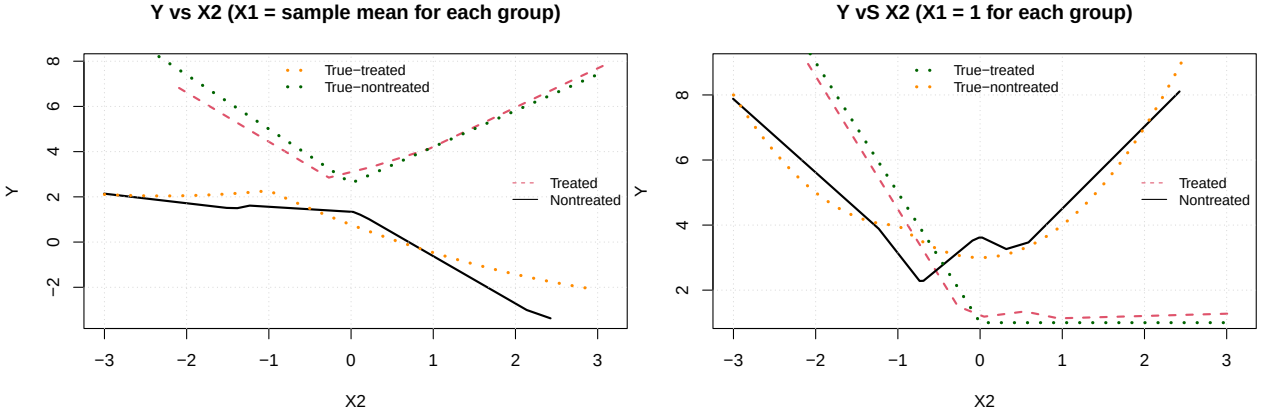
In the case of multiple confounders with interactions, the shape of the fitted outcome with respect to one confounder depends on the value of the other confounders. Without interaction, changing the value of the other confounders only shifts the fitted line without changing its shape. The following graphs 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 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 mode 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 method is `print`. The method `update`, which is used by `estSLSE` to select knots before estimating the model is not exported.
- **cslseKnots**: The object is created by the function `cslseKnots` and it is a list of `slseKnots` objects. As for `slseKnots` object, the only exported registered method is `print` and there is an non-exported method `update`.
- **cslseModel**: The object is created by the function `cslseModel` and the exported registered methods are `print`, `estSLSE` (estimate the regression model), `selSLSE` (optimal selection of knots) and `causalSLSE` (to compute 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**: The object is created by the method `estSLSE` 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)
- **summary.slseFit**: The object is created by the `summary` method for `slseFit` objects. The only exported registered method is `print`.
- **cslse**: The object is created by any `causalSLSE` method. It inherits from `slseFit` object. 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`)

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

References

Giurcanu, M., M. Capanu, P. Chaussé, and G. Luta. 2023. "Semiparametric Thresholding Least Squares 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.